Un système tutoriel intelligent et adaptatif pour l'apprentissage de compétences en environnement virtuel de formation

Cédric Buche

Université de Bretagne Occidentale

— Mémoire de thèse —

Spécialité : Informatique

Un système tutoriel intelligent et adaptatif pour l'apprentissage de compétences en environnement virtuel de formation

Cédric Buche

Soutenue le 14 novembre 2005 devant la commission d'examen :

Rapporteurs		
Bruno	Arnaldi	Professeur à l'INSA de Rennes
Pierre	Tchounikine	Professeur à l'Université du Maine
Examinateurs		
Pierre	De Loor	Maître de conférences à l'ENIB
Jean-Yves	Guinard	Maître de conférences à l'UBO
Lionel	Marcé	Professeur à l'UBO
Jacques	Tisseau	Professeur à l'ENIB
Invité		
Ronan	Querrec	Maître de conférences à l'ENIB

Un système tutoriel intelligent et adaptatif pour l'apprentissage de compétences en environnement virtuel de formation

Mémoire de thèse

Cédric Buche

— Novembre 2005 —

Laboratoire d'Informatique des SYstèmes Complexes EA 3883 UBO-ENIB

CÉDRIC BUCHE
e-mail : buche@enib.fr
url : http ://www.cerv.fr/~buche
tel : +33 (0)2 98 05 89 56

École Nationale d'Ingénieurs de Brest (ENIB)

Technopôle Brest-Iroise
Parvis Blaise Pascal
C.S. 73862 F-29208 Brest Cedex
(http ://www.enib.fr)

Centre Européen de Réalité Virtuelle (CERV)

Technopôle Brest-Iroise
25, rue Claude Chappe
BP 38 F-29280 Plouzané
(http ://www.cerv.fr)

Remerciements

Je suis heureux d'avoir l'occasion de présenter ma reconnaissance à tous ceux qui m'ont soutenu dans la réalisation de cette thèse.

Je tiens à remercier Jacques Tisseau pour la confiance qu'il m'a témoigné en m'accueillant dans son laboratoire et en acceptant la direction de cette thèse. Je tiens également à remercier Pierre De Loor et Ronan Querrec pour leur disponibilité et pour la qualité de leur encadrement.

Je remercie Bruno Arnaldi et Pierre Tchounikine pour avoir accepté de rapporter cette thèse. J'adresse également mes remerciements à Jean-Yves Guinard et à Lionel Marcé pour avoir participé à mon jury.

Je souhaite remercier l'ensemble des personnes du Centre Européen de Réalité Virtuelle pour leur sympathie.

Ces travaux n'auraient pu aboutir sans les personnes impliquées dans les équipes Simulation Participative et Immersive (SPI) et Attitudes, Stratégies, Apprentissages et Performances (ASAP). Je remercie plus particulièrement Pierre Chevaillier et Gilles Kermarrec.

Enfin, je tiens à remercier ma famille et mes amis qui m'ont soutenu tout au long de ces années de thèse. Merci à Bidou, Cykril, Damoizo, Gabouze, Gireg, Grace, Gwen, Mauï, Patoche, Perso, Pit, Poirot, Rhum, Rauchefordisse, Vaness et tous les autres !

Avant-propos

Mon projet de recherche s'inscrit dans le cadre du Centre Européen de Réalité Virtuelle de Brest (CERV), au sein du Laboratoire Informatique des SYstèmes Complexes (LISYC) regroupant des chercheurs de l'Université de Bretagne Occidentale (UBO) et de l'École Nationale d'Ingénieurs de Brest (ENIB). Plus précisément, les travaux se positionnent dans l'équipe Simulation Participative et Immersive (SPI) qui a pour objectif l'étude de la réalité virtuelle et des systèmes multi-agents dans lesquels des acteurs humains et des agents artificiels collaborent pour la réalisation d'une tâche.

Cette thèse s'intègre dans le projet MASCARET (Multi Agent System for Collaborative and Adaptive Realistic Environment for Training) comme l'utilisation des systèmes multi-agents pour simuler les environnements réalistes, collaboratifs et adaptatifs pour l'entraînement. Il s'agit d'un environnement virtuel qui permet la formation au travail procédural et collaboratif.

Table des matières

Liste des figures

Liste des tableaux

Chapitre 1

Introduction

1.1 Problématique

De nombreux domaines de formation, tels que la conduite automobile ou la formation des pompiers professionnels, nécessitent la mise en situation des apprenants ; ceux-ci doivent acquérir non seulement des connaissances, mais encore de véritables compétences. La compétence se caractérise par un pouvoir d'agir, permettant d'être efficace dans un ensemble de situations, au sein d'un domaine de référence. Pour devenir progressivement efficace en situation, l'apprenant doit apprendre par l'action (Nguyen-Xuan, 1995). Or, la mise en action des apprenants peut s'avérer coûteuse (d'un point de vue matériel) ou risquée (d'un point de vue humain). C'est le cas lorsqu'il s'agit d'apprendre à agir et à réagir face à des accidents (non respect des règles habituelles de l'environnement routier par un des conducteurs), des évènements peu prévisibles (un enfant traverse la route de façon inopinée), des dysfonctionnements (panne matérielle ou effondrement psychologique d'une personne lors d'une intervention risquée). Cette compétence consistant à résoudre des problèmes en situation dynamique (incertaine, évolutive, et à forte contrainte temporelle) est particulièrement difficile à aborder par une formation classique : étude de cas, proposition de règles générales, instructions relatives à des scénarios probables, *etc.* Au contraire, la simulation informatique permet d'immerger les apprenants dans les environnements où ils peuvent essayer, choisir, prendre des initiatives, échouer et recommencer. La confrontation aux situations permet d'élaborer en mémoire à long terme[1] des schémas d'action[2] articulant divers composants de la compétence : des connaissances générales liées au domaine, des stratégies ou des règles contextualisées, des procédures cognitives et des savoir-faire. Les environnements informatiques de formation utilisant la technologie de la *réalité virtuelle* (RV) sont particulièrement appropriés dans la mesure où ils proposent des situations à la fois réalistes, complexes et incertaines. L'ensemble de ces environnements virtuels peut être

[1] En psychologie cognitive, la mémoire à long terme représente la mémoire au sens courant. Elle permet le codage et le stockage durable des informations de manière organisée. Elle nous permet par exemple de nous rappeler d'événements passés, de connaissances générales ou de compétences motrices.

[2] Un schéma d'action est la forme sous laquelle la connaissance se rapportant à l'action est stockée (Richard, 1986).

regroupé sous l'acronyme de EVF (Environnement Virtuel de Formation).

La recherche dans le domaine des EVF s'est souvent focalisée sur les aspects techniques, comme dans Gvt (Cazeaux et al., 2005) et Cs-Wave (Steib et al., 2005), sans pour autant considérer l'intégration de concepts éducatifs intrinsèques au système. Bien que l'introduction de méthodes d'enseignement a pu être étudiée, notamment aux travers de travaux portant sur des méthodologies de conception (Lourdeaux, 2001; Mellet d'Huart, 2004), l'intégration de principes pédagogiques pose toujours problème. En effet, la plupart des EVF se contente de simuler l'environnement (Levesque, 2003; Popovici et al., 2004) et ne dispose pas de connaissances explicites utilisables pédagogiquement. D'autres n'incorporent qu'une représentation des connaissances sur le domaine et/ou l'apprenant, comme Steve (Rickel et Johnson, 1999). Les plus évoluées proposent un modèle de diagnostic, comme Hal (Lourdeaux, 2001), néanmoins les assistances proposées sont spécifiées pour chaque exercice, et donc non réutilisables. Le modèle pédagogique doit être reprécisé pour chaque domaine ou chaque exercice.

Dans ce contexte, un certain nombre de questions restent ouvertes :
- ▷ Comment introduire de la pédagogie dans un EVF ?
- ▷ Comment réaliser un lien cohérent entre les objets informatiques et les ressources pédagogiques ?
- ▷ Jusqu'à quel point cette intégration est dépendante de l'environnement simulé et du domaine étudié ?
- ▷ Est-il possible de définir des assistances pédagogiques génériques, associées à un guide de conception permettant son adaptation automatique à un domaine particulier ?
- ▷ Comment un tel système peut-il s'adapter « en ligne » à l'apprenant pour améliorer l'apprentissage ?

Cette thèse aborde l'ensemble de ces questions par le biais d'une proposition.

1.2 Proposition

Cette thèse est une contribution à la réalisation d'environnements virtuels de formation visant l'acquisition de compétences par l'expérimentation. Nous nous plaçons dans le cadre de l'apprentissage de procédures collaboratives, où les apprenants sont formés à la prise de décision et non au geste technique[3].

Plus précisément, nos travaux portent sur des modèles qui permettent la réalisation de logiciels pour assister les apprenants et les formateurs. Cet accompagnement des acteurs de la formation est réalisé par l'utilisation d'un *système tutoriel intelligent* (ITS : Intelligent Tutoring System). Complémentaire de la réalité virtuelle pour l'apprentissage de compétences, il permet l'adaptation de la situation d'apprentissage en fonction des activités de l'apprenant. Notre proposition consiste alors en un modèle d'ITS « générique » qui simule un raisonnement pédagogique au sein d'un EVF. En effet, le modèle pédagogique est défini et utilisable indépendamment de l'environnement simulé et de l'exercice à effectuer. Plus précisément,

[3] 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 (Querrec, 2002).

bénéficiant d'un modèle de description de l'environnement, le raisonnement pédagogique manipule des connaissances génériques portant sur l'apprenant et sur le travail à réaliser.

La figure 1.1 présente schématiquement le comportement de notre ITS dans l'EVF. L'ITS propose des assistances pédagogiques au formateur en considérant un ensemble d'informations de l'environnement virtuel utilisé pour prendre une décision pédagogique. Ces informations proviennent de l'analyse de l'environnement et des activités de l'apprenant. Le formateur, bénéficiant des informations et des suggestions de l'ITS, choisit l'assistance pédagogique la plus appropriée. Le comportement qui propose des assistances pédagogiques est initialement défini par des pédagogues. Le mécanisme amenant aux propositions s'organisent au fur et à mesure des expériences (processus d'auto-modification) afin de s'adapter en fonction des choix du formateur. L'ITS adaptatif réalise alors l'adéquation entre l'apprenant et le formateur en se basant sur la spécification du pédagogue.

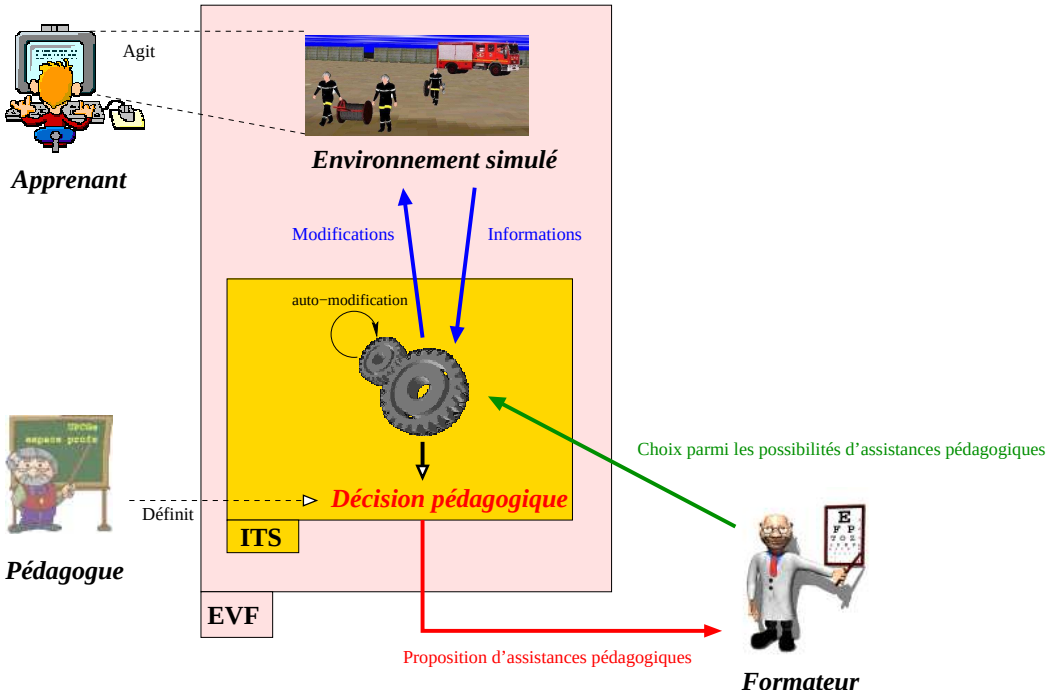


FIGURE 1.1 – Représentation simplifiée du système.

Notre modèle d'ITS est un système multi-agents, qui s'intègre dans un environnement virtuel de formation informé et qui défini un agent particulier : l'agent pédagogique (*cf.* figure 1.2).

1. **L'environnement virtuel de formation informé** est utilisé pour :

   (a) *Représenter et simuler l'environnement*

   Simulation

   Il s'agit de construire l'environnement de l'apprenant. Pour cela, nous nous appuyons sur les techniques de réalité virtuelle. Chaque apprenant est immergé dans un environnement virtuel qui simule son environnement social et rend compte de son environnement physique. Pour représenter l'environnement social, nous nous basons sur les modèles organisationnels de travail collaboratif proposés par Querrec (2002). Pour représenter l'environnement physique, nous définissons les entités virtuelles possédant des comportements génériques, nous permettant d'ajouter facilement des comportements d'assistances pédagogiques à toute simulation existante. Nous définissons de telles entités en utilisant le modèle VEHA (Virtual Environment supporting Human Activity).

   (b) *Représenter les connaissances pour la pédagogie*

   Situation pédagogique

   Notre ITS doit proposer un modèle qui représente les éléments pertinents de l'environnement, en terme de pédagogie. Ces informations sont construites à partir de l'environnement et des caractéristiques de l'apprenant. Il s'agit de réifier et de mettre à jour, en cours de simulation, les connaissances sur le domaine, l'apprenant et l'interface (interaction système-apprenant). Nous appelons ces connaissances la « situation pédagogique ». Elle sert de base de connaissances au raisonnement effectué par un agent pédagogique.

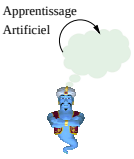
2. **L'agent pédagogique adaptatif** exploite les informations de cet environnement virtuel informé pour :

   (a) *Simuler un raisonnement pédagogique*

   Considérant les informations précédemment représentées, nous ajoutons un mécanisme qui simule une prise de décision pédagogique afin d'effectuer les propositions d'assistances pédagogiques au formateur, appropriées à la situation. La simulation du raisonnement pédagogique est réalisée par un « agent pédagogique », entité faisant partie de l'ITS. Son comportement initial est spécifié par un pédagogue de façon générique, *i.e.* indépendamment du domaine.

   (b) *S'adapter*

   Agent pédagogique

   L'agent pédagogique s'adapte au couple apprenant-formateur. Il modifie ses propositions dynamiquement en utilisant un mécanisme d'apprentissage artificiel.

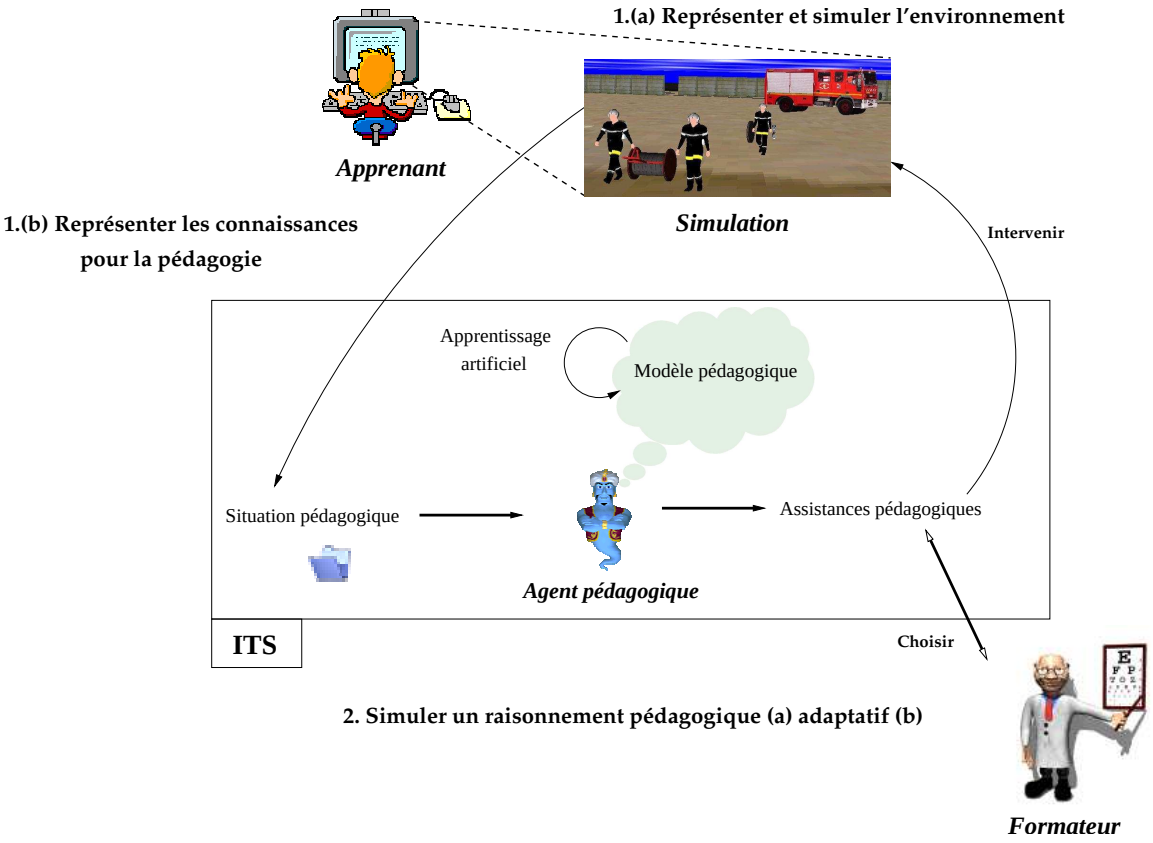


FIGURE 1.2 – Les différentes parties de l'ITS.

1.3 Mise en application

Pour montrer l'efficience de notre proposition, ces travaux sont appliqués dans le cadre d'un environnement virtuel de formation appelé Gaspar : Gestion de l'Activité aviation et des Sinistres sur Porte-avions par la Réalité virtuelle.

Dans cet environnement, le travail est collaboratif. La formation se déroule à la manière d'un jeu de rôles. Le formateur affecte les rôles aux apprenants. Les membres de l'équipe doivent collaborer pour remplir la mission. 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. La figure 1.3 montre un exemple de scène dans laquelle les apprenants et formateurs sont immergés.

Figure 1.3 – Exemple de scène dans l'application Gaspar.

La généricité de notre modèle est mise en avant par son intégration dans d'autres applications, notamment à travers l'environnement SécuRéVi[4] (Sécurité et Réalité Virtuelle (Querrec et al., 2004)).

[4] Cette application vise la formation des officiers sapeurs-pompiers à la gestion opérationnelle et au commandement.

1.4 Organisation de ce mémoire

Ce mémoire s'articule autour de trois parties : le contexte qui oriente nos travaux et qui présente les concepts constituant le fondement de notre approche (chapitre 2), le modèle que nous proposons (chapitres 3 et 4) et l'application qui l'utilise et le valide (chapitre 5).

Dans le **chapitre 2**, « Les environnements virtuels de formation », nous évaluons l'intégration des ITS au sein de la RV pour l'apprentissage de compétences. Nous présentons les ITS au travers des quatre modèles principaux et la RV comme un domaine à fort potentiel de transfert de compétences.

Dans le **chapitre 3**, « Environnement virtuel de formation informé », nous proposons un modèle d'ITS. Nous définissons l'implémentation, sous forme d'agents, de modèles fondamentaux des ITS pour Mascaret. Le système multi-agents construit la « situation pédagogique » qui est utilisée par le raisonnement d'un agent pédagogique.

Le **chapitre 4**, « Agent pédagogique », est consacré à la modélisation du comportement de prise de décision de notre agent pédagogique. Nous proposons un modèle simulant un processus pédagogique, basé sur un système de classeurs hiérarchiques apprenant.

Dans le **chapitre 5**, « Application : le projet Gaspar », nous montrons comment notre modèle d'ITS s'intègre dans plusieurs applications. Nous nous intéressons plus particulièrement à l'application Gaspar qui simule l'activité aviation sur un porte-avions.

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

Remarque complémentaire

Les différents acteurs de notre EVF sont représentés ci-dessous. Ces images seront utilisées tout au long de ce mémoire.

 ***Apprenant***	L'apprenant est la personne qui doit acquérir des compétences. On préférera le terme apprenant à celui de formé, qui évoque une vision passive de l'apprentissage.
 ***Formateur***	Le formateur est un éducateur dont de rôle est de favoriser l'apprentissage de compétences chez l'apprenant. Bien qu'il soit spécialiste du domaine enseigné, sa charge ne contient pas la définition des procédures à suivre.
 ***Pédagogue***	Le pédagogue est la personne attachée à l'éducation dans le processus de formation. Il définit les méthodes d'enseignement propres à une discipline.
 ***Expert du domaine***	L'expert du domaine est la personne qui définit les procédures à suivre dans un domaine particulier. Son rôle n'est pas de former des personnes au domaine concerné.
 ***Concepteur de l'EV***	Le concepteur de l'EV est l'informaticien qui met en place une simulation informatique.
 ***Concepteur de l'EVF***	Le concepteur de l'EVF est la personne qui ajoute à une simulation informatique des composantes éducatives, ou qui met en place la totalité de l'environnement de formation.
 ***Agent pédagogique***	L'agent pédagogique est une entité virtuelle qui simule un raisonnement pédagogique dans un EVF.
 ***Système***	Le système est un programme informatique qui intègre toutes les fonctionnalités ajoutées à la simulation pour que l'environnement devienne un environnement de formation.

Chapitre 2

Les environnements virtuels de formation

Tu me dis, j'oublie.
Tu m'enseignes, je me souviens.
Tu m'impliques, j'apprends.

Benjamin Franklin

Ce chapitre s'intéresse aux systèmes informatiques destinés à la formation. Nous proposons de montrer l'intérêt de la Réalité Virtuelle (RV) et des Systèmes Tutoriels Intelligents (ITS) pour l'apprentissage de compétences. Les ITS permettent de représenter, sous la forme de modèles, les informations sur un domaine d'apprentissage, l'apprenant, la communication entre l'apprenant et le système, et les méthodes d'enseignement. Ils analysent les activités de l'apprenant et proposent des assistances pédagogiques individualisées. La RV est définie selon trois axes : autonomie, interaction et immersion. L'autonomie permet la variabilité des contextes, ce qui constitue une condition nécessaire à la construction de compétences transférables. Nous évaluons ensuite l'intégration des ITS dans les Environnements Virtuels de Formation (EVF) existants. Nous montrons que cette intégration est actuellement limitée.

Ce chapitre positionne la simulation informatique, et plus particulièrement la réalité virtuelle, comme situation de formation. Si on représente, comme le propose Houssaye (1988), la situation d'apprentissage selon trois pôles : l'apprenant, la compétence à acquérir (objet du savoir) et le formateur, la simulation informatique fournit un environnement commun à ces trois éléments (*cf.* figure 2.1). Elle médiatise la relation d'apprentissage (apprenant-compétence), la relation didactique (formateur-compétence) et la relation pédagogique (formateur-apprenant). Nous faisons ici l'hypothèse que le transfert des compétences acquises en environnement simulé vers l'environnement réel dépend notamment de la relation apprenant-compétence (apprentissage) et de la relation apprenant-formateur (pédagogique).

Dans ce chapitre, nous soutenons que l'intégration des capacités des systèmes tutoriels intelligents (ITS) au sein de la réalité virtuelle favorise ces relations, et par conséquent l'acquisition de compétences transférables. Nous montrons que l'intégration d'ITS dans

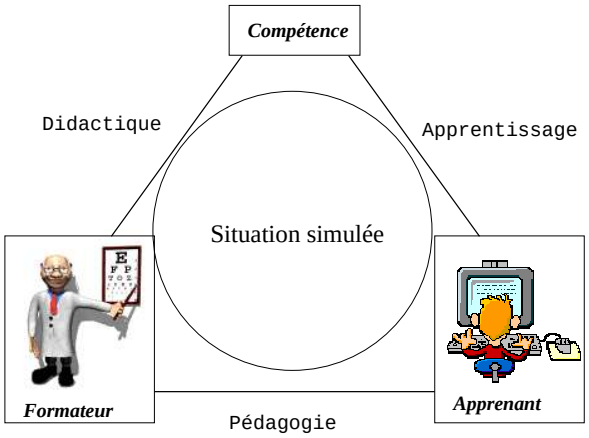


Figure 2.1 – La situation d'apprentissage.

les environnements virtuels de formation (EVF) est aujourd'hui partielle mais offre des perspectives intéressantes.

Ce chapitre s'articule autour de quatre parties. La première partie (2.1) étudie dans quelle mesure la simulation informatique permet de favoriser l'apprentissage de compétences. Cette analyse nous amène dans la partie 2.2 à présenter les systèmes tutoriels intelligents. Dans la partie 2.3, après avoir défini la réalité virtuelle, nous précisons son apport au transfert d'apprentissage. Cela nous permet, dans la partie 2.4, d'aboutir à l'évaluation de l'intégration des ITS au sein des EVF existants.

2.1 Modèles informatiques et apprentissage

Dans quelle mesure la simulation informatique permet-elle de favoriser l'apprentissage de compétences ? Dans cette section, nous étudions la relation entre simulation informatique et apprentissage de compétences à l'aune des théories de l'apprentissage (psychologie cognitive). Celle-ci nous conduit notamment à placer l'individualisation pédagogique et le problème du transfert au centre de nos préoccupations.

2.1.1 L'apprenant au centre du système de formation

Les différents modèles théoriques de l'apprentissage ont directement influencé la production de systèmes informatiques de formation.

Dès le début du vingtième siècle, l'apprentissage est décrit comme une succession d'essais et d'erreurs, permettant d'associer un stimulus à une réponse. Plus tard, Skinner contribue à l'évolution du conditionnement classique en mettant en évidence le « conditionnement opérant » (Skinner, 1958, 1974) : ces travaux inspirent directement les premières « machines à enseigner ». Ces systèmes de formation prennent comme principe la décomposition d'une connaissance ou d'un concept en éléments plus simples ; chaque élément est ensuite présenté

successivement à l'apprenant par un système de questions-réponses. Cet « enseignement programmé » est linéaire (les informations sont présentées séquentiellement à l'apprenant) et les informations nécessaires au système reposent uniquement sur le programme de connaissances à acquérir.

Crowder propose une évolution de l'enseignement programmé à l'aide d'une machine sophistiquée. Celle-ci contient des rouleaux de films sur lesquels sont fixées des séquences d'instructions multiples (Crowder, 1959). Contrairement à Skinner, il considère que les erreurs commises par un apprenant doivent être prises en considération. Crowder préconise d'utiliser les erreurs en imaginant un mécanisme qui prévoit les corrections : l'enseignement assisté par ordinateur devient algorithmique et non plus linéaire, et l'activité de l'apprenant commence à être prise en compte.

Dans les années soixante, les « machines à enseigner » laissent place aux premiers systèmes d'enseignement informatique. Les logiciels d'Enseignement Assisté par Ordinateur (EAO) s'attachent principalement à améliorer les connaissances des apprenants, en leur demandant de résoudre certains problèmes. L'approche cognitive symbolique, impulsée notamment par Simon et Hayes (1976), perçoit l'apprentissage comme un processus de représentation et d'interprétation du contexte. En résolvant les problèmes proposés par l'EAO, le sujet augmente progressivement ses connaissances. Contemporain de l'approche symbolique, le constructivisme piagétien s'intéresse également aux structures internes du sujet, cependant Piaget considère l'apprentissage comme un processus interactif de construction des connaissances par expériences successives, en relation avec un environnement (Piaget, 1974). Cette adaptation que l'on appelle aussi apprentissage, nécessite un sujet actif. Ainsi, l'action est primordiale (Hoc, 1990), ce qui est essentiel pour apprendre à agir et à réagir face à un environnement, bref pour devenir compétent. Les systèmes informatiques qui se rapprochent de ce courant de pensée sont qualifiés d'Environnements Interactifs d'Apprentissage par Ordinateur (EIAO) car ils sont susceptibles d'évoluer, de se modifier en fonction des réussites et des échecs de l'apprenant. Ce dernier « agit » et « interagit » pour apprendre. De plus en plus perfectionnées, ces machines peuvent être même qualifiées d'intelligentes : quand elles sont en mesure de raisonner sur le domaine enseigné et de s'adapter aux caractéristiques de chaque apprenant, on parle de système d'Enseignement Intelligemment Assisté par Ordinateur (EIAO).

Cette perspective conduit, au sein d'environnements informatiques, à valoriser l'accompagnement individualisé de l'apprenant. Cette médiation entre un objet d'apprentissage et un apprenant peut prendre différentes formes telles que des personnages autonomes simulés (tuteurs, compagnons, trouble-fête, *etc.*), des bilans individualisés de performances, des suggestions de solutions au problème, *etc.*

Valoriser la pédagogie, c'est valoriser l'interaction dans le processus d'apprentissage. Dans cette perspective *socio-constructiviste* (Bruner, 1983), le développement de nouvelles compétences repose sur la qualité de l'aide, de la médiation offerte par l'environnement de formation qui doit permettre à l'apprenant de réaliser des tâches qu'il ne peut pas, pour l'instant, réaliser seul (Vygotski, 1985) [1].

[1] D'un point de vue psychologique, l'apprenant se situe dans la ZPD (Zone Proximale de Développement).

2.1.2 Transfert d'apprentissage et environnements informatiques de formation

De façon complémentaire, depuis la fin des années 1970, l'évolution historique des modèles théoriques de l'apprentissage, conduit à reconnaître l'importance des « effets de contexte » (Richard, 1990). Nourrie des modèles behavioristes (conditionnement, loi de l'effet, *cf.* Thorndike 1932) et du constructivisme piagétien (le sujet se construit par l'expérience), la cognition située décrit l'apprentissage comme un processus d'adaptation permettant l'émergence d'actions singulières dans des contextes eux-mêmes singuliers.

Toute action est donc située et, toute acquisition en mémoire est liée, connectée spécifiquement à la situation de formation (notion d'encodage spécifique, Tulving 1976). Dans cette perspective, comment l'adaptation d'un apprenant à une situation de simulation informatique peut-elle favoriser son adaptation à des situations réelles ? Le transfert d'apprentissage désigne l'influence d'une acquisition sur une autre acquisition, une sorte de recontextualisation d'une connaissance ou d'une compétence (Tardif, 1999). Le transfert devient d'autant plus aléatoire que les contextes d'activation de la compétence (situations réelles) sont perçus comme éloignés du contexte d'apprentissage (simulation).

La diversité des expériences de l'apprenant et, donc les interférences entre les situations données à vivre, favorisent l'abstraction d'invariants, des éléments communs aux diverses situations. Apprendre consiste alors à abstraire, *i.e.* mettre en relation diverses informations communes aux contextes internes (connaissances et habiletés disponibles, émotions, *etc.*) et externes (consignes, informations visuelles, *etc.*) de l'apprentissage.

D'un point de vue théorique, différentes conditions d'apprentissage favorables au transfert peuvent être définies en fonction des caractéristiques de la compétence visée (Mendelsohn, 1994). Cette dernière est étudiée dans la section suivante.

2.1.3 Compétence : composants et conditions d'apprentissage

La compétence nécessite l'acquisition de savoir-faire et de savoir-dire ou connaissances, afin de les mobiliser « en action », dans un domaine professionnel (De Montmollin, 1984; De Terssac, 1996). L'acquisition de ces savoir-faire et de ces connaissances constituent des objectifs de formation.

Les savoir-faire mettent en jeu le corps (les capacités perceptivo-motrices : coordonner un déplacement, maintenir une posture de sécurité...) ou permettent d'agir sur l'environnement symbolique (procédures cognitives de choix dans les situations risquées, procédures de traitement de l'information dans les situations incertaines, reconnaissance de formes). Ces composants sont stockés en *mémoire procédurale*[2] (*cf.* figure 2.2). Ils se développent prioritai-

[2] La mémoire à long terme distingue :
▷ La mémoire procédurale qui permet l'acquisition d'habiletés perceptives, motrices et intellectuelles. Le contenu de cette mémoire n'est pas accessible volontairement et ne peut être analysé directement ou consciemment (parler, monter à bicyclette, conduire une automobile).

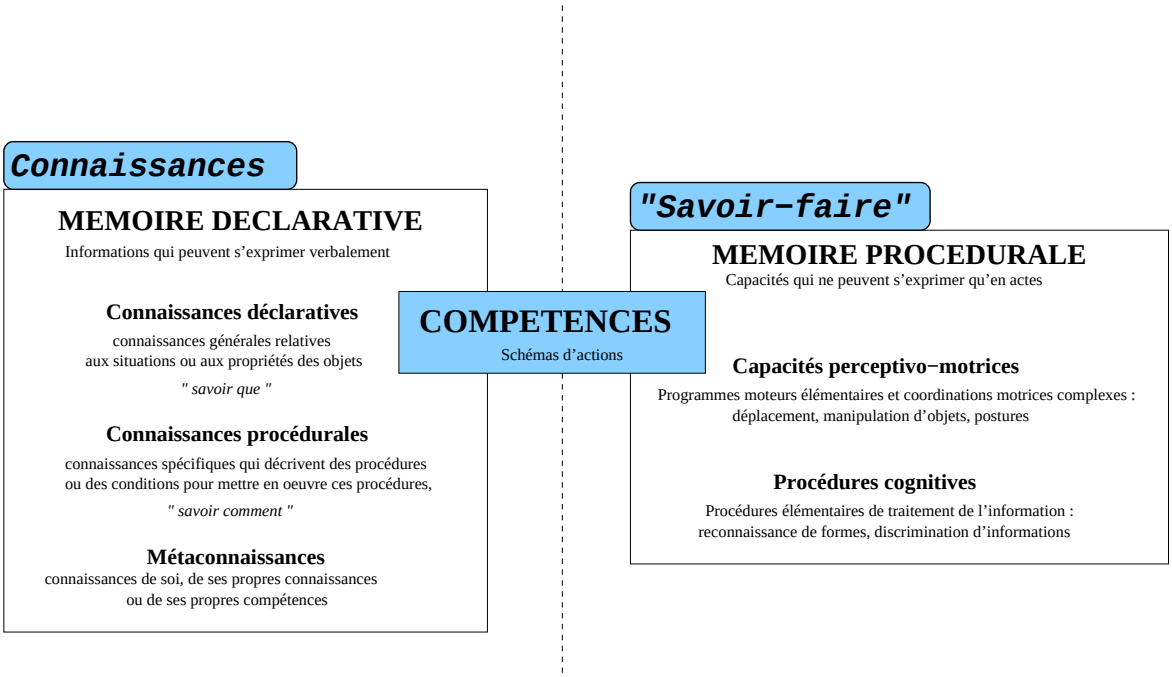


Figure 2.2 – Les composants de la compétence, à partir d'une figure de Wall (1986).

rement par l'action et par la répétition. Plus précisément, la répétition de situations variées et aléatoires permet de construire un savoir-faire adaptable.

Complémentairement, à cette manifestation « en acte », la compétence au sein d'un domaine repose aussi sur la possibilité d'expliciter, de décrire des opérations et des conditions permettant d'être efficace. Ce « savoir que faire » (De Terssac, 1996) ou « savoir comment faire » (George, 1983), ces informations utiles pour diriger l'action (règles d'actions) sont stockées en *mémoire déclarative* et sont parfois qualifiées de connaissances procédurales. D'autres informations verbalisables peuvent concerner l'action ou son contexte de réalisation (savoir que) mais sans en influencer la réalisation ; elles sont désignées par le terme de connaissances déclaratives (les règles générales de fonctionnement d'un système, les aspects historiques d'un domaine, les principes de sécurité, *etc.*).

Les connaissances procédurales peuvent être acquises par instruction. Elles peuvent aussi se construire dans l'action ou lors d'une mise à distance de l'action : réflexion sur ses réponses, confrontation entre apprenants, consultation de sources d'informations complémentaires face à un problème. Les connaissances déclaratives (plus générales, moins dépendantes des situations) nécessitent souvent un apport d'informations extérieures ; elles sont importantes dans une formation dans la mesure où elles favorisent le transfert des connaissances procédurales, et par conséquent la compétence, dans différentes situations, voire au-delà du domaine initial de formation.

▷ La mémoire déclarative qui est responsable de la mémorisation de toutes les informations sous forme verbale, *i.e.* celles que l'ont peut exprimer avec notre langage.

Enfin, certaines connaissances constitutives de la compétence concernent le sujet lui-même, et pas uniquement le domaine de référence. Ces métaconnaissances (Flavell, 1985) sont des informations que l'apprenant intègre à propos de ses propres connaissances ou de ses propres compétences. Elles sont accessibles à la conscience si on sollicite chez l'apprenant un processus métacognitif : auto-évaluation, autoscopie, analyse de sa propre activité... Cette connaissance de soi est particulièrement importante pour devenir compétent dans des contextes « coûteux » pour l'apprenant (risque, charge émotionnelle) qui justifient l'utilisation de la formation par simulation sans « mise en danger » objective.

La compétence se manifeste par la possibilité d'agir et de réagir de façon contextualisée tout en étant adaptable, efficace dans des situations variées au sein d'un domaine de référence. Former un individu compétent nécessite donc qu'il puisse transférer ce potentiel d'action du contexte de simulation vers diverses situations réelles. Les procédures pédagogiques les plus favorables au transfert d'apprentissage sont dépendantes des composants de la compétence considérés comme « dominants ».

Certaines compétences reposent essentiellement sur des savoir-faire (De Terssac, 1996), dans ce cas le transfert des apprentissages procéduraux peut être favorisé par « la variation systématique et aléatoire des contextes » (Mendelsohn, 1994).

D'autres compétences mobilisent prioritairement des connaissances procédurales et déclaratives, elles nécessitent « un savoir-juger » (Perrenoud, 1996). Le transfert nécessite ici une mise à distance de l'action, un effort de prise de conscience des procédures et de leurs conditions d'utilisation. Par exemple, on peut favoriser le transfert, si, dès la situation d'apprentissage de la compétence, on informe l'apprenant des analogies entre le contexte de la formation et le contexte de réinvestissement : notion de « transfert informé » (Gick et Holyoak, 1983).

Enfin, quels que soient les domaines professionnels, on reconnaît aujourd'hui la dimension sociale de la compétence : l'apprenant doit agir et réagir avec d'autres et en fonction des décisions d'autrui (Vergnaud, 1995). Deux niveaux d'interactions sociales (deux niveaux de compétences) peuvent être distingués : ajuster son activité individuelle à l'activité collective ; articuler, organiser son activité avec celle du groupe (De Terssac, 1996). Penser au transfert, c'est alors penser l'apprentissage en « communauté de pratiques » (Mendelsohn, 1994), c'est repenser la simulation comme situation d'interactions sociales.

2.1.4 EIAH et compétences

Nous venons de montrer, d'une part que les environnements informatiques de formation doivent proposer une aide pédagogique adaptée aux apprenants, et d'autre part qu'il est nécessaire de mettre le transfert des compétences au cœur des préoccupations de toute formation. Il s'agit maintenant de s'interroger sur les solutions que peut apporter un environnement informatique de formation pour répondre à ce double objectif.

L'**EIAH** regroupe les travaux portant sur les **E**nvironnements **I**nformatiques ou Interactifs pour l'**A**pprentissage **H**umain (Balacheff, 1998). Il est définit comme

> « un environnement informatique conçu dans le but de favoriser l'apprentissage humain [...]. Ce type d'apprentissage mobilise des agents humains [...] et artificiels

> [...] et leur offre des situations d'interaction [...] ainsi que des conditions d'accès à des ressources formatives. » (Tchounikine et al., 2004).

Les travaux sur la conception des EIAH portent sur plusieurs facettes que nous regroupons selon trois catégories :

1. les *hypermédias* qui visent à faciliter l'accès aux ressources pédagogiques. Ils regroupent les outils permettant la conception et la mise en page de présentations multimédias destinées à l'Internet ou à un intranet. L'hypermédia n'impose pas de parcours, mais permet à l'apprenant de naviguer entre les données. Apprendre consiste alors à s'informer, ce qui renvoie à une conception cognitive symbolique. On peut douter que la mémorisation de ces données verbales, ces connaissances liées à un domaine, suffisent à elles seules à la construction d'une réelle compétence. En effet, dans les situations professionnelles complexes, le passage à l'action ne peut être considéré comme la simple conséquence d'une base de connaissances.
2. les *micromondes* qui simulent des situations d'apprentissage. Ce sont des systèmes informatiques ouverts permettant à l'apprenant (ou l'utilisateur) d'explorer un domaine ou un dispositif avec un minimum de contraintes de la part du système, en combinant des opérations élémentaires généralement analogues à des schémas familiers (déplacement, construction, sélection, *etc.*). Les micromondes reprennent le principe constructiviste d'un apprentissage par expérience en soulignant l'apport d'une pédagogie centrée sur la découverte et l'intérêt. Le rôle du professeur consiste à proposer un environnement structuré et riche afin que l'apprenant découvre par lui-même les contradictions et ainsi invente de nouvelles structures. Cette conception de l'apprentissage est très présente dans l'enseignement scientifique : par exemple, Papert (1981) propose d'utiliser les micromondes comme outils d'enseignement. L'objectif pédagogique assigné à ces environnements est souvent ambitieux. L'élève doit apprendre en faisant, voire apprendre à apprendre. Il se sert de l'environnement pour s'adapter et stabiliser de nouveaux savoir-faire, mais il est aussi censé réfléchir sur ses propres connaissances et ses propres stratégies d'apprentissage (acquisition de métaconnaissances). Cependant, les micromondes ne proposent pas d'aides pédagogiques. Ces systèmes éliminent donc d'emblée ce qui est aujourd'hui reconnu comme un vecteur essentiel d'apprentissage : la médiation, le guidage, l'accompagnement (Bruner, 1983).
3. les *Systèmes Tutoriels Intelligents* (ITS : Intelligent Tutoring System) qui fournissent une assistance aux différents acteurs de l'apprentissage (apprenant ou formateur). Ils s'appuient sur les techniques de l'Intelligence Artificielle pour représenter des connaissances et effectuer un raisonnement. Ils visent à produire des systèmes qui simulent un enseignant humain, en ajoutant des capacités de résolution (d'où l'adjectif Intelligent), lui permettant ainsi de solliciter l'apprenant lorsque ce dernier commet une erreur dans la résolution d'exercice (d'où l'adjectif Tutoriel). Dès lors, on peut définir un ITS comme étant un système qui permet à l'environnement de formation de s'adapter à la diversité des apprenants. Il peut donner aux EIAH les moyens de leurs finalité principale : faire acquérir des compétences dans un domaine particulier, en fournissant des informations individualisées. Il permet aussi de faire réfléchir chaque apprenant sur la façon dont il s'y est pris (méthode) pour résoudre le problème (apprendre à apprendre). Enfin, un ITS peut fournir au formateur un ou plusieurs scénarios pédagogiques, particulièrement adaptés à chaque apprenant. Ainsi, les ITS sont particulièrement intéressants

pour l'apprentissage de compétences. La section suivante présente ces systèmes, décrit leurs composantes et leurs utilisations pour la formation.

2.2 Systèmes tutoriels intelligents

Les ITS forment un courant particulier de l'EIAO, qui a pour principe la personnalisation dans la formation. L'idée est d'introduire un système qui prête attention aux besoins spécifiques de l'apprenant, évalue et diagnostique ses problèmes, et fournit l'aide nécessaire. Ainsi, les ITS répondent à la nécessité de placer l'apprenant au centre de la situation d'apprentissage (section 2.1.1).

2.2.1 Présentation

Les ITS offrent des capacités de raisonnement et de résolution, passant par le recueil de connaissances sur l'apprenant. Ils évaluent les connaissances acquises par l'apprenant, en comparant ses activités et les informations sur le domaine, ils proposent alors des assistances adaptées. L'assistance est de différentes natures : suivi des activités de l'apprenant, analyse des difficultés de l'apprenant, instructions pour aider l'apprenant dans la relation d'apprentissage compétence-apprenant, propositions d'intervention pédagogique destinées au formateur (relation pédagogique formateur-apprenant), *etc.*

La conception de tels systèmes fait intervenir des spécialistes de l'IA, du domaine enseigné et de l'enseignement (Zampa, 2003). Chacun a un rôle à jouer dans les composantes du système que nous présentons maintenant.

2.2.2 Composantes des systèmes tutoriels intelligents

Un ITS est décrit en utilisant plusieurs fonctions ou composantes majeures (Wenger, 1987). Les premiers ITS étaient composés d'une expertise sur le domaine, d'une expertise sur ce qui doit être appris, et d'une représentation de ce que l'étudiant a appris ou non. Burns et Capps (1988) identifient ces composantes comme les trois modules d'un ITS. Ils correspondent au « module de l'expert » (Anderson, 1988), au « module de diagnostic de l'étudiant » (Van Lehn, 1988) et au « module d'instruction et de curriculum » (Halff, 1988). Plus tard, un quatrième module est venu s'ajouter au trois premiers, le « module d'interface » qui permet de représenter les connaissances dans l'environnement. Issus des modules précédents, les quatre modèles d'un ITS (*cf.* figure 2.3) sont donc (Woolf, 1992) :

1. le *modèle du domaine*, représentant la connaissance de l'expert sur le domaine ;
2. le *modèle de l'apprenant*, permettant d'établir l'état de ses connaissances à un instant donné ;
3. le *modèle pédagogique*, permettant d'effectuer des choix d'enseignement selon le comportement et le modèle de l'apprenant ;

4. le *modèle d'interface*, permettant l'échange d'informations entre le système et l'utilisateur.

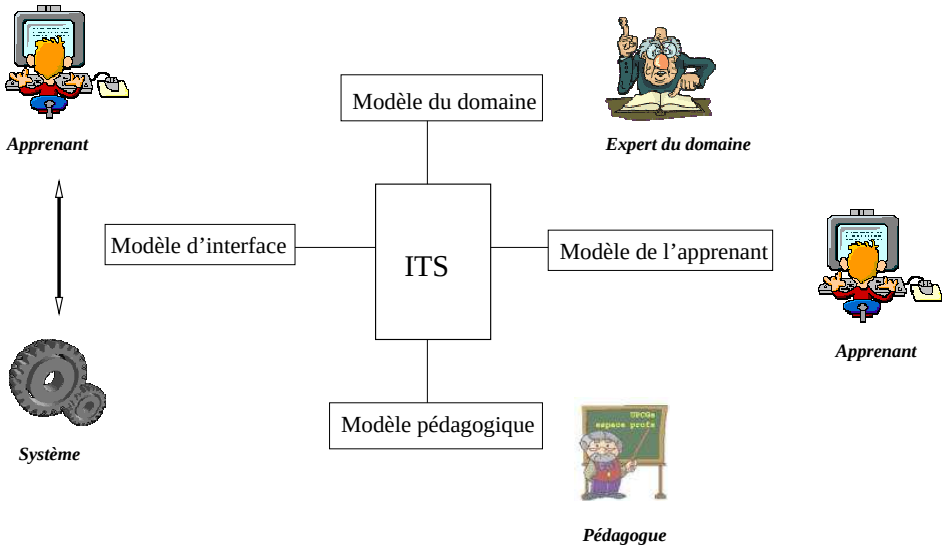


FIGURE 2.3 – Les quatre modèles d'un ITS.

Les sections suivantes précisent chacun de ces modèles.

2.2.2 - A Modèle du domaine

Expert du domaine

Le modèle du domaine représente l'expertise de l'enseignant sur le domaine. Il est aussi appelé modèle de l'expert puisqu'il définit les connaissances d'un expert sur un domaine particulier. Il ne contient pas seulement une description des compétences à acquérir, il propose une *représentation interne* de la compétence à construire.

Le modèle du domaine doit être en mesure de générer des solutions aux problèmes dans le même contexte que celui où se trouve l'apprenant, afin que les réponses respectives puissent être comparées. Ainsi, le système est en mesure de déterminer les différences et correspondances entre les actions de l'apprenant et celles qui sont attendues. Enfin, les connaissances sur le domaine permettent de générer des explications relatives à la solution de l'expert. La réalisation de ces fonctions nécessite la mise en place de la représentation des connaissances de l'expert.

Formalisme pour la représentation des connaissances

La représentation des connaissances nécessite l'utilisation d'un formalisme. Le formalisme logique a été l'un des premiers proposés pour représenter des connaissances, et constitue toujours la base de nombreuses recherches en IA. Ce formalisme utilise un langage, des axiomes et des règles (logique des propositions, floue, modale, *etc.*) permettant de représenter la véracité, l'incertitude, la temporalité, *etc.* Par exemple, si l'on considère des procédures, la

connaissance à exprimer est l'enchaînement des actions. Elle peut s'écrire en logique[3] par un prédicat du type « après[action1, et(action2,action3)] » spécifiant que l'action1 sera suivie de l'action2 et de l'action3.

Les sciences cognitives, qui s'intéressent aux mécanismes de l'intelligence, utilisent généralement un autre formalisme à base de graphes. Il réunit, sous la forme d'un graphe, les notions représentant les connaissances et leurs interconnexions. Les nœuds représentent les connaissances du domaine à acquérir par l'apprenant et les liens représentent leurs connections. Ces liens ne sont pas considérés comme des connaissances à acquérir par l'apprenant mais fournissent une information au système tutoriel lors de ses choix pédagogiques. Les réseaux sémantiques (Quillian, 1968), les réseaux à propagation de marqueurs (Fahlman, 1979) ou les graphes conceptuels (Sowa, 1984) permettent de représenter le savoir et un mécanisme d'inférence de l'interpréter pour apporter le savoir-faire.

Enfin, un formalisme de description logique de représentation des connaissances appelé « frame » (Minsky, 1975) est également utilisé. On définit un « frame » comme une structure de données regroupant l'ensemble des éléments relatifs à une connaissance. Divers types d'informations sont associés à chaque « frame », certaines d'entre elles concernent l'utilisation de ce « frame » (savoir-faire).

Utilisation des connaissances

Il est possible de manipuler et d'interpréter la connaissance implémentée avec l'un des formalismes mentionnés précédement, ce qui revient à simuler le savoir-faire tel qu'il est défini en section 2.1.3. Un problème classique réside dans le choix entre plusieurs règles contradictoires activées. Des moteurs d'inférence (Dillenbourg, 1994) comme Soar (Newell, 1990) ou Prolog (Kowalski, 1974) imposent ces choix. Une autre solution est d'intégrer des méta-règles au sein de systèmes experts qui gèrent l'activation des règles. Cette deuxième solution permet de personnaliser le raisonnement et peut exprimer, par exemple, que si l'apprenant est débutant, alors telles ou telles règles ne sont pas utilisables.

2.2.2 - B Modèle de l'apprenant

Apprenant

Le modèle de l'apprenant apporte une mesure des connaissances de l'apprenant sur le problème (Leman et al., 1996; Py, 1998; Webber et Pesty, 2002). Il est aussi appelé modèle de diagnostic puisqu'il permet de mesurer la progression de l'apprenant. Ce modèle doit contenir une représentation du profil de l'apprenant, établie et mise à jour soit par un dispositif hors ligne (questionnaire par exemple), soit directement à partir des interactions que l'apprenant opère avec son environnement.

La modélisation de l'apprenant est un problème réputé difficile (Bruillard, 1997). Self (1988) offre une clarification sur son rôle et son usage. L'auteur précise que le modèle de l'apprenant doit permettre de répondre à quatre types de questions :

▷ que *peut faire* l'apprenant ?

[3] Pour être plus précis, cet exemple utilise la logique temporelle de Allen (1983).

- ▷ que *sait-il* ou que *sait-il sur « le faire »* ?
- ▷ quel *type* d'apprenant est-il ?
- ▷ qu'a *déjà fait* l'apprenant ?

Ainsi, le modèle de l'apprenant est un quadruplet (P,C,T,H). P représente le savoir-faire, C les connaissances déclaratives et procédurales, T les traits individuels (profil cognitif) et H l'historique de l'apprenant (Zampa, 2003). Répondre à ces quatre questions permet d'évaluer les compétences de l'apprenant. Le modèle offre alors au système la possibilité de s'adapter à l'apprenant. La mise à jour des informations concernant l'acquisition de connaissances et de savoir-faire est une difficulté qui pose le problème de la construction d'un tel modèle.

Construction

La construction du modèle peut utiliser différentes méthodes :

- ▷ la méthode dite de l'*overlay*, où l'expertise sur le domaine d'apprentissage est découpée en unités de base et où le modèle de l'élève se compose d'un sous-ensemble de ces unités. La connaissance de l'étudiant est considérée comme une sous-partie de la connaissance de l'expert.

  > « Un modèle vide correspond à l'élève qui n'aurait aucune connaissance du domaine, tandis qu'un modèle identique à celui de l'expert correspond à l'élève qui aurait atteint le même niveau de maîtrise qu'un expert du domaine »(Py, 1998).

  Chaque item de connaissance peut être étiqueté par une valeur discrète (connue / inconnue) ou continue (de 0 à 1). Ce principe a été utilisé dans le cadre des tuteurs Wusor (Stansfield et al., 1976) et Guidon (Clancey, 1983). Ce modèle considère que l'étudiant ne va rien apprendre en dehors de ce que l'expert a prévu (Bruillard, 1997). Ainsi, il n'y a pas de mécanisme pour distinguer les connaissances non acquises de celles qui n'ont pas été encore présentées. Les erreurs commises par l'élève s'expliquent en termes d'absence de connaissances : c'est l'ignorance de telle règle ou de tel concept qui amène l'élève à ne pas jouer le meilleur coup possible ;
- ▷ la méthode dite *différentielle* qui est une extension de l'*overlay*, où on distingue les connaissances disponibles des connaissances et procédures à activer dans une situation particulière. Cette méthode a été utilisée pour le tuteur West (Burton et Brown, 1982) ;
- ▷ la méthode dite du *buggy model* qui est également une extension de l'overlay. L'approche consiste à présenter dans le modèle de l'apprenant des règles dont l'application produit un résultat incorrect (Webber, 2003). Cette méthode a été utilisée par Brown et Burton dans les systèmes Buggy (Brown et Burton, 1978) et Debuggy (Burton, 1982). Les connaissances sont représentées par un réseau de procédures élémentaires. Toute procédure correcte peut être remplacée par une procédure incorrecte ayant le même domaine d'application. A partir d'un ensemble de réponses données par un élève, Debuggy construit le réseau de procédures correctes et incorrectes dont le comportement se rapproche le plus de l'élève.

Évaluation des compétences

L'un des objectifs du modèle de l'apprenant est d'évaluer les compétences qu'il reste à acquérir. Plusieurs méthodes existent :

- ▷ le *model tracing* compare les étapes effectuées par l'étudiant et les étapes existantes dans les règles procédurales définies dans le modèle du domaine. Cette approche a été utilisée par le tuteur Lisp (Anderson et Reiser, 1985) ;
- ▷ le *issue tracing* est une modification du *model tracing*. Ce modèle n'a pas pour but de modéliser le processus de résolution du problème mais plutôt de déterminer ce qu'il reste à apprendre à partir d'une mise à jour des compétences acquises par l'étudiant. Le tuteur West a utilisé cette méthode (Burton et Brown, 1982).

Évaluation du profil : ABITS

Abits (Capuano et al., 2000) est un ITS qui évalue à chaque instant le profil de l'apprenant et son état cognitif pour mettre à jour le modèle de l'apprenant. Cet état cognitif regroupe l'ensemble des degrés de connaissance atteints par l'apprenant. Un nombre flou est utilisé pour représenter le niveau dans chaque concept. Abits applique une fonction sur l'état cognitif pour prendre en compte l'oubli au fur et à mesure du temps. Le profil de l'apprenant est composé d'un ensemble de préférences : modalité, niveau d'interaction, difficulté, *etc.* L'évaluation des préférences utilise également des nombres flous et est réalisée en mettant en rapport les degrés de connaissance lors d'étapes clefs. Par exemple, si l'apprenant maîtrise mieux un concept et que les ressources visitées à propos de ce concept ont été en grande partie des simulations, Abits en déduit que cet apprenant y est réceptif. Par conséquent, le système augmente la valeur du nombre flou correspondant à cette modalité.

2.2.2 - C Modèle pédagogique

Pédagogue

Le modèle pédagogique permet de définir les médiations visant à aider l'apprenant dans le processus d'apprentissage. Il permet de simuler le comportement décisionnel d'un enseignant. Il doit considérer des principes éducatifs, pédagogiques et psychologiques (Davies et al., 2001). Il se base sur le modèle de l'apprenant et sur le modèle du domaine. L'objectif principal du modèle pédagogique est de répondre à trois questions (Wenger, 1987; Lourdeaux, 2001) : *pourquoi intervenir ? quand intervenir ? comment intervenir ?*

Pourquoi intervenir ? Il s'agit de définir l'objectif spécifique de chaque intervention ; *i.e.* une connaissance ou un savoir-faire à acquérir. Mais ceci n'est pas suffisant, l'objectif est également guidé par des choix pédagogiques. Il peut être question de privilégier des acquisitions liées au domaine ou plutôt envisager l'acquisition d'une démarche, d'une stratégie d'apprentissage réinvestissable dans d'autres domaines.

Quand intervenir ? Le modèle pédagogique détermine quand une intervention est souhaitable, si l'apprenant doit être interrompu ou pas. L'intervention peut survenir à différents moments : suite à des erreurs commises par les formés, avant les erreurs pour mettre en avant les risques d'erreurs, suite à des questions de l'apprenant, *etc.* Déterminer quand intervenir est une décision subtile. Pour guider un apprenant, il est parfois plus efficace

de laisser l'apprenant chercher pendant un moment que de l'interrompre à chaque fois. D'un autre côté, livré à lui même, l'apprenant serait probablement découragé.

Comment intervenir ? Différentes méthodes peuvent être envisagées :
- ▷ la méthode *socratique*, où le système interroge l'apprenant afin de l'encourager à analyser ses propres erreurs (utilisé par Scholar (Collins et al., 1975) et Why (Stevens et al., 1982)) ;
- ▷ la méthode du *learning by doing*, où le système est actif et incite l'apprenant à solliciter des informations ;
- ▷ la méthode du *learning while doing*, où le système reste en tâche de fond et donne seulement des conseils ponctuellement ;
- ▷ la méthode du *coaching*, où le système laisse l'apprenant agir et attend jusqu'à ce qu'il demande de l'aide (utilisé par Sophie (Brown et al., 1975), Wumpus (Stansfield et al., 1976) et West).

Pour s'inscrire dans la dynamique de l'évolution des conceptions de l'apprentissage, le modèle pédagogique proposé par les environnements informatiques de formation devrait pouvoir proposer un apprentissage fondé sur l'interaction entre plusieurs personnes. L'apprentissage serait alors interactif et constructif. Dans cette perspective, quelques études s'inspirent des modèles socio-constructivistes, qui utilisent les interactions sociales et mettent en œuvre un apprenant et deux participants simulés : le tuteur et un autre apprenant (Chan et Baskin, 1990). Plus récemment, une autre stratégie pédagogique propose de simuler un apprenant « trouble-fête » qui va aider ou perturber l'apprenant humain (Aïmeur et al., 2000). Dans les deux cas, les notions de coopération ou de compétition, de conflit socio-cognitif jouent un rôle important dans le processus d'apprentissage (Aïmeur et al., 2001).

Systèmes existants

Les systèmes existants sont pour la plupart issus des « outils-auteurs »[4] (Murray, 1999). Les systèmes Iris (Arruarte et al., 1997), Redeem (Major et al., 1997), Ide (Russell et al., 1998) et Gte (Van Marcke, 1998) proposent une représentation hiérarchique à plusieurs niveaux. Ces derniers distinguent les objectifs pédagogiques, stratégiques et des tâches (également appelés buts, événements et actions).

Une partie de ces systèmes contient un mécanisme basé sur des plans. Dans Ide et Gte, il est possible de spécifier une logique pour chaque règle de planification. Par exemple : « Pour enseigner les fonctions ⇨ ① Présenter un résumé, ② Enseigner les processus liés, ③ Enseigner les sous-fonctions, *etc.* ». La partie « Enseigner les processus liés » pourra être définie en utilisant une autre règle.

Certains systèmes proposent un enseignement « multi-stratégiques » et choisissent dynamiquement la stratégie la mieux adaptée en se basant sur les caractéristiques de l'apprenant. Certains systèmes contiennent des stratégies simples, chacune étant associée à une tâche ou à une connaissance. Par exemple, des stratégies différentes seront utilisées pour enseigner des faits, des procédures ou des concepts. D'autres systèmes comme Eon

[4] Les outils-auteurs sont des applications permettant de définir les différents aspects d'un tuteur informatique.

(Murray, 1996) et Redeem proposent de définir des conditions de sélectivité plus pointues, par exemple la stratégie « étudiant avancé » est choisie lorsque l'élève est expérimenté, le contenu de l'exercice de difficulté moyenne, *etc.*

La grande majorité des ITS qui comportent un modèle pédagogique, ne permet pas de le modifier. Néanmoins, Eon, Coca, Redeem, Ide et Gte permettent de le définir. Coca (Major et Reichgelt, 1992) utilise une méthode basée sur des règles du type SI-ALORS. Il propose un menu permettant de spécifier les deux composantes de chaque règle. De la même manière Eon fournit une interface graphique permettant de spécifier les procédures d'instructions. Dans Redeem, il est possible de définir ses propres « stratégies pédagogiques », en initialisant des paramètres clefs tels que « le taux de choix de l'étudiant » ou « le degré de retour ». Un exemple est une stratégie appelée « étudiants avancés » qui a un taux important de « choix de l'élève » et un « degré de retour » moyen.

2.2.2 - D Modèle d'interface

Le modèle d'interface (ou modèle d'interaction) a la responsabilité de la liaison entre la représentation interne du système et une interface ergonomique pour l'apprenant (Wenger, 1987). Il est en coopération avec le diagnostic (évaluation des compétences de l'apprenant) et la didactique du système. Une de ses fonctions est de finaliser la forme par laquelle le système transmet une information. En effet, même si le modèle pédagogique décide du déroulement et du contenu des actions didactiques, le modèle d'interface prend, quant à lui, en charge sa forme finale.

Pour définir ce modèle, on peut chercher à modéliser l'interaction humain-machine, *i.e.* les communications bidirectionnelles entre l'apprenant et le système (Miller, 1988). Plus précisément, comment peut-on étudier l'activité d'un apprenant face à un ordinateur ? A la suite des modèles, déjà anciens de Schiffrin et Schneider (1977) ou de Rasmussen (1986), Richard (1996) propose de concevoir l'interaction sujet-environnement d'un point de vue théorique à partir d'une double modalité de contrôle de l'activité : le contrôle externe *versus* le contrôle interne.

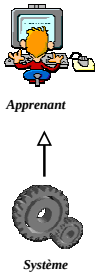
D'une part, l'activité, ici l'apprentissage, est l'objet d'un contrôle interne, volontaire, coûteux en attention, où le sujet oriente son apprentissage de façon stratégique en vue d'atteindre son but (Kermarrec et al., 2004). On considère qu'un utilisateur active volontairement un système d'aide s'il dispose de connaissances sur ses propres compétences (métaconnaissances), autrement dit, s'il est capable d'identifier ses difficultés.

D'autre part, l'activité du sujet est orientée par un contrôle externe, automatique, rapide, qui s'effectue à partir d'une activation directe des informations de sa mémoire à long terme par des informations contextuelles : les affordances. Le rôle des affordances dans l'interaction sujet-machine peut être décrit à partir du modèle « perception-action » ou modèle de la perception directe initié par Gibson (1958). Même si Gibson a circonscrit le champ d'application de son modèle à l'étude des actions motrices finalisées (locomotion vers un but, évitements d'obstacles, freinage, interception de mobiles) ce qui en fait un modèle particulièrement intéressant pour comprendre l'émergence de compétences motrices, la psychologie écologique a développé de nombreuses applications liées à ce concept. Par conséquent, la notion d'affordance a pris une place importante dans le domaine de la simulation

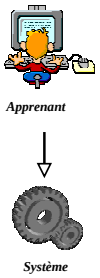
informatique. Ce modèle « écologique » n'est pas initialement un modèle de l'apprentissage, mais il décrit le fonctionnement d'un sujet confronté à un environnement dynamique (incertain et complexe).

Cette double modalité du contrôle (interne et externe) implique une communication bidirectionnelle entre l'apprenant et le système.

1. du *système vers l'apprenant.*
   L'interface se définit au travers des médias utilisés, de sa composition (Depover et al., 1998), mais aussi en fonction de la hiérarchisation des informations. Définir le type de média à utiliser pour traduire l'information conduit à privilégier l'immersion dans un environnement multisensoriel. Les interfaces immersives utilisées en réalité virtuelle proposent une interaction multimodale intéressante. L'utilisateur est immergé dans l'environnement (gant de données, casque stéréoscopique, *etc.*). Il se pose alors le problème de la difficulté d'adaptation des utilisateurs pour leur proposer des interfaces réellement « affordantes ». Par exemple, certaines personnes sont plus réceptives à une représentation purement visuelle, d'autres sonores, *etc.* Le concepteur de l'interface peut s'appuyer sur le modèle de l'apprenant pour déterminer à quel type de stimulus l'utilisateur sera le plus réceptif. Pour cela, on pourra utiliser des modèles (Card et al., 1983) prédisant les performances des utilisateurs (Zorola, 1995) et permettant de mesurer les critères d'utilisabilité (Nielsen, 1993; Coutaz, 1994; Le Bodic, 2005).



2. de l'*apprenant vers le système.*
   La comparaison d'un écart, ou dissonance, entre un but et un résultat active une boucle de régulation de l'activité de l'apprenant et provoque une communication de l'apprenant vers le système. Chez Norman (1980) l'intention, le but ou le motif d'agir est associé à l'allocation de ressources attentionnelles, à la recherche d'informations en mémoire à long terme ou à la reconnaissance de formes, à la détection d'éléments contextuels susceptibles de l'aider à accomplir sa tâche.



2.2.3 Évaluation des systèmes tutoriels intelligents pour la formation

L'efficacité des ITS a été montrée (Shute et Reigian, 1990), la plupart du temps en comparant deux groupes d'apprenants dans un environnement de formation bénéficiant ou non de l'assistance des ITS. Par exemple, une expérimentation a contrôlé des apprenants utilisant le tuteur Lisp (Anderson, 1990). Elle montre qu'ils terminent leurs exercices 30 % plus rapidement que ceux qui reçoivent une formation traditionnelle. L'examen final montre une différence de 43 % en terme de résultat entre les deux méthodes (Ong et Ramachandran, 2000). Dans une autre application, des apprenants utilisent un tuteur pour l'électronique dans l'AirForce. Ils obtiennent en 20 heures le niveau des apprenants ayant suivi une formation traditionnelle pendant 40 mois (Lesgold, 1988).

Nombre de méthodes traditionnelles d'instruction utilisent les ITS pour présenter à l'apprenant des faits et des concepts. Ces méthodes sont efficaces en exposant des grandes quantités d'information et en examinant la compréhension de l'apprenant (souvent par des questionnaires de contrôle). Cependant, elles inculquent souvent « une connaissance inerte ».

En effet, les apprenants acquièrent souvent les connaissances demandées (savoir), mais ils ne sont pas capables de l'appliquer correctement (savoir-faire). Ces méthodes ne sont donc pas adaptées à l'apprentissage de compétences.

En revanche, les systèmes qui emploient les simulations et d'autres environnements fortement interactifs offrent la possibilité d'appliquer les connaissances. Ces environnements actifs les aident à mettre en œuvre leur savoir en situation. Ces simulations, combinées avec un ITS, ont la possibilité d'améliorer les compétences des apprenants puisqu'ils associent connaissances et savoir-faire.

Un environnement de simulation interactif utilise souvent les notions présentes en réalité virtuelle. Cette dernière a déjà été considérée comme une technologie pouvant améliorer l'apprentissage dans les EIAH (Lourdeaux, 2001; Querrec, 2002). Elle a souvent été abordée uniquement par son aspect technologique. Elle fournit en effet, grâce aux périphériques adaptés, des interactions naturelles et une meilleure immersion dans le modèle simulé, une meilleure « présence » en somme (Fuchs et Moreau, 2003). Mais la réalité virtuelle ne doit pas être considérée uniquement par ses aspects technologiques ; elle peut bien plus pour l'apprentissage et le transfert (Burkhardt et al., 2003; Winn, 2002), ce que nous développerons dans la section suivante.

2.3 Réalité virtuelle

La réalité virtuelle est une discipline des sciences de l'ingénieur qui concerne la conception et la réalisation d'environnements virtuels participatifs. Elle est exploitée dans différents domaines industriels, dont la formation. A l'origine simple extension des techniques de synthèse d'images (fixes ou en séquences), elle est devenue interdisciplinaire en rassemblant divers champs des sciences de l'ingénieur et des sciences humaines et sociales. Avant d'analyser son utilisation pour la formation, il convient de préciser les caractéristiques essentielles d'un système de réalité virtuelle.

2.3.1 Définitions

D'un point de vue opérationnel, la réalité virtuelle se définit comme un système composé d'éléments logiciels et matériels simulant l'interaction réaliste d'un humain avec des objets virtuels qui sont des modélisations informatiques d'objets réels ou imaginaires (Fuchs et Moreau, 2003). Le mot virtuel qualifie ici la chose en puissance (qui a en soi toutes les conditions de son actualisation), par opposition à l'actuel (qui est en acte, ici et maintenant) (Tisseau, 2001). Indépendamment de son statut d'actuel ou de virtuel, l'objet peut être réel ou possible (imaginable). Ainsi est levée l'apparente opposition entre réel et virtuel.

D'après Tisseau (2001), comme toute réalité, la réalité virtuelle est accessible au sujet par trois types de médiations indissociables :

- ▷ la *médiation des sens* : l'objet est-il accessible à nos sens ? Il est alors perçu ;
- ▷ la *médiation de l'action* : réagit-il à nos sollicitations ? Il est alors expérimenté ;

▷ la *médiation de l'esprit* : peut-on s'en construire une représentation mentale ? L'objet est alors imaginé (modélisé).

L'objectif d'un système de réalité virtuelle est de faire éprouver des expériences au sujet humain (Deutsch, 2003). La spécification du système consiste, entre autre, à préciser comment cette triple médiation est exploitée. La conception du système définit, quant à elle, les supports de ces médiations : les générateurs d'images (visuelles, sonores, olfactives), les capteurs et actionneurs pour détecter les actions du sujet, et enfin les algorithmes de calcul des modèles. Les deux premiers points sont des problèmes que la technologie saura résoudre à court ou moyen terme (Deutsch, 2003), le défi se situe dans la programmation du comportement des environnements que les utilisateurs peuvent expérimenter, *i.e.* la manière dont l'environnement agit, par lui-même, et réagit aux actions de l'utilisateur.

2.3.2 Composantes de la réalité virtuelle

Un système de réalité virtuelle se distingue d'une autre application informatique, par le fait qu'il offre à l'utilisateur, la sensation d'être dans le monde virtuel et d'y agir. Cette notion de présence de l'utilisateur dans l'environnement virtuel a donc deux composantes : l'immersion, généralement multisensorielle, et l'interaction. Pour être complet, un environnement virtuel ne doit pas seulement assurer cette présence de l'utilisateur, il faut aussi « qu'il s'y passe quelque chose », et pas seulement en résultat d'une action de l'utilisateur. Les objets de l'environnement doivent donc avoir un comportement autonome. Cette notion d'autonomie est essentielle pour associer au rendu multisensoriel de l'informatique graphique, le rendu comportemental nécessaire en réalité virtuelle.

Les univers virtuels sont évalués en considérant des composantes préalablement définies (Zelter, 1992). Ainsi, Tisseau et Harrouet (2003) envisagent trois axes pour caractériser un système de réalité virtuelle : (**A**, **I**, **I**) pour **A**utonomie, **I**nteraction et **I**mmersion. Tout système informatique peut se positionner dans ce repère qui délimite un cube de côté 1 (*cf.* figure 2.4). Cette figure traduit les deux dimensions de la notion de présence (Interaction et Immersion).

Analysons les situations extrêmes qui correspondent aux sommets de ce cube auxquels on associe à titre d'illustration des exemples idéaux.

1. Selon l'axe **A**utonomie, on passe d'un système complètement sous le contrôle de son utilisateur, à un système complètement autonome sur lequel on a plus de contrôle (au moins directement). Un cas extrême est un virus informatique mutant qui est complètement hors de contrôle, qui n'offre aucune interaction et dans lequel on ne peut s'immerger ; ces coordonnées dans le repère (A,I,I) sont (1,0,0).
2. Suivant l'axe **I**nteraction, on augmente les possibilités d'interagir avec le système. Un système en (0,1,0) est par exemple un jeu vidéo tel que ceux de première génération (pensons au célèbre ping-pong). Le système est fortement interactif, il réagit aux sollicitations du joueur, mais ne fait rien de sa propre initiative (il n'a aucune autonomie) et l'immersion est pour le moins réduite.
3. Le troisième axe reflète la sensation d'**I**mmersion dans le système qui est liée au fait que l'environnement est directement (« naturellement ») accessible par les sens. En (0,0,1),

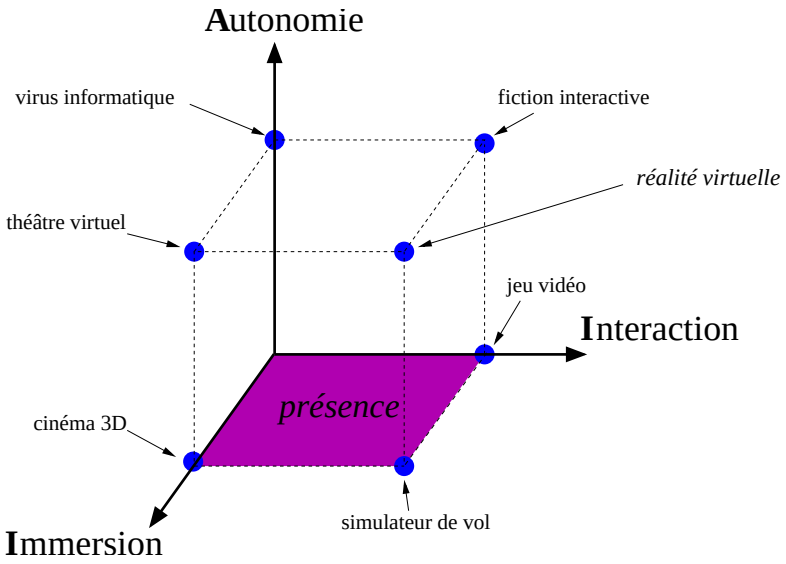


Figure 2.4 – Présence et autonomie, d'après (Tisseau et Harrouet, 2003).

on est par exemple dans une salle de cinéma, entouré d'images, la salle est équipée d'un système de son spatialisé avec éventuellement des sièges sur vérin : on est littéralement dans le film mais on n'agit pas sur le cours de l'histoire et le système est sous contrôle (quelle que soit la séance, on voit toujours le même film).

Un simulateur de vol se situe en (0,1,1). Son comportement est imposé et les situations qu'il permet d'expérimenter sont finies (même si le nombre de combinaisons est énorme). L'accent est principalement mis sur le réalisme du rendu, associé à une immersion totale (le pilote est dans une cabine en tout point identique à une cabine réelle), et sur l'interactivité, donc sur la présence de l'utilisateur dans le monde simulé. Cela passe par la définition d'interfaces comportementales performantes.

En (1,0,1) se situe le théâtre virtuel : les acteurs virtuels improvisent en partie leurs jeux et peuvent avoir des réactions différentes aux actions des autres acteurs ; leur comportement n'est donc pas totalement sous le contrôle du scénariste. De ce fait, le déroulement effectif de la pièce n'est pas envisageable à priori, et l'utilisateur se voit offrir un contexte différent à chaque représentation : il vit une expérience externe différente à chaque fois.

Le système extrême se situe en (1,1,1). On est immergé dans un environnement que l'on perçoit par différents sens ; on peut l'expérimenter par la médiation de l'action. Ce monde perçu et expérimenté, agit plus ou moins indépendamment de soi. Enfin, pour le comprendre, et donc continuer à interagir avec lui, il faut s'en construire une représentation mentale afin de « calculer » soi-même son propre comportement. Le sujet expérimente alors le monde par les trois médiations, il est spectateur, acteur et créateur.

2.3.3 Réalité virtuelle et formation

Le repère (A, I, I) fournit un cadre pour le positionnement des applications informatiques vis-à-vis de l'utilisation qui est faite de la réalité virtuelle Il ne s'agit bien sûr que d'un positionnement relatif et imprécis : il s'agit d'identifier sur quelle face ou sur quel sommet telle application se situe. Néanmoins, cette analyse permet de mieux cerner les intérêts de la réalité virtuelle pour la formation, notamment sous l'angle du transfert de compétences.

Les premiers systèmes d'EAO sont les plus proches du point (0,0,0) et le glissement vers les EIAO (i comme interactif) traduit bien un déplacement sur le deuxième axe vers le sommet (0,1,0). L'immersion dans la situation d'apprentissage est faible (celle-ci est de ce fait peu réaliste) ; le comportement du système est parfaitement prédictible et répétitif ; l'interaction consiste le plus souvent à sélectionner une réponse et d'obtenir un feed-back (précalculé). Ces systèmes sont donc intéressants pour enseigner des savoir-faire, des procédures contextualisées, mais qui s'avéreront peu transférables.

Les systèmes d'EAO multimédias jouent sur l'axe Immersion : on propose à l'apprenant des images, des documents sonores ou des vidéogrammes qui sont statiques (précalculés) ; ils sont donc proches du sommet (0,0,1) ; l'interaction est du même ordre que dans les systèmes d'EAO classiques.

Les logiciels d'autoformation pour les enfants sont en général dans la région $(0,\frac{1}{2},\frac{1}{2})$ du cube : l'interactivité est plus ou moins forte et l'immersion généralement faible ; la sensation de présence est peu développée, et le reproche souvent formulé à leur égard est que les enfants ont parfois du mal à « entrer » dans la situation. Les potentialités de transfert de compétences sont donc faibles, car la signification ou la perception de la situation d'apprentissage est éloignée des significations que peuvent prendre les situations réelles.

Les simulateurs de conduite, et plus généralement les micromondes, sont en (0,1,1). Le réalisme du rendu sensoriel et comportemental est fort. Dans ce contexte, on exploite souvent le processus de renforcement, ce qui favorise l'acquisition de savoir-faire. Le transfert d'apprentissage dépend alors du réalisme et de la variabilité des contextes proposés par le simulateur.

Plus un EVF est proche du sommet (1,1,1), plus l'apprenant est confronté à des situations variables et à un monde complexe. Cette variabilité (apportée par l'autonomie de certaines entités dans certains contextes) est doublement intéressante pour construire, par abstraction, des nouveaux schémas d'actions adaptables. Soit l'apprentissage est procédural quand il s'agit d'acquérir des compétences où des savoir-faire sont « dominants », soit l'apprentissage est déclaratif quand la compétence requiert un effort de compréhension, la mobilisation de connaissances. De façon générale, la variabilité « systématique et aléatoire des contextes » a été présentée comme une condition essentielle d'abstraction et donc de transfert (Mendelsohn, 1994). La pratique variée, *i.e.* la répétition de situations différentes (mais analogues), provoque des interférences entre situations, ce qui favorise l'oubli : on ne retient alors que les points communs aux deux situations. La simulation informatique (autorisant de nombreuses répétitions) et l'utilisation de la réalité virtuelle (permettant une importante variabilité des situations grâce à l'autonomie des agents) offrent des perspectives séduisantes pour le transfert.

Assez peu d'études portent directement sur le transfert des compétences acquises en situation de simulation en réalité virtuelle vers des situations réelles (Wilson, 2000b). Si la situation de simulation permet d'apprendre, elle ne permet pas toujours le transfert d'apprentissage (Kozak et al., 1993). Toutefois, les résultats sont particulièrement encourageants quand il s'agit de construire un espace (Regian et al., 1992), de mémoriser des localisations (Witmer et al., 1996; Johnson, 1998), d'apprendre à piloter un hélicoptère (Johnson et Wightman, 1995), d'adopter des conduites d'urgence (Bliss et al., 1997), d'acquérir des habiletés sensori-motrices (Rose et al., 2000) ou lorsqu'on vise des compétences exigeantes du point de vue des facteurs cognitifs, affectifs et des capacités sociales (Depover et al., 1998). Ce qui caractérise ces études, c'est l'étroite dépendance entre l'efficacité de la formation et les conditions d'apprentissage proposées aux sujets, ce qui souligne bien le rôle que pourraient jouer les ITS au niveau de la relation apprenant-compétence.

2.4 Intégration des modèles des ITS dans les EVF

Cette partie unifie les ITS et la réalité virtuelle dans les EIAH. Nous étudions dans quelle mesure les environnements virtuels de formation actuels intègrent les modèles des ITS présentés précédemment (section 2.2.2). Nous montrons quels sont les apports engendrés pour la formation. Afin de structurer cette présentation, nous proposons trois catégories d'environnement virtuel. Cette division se base sur les modèles des ITS qu'ils intègrent.

La première catégorie (section 2.4.1) regroupe les applications n'intégrant aucun de ces modèles. Aucun raisonnement sur le domaine d'apprentissage ne peut alors être proposé par le système. De même, n'ayant pas de représentation de l'apprenant, le système ne peut adapter l'environnement. Enfin, n'ayant pas de connaissances pédagogiques, il ne peut guider l'apprenant dans son apprentissage, et ne peut pas aider le formateur à organiser ses sessions de formation. Ces environnements offrent tout de même la possibilité d'appréhender et de manipuler l'environnement comme dans le réel. Ils sont alors destinés à l'acquisition de savoir-faire.

La deuxième catégorie d'environnements virtuels pour la formation (section 2.4.2) est constituée des applications intégrant un modèle du domaine et/ou un modèle de l'apprenant. Le système peut alors adapter l'environnement à l'apprenant, lui proposer des explications sur le sujet de la formation et agir à sa place. Ces systèmes sont également capables d'assister le formateur dans le suivi et l'évaluation de l'élève. Ils permettent donc, non seulement de manipuler l'environnement et de favoriser l'acquisition de savoir-faire, mais également d'accéder à des connaissances sur le sujet de la formation. Il s'agit là des deux composantes de la compétence. Néanmoins, le système ne dispose pas de connaissances pédagogiques qui lui aurait permis de décider (aide à l'apprenant) ou de suggérer (assistance au formateur) quand, comment et pourquoi intervenir lors de la session de formation.

La dernière catégorie (section 2.4.3) rassemble les environnements virtuels qui présentent non seulement les modèles du domaine et de l'apprenant, mais également un modèle pédagogique. Ainsi, le système peut s'adapter à l'apprenant et lui proposer des aides, et également intervenir de manière autonome pour favoriser chez l'apprenant l'acquisition de compétences transférables.

2.4.1 Environnements n'intégrant aucun des modèles

Le premier niveau d'utilisation de la réalité virtuelle dans la formation est représenté par les simulateurs. Aucun des modèles présentés précédemment (section 2.2.2) n'est représenté explicitement dans le système. Les connaissances sur le domaine, utilisées lors de la simulation ou lors de la phase de paramétrage, ne sont pas réifiées (aucune représentation interne) (Winn, 1993) ; il n'est donc pas possible pour le système de raisonner dans un but pédagogique. Enfin, l'utilisation de tels outils est de la responsabilité du formateur ; c'est lui qui doit gérer les scenarii ou les niveaux de difficultés, même si certaines de ces simulations fournissent des outils pour l'aider. Les exemples les plus connus et utilisés sont issus de la formation professionnelle, nous en étudions ici quelques uns. Ces applications se placent selon le repère (A, I, I) proposé en section 2.3 en un point (0,1,1).

EDF[5] *R & D* développe un atelier d'aide à la maintenance et à la conduite des ponts polaires. Le « projet polaire » simule l'utilisation d'une grue pour la manutention de charges dans le bâtiment réacteur des centrales nucléaires (Levesque, 2003). Le pontier, n'ayant pas toujours la visibilité sur la charge qu'il transporte, doit communiquer avec le chargé de manœuvre. Le pontier est immergé dans l'environnement par le biais d'un visiocasque et d'une console à leviers contrôlant la grue. Le chargé de manœuvre est représenté par un avatar contrôlé par un *joystick* ou par une commande vocale. Le simulateur permet de représenter les conséquences qui découleraient de certains choix de manutention (*cf.* figure 2.5). Il s'agit pour le formateur d'observer, d'accompagner et d'évaluer plutôt que de transmettre des connaissances. Dans cet environnement, l'objectif pédagogique n'est pas la manipulation de la grue, mais la communication entre les différents acteurs de la tâche à réaliser.

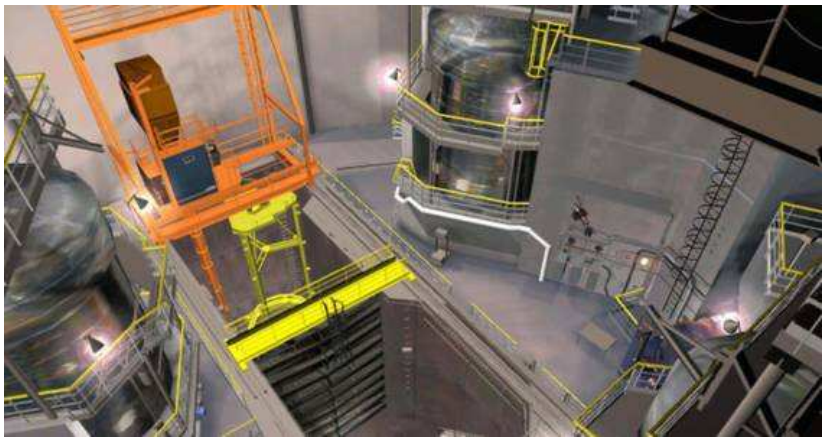


Figure 2.5 – Le projet polaire vu par le chef de manœuvre (Levesque, 2003).

Thalès Training & Simulation propose un simulateur de formation, dans lequel l'objectif est de former à l'utilisation d'un outil professionnel et de mettre l'apprenant en conditions réelles ou extrêmes. Ce simulateur est destiné à la conduite de poids lourds Trust 800[6]. Il est présenté comme un outil au service de la pédagogie, puisque le formateur peut créer des situations particulièrement dangereuses. L'apprentissage se déroule en trois phases. La

[5] EDF : Electricité De France

[6] `http ://www.ttsl.co.uk/produtcs/roads.htm` accédé le 01/04/05

première phase correspond à un apprentissage de base de la conduite. La seconde permet le transfert en situation réelle, le simulateur est utilisé pour reprendre des apprentissages non acquis. La dernière développe les compétences comportementales liées à des événements dangereux ou critiques (déversement de camion, verglas, *etc.*). Après chaque utilisation, une phase de « débriefing » est organisée favorisant la prise de conscience et la représentation des comportements, ce qui devrait favoriser le développement d'une compétence généralisable, un savoir « juger les risques ».

Ces deux derniers environnements sont destinés à la formation professionnelle, il existe également des environnements virtuels pour l'éducation scolaire. L'application Eve (Environnements Virtuels pour les Enfants) est un environnement virtuel destiné aux enfants de classes primaires (Popovici et al., 2004). Il concerne l'apprentissage de la lecture, la compréhension de l'écrit et la construction de phrases. Il forme également les enfants au travail en équipe faisant coopérer des enfants de Roumanie, du Maroc et de France via Internet. Lors de la phase de reconstruction de phrase (*cf.* figure 2.6), le système indique à l'enfant si la position des mots est correcte ou incorrecte. Dans le premier scénario, les enfants travaillent en parallèle dans des salles différentes. Le second scénario propose aux enfants de se retrouver dans une salle commune et de reconstruire ensemble une histoire en utilisant un mécanisme de vote électronique. Les interactions sociales sont utilisées pour une prise de décision à la majorité (introduction au principe de démocratie). Eve permet des interactions sociales, des apprentissages coopératifs ou des conflits socio-cognitifs, favorables au développement de nouvelles compétences (modèle socio-constructiviste, *cf.* section 2.1.3). Selon le repère proposé pour classer les environnements virtuels, Eve est plus proche de l'origine sur les axes immersion et interaction que les deux applications précédentes. La caractéristique principale de Eve est le partage de l'environnement virtuel par plusieurs utilisateurs.

Figure 2.6 – L'application Eve (Environnements Virtuels pour les Enfants) propose aux enfants la reconstruction de phrase (à gauche en français, à droite en japonais) (Popovici et al., 2004).

Ces exemples d'environnements virtuels utilisent la simulation qui présente des avantages en terme de formation (Patrick, 1992; Gruau et al., 1998). En effet, elle permet de simuler des situations proches de la réalité, mais sans les contraintes de leur actualisation. De plus, des fonctionnalités supplémentaires à une simple reproduction de la réalité peuvent être ajoutées et ainsi fournir des outils pour la formation (gel de la situation, rejeu, répétition, *etc.*). Bien plus, l'apprenant intervient dans l'environnement informatique et peut observer les conséquences de son action. L'utilisation de cette information rétroactive est à la fois une caractéristique et une

nécessité pour l'apprentissage par l'action (section 2.1.1). Ainsi, ces environnements favorisent l'apprentissage, cependant ils n'intègrent aucune possibilité d'assistance automatique élaborée. La responsabilité de la formation est déléguée aux formateurs, sans que le système lui propose des aides (informations sur le suivi de l'apprenant, analyse de ses difficultés, *etc.*). Même si certains environnements proposent des outils pour le formateur, ce dernier a la charge de leur intégration dans le cursus de l'apprenant. Or, dans la formation professionnelle, le formateur n'est pas souvent un pédagogue, mais un expert du domaine, il risque alors de peu se soucier de l'impact pédagogique de tels outils.

2.4.2 Environnements intégrant les modèles de domaine et/ou de l'apprenant

Un niveau plus élevé de l'utilisation de la réalité virtuelle pour la formation est représenté par les environnements intégrant un modèle du domaine et/ou de l'apprenant. Le modèle du domaine permet notamment de répondre automatiquement à des questions et d'effectuer une tâche à la place de l'apprenant. Comparant le modèle du domaine et le modèle de l'apprenant, le système est capable de détecter des erreurs potentielles. Par contre, ces environnements n'intègrent pas de connaissances pédagogiques qui pourraient les guider pour réagir aux erreurs de l'apprenant.

Mascaret est un modèle permettant la création d'environnements virtuels de formation en situations réalistes et collaboratives (Querrec, 2002). Il permet de décrire l'environnement physique dans lequel l'apprenant évolue ainsi que son environnement social collaboratif. La dimension sociale n'est pas utilisée ici comme un moyen permettant de favoriser l'apprentissage, mais constitue un des éléments de la compétence à acquérir (*cf.* définition de la compétence section 2.1.3). Mascaret s'appuie sur la méthodologie des systèmes multi-agents pour représenter les éléments constituant ces environnements ainsi que leurs interactions. SécuRéVi (Sécurité et Réalité Virtuelle) est une application de Mascaret à la sécurité civile (Querrec et al., 2003b) (*cf.* figure 2.7). Elle est destinée à la formation des officiers sapeurs-pompiers à la gestion opérationnelle et au commandement.

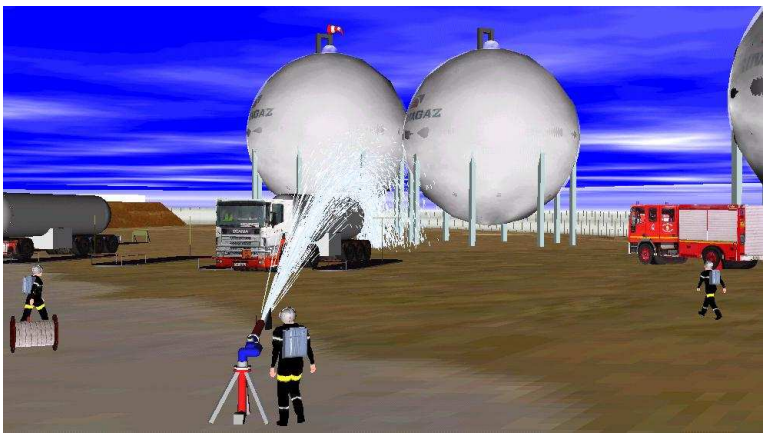


Figure 2.7 – Exemple de scène dans SécuRéVi, d'après (Querrec et al., 2003b).

Les phénomènes mécaniques, chimiques et thermiques (feu, nuage de gaz...) subis ou influencés par l'apprenant sont modélisés sous forme d'un ensemble d'agents en interaction. Les influences entre les éléments constituant le phénomène sont fixés par l'expert du domaine et forment un graphe d'influence régi par une organisation d'agents. L'ensemble de ces organisations forme le modèle du domaine sur l'environnement physique. Ce modèle peut alors renseigner l'utilisateur (formateur ou apprenant) sur les liens de causalité entre phénomènes (le vent détermine la direction de propagation d'un nuage de gaz) et permet également au formateur de modifier le comportement de ces phénomènes (modification du sens du vent par exemple). L'environnement social est constitué des intervenants participant à la tâche collaborative qui doit être réalisée par l'apprenant. Cette tâche est définie par une procédure qui agence les actions à réaliser. Mascaret représente cet environnement social par une organisation multi-agents. Chaque intervenant (agent) y est représenté par le rôle qu'il joue dans l'organisation. Un rôle décrit l'ensemble des actions qui sont de la responsabilité de l'agent. La procédure ordonnance (avant, pendant...) les actions qui sont définies par leurs pré-conditions et post-conditions. L'ensemble de ces organisations forme alors le modèle du domaine sur l'environnement social. Le système fournit toutes les explications sur les actions, procédures et rôles de chaque équipe. Par conséquent, le système est capable de fournir un raisonnement déductif et explicatif sur la procédure. L'apprenant peut prendre ou rendre dynamiquement la main sur un agent qui devient alors son avatar. Ce dernier dispose alors des connaissances sur le domaine et connaît également les actions réalisées. Le modèle Mascaret permet donc de représenter explicitement les comportements et les interactions des agents. L'application SécuRéVi utilise cette capacité dans un but déductif, et non explicatif. Les agents peuvent agir de manière autonome dans l'environnement grâce à ces connaissances. Selon le repère proposé en section 2.3, SécuRéVi, et plus largement les applications dérivant de Mascaret, s'approchent des coordonnées (1,1,1). Si Buche et al. (2004) ont montré que Mascaret représente efficacement le modèle du domaine et une partie du modèle de l'apprenant, il reste à définir le mécanisme permettant d'intégrer un raisonnement pédagogique.

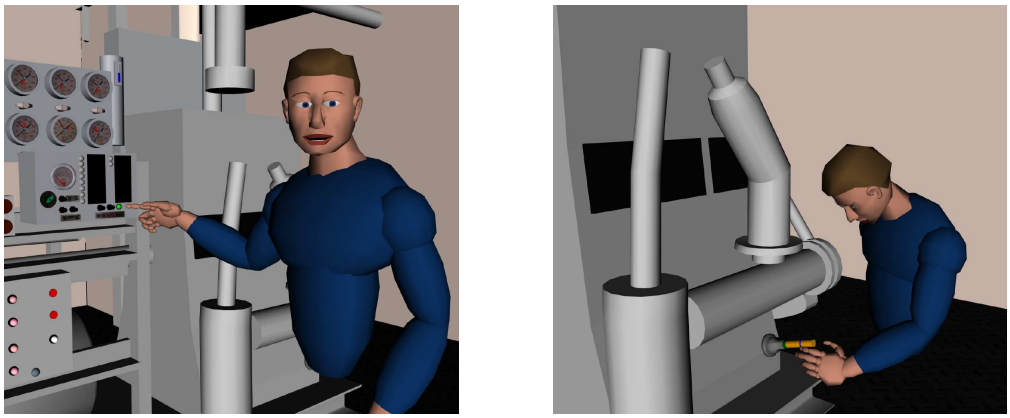


Figure 2.8 – Steve explique et réalise une procédure, d'après (Rickel et Johnson, 1999).

Steve (Soar Training Expert for Virtual Environments) est un agent virtuel animé (Rickel et Johnson, 1999). Il évolue avec l'apprenant dans un environnement conçu pour la formation à des tâches procédurales. STEVE fournit quelques aides pédagogiques automatisées. 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 bateaux (*cf.* figure 2.8). Il existe une version nommée Adele intégrant des capacités d'apprentissage distant (Shaw et al., 1999).

Dans Steve, l'expertise du domaine est modélisée par un ensemble de tâches. L'exemple de la figure 2.9 montre une tâche définie par un ensemble d'actions atomiques ou de sous-tâches (`press-function-test`, *etc.*), 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 associée 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. Selon le repère (A,I,I), l'application Steve s'approche des coordonnées (1,1,1), même si seul le tuteur est autonome et non le reste de l'environnement.

```
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 2.9 – Exemple de tâche dans Steve, d'après (Rickel et Johnson, 1999).

Par ces deux exemples, nous pouvons voir qu'une étape importante dans les environnements virtuels de formation est franchie, car ils permettent d'accéder aux connaissances du domaine et/ou sur l'apprenant. Ces informations peuvent être exploitées pour favoriser la relation d'apprentissage compétence-apprenant. Par exemple, dans le télescope *Hubble Space*, les informations du domaine permettent de colorer d'un vert intense les objets liés à la tâche à effectuer (Loftin et Kenney, 1995). De plus, les connaissances du domaine et celles portant sur l'apprenant peuvent être analysées automatiquement, pour diagnostiquer les erreurs de l'apprenant (Aka et Frasson, 2002) et en déterminer la cause. Plus généralement, elles permettent d'effectuer un suivi automatique de l'apprenant. Le formateur peut alors utiliser ces informations dans la relation pédagogique qui le lie avec l'apprenant. Toutefois, ces systèmes informatiques ne disposent pas de « compétences pédagogiques » leur permettant de prendre l'initiative de médiation au sein du processus enseigner-apprendre. Même si certains proposent une diversité d'outils pour des utilisateurs enseignants, ils seront considérés comme un outil pédagogique à part entière, quand ils pourront relayer l'intervention de l'enseignant et proposer, en toute autonomie, des aides pédagogiques adaptées à la diversité des apprenants.

2.4.3 Environnements intégrant les modèles du domaine, de l'apprenant et pédagogique

L'utilisation d'un modèle pédagogique permet, soit de remplacer le formateur pour une auto-formation (tuteur, système conseiller, aide), soit de lui apporter des aides et lui permettre de gérer plusieurs apprenants en même temps, ce qui correspond aux situations réelles.

Hal (Helpful Agent for Learning) (Lourdeaux, 2001) est un agent pédagogique qui a été spécifié pour optimiser les processus d'apprentissage en environnement virtuel. Cet agent tutoriel permet de tirer parti de l'utilisation de la RV pour la formation dans le cadre de la formation des conducteurs de TGV[7] à l'intervention sur infrastructure ferroviaire à la SNCF[8] (descente sur les voies et manipulation d'appareils de voie). Cette application s'approche des coordonnées (1,1,1) du repère (A,I,I) ; en effet l'environnement est décrit à l'aide d'un système multi-agents. Hal aide les formateurs à construire un discours pédagogique en proposant deux types de « stratégies pédagogiques » adaptatives. Les premières modifient des aspects du scénario (pannes, conditions météorologiques, *etc.*). Les secondes fournissent des aides pour la compréhension de la situation (suggérer où le formé peut trouver la connaissance, expliquer une règle, montrer la conséquence de ses erreurs, *etc.*). Ces stratégies peuvent être mises en œuvre grâce à différentes formes d'assistance pédagogique, gérant différents niveaux de réalisme (enrichissement, visualisation de mécanismes cachés, modélisation de concepts abstraits, *etc.*). Hal s'articule autour de la connaissance du formé (modèle du formé), de l'activité à apprendre (modèle de référence), des connaissances des stratégies d'enseignement (modèle pédagogique) et de la manière de les expliquer (module de diagnostic) (*cf.* figure 2.10). La connaissance du formé (modèle du formé) est constituée des caractéristiques individuelles (niveau, expérience, profil, faculté), d'un historique de ses actions et d'une représentation de ses compétences initiales. La compétence à apprendre (modèle de référence) est composée de plans de tâches et de critères relatifs de préférence. Enfin, le modèle pédagogique structure l'expertise pédagogique. Il est composé de paramètres correspondants au type d'intervention et aux interventions pédagogiques. Le diagnostic compare le modèle du formé et le modèle de référence afin d'évaluer, d'analyser, d'identifier et de quantifier le comportement du formé.

Pour chaque comportement attendu ou problématique de tâche, les assistances pédagogiques et les niveaux de réalisme associés sont décrits (tableau 2.1). Par conséquent, le formateur doit lister de manière exhaustive les comportements attendus (erreurs types) pour chaque connaissance. De plus pour chacune de ces erreurs (colonne 2), il doit préciser la manière dont les stratégies pédagogiques (colonne 3) sont réalisées par des assistances pédagogiques (colonne 4) et cela pour chaque exercice.

Hal commence par identifier le comportement du formé, *i.e.* identifier la connaissance mise en jeu par ses activités. Il quantifie la maîtrise de cette connaissance par l'apprenant (calcul du critère de performance C_p). Ce critère évolue en fonction des actions réalisées par l'apprenant et du temps : tant que l'apprenant exécute une mauvaise action, le critère de performance correspondant au comportement attendu diminue. Lors de l'exercice le formateur choisit une méthode pédagogique (explicative ou active). Pour chacune de ces méthodes,

[7] TGV : Train Grande Vitesse

[8] SNCF : Société Nationale des Chemins de Fer

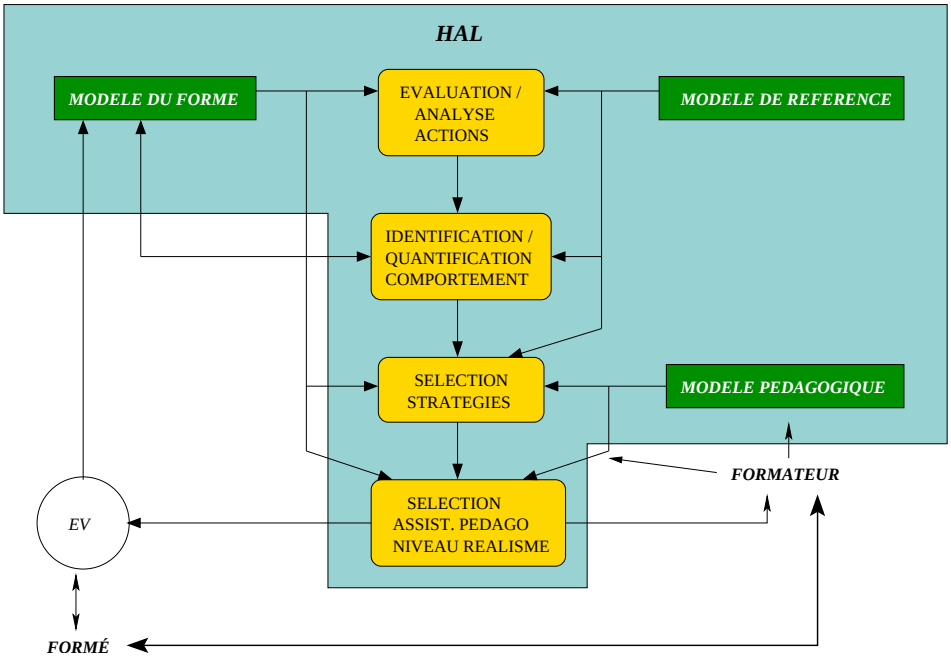


Figure 2.10 – Architecture de Hal (Helpful Agent for Learning), d'après (Lourdeaux, 2001).

Tâche : T				
Connaissances conceptuelles				
	Connaissance 1	Erreur A	Montrer la conséquence	Animation
			Expliquer	Simplification Décentrage
			Suggérer connaissance	Enrichissement
			etc.	
		Erreur B	Expliquer	Animation
			Montrer la conséquence	Décentrage
			Suggérer connaissance	Modification
			etc.	
	Connaissance 2			
Connaissances procédurales				
	Connaissance 1	Erreur A	Expliquer	Enrichissement

Tableau 2.1 – Assistances pédagogiques associées aux stratégies et aux comportements dans Hal, d'après (Lourdeaux, 2001).

Hal propose une base de connaissances sur le choix des stratégies pédagogiques, en fonction du niveau de l'apprenant et du critère de performance (tableau 2.2). Dès que le critère de performance descend sous le seuil précisé dans cette base de connaissance (colonne 1), Hal sélectionne la stratégie suivante. Par exemple dans le cas d'un débutant, après avoir « montré la connaissance », Hal « montre l'écart » puis « explique » et enfin « montre la conséquence » ($S_{SRK1} > S_{SRK2} > S_{SRK3}$).

	Débutant	Intermédiaire	Confirmé
$C_p > S_{SRK1}$	Montrer connaissance	Laisser Faire	Laisser Faire
$S_{SRK1} > C_p > S_{SRK2}$	Montrer l'écart	Suggérer connaissance	Laisser Faire
$S_{SRK2} > C_p > S_{SRK3}$	Expliquer	Suggérer l'écart	Montrer la conséquence
$C_p < S_{SRK3}$	Montrer la conséquence	Expliquer	Expliquer

Tableau 2.2 – Initialisation des stratégies liées au guidage pour la méthode explicative, d'après (Lourdeaux, 2001).

Le formateur n'a accès qu'a deux types de méthode pédagogique et le modèle doit être repensé si le pédagogue désire intégrer d'autres méthodes, ou stratégies pédagogiques. De plus l'enchaînement de stratégies pédagogiques est figé et non accessible au formateur.

Les exemples précédents montrent qu'actuellement les modèles pédagogiques dans les EVF restent des modèles très *ad hoc* et ne sont pas forcément fondés sur des concepts issus de la pédagogie mais plutôt sur des expertises métiers.

2.5 Bilan

L'objectif de ce chapitre était d'évaluer l'intégration des ITS au sein de la réalité virtuelle dans une perspective d'apprentissage de compétences. Nous avons présenté la réalité virtuelle comme un domaine prometteur pour concevoir des situations de formation à fort potentiel de transfert. Complémentairement, nous avons montré que les ITS permettent d'individualiser la relation apprenant-compétence (apprentissage) et de diversifier, au sein des EIAH, les relations formateur-apprenant (pédagogies). L'intégration des ITS au cœur de la situation de formation permet alors d'assister le formateur et d'aider l'apprenant. Plus précisément, l'utilisation de la réalité virtuelle permet la variabilité des conditions d'apprentissage et l'intégration des ITS l'adaptation à l'apprenant. Ces deux facteurs sont favorables, sur un plan théorique, au transfert de compétences acquises.

Nous avons analysé quelques utilisations des ITS au sein des environnements virtuels de formation. La plupart n'incorporent que la représentation des connaissances sur le domaine. Pour les systèmes qui proposent un module de diagnostic, ils ne fournissent que très rarement un mécanisme d'assistance pédagogique. Nous considérons que Hal est le système le plus abouti. Néanmoins, le formateur doit lister les erreurs et préciser les stratégies pédagogiques pour chaque exercice.

Nos travaux de recherche s'orientent vers la définition d'un système tutoriel intelligent au sein d'une application de réalité virtuelle. Ce système devrait proposer un modèle pédagogique

modulable, *i.e.* permettant d'intégrer, de modifier ou de supprimer des concepts pédagogiques facilement. De plus, il devrait être générique dans le sens où il intégrerait une caractérisation des erreurs et une définition du modèle pédagogique utilisable indépendamment de l'exercice à réaliser. Les connaissances du modèle pédagogique et les expériences passées pourront être utilisées pour proposer automatiquement les interventions appropriées, en considérant l'apprenant et le contexte de simulation. Contrairement à une formation classique où les formateurs ne sont pas toujours des pédagogues ou des spécialistes de la réalité virtuelle (Lourdeaux, 2001), ce système permettrait de bénéficier d'une réflexion pédagogique simulée et d'utiliser toutes les potentialités de la réalité virtuelle. Cependant, nous pensons que la décision finale d'intervention devrait être du ressort du formateur, et non pas appliquée automatiquement, puisqu'il existera toujours des éléments qu'un programme informatique ne peut prendre en compte : éléments de l'environnement, caractéristiques liées au groupe d'apprenants, vécu des apprenants, niveau de stress, interactions directes entre humains, *etc.*

L'article (Buche et al., 2005) présente l'essentiel de l'étude proposée dans ce chapitre.

Chapitre 3

Environnement virtuel de formation informé

Connais-toi toi-même.

Socrate

Dans ce chapitre, nous proposons une modélisation du système tutoriel intelligent. La représentation des connaissances portant sur l'environnement simulé et sur le travail à réaliser (procédural et collaboratif) offre la possibilité de manipuler les informations nécessaires à la prise de décision pédagogique. Un modèle d'entité virtuelle, un modèle organisationnel et les diagrammes d'activités UML nous permettent de réifier ces informations. Ces connaissances sont utilisées par l'ITS au travers de plusieurs modèles (domaine, apprenant, erreur, interface, formateur et pédagogique) implémentés sous la forme d'agents. Une analyse des rôles de chacun définit leur base de connaissances et leurs comportements participants à la prise de décision. Ils analysent les activités de l'apprenant dans l'univers 3D et caractérisent, de manière générique, les erreurs effectuées. Les interactions entre agents produisent un ensemble de connaissances, appelé situation pédagogique, considéré pour simuler un raisonnement pédagogique. Elle fournit les connaissances permettant d'identifier des situations particulières et extrait de l'environnement des ressources privilégiées pour déclencher des assistances pédagogiques.

Le chapitre précédent a souligné les apports de la RV et des ITS en terme de formation. Il a également montré leur complémentarité pour l'apprentissage de compétences. Nous proposons alors l'utilisation conjointe de ces deux domaines.

Notre objectif est d'intégrer un ITS à des simulations basées sur les techniques de RV. Pour faciliter cette intégration, nous devons définir les échanges d'information entre la RV et l'ITS. La simulation doit notamment fournir des informations à l'ITS qui doivent pouvoir être manipulées pour la prise de décision pédagogique. Il s'agit alors de représenter (réifier) ces connaissances pour informer l'ITS. Ainsi, la simulation prend place au sein d'un *environnement virtuel informé*.

Nous pouvons alors nous interroger afin de définir quelles sont les connaissances à représenter ? Elles portent sur les éléments présents dans la simulation et surtout sur la

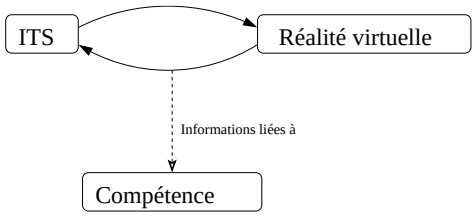


Figure 3.1 – Les simulations de RV doivent fournir des informations à l'ITS.

compétence à acquérir (*cf.* figure 3.1). Dans le cadre du projet Mascaret [1], cette compétence porte sur le travail procédural et collaboratif.

La représentation des connaissances doit être suffisamment abstraite pour offrir à l'ITS la possibilité de raisonner indépendamment de l'environnement simulé et de l'exercice à réaliser. Nous nous appuyons sur l'architecture présentée en figure 3.2 et distinguons les modèles suivants :

- ▷ Les deux modèles représentants les connaissances sur l'environnement virtuel. Ils constituent l'environnement virtuel informé :
  - Veha permet de représenter les connaissances sur les entités, *i.e.* les composantes de l'environnement simulé.
  - Le `Modèle organisationnel` permet de représenter les connaissances organisationnelles, structurant le travail à réaliser et définissant les équipes, les procédures et les actions à effectuer.
- ▷ L'ITS qui manipule les connaissances de l'environnement virtuel informé, *i.e.* les deux types de connaissances précédentes. L'ITS implémente les différents modèles que nous avons présentés dans le chapitre précédent et se place dans le projet Mascaret.
- ▷ La structure d'`Agent`[2] utilisée par Mascaret, et plus particulièrement par notre ITS. Elle contient une base de connaissances et des comportements définis à partir de cette base.

L'ensemble forme un environnement virtuel de formation informé.

L'objectif de ce chapitre est de montrer l'ensemble de cette architecture. Dans la section 3.1, nous présentons la modélisation de l'environnement virtuel informé. Nous précisons comment nous représentons les connaissances portant sur l'environnement simulé (modèle Veha) et sur le travail à réaliser (`Modèle organisationnel`). A partir de cet environnement, nous dégageons des connaissances liées à la formation. Nous proposons le système tutoriel intelligent (`ITS`) en section 3.2. Ce dernier est modélisé par un système multi-agents. Enfin, ces connaissances nous permettent d'extraire un ensemble d'informations considérées comme pertinentes, qui peut être utilisé pour la prise de décision pédagogique. Cet ensemble, que nous définissons en section 3.3, est appelé la « situation pédagogique ».

[1] Il est important de relever que le projet Mascaret dans lequel s'inscrit cette thèse est à distinguer des « modèles de Mascaret » présentés au chapitre présent. En effet, il s'agissait d'une version antérieure proposée par Querrec (2002) qui désignait un ensemble de modèles. Mascaret a évolué et désigne aujourd'hui un projet et non un modèle unique.

[2] Nous distinguons les agents, qui possèdent des comportements autonomes, des entités contenues dans le modèle Veha, qui ne possèdent que des opérations.

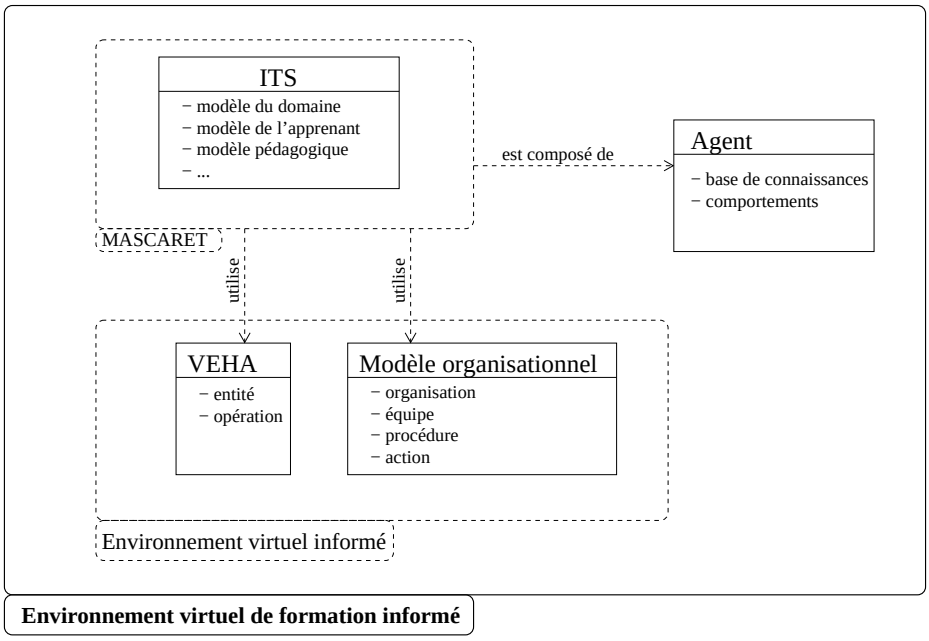


Figure 3.2 – Modèles de l'environnement virtuel de formation informé.

3.1 Environnement virtuel informé

La représentation de l'environnement virtuel que nous proposons réifie les éléments manipulés par un programme informatique pour réaliser un raisonnement. L'environnement virtuel contient alors des informations exploitables, il devient un « environnement virtuel informé ».

Dans la section 3.1.1, nous montrons comment nous simulons et représentons l'environnement virtuel. Nous définissons notamment le modèle d'entité virtuelle et nous nous intéressons plus particulièrement à la description des comportements.

L'environnement virtuel étant représenté, il s'agit de réifier les connaissances liées au travail à effectuer par l'apprenant. Dans notre cas, il s'agit de réaliser une procédure en collaborant avec les autres membres d'une équipe. Dans la section 3.1.2, nous expliquons comment nous représentons les aspects organisationnels qui structurent les interactions au sein des équipes. Nous montrons également comment nous représentons les procédures en nous intéressant plus particulièrement à la notion d'action.

3.1.1 Représenter l'environnement virtuel

Notre équipe scientifique développe une librairie informatique appelée *Virtual Environment supporting Human Activity* (Veha). Elle définit les composantes des entités dans un monde virtuel (section 3.1.1 - A) et précise comment on peut leur donner un comportement (section 3.1.1 - B). Ce modèle s'inspire du méta-modèle UML (Rumbaugh et al., 1999) et y ajoute quelques notions issues de l'approche multi-agents.

3.1.1 - A Entité virtuelle

Nous définissons tout d'abord la notion de classe comme une abstraction qui décrit un ensemble potentiellement infini d'objets individuels caractérisés par des propriétés similaires. Chaque objet membre de l'ensemble est appelé *entité*. La notion de classe nous permet de raisonner sur les caractéristiques d'une entité indépendamment de ses particularités, *i.e.* qu'elle rend possible la représentation des connaissances sur les variables qu'elle contient et sur les comportements dont elle dispose. En d'autres termes, il est possible de raisonner sur un ensemble d'entités de même classe sans considérer une instance particulière.

Notion de classe

Une classe (`Class`) est la représentation d'un élément générique de l'environnement (*cf.* figure 3.3). Cet élément a des caractéristiques (`Feature`) qui lui sont propre et observables d'un point de vue externe. Cette représentation offre la possibilité de manipuler des connaissances liées à un type d'entité. Nous pouvons notamment avoir accès aux opérations qu'il peut exécuter (`Operation`), à ses propriétés (`Attribute`) et à ses états (`State`).

Exemple : L'environnement décrit un objet « piéton » qui possède un comportement de déplacement, a une vitesse de déplacement et peut être dans un état mobile ou immobile.
⇨ *VEHA crée une instance de la classe `Class` piéton, lui donne une instance de la classe `Operation` opération de déplacement, une propriété vitesse de déplacement (instance de la classe `Attribute`) et deux états : mobile et immobile (instances de la classe `State`).*

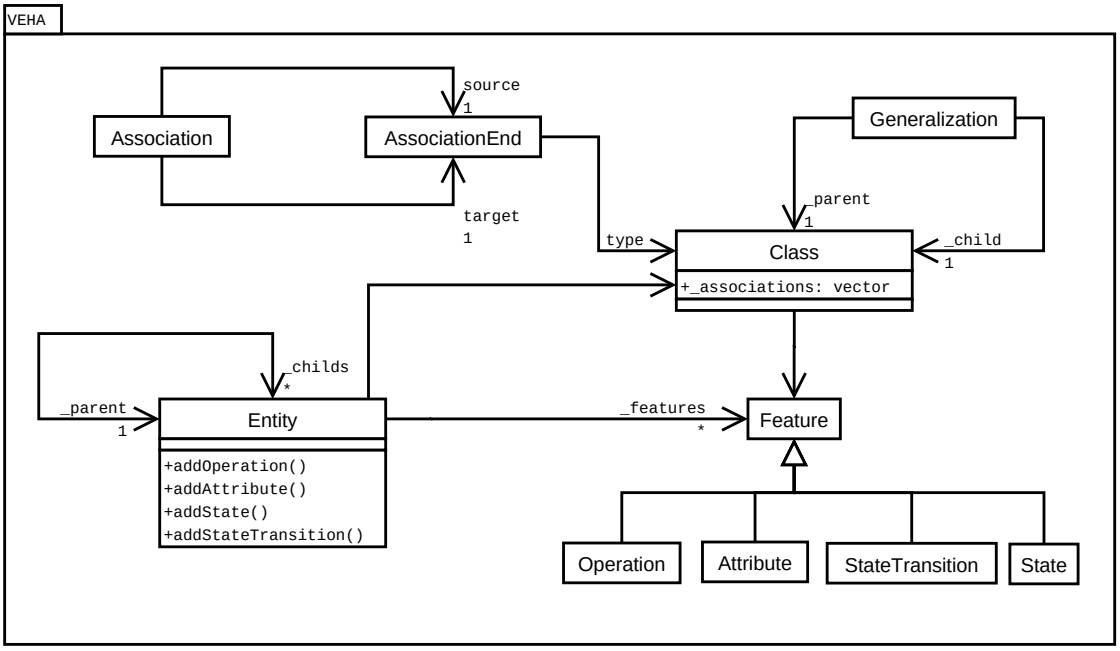


Figure 3.3 – Notion de classe dans Veha.

Les classes sont liées par des associations (`Association`) qui expriment une connexion sémantique bidirectionnelle entre deux classes.

Exemple : La classe « personne » est associée à la classe « entreprise » par l'association « travail »

Les classes sont hiérarchisées par la notion de spécialisation et de généralisation (`Generalization`) du paradigme objet[3] (Meyer, 2000).

Notion d'entité

Pour créer un élément particulier (une instance de classe), nous définissons la notion d'entité (`Entity`).

Exemple : Un piéton particulier s'appelle Robert et a pour caractéristique une vitesse de déplacement dont la valeur est 10km/h.
⇨ VEHA crée une instance particulière de la classe piéton nommée Robert et la valeur de la propriété vitesse de déplacement est de 10km/h.

Nous venons de définir la notion d'entité, à présent il s'agit de préciser les mécanismes permettant de mettre à sa disposition un comportement.
✎ Les attributs sont des objets indéfinis (les types classiques sont disponibles : double, string, *etc.*) et les états sont régis par une machine à états. Ces deux concepts sont classiques et ne nécessitent pas de précision supplémentaire, contrairement aux opérations (`Operation`) qui constituent un point clef dans nos travaux.

3.1.1 - B Opération

Une entité peut exécuter des opérations (`Operation`). Elle correspond au niveau le plus fin de description (opération atomique) du comportement d'une entité (*cf.* figure 3.4).

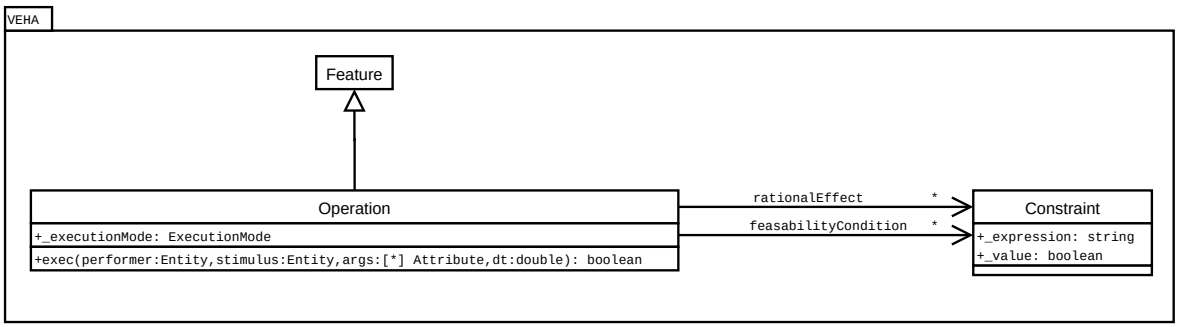


Figure 3.4 – Une opération dans Veha.

Les opérations disposent de pré-conditions (`feasabilityCondition`) sous la forme d'expressions booléennes représentant des sous-états observables du monde virtuel. Une opération

[3] Ce mécanisme permet de créer une nouvelle classe en partant d'une classe existante. La nouvelle classe conservera les caractéristiques de la classe d'origine, tout en pouvant en ajouter.

possède également une post-condition (`rationalEffect`) qui est une expression booléenne représentant le sous-état du monde (théorique) après l'exécution de l'opération. Pour exécuter une opération, il faut fournir l'entité qui la sollicite (`performer`)[4] et une liste de paramètres (`args`). L'entité ayant provoquée l'exécution de l'opération chez le `performer` est appelée `stimulus`.

Une entité contient des opérations qui correspondent à des comportements atomiques génériques. Ainsi, nous pouvons définir des opérations « pédagogiques » (*e.g.* faire clignoter, encadrer, déformer, *etc.*) qui peuvent facilement être ajoutées aux entités d'une application existante. Ce principe nous sert de base pour que notre ITS agisse dans l'environnement et réalise des assistances pédagogiques.

3.1.1 - C Bilan

Les modèles proposés sont une structure d'accueil de description de comportements simulables d'une entité virtuelle. Bien plus, ils permettent la représentation des connaissances la concernant, ce qui est un point essentiel concernant l'usage de l'environnement pour la formation. L'ITS peut alors manipuler les connaissances liées aux entités virtuelles dans un but pédagogique.

Au delà de l'environnement, l'ITS doit également manipuler les connaissances portant sur la compétence à acquérir. Dans cette optique, la section suivante précise les modèles permettant de simuler et de représenter les connaissances liées au travail à effectuer. Dans notre cas, il s'agit du travail procédural et collaboratif.

3.1.2 Représenter le travail à effectuer

Le travail à effectuer est un travail collaboratif. Le modèle utilisé doit alors représenter les rôles de chaque participant au sein d'organisations. Les participants évoluent dans le cadre d'une équipe en fonction des compétences qu'ils peuvent apporter et en respectant les contraintes liées à leur collaboration avec les autres intervenants. Ces contraintes sont fixées par des procédures qui agencent les relations entre les actions à effectuer.

Dans cette section, nous présentons un modèle de structure organisationnelle (section 3.1.2 - A) pour le travail d'équipe (section 3.1.2 - B). Nous intégrons la notion de procédure (section 3.1.2 - C) et nous précisons la notion d'action (section 3.1.2 - D).

Les modèles qui suivent s'inspirent des travaux de Querrec (2002) qui proposent un modèle utilisant des systèmes multi-agents[5] pour simuler les environnements collaboratifs. Il s'agit d'un modèle permettant de structurer les interactions entre les différents intervenants qui sont ici des agents.

[4] En effet, l'opération peut nécessiter la prise en compte de certains attributs du performer.

[5] Ce modèle considère que tous les éléments de l'environnement (physique et social) sont des entités autonomes appelées « agents ».

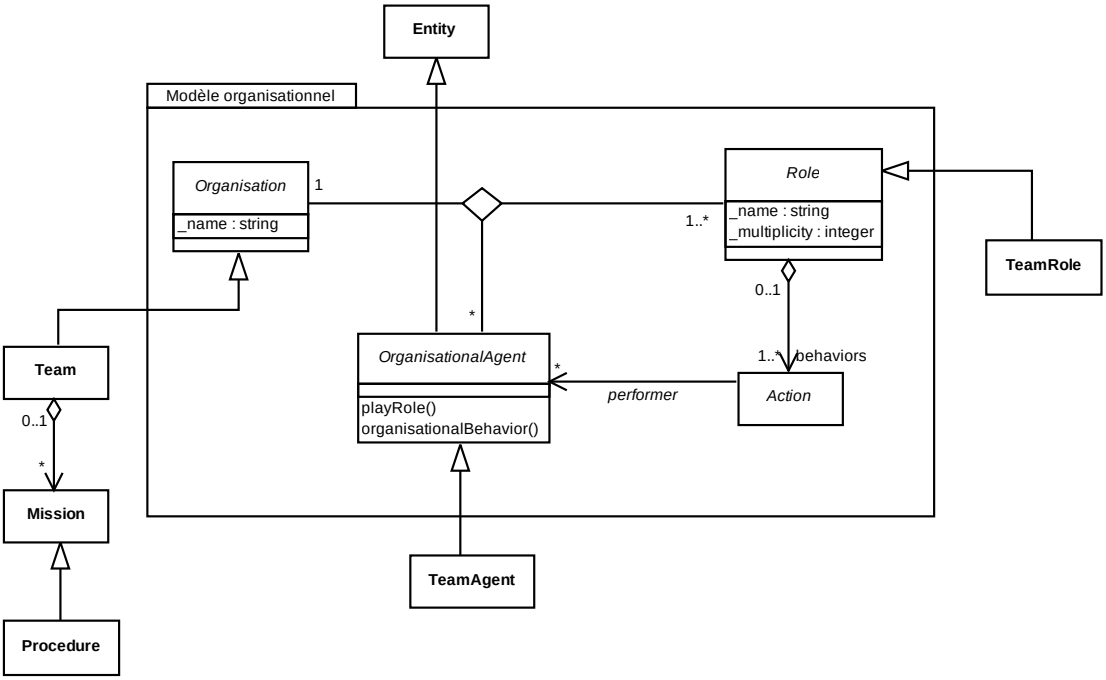


Figure 3.5 – Travail procédural en équipe dans Mascaret d'après Querrec (2002).

3.1.2 - A Organisation

Le modèle organisationnel est présenté par le diagramme de classes de la figure 3.5. Il s'articule autour de quatre concepts : l'organisation, le rôle, l'agent et l'action. Il s'agit d'un modèle générique (toutes les classes sont abstraites).

L'`Organisation` structure les interactions, les agents peuvent 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 (`Action`) que doit adopter l'agent jouant le rôle.

3.1.2 - B Équipe

L'équipe (`Team`) est une organisation particulière dont les membres sont des `TeamAgent`. Ces agents ont un comportement collaboratif qui prend en compte les autres membres de l'équipe en fonction des rôles qu'ils jouent.

Une équipe doit remplir plusieurs missions (`Mission`) qui peuvent s'exécuter en parallèle ou en séquence. Les missions représentent les buts communs des membres de l'équipe et lorsqu'une mission est sélectionnée, chaque intervenant œuvre pour atteindre ces buts. 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 particulier où cette coordination est déjà prévue et écrite dans des plans d'actions que l'on désigne par le terme de procédure.

3.1.2 - C Procédure

Les procédures (`Procedure`) sont des missions représentées par des ensembles de contraintes décrivant l'agencement des actions (`Action`) effectuées par les agents jouant les rôles de l'équipe. Les procédures doivent permettre de définir ces contraintes, *i.e.* les conditions de liaison entre actions, leur ordonnancement ainsi que les responsabilités. Il existe plusieurs modèles permettant de représenter ces procédures. Querrec (2002) a notamment effectué deux propositions, néanmoins ces modèles présentent leurs limitations. En effet, le plus abouti utilise une logique basée sur deux opérateurs (parallèle / séquence) et n'est pas assez riche pour exprimer tout type de procédure. Afin de ne pas proposer un modèle supplémentaire, nous utilisons les diagrammes d'activités UML qui sont définis et normalisés par l'OMG[6].

Diagramme d'activités

Les diagrammes d'activités sont utilisés pour montrer la façon dont les *workflows*[7] ou les traitements sont créés, comment ils démarrent, les nombreux chemins décisionnels qu'ils peuvent emprunter, ainsi que les opérations effectuées en parallèle.

Le modèle UML d'un diagramme d'activités est représenté en figure 3.6. Un diagramme d'activités (`ActivityGraph`) représente un cas particulier d'une machine à états (`StateMachine`). Une activité peut être représentée par un ensemble de sous-états (`CompositeState`) ou par un état unitaire (`SimpleState`). Dans les diagrammes d'activités, un état unitaire est une action qui peut être effectuée (`ActionState`).

Les diagrammes d'activités décrivent l'organisation des activités. Une transition (`Transition`) contient un nœud (`StateVertex`) entrant et un nœud sortant. Un nœud peut être un état (`State`) ou une liaison (`PeudoState`) typée (`kind`). Une liaison peut être une séquence (liaison PUIS), amener à une alternative (liaison OU) ou à du parallélisme (liaison ET). Une garde (`Guard`) indique que la transition est conditionnelle.

Les diagrammes d'activités peuvent être agencés en zones, séparées par des lignes verticales appelées couloirs d'activités (`Partition`). Les couloirs représentent les responsabilités. Dans l'exemple du diagramme représenté en figure 3.7, deux rôles sont utilisés : le chef-pompier et le sous-chef. L'action « ordonner le ralliement » débute la procédure et est à la charge du chef-pompier. Une fois que cette dernière est terminée, c'est au sous-chef d'« aller au point de ralliement » (le chef-pompier est déjà au point de ralliement). Ensuite, le chef-pompier doit « aller chercher le tuyau » et le sous-chef doit « aller chercher le dévidoir ». Notons la liaison OU à la suite de l'action « aller chercher le tuyau ». Si le produit est un hydrocarbure, le chef-pompier doit « prendre la lance à mousse ». Sinon, il « prend la lance à eau ».

[6] OMG : Object Management Group, groupement international définissant des standards relatifs à la programmation objet. `http ://www.omg.org/` (accédé le 19/09/2005).

[7] On appelle *workflow* la modélisation et la gestion informatique de l'ensemble des tâches à accomplir et des différents acteurs impliqués dans la réalisation d'un processus métier.

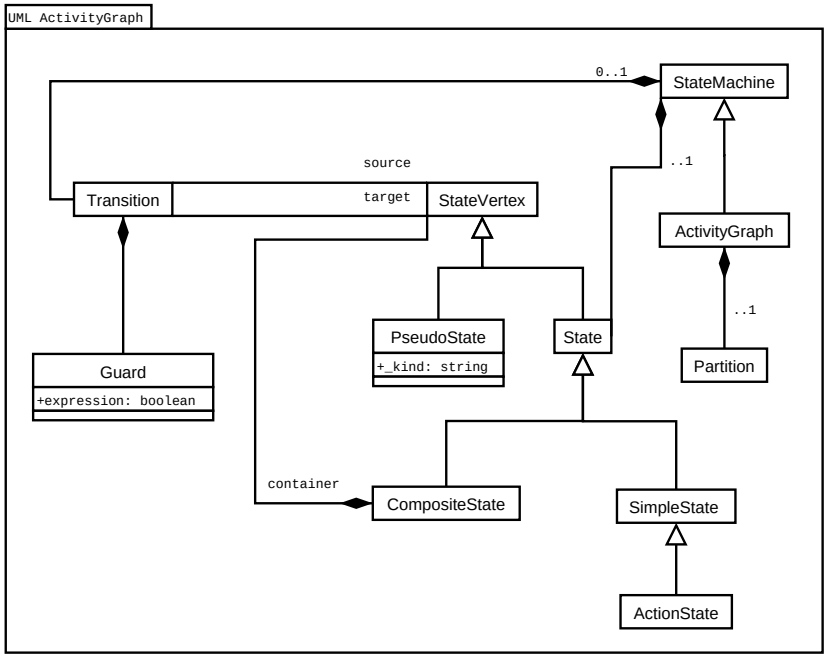


Figure 3.6 – Diag. de classes du modèle de diagramme d'activités UML.

Du diagramme d'activités à la procédure

Pour effectuer la liaison entre notre modèle de travail procédural en équipe décrit en figure 3.5 et le diagramme d'activités UML de la figure 3.6, nous définissons les relations suivantes :

1. La procédure se base sur un diagramme d'activités
   (liaison `Procedure` ⟶ `ActivityGraph`).
2. Une action représente un état unitaire du diagramme d'activités
   (liaison `Action` ⟶ `ActionState`).
3. Un rôle correspond à une partition du diagramme d'activités
   (liaison `Role` ⟶ `Partition`).

Nous obtenons une représentation des connaissances liées aux procédures. La notion d'action est précisée dans la section suivante.

3.1.2 - D Action

Cette section a pour objectif de proposer une représentation de la notion d'action. Ce concept est primordial puisque l'action est l'objet de la formation dans notre cas d'étude. Afin de mieux comprendre la notion d'action, nous nous intéressons aux études effectuées à ce sujet en psychologie cognitive. Cette analyse sert de support à l'élaboration de notre proposition.

Le contenu des connaissances sur l'action est un domaine peu étudié jusqu'à présent (Richard, 1998). Les travaux qui ont posé directement le problème des connaissances sur

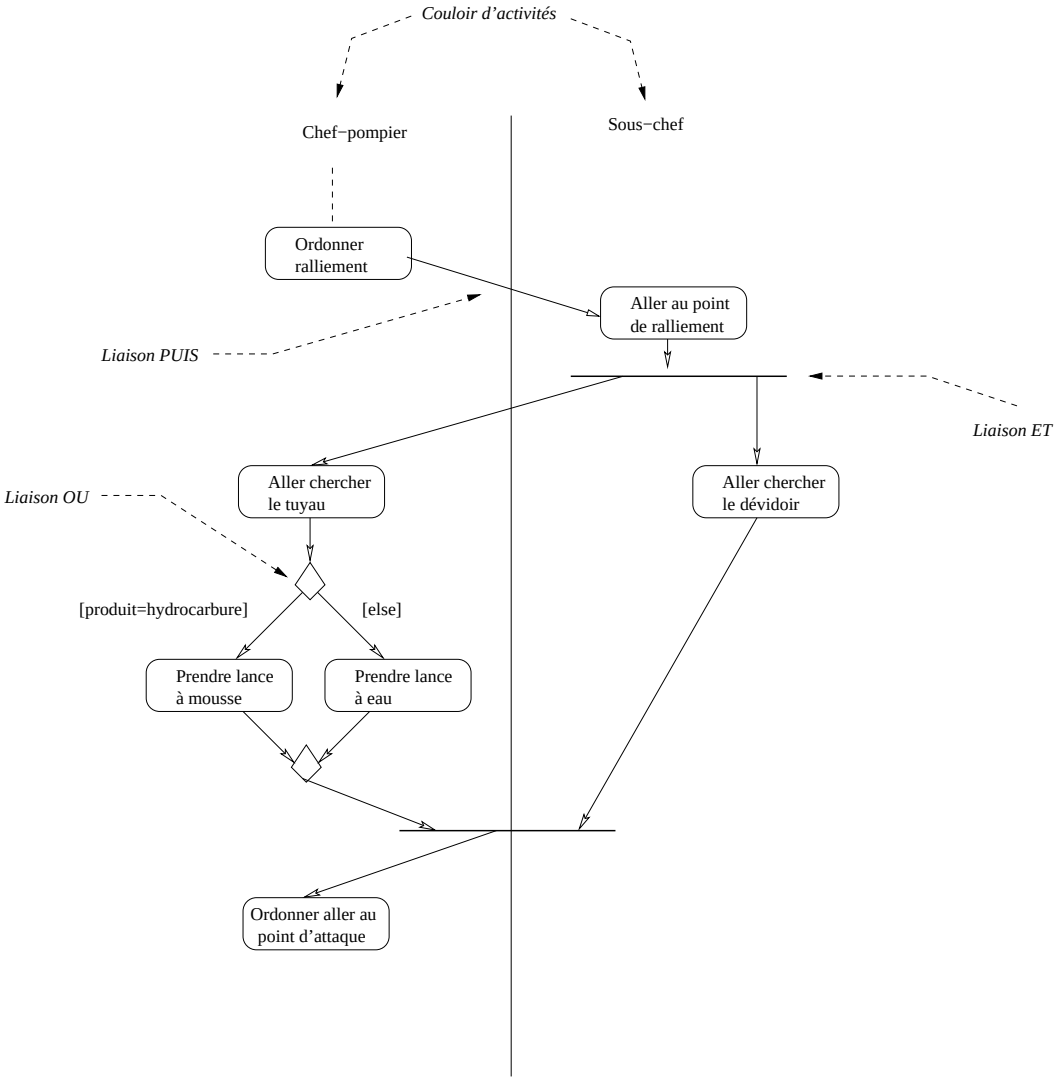


FIGURE 3.7 – Exemple de diagramme d'activités UML.

l'action portent essentiellement sur la planification (Van Lehn et Brown, 1980). C'est en psychologie cognitive que les études sont les plus récentes. Pour Richard (1998), une action peut être envisagée sous un double aspect. D'une part, une action a une composante déclarative : elle exprime un changement de l'état du monde. D'autre part, elle possède également une composante procédurale : elle décrit un processus qui est à la fois un déroulement et un mode de réalisation du résultat. Par conséquent, la notion d'action est décrit selon :

1. *Son résultat* : l'état auquel on aboutit.
2. *Son déroulement* : c'est l'exécution de l'action, son mode de réalisation.

Ces deux aspects sont essentiels puisque le résultat permet de choisir l'action adéquate pour un objectif donné et que le déroulement indique son exécution. Richard ajoute à cette définition un troisième aspect : les pré-requis. Ils définissent quelles conditions doivent être réalisées pour que l'action puisse être exécutée.

Nous pouvons retenir de l'étude précédente la nécessité de différencier l'exécution du résultat d'une action. Pour cela nous effectuons la distinction entre la notion d'action (`Action`) qui représente la décision (liée au résultat) et la notion d'opération (`Operation`) qui représente son exécution (liée au déroulement). Bien que ces deux notions sont à distinguer, elles n'en sont pas moins liées. Par conséquent, chaque action pointe sur une opération exécutable (*cf.* figure 3.8). Ainsi, la notion d'action permet de représenter les connaissances sur les opérations sans implémenter leur exécution.

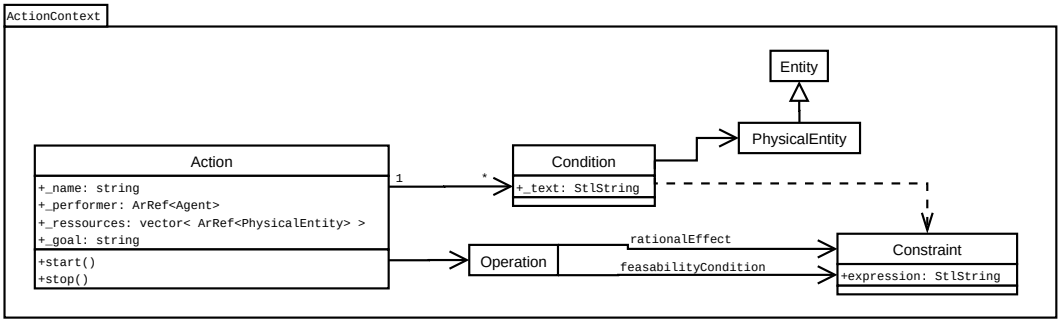


Figure 3.8 – Diag. de classes UML de la notion d'action.

Une action est définie par son résultat (`goal`) et fait l'objet de conditions d'utilisation (`Condition`). Une condition pointe vers une contrainte liée à l'exécution et possède un texte explicatif. Elle peut être associée à des entités physiques de l'environnement, *i.e.* qui ont une représentation physique (`PhysicalEntity`). De plus, une action porte sur des ressources qui sont des entités physiques.

Exemple : L'action « Arroser Camion » porte sur l'opération « Arroser ». La condition d'utilisation de cette action est « posséder une lance » qui est associée à l'entité « Lance » et qui utilise la contrainte de faisabilité de l'opération « possedeLance==true ».

Les connaissances sur l'environnement et sur le travail à réaliser sont représentées. Notre système tutoriel intelligent peut donc les manipuler pour construire ses propres connaissances et simuler un raisonnement pédagogique.

3.2 Proposition d'un système tutoriel intelligent

Dans cette section, nous montrons comment nous représentons les connaissances qui servent de base au raisonnement pédagogique de notre système. Ces connaissances sont contenues dans des modèles inspirés de ceux qui sont utilisés classiquement dans un ITS (Buche et al., 2004). Comme Py (1998), nous considérons les erreurs comme des informations cruciales. Par conséquent nous avons décidé d'introduire un modèle supplémentaire (par rapport aux modèles classiques des ITS) appelé « modèle des erreurs ». En effet, la prise en compte des erreurs a déjà fait l'objet d'études, mais n'a pas été envisagée jusqu'à lors en terme de module indépendant. De plus, nous ajoutons un « modèle du formateur » qui définit les connaissances sur l'exercice à réaliser précisées par le formateur.

Ainsi, nous proposons les modèles suivants (*cf.* figure 3.9) :

1. le *modèle du domaine*, représentant la connaissance, utile pédagogiquement, de l'expert sur le domaine ;
2. le *modèle des erreurs*, proposant de caractériser les erreurs et contenant une base de connaissances sur les erreurs classiques ;
3. le *modèle de l'apprenant*, permettant d'établir l'état de ses connaissances à un instant donné ainsi que son profil ;
4. le *modèle d'interface*, permettant l'échange d'informations entre le système et l'utilisateur ;
5. le *modèle du formateur*, permettant de préciser les instructions du formateur liées à l'exercice à effectuer et fournissant au formateur des possibilités privilégiées en terme de pédagogie ;
6. le *modèle pédagogique*, prenant les décisions d'enseignement.

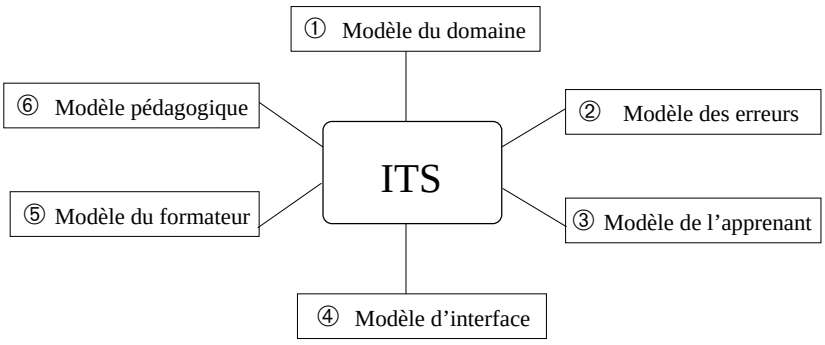


Figure 3.9 – Les six modèles de l'ITS.

Pour que chaque modèle puisse partager ses informations et effectuer ses analyses en toute autonomie (indépendamment de la simulation et des autres modèles), une entité autonome (appelée agent) est associée à chaque modèle. Ainsi, nous considérons les associations suivantes :

- ▷ modèle du domaine — `ExpertAgent`
- ▷ modèle des erreurs — `ErrorAgent`
- ▷ modèle de l'apprenant — `LearnerAgent`
- ▷ modèle d'interface — `InterfaceAgent`
- ▷ modèle du formateur — `TeacherAgent`[8]
- ▷ modèle pédagogique — `PedagogicalAgent`

Chaque agent est en interaction avec un ou plusieurs agents, ce qui lui permet de mettre à jour ses connaissances et de prendre en compte l'analyse effectuée par les autres modèles. Pour que cet échange soit possible, chacun de ces modèles se base sur une structure commune (`AgentPackage` *cf.* figure 3.10) plus abstraite d'agent (`Agent`), de comportements (`Behavior`) et de connaissances (`KB`). Ce modèle abstrait nécessiterait certainement une étude approfondie qui nous éloignerait cependant de nos objectifs. Par conséquent, nous nous en contentons, car il est suffisant pour définir une base commune aux agents contenus dans notre ITS.

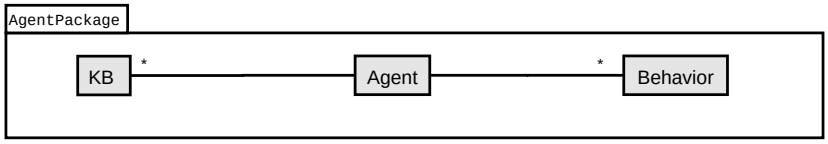


Figure 3.10 – Diag. de classe UML représentant le modèle abstrait d'agent.

Cette section présente notre proposition d'ITS en suivant le plan qui suit. Dans un premier temps, nous présentons une vision globale de l'ITS en montrant les liens fonctionnels reliant les modèles utilisés. Ces relations permettent de mettre à jour les connaissances de chacun et définissent un processus amenant à une décision pédagogique (section 3.2.1). Ensuite, nous détaillons les modèles du domaine, de l'apprenant, des erreurs, d'interface et du formateur (section 3.2.2 à 3.2.6). Ils fournissent les connaissances qui sont utilisées pour effectuer un raisonnement pédagogique, rôle endossé par le modèle pédagogique (section 3.2.7). Pour chaque modèle, nous précisons :

- ▷ son rôle dans la *situation d'apprentissage*[9],
- ▷ les composantes de l'agent qui lui sont associées, *i.e.* :
  - la base de connaissances représentant les informations contenues dans le modèle,
  - les comportements de l'agent.

[8] Le formateur humain intervient en utilisant le `TeacherAgent`.

[9] Nous rappelons que la situation d'apprentissage a été définie au chapitre 2 comme un triangle constitué de trois pôles : l'apprenant, la compétence à acquérir et le formateur.

3.2.1 Interaction entre les modèles

Les modèles interagissent et s'échangent des données qui peuvent être extraites de la simulation ou déduites d'un raisonnement interne au modèle. Nous proposons un processus en cinq étapes (*cf.* figure 3.11) :

Étape 1. *Observer* (`InterfaceAgent`) :
Utilisant le modèle d'interface, le système analyse les activités de l'apprenant. Les éléments pertinents pour la formation sont fournis au modèle de l'apprenant. Ces informations concernent (*cf.* section 3.2.5) :

- ▷ les actions de l'apprenant,
- ▷ les éléments observables par l'apprenant,
- ▷ les déplacements de l'apprenant.

Étape 2. *Détecter et Identifier une erreur* (`LearnerAgent`, `ExpertAgent` et `ErrorAgent`) :
Le système analyse les actions de l'apprenant (modèle de l'apprenant) et les compare aux actions à effectuer (modèle du domaine). Cette confrontation permet de détecter une éventuelle erreur. Si une erreur a été détectée, un mécanisme d'identification de l'erreur est mis en place (en utilisant le modèle des erreurs). De plus, les règles d'usage général du modèle du domaine sont vérifiées (règles indépendantes de l'ordonnancement de la procédure).

Étape 3. *Proposer des assistances pédagogiques* (`PedagogicalAgent`) :
Utilisant le modèle de l'apprenant (caractéristiques, activités, erreur, *etc.*) et le modèle du domaine (connaissances sur les structures organisationnelles), un mécanisme simulant un raisonnement pédagogique préconise les assistances pédagogiques adaptées à la situation.
✎ Notons que cette étape n'est pas conditionnelle, elle intervient même si aucune erreur n'est détectée.

Étape 4. *Choisir une assistance pédagogique* (`TeacherAgent`) :
Le formateur peut sélectionner une assistance pédagogique parmi celles qui ont été préconisées.

Étape 5. *Représenter l'assistance pédagogique* (`InterfaceAgent`) :
L'assistance pédagogique sélectionnée est représentée dans l'environnement virtuel.

Les sections qui suivent présentent les différents modèles individuellement. Un modèle est constitué d'une base de connaissances. Il contient également des méthodes permettant de manipuler ces connaissances, *i.e.* un agent qui possède des comportements.

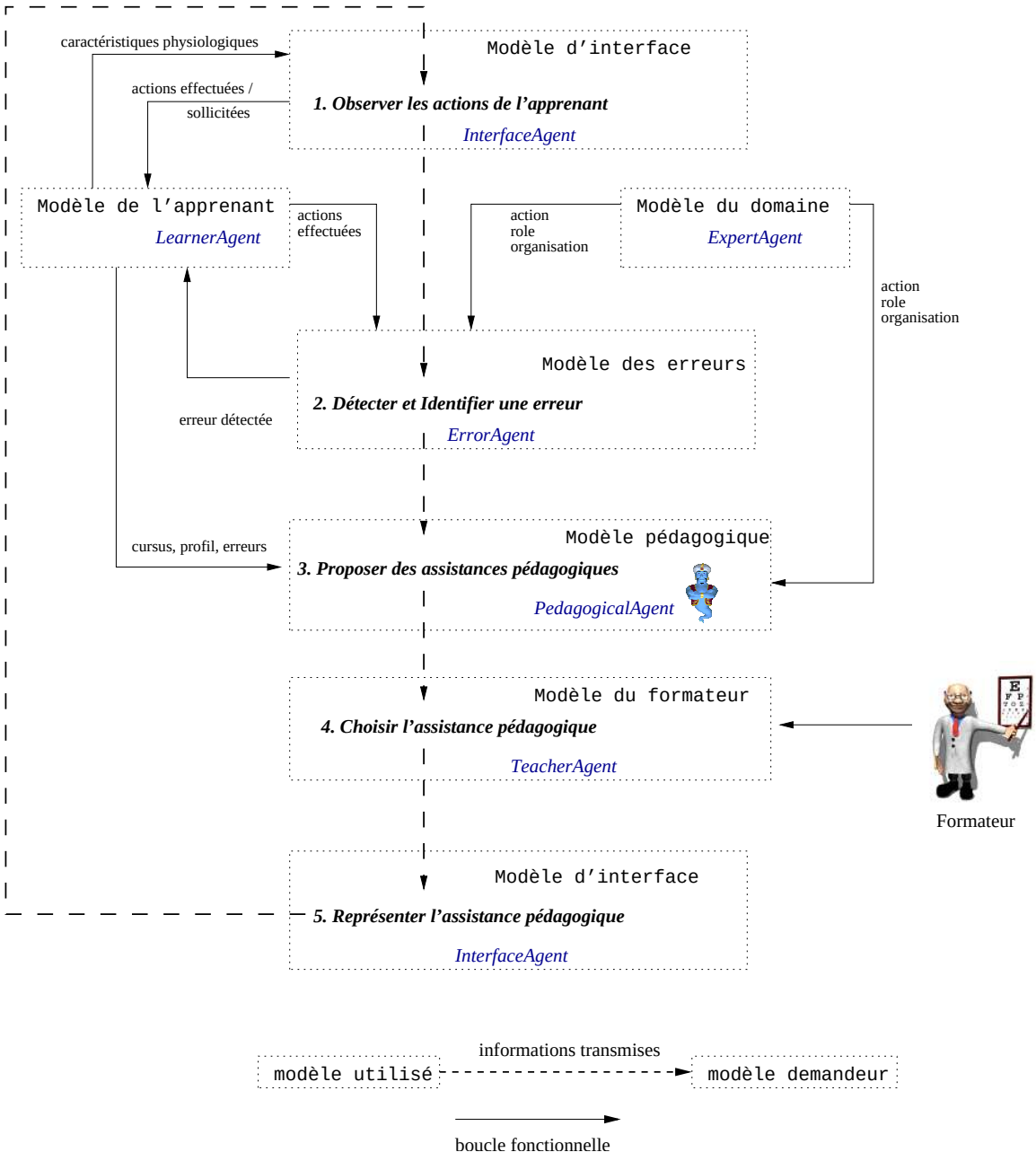


FIGURE 3.11 – Processus pédagogique de notre système constitué de cinq étapes.

3.2.2 Modèle du domaine

Le modèle du domaine est une représentation de la connaissance que l'apprenant doit acquérir, il représente l'ensemble des connaissances expertes sur le domaine d'apprentissage. Il contient également des informations permettant d'interpréter ces connaissances, afin de permettre l'exécution des actions pour « *faire à la place* » de l'apprenant. Le modèle du domaine contient alors des connaissances qui sont comparables aux connaissances déclaratives (savoir) et procédurales (savoir-faire) liées à la compétence selon la définition de la section 2.1.3.

3.2.2 - A Rôle

Comme nous venons de le souligner, le modèle du domaine (MD) contient les connaissances liées à la compétence à acquérir (*cf.* figure 3.12). Dans notre cadre d'étude, l'apprentissage porte sur la réalisation de procédures collaboratives. Il s'agit alors de spécifier les équipes, les procédures, les actions et les règles d'usage général. Cette tâche est à la charge de l'expert du domaine. Ce modèle doit également avoir une capacité de raisonnement sur le domaine pour renseigner les autres modèles (responsabilités, ordonnancement, conditions, *etc.*).

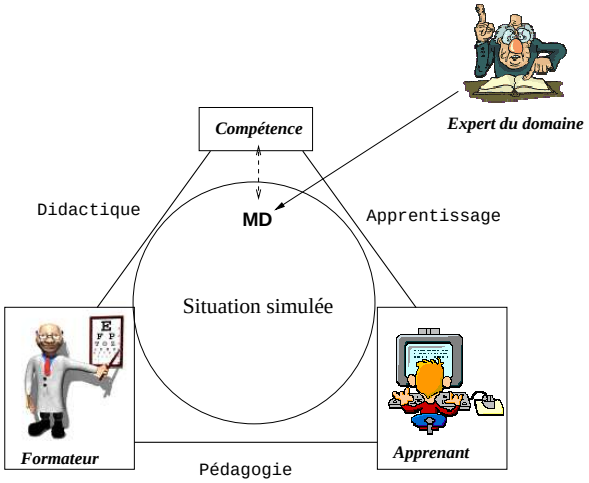


Figure 3.12 – Le modèle du domaine dans la situation d'apprentissage.

3.2.2 - B Agent expert

Le modèle du domaine intègre un agent expert. Sa base de connaissances et ses comportements sont représentés en figure 3.13.

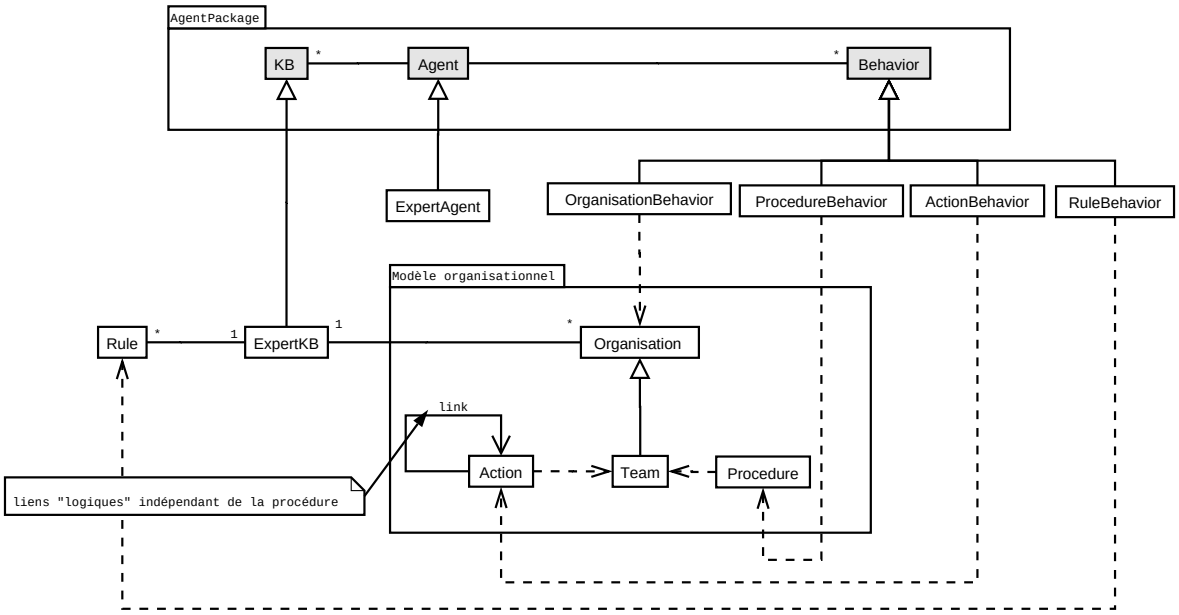


Figure 3.13 – Diag. de classe UML du modèle du domaine.

Base de connaissances

Dans un premier temps, il s'agit de s'interroger afin de définir quelles sont les connaissances qui doivent être représentées. Pour répondre à cette question, nous distinguons deux types d'informations qui doivent être intégrées. Il s'agit des éléments liés au travail à réaliser (procédural en équipe) et ceux liés aux règles d'usage général :

1. Travail procédural en équipe.
   Dans le cas de l'apprentissage de procédures collaboratives, il faut représenter les connaissances qui portent sur les équipes, les procédures et les actions :

   (a) *Les équipes*
   La connaissance sur l'environnement social est constituée d'un ensemble de connaissances expertes sur les équipes. Elle contient une représentation des rôles, des relations hiérarchiques entre rôles, et des responsabilités au sein des procédures.

   (b) *Les procédures*
   La connaissance sur les procédures doit permettre la définition de l'ordonnancement des actions.

   (c) *Les actions*
   La connaissance sur les actions doit contenir les buts des actions et doit préciser leurs conditions (pré/post conditions). Elle doit également fournir les liens de dépendance qui peuvent exister entre les actions, indépendamment de l'ordonnancement de la procédure.

   Exemple : L'action « Prendre la lance » permet de posséder la lance. L'action « Arroser le camion » dépend de l'action « Prendre la lance » sans pour autant y être directement liée dans la procédure.

2. Les règles d'usage général.
   Ces règles sont indépendantes du travail procédural en équipe. Elles sont pourtant importantes pour garantir la crédibilité des exercices. Ces règles sont à respecter et ne sont pas soumises à l'ordonnancement de la procédure.

   Exemple : il ne faut jamais traverser un passage clouté lorsque le signal de feu piéton est rouge.

Les connaissances à représenter sont à présent identifiées. Nous décrivons ici les formalismes utilisés pour réifier ces informations pour chaque type de connaissance :

1. Travail procédural en équipe.
   La représentation des connaissances sur les équipes utilisent la structure organisationnelle définie en section 3.1.2 qui précise les responsabilités. L'ordonnancement des actions au sein de la procédure est représenté par la structure du diagramme d'activités. Les actions sont définies par un but et des conditions évaluables. Ainsi, le modèle du domaine utilise directement les informations issues du `Modèle organisationnel` que nous avons précisé pour le travail procédural en équipe.
   Plus précisément, le modèle du domaine maintient des connaissances générales indépendantes du contexte dans lequel elles sont utilisées. Par conséquent, le modèle utilisé s'appuie sur les organisations, mais ne référence pas les instances de ces organisations. Ainsi, un agent sollicitant l'`ExpertAgent`, ne demande pas des informations spécifiques (*e.g.* Est ce que l'agent Y doit faire l'action « marcher vers l'objet X » ?), mais pose une question générique : « quelles actions B doivent être effectuées, après l'action A, dans la mission M, considérant le rôle R_1 et dans l'organisation O ? ».
   De plus, le modèle du domaine contient les liens entre actions qui ne sont pas exprimés dans la procédure. Les actions sont liées entre elles en considérant les relations de dépendance « logique », indépendamment de l'agencement de la procédure. Ces liens sont spécifiés par l'expert du domaine.

   Exemple : L'action « démarrer la voiture » est liée à l'action « prendre les clefs de la voiture ».

   Chaque action connaît les actions qui lui sont dépendantes et celles dont elle-même est dépendante. Ainsi, il est possible d'expliquer qu'il faut faire l'action A puisqu'elle est nécessaire pour effectuer l'action B.

2. Les règles d'usage général.
   La représentation des règles d'usage utilise un formalisme fondé sur des règles logiques. Nous souhaitons obtenir des explications (telle condition est non remplie). Nous proposons alors qu'une règle soit constituée de composantes qui correspondent à des termes logiques évaluables. L'évaluation des termes permet de générer une explication le cas échéant.

   Exemple de règle : un véhicule peut ne pas s'arrêter à un stop si il est un véhicule prioritaire (composante logique 1) et qu'il a actionné sa sirène (composante logique 2).

Pour résumer, l'expert du domaine doit préciser la structure organisationnelle, les procédures, les actions, les liens entre actions et les règles d'usage général. Ces informations constituent la base de connaissances de l'agent expert.

Comportements

L'agent expert (`ExpertAgent`) n'a pas un comportement réactif opérationnalisé. Il a un rôle d'informateur. En effet, l'agent expert renseigne tout modèle qui souhaite l'utiliser. Il peut raisonner sur le domaine concernant les éléments suivants :

1. *L'environnement social* : `OrganisationBehavior`.
   Il possède la capacité de raisonnement sur les agents et les rôles. L'`ExpertAgent` est capable de renseigner sur les responsabilités de chaque intervenant. En considérant la connaissance sur l'environnement social, il répond à la question « est ce que l'action A est de la responsabilité du rôle R ? ». Il peut également renseigner des rôles affectés à chaque agent par organisation.
2. *Les procédures* : `ProcedureBehavior`.
   Il peut fournir l'ordonnancement des actions ainsi que les buts de chaque étape (procédure / action). Comme Steve (Rickel et Johnson, 1999), l'`ExpertAgent` est capable de fournir des explications. Il répond à la question « quelles sont les actions à réaliser après l'action A ? ». Il précise le type de relation entre les actions.
   
   Exemple « Après l'action A, il faut faire l'action B ou (l'action C et l'action D) ».
3. *Les actions* : `ActionBehavior`.
   Le `Modèle organisationnel`, défini en section 3.1.2, propose un modèle d'actions soumis à des pré/post conditions. Ces dernières sont représentées sous la forme de règles. Comme Steve, l'`ExpertAgent` est capable de fournir des explications et de faire à la place de l'apprenant. Il peut fournir les pré/post conditions et expliquer pourquoi elles sont (ou ne sont pas) satisfaites. Il peut également donner les liens entre actions indépendants des procédures.
4. *Les règles du domaine* : `RuleBehavior`.
   L'`ExpertAgent` a la possibilité de fournir des explications sur les « règles d'usage » (indépendantes de la procédure) (`Rule`) en s'appuyant sur les réifications des termes de la règle. En effet, déterminer si une règle est transgressée revient en une évaluation individualisée des termes de la règle. Il est alors possible de déterminer quel terme est impliquée dans le non respect de la règle.

3.2.3 Modèle des erreurs

L'apprenant doit réaliser une procédure dans laquelle il joue un ou plusieurs rôles. Lorsqu'il évolue dans l'environnement, il peut effectuer des comportements qui sont en désaccord avec le travail à réaliser ou avec des règles d'usage général. Le modèle des erreurs offrent alors une caractérisation générique des erreurs permettant de les détecter et de fournir à l'agent pédagogique des moyens d'y réagir.

3.2.3 - A Rôle

Le modèle des erreurs (`ME`) permet de détecter et de typer des erreurs effectuées par l'apprenant indépendamment de l'exercice. Outre cette capacité à caractériser une erreur

d'une manière générique, il contient une base de connaissances sur des erreurs classiques dans un domaine particulier permettant d'expliquer pourquoi il y a erreur, et d'associer des éléments de l'environnement impliqués pouvant être utilisés pour faciliter leur compréhension. Ces connaissances sont spécifiées par le formateur (*cf.* figure 3.14). Les erreurs détectées et éventuellement associées à des erreurs classiques, portent sur la compétence à acquérir.

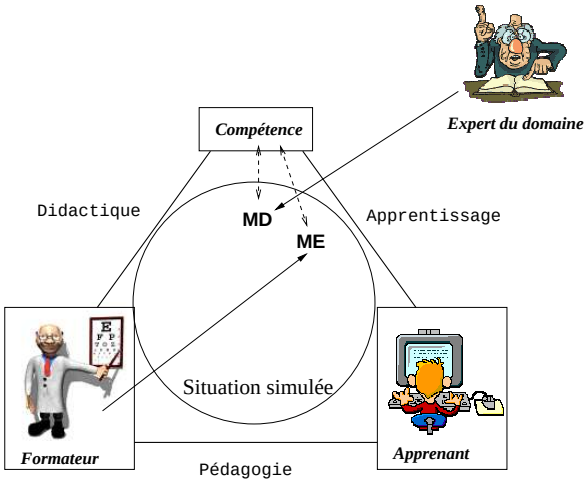


Figure 3.14 – Le modèle des erreurs dans la situation d'apprentissage.

3.2.3 - B Agent erreur

Le modèle des erreurs intègre un agent erreur (`ErrorAgent`). Sa base de connaissances et ses comportements sont représentés en figure 3.15.

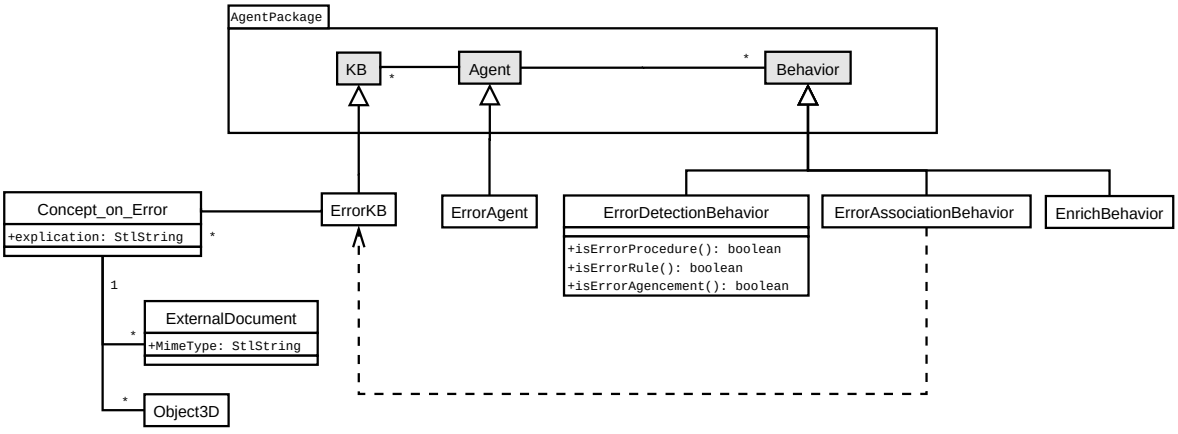


Figure 3.15 – Diag. de classe UML du modèle des erreurs.

Base de connaissances

Le premier objectif du modèle des erreurs est de caractériser les erreurs de manière générique. Pour cela, il faut représenter les informations qui explicitent la distinction entre telle ou telle erreur, sans considérer l'exercice particulier à traiter. Nous proposons alors de distinguer différents types d'erreurs caractéristiques (*cf.* figure 3.16) :

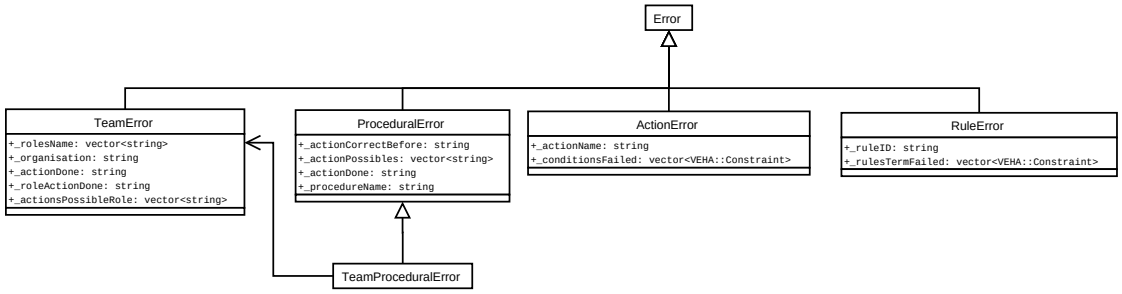


FIGURE 3.16 – Différents types d'erreurs.

1. Les règles d'usage (`RuleError`).
Cette erreur précise les termes impliqués dans la mise en échec de la règle.

> *Considérons la règle d'usage suivante : « en cas de fuite de gaz, il faut arroser la source de la fuite seulement si la chaleur est supérieure à trente degrés et que le vent est inférieur à 70km/h. » Une fuite est survenue, l'apprenant donne l'ordre d'arroser la source. La température est de vingt degrés et le vent est de 20 km/h. C'est une erreur portant sur une règle d'usage, le terme impliqué dans la mise en échec est « la chaleur est supérieure à trente degrés ».*

2. Sur les actions (`ActionError` : pré-condition).
Elle détermine les parties de la pré-condition d'une action qui sont en échec.

> *Considérons l'action A_1 « faire une mesure d'explosimétrie » qui a pour pré-condition « possède un explosimètre ». L'apprenant sollicite l'action A_1 sur le pompier chef, ce dernier ne possède pas un explosimètre. C'est une erreur portant sur une action. La pré-condition en échec est « possède un explosimètre ».*

3. Sur les rôles (`TeamError` : rôle dans l'équipe).
Elle précise la différence de responsabilité entre des rôles à jouer et le rôle lié à une action particulière. Cette erreur intervient lorsque l'action sollicitée fait partie des actions à réaliser, mais n'est pas de la responsabilité de l'apprenant.

> *Exemple : Dans la procédure courante P_1, la dernière action correcte effectuée est l'action A_0, les actions possibles sont $\{A_1, A_2$ et $A_3\}$. Les rôles joués par l'apprenant contiennent les actions $\{A_0, A_1$ et $A_7\}$. L'action réalisée est l'action A_3. Bien que cette action fasse partie des actions à réaliser, elle n'est pas contenue dans l'ensemble des actions possibles pour les rôles joués, il s'agit donc d'une erreur.*

4. Sur la procédure (`ProceduralError` : ordonnancement).
Elle souligne la différence entre une action et les actions possibles pour une procédure

donnée. Cette erreur intervient lorsque l'action sollicitée ne fait pas partie des actions à réaliser.

Exemple : Dans la procédure courante P_1, la dernière action correcte effectuée est l'action A_0, les actions possibles sont {A_1, A_2 et A_3}. L'action réalisée est l'action A_8. Cette action n'est pas contenue dans l'ensemble des actions possibles, il s'agit donc d'une erreur.

5. Sur les rôles dans la procédure (`TeamProceduralError` : rôle dans l'ordonnancement). Il s'agit d'une spécialisation de `ProceduralError`. Cette erreur intervient lorsque l'action sollicitée ne fait pas partie des actions à réaliser et en plus lorsqu'elle n'est pas de la responsabilité de l'apprenant.

Exemple : Dans la procédure courante P_1, la dernière action correcte effectuée est l'action A_0, les actions possibles sont {A_1, A_2 et A_3}. Les rôles joués par l'apprenant contiennent les actions {A_0, A_1 et A_7}. L'action réalisée est l'action A_8. Cette action ne fait pas partie des actions à réaliser, en plus cette action ne fait pas partie d'un des rôles joués, il s'agit donc d'une erreur.

Pour déterminer le type de l'erreur, il suffit de connaître la procédure courante, les rôles joués par l'apprenant et les règles d'usage général.

Comme nous l'avons souligné précédemment, le modèle des erreurs offrent bien plus qu'une caractérisation générique des erreurs. En effet, la base de connaissances de l'agent erreur est constituée d'une liste d'erreurs « classiques » du domaine faisant référence à des cas particuliers des exercices à réaliser. Ces erreurs sont classées par type et associées à des informations supplémentaires. Ces dernières prennent la forme d'une règle qui est spécifiée par le formateur. Une règle est composée de deux parties. La première précise les conditions qui permettent de détecter l'erreur en question. Les conditions portent sur le type générique de l'erreur (`RuleError`, `TeamError`, `TeamProceduralError`, `ProceduralError` ou `ActionError`) et sur les informations particulières de l'erreur.

Exemple : « Si l'erreur est de type ProceduralError et que l'apprenant a effectué l'action A au lieu de l'action B ».

La connaissance est alors représentée en spécifiant les caractéristiques de ces erreurs classiques (partie gauche de la règle) que nous augmentons par des informations supplémentaires associées (partie droite de la règle). Plus précisément ces informations, constituant la deuxième partie de la règle, sont :

- ▷ un texte explicatif précisant en quoi c'est une erreur,
- ▷ des éléments de l'environnement pouvant être utilisés pédagogiquement en relation à cette erreur (objet 3D),
- ▷ des éléments externes à l'environnement (vidéo, documents écrits, *etc.*) en relation à cette erreur.

Nous appelons les règles représentant les erreurs classiques du domaine des concepts sur erreurs (`Concept_on_Error`).

✎ Rappelons que ces règles, qui visent des particularités d'un domaine, ne sont pas nécessairement spécifiées. Dans tous les cas, le mécanisme « générique », qui caractérise les erreurs, affecte un type (`RuleError`, `TeamError`, `TeamProceduralError`, `ProceduralError` ou `ActionError`). Par conséquent, la prise de décision pédagogique peut raisonner sur les erreurs, indépendamment des particularités de l'exercice à traiter.

Exemple : « Si l'erreur est de type ProceduralError alors effectuer l'assistance pédagogique 1 ».

Comportements

L'agent erreur (`ErrorAgent`) a pour principal objectif de détecter une erreur et d'en informer les autres modèles. Ce renseignement est primordial, notamment pour la prise de décision pédagogique. De plus, comme nous l'avons vu, l'agent erreur peut éventuellement reconnaître une erreur classique et dans ce cas fournir des informations complémentaires qui constituent des ressources pédagogiques adaptées à la situation. Ainsi, l'`ErrorAgent` propose les comportements suivants :

1. *Détecter et typer les erreurs :* `ErrorDetectionBehavior`.
   Il détecte une erreur potentielle effectuée par l'étudiant :

   (a) Il vérifie que les pré-conditions de l'action sollicitée par l'apprenant sont satisfaites (`ActionError`).

   (b) Il compare les actions possibles pour avancer d'un pas dans la procédure (fournies par le modèle du domaine) et l'action sollicitée par l'apprenant (fournie par le modèle de l'apprenant). Les informations sur ces actions sont récupérées en dialoguant respectivement avec l'`ExpertAgent` et le `LearnerAgent` associé à l'apprenant. Il s'agit de vérifier si l'action sollicitée fait partie des actions potentiellement possibles en terme de procédure, *i.e.* qu'elle est l'une des actions qui doit être effectuée en considérant la dernière action correcte et l'ordonnancement de la procédure (`ProceduralError`). Il s'assure également que l'apprenant à bien la responsabilité (définie par les rôles qu'il doit jouer) d'effectuer l'action sollicitée (`TeamError` ou `TeamProceduralError`).

   (c) Il vérifie les termes des règles d'usage général du modèle du domaine (`RuleError`).

2. *Reconnaître une erreur classique du domaine :* `ErrorAssociationBehavior`.
   Notre modèle des erreurs contient une base de connaissances sur les erreurs classiques effectuées par un étudiant dans le domaine concerné. Il peut être comparé au "*Buggy model*" présent dans d'autres ITS (Brown et Burton, 1978). L'`ErrorAgent` est capable de reconnaître une telle erreur en comparant l'erreur à la partie gauche des règles correspondant aux erreurs connues de la base de connaissances. Il enrichit alors l'erreur (qui a été préalablement typée) avec des éléments supplémentaires (`Concept_on_Error`).
   Lorsque l'`ErrorAgent` a détecté, typé une erreur (et éventuellement reconnu une erreur classique du domaine), il transmet ses informations à l'agent pédagogique.

3. *S'auto enrichir :* `EnrichBehavior`.
   L'`ErrorAgent` enregistre des informations sur la fréquence des erreurs afin d'en rendre compte au formateur. Un mécanisme, sous le contrôle du formateur, utilise cette information pour enrichir la connaissance sur les erreurs classiques du domaine.

3.2.4 Modèle de l'apprenant

Il existe à ce jour de nombreux modèles qui proposent de modéliser un utilisateur (Rouse, 1982; Rasmussen, 1986; Anderson, 1993). Contrairement à ces approches, nous n'avons pas l'ambition de modéliser un humain virtuel, mais plutôt de récolter des informations dans un cadre de formation.

3.2.4 - A Rôle

Le modèle de l'apprenant (`MA`) s'intéresse aux caractéristiques et aux activités de l'apprenant (*cf.* figure 3.17). Il doit représenter les informations caractérisant l'étudiant tant d'un point de vue prototypique (profil de l'étudiant) que d'un point de vue cursus (progression de l'étudiant). Il correspond alors à la représentation de l'apprenant dans la situation d'apprentissage.

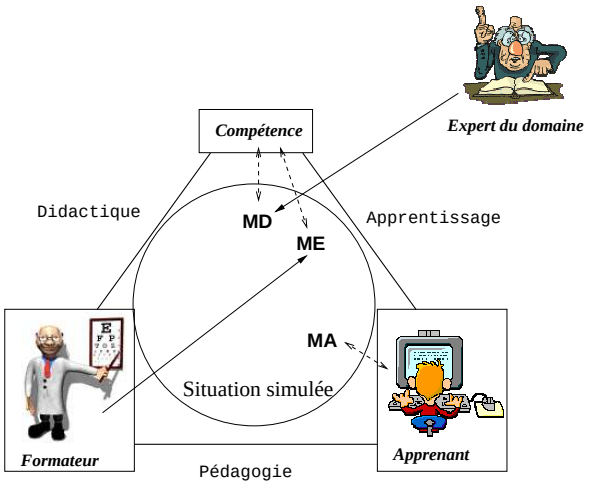


Figure 3.17 – Le modèle de l'apprenant dans la situation d'apprentissage.

3.2.4 - B Agent apprenant

Le modèle de l'apprenant intègre un agent apprenant. Sa base de connaissances et ses comportements sont représentés en figure 3.18.

Rappelons que l'objectif est de fournir un apprentissage différencié dans des environnements virtuels multi-apprenants. Par conséquent pour chaque apprenant, un agent interface (`InterfaceAgent`) et un agent apprenant (`LearnerAgent`) sont utilisés. L'`InterfaceAgent` met à jour le modèle de l'apprenant en lui indiquant les activités de l'apprenant. Nous décrivons plus loin cet agent.

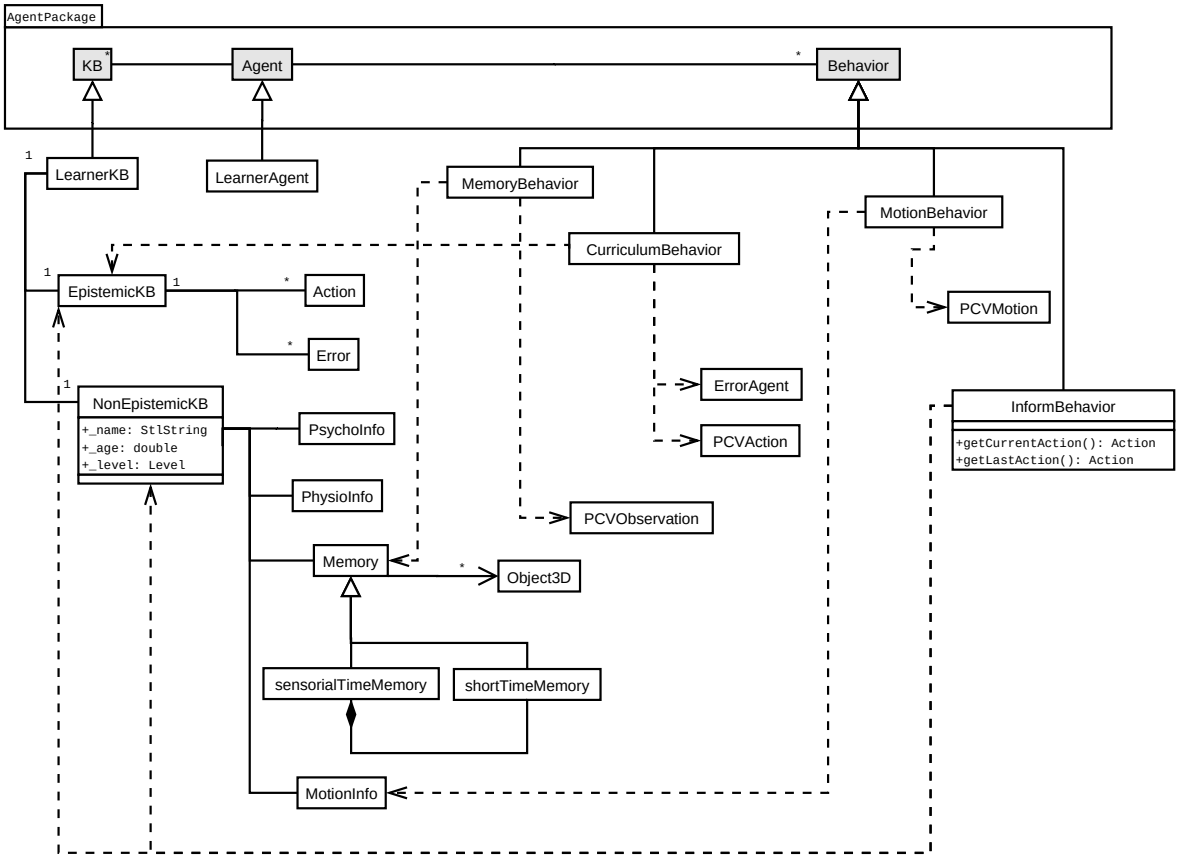


Figure 3.18 – Diag. de classe UML du modèle de l'apprenant.

Base de connaissances

Le modèle de l'apprenant représente les connaissances qui caractérisent l'apprenant. Dans notre cas, les informations liées à l'acquisition de compétences sont particulièrement importantes. De plus, nous devons considérer les particularités de l'apprenant qui ne sont pas liées à son cursus (*e.g.* capacité auditive).

Pour représenter ces connaissances, nous avons utilisé la distinction proposée par (Delestre, 2000), séparant dans la base de connaissances la partie *épistémique* et la partie *non épistémique* :

1. La partie *épistémique* (`EpistemicKB`) fournit les informations sur l'état des connaissances de l'apprenant pour les notions présentes dans le domaine. Elle contient une représentation des actions et des erreurs effectuées par l'apprenant. Les actions et les erreurs de l'apprenant sont stockées en suivant le déroulement de la simulation, nous obtenons alors son cursus lors de l'exercice.

   Comme dans la proposition de Lourdeaux (2001), notre modèle de l'apprenant contient un sous-modèle du domaine en représentant les actions effectuées, reprenant la méthode de l'*overlay* (Stansfield et al., 1976). Nous faisons alors l'hypothèse simplificatrice considérant que les actions reflètent les acquis de l'apprenant.

2. La partie *non épistémique* (`NonEpistemicKB`) propose une représentation de trois concepts issus de l'ingénierie cognitive :
   (a) un profil *psychologique* (par rapport à la tâche) : cette connaissance provient d'une analyse d'un questionnaire rempli préalablement à l'exercice. Il permet de définir notamment la stratégie d'apprentissage privilégiée de l'apprenant (Kermarrec, 2002) ;
   (b) des caractéristiques *physiologiques* : elles expriment les capacités auditives et visuelles de l'étudiant. Ces informations sont utilisées par le `PCVObservation` (*cf.* modèle d'interface 3.2.5) ;
   (c) une image *mémorielle* : elle reflète des éléments potentiellement observés par l'apprenant, de manière instantanée (`sensorialTimeMemory`) ou à court terme sur les dernières secondes (`shortTimeMemory`). Un mécanisme d'oubli efface les éléments de la partie à court terme qui n'ont pu être observés récemment[10].

   L'apprenant évolue dans un environnement virtuel. Nous devons alors considérer les activités de l'apprenant et notamment ses déplacements dans l'univers 3D. Par conséquent, la partie non épistémique possède des informations relatives aux déplacements de l'apprenant (zones, directions, vitesses, *etc.*) (`MotionInfo`). Ces informations sont fournies par le `PCVMotion` (*cf.* modèle d'interface 3.2.5).

Comportements

L'agent apprenant met à jour les informations associées à l'apprenant cible. De plus, il informe les autres modèles qui souhaitent connaître les activités et particularités de l'apprenant.

1. Le `LearnerAgent` met à jour les informations dynamiques concernant les activités de l'apprenant :
   (a) *Mise à jour du curriculum* : `CurriculumBehavior`
   Il stocke les nouvelles actions et les nouvelles erreurs effectuées par l'apprenant. Ces informations sont mises à jour en utilisant le `PCVAction` et l'`ErrorAgent`.
   (b) *Mise à jour de la mémoire* : `MemoryBehavior`
   Il stocke les objets qui ont pu être observés par l'apprenant en considérant ces caractéristiques physiologiques. Ces informations sont mises à jour en utilisant le `PCVObservation`.
   (c) *Mise à jour des informations liées aux déplacements* : `MotionBehavior`
   Il stocke la position de l'apprenant (les applications sont structurées en zones), les directions et vitesses moyennes et instantanées. Ces informations sont mises à jour en utilisant le `PCVMotion`.

[10] Il s'agit simplement d'un conteneur d'éléments 3D, il ne faut donc pas faire de rapprochement abusif avec la notion de mémoire à court terme utilisée en psychologie cognitive. En effet, nous n'avons pas réifié les connaissances qui pourraient permettre la représentation d'un tel concept. Il en est de même pour le concept de mémoire à long terme.

2. Le `LearnerAgent` répond aux requêtes d'autres agents (`InformBehavior`) concernant les activités (exemple : dernière action effectuée) ou particularités de l'apprenant (exemple : stratégie d'apprentissage privilégiée).

3.2.5 Modèle d'interface

Le modèle d'interface a la charge de la communication bidirectionnelle entre l'apprenant et le système. Dans un cadre de formation, il s'agit d'offrir à l'apprenant la possibilité d'interagir avec son environnement et d'analyser ses activités. Le modèle d'interface est particulièrement important lorsqu'on étudie une application de réalité virtuelle.

3.2.5 - A Rôle

Le modèle d'interface (`MI`) représente et analyse les interactions entre l'apprenant et le système. Plus précisément, son rôle est de régir la communication et les interactions avec l'environnement. De plus, il doit détecter les activités de l'apprenant. Pour tout cela, le modèle de l'interface doit considérer les caractéristiques particulières de l'apprenant. Il se place comme un médiateur entre le système et l'apprenant dans la situation d'apprentissage (*cf.* figure 3.19).

Deux acteurs humain jouent un rôle dans la spécification du modèle de l'interface : le formateur et le concepteur de l'EVF. En effet, lorsque le formateur précise l'exercice, il définit les éléments physiques à considérer dans l'analyse des observations de l'apprenant. De plus, le concepteur de l'EVF implémente les moyens d'interaction entre l'apprenant et le système.

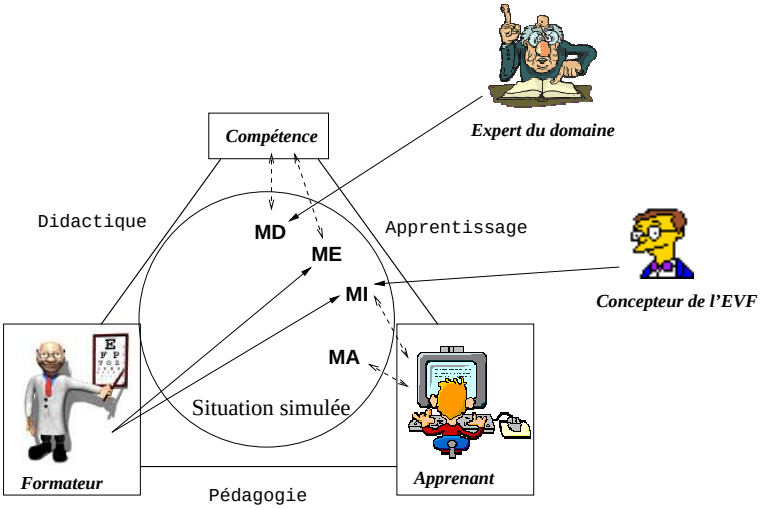


Figure 3.19 – Le modèle d'interface dans la situation d'apprentissage.

3.2.5 - B Agent interface

Le modèle d'interface intègre un agent interface (`InterfaceAgent`). Sa base de connaissances et ses comportements sont représentés en figure 3.20.

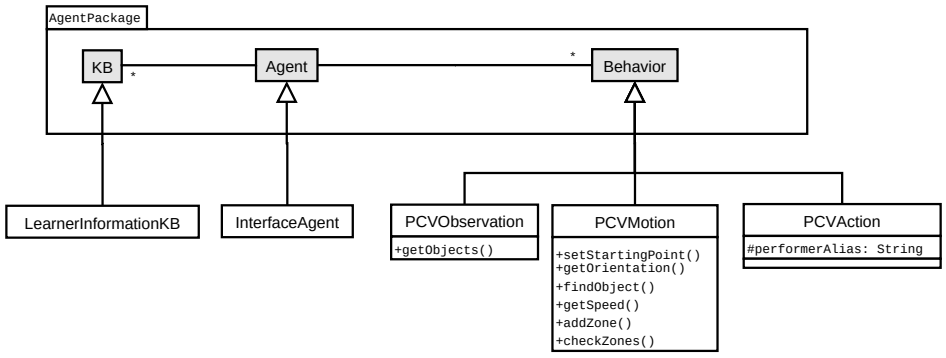


Figure 3.20 – Diag. de classes UML du modèle d'interface.

Le modèle d'interface a pour objectif principal l'analyse des comportements de l'apprenant dans l'univers virtuel. Par conséquent, pour représenter le modèle d'interface, nous nous inspirons des Primitives Comportementales Virtuelles (PCV) (Fuchs et Moreau, 2003). En effet, selon Fuchs et Burkhardt (2003), les activités d'un sujet dans une application de réalité virtuelle sont toujours décomposables en quelques comportements de base correspondant (les PCV). Les auteurs les regroupe en quatre catégories :

- ▷ observer le monde virtuel ;
- ▷ se déplacer dans le monde virtuel ;
- ▷ agir sur le monde virtuel ;
- ▷ communiquer avec autrui.

Base de connaissances

Afin de prendre en compte les caractéristiques personnelles de chaque apprenant pour analyser ses comportements, notre modèle d'interface contient une base de connaissances sur l'apprenant (`LearnerInformationKB` : sous partie de la base de connaissances du modèle de l'apprenant). Nous avons, par exemple, considéré ses particularités visuelles pour déterminer les éléments qu'il peut potentiellement observer dans le monde. De la même manière, le niveau de l'apprenant est utilisé pour régir ses possibilités d'actions. Les particularités de l'apprenant sont élaborées à partir du dialogue entre l'`InterfaceAgent` et le `LearnerAgent` (modèle de l'apprenant).

Comportements

L'`InterfaceAgent` possède trois comportements qui correspondent à trois des quatre PCV (*cf.* figure 3.20). En effet, nous n'avons pas traité la PCV communication, car les connaissances sur ce point semblent, pour l'instant, délicates à exploiter pédagogiquement.

Le `PCVAction` propose à l'apprenant d'interagir avec les objets de l'environnement et détecte les actions de l'apprenant. Le `PCVObservation` analyse ses observations potentielles. Enfin, le `PCVMotion` analyse ses déplacements. Nous détaillons ici chacun de ces comportements.

PCVAction

Le `PCVAction` offre d'une part des moyens d'interaction entre l'apprenant et l'environnement et d'autre part détecte les actions sollicitées :

1. *Les moyens d'interaction*
   Dans un premier temps, il s'agit de proposer à l'apprenant des moyens d'interaction avec les objets de l'environnement virtuel correspondant à ses prises de décision. Ces interactions peuvent être régulées en s'adaptant à l'apprenant (en utilisant la base de connaissances sur l'apprenant et en sollicitant l'agent pédagogique). En effet, on peut penser qu'il ne faut pas surcharger un apprenant novice en lui proposant des décisions irréalisables sur le moment, alors que cela peut être intéressant pour un apprenant expérimenté.
   Pour prendre une décision, l'apprenant va solliciter des objets physiques en leur demandant l'ensemble des actions qu'ils proposent. Les actions sur les objets d'un environnement virtuel peuvent être variées (rotation, translation, déformation, *etc.*). Néanmoins, notre cadre de travail n'est pas la formation au geste technique, mais à la prise de décision. Les objets physiques possèdent des connaissances sur les agents qu'ils représentent et par conséquent sur les actions qu'ils peuvent effectuer[11]. Lorsque l'apprenant sollicite une action sur un de ces objets :
   (a) la `PCVAction` récupère les informations intrinsèques à l'objet (ensembles des actions possibles que peut effectuer l'entité en considérant le rôle à jouer ou en considérant tous les rôles),
   (b) elle va ensuite demander à l'agent pédagogique quelles sont les actions à proposer, à partir de telles informations et des caractéristiques de l'apprenant,
   (c) la `PCVAction` affiche les actions proposées.
   La `PCVAction` permet de faciliter la sélection des diverses tâches à effectuer. Nous mettons en place une « substitution sensori-motrice » (Lourdeaux, 2001) sous la forme d'un menu OSD[12] (*cf.* figure 3.21). La `PCVAction` va afficher les diverses actions proposées à l'utilisateur.

2. *Les actions sollicitées*
   Dans un second temps, il s'agit de détecter les actions sollicitées et effectuées par l'apprenant, afin de mettre à jour le modèle de l'apprenant. Ce dernier peut solliciter une action en sélectionnant une icône du menu par la souris ou par la voix (nous utilisons un programme de reconnaissance vocale)[13]. Il prend alors sa décision et sollicite

[11] Nous envisageons d'intégrer une autre solution qui consiste à inspecter les opérations de l'objet et d'en déduire les actions dans la procédure.

[12] OSD : On Screen Display. Expression anglaise se référant à un appareil qui utilise l'écran de télévision pour afficher ses différents menus, afin de faciliter les réglages ou son utilisation courante. Concrètement l'élément en OSD est au premier plan en deux dimensions.

[13] Pour l'instant, nous n'avons implémenté que ces deux types d'interaction (souris et reconnaissance vocale) mais d'autres types d'interaction peuvent être envisagés (gant de données, reconnaissance de gestes, *etc.*).

Figure 3.21 – Un exemple d'utilisation du menu OSD dans l'environnement.

une action. Le `PCVAction` récupère cette information et va la transmettre au reste du système. Le `LearnerAgent` récupère l'information de sollicitation et met à jour son curriculum. Après une analyse de la décision par l'`ErrorAgent`, l'agent pédagogique choisit d'autoriser ou non l'action. Retenons qu'une action erronée ne sera pas forcément interdite selon la stratégie de l'agent pédagogique. En effet, on peut penser qu'il peut être intéressant d'effectuer une action, même s'il y a une erreur, pour certains apprenants[14]. Le `PCVAction` repère l'information relative à l'exécution de l'action (l'action a été réalisée) et informe le `LearnerAgent` le cas échéant.

PCVObservation

Il est intéressant de connaître les éléments qui sont, ou ont été, dans le champ de vision de l'apprenant pour prendre une décision pédagogique. En effet, considérons par exemple que la pré-condition d'une action A soit la présence de l'objet B. Nous pouvons penser que l'assistance pédagogique peut prendre en compte le fait que l'apprenant ait eu la possibilité d'observer cet objet ou non.

La PCV d'observation telle que nous l'avons définie, va permettre d'obtenir une liste des objets qui apparaissent dans le champ de vision de l'utilisateur, ainsi que leurs distances et leurs orientations par rapport à l'utilisateur. Bien qu'il soit réducteur de penser qu'un objet visible soit forcement perçu par l'apprenant, nous effectuons cette simplification pour notre modèle.

Afin de pouvoir s'adapter à tous les utilisateurs, la PCV d'observation va s'appuyer sur le modèle de l'apprenant pour obtenir des données physiologiques nécessaires (profondeur et largeur du champ de vision de l'utilisateur, contenu dans `LearnerInformationKB`). D'un autre côté, le `LearnerAgent` met à jour le modèle de l'apprenant (`Memory`) en faisant appel aux services de la PCV d'observation. Ce modèle ne couvre pas l'observation auditive et tactile, mais peut être étendu en adaptant les périphériques d'entrée.

Dans l'exemple illustré par la figure 3.22, nous avons disposé des objets dans un environnement. La PCV d'observation analyse en permanence l'environnement avec ici un

[14] Nous nous intéressons plus loin au problème de la « récupération » sur erreur dans un EVF (mécanisme de retour arrière).

champ de vision[15] réduit à un angle de $\frac{\pi}{2}$. Pour les besoins de l'explication, les informations sur la visibilité des objets sont représentées sous la forme d'un pourcentage affiché au dessus de chaque objet (a). Lorsque l'utilisateur se déplace, la visibilité des objets est mise à jour (b). La fréquence de l'analyse est paramétrable afin de trouver un compromis entre le besoin en terme de mise à jour des informations et la charge en terme de temps de calcul.

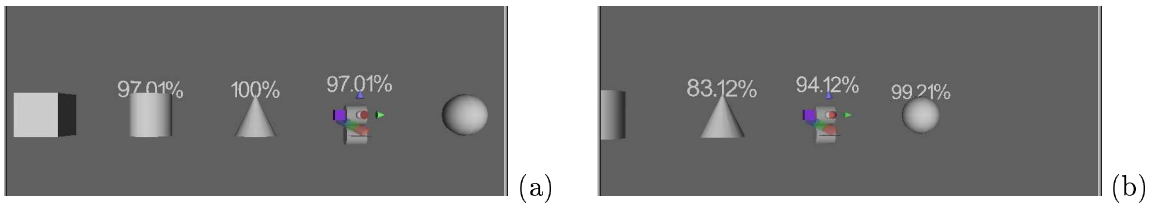


Figure 3.22 – Exemple d'utilisation de la PCV d'observation.

PCVMotion

Il n'est pas nécessaire de connaître en permanence la position exacte de l'apprenant dans l'environnement. En revanche, lors d'une tâche précise, les informations permettant de savoir si l'apprenant se déplace vers ou en dehors du lieu où se déroulera l'action, si l'apprenant s'approche d'un objet précis et à quelle allure, sont des données intéressantes pour la prise de décision pédagogique.

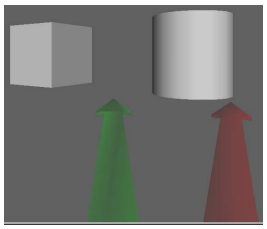
La PCV de déplacement répond à ces besoins en permettant :

- ▷ d'obtenir l'orientation moyenne de l'utilisateur depuis un point de départ donné ;
- ▷ de savoir quel est l'objet vers lequel l'utilisateur se dirige (moyenne relative aux déplacements) ;
- ▷ de connaître la vitesse instantanée de l'utilisateur ;
- ▷ de savoir les distances (sous forme de variables floues : au dehors, loin, près, au dedans) entre des zones et l'utilisateur ;

Le `LearnerAgent` va définir un point de départ pour le calcul du déplacement moyen (basé sur le lieu de l'action précédente par exemple). Depuis ce point, la PCV de déplacement va envoyer les informations sur les zones, la direction et la vitesse en permanence au `LearnerAgent` permettant de mettre à jour le modèle de l'apprenant.

Dans l'exemple suivant (*cf.* figure 3.23), nous avons disposé des objets dans un environnement et défini des zones autour de chaque objet. Lors de l'approche d'une zone, l'information de proximité (b) se met à jour. La direction instantanée de l'utilisateur est représentée par la flèche claire et la direction moyenne (*i.e.* la direction qu'a pris l'apprenant depuis un point de départ donné) par la flèche foncée (a).

[15] Le cône de vision est relatif à la perception de l'image sur un écran.


(a)

(b)

Figure 3.23 – Exemple d'utilisation de la PCV de déplacement.

3.2.6 Modèle du formateur

Le modèle du formateur fournit au formateur des possibilités en terme de pédagogie et définit l'exercice à réaliser. Un exercice (IMS, 2003) est caractérisé par un environnement simulé faisant intervenir une ou plusieurs équipes qui effectuent une procédure collaborative.

3.2.6 - A Rôle

Le modèle du formateur (MF) est la représentation du formateur dans la situation simulée (*cf.* figure 3.24). Il fixe « les règles du jeu », *i.e.* quelle est la ou les procédures à effectuer, et le ou les rôles à jouer par tel apprenant (et par conséquent ceux qui seront joués automatiquement). De plus, le modèle du formateur offre également la possibilité d'intervenir par des moyens d'interaction privilégiés dans l'environnement au travers de l'agent formateur. Aussi, il offre une vue d'ensemble de la procédure à réaliser (qui a fait quoi) permettant un suivi de l'exercice courant. Enfin, il récolte les propositions d'assistances pédagogiques résultant du raisonnement de l'agent pédagogique.

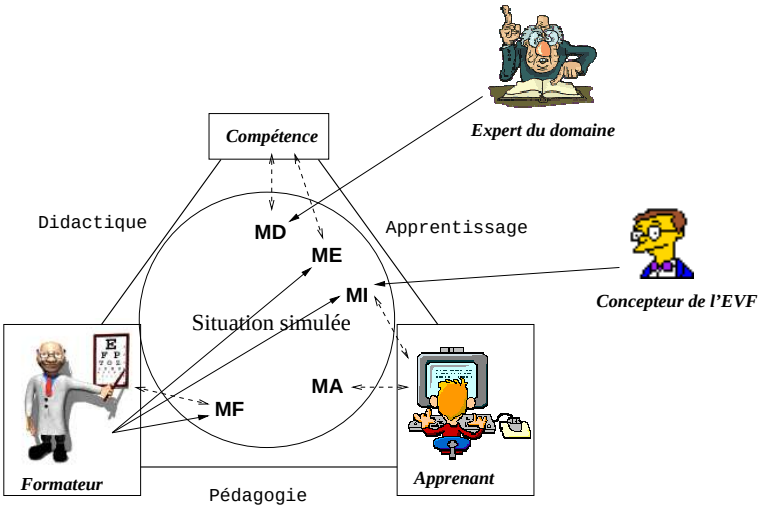


Figure 3.24 – Le modèle du formateur dans la situation d'apprentissage.

3.2.6 - B Agent formateur

Le modèle du formateur intègre un agent formateur. Sa base de connaissances et ses comportements sont représentés en figure 3.25.

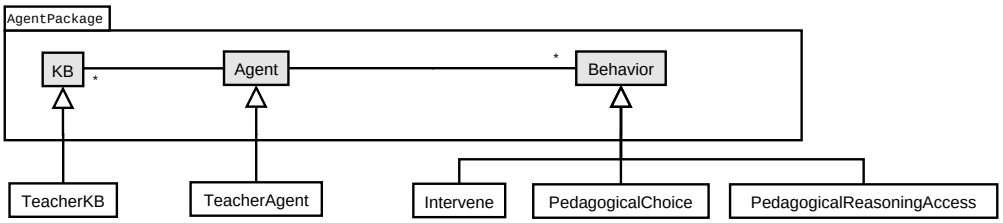


Figure 3.25 – Diag. de classes UML du modèle du formateur.

Base de connaissances

Pour qu'un exercice ait lieu, il faut que le formateur détermine les consignes, *i.e.* les règles à respecter. La base de connaissances de l'agent formateur précise alors, pour chaque exercice à effectuer, les informations suivantes :

- ▷ l'équipe dans laquelle les apprenants évoluent,
- ▷ la procédure à réaliser,
- ▷ le(s) rôle(s) à jouer.

L'exercice étant précisé, il s'agit d'assister le formateur dans le suivi des activités au sein de la procédure à réaliser. Pour cela, le modèle maintient un suivi de la procédure, en considérant l'exercice courant, qui centralise les informations concernant les actions et les erreurs effectuées automatiquement par le système, ou réalisées par les apprenants. Les informations concernant ces derniers, lui sont fournies par les `LearnerAgent` en charge de chaque apprenant.

Comportements

L'agent formateur (`TeacherAgent`) dispose de trois comportements qui offrent au formateur les possibilités suivantes :

1. *Intervenir dans l'environnement :* `Intervene`.
   Le formateur est un utilisateur particulier et par conséquent, son statut doit pouvoir lui permettre d'intervenir dans l'environnement. Dans cette optique, l'agent formateur fournit des moyens d'interaction privilégiés avec l'application :
   - ▷ il offre la possibilité de prendre le contrôle d'agents, le formateur peut alors jouer un rôle dans l'équipe ;
   - ▷ il permet de solliciter des opérations.

   Exemples : Sélectionner un objet physique et le faire clignoter. Afficher sur l'ordinateur de l'apprenant un message, etc.

2. *Choisir parmi les propositions d'assistances pédagogiques :* `PedagogicalChoice`.
   L'agent formateur propose les assistances pédagogiques qui sont élues par l'agent pédagogique. Le formateur fait son choix *via* l'agent formateur.
3. *Accéder au raisonnement pédagogique simulé :* `PedagogicalReasoningAccess`.
   L'agent formateur donne accès au raisonnement pédagogique simulé *via* l'agent pédagogique. Le formateur peut alors essayer de comprendre pourquoi l'agent pédagogique a proposé telle ou telle assistance pédagogique.

Notre modèle du formateur reste actuellement très limité. En effet, nous avons borné notre étude aux besoins minimums. Néanmoins, des travaux futurs pourront exploiter ce modèle pour envisager un environnement qui tient compte des préférences du formateur en terme de formation ou qui considère plusieurs formateurs.

3.2.7 Modèle pédagogique

Le modèle pédagogique simule un processus décisionnel se référant à une intervention pédagogique.

3.2.7 - A Rôle

Le modèle pédagogique (`MP`) est la représentation du pédagogue dans la situation d'apprentissage (*cf.* figure 3.26). Il simule un raisonnement pédagogique, par conséquent il influe sur la pédagogie et sur l'apprentissage.

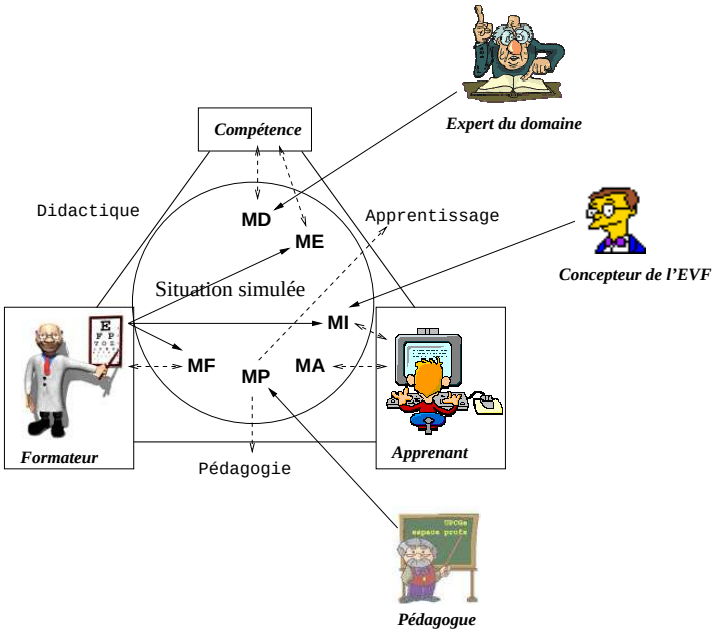


Figure 3.26 – Le modèle pédagogique dans la situation d'apprentissage.

3.2.7 - B Agent pédagogique

Le modèle pédagogique intègre un agent pédagogique. Sa base de connaissances et ses comportements sont représentés en figure 3.27.

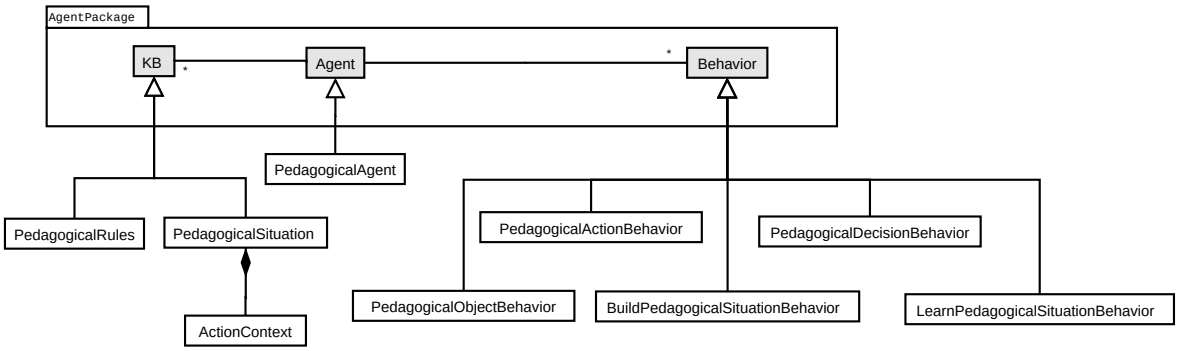


FIGURE 3.27 – Diag. de classe UML du modèle pédagogique.

Base de connaissances

Pour raisonner, l'agent pédagogique doit obtenir une représentation des connaissances qui caractérisent une situation, *i.e.* un ensemble d'éléments qui détermine les circonstances dans lesquelles l'apprenant évolue à un instant donné. De plus, l'agent pédagogique doit posséder les connaissances qui, considérant la situation actuelle (appelée situation pédagogique), définissent son comportement de prise de décision. Ainsi, la base de connaissances comporte deux parties :

1. *La situation pédagogique* : `PedagogicalSituation`
   La situation pédagogique offre la possibilité de caractériser des contextes, en « filtrant » l'ensemble des connaissances disponibles pour ne conserver que les informations clefs. Elle représente l'ensemble des informations considérées pour la prise de décision pédagogique. Ces connaissances portent sur l'apprenant et sur le travail à réaliser. Elles utilisent le résultat de l'analyse des modèles précédents de notre ITS. Ce concept est primordial dans notre problématique, il fait donc l'objet d'une description complète présentée en section 3.3.

2. *Les règles pédagogiques* : `PedagogicalRules`
   Les décisions pédagogiques sont prises en considérant des règles pédagogiques[16]. Ces règles portent sur les éléments de la situation pédagogique.

   Exemple : si l'apprenant est novice et que l'erreur est de type « agencement », alors faire clignoter les ressources des actions correctes.

[16] Le choix du formalisme à base de règles, est justifié dans le chapitre qui suit.

Comportements

L'agent pédagogique possède des comportements « réactifs » (utilisation direct des informations disponibles) et des comportements « cognitifs » (qui nécessitent une analyse à partir de connaissances).

Comportements réactifs

L'agent pédagogique a des comportements réactifs simples, qui concernent deux aspects de la pédagogie du système :

1. *Choix des actions possibles :* `PedagogicalObjectBehavior`.
   ⇨ Liberté d'action de l'apprenant.
   Pour agir dans l'environnement, l'apprenant doit solliciter des objets qui vont proposer un ensemble d'actions possibles. L'`InterfaceAgent` demande au `PedagogicalAgent` quelles actions doivent être proposées à l'apprenant. Ainsi, l'objectif de ce comportement est de fournir les actions à proposer. Pour cela, l'agent pédagogique se base sur des règles contenues dans sa base de connaissances. Dans l'exemple suivant, le `PedagogicalAgent` utilise des connaissances sur l'apprenant.
   - *Exemple 1 : si l'apprenant est novice, alors afficher les actions suivantes.*
   - *Exemple 2 : si l'apprenant est expérimenté, alors afficher toutes les actions que l'agent sait réaliser (en considérant ses rôles).*

2. *Décision de poursuite :* `PedagogicalActionBehavior`.
   ⇨ Exécuter une action malgré une erreur.
   Une fois que l'apprenant a sollicité une action, le système la détecte par l'`InterfaceAgent`. Ensuite, l'`ErrorAgent` l'analyse pour caractériser une éventuelle erreur. Dans le cas où une erreur est détectée, le `PedagogicalAgent` définit si l'action sollicitée doit être effectivement exécutée. Dans les exemples suivants, le `PedagogicalAgent` utilise le modèle de l'apprenant et le modèle des erreurs.
   - *Exemple : si l'apprenant est expérimenté et que l'erreur est de type agencement, alors appliquer l'action dans l'environnement.*
     Exemple : si l'apprenant a réalisé au moins cinq erreurs de type agencement, alors appliquer l'action dans l'environnement.

Comportements cognitifs

L'agent pédagogique possède trois comportements plus complexes, *i.e.* des processus qui nécessitent un traitement plus important qu'une simple règle. Il s'agit de trois points clefs liés à la prise de décision pédagogique portant sur les propositions d'assistances pédagogiques. Afin de pouvoir expliquer leur fonctionnement respectif, ces comportements sont détaillés plus loin. Il s'agit des comportements qui visent les aspects suivants :

1. *Construire la situation pédagogique* : `BuildPedagogicalSituationBehavior`.
   L'agent pédagogique analyse les informations extraites des modèles précédents. Il fournit alors une base de connaissances dynamique, appelée situation pédagogique, pour la prise de décision pédagogique. La construction de la situation pédagogique fait l'objet de la section 3.3.

2. *Prendre une décision d'intervention pédagogique* : `PedagogicalDecisionBehavior`.
   En considérant la situation pédagogique, l'agent pédagogique simule un raisonnement pédagogique. L'objectif est d'intégrer les connaissances sur la pédagogie qui sont indépendantes de l'exercice à réaliser. La prise de décision d'intervention pédagogique est explicitée au chapitre 4.

3. *Adapter la prise de décision d'intervention pédagogique* : `LearnPedagogicalDecisionBehavior`.
   Le modèle simulant un raisonnement pédagogique doit s'adapter au couple apprenant-formateur. L'agent pédagogique modifie alors son modèle décisionnel en cours de simulation. Ainsi, les propositions d'assistances pédagogiques s'ajustent à mesure des simulations afin d'être de plus en plus efficaces. Le mécanisme d'adaptation est détaillé lors du chapitre 4.

3.3 Situation pédagogique

La situation pédagogique est utilisée pour définir un contexte (Pomerol et Brézillon, 2001). Turner (1993) souligne l'importance de cette notion pour créer un comportement « intelligent ». Dans notre cas, la situation pédagogique sert de base pour la prise de décision pédagogique (*cf.* figure 3.28). L'objectif est de définir un tel contexte d'un point de vue « générique », ce qui permet de manipuler des informations sans considérer l'exercice à réaliser.

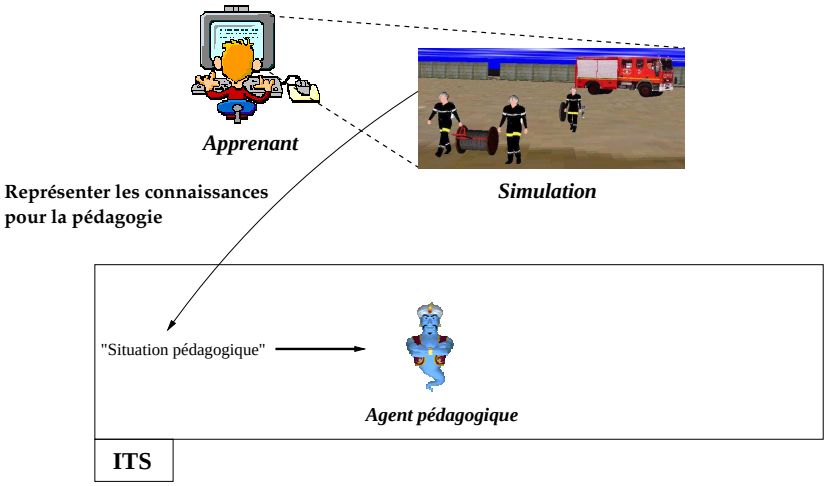


Figure 3.28 – La situation pédagogique est l'ensemble des connaissances qui peuvent être utilisées par l'agent pédagogique.

Dans le chapitre 2, nous avons montré d'une part que l'apprenant doit être au centre du système de formation, et d'autre part que la compétence est l'élément clef des formations que nous visons. Par conséquent, la prise de décision pédagogique doit se baser sur les connaissances portant sur l'apprenant et sur la compétence à acquérir. Ainsi, la situation pédagogique est constituée :

1. des informations sur le travail à effectuer, *i.e.* les éléments liés à la procédure à réaliser que nous détaillons dans la section suivante (section 3.3.1) ;
2. des informations relatives à l'apprenant, représentant les informations sur ses caractéristiques et sur ses activités que nous développons en section 3.3.2.

3.3.1 Informations concernant le travail à effectuer

Nous nous plaçons dans le cadre d'une formation au travail procédural. L'objectif est d'aider l'apprenant pour qu'il progresse dans la procédure. La compétence à acquérir porte donc sur la réalisation de la procédure dans un environnement dynamique.

Si on se rattache à la définition de la notion de compétence présentée au chapitre 2, on peut considérer qu'être compétent consiste à allier savoir-faire et savoir. En ce qui concerne le savoir-faire, l'apprenant agit dans l'environnement, il effectue les actions de la procédure et observe leurs effets. Nous rappelons que l'objectif n'est pas la formation au geste technique mais à la prise de décision. Les médias utilisés dans les applications de RV offrent l'interaction nécessaire entre l'environnement et l'apprenant. En ce qui concerne le savoir, il s'agit des connaissances sur l'agencement des actions dans la procédure. Ainsi, afin de favoriser l'acquisition de compétences, l'agent pédagogique doit disposer d'une représentation des connaissances portant sur l'agencement des procédures.

Considérant cette analyse, les connaissances manipulées dans la prise de décision pédagogique doivent être choisies judicieusement. En effet, ces connaissances sont utilisées afin de faire progresser l'apprenant dans la procédure, il s'agit alors de prendre en compte les éléments clefs. La section qui suit identifie de tels éléments.

3.3.1 - A Identification des concepts pertinents

L'objectif ici est de définir les éléments considérés, concernant le travail à effectuer, dans la situation pédagogique. On peut se demander ce qui pourrait être fait pour que l'apprenant mémorise, voire comprenne, l'enchaînement des actions.

Dans un premier temps, nous pouvons considérer la procédure comme un enchaînement d'actions défini par un expert du domaine. Les éléments à considérer sont donc sujets à un *agencement* qui n'est pas discutable et parfois non explicable. Dans un second temps, nous pouvons penser que, mémoriser l'enchaînement des actions, pourrait être facilité par sa compréhension. Dans ce cadre, Richard (1990) propose de se rattacher à la notion de sous-buts dans la procédure. Pour aboutir à l'objectif, *i.e.* la réalisation de la procédure, il faut effectuer un ensemble de sous-buts liés logiquement. Il s'agit alors d'étudier la procédure en considérant la distance au but de la procédure d'un point de vue *logique*, et non plus d'un point de vue chronologique.

L'analyse précédente fait donc apparaître deux manières d'aborder l'apprentissage de procédures : l'étude des liens d'ordonnancement métier qui sont fortement liés aux rôles dans la procédure et l'étude des liens logiques entre sous buts :

1. *Les liens d'ordonnancement*
   Ils effectuent les relations entre les actions en utilisant la description stricte de la procédure. Ils sont la conséquence même de l'ordonnancement des actions qui est défini par l'expert. Nous considérons les informations liées aux actions qui sont au plus proches de l'action sollicitée par l'apprenant. Plus précisément, il s'agit de :

   ▷ La dernière action correcte avant celle qu'il vient juste de solliciter.
   ▷ L'action qui vient d'être sollicitée par l'apprenant.
   ▷ Les actions correctes à effectuer en considérant le(s) rôle(s) à jouer (potentiellement différentes de l'action sollicitée).
   ▷ Les actions correctes à réaliser en considérant que tous les rôles sont joués par l'apprenant.
   ▷ Les actions qui suivent toutes les actions correctes.

   Nous avons choisi les actions au plus proches de l'action sollicitée pour essayer de réduire la « distance » entre le but (la fin de la procédure) et la position de l'apprenant dans la procédure. Ainsi, la situation pédagogique conserve les connaissances liées aux actions qui se trouvent à proximité de l'action sollicitée, d'un point de vue chronologique. Bien que ce choix n'ait pas fait l'objet d'études particulières qui en montrent l'efficience, sélectionner ces informations nous semble rationnel.

2. *Les liens logiques entre sous-buts*
   La procédure peut être considérée comme un graphe représentant l'enchaînement des sous buts logiques. Nous considérons alors toutes les actions liées à l'action effectuée par l'apprenant. Concrètement, il s'agit des actions nécessitant l'effet de l'action correcte (condition d'utilisation, état d'une ressource, *etc.*). Il faut bien distinguer ces liens qui correspondent à une logique individuelle alors que les liens d'ordonnancement correspondent à l'organisation d'une procédure collective.

Toutes les actions identifiées précédement constituent la situation pédagogique. Plus précisément nous nous intéressons aux informations liées à certaines actions. A ce stade, il devient nécessaire de préciser les connaissances relatives à la notion d'action. Dans cette optique, la section suivante définit le « contexte d'action ».

3.3.1 - B Contexte d'action

Le « contexte d'action » est constitué des connaissances liées directement à l'`Action` (définie en section 3.1.2), des connaissances liées à l'`Operation` (définie en section 3.1.1) cible de l'`Action`, ainsi que les connaissances liées à l'agent qui a sollicité l'action puisqu'il est le protagoniste (le `Performer`, *i.e.* un `TeamAgent`). Finalement, nous considérons les connaissances précisées dans le tableau 3.1. Nous illustrons la notion de contexte d'action par un exemple au travers du tableau 3.2.

Contexte d'action	
Information liées à	*Connaissance*
`Action`	▷ Les ressources, *i.e.* des entités physiques (`PhysicalEntity`). ▷ Les conditions d'utilisation, contenant un texte explicatif associé à une pré-condition évaluable de l'opération. ▷ L'objectif de l'action, sous la forme d'une explication. ▷ Le résultat de l'action, exprimant une modification de l'environnement.
`Operation` cible de l'`Action`	▷ Les pré-conditions d'exécution évaluables. ▷ Le nom de l'opération. ▷ Les post-conditions d'exécution évaluables. ▷ Des arguments paramètres de l'opération.
`Performer`	⇨ `TeamAgent`

Tableau 3.1 – Définition du contexte d'action.

Exemple : L'action « Arroser Camion » effectuée par le « pompier ».

Information liées à	*Connaissance*	
`Action` : « Arroser Camion »	▷ Ressource :	« Lance »
	▷ Condition :	« Il faut posséder une lance »
	▷ Objectif :	« Refroidir le camion »
	▷ Résultat :	« Température camion inférieur à 30 degrés »
`Operation` : « Arroser »	▷ Pre-condition :	« PossedeLance==true »
	▷ Nom :	« Arroser »
	▷ Post-condition :	« target→getAttribut("Température") < target→getAttribut("TempératureStable") »
	▷ Argument :	target=« Camion »
`Performer`	⇨ `TeamAgent` :	« Pompier »

Tableau 3.2 – Exemple de définition d'un contexte d'action.

3.3.1 - C Bilan des connaissances considérées

Ainsi, nous utilisons les *contextes d'action* pour représenter les connaissances associées aux actions. Rappelons que notre objectif ici est d'extraire les connaissances qui portent sur le travail à effectuer pour la prise de décision pédagogique. Dans ce cadre, reprenant la liste des actions que nous avons établie en section 3.3.1 - A, nous considérons les connaissances du tableau 3.3.

Connaissances	Description
① Contexte d'action précédente	La dernière action correcte a avoir été réalisée. Cette action fait référence et permet de se positionner dans la procédure.
② Contexte d'action sollicitée	Il s'agit de l'action sollicitée. Cette action peut être correcte, comme erronée. Elle n'est pas forcément réalisée, conformément au modèle pédagogique (*cf.* section 3.2.7 - B).
③ Contexte d'action(s) correcte(s) sans considération sur les rôles	En considérant la dernière action correcte, nous déterminons les actions à effectuer selon la procédure courante.
④ Contexte d'action(s) correcte(s)	Il s'agit d'un sous-ensemble de l'item précédent, qui ne considère que les rôles joués par l'apprenant.
⑤ Contexte d'action(s) suivante(s)	Pour chaque action correcte, nous déterminons les actions qui suivent selon la procédure courante.
⑥ Contexte d'action(s) liée(s)	En considérant les actions à réaliser à la suite de la dernière action correcte, nous récupérons les liens « logiques » entre actions indépendamment de la procédure. Nous obtenons les actions qui sont liées.

Tableau 3.3 – La situation pédagogique (1/2) : connaissances sur le travail à réaliser.

La construction de cet ensemble de connaissances est à la charge de l'agent pédagogique. Plus précisément, il s'agit d'une sous-partie du comportement `BuildPedagogicalSituation-Behavior` que nous avons évoqué en section (3.2.7 - B). Nous décrivons en figure 3.29 l'implémentation qui permet de récupérer les différentes connaissances manipulées par l'agent pédagogique. Remarquons que l'agent pédagogique récupère ou construit les connaissances sur le travail à réaliser lorsqu'il reçoit un message de l'agent interface qui précise qu'une action vient d'être sollicitée. Ce choix peut être discuté. En effet, un autre solution est de mettre à jour les connaissances sur erreur. Nous avons préférer reconstruire ces connaissances sur action pour offrir la possibilité d'intervenir, même si le comportement de l'apprenant est correct. Cela permet d'envisager des assistances pédagogiques qui le confortent dans ses décisions, qui l'encouragent ou qui tentent de semer un doute (*e.g.* affirmer des règles fausses qui entrent en contradiction avec ses choix).

Récupérer les connaissances revient à solliciter les agents de l'ITS qui contiennent les informations souhaitées. Les informations manipulées se basent sur l'agent apprenant, l'agent interface et l'agent expert. Le tableau 3.4 effectue un bilan sur la provenance des connaissances.

```
void PedagogicalAgent::buildPedagogicalSituationBehavior(void)
{
  ArConstRef<Message> msg=getNextMessage();
  if( msg.valid() )
   {
    // message de l'agent interface => une action vient d'être sollicitée
    if( msg->getText() == "new action" &&
        msg->getEmitter()->getClass()->isA( InterfaceAgent::CLASS() )  )
        {

        /*****************************************************************************************/
        /** La situation pédagogique construit les connaissances portant sur le travail à réaliser **/
        /*****************************************************************************************/

        // 1. Quelle a été la dernière action correcte avant l'action venant d'être sollicitée ?
        ArRef<ActionContext> actionBefore = getLastActionCorrect();

        // 2. Quelle est l'action qui vient d'être sollicitée ?
        ArRef<ActionContext> actionDo = getAction();

        // 3. Qu'est ce qui doit être fait dans la procédure ?
        vector< ArRef<ActionContext> > actionsPossible = getActionsPossible();

        // 4. Qu'est ce que l'apprenant doit faire dans la procédure pour ses rôles ?
        vector< ArRef<ActionContext> > actionsPossibleRole = getActionsPossibleRole();

        // 5. Quelles sont les actions suivantes dans la procédure ?
        map< ArRef<ActionContext> , vector< ArRef<ActionContext> > >  actionsNextProcedure;
        actionsNextProcedure = constructActionCorrectToActionsNextProcedure();

        // 6. Quelles sont les actions liées d'un point de vue logique
        //    aux actions qui doivent être faites dans la procédure ?
        map< ArRef<ActionContext> , vector< ArRef<ActionContext> > >  actionsNextLogic;
        actionsNextLogic = constructActionCorrectToActionsNextLogic(actionsPossible);

        // => Creation de la situation pédagogique
        createPedagogicalSituationPart_ProceduralWork(actionBefore,
                                                      actionDo,
                                                      actionsPossible,
                                                      actionsPossibleRole,
                                                      actionsNextProcedure,
                                                      actionsNextLogic);


        }
    }
  /*****************************************************************************************/
  /** La situation pédagogique est complétée par les connaissances portant sur l'apprenant **/
  /*****************************************************************************************/
  ...

}
```

Figure 3.29 – Implémentation C++ simplifiée de la méthode qui génère les connaissances portant sur le travail à réaliser.

Travail à effectuer						
Provenance	LearnerModel	InterfaceModel	ExpertModel	ExpertModel	ExpertModel	ExpertModel
Contexte d'action	Précédente	Sollicitée	Correcte(s)	Correcte(s) sans rôle	Suivante(s)	Liée(s) logiquement

Tableau 3.4 – Informations disponibles concernant le travail à effectuer dans la situation pédagogique.

3.3.2 Informations concernant l'apprenant

Comme nous venons de le voir, la situation pédagogique est constituée de deux parties : les connaissances portant sur le travail à réaliser que nous avons précisées dans la section précédente, et les connaissances portant sur l'apprenant qui sont exprimées ici.

Les informations liées à l'apprenant sont représentées par le tableau 3.5. Nous précisons la provenance de chaque information, *i.e.* le modèle qui génère originellement l'information. Nous énumérons les différents sous-types des informations ainsi que les valeurs possibles.

Les informations concernant l'apprenant sont d'origines diverses, mais sont toutes recueillies par le modèle de l'apprenant. Elles concernent aussi bien des données statiques (*e.g.* âge) que dynamiques (*e.g.* éléments de la mémoire instantanée). Notons que les erreurs effectuées par l'apprenant sont stockées, *i.e.* le type et éventuellement le concept sur erreur. Ces données offrent, par exemple, la possibilité de vérifier la fréquence de telle ou telle erreur, ou de se rendre compte que l'apprenant effectue la plupart du temps des erreurs de type agencement. De la même manière, les contextes associés aux actions sont conservés. Ces informations permettent, par exemple, de savoir si l'apprenant a déjà utilisé telle ou telle ressource.

Le choix des connaissances représentées ici n'a pas fait l'objet d'études particulières. Ces connaissances correspondent aux données qui sont à notre disposition dans l'environnement, elles pourraient être enrichies dans l'avenir par de nouveaux éléments.

3.3.3 Utilisation des connaissances de la situation pédagogique

La situation pédagogique contient les informations représentées dans les tableaux 3.4 et 3.5. Ces éléments peuvent être utilisés de deux manières :

1. *Consulter une information*
   Les informations liées à l'apprenant et au travail à effectuer, peuvent être consultées pour caractériser un contexte. En effet, ces données sont mises à jour en cours de simulation et peuvent être manipulées.

   Exemple : L'apprenant d'un niveau novice, vient d'effectuer une erreur de type procédurale et a dans son image mémorielle à court terme les ressources de l'une des actions correctes.

   Nous pouvons remarquer que la situation pédagogique est définie d'un point de vue « générique », *i.e.* que nous manipulons des informations sans considérer l'exercice à réaliser. Nous manipulons, par exemple, l'information relative à l'action effectuée par

Apprenant

Provenance	LearnerModel		LearnerModel		LearnerModel
Information	Informations générales		Physiologie		Psychologie
	Niveau	Âge	Auditif	Visuel	Stratégie d'apprentissage
	▷ Novice ▷ Débutant ▷ Expérimenté	▷ Cadet ▷ Junior ▷ Senior ▷ Vétéran	▷ Faible ▷ Normal	▷ Daltonien ▷ Normal	▷ 1 ▷ 2 ▷ 3 ▷ 4

Provenance	ExpertModel	
Information	Contextes d'Actions effectuées	
	Actions	Operations
	▷ vector< `PhysicalEntity` > (ressources) ▷ textes explicatifs (pré-conditions) ▷ explication de l'objectif de l'action ▷ texte explicatif post-conditions	▷ vector< `Constraint` > (pré-conditions) ▷ nom de l'opération ▷ vector< `Constraint` > (post-conditions) ▷ vector< `Attribut` > (paramètres de l'opération)

Provenance	ErrorModel	
Information	Erreurs effectuées	
	Type	Concept sur erreur
	▷ Équipe (`TeamError`) ▷ Procédural (`ProceduralError`) ▷ Action (`ActionError`) ▷ Rule (`RuleError`)	▷ vector< `PhysicalEntity` > ▷ texte explicatifs ▷ vector< string > (URL docs)

Provenance	InterfaceModel		InterfaceModel	
Information	Image mémorielle		Déplacement	
	Instantanée	Court terme	Direction	Zones
	vector< `PhysicalEntity` >	vector< `PhysicalEntity` >	`PhysicalEntity`	vector< `Zones3D` >

Tableau 3.5 – La situation pédagogique (2/2) : connaissances sur l'apprenant.

l'apprenant (pour récupérer les ressources par exemple) sans savoir de quelle action il s'agit. Enfin, nous tenons à souligner que les connaissances de la situation pédagogique permettent le conditionnement de la prise de décision pédagogique (dans tel contexte appliquer telle assistance pédagogique).

2. *Déclencher une assistance pédagogique*
   Les informations de la situation pédagogique portent sur des éléments de l'environnement. Par exemple, nous pouvons demander les ressources (`PhysicalEntity`) des actions correctes ou bien récupérer le texte explicatif associé au concept sur l'erreur qui vient d'être effectuée. Chaque entité étant une `Entity` au sens de VEHA, il est donc possible de solliciter des opérations sur tel ou tel objet. De plus, chaque entité possède des opérations particulières qui représentent des assistances pédagogiques. Il est donc possible d'utiliser la situation pédagogique pour solliciter des assistances pédagogiques.

   Exemple : Faire clignoter la représentation physique des agents possédant les rôles qui permettent de réaliser les actions correctes.

Concrètement, nous venons de définir les informations disponibles en entrée et les éléments définis comme pertinents sur lesquels on peut agir en sortie pour une décision pédagogique. La figure 3.30 illustre ce fait, et met en exergue la nécessité de faire la relation entre ces entrées et ces sorties, en simulant un raisonnement pédagogique, ce qui est l'objet du chapitre suivant.

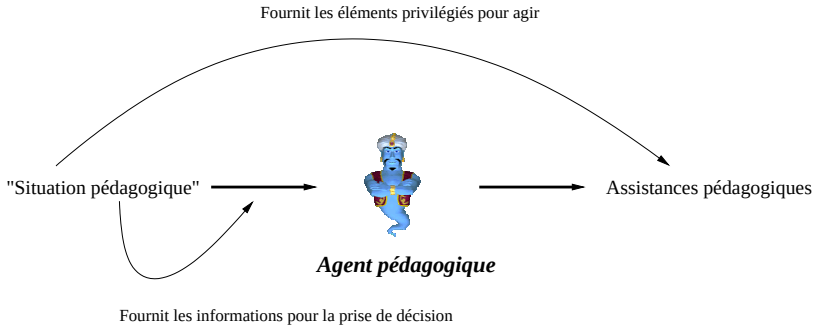


FIGURE 3.30 – La situation pédagogique fournit les entrées/sorties du mécanisme de prise de décision pédagogique.

3.4 Bilan

Dans ce chapitre, l'objectif était de définir les modèles permettant non seulement de simuler l'environnement de formation, mais également de représenter les connaissances, afin de les manipuler pour la prise de décision pédagogique.

La première étape a consisté à représenter les connaissances portant sur l'environnement virtuel. Pour cela, nous avons défini la représentation des entités virtuelles constituant l'environnement. Les modèles proposés proviennent de la librairie VEHA. Ensuite, nous avons représenté les connaissances liées au travail à effectuer qui est collaboratif et procédural.

Nous avons utilisé des modèles pour représenter la structure organisationnelle structurant les équipes. Enfin, nous avons utilisé les diagrammes d'activités UML pour modéliser les procédures.

Les connaissances représentées par ces modèles sont manipulables, elles peuvent alors être utilisées pour la pédagogie. Il restait à représenter les connaissances qui portent sur l'apprentissage lié aux activités de l'apprenant. Ces connaissances sont contenues dans des modèles inspirés de ceux qui sont utilisés classiquement dans un ITS.

Notre proposition définit les composantes d'un ITS sous la forme d'un système multi-agents. Les agents ont la responsabilité des modèles classiques des ITS. Ainsi, `ExpertAgent`, `LearnerAgent`, `ErrorAgent`, `TeacherAgent` et `InterfaceAgent` ont respectivement la charge du modèle du domaine, de l'apprenant, des erreurs, du formateur et d'interface. Ce système multi-agents fournit une représentation des activités de l'apprenant. Il construit également dynamiquement un ensemble d'informations relatives au travail à effectuer en considérant l'avancée de l'apprenant dans la procédure. L'ensemble de ces connaissances (relatives à l'apprenant et à son avancée dans la procédure) forme la « situation pédagogique ». Cette dernière sert de base de connaissances au raisonnement effectué par l'agent pédagogique (`PedagogicalAgent`).

La situation pédagogique offre la possibilité de déclencher des assistances pédagogiques sur les éléments qu'elle contient, elle fournit alors les sorties possibles de la prise de décision pédagogique. Le chapitre suivant explique comment l'agent pédagogique prend cette décision, en explicitant le mécanisme qui simule un raisonnement pédagogique et qui considère la situation pédagogique.

La modélisation présentée ici a fait l'objet de plusieurs publications (Buche et Querrec, 2005a,b; Buche et al., 2005a,b, 2004, 2003a,b) et d'une présentation au groupe de travail EIAH.

Chapitre 4

Agent pédagogique

Laisser à l'élève le soin de résoudre le problème d'apprendre, c'est se soustraire au devoir de résoudre le problème d'enseigner.

Bhurrus Frédéric Skinner

Dans ce chapitre, nous présentons le modèle de l'agent pédagogique. Plus précisément, nous décrivons le comportement de prise de décision qui effectue les propositions d'assistance pédagogique au formateur. Une étude des architectures comportementales existantes, tenant compte des besoins en terme de spécification du modèle, de hiérarchisation, de modularité, de réactivité et d'adaptabilité, nous amène à choisir les systèmes de classeurs. L'agent pédagogique organise ses connaissances selon une structure hiérarchisée, composée de trois niveaux représentants différentes abstractions des connaissances : les méthodes, les attitudes et les techniques pédagogiques. Chaque niveau contient des ensembles de règles représentant les différentes manières d'aborder une démarche, une attitude ou une technique. Les règles effectuent le lien entre les différents niveaux, chaque règle favorisant un ensemble du niveau inférieur. Une description générale des mécanismes, des principaux paramètres et l'introduction de hiérarchie mis en jeu dans les systèmes de classeurs, est présentée. Nous montrons alors l'adaptation de ces systèmes pour le comportement de l'agent pédagogique. Dans un premier temps, le modèle propose un mécanisme de diffusion au travers des trois niveaux, pour finalement ordonner les propositions d'assistances pédagogiques effectuées au dernier niveau. Dans un second temps, le modèle intègre un mécanisme d'apprentissage artificiel de type bucket brigade, basé sur un renforcement provenant des choix du formateur, permettant de personnaliser les aides pédagogiques.

Dans le chapitre précédent, nous nous sommes attachés à la description d'un modèle d'ITS selon plusieurs composantes : les modèles du domaine, des erreurs, de l'apprenant, du formateur, de l'interface et de la pédagogie. Nous nous intéressons ici au mécanisme de prise de décision contenu dans le modèle pédagogique, plus précisément nous détaillons le comportement de l'agent pédagogique dont l'objectif est de proposer des interventions pédagogiques au formateur.

Les agents pédagogiques et plus généralement les agents éducationnels, ont fait l'objet

de plusieurs études visant à fournir une classification (Baylor, 1999; Chou et al., 2003; Payr, 2003). Afin de situer notre agent, il convient de prendre en compte les principaux facteurs discriminants. Les agents éducationnels peuvent être personnifiés (Lester et al., 1997) ou non (Graesser et al., 1999). De plus, leur rôle peut s'apparenter à un tuteur (Rickel et Johnson, 1999), un compagnon (Goodman et al., 1998) ou un assistant du formateur (Lourdeaux, 2001). Aussi, l'environnement d'apprentissage visé peut être au format « WEB », *i.e.* du texte et des images 2D (Webber et Pesty, 2002), ou un environnement virtuel 3D (Lourdeaux, 2001). L'agent pédagogique que nous proposons est une entité non-personnifiée qui assiste le formateur, afin d'aider l'apprenant à réaliser une tâche collaborative (Constantino et Suthers, 2000) dans un environnement virtuel 3D.

C'est une entité autonome qui suit une boucle perception - décision - action. Les informations liées à la perception et les possibilités d'action, sont fournies par la situation pédagogique qui a déjà été définie. Ce chapitre présente la modélisation de la partie décisionnelle.

Le plan de ce chapitre est comme suit. Dans la section 4.1, nous choisissons une architecture comportementale pour l'agent pédagogique en considérant les contraintes induites par la problématique de prise de décision pédagogique. Dans la section 4.2, nous présentons notre proposition de modèle pédagogique[1]. Nous décrivons la structure et la dynamique du modèle, nous montrons également comment le spécifier. Enfin, la section 4.3 définit l'implémentation informatique.

4.1 Choix d'une architecture comportementale

4.1.1 Contraintes liées à notre cadre d'étude

Les critères que nous avons retenus dans le cadre de cette étude sont l'expressivité, la hiérarchisation, la modularité, la réactivité et l'adaptabilité.

Expressivité

Pédagogue

Le comportement de l'agent pédagogique est défini en suivant les instructions du pédagogue. En effet, ce dernier est en mesure de spécifier le modèle pédagogique sans tenir compte des particularités de l'exercice à traiter. Toutefois, la spécification comportementale d'une entité informatique par un spécialiste du domaine (ici le pédagogue) non informaticien, pose problème. En effet, le pédagogue n'a pas la connaissance requise pour utiliser un langage informatique.

Il faut alors utiliser un formalisme permettant d'effectuer la médiation entre le pédagogue et le modèle pédagogique. Dans ce cas, une possibilité est de proposer au spécialiste de définir

[1] Dans le chapitre précédent, nous avons présenté le modèle pédagogique. Il contient une base de connaissances et intègre l'agent pédagogique. Dans ce chapitre, nous utilisons le terme « modèle pédagogique » pour faire référence à la structure comportementale et aux connaissances de l'agent pédagogique utilisés pour la prise de décision amenant aux propositions d'assistances pédagogiques.

le comportement par un ensemble de règles, du type « si condition(s) alors effet(s) ». Elles ont l'avantage de représenter explicitement les composantes du comportement (spécification par le pédagogue) et d'être facilement interprétables (utilisation par le programme informatique).

Néanmoins, une telle description peut impliquer un nombre de règles important mettant en jeu des connaissances ayant des niveaux d'abstractions différents. L'utilisation d'une hiérarchie facilite la spécification de telles règles.

Hiérarchisation

Le raisonnement qui nous intéresse, met en jeu des connaissances de différentes natures, allant des notions abstraites de la pédagogie (adopter une méthode active face à l'apprenant, essayer de le perturber, *etc.*) jusqu'aux interventions concrètes dans l'environnement (décider de mettre tel ou tel élément en sur-brillance, expliquer telle connaissance, *etc.*). Nous pouvons alors favoriser les architectures hiérarchiques qui introduisent différents niveaux d'abstractions de données, structurants le raisonnement et permettant de prendre en compte des données hétérogènes.

Modularité

Le pédagogue est considéré comme un expert qui doit représenter son raisonnement. Ce travail nécessite une réflexion sur ses propres connaissances. Pour faciliter ce processus, les connaissances représentées doivent pouvoir être modifiées ou enrichies, sans pour autant remettre en cause celles qui sont déjà établies. Les connaissances peuvent alors être contenues dans des modules indépendants et réutilisables. La modularité garantit les capacités d'évolution du système puisque chaque composant peut être mis à jour individuellement sans que l'ensemble du système ne doive être réécrit.

Réactivité

La nécessité de crédibiliser des scènes simulées implique une contrainte temporelle, particulièrement importante pour les applications dédiées à l'apprentissage de procédures. Les entités évoluent donc en temps réel, et doivent réagir dans un temps « raisonnable » pour que la simulation soit crédible. De la même manière, pour être efficace, l'agent pédagogique doit réagir le plus rapidement possible face à une situation, sa prise de décision ne peut intégrer des mécanismes complexes tels que des recherches dans des graphes ou des retours arrières, elle doit donc être réactive.

Adaptabilité

Pour agir efficacement, l'agent pédagogique doit s'adapter au cas particulier en considérant l'apprenant réalisant l'exercice et le formateur. Il est alors intéressant de prendre en compte les caractéristiques individuels de chaque apprenant pour la prise de décision. Bien plus, nous pouvons doter l'agent pédagogique de capacités d'évolution lui permettant de

s'adapter dynamiquement (apprentissage artificiel en ligne[2]). Un mécanisme d'apprentissage artificiel permettrait de tirer partie des expériences passées et ainsi de s'adapter au couple formateur-apprenant. On peut considérer que le formateur a un rôle d'évaluateur jugeant le comportement de l'agent pédagogique. Il est donc envisageable qu'il puisse affirmer que les propositions d'interventions pédagogiques sont pertinentes ou non, sans pour autant associer des valeurs précises évaluant ces propositions, on parle alors d'apprentissage artificiel par renforcement[3].

Considérant ces contraintes, il s'agit de déterminer quel type d'architecture comportementale peut convenir.

4.1.2 Architectures décisionnelles classiques

Les architectures classiquement utilisées pour définir un modèle décisionnel, peuvent être divisées en trois grands types d'architectures comportementales (Donikian, 2004) :

1. Les architectures *connexionnistes* définissent le comportement d'un acteur à partir d'un ensemble de capteurs et d'effecteurs reliés entre eux par un réseau de nœuds transformant l'information. Cette approche comprend les modèles de type réseaux de neurones (Van de Panne et Fiume, 1993). Bien qu'intéressantes par ses capacités réactives et ses capacités d'apprentissage, les architectures à base de réseaux de neurones présentent des inconvénients non négligeables. En effet, l'approche stimulus-réponse constitue un niveau d'abstraction faible, et donc ne permet généralement que la modélisation de comportements instinctifs (*e.g.* le déplacement d'une entité). De plus, l'inconvénient majeur est qu'elle constitue une « boite noire ». En effet, les neurones et les synapses ne sont associés à aucune sémantique, le pédagogue ne peut donc pas représenter et manipuler ses connaissances. Nous ne pouvons donc pas choisir ce type d'architecture.
2. Les architectures à base d'*automates* décrivent un comportement par un ensemble d'états et de transitions. Le passage d'un état à un autre s'effectue lorsqu'un événement intervient. Les comportements peuvent être assemblés en utilisant plusieurs automates (Brooks, 1986; Maes, 1991; Donikian, 2001). Les automates à états finis classiques sont simples à comprendre et à implémenter. Cette méthode est peu coûteuse en mémoire et rapide en exécution. Ils permettent de combiner réactivité et abstraction. Néanmoins, il faut gérer toutes les conditions de transition entre les états (E états et T transitions engendrent E*T cas). Cette méthode devient fastidieuse lorsque les paramètres à prendre en compte sont nombreux. Aussi, une modification du comportement peut amener à une refonte intégrale de l'automate. De plus, ces automates sont prédictibles et réagiront toujours de la même manière. Enfin, exprimer la concurrence entre des comportements conflictuels est compliqué. Il est alors difficile de mettre en concurrence deux points de vue pédagogique (*e.g.* aider ou perturber l'apprenant). Nous ne pouvons donc pas choisir ce type d'architecture.

[2] Il faut distinguer l'apprentissage *hors ligne*, dans lequel toutes les données d'apprentissage sont fournies d'un seul coup, de l'apprentissage *en ligne*, dans lequel les données arrivent en séquences. Pour ce dernier, le mécanisme doit délibérer pour chaque arrivée et fournir une réponse.

[3] Les paradigmes d'apprentissage (supervisé, non-supervisé, renforcé) sont décrits en annexe A.

3. Les architectures utilisant des *règles* mathématiques, de scripts ou d'inférences, représentent un comportement par une approche symbolique (Perlin et Goldberg, 1996; Blumberg et al., 2002). Un des avantages des systèmes de production[4] est leur modularité. Chaque production individuelle de la base peut être additionnée, supprimée, changée indépendamment les unes des autres. Cette modularité facilite la création de la base de données. De plus, la structure rigide de ces bases les rend d'un accès facile pour une personne non initiée voire pour le système lui-même. Les connaissances sont exprimées d'une manière relativement naturelle. Bien plus, la représentation des connaissances permet différents niveaux d'abstraction. Les règles peuvent être gérées sous forme de modules hiérarchiques. Enfin, il est possible de faire évoluer les règles par apprentissage artificiel (Sanza, 2001).

Chaque type d'architecture comporte des avantages et des inconvénients. Pour autant, en considérant nos contraintes (spécification par un pédagogue, modularité, réactivité, hiérarchisation et adaptabilité), les architectures à base de règles sont les plus adaptées.

Nous nous orientons alors vers les architectures à base de règles possédant les caractéristiques décrites en section 4.1.1, plus précisément nous choisissons les systèmes de classeurs (Wilson, 2000b). En effet, un système de classeurs est une architecture réactive qui utilise une représentation déclarative intuitive et qui offre la possibilité de modifier la base de règles (modification, ajout et suppression). La structure comportementale fonctionne de manière autonome, elle peut donc être intégrée facilement sous forme de modules et, par conséquent, faire partie d'architectures hiérarchisées complexes. Enfin, les systèmes de classeurs intègrent des mécanismes d'apprentissage artificiel permettant d'optimiser en ligne les règles existantes (classiquement en utilisant un apprentissage par renforcement) et/ou de créer des nouvelles règles. Cette architecture présente donc les caractéristiques demandées, nous la présentons plus en détail dans la section 4.3.

4.2 Raisonnement pédagogique

Nous venons de choisir les systèmes de classeurs comme architecture comportementale contrôlant le comportement de l'agent pédagogique. Les informations utilisées par cette architecture prennent la forme de règles qui représentent les connaissances du pédagogue. Dans ce cadre, cette section s'intéresse à la simulation du raisonnement pédagogique à base de règles.

Les règles du modèle pédagogique servent de base à un moteur d'interprétation qui va permettre de simuler le raisonnement pédagogique. Pour mettre en place ce moteur, il faut se demander comment simuler un comportement pédagogique en utilisant des règles du type « si-alors », *i.e.* s'interroger sur comment faire la liaison entre les informations disponibles et les assistances pédagogiques à mettre en place. La figure 4.1 illustre le problème. Nous

[4] Un système de production est un programme qui représente le problème qu'il doit résoudre sous forme d'un ensemble d'informations factuelles appelées « faits » (*e.g.* le patient est âgé de 55 ans). Les connaissances nécessaires pour résoudre un problème sont représentées par un ensemble d'énoncés appelés « règles » (*e.g.* si le patient est âgé de plus de 40 ans, alors la probabilité d'une maladie cardio-vasculaire est grande).

disposons de la situation pédagogique, il s'agit de simuler un raisonnement spécifié par le pédagogue afin de choisir des assistances pédagogiques.

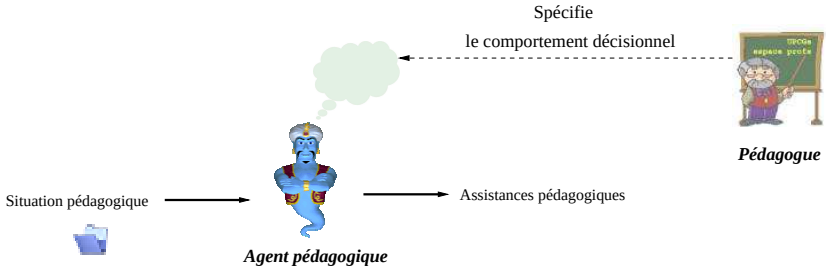


Figure 4.1 – Cadre du raisonnement de l'agent pédagogique.

On pourrait demander au pédagogue de définir des règles qui associent directement les grandeurs perçues et l'élection d'assistances. Cependant, ceci ne peut représenter une réflexion aussi complexe qu'une prise de décision sur une intervention pédagogique (Bruillard, 1997). En effet, un raisonnement pédagogique considère non seulement les éléments perçus, mais également des connaissances qui peuvent être la conséquence d'inférences internes interdisant un comportement purement réactif. Ces inférences imposent l'utilisation de variables et de mécanismes de chaînage pour la recherche d'assistances pédagogiques.

Les sections suivantes présentent un système à base de règles qui permet de représenter et de simuler un comportement adéquat. Plus précisément, nous définissons la structuration des connaissances (section 4.2.1) et la dynamique (section 4.2.2) du modèle pédagogique en charge du raisonnement pédagogique. Nous précisons également comment spécifier les connaissances du modèle (section 4.2.3). La réflexion sur la structuration des connaissances et sur la dynamique nous amène à une utilisation particulière des systèmes de classeurs, et entraîne des modifications de l'architecture classique qui sont explicitées en section 4.3.

4.2.1 Structuration des connaissances

Un raisonnement pédagogique est un comportement suffisamment complexe pour envisager une modélisation plus élaborée qu'une association directe entre les informations disponibles et les assistances pédagogiques à appliquer. Nous proposons une modélisation du processus de décision pédagogique selon deux points :

1. Nous proposons de structurer le processus de décision pédagogique. Il s'agit de permettre la décomposition du raisonnement en sous tâches et de considérer différents niveaux d'*abstraction des connaissances*. Nous pouvons alors hiérarchiser les règles, comme nous l'avons évoqué en section 4.1.1. Les règles peuvent alors être très abstraites (décrivant les démarches pédagogiques) ou très concrètes (définir une assistance pédagogique applicable directement dans l'environnement). Ce point est développé en section 4.2.1 - A.
2. Nous proposons de contextualiser une règle au sein d'autres règles. Une règle est alors située dans un ensemble. Un *ensemble de règles* traduit une orientation pédagogique et se situe dans un niveau d'abstraction donné. Contextualiser une règle est justifié par le fait qu'une règle peut être applicable pour une orientation pédagogique donnée et ne

pas avoir les mêmes effets ou ne plus avoir lieu d'être pour une autre orientation. Nous traitons cet aspect en section 4.2.1 - B.

4.2.1 - A Situer les règles dans des niveaux d'abstraction

La représentation des connaissances utilisées dans le modèle pédagogique est liée à un niveau d'abstraction. Murray (1996) définit six niveaux possibles d'abstraction, qui sont illustrés par les exemples suivants :

① si le bouton 1 de l'écran 5 est activé, alors aller à l'écran 12 ;

② si la question 12 obtient une réponse fausse, alors donner l'explication 5 ;

③ si l'étudiant donne une réponse fausse deux fois, alors fournir une explication complète ;

④ si l'étudiant est très confus, alors fournir un niveau de retour supplémentaire ;

⑤ donner aux étudiants plusieurs opportunités de réfléchir à chaque situation afin qu'ils apprennent à partir de leurs erreurs, alors mettre en place un retour portant sur l'augmentation du niveau de spécificité ;

⑥ l'apprentissage intervient à travers un processus actif de formation au concept, tout en tenant compte des informations nouvelles fournies par le contexte des connaissances précédentes.

Ces exemples montrent que le modèle pédagogique peut manipuler des connaissances plus ou moins abstraites. Nous considérons qu'un raisonnement est le résultat d'une réflexion structurée qui est sujet à un cheminement. Chaque étape utilise une abstraction des connaissances allant des connaissances abstraites de la pédagogie jusqu'à des connaissances concrètes directement applicables dans l'environnement.

Il s'agit de décomposer le modèle pédagogique en plusieurs parties. Pour cela, nous nous inspirons des travaux de Lourdeaux (2001) qui a proposé une modélisation s'articulant autour de trois niveaux : ① méthode pédagogique, ② stratégie pédagogique, ③ type et réalisme d'assistance pédagogique. Afin de se rapprocher des termes utilisés en pédagogie et pour qu'ils soient plus explicites pour un pédagogue, nous proposons de conserver ces trois étapes en modifiant les énoncés des différents niveaux. Notre modèle pédagogique distingue alors les trois étapes suivantes : les *démarches*, les *attitudes* et les *techniques* pédagogiques (*cf.* figure 4.2).

1. Les *démarches pédagogiques* (⇨ exemple : active, affirmative, *etc.*).
   Elles représentent les connaissances les plus abstraites. Une démarche pédagogique décrit la manière de conduire la pédagogie adoptée par l'enseignant pour favoriser l'apprentissage. En règle général, un enseignant valorise plus à un instant donné une démarche qu'une autre. La démarche unique imposée ou obligatoire serait une erreur, car elle est souvent affaire de circonstances. Une démarche pédagogique peut être mise en œuvre par des attitudes pédagogiques différentes.

   Exemple : pour une démarche pédagogique « active », nous pouvons envisager les attitudes consistant à « réaliser » à la place de l'apprenant ou à « expliquer ».

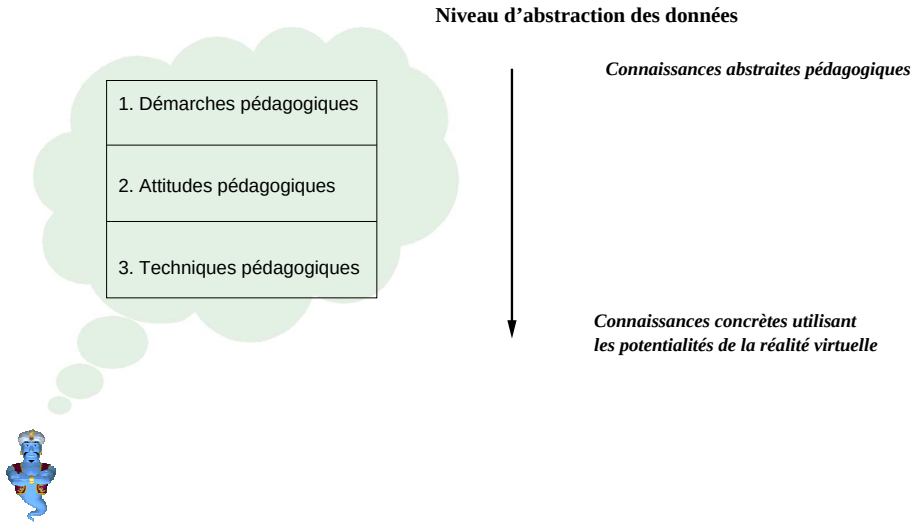


Figure 4.2 – Différents niveaux d'abstraction des données dans le modèle pédagogique.

2. Les *attitudes pédagogiques* (⇨ exemple : réaliser, expliquer, *etc.*).
Elles représentent un niveau intermédiaire entre les concepts pédagogiques abstraits et les interventions concrètes dans l'environnement virtuel. Une attitude pédagogique peut être représentée dans l'environnement par plusieurs techniques, objet du troisième niveau.

Exemple : pour une attitude pédagogique « expliquer », nous pouvons envisager les techniques de « rehaussement » ou de « simplification » de l'environnement.

3. Les *techniques pédagogiques* (⇨ exemple : enrichissement, dégradation, *etc.*).
Elles représentent des types d'interventions portant sur l'environnement. A ce sujet, une liste des types d'interventions en environnement virtuel a été proposée par Lourdeaux (2001). Les techniques pédagogiques représentent le niveau le plus concret. Ce niveau utilise les potentialités de la réalité virtuelle en terme de formation. Elles permettent de représenter les attitudes pédagogiques de différentes manières. Chaque technique peut être mise en œuvre par différentes assistances pédagogiques applicables dans l'environnement.

Exemple : pour une technique pédagogique de type « rehaussement », nous pouvons envisager les assistances consistant à « agrandir les ressources physiques » de l'action à effectuer ou à les « rendre plus lumineux ».

4.2.1 - B Situer une règle dans un ensemble

A un niveau d'abstraction donné, une règle est contextualisée par rapport à un ensemble constitué de règles (*cf.* figure 4.3). En effet, les règles dérivent des différentes façons d'aborder une démarche, une attitude ou une technique pédagogique. Il est alors intéressant de gérer les règles par groupe, chacun représentant une démarche, une attitude ou une technique pédagogique particulière. Cette approche permet de décrire un ensemble de règles sans se soucier des autres ensembles.

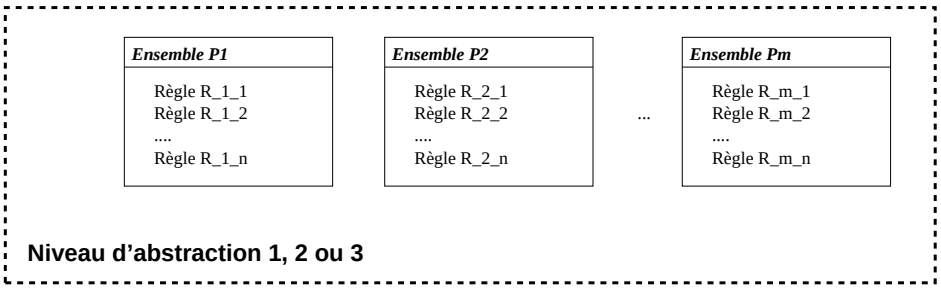


FIGURE 4.3 – Les règles sont regroupées par ensembles.

Exemple d'ensembles : une démarche « active », une attitude qui consiste à « encourager », une technique qui « enrichit l'environnement ».

4.2.2 Liens entre les niveaux d'abstraction

Le raisonnement pédagogique est structuré selon les trois niveaux d'abstraction précédemment décrits. Chaque niveau est constitué de plusieurs ensembles de règles. L'objectif de chaque niveau est de donner les orientations à suivre par le niveau suivant. Il ne s'agit pas d'élire définitivement une orientation plus qu'une autre, mais plutôt d'envisager les différentes possibilités en privilégiant certaines, au fur et à mesure de la réflexion.

Les règles permettent de faire la relation entre les niveaux. La partie effet d'une règle permet de passer à un niveau d'abstraction inférieur, en favorisant un ensemble de règles du niveau inférieur.

Exemple : dans une démarche pédagogique « active » (au niveau 1), une règle est de la forme :
si (l'étudiant est novice) alors (favoriser l'attitude pédagogique « réaliser »)

La figure 4.4 illustre la structure et la dynamique du modèle pédagogique contrôlant le comportement de l'agent pédagogique. Les informations prises en considération dans les parties conditions des règles, sont procurées par notre ITS (situation pédagogique). Ces « entrées » sont disponibles sur les trois niveaux d'abstraction des données (démarches, attitudes et techniques pédagogiques). Les règles, dont la partie condition est satisfaite par rapport aux entrées, favorisent certains ensembles de règles pédagogiques du niveau inférieur. Le dernier niveau (techniques) favorise directement des assistances pédagogiques applicables dans l'environnement. Ces dernières sont proposées au formateur qui choisit celle qui est, selon lui, la plus appropriée. La section suivante montre comment le pédagogue peut spécifier le modèle pédagogique.

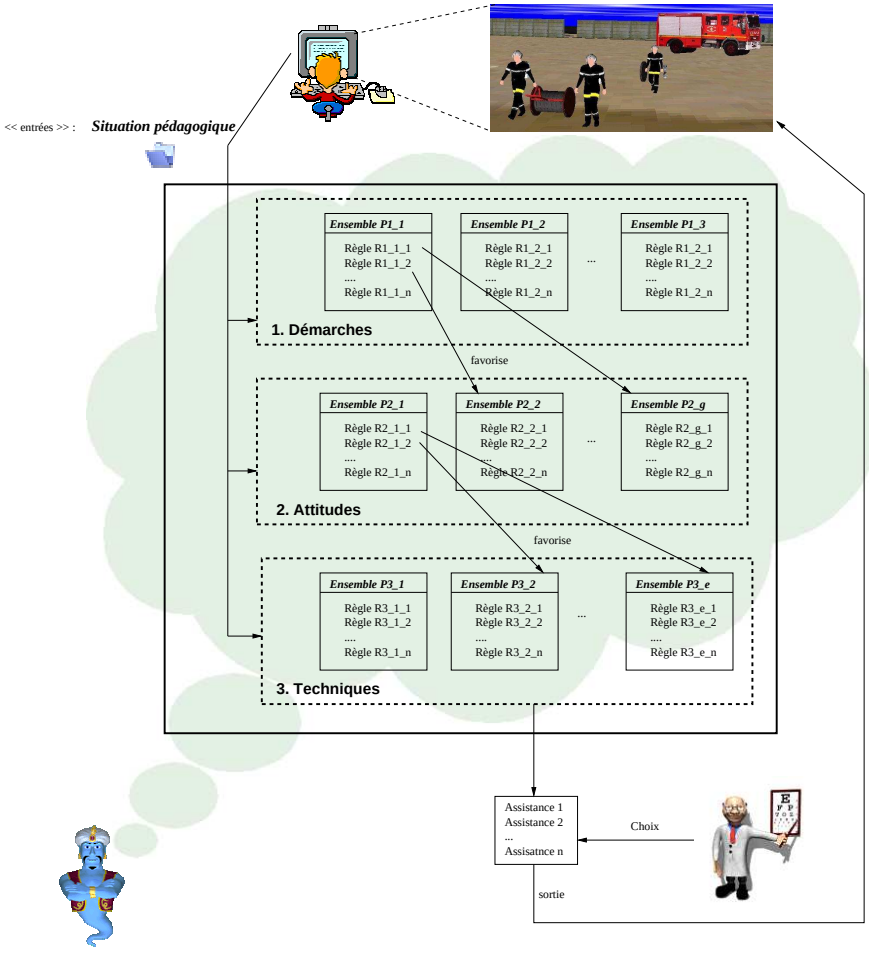


Figure 4.4 – Représentation complète du modèle pédagogique.

4.2.3 Spécification du modèle pédagogique

Pour mettre en œuvre le modèle pédagogique, le travail du pédagogue est de spécifier :

1. Les ensembles de règles pour les trois niveaux d'abstraction.
2. Les règles pédagogiques pour chaque ensemble de règles.

Nous montrons ici des informations issues de la littérature qui peuvent être utilisées pour la spécification du modèle pédagogique.

Spécifier les ensembles de règles pédagogiques

Pour spécifier les ensembles de règles pédagogiques, nous nous basons sur les références (Lourdeaux, 2001; Burkhardt et al., 2003). Nous obtenons les tableaux 4.1, 4.2 et 4.3 correspondant respectivement aux trois niveaux (démarches, attitudes et techniques). Ces informations nous permettent d'effectuer un bilan, en figure 4.5, qui représentent une possibilité de spécification des ensembles de règles des trois niveaux.

Démarches pédagogiques	**Description**
Active / Constructiviste	Les démarches actives sont centrées sur l'apprenant, considérant qu'il est acteur principal de son apprentissage. Elles lui proposent des techniques à travers desquelles il est amené à produire, à créer, à chercher. Le savoir est dans l'environnement.
Expositive / Affirmative	C'est la démarche la plus traditionnelle qui utilise la technique de l'exposé. Elle repose sur une démarche de transmission de contenus. Le savoir est externe.
Interrogative	Elle préconise de diriger l'apprenant vers les solutions recherchées. L'apprenant peut avoir l'impression de découvrir quelque chose, mais c'est toujours le formateur qui conduit la réflexion. Le savoir est interne.

Tableau 4.1 – Exemples de définition des ensembles pour le niveau d'abstraction « Démarches pédagogiques », en se basant sur (Lourdeaux, 2001; Burkhardt et al., 2003).

Spécifier les règles pédagogiques

Les ensembles de règles pédagogiques étant définies, il faut maintenant que le pédagogue spécifie les règles associées. Une règle est représentée par une chaîne de caractères. Les parties condition et effet, portent sur les éléments de la situation pédagogique.

Exemple : « Si l'apprenant est novice et que l'action effectuée est différente de l'action correcte alors favoriser l'attitude expliquer » se traduit par :
⇨ *if (Apprenant.Niveau == novice && Travail.ActionSollicitée in Travail.ActionCorrectes) then (Expliquer)*

Attitudes pédagogiques	Description
Réaliser	Réaliser à la place des formés. Cette stratégie permet au formateur de montrer, par exemple, le bon geste ou la bonne technique.
Perturber	Certains formateurs taquins perturbent les formés, en donnant de mauvaises informations ou des solutions potentiellement fausses, afin de tester la confiance des formés dans leurs raisonnements.
Suggérer	Montrer où les formés peuvent trouver les connaissances théoriques ou bien, où trouver les connaissances sur le terrain. Ces attitudes permettent au formateur de montrer aux formés qu'ils peuvent trouver les connaissances par eux-mêmes, et donc, traiter la situation calmement.
Laisser faire	Elle propose au formateur de ne pas intervenir, mais plutôt de rester en tâche de fond, en tant qu'observateur.
Expliquer	Les explications et les informations ont aussi pour but de permettre tout simplement, d'expliquer le fonctionnement de certains appareils, les règles de raisonnement, les règles de sécurité, *etc.*
Encourager	Encourager les formés lorsqu'ils réalisent correctement la tâche.

Tableau 4.2 – Exemples de définition des ensembles pour le niveau d'abstraction « Attitudes pédagogiques », en se basant sur (Lourdeaux, 2001).

Techniques pédagogiques	Description
Enrichissement	Ajout de symboles visuels, sonores ou de films d'animation.
Dégradation	Détérioration du réalisme (repères effacés, *feed-back* proprioceptifs dégradés, couleurs atténuées, flous à l'arrière-plan et/ou sur les côtés, réduction d'objets, icônisation, *etc.*).
Rehaussement	Exagération de la réalité (représentation d'objets à plus grande échelle, objets surréalistes, plus lumineux, plus brillants, *etc.*).
Simplification	Allègement de la scène virtuelle (une foule peut être représentée par des personnes aux mouvements simplifiés, objets simplifiés, systèmes kinesthésiques simplifiés, représentation en fil de fer, *etc.*), représentation schématique de certains appareils.
Restriction	Limitation de certains déplacements ou manipulations (limitations du périmètre dans lequel l'utilisateur peut se déplacer, *etc.*)
Animation	Séquence animée (positionnement automatique, clef qui tourne automatiquement une fois mise en place, *etc.*).
Décentrage	Changement du point de vue habituellement attaché à l'œil du formé immergé (vue de derrière, au-dessus, *etc.*).
Modification	Modification d'aspect, de texture (changement de couleurs, clignotement d'objets, *etc.*).
Modélisation	Représentation de concepts abstraits, de phénomènes physiques invisibles à l'œil nu, de type de pannes, *etc.*
Visualisation	Mécanismes cachés (intérieur d'un moteur, engrenages, *etc.*).

Tableau 4.3 – Exemples de définition des ensembles pour le niveau d'abstraction « Techniques pédagogiques », en se basant sur (Lourdeaux, 2001).

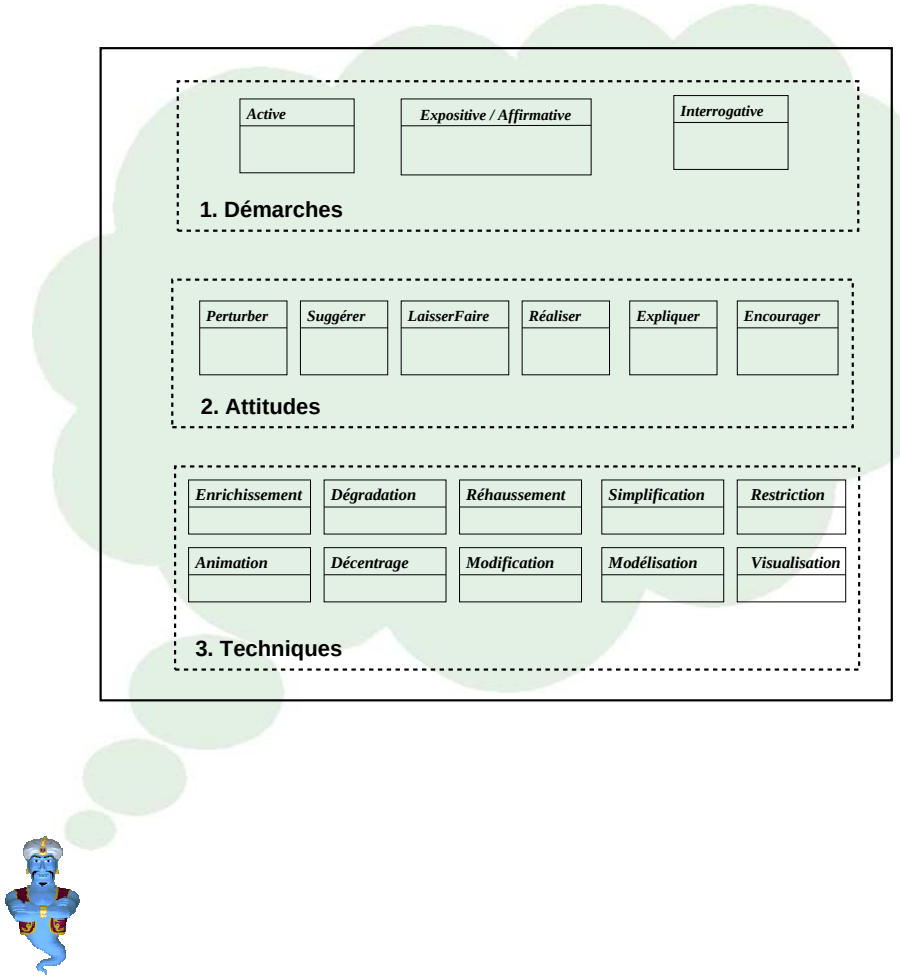


Figure 4.5 – Spécification des trois niveaux du modèle de décision d'intervention pédagogique.

Rappelons que les connaissances représentées portent sur l'apprenant (noté `Apprenant`) ou sur le travail à réaliser (noté `Travail`). Les règles accèdent aux connaissances de la situation pédagogique par une syntaxe particulière. Par exemple, nous écrivons « Apprenant.Niveau » pour récupérer la connaissance sur le niveau de l'apprenant. Chaque attribut de la situation pédagogique défini dans le chapitre 3 est manipulable.

Cette section a montré les principes nous permettant de simuler un raisonnement pédagogique sans pour autant détailler les mécanismes[5]. La section suivante va plus loin en présentant l'implémentation d'un tel modèle, elle précise alors les détails liés à l'exécution.

4.3 Implémentation du modèle pédagogique

Cette section décrit l'implémentation du modèle pédagogique. Nous considérons le choix des systèmes de classeurs comme architecture comportementale effectué en section 4.1, et nous utilisons la définition du modèle pédagogique de la section 4.2.

4.3.1 Systèmes de classeurs

Cette section propose une vue d'ensemble des systèmes de classeurs. Nous nous limitons ici à l'abord des éléments qui sont utiles pour la compréhension de notre modèle. Pour aller plus loin, nous proposons au lecteur une présentation plus complète des systèmes de classeurs en annexe B.

Les systèmes de classeurs sont des architectures de contrôle pour la prise de décision, qui utilisent un apprentissage par renforcement. Ils sont utilisés, par exemple, pour effectuer des diagnostics médicaux (Bonelli et al., 1990) ou élaborer des stratégies dans le combat aérien (Smith et al., 1999). En économie, ils ont permis d'étudier des scénarios de négociation (achat/vente d'actions) et d'apprendre des stratégies économiques (Schulenburg et Ross, 2001). Ils constituent pour Girard et al. (2001) le mécanisme décisionnel de vaisseaux spatiaux. Sanza (2001) les utilise pour contrôler la sélection d'actions d'entités virtuelles coopératives appliquées dans un jeu de football virtuel où chacun des 22 acteurs possède un système de classeurs. Enfin, Robert (2005) les implique dans la sélection d'action de personnages de jeu en ligne persistant et massivement multi-joueurs. Cette liste d'exemples, non exhaustive, permet d'évaluer le large champ d'application de ces systèmes.

Les premiers modèles sont proposés par Holland (1976). La première implémentation, appelée CS1, apparaît en 1978 (Holland et Reitman, 1978) : une situation ou « état perçu » est stockée dans une mémoire perceptive de taille limitée, assimilée à la « mémoire à court terme » des sciences cognitives (appelée liste des messages). Une base de règles (« condition-action ») constitue l'équivalant de la mémoire à long terme. A chaque cycle de fonctionnement,

[5] Notamment au niveau du lien entre les ensembles de règles des différents niveaux.

un mécanisme d'appariement entre les conditions des règles et les perceptions permet d'identifier la ou les actions pouvant être appliquées (action(s) stockée(s) dans la liste des messages). L'exemple support proposé porte sur l'exploration d'un labyrinthe, au sein duquel se trouve de la nourriture qui rapporte des points. L'objectif est d'apprendre en ligne (ou par expérimentation) l'association de conditions et d'actions maximisant la récolte de points. Pour éviter l'explosion combinatoire du nombre de règles possibles, Holland utilise des algorithmes génétiques (Holland, 1975; Goldberg, 1989) recherchant des règles généralisantes. Le système devient adaptatif grâce à l'ajout d'un paramètre préférentiel dynamique (souvent appelé *force* noté f) à chacun des classeurs. Les forces des règles sont alors modifiées dynamiquement par le biais d'un apprentissage par renforcement. Les règles de force importante ont plus de chance d'être choisies par le système, tandis que les règles de faible force sont détruites par l'algorithme génétique.

La figure 4.6 synthétise les interactions entre un système de classeurs et l'environnement. Nous introduisons ici deux interfaces qui sont utilisées dans notre description : l'interface d'entrée, qui permet de représenter une perception de l'état de l'environnement et celle de sortie qui permet d'exécuter l'action choisie et, donc, d'agir sur l'environnement. Elle montre également les flux d'information : l'opération d'appariement, l'élection d'action, le renforcement et l'adaptation par algorithmes évolutionnistes.

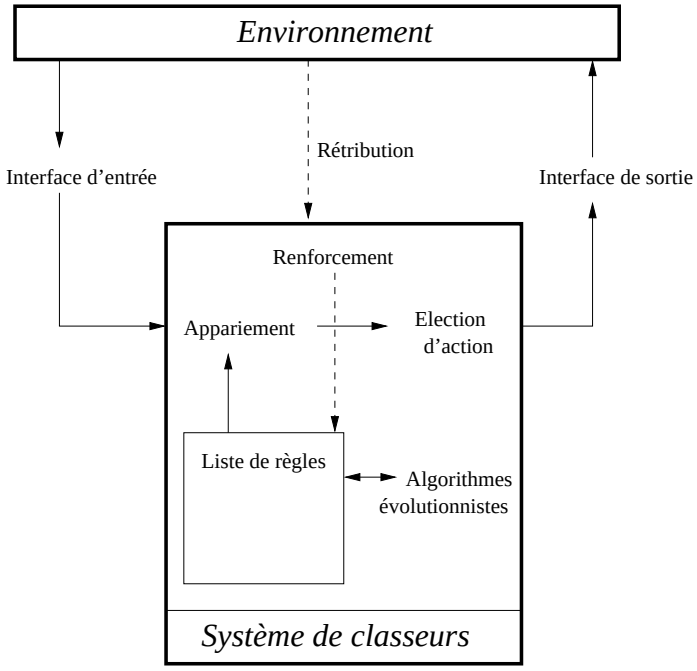


Figure 4.6 – Interactions entre l'environnement et le système de classeurs.

La figure 4.7 précise notre utilisation d'un système de classeurs. L'interface d'entrée est constituée des connaissances contenues dans la situation pédagogique. L'interface de sortie vise des assistances pédagogiques[6]. Le renforcement dépend du choix du formateur.

[6] Nous verrons par la suite que nous nous plaçons dans le cadre d'une architecture hiérarchique, aussi, selon la position du système de classeurs dans la hiérarchie, l'interface de sortie n'élit pas forcément directement les assistances pédagogiques. Dans tous les cas, elle contribue à cette élection.

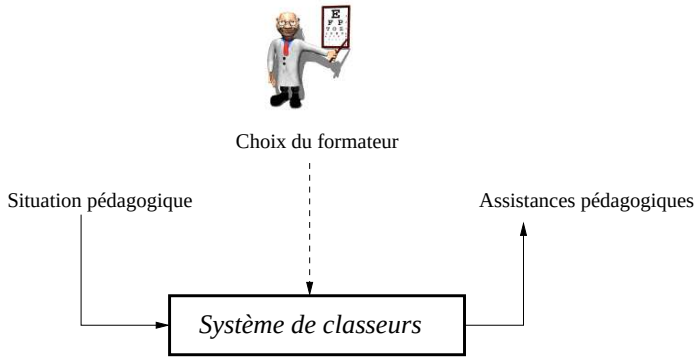


Figure 4.7 – Interactions entre l'environnement de formation et le système de classeurs.

4.3.1 - A Mécanismes de base

Structure de base

La structure d'un système de classeurs, présentée en figure 4.8, est constituée de différents ensembles jouant un rôle dans le choix de l'action à réaliser relative à une situation donnée. Formellement, un système de classeurs est un septuplet (`Ie`, `[P]`, `[M]`, `[A]`, `Comparaison`, `Sélection`, `Io`) :

- `Ie` est l'interface d'entrée, faisant correspondre à toute `Perception` de l'environnement une représentation interne.
- `[P]`, appelé *population*, est l'ensemble des classeurs du système. Les représentations généralisantes contiennent des symboles # correspondant à une valeur indéterminée d'un élément perçu. Une règle est un couple (C, A) avec :
  - C : la condition d'application de la règle.
  - A : la ou les actions associées à l'application de la règle.
- `[M]` $\subseteq$ `[P]` est l'ensemble des classeurs dont la partie condition s'apparie avec les informations perçues de l'environnement pour un cycle de sélection. Il est appelé *Match-set*.
- `[A]` $\subseteq$ `[M]` est l'ensemble des classeurs représentant l'action sélectionnée. Il est appelé *Action-set*.
- `Comparaison` est le mécanisme permettant de passer de `[P]` à `[M]`. Il s'agit généralement d'une règle d'appariement entre C et l'information provenant de `Ie`.
- `Sélection` est le mécanisme permettant de passer de `[M]` à `[A]`. Il détermine, en fonction de critères spécifiques, l'action choisie.
- `Io` est l'interface de sortie faisant correspondre à une représentation interne l'activation d'`Action`.

Chaque type de système de classeurs définit ses propres mécanismes de `Comparaison` et de `Sélection`.

Dynamique d'un système de classeurs

Les différents mécanismes impliqués dans la dynamique d'un système de classeurs, s'enchaînent cycliquement selon l'ordre `Perception` / `Comparaison` / `Sélection` / `Action`. Ce

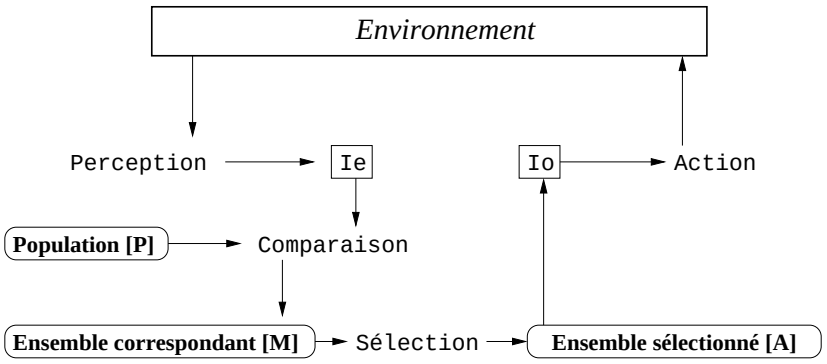


Figure 4.8 – Structure et dynamique d'un système de classeurs.

cycle est répété et peut être soumis à un critère de terminaison (temps maximal de résolution du problème par exemple).

Introduction de l'apprentissage

L'apprentissage des systèmes de classeurs repose sur les algorithmes d'apprentissages par renforcement (Sutton, 1991). Cet apprentissage est élaboré par essai-erreur et se classe donc parmi les apprentissages dits « non supervisés » (Reignier, 1994), *i.e.* qui ne nécessitent ni une série d'exemples à apprendre comme dans les réseaux de neuromimétiques de type *Perceptron* (McCulloch et Pitts, 1943a) ni une élaboration d'une fonction de *« fitness »* indiquant précisément la *qualité* d'une solution, comme c'est le cas avec les algorithmes génétiques (Goldberg, 1989). Plus précisément, les systèmes de classeurs utilisent des algorithmes d'apprentissage par différence temporelle (Klopf, 1972; Sutton, 1984, 1988) qui unifient les théories de l'apprentissage par renforcement.

L'apprentissage s'effectue grâce à une évaluation de la qualité des règles, représentée par un ou plusieurs paramètres supplémentaires. La définition d'un classeur est donc étendue à un triplet $R = (C, A, f)$ où f caractérise la qualité de ce dernier. C'est ce critère qui sert de base à la sélection. Plus f est élevée, plus la règle est considérée pertinente. L'apprentissage est réalisé par une `Rétribution` des règles, modifiant leur qualité grâce à des algorithmes d'apprentissage par renforcement et par `Génération` de règles à l'aide d'algorithmes évolutionnistes et d'heuristiques de recouvrement (*covering*). La dynamique des systèmes de classeurs apprenants est alors basée sur le cycle : `Perception` / `Comparaison` / `Génération` *(covering)* / `Sélection` / `Action` / `Rétribution` / `Génération` (algorithme évolutionniste).

4.3.1 - B Principaux paramètres et mécanismes

Paramètres des classeurs

Pour chaque classeur, des paramètres permettant de définir la qualité de la règle sont définis. Ils permettent de préciser les mécanismes de `Sélection`, de `Rétribution` et de `Génération`. Une valeur de *force* f est généralement associée à une règle. Elle est identique à

la création de tous les classeurs et évoluant au cours de l'apprentissage. Ainsi

$$R = (C, A, f) \tag{4.1}$$

Dans ce cas, la fonction d'utilité définie en Q-learning peut être assimilée à la force des classeurs. D'autres critères, tels que la *spécificité* s (proportionnelle au nombre de # que contient le classeur) ou la prédiction de paiement, sont utilisés.

Mécanisme de `Sélection`

Le mécanisme de `Sélection` permet d'effectuer le passage de l'ensemble de classeurs appariés `[M]` à l'ensemble `[A]`. Les classeurs de `[M]` sont regroupés par paquets, chacun étant constitué de règles dont la partie *Action* s'apparie avec les autres règles du même paquet. Une combinaison des paramètres des classeurs de chaque paquet est utilisée pour définir leur sélectivité. Ici, il s'agit d'une fonction des f de chaque classeur. En phase d'exploitation, ce critère est utilisé directement pour sélectionner l'ensemble `[A]`. En phase d'exploration, un mécanisme de roue de la fortune[7] est généralement appliqué, ce qui signifie que chaque paquet a une probabilité proportionnelle à sa sélectivité d'être sélectionné.

Mécanisme de `Rétribution`

La rétribution, appelée *credit assignment*, modifie les forces des classeurs existants dans la population `[P]`. La répartition de la rétribution de l'environnement favorise ou défavorise les règles selon leur efficacité. Le problème de la rétribution revient à décider quelles sont les règles qui, à un temps t, ont été nécessaires et suffisantes pour atteindre le but à un temps $t + n$, sachant que différentes règles ont été actives à chaque pas. Dans la recherche de but dans un labyrinthe, on peut se demander quels mouvements précédents ont permis d'atteindre le but. La plupart des algorithmes modifient la force des classeurs.

Mécanisme de `Génération`

Un des objectifs des systèmes de classeurs apprenant est de limiter le nombre de règles en supprimant les moins efficaces mais également en cherchant des meilleures. Les meilleures règles sont celles qui sont les plus généralisantes possibles tout en ayant une force maximale. Pour cela deux mécanismes de génération, appelée *rule discovery*, sont utilisés ; il s'agit du *covering* et des algorithmes génétiques :

1. Le *covering* permet de créer des règles lorsque aucun classeur ne s'apparie à la perception de l'environnement. Cela signifie que la population `[P]` ne possède pas un nombre suffisant de règles permettant de proposer une action relative à la situation. Le *covering* peut également créer des règles lorsque les qualités des classeurs appariés sont considérées insuffisantes. Dans les deux cas, de nouvelles règles sont créées avec une partie condition(s) adéquate(s) et plus ou moins généralisante.
2. Les *algorithmes génétiques* (Holland, 1975; Goldberg, 1989) sont utilisés comme algorithmes évolutionnistes pour générer de nouvelles règles à partir de celles existantes.

[7] Le mécanisme de roue de la fortune est une métaphore où il faut imaginer une roulette de casino sur laquelle est placé chacun des éléments sélectionnables, la place accordée à chacun étant proportionnelle à sa sélectivité. Ensuite la bille est lancée et l'endroit où elle s'arrête indique l'élément sélectionné.

Le mécanisme de suppression est simple puisqu'il s'agit d'éliminer les individus les « moins bons ». La qualité d'un individu reste sujette à débat. Elle dépend des types d'application.

Nous justifions plus loin pourquoi, dans notre cas, nous n'utilisons pas de mécanisme de génération.

4.3.1 - C Systèmes hiérarchiques

Bien que l'utilisation des systèmes de classeurs classiques produit des résultats intéressants, les modèles souffrent d'un problème de granularité de représentation lorsqu'il s'agit d'établir des connaissances mêlant des informations de portées sémantiques différentes (Barry, 1993). En effet, l'hétérogénéité des données peut impliquer des niveaux de décision différents, avec éventuellement des échelles de temps différents. L'introduction de structures dans les systèmes de classeurs permet d'abstraire (tâches simples et tâches complexes), de décomposer (décomposer un problème complexe pour le résoudre plus facilement) et de réutiliser (solutions des sous-problèmes) les connaissances (Barry, 1996), pour améliorer leurs performances. C'est ce que proposent les systèmes hiérarchiques. Ils ont notamment été utilisés dans les systèmes Alecsys (Dorigo, 1995), Ocs (Takadama et al., 1999) et Mhics (Robert et Guillot, 2003).

4.3.2 Adaptation des systèmes de classeurs au modèle pédagogique

En début de chapitre, nous avons choisi de nous baser sur les systèmes de classeurs pour modéliser l'architecture comportementale de l'agent pédagogique (*cf.* section 4.1). Ensuite, nous avons décrit le comportement pédagogique comme étant une structure hiérarchisée (*cf.* section 4.2). Les systèmes de classeurs *hiérarchiques* constituent alors une solution adaptée à notre problématique. Cette section présente l'adaptation des systèmes de classeurs à la structure imposée par la section 4.2.

4.3.2 - A Structure

Notre modèle est représenté sous la forme d'un diagramme de classe UML en figure 4.9.

L'agent pédagogique (`PedagogicalAgent`) possède pour base de connaissances les règles pédagogiques (`PedagogicalRules`) définies par le pédagogue (règles du type « si-alors »). Le comportement simulant le raisonnement pédagogique (`PedagogicalDecisionBehavior`) utilise le modèle pédagogique (`PedagogicalModel`).

Le modèle pédagogique est composé de trois niveaux. Chaque niveau (`Classifier-SystemSet`) est constitué de plusieurs ensembles de règles pédagogiques, *i.e.* plusieurs systèmes de classeurs.

Un ensemble de règles pédagogiques est implémenté sous la forme d'un système de classeurs (`CS_System`). Le modèle proposé utilise le modèle générique de système de classeurs

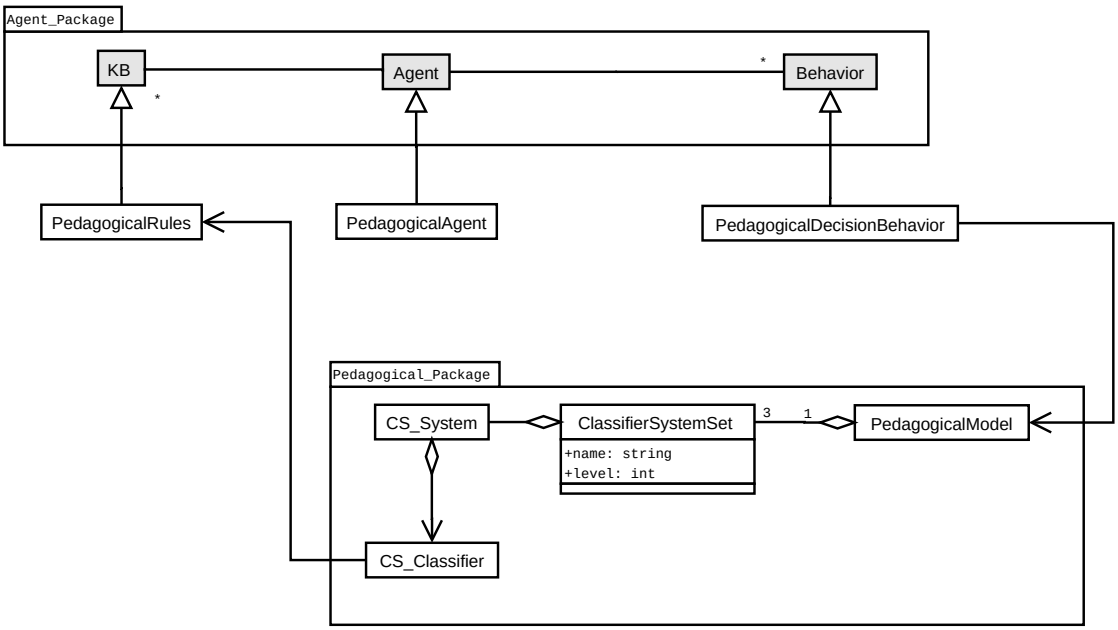


Figure 4.9 – Diag. de classe UML du modèle pédagogique.

présenté en annexe C[8]. Son comportement est contrôlé par des classeurs (`CS_Classifier`), représentant les règles pédagogiques.

4.3.2 - B Dynamique

Les informations concernant la structure des niveaux, ont été définies, il s'agit de choisir un mécanisme qui va spécifier la dynamique de notre système.

Une liaison par diffusion

Nous souhaitons que les règles, dont la partie condition est satisfaite, favorisent certains ensembles de règles des niveaux inférieurs. La liaison entre les différents niveaux s'effectue alors par un mécanisme de diffusion, dans la lignée des architectures dites de « hiérarchies à libres flux » (FFH)[9]. Celles-ci s'opposent aux architectures dites hiérarchiques « gagnant prend tout » (WTA)[10]. Les principes de ces deux architectures[11] sont représentés en figure 4.10 :

- ▷ Dans un WTA, le premier niveau diffuse des activations diverses dans les différents éléments du second niveau. Seul l'élément recevant l'activation la plus forte pourra

[8] Le modèle générique de système de classeurs utilisé ici fait suite à des travaux que nous avons effectués, mais qui n'ont pas de rapport direct avec cette thèse. Le lecteur trouvera une description complète ainsi que les résultats obtenus sur diverses applications dans (Buche et al., 2006).

[9] FFH : *Free-Flow Hierarchy* (Rosenblatt et Payton, 1989).

[10] WTA : *Winner-Takes-All* (Tinbergen, 1951).

[11] Tyrrell (1993) a réalisé une comparaison entre ces deux types d'architectures pour le contrôle de comportement d'entité autonome. Il montre que les WTA présentent plusieurs limitations, dont l'une est d'empêcher les comportements de compromis résultants de motivations différentes.

diffuser une activation dans les éléments du niveau inférieur. Cette approche met en compétition les éléments, chaque niveau n'en garde qu'un parmi l'ensemble des possibilités.

▷ Dans une architecture FFH, plusieurs éléments du second niveau pourront envoyer leurs activations aux niveaux inférieurs. Les éléments sont en coopération, cette approche permet de combiner leurs effets.

Nous souhaitons que les règles favorisent les ensembles de règles des niveaux inférieurs, sans forcément élire un ensemble vainqueur. En effet, nous allons faire des « propositions » au formateur qui peuvent être issues de différents ensembles. Les architectures WTA ne sont donc pas appropriées.

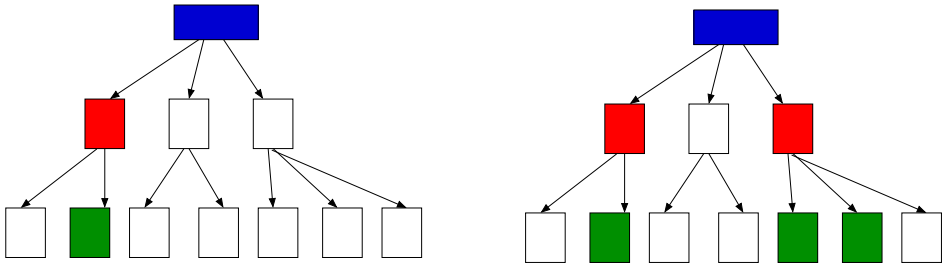


Figure 4.10 – Architectures hiérarchiques « gagnant prend tout » (WTA, gauche) et hiérarchique à « libres flux » (FFH, droite) à trois niveaux de hiérarchies.

Formalisation

Nous formalisons ici le modèle pédagogique. Nous nous efforçons de conserver les notations utilisées classiquement dans les systèmes de classeurs. Les éléments mis en jeu sont les règles et les ensembles de règles.

1. *Règle* (R)
   Une règle est définie par sa partie condition (`C`) et sa partie effet (`A`) sous la forme de chaînes de caractères.
   $$\mathtt{R} = (\mathtt{C}, \mathtt{A}) \tag{4.2}$$

2. *Ensemble de règles* (CS)
   Chaque règle est contenue dans un ensemble. La partie effet d'une règle pointe vers un des ensembles du niveau inférieur. On associe à chaque ensemble de règles un paramètre appelé intensité. Il permet de mettre en place le mécanisme de diffusion du flux dans le système. Enfin, un ensemble de règles est situé dans un niveau d'abstraction. Formellement, un ensemble est défini par p règles (`R`), une intensité (`intensity`) et un niveau (`level`).
   $$\mathtt{CS} = (\{\mathtt{R}_1, \ldots, \mathtt{R}_p\}, \mathtt{intensity}, \mathtt{level}) \tag{4.3}$$

   La dynamique, de type FFH, met à jour les intensités des ensembles de règles dans les niveaux successifs. L'intensité représente l'activation de l'ensemble de règles qui résulte de la diffusion. Plus il existe de règles dont la partie condition est satisfaite et dont la partie effet pointe vers un ensemble, plus l'intensité de cet ensemble est grand.

Les mécanismes

Le cycle itératif effectué pour chaque niveau est décrit en figure 4.11.

```
t := 0 ;
while not done {
   t := t + 1 ;
   for each level              // Du niveau 1 au niveau 3
   {
       for each CS
      {
          // 1. Perception
          Ie(t) := readDetectors(t) ;
          // 2. Intensité
          intensity := computeIntensity(t) ;
          // 3. Appariement
          M(t) := matchClassifiers( Ie(t),P(t),Comparaison ) ;
          // 4. Actions
          Io(t) := sendEffectors( M(t) ) ;
      }
   }
}
```

Figure 4.11 – Dynamique du modèle pédagogique.

Chaque `CS` effectue les opérations suivantes :

1. *Perception.*
   Le système perçoit les informations de la situation pédagogique. Rappelons que, dans notre cas, la situation pédagogique est constituée de connaissances qui portent sur le travail à réaliser et sur l'apprenant. Les connaissances utilisées sont celles qui ont été présentées dans les tableaux 3.4 et 3.5 du chapitre 3.

$$\texttt{Ie} = (\texttt{Travail}, \texttt{Apprenant})$$

$$\textit{avec} \begin{cases} \texttt{Travail} = (\ \texttt{Contexte d'action précédente, sollicitée, correctes, correctes sans rôle, suivantes, liées logiquement}\ ) \\ \texttt{Apprenant} = (\ \texttt{Informations générales, Physiologique, Psychologique, Contextes d'Actions effectuées, Erreurs effectuées, Image mémorielle, Déplacement}\ ) \end{cases} \tag{4.4}$$

2. *Mise à jour de l'intensité.*
   Elle s'effectue en mettant à jour le paramètre `intensity`, en considérant les effets des règles appariées du niveau en amont. L'intensité de chaque ensemble augmente, en considérant les intensités des ensembles qui contiennent les règles appariées du niveau en amont favorisant cet ensemble. L'intensité de chaque ensemble du niveau 1 (le plus en amont) a pour valeur initiale 1 (tous les ensembles sont potentiellement activables). Pour les autres niveaux, il s'agit de sommer les intensités obtenues par le mécanisme de diffusion.

3. *Appariement.* ⇨ Déterminer les règles concernées.
   Chaque système de classeurs effectue les opérations d'appariements (mécanisme de `Comparaison`), afin de déterminer quelles sont les règles satisfaites (passage de `[P]` à `[M]`). Concrètement, chaque règle utilise les connaissances de la situation pédagogique pour vérifier sa partie condition.

4. *Diffusion.*
   Les classeurs de `[M]` émettent des messages pour favoriser les systèmes de classeurs des niveaux en aval en utilisant l'intensité de l'ensemble de règles associé. Concrètement, chaque règle de `[M]` dont la partie condition est satisfaite, transmet la valeur de l'intensité de son ensemble (celui qui la contient) à l'ensemble pointé par sa partie effet.
   Une fois que la diffusion est terminée, le résultat au niveau 3 (le plus en aval) est récupéré. Plus précisément, il s'agit du couple (règle) - (intensité de l'ensemble contenant la règle). Les intensités permettent d'ordonner les propositions d'assistances qui sont présentées au formateur.

Exemple de diffusion

La figure 4.12 illustre le mécanisme de diffusion que nous venons de définir. Les règles sur-lignées sont les règles appariées dans la situation courante.

Dans cet exemple, trois règles appariées au niveau 1 proposent l'attitude « réaliser ». Par conséquent, l'intensité du `CS` « réaliser » est mise à jour avec la valeur 3 (= 1+1+1). Au niveau 2, une règle propose la technique « simplification » (provenant de l'attitude « montrer » intensité = 1) et deux règles proposent la technique « modification » (une provient de l'attitude « montrer » intensité = 1 et l'autre de l'attitude « réaliser » intensité = 3). Par conséquent, la technique « simplification » obtient une intensité de 1 alors que la technique « modification » récupère une intensité de 4 (3+1). Finalement, l'assistance « RendreTransparent y » est proposée par la technique « simplification » (intensité = 1) et deux fois par la technique « modification » (intensité = 4), elle obtient alors un poids de 9 (1+4+4). L'assistance « ChangerTexture de x » a obtenu un poids de 4. Ainsi, les propositions effectuées au formateur pour cette situation sont donc :

① RendreTransparent y
② ChangerTexture de x

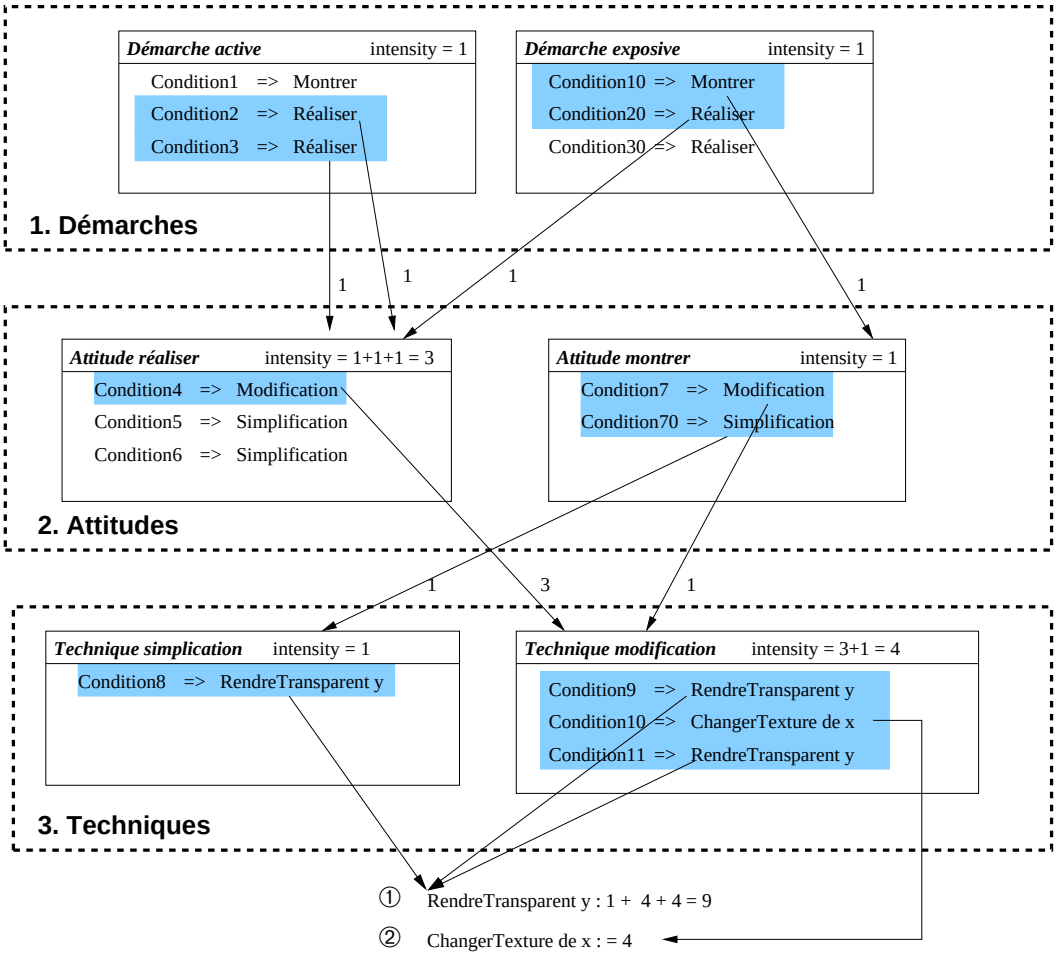


Figure 4.12 – Exemple de diffusion dans le modèle pédagogique.

Bilan

Nous avons défini un modèle simulant un raisonnement pédagogique, constitué de trois niveaux. Chaque niveau possède des ensembles de règles. Une règle participe au raisonnement si sa partie condition est satisfaite, et si l'intensité de l'ensemble qui la contient est non nulle. Les règles permettent de faire les liaisons entre les niveaux : elles influencent les activations des ensembles des niveaux inférieurs.

Utilisant les règles du niveau le plus en aval, le système propose au formateur des assistances pédagogiques, celles-ci sont ordonnées par ordre de préférence *a priori* (résultat de la diffusion des intensités dans la structure). Le formateur ne choisit pas forcément l'assistance pédagogique qui lui est proposée en premier choix.

Ce modèle est intéressant puisqu'il assiste le formateur dans deux domaines :

1. en *pédagogie*, puisque le modèle simule un raisonnement pédagogique adapté à la situation pédagogique en considérant les connaissances fournies par le pédagogue ;
2. en *réalité virtuelle*, puisque le modèle propose des assistances pédagogiques utilisant toutes les potentialités de la réalité virtuelle en terme de formation.

Nécessité de l'apprentissage artificiel

Nous nous plaçons dans le cadre d'un environnement multi-apprenants. Le formateur a la responsabilité de plusieurs étudiants qui agissent en parallèle. Il peut difficilement suivre tous les apprenants et choisir les assistances pédagogiques à mettre en place. Pour que chaque étudiant puisse bénéficier d'une aide pédagogique, nous proposons un mode où les assistances pédagogiques sont effectuées automatiquement. Dans ce cas, le système se substitue au formateur en sélectionnant l'assistance placée en première position dans la liste ordonnée de propositions. Ce mode est activable / désactivable à tout moment, le formateur peut alors suivre un apprenant lorsqu'il a des difficultés importantes et passer la main au système par la suite.

Néanmoins, dans ce contexte, notre modèle présente une limitation majeure : si les propositions d'assistances pédagogiques proposées sont pertinentes, puisqu'elles répondent aux situations définies par le pédagogue, l'ordre des propositions est sujet à discussion, puisqu'il n'est pas totalement spécifié par le pédagogue. En effet, celui-ci ne peut tenir compte de la dynamique du modèle lorsqu'il spécifie les règles pédagogiques. Cela impliquerait qu'il anticipe toutes les combinaisons de scénario sur la dynamique du modèle pour prévoir les propositions qui vont intervenir dans telle ou telle situation. Par conséquent, il devrait bouleverser toute sa spécification lorsqu'il souhaite modifier l'espace des règles, ce qui va à l'encontre d'un modèle modulable. D'autre part, certaines interventions pédagogiques sont plus efficaces sur un apprenant que sur un autre, sans pour autant qu'il y ait des différences dans leur profil. En effet, même si les règles pédagogiques prennent en compte des caractéristiques des apprenants, ces derniers sont, par définition, incomplets et ne peuvent représenter qu'un profil général. Ainsi, l'ordre des propositions peut être non optimal. De plus, nous pouvons remarquer que cet ordre reste constant pour une situation donnée. A aucun moment cet ordre n'est remis en cause, même si les propositions en début de liste ne sont pas plus adaptées. Or, dans le mode automatique, l'assistance sélectionnée est celle en première position.

Il s'agit de répondre à cette limitation. Une solution est d'intégrer un mécanisme d'évolution permettant d'auto-adapter le modèle en cours de simulation. Ce mécanisme se base sur les choix du formateur.

La figure 4.13 montre une vue générale du système. Chaque apprenant dispose d'un modèle pédagogique qui lui est personnel. A tout moment, le formateur peut entrer dans la boucle est prendre le contrôle de ceux-ci. Dans ce cas, c'est lui qui choisit parmi les propositions d'assistances pédagogiques. Le système tient compte des choix du formateur et adapte le modèle pédagogique concerné. Dans l'exemple de la figure, le formateur a pris en charge la formation de l'étudiant 3. La section suivante présente l'intégration du mécanisme d'adaptation.

4.3.2 - C Apprentissage artificiel

Dans un premier temps, nous explicitons l'objectif particulier de la mise en place du mécanisme d'apprentissage artificiel. Ensuite, nous analysons les critères à prendre en compte dans notre cadre d'étude. Cette réflexion nous permet de choisir un apprentissage par renforcement de type *bucket brigade*, que nous décrivons et que nous mettons en œuvre.

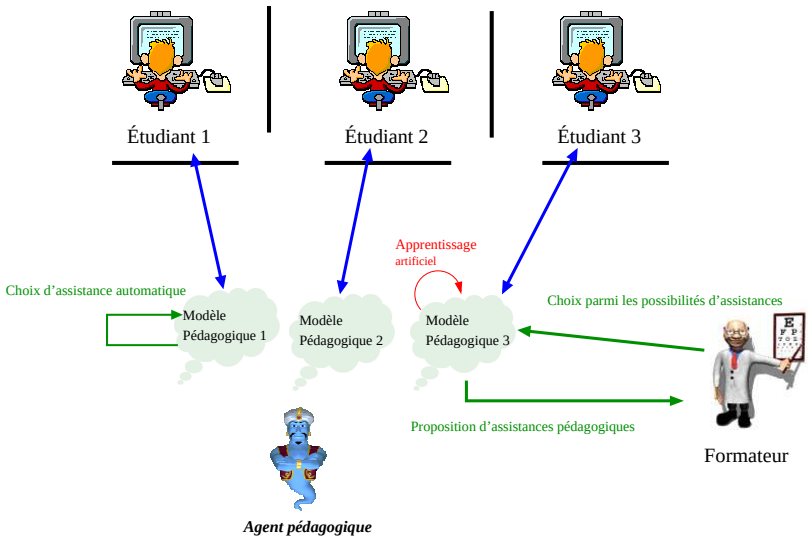


Figure 4.13 – Vue d'ensemble du système utilisant l'apprentissage artificiel.

Apprendre quoi ?

Améliorer le modèle

Nous proposons de tenir compte des choix du formateur relativement au comportement de l'apprenant, et d'adapter le modèle pédagogique en conséquence. Le système doit favoriser les règles qui ont permis d'aboutir au résultat choisi par le formateur. L'objectif de l'apprentissage artificiel est alors d'adapter les valeurs de paramètres représentants les forces des règles par expérience (la diffusion est alors un combinaison des forces des règles et des intensités des ensembles de règles). Pour une situation donnée, il s'agit de favoriser les chemins qui ont permis d'aboutir au choix du formateur à travers le réseau.

Générer le modèle

En apprentissage artificiel, il est possible de générer une partie ou la totalité d'un modèle. Pour cela, les mécanismes de *covering* et les algorithmes génétiques sont utilisés par les systèmes de classeurs classiques. Nous pensons que la découverte de nouvelles règles dans notre cas n'est pas souhaitable. En effet, une génération pseudo-aléatoire de règles va enrichir ou modifier la base de règles définie par le pédagogue. Dans le premier cas, le formateur obtiendra un nombre croissant de propositions. Il passera alors plus de temps à choisir parmi les propositions et l'intervention pédagogique sera moins rapidement mise en place dans l'environnement. De plus, les règles créées seront, en partie, incohérentes (génération aléatoire). En attendant que ces dernières soient éventuellement supprimées par un autre mécanisme, le raisonnement du système sera faussé. Dans le second cas, le mécanisme d'apprentissage risque d'éliminer des règles que le pédagogue a spécifiées, alors qu'elles sont probablement plus pertinentes à court terme que celles obtenues aléatoirement.

Caractéristiques de l'apprentissage artificiel

Nous avons choisi les systèmes de classeurs hiérarchiques. Il s'agit d'intégrer un mécanisme d'apprentissage permettant de faire évoluer le modèle. Afin de déterminer l'algorithme que nous allons utiliser, il convient de préciser les caractéristiques attendues.

1. L'adaptation du comportement de l'agent pédagogique doit s'effectuer en cours de simulation. En effet, il s'agit de s'adapter dynamiquement au couple apprenant-formateur pour modifier dynamiquement le raisonnement pédagogique, si ce dernier ne convient pas. L'apprentissage doit être en ligne.
2. Le raisonnement pédagogique et son adaptation doivent prendre un minimum de temps. En effet, l'agent pédagogique doit réagir le plus rapidement possible pour être efficace. L'apprentissage doit être réactif.
3. L'objectif est de modifier les forces des classeurs qui ont participé à la proposition d'une assistance pédagogique, il s'agit d'un apprentissage numérique[12].
4. Le formateur a un rôle d'évaluateur. Il fournit une information que l'on peut considérer comme étant une rétribution, c'est donc un apprentissage par renforcement.
5. L'objectif de l'apprentissage est de favoriser un chemin dans le réseau pour une situation donnée : il s'agit alors de favoriser l'enchaînement de règles.

Algorithme du bucket brigade

L'algorithme du *bucket brigade*[13] est un algorithme classiquement utilisé dans les systèmes de classeurs (Holland, 1986; Wilson, 1987). L'algorithme propose un apprentissage modifiant les forces des règles ayant participées à la récolte du renforcement. Il permet d'apprendre en ligne et indépendamment du domaine d'application. De plus, l'apprentissage est réactif, contrairement à un apprentissage du type rétropropagation du gradient. Enfin, cet algorithme favorise l'enchaînement de règles. Il présente donc les caractéristiques dont nous avons besoin.

Les sous-sections qui suivent ne présentent pas tout de suite l'utilisation de cet algorithme dans notre architecture à trois niveaux. En effet, nous décrivons dans un premier temps l'algorithme du *bucket brigade*, sans nous attacher aux particularités liées à notre architecture. Nous présentons ensuite son adaptation à notre modèle hiérarchique.

Principe

L'algorithme du *bucket brigade* s'applique aux systèmes de classeurs contenant une liste de messages (*cf.* figure 4.14). Ces systèmes élisent des actions (appliquées dans l'environnement) ou provoquent des messages internes entraînant des cycles d'inférences internes. Toutes les règles appariées contribuent à l'inférence courante. Une fois l'action effectuée dans l'environnement, un algorithme de distribution des crédits assure l'apprentissage par renforcement, en répartissant équitablement le gain parmi différents classeurs successivement activés.

[12] La distinction entre un apprentissage *numérique* et un apprentissage *symbolique* est précisée en annexe A.

[13] *Bucket brigade* : chaîne des seaux.

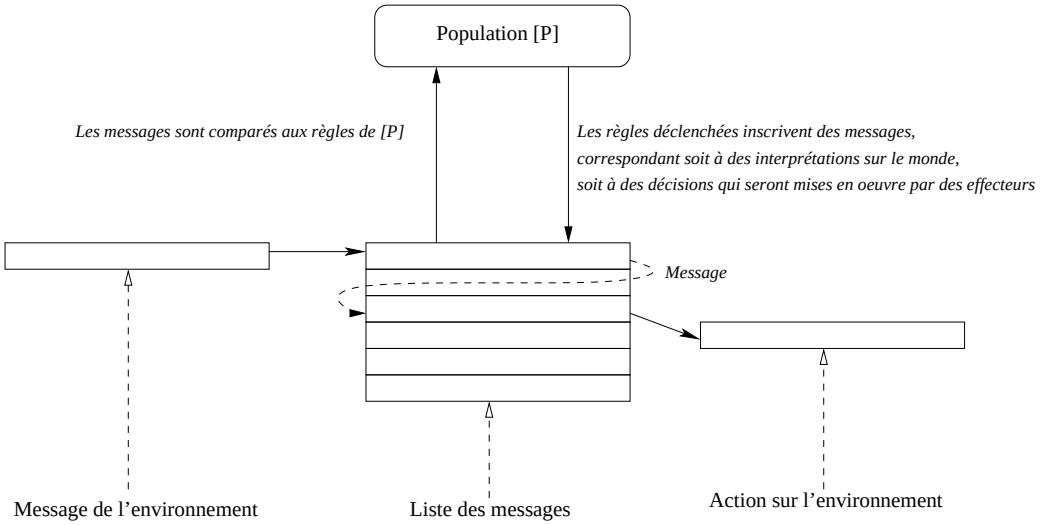


Figure 4.14 – Schéma d'un système de classeurs possédant une liste des messages.

Métaphore économique

L'algorithme intègre un système économique dans la population de classeurs `[P]`. L'idée est d'assimiler les classeurs et l'environnement à des agents financiers. La force d'une règle peut alors être vue comme le capital de l'agent. Le mécanisme de marché met en place une compétition entre les agents concernés par la proposition de l'environnement pour avoir le droit de participer aux enchères.

Description

Pour chaque classeur de `[P]`, nous introduisons les paramètres suivants : une *force*, une *spécificité*, une *enchère*, une *taxe* et une *rétribution* (respectivement `f`, `s`, `Cbid`, `Ctaxe` et `Cr`). Nous pouvons considérer un classeur sous la forme :

$$R = (C, A, f, s, Cbid, Ctaxe, Cr) \tag{4.5}$$

Le mécanisme est constitué de trois (éventuellement quatre) étapes :

1. *La vente aux enchères* :
   Chaque agent concerné, *i.e.* dont la partie condition est satisfaite, donne une enchère pour participer. Elle consiste en un calcul d'une valeur *Cbid* pour les classeurs étant dans `[M]`. L'enchère proposée par chaque classeur est proportionnelle à sa *force* f et à sa *spécificité* s . Elle est calculée en suivant :

$$\begin{cases} Cbid = r_{const} * s * f \\ 0 < r_{const} \leqslant 1 \\ 0 < s \leqslant 1 \end{cases} \tag{4.6}$$

   (a) La constante r_{const} est un facteur de risque représentant la perte proportionnelle que peut subir la force du classeur sur une itération. Elle peut être comparée au taux d'apprentissage d'un algorithme connexionniste. Elle est la même pour tous les classeurs.

(b) La force représente l'utilité du classeur pour la résolution du problème. Plus la force du classeur est élevée, plus son enchère est forte.

(c) Plus un classeur est spécialisé, plus il aura de chance de proposer une enchère élevée. L'intégration de la spécificité dans le calcul de l'enchère permet aux classeurs spécialisés de proposer des enchères importantes, et donc d'être plus compétitifs face aux classeurs généralisés.

Exemple :

$Classeur_1$: si l'apprenant est novice alors ...

$Classeur_2$: si l'apprenant est expert et qu'il a moins de 30 ans alors ...

⇨ *Le $Classeur_2$ est plus spécifique que le $Classeur_1$.*

La spécificité correspond au nombre d'attributs dont la valeur est précisée.

2. *La compétition* :
Une compétition est mise en place pour déterminer les agents vainqueurs. La probabilité d'être victorieux est proportionnelle à l'enchère relative. Elle consiste à calculer la probabilité qu'un classeur de `[M]` puisse gagner (mécanisme de roue de la fortune). C'est le mécanisme de `Sélection`. Les classeurs non sélectionnés réinitialisent leur enchère *Cbid* à 0. Les classeurs vainqueurs sont alors dans `[A]`.

3. *La répartition des rétributions* :
La rétribution va modifier la force des classeurs de `[A]`. Chacun d'entres-eux :

(a) reçoit une *rétribution de l'environnement*
s'il est un des vainqueurs directs (les classeurs qui ont fourni l'action effective) : la rétribution *Cr* dépend de l'environnement fournissant *r*. Classiquement, *Cr* est une valeur proportionnelle à *r* (la rétribution *r* est découpée selon le nombre de vainqueurs directs) ;

(b) doit *payer son enchère*

$$f(t) = f(t-1) - CBid \tag{4.7}$$

(c) reçoit une *rétribution des vainqueurs*
s'il est à l'origine de la situation, *i.e.* s'il a contribué à la proposition de la règle vainqueur. La rétribution *Cr* dépend alors des enchères des règles en amont.

La propagation inverse des récompenses permet une distribution qui renforce les classeurs amenant au maximum de récompense. De ce fait, ils proposent des enchères de plus en plus importantes et ainsi accroissent les récompenses obtenues de toute la chaîne de classeurs amenant à leur activation. Le mécanisme fonctionne comme une économie où des entreprises en compétition rémunèrent leurs sous-traitants, s'enrichissant ou s'appauvrissant selon que le contrat soit remporté ou non.

4. *Taxes* (optionnel) :
La mise à jour de la force du classeur peut être soumise à un mécanisme de taxation. Celui-ci diminue la force des classeurs qui ne sont jamais choisis afin de défavoriser les règles qui ne sont jamais invoquées. La taxe *Ctaxe* est prélevée proportionnellement à la force des classeurs de `[P]`.

En résumé, la *Force f* de chaque classeur de `[P]` est mise à jour à chaque cycle de fonctionnement, suivant la formule :

$$f(t) = f(t-1) - Cbid - Ctaxe + Cr \tag{4.8}$$

Illustration

La figure 4.15 montre un exemple d'utilisation de l'algorithme du *bucket brigade*. Ici, la représentation interne des perceptions est un code binaire. Ainsi, une règle est codée par une succession de n bits avec $C \cup A \in \{0, 1, \#\}^n$. Les règles sont représentées en utilisant la convention $C :: A \mid f$.

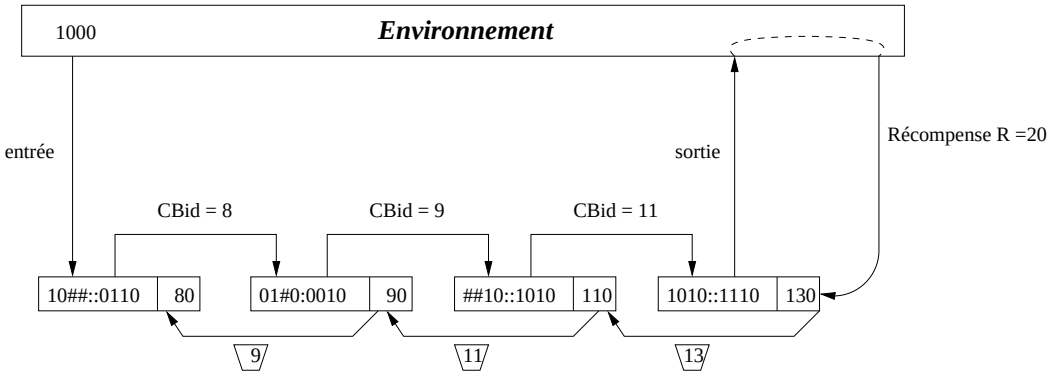


FIGURE 4.15 – Algorithme du *bucket brigade*. Dans cet exemple, les enchères sont calculées selon $CBid = k_0 * f$ avec $k_0 = 0.1$. La figure est tirée de (Sanchez, 2004).

L'interface d'entrée a détecté la situation 1000. La règle de `[P]` $R_1 = 10\#\# :: 0110 \mid 80$ est appariée et choisie. L'effet 0110 est considéré en interne et permet d'apparier la règle $R_2 = 01\#0 :: 0010 \mid 90$. Le processus continue ainsi jusqu'au dernier classeur produisant une action effective. La récompense de l'environnement est directement transmise au dernier classeur de la chaîne. Selon le principe de la chaîne des seaux, les classeurs précédents reçoivent en paiement l'enchère du classeur qu'ils activent.

La figure 4.16 illustre l'évolution des forces des classeurs mis en jeu dans l'exemple de la figure 4.15 lorsque la récompense de l'environnement est de 20. La propagation inverse des récompenses rétribue tous les classeurs ayant contribués à l'élection de l'action effective.

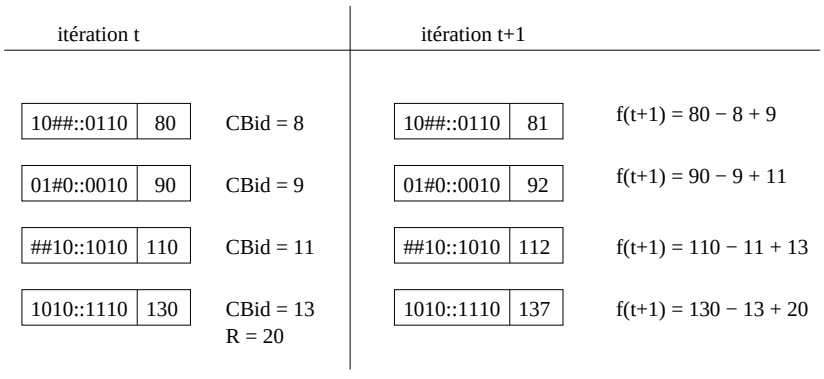


FIGURE 4.16 – Évolution des forces des classeurs.

Mise en œuvre

Nous avons adapté l'algorithme du *bucket brigade* à notre modèle. Cette adaptation est fondée sur une analogie entre les cycles d'inférences internes vus précédemment pour les architectures classiques et la structure hiérarchique de notre modèle. En effet, notre modélisation à trois niveaux peut être vue comme une succession de messages internes où l'action effective est représentée par le choix du formateur. Ainsi le déclenchement d'une action est consécutif à l'activation de règles sur trois niveaux (messages internes).

Formalisation des éléments mis en jeu

On considère :

1. Une règle (R) est un 8-uplet

$$\texttt{R} = (\texttt{C}, \texttt{A}, \texttt{s}, \texttt{f}, \texttt{Cbid}, \texttt{Cr}, \texttt{Ctaxe}, \texttt{CActivity}) \tag{4.9}$$

- ▷ Les deux parties de la règles $\texttt{C} \cup \texttt{A} \in \{string, \#\}^n$
  - `C` : la condition d'application de la règle.
  - `A` : l'effet associé à l'application de la règle.
- ▷ Les paramètres
  - `s` : spécificité de la règle.
  - `f` : force (ou qualité) de la règle.
  - `Cbid` : enchère proposée.
  - `Cr` : rétribution obtenue.
  - `Ctaxe` : taxe appliquée.
  - `CActivity` : activité de la règle, *i.e.* la conjonction de la force de la règle et de l'intensité de l'ensemble contenant la règle (le système de classeurs parent).

2. Un système de classeurs (CS) est un 10-uplet

$$\texttt{CS} = (\texttt{Ie}, \texttt{[P]}, \texttt{[M]}, \texttt{[A]}, \texttt{Comparaison}, \texttt{Sélection}, \texttt{Rétribution}, \texttt{Io}, \texttt{intensity}, \texttt{level}) \tag{4.10}$$

- ▷ `Ie` est l'interface d'entrée.
- ▷ `[P]` est la population de classeurs.
- ▷ $\texttt{[M]} \subseteq \texttt{[P]}$ est l'ensemble *Match-set*.
- ▷ $\texttt{[A]} \subseteq \texttt{[M]}$ est l'ensemble *Action-set*.
- ▷ `Comparaison` est le mécanisme d'appariement permettant de passer de `[P]` à `[M]`.
- ▷ `Sélection` est le mécanisme d'élection permettant de passer de `[M]` à `[A]`.
- ▷ `Rétribution` est le mécanisme permettant de rétribuer les règles.
- ▷ `Io` est l'interface de sortie.
- ▷ `intensity` est le paramètre représentant le degré d'activation du système de classeurs.
- ▷ `level` précise le niveau d'abstraction. $\texttt{level} \in \{1, 2, 3\}$.

Description de l'algorithme

L'algorithme est le suivant :

1. *Perception :*
   Les éléments de perception proviennent de la situation pédagogique.

```
for each level
{
    for each CS
     {
         // Perception
         Ie(t) := readDetectors(t) ;
     }
}
```

2. *Appariement :* ⇨ Déterminer les règles concernées
   Chaque `CS` effectue les opérations d'appariement, afin de savoir quelles sont les règles activables (passage de `[P]` à `[M]`). Pour cela, il utilise le mécanisme de `Comparaison`. Il s'agit d'une règle d'appariement entre les parties `C` de `[P]` et l'information provenant de `Ie`. Cette règle sait interpréter les symboles de généralisation composant les conditions des classeurs.

```
for each level
{
    for each CS
   {
       // Comparaison
       M(t) := matchClassifiers( Ie(t),P(t),Comparaison ) ;
   }
}
```

3. *Vente aux enchères* :
   Les classeurs présents dans les différents `[M]` calculent leur enchère.

```
for each level
{
    for each CS
   {
       for each R ∈ M(t)
      {
          // Calcul des enchères
          Cbid = coef * s * f ;
      }
   }
}
```

4. *Compétition* :
Mécanisme de diffusion de type FFH. L'algorithme parcourt les niveaux en descendant (du niveau 1 au niveau 3).

L'intensité des CS du niveau 1 a pour valeur 1. De ce fait, tous les ensembles du niveau 1 sont activés et les règles contenues peuvent ainsi participer à la prise de décision.

```
for each level {
 for each CS {
    // Compétition
    for each R1 ∈ [M] {

    // Diffusion (1) : mise à jour des activités de chaque classeur
       CActivity = Cbid * getCSParent() → getIntensity() ;

    // Diffusion (2) :   contribution de l'activité de R1
                         aux intensités des CS du niveau inférieur
       for each CS2 ∈ (level + 1) {
            if ( CS2 → getName() ∈ R1 → getAction() )
                 CS2 → addIntensity( R1 → getCActivity() ) ;
            }

        }
    }
}
```

5. *Sélection* :
Le formateur obtient une liste ordonnée des différentes règles du niveau 3, triée par activité. L'assistance vainqueur est déterminée directement par le choix du formateur. C'est le mécanisme de `Sélection`. L'assistance vainqueur définit les ensembles `[A]` du niveau 3.

6. *Répartition des rétributions* :
L'algorithme rétribue les règles en remontant les niveaux (du niveau 3 au niveau 1). Pour cela, il utilise le mécanisme de `Rétribution`.

(a) *Rétribution (1) :* Réception de la rétribution de l'environnement. Dans notre cas, il s'agit d'une valeur constante. La rétribution intervient lorsque le formateur choisit une assistance pédagogique. Ce paramètre pourrait faire l'objet d'un calcul plus élaboré, mais les critères associés n'ont pas encore été déterminés.

(b) *Rétribution (2) :* L'algorithme distribue les rétributions aux règles qui ont contribué à les obtenir (déterminer les ensembles `[A]` pour chaque niveau impliqué par la sélection du formateur). Il augmente la qualité des règles ayant participées à l'obtention de la rétribution. Deux parties sont distinguées :

i. *Déterminer les règles impliquées*

```
for each level {
 for each CS {

   // Rétribution (2.i.) : Déterminer les règles (R) impliquées

        // Association[a] action ⇨ enchère + règles du niveau inférieur
   map< Action, pair< Cbid, vector<R> > > P ;

   for each R1 ∈ [A] {
        // pair[b] : enchère + règles du niveau inférieur
     pair< Cbid, vector<R> > p ;
     p = P[ R1 → getAction() ] ;
     p → first() += R1 → getCbid() ;

     for each (level - 1) {
       for each CS {
         for each R2 ∈ [M]  {
           if ( R1 → getCSParent() → getName() ∈ R2 → getAction()
                 && R2 not in p → second() )
                {
                   p → second() → add(R2) ;
                   [A] → add(R2) ;
                }
           }
        }
     }

   // Remise à zéro des enchères
   for each R1 ∈ [M] {
     if( R1 ∉ [A] )
       Cbid = 0 ;
   }

  }
}
```

[a] Concrètement, il s'agit d'une table associative (une `map` au sens STL), *i.e.* une liste de paires dont le premier composant est la clef et le second la valeur. L'opérateur crochet permet, à partir d'une clef, d'obtenir la valeur associée. Une `map` est souvent comparée à un dictionnaire car elle fait correspondre à une clef une définition.

[b] Considérons que les méthodes `first()` et `second()` permettent d'accéder respectivement au premier et au deuxième élément de la pair.

ii. *Calcul du renforcement par classeur.*
Il s'agit d'une répartition équitable de l'enchère.

```
for each p ∈ P {
  // Rétribution (2.ii.) : Calcul du renforcement par classeur
  for each R1 ∈ ( p → second() ) {
        Cr = p→first() / p→second()→size() ;
  }
}
```

7. *Taxes*
La plupart des algorithmes utilisant le *bucket brigade* utilise une taxe sur l'ensemble [P] pour favoriser les règles productives. Avant d'appliquer la taxe sur cet ensemble, nous devons considérer les différences entre notre système et les systèmes classiques. En effet, nous ne souhaitons pas défavoriser les règles qui sont utilisées épisodiquement. Ces dernières ont été définies par le pédagogue, et sont importantes, même si les situations qu'elles recouvrent sont peu fréquentes. Nous pouvons alors penser que le mécanisme de taxe n'est pas obligatoire. Néanmoins, nous devons considérer que nous fournissons une rétribution à chaque proposition d'assistance alors que dans les systèmes traditionnels la récompense est épisodique. De plus, nous n'appliquons aucun mécanisme de « punition ». Les forces des règles qui ont contribué au choix du formateur augmentent et peuvent rapidement atteindrent un état d'équilibre qui stabilise la rétribution reçue et leur enchère. Par conséquent, nous décidons de taxer les règles présentes dans [M].

```
for each level {
  for each CS {
    // Taxes
    for each R1 ∈ [M] {
        Ctaxe = coef2 * f ;
    }
  }
}
```

L'impôt est proportionnel à la force du classeur ($0 \leqslant$ coef2 < 1). Ce mécanisme défavorise les règles appariées qui n'ont pas contribué à l'élection de l'assistance choisie par le formateur.

8. *Mise à jour de la force*

```
for each level {
  for each CS {
    // Mise à jour des forces
    for each R {
        f(t+1) = f(t) - CBid - Ctaxe + Cr ;
    }
  }
}
```

4.4 Bilan

L'objectif de ce chapitre était de définir un modèle simulant le comportement décisionnel de l'agent pédagogique qui fournit les propositions d'assistance. Notre modélisation devait considérer les particularités induites par la spécification du comportement par un pédagogue, l'hétérogénéité et la nature des connaissances impliquées, et présenter des capacités d'adaptation.

Dans un premier temps, nous nous sommes intéressés aux familles d'architectures comportementales existantes. Nous avons opté pour celles qui utilisent des règles. Plus précisément, nous avons choisi les systèmes de classeurs qui répondent au mieux aux critères que nous avons retenus dans le cadre de notre étude (expressivité, hiérarchisation, modularité, réactivité et adaptabilité). Il s'agit d'une architecture réactive et adaptative, fondée sur des règles conditionnelles.

Nous avons ensuite proposé un modèle basé sur un système de classeurs hiérarchiques. Il organise les connaissances en considérant l'abstraction des données mises en jeu. Il structure les connaissances sur trois niveaux, allant des règles basées sur des connaissances abstraites de la pédagogie (les démarches pédagogiques) jusqu'aux règles fondées sur des connaissances concrètes de la réalité virtuelle (les techniques pédagogiques), en passant par un niveau intermédiaire (les attitudes pédagogiques).

Chaque niveau d'abstraction contient des ensembles qui regroupent plusieurs règles. Un ensemble représente une façon d'aborder une démarche, une attitude ou une technique pédagogique. Les règles sont conditionnées par les éléments de la situation pédagogique et ont pour effet de favoriser des ensembles du niveau inférieur. Le système utilise alors un mécanisme de diffusion dans les trois niveaux qui considère les règles appariées par la situation pédagogique. Il aboutit à une liste qui ordonne les propositions d'assistance pédagogique.

Enfin, nous intégrons un mécanisme d'apprentissage artificiel, de type *bucket brigade*, qui adapte le comportement décisionnel de l'agent pédagogique au couple apprenant-formateur. Il est fondé sur un renforcement provenant des choix du formateur. Ainsi, au fur et à mesure des exercices, l'agent pédagogique doit proposer des choix de plus en plus en accord avec les décisions du formateur. L'agent pédagogique pourrait alors, dans le cadre d'un apprentissage multi-apprenants, prendre le relais temporairement et appliquer directement les assistances qu'il a choisies.

Le modèle pédagogique a fait l'objet de plusieurs publications, notamment (Buche et Querrec, 2005b; Buche et al., 2006).

Chapitre 5

Application : le projet GASPAR

> Tout ce que je sais du monde, même par science, je le sais à partir d'une vue mienne ou d'une expérience du monde sans laquelle les symboles de la science ne voudraient rien dire.
>
> MERLEAU-PONTY

Ce chapitre est consacré à la mise en application de notre proposition d'ITS. Nous décrivons un processus générique permettant de créer des environnements virtuels de formation. Dans un premier temps, le concepteur spécifie les modèles UML qui représentent les entités de l'environnement, les agents et les organisations. Ils sont ensuite transformés en données au format XMI et par conséquent deviennent manipulables. Le modèle VEHA*, implémenté en* ARÉVI*, interprète le fichier XMI et génère la structure des éléments de l'environnement. Enfin, les instances sont précisées par des fichiers XML, dont certains peuvent être générés à l'aide d'utilitaires logiciels. L'intégration de notre modèle d'ITS utilise la connaissance sur les organisations impliquées, et bénéficie de la représentation des connaissances effectuées par le modèle* VEHA *pour effectuer ses traitements. Les différentes parties de notre modèle d'ITS sont paramétrables à l'aide d'une interface graphique interactive. L'ITS propose également des interfaces de suivi des apprenants, où le formateur peut visualiser l'analyse portant sur la procédure, les statistiques, la situation pédagogique et les activités pédagogiques. L'exemple support de ce chapitre est l'environnement* GASPAR*, qui simule l'activité aviation sur un porte-avions. Ce chapitre s'achève en soulignant le caractère générique de notre proposition, idée qui est appuyée par la mise en œuvre de nos modèles dans d'autres environnements.*

LA partie précédente était consacrée à la présentation du modèle d'ITS. Nous avons montré qu'il fournit la situation pédagogique, *i.e.* les éléments utilisables pour la prise de décision pédagogique (*cf.* chapitre 3). Le modèle utilise ces connaissances pour simuler un raisonnement qui aboutit à des propositions d'assistances pédagogiques offertes au formateur (*cf.* chapitre 4).

Dans ce chapitre, nous présentons la mise en application de ce modèle. Sa principale utilisation concerne le projet GASPAR.

Dans un premier temps, nous présentons l'environnement de l'application Gaspar (section 5.1). Nous précisons ensuite comment nous réalisons l'implémentation d'applications utilisant le modèle Veha (section 5.2). Puis, nous décrivons l'intégration de notre modèle d'ITS dans cet environnement (section 5.3). Nous montrons les possibilités portant sur les paramètres des modèles de l'ITS et concernant le suivi de l'apprenant. Nous terminons ce chapitre en abordant l'intégration de ce modèle dans d'autres applications, telle que SécuRéVi [1], et soulignons ainsi la généricité de notre proposition (section 5.4).

5.1 L'environnement GASPAR

Le projet Gaspar (Gestion de l'Activité aviation et des Sinistres sur Porte-avions par la Réalité virtuelle) simule « l'activité aviation » sur un porte-avions (*cf.* figure 5.1). Cette application permet d'aider à la formation de la « gestion aviation ». Elle permet la visualisation des décollages (catapultages) et des atterrissages (appontages), l'observation et la création des déplacements d'avion et l'observation et la création d'interventions sur les avions (entretien, positionnement, *etc.*).

Figure 5.1 – L'application Gaspar.

Cette application, à laquelle nous avons pris part, est issue d'un projet mené au sein du CERV.

5.1.1 Les éléments de la simulation

La partie centrale de l'application est le porte-avions. Ce dernier est un bâtiment contenant plusieurs étages. A chaque étage, on peut trouver des zones qui peuvent être closes (pièces ou hangars), ou ouvertes (zones de parking, piste d'appontage, zone de catapultage). Les zones sont liées entre elles par des accès (portes, élévateurs, points de passage). Dans

[1] SécuRéVi : Sécurité et réalité virtuelle. Il s'agit d'une application permettant d'aider la formation des officiers sapeurs pompiers.

chacune de ces zones, on retrouve des éléments du porte-avions. Les étages du porte-avions sont le pont, le(s) hangar(s) pour les avions, et les différents hangars pour les munitions. Les étages communiquent entre eux par des élévateurs (monte-charge, plates-formes élévatrices). Le pont est constitué de brins d'arrêt, de catapultes, de déflecteurs, d'une piste d'appontage, de zones de stationnement pour les aéronefs, d'un îlot, d'une grue et de marquages au sol.

Les éléments de la simulation évoluent sur ce porte-avions (sur le pont ou dans les hangars). Certains de ces éléments sont des objets statiques (les déflecteurs, l'îlot, *etc.*), d'autres sont déplaçables (missiles, *etc.*) et d'autres mobiles (aéronefs, tracteurs, *etc.*). Les objets peuvent s'attacher les uns aux autres, par exemple un Rafale remorqué par un tracteur ou un missile porté par un chariot.

Les personnels sont des objets actifs qui possèdent une représentation graphique, des attributs et des comportements. La représentation graphique est constituée d'un corps réaliste d'humain, d'une texture de vêtements selon le rôle qu'il joue dans la simulation et d'animation pour chaque action « métier » qu'il doit réaliser. Un membre du personnel dispose des mêmes comportements que les objets mobiles (déplacement, évitement, *etc.*), ainsi que des comportements « métier » comme vérifier un élément particulier du Rafale, fixer un aéronef à la catapulte, *etc.* Il ne s'agit pas ici de simuler précisément le geste technique réalisé par le personnel, mais surtout son effet sur l'environnement (changement des valeurs des attributs de l'objet manipulé ou déclenchement de comportement). Ainsi les gestes sont représentés par une animation. Les personnels participent à des procédures collaboratives.

5.1.2 Les procédures collaboratives

Les procédures liées aux aéronefs sont l'avitaillement en carburant, l'armement, le catapultage, l'appontage et la vérification du bon fonctionnement. L'exemple utilisé tout au long de ce chapitre est la procédure de catapultage. Elle fait intervenir un Rafale, une catapulte, un déflecteur, le pilote du Rafale, l'officier qui gère le catapultage, les membres du personnel qui fixent le Rafale à la catapulte et manipulent le déflecteur.

Dans le cas de l'activité aviation sur porte-avions, les procédures sont collaboratives, *i.e.* elles font intervenir plusieurs personnes membres du personnel. Ces personnes sont des officiers ou des sous-officiers impliquées dans la procédure à réaliser.

Dans un exercice, un apprenant se voit donc affecter un ou plusieurs rôle(s) parmi ces personnels. Un exercice doit contenir au moins un apprenant, les autres rôles étant alors joués par d'autres apprenants, par le système ou par le formateur lui-même.

5.2 Implémentation d'application en Veha

Rappelons que les applications développées se basent sur le modèle Veha. Ce modèle a fait l'objet d'une description dans la chapitre 3. Il permet, à partir d'une spécification UML, de « générer » les classes des objets utilisées, ainsi que les procédures. Il est alors possible de manipuler ces informations, notamment pour la prise de décision pédagogique.

La démarche à suivre pour obtenir une application est la suivante :

1. Spécification des modèles en UML en utilisant un logiciel adapté (Rational Rose© [2], Objecteering© [3], ArgoUML[4], MagicDraw[5], Umbrello[6], *etc.*).
2. Exporter les modèles en données manipulables *via* le format XMI.
3. Transformer les données XMI en classes Veha.
4. Préciser les instances de classes.

Ces quatre points sont décrits successivement dans cette section.

5.2.1 Spécification des modèles en UML

La modélisation est effectuée à l'aide d'un logiciel permettant de spécifier les modèles UML. L'outil permet généralement la modélisation de façon interactive et graphique à l'aide d'une interface utilisateur. Le concepteur de l'EV organise son modèle en un ensemble de diagrammes (diagramme de classes et diagramme d'activités) qui sont stockés dans des packages. Le concepteur s'intéresse alors uniquement à une description métier sans se soucier de l'implémentation.

L'organisation d'une application sous la forme de packages permet de structurer les informations, et offre une certaine modularité dans la spécification. De plus, ce découpage permet de distinguer les parties du modèle UML de natures différentes qui sont interprétées séparément dans le modèle Veha. Pour l'application Gaspar, nous nous sommes inspirés de l'approche *Voyelles* qui analyse les systèmes multi-agents selon quatre points de vue : Agents, Environnements, Interactions et Organisations (voyelles A,E,I,O) (Demazeau, 1995). Pour l'instant, nous ne nous sommes pas occupés de la partie interaction, Gaspar est donc composé de trois packages que nous décrivons ici.

5.2.1 - A Package « Environnement »

Ce package décrit les entités constituant l'environnement. Elles sont définies selon trois sous-packages :

[2] `http ://www-306.ibm.com/software/rational/`

[3] `http ://www.objecteering.com/`

[4] `http ://argouml.tigris.org/`

[5] `http ://www.magicdraw.com/`

[6] `http ://uml.sourceforge.net/index.php`

Sous-package	*Exemples*	*Aperçu*
① Porte-avions	Catapulte, poste de commandement, îlot, *etc.*	 Îlot
② Aéronef	Rafale, hélicoptère, *etc.*	 Rafale
③ Outil	Tracteur, grue, *etc.*	 Grue

Nous définissons le diagramme de classe, associé à chaque type d'entité, en suivant le modèle de classe (méta-modèle) d'une entité Veha. Plus précisément, il s'agit de spécifier ses attributs, ses opérations, sa machine à états et les associations avec d'autres entités. La figure 5.2 montre l'exemple de la spécification d'un porte-avions et d'une catapulte.

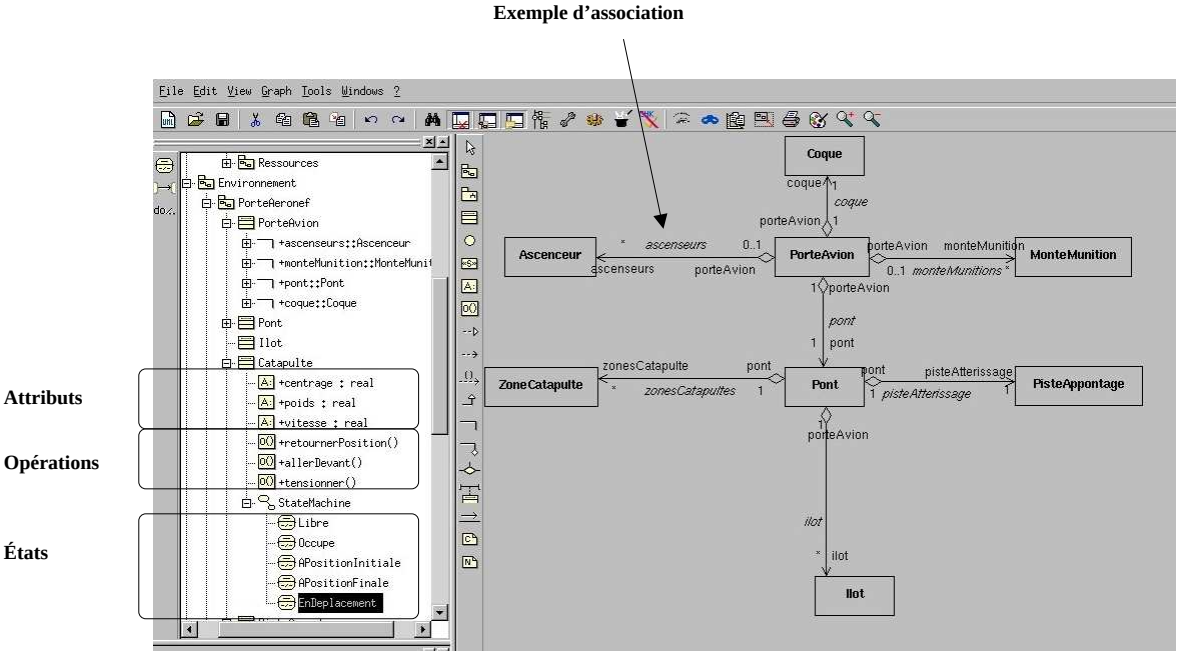


Figure 5.2 – Spécification UML des classes : exemple du porte-avions.

Un porte-avions (`PorteAvion`) est constitué d'un pont, d'une coque, de monte-munitions et d'ascenseurs. Une `Catapulte` possède :

- ▷ des attributs : un `poids`, une `vitesse` ;
- ▷ des opérations telles que `tensionner()` ;
- ▷ une machine à états qui contient les états `Libre`, `Occupé`, `APositionInitiale`, `APositionFinale`, `EnDéplacement`.

5.2.1 - B Package « Agent »

Dans l'application Gaspar, les agents faisant partie des organisations sont des personnels représentés par des humanoïdes. La figure 5.3 montre les agents impliqués dans la procédure exemple de catapultage. Un seul type d'agent est utilisé : les `Personnel`. Ils possèdent un certain nombre d'opérations (`AllerA`, `Suivre`, *etc.*) et jouent un ou plusieurs rôles au sein d'une organisation.

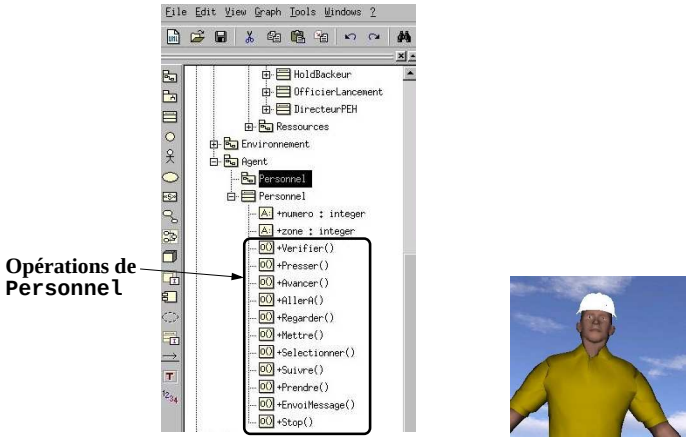


Figure 5.3 – Spécification UML des classes : définition du personnel dans Gaspar.

5.2.1 - C Package « Organisation »

Ce package précise, pour chaque type d'organisation, la définition des rôles, des procédures et des ressources utilisées. Pour chaque organisation, les rôles sont précisés par énumération. Les procédures sont définies par un diagramme d'activité UML. Les rôles sont symbolisés par un couloir d'activité dans le diagramme. Chaque rôle contient un ensemble d'actions. Nous verrons plus en avant (section 5.2.5) comment nous effectuons le lien entre les actions des rôles et les opérations des agents.

La figure 5.4 montre l'exemple de l'organisation de type `Pont`. Cette dernière possède la procédure `Catapultage`. Les actions associées au rôle `Pilote` (`AvancerAHoldBack`, *etc.*) sont définies en figure 5.4 (a). La procédure de catapultage est représentée en figure 5.4 (b). Dans cette procédure, l'`Officier Catapulte` commence par vérifier la position du déflecteur de l'avion alors que le `Maître IA`[7] informe du retour catapulte. A la suite de sa première action, l'`Officier Catapulte` vérifie le verrou de la catapulte. Cette figure est une description partielle d'une procédure qui contient au total 8 rôles, 61 étapes et 77 transitions.

[7] Maître IA : personne responsable des installations aviations, ici en charge de la catapulte sur le rafale.

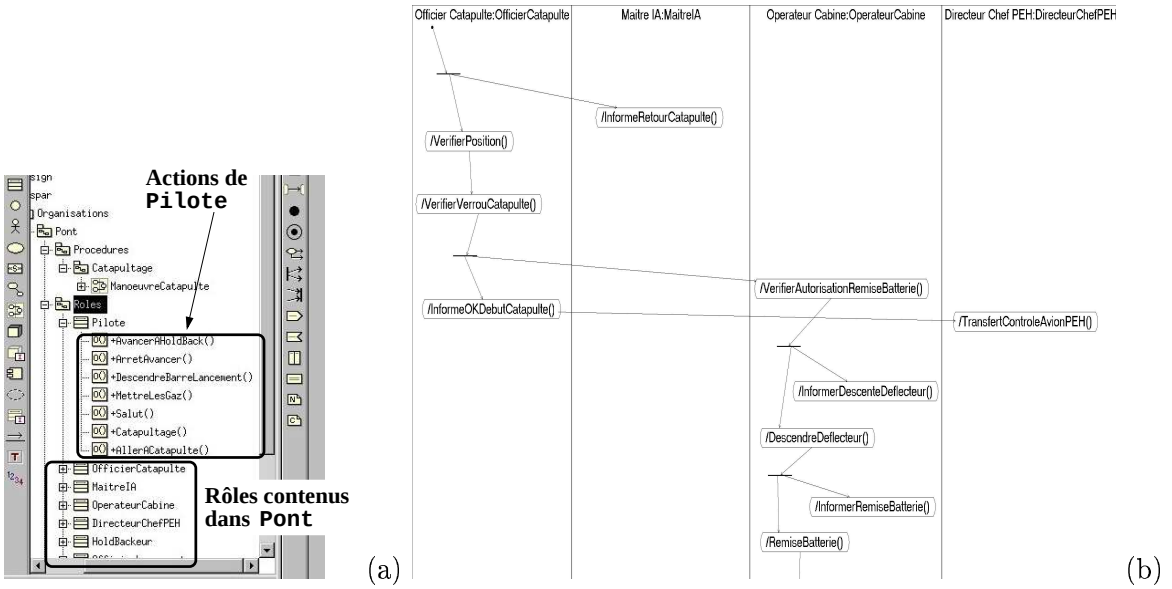


Figure 5.4 – Définition des rôles : exemple du rôle Pilote (a) et diag. d'activité UML : exemple de la procédure catapultage (b).

5.2.2 Manipuler les modèles UML : le format XMI

Après que les modèles UML des entités de l'environnement, des agents et des organisations aient été définis, il s'agit de manipuler ces informations afin de créer l'application. Nous utilisons pour cela une transformation des informations des modèles UML en données au format XMI[8]. Ce dernier définit un standard d'échange de méta-données UML en XML[9].

5.2.2 - A XMI pour les modèles UML

Bien que les modèles UML et les modèles XML différent, puisque les premiers utilisent un graphisme alors que les seconds sont basés sur du texte, ces deux types ont en commun la description de données des modèles (*cf.* figure 5.5 (a)). De plus, les premiers sont structurés par un méta-modèle UML et les seconds sont structurés par une grammaire appelée schéma[10] (*cf.* figure 5.5 (b)).

Le format XMI, défini par l'OMG, est une passerelle qui permet le passage des modèles UML aux modèles XML (*cf.* figure 5.6). En effet, XMI a été utilisé sur UML pour construire le schéma XML des modèles UML. Ainsi il est possible d'encoder un modèle UML dans un document XML. La plupart des logiciels de modélisation UML permette d'importer et

[8] XMI : XML Metadata Interchange.

[9] XML : eXtensible Markup Language (langage de balisage extensible).

[10] XML Schema est un langage de description de format de document XML, permettant de définir la structure d'un document XML. La connaissance de la structure d'un document XML permet notamment de vérifier la validité de ce document. Un fichier de description de structure (XML Schema Description en anglais, ou fichier XSD) est donc lui-même un document XML.

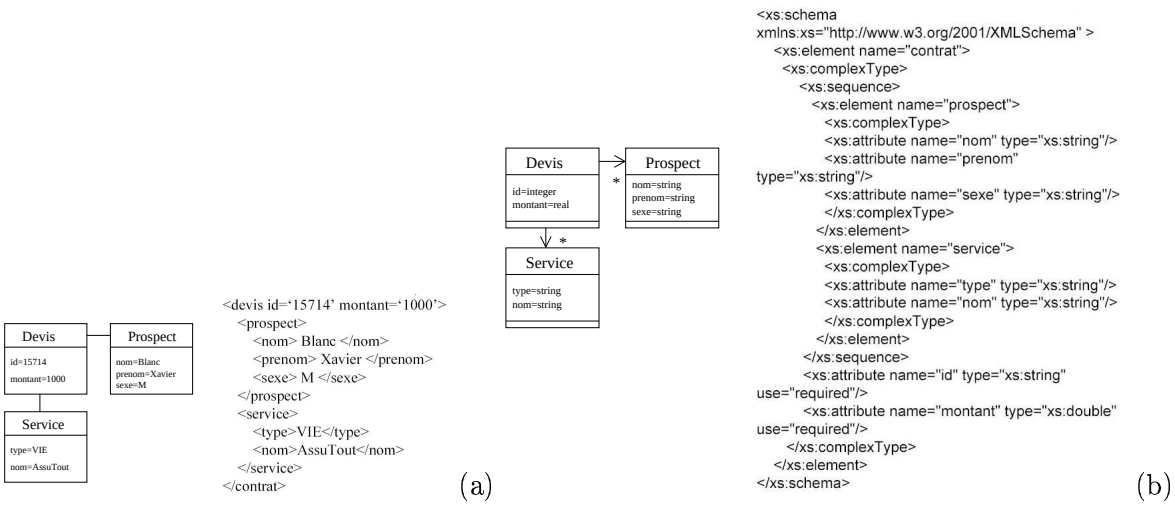


FIGURE 5.5 – Modèle vs XML (a) et Méta-Modèle vs Schéma (b).

d'exporter les informations contenues dans les représentations graphiques des diagrammes de classes et d'activités en fichiers XMI.

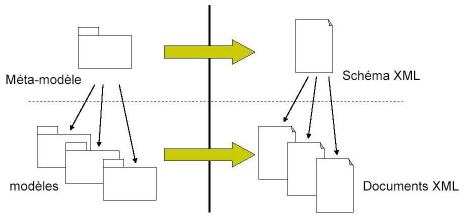


FIGURE 5.6 – La passerelle OMG : XMI.

5.2.3 XMI vers VEHA

Nous venons de voir comment effectuer le passage de modèles UML aux données XML. Il s'agit à présent de manipuler ces informations afin de produire une interprétation en VEHA pour obtenir une application. VEHA est implémenté en utilisant la bibliothèque ARÉVI[11] développée au CERV. Cette bibliothèque `C++` propose des facilités pour concevoir des applications à base d'entités autonomes dans des univers tridimensionnels.

La figure 5.7 illustre la méthode globale à suivre pour obtenir l'implémentation ARÉVI des données définies en UML. Une fois les fichiers XMI obtenus, il reste deux étapes qui s'effectuent sous ARÉVI :

1. *Interpréter les fichiers XMI et réifier les éléments XMI.*
   Cette étape est un intermédiaire entre la version XMI et l'implémentation VEHA. Il s'agit d'un interpréteur utilisant les données des fichiers XMI pour les transformer en classe d'éléments XMI. La manipulation des données est alors possible, notamment pour

[11] ARÉVI : Atelier de Réalité Virtuelle, disponible sur `http ://sourceforge.net/projects/arevi/`

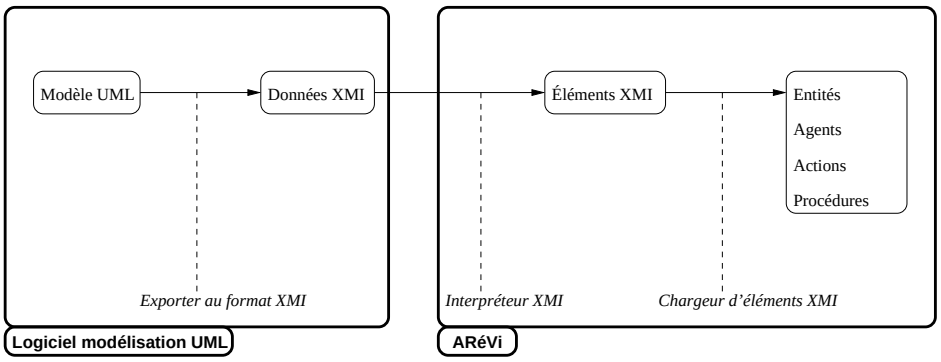


Figure 5.7 – Méthode pour obtenir l'implémentation ARéVi des données définies en UML.

une interprétation selon le modèle Veha. Cette étape est nécessaire puisqu'elle réalise une passerelle entre le format XMI et le modèle Veha, les évolutions du format XMI sont alors « transparentes ».

2. *Utiliser ces éléments.*
   Les éléments de Veha sont désormais des objets ARéVi, ils sont donc réifiés et exploitables par l'interpréteur Veha. Ce dernier est lui même écrit en ARéVi. Il propose de :

   (a) générer les classes d'entités de l'environnement,

   (b) générer les classes d'agents,

   (c) générer les organisations.

Nous décrivons ici ces trois points.

5.2.3 - A Générer les classes d'entités de l'environnement

Tout d'abord, nous générons les classes représentants des types d'entités de l'environnement à partir des fichiers XMI, plus précisément nous représentons :

- ▷ la notion d'héritage,
- ▷ les attributs,
- ▷ les opérations,
- ▷ les machines à états,
- ▷ les associations.

L'extrait du fichier XMI qui suit, illustre les données pour la classe Rafale. Dans cet exemple, la balise `Generalization` montre que le Rafale est un Avion (référencé par `a3861387812-1160`). Il possède un attribut `vitesseMaxRoulage` de type `real` (référencé par `a4-11`).

```
<UML:Class xmi.id = 'a3861387812-955'>
  <UML:ModelElement.name>Rafale</UML:ModelElement.name>
  ...
  <UML:Namespace.ownedElement>
      <UML:Generalization xmi.id = 'a3861387812-1165'>
          <UML:ModelElement.name>Generalization</UML:ModelElement.name>
                    <!-- Le Rafale est un Avion -->
          <UML:Generalization.parent>
              <UML:GeneralizableElement xmi.idref = 'a3861387812-1160' />
          </UML:Generalization.parent>
      </UML:Generalization>
  </UML:Namespace.ownedElement>
  <UML:Classifier.feature>
                    <!-- Le Rafale a un attribut vitesseMaxRoulage -->
      <UML:Attribute xmi.id = 'a3861387812-1214'>
          <UML:ModelElement.name>vitesseMaxRoulage</UML:ModelElement.name>
          ...
          <UML:StructuralFeature.type>
              <UML:Classifier xmi.idref = 'a4-11' />     <!-- real -->
          </UML:StructuralFeature.type>
      </UML:Attribute>
  </UML:Classifier.feature>
  ....
```

L'extrait du fichier XMI qui suit, illustre les données pour les associations à la classe Rafale, plus spécifiquement l'association du Rafale à l'aileron (liaison appelée `myAileronAv`). Dans cet exemple, la liaison `myAileronAv` associe la classe `Rafale` à la classe `Aileron` par le champ `_aileronAv` avec une multiplicité de `1...1`. Elle associe également la classe `Aileron` à la classe `Rafale` par le champ `_rafale` avec une multiplicité de `1...1`.

```
<UML:ModelElement.name>myAileronAv</UML:ModelElement.name>
<UML:Association.connection>
    <UML:AssociationEnd xmi.id = 'a3861387812-987'>
        <UML:ModelElement.name>_aileronAv</UML:ModelElement.name>
        ...
        <UML:AssociationEnd.multiplicity>
            <UML:Multiplicity>
                 <UML:MultiplicityRange>
                     <UML:MultiplicityRange.lower>1</UML:MultiplicityRange.lower>
                     <UML:MultiplicityRange.upper>1</UML:MultiplicityRange.upper>
                 </UML:MultiplicityRange>
            </UML:Multiplicity>
        </UML:AssociationEnd.multiplicity>
        <UML:AssociationEnd.participant>
            <UML:Classifier xmi.idref = 'a3861387812-959' />   <!-- Aileron -->
        </UML:AssociationEnd.participant>
    </UML:AssociationEnd>

    <UML:AssociationEnd xmi.id = 'a3861387812-988'>
        <UML:ModelElement.name>_rafale</UML:ModelElement.name>
        ...
        <UML:AssociationEnd.multiplicity>
            <UML:Multiplicity>
                 <UML:MultiplicityRange>
                     <UML:MultiplicityRange.lower>1</UML:MultiplicityRange.lower>
                     <UML:MultiplicityRange.upper>1</UML:MultiplicityRange.upper>
                 </UML:MultiplicityRange>
            </UML:Multiplicity>
        </UML:AssociationEnd.multiplicity>
        <UML:AssociationEnd.participant>
            <UML:Classifier xmi.idref = 'a3861387812-955' />   <!-- Rafale -->
        </UML:AssociationEnd.participant>
    </UML:AssociationEnd>
</UML:Association.connection>
...
```

5.2.3 - B Générer les classes d'agents

De la même manière que pour les entités de l'environnement, les agents participant aux organisations sont générés. Dans le cas de l'application Gaspar, nous n'avons qu'un type d'agent : `Personnel` (*cf.* figure 5.4). Ils sont capables de réaliser un certain nombre d'opérations. Dans l'exemple, il s'agit de `Presser`, `Avancer` et `AllerA`. La classe `Personnel` est extraite de la partie du fichier XMI suivant :

```
<UML:Class xmi.id = 'a3861387812-1221'>
   <UML:ModelElement.name>Personnel</UML:ModelElement.name>
      ...
      <UML:Classifier.feature>
          <UML:Operation xmi.id = 'a3861387812-1556'>
             <UML:ModelElement.name>Presser</UML:ModelElement.name>
               ...
          </UML:Operation>
          <UML:Operation xmi.id = 'a3861387812-1562'>
             <UML:ModelElement.name>Avancer</UML:ModelElement.name>
                ...
          </UML:Operation>
          <UML:Operation xmi.id = 'a3861387812-1563'>
             <UML:ModelElement.name>AllerA</UML:ModelElement.name>
                                    ...
          </UML:Operation>
       </UML:Classifier.feature>
</UML:Class>
```

5.2.3 - C Générer les organisations

Les structures des organisations sont générées en précisant un type d'organisation, les rôles qu'elle contient et les actions associées à ces rôles. L'exemple suivant illustre la structure d'une organisation de type `Pont`. Elle définit un rôle `Pilote` qui contient les actions `AvancerAHoldBack` et `ArretAvancer`.

```
<UML:ModelElement.name>Organisations</UML:ModelElement.name>
...
  <UML:ModelElement.name>Pont</UML:ModelElement.name>
  ...
    <UML:ModelElement.name>Roles</UML:ModelElement.name>
    ...
       <UML:ModelElement.name>Pilote</UML:ModelElement.name>
       ...
          <UML:Operation xmi.id = 'a2996044088-17'>
               <UML:ModelElement.name>AvancerAHoldBack</UML:ModelElement.name>
               ...
          </UML:Operation>
          <UML:Operation xmi.id = 'a2996044088-18'>
               <UML:ModelElement.name>ArretAvancer</UML:ModelElement.name>
                ...
```

Une organisation contient des procédures à réaliser. Ces dernières sont générées en interprétant les éléments constituant les diagrammes d'activités composés d'états (`ActionState`) et de `Transition`. Cette dernière peut être de type séquentielle ou alternative et, éventuellement, utiliser une garde (contraintes de transition). Le diagramme d'activité est celui que nous avons présenté en figure 5.4 (b) page 129.

L'exemple suivant illustre le début de la procédure de catapultage décrit en XMI. Dans l'exemple, la procédure débute par l'action `VerifierPosition` qui a pour ressource le `Deflecteur Avion`.

```
<!-- ActivityGraph -->
  <UML:ActivityGraph xmi.id = 'a3861387788-20'>
      <UML:ModelElement.name>ManoeuvreCatapulte</UML:ModelElement.name>
      <!-- TopState of the StateMachine -->
      ...
     <UML:ActionState xmi.id = 'a3861387788-43' >
         <UML:ModelElement.name>/VerifierPosition()</UML:ModelElement.name>
         <UML:ActionState.isDynamic xmi.value='true'/>
         <UML:ActionState.dynamicArguments>
             <UML:ArgListsExpression>
                 <UML:Expression.body>Deflecteur Avion</UML:Expression.body>
             </UML:ArgListsExpression>
         </UML:ActionState.dynamicArguments>
         ...

         ...
      </UML:ActionState>
...
```

La transition, définie ci-dessous, a pour source l'action `VerifierPosition` et pour destination l'action `VerifierVerrouCatapulte`.

```
<!-- Transitions of the StateMachine -->
  <UML:StateMachine.transitions>
      <UML:Transition xmi.id = 'a3861387788-51'>
          <UML:ModelElement.name></UML:ModelElement.name>
          <UML:Transition.source>
              <!-- /VerifierPosition() -->
              <UML:StateVertex xmi.idref = 'a3861387788-43' />
          </UML:Transition.source>
          <UML:Transition.target>
              <!-- /VerifierVerrouCatapulte() -->
              <UML:StateVertex xmi.idref = 'a3861387788-48' />
          </UML:Transition.target>
          <UML:Transition.guard>
          </UML:Transition.guard>
      </UML:Transition>
...
```

5.2.4 Les instances

Cette section montre la création des instances, *i.e.* les objets issus d'une classe. Plus précisément, nous générons les instances d'entités, d'agents et d'organisations.

5.2.4 - A Générer les instances d'entités et d'agents

Les instances de classes des entités et des agents auraient pu être spécifiées en utilisant le logiciel de modélisation UML de la même manière que nous avons défini les modèles de classes. Néanmoins, cette solution n'est pas satisfaisante. En effet, préciser chaque instance est

fastidieux. De plus, nous pensons qu'il est intéressant de distinguer la définition métier (modèle de classe UML) de la partie application (spécifique à tel ou tel environnement). Enfin, nous développons actuellement une interface graphique permettant de préciser les entités et agents pour une application donnée (saisir les valeurs des attributs, ajout d'opérations, *etc.*). Le logiciel interactif permet notamment de placer les éléments directement dans l'environnement virtuel, il est alors possible de vérifier si la position convient en cours de spécification. De plus, les formes et textures sont également modifiables et mises à jour en ligne (*cf.* figure 5.8).

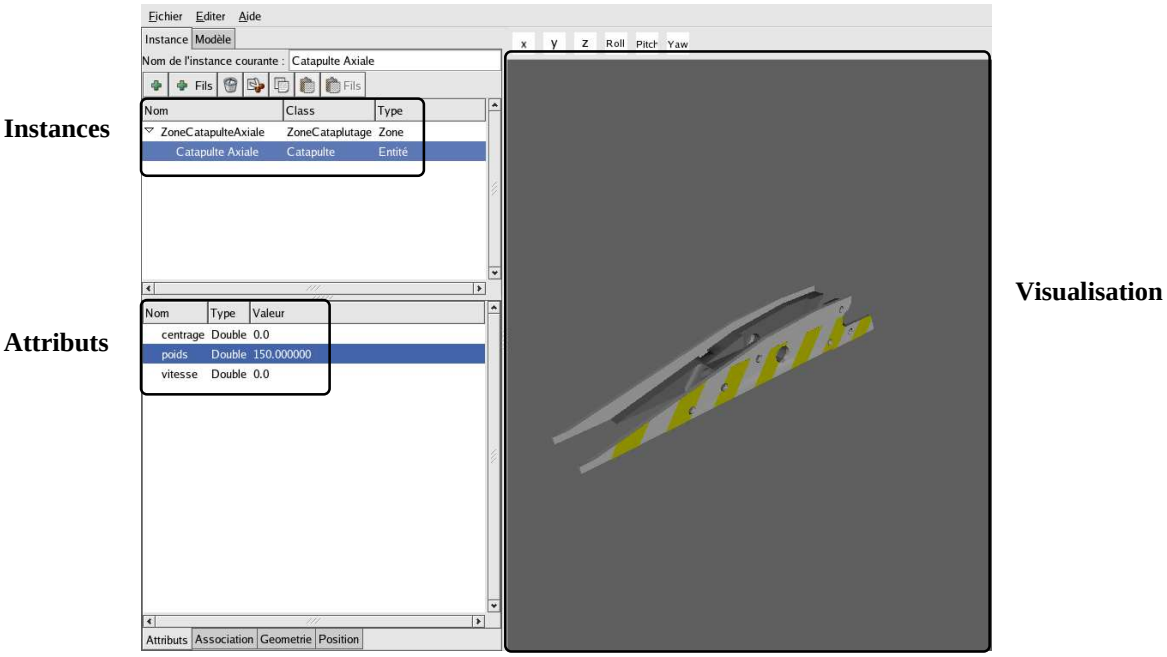


Figure 5.8 – Application de création d'instances d'entités.

L'application récupère les classes définies par le fichier XMI, et propose d'instancier des entités. Dans l'exemple, nous précisons les valeurs des attributs. Une vue 3D nous montre la position de l'entité dans la scène et permet de la déplacer en ligne.

Les instances de classes des entités et des agents sont définies dans un fichier XML à part. Il spécifie la position dans l'environnement, les associations, la forme géométrique et les valeurs des attributs de chaque entité.

L'exemple suivant illustre une instance d'une entité de type `Catapulte` appelée `Catapulte Axiale`. La `Catapulte Axiale` spécifie les valeurs des attributs `poids` et `vitesse`. Les entités sont situées dans des zones. Dans l'exemple, la `Catapulte Axiale` se place dans la `Zone Catapulte Axiale`.

```
<Zone name = "Zone Catapulte Axiale" class = "ZoneCatapulte" >
        <Location x="0" y="0" z="0"/>
        <Entity name="Catapulte Axiale" class = "Catapulte">
          <Location x="173" y="5" z="0"/>
          <Shape url="VRMLS/PorteAvion2Romeo/wrl/crocCatapulte-HI.wrl"/>
          <Attribut name = "centrage" value ="0" />
          <Attribut name = "poids" value = "150" />
          <Attribut name = "vitesse" value ="0" />
        </Entity>
        <Entity name="....
        .....
</Zone>
```

L'exemple suivant illustre la création d'une instance d'un agent de type `Personnel`, appelée `Laverdure` :

```
<Agent name = "Laverdure" class = "Personnel" >
   <Location  x = "162.0" y="-7" z="0.0"/>
   <Orientation roll = "0" pitch = "0" yaw = "3.14" />
   <Shape url="VRMLS_OLD/Personnages/personnage2.wrl"/>
</Agent>
```

5.2.4 - B Générer les instances d'organisation

Un fichier XML décrit les instances d'organisations et l'affectation des rôles aux agents. L'exemple suivant montre l'équipe `Equipe catapulte axiale` qui est une instance d'organisation de type `Pont`. Le rôle `Pilote` est joué par l'agent `Laverdure`.

```
<Team name ="Equipe catapulte axiale">
  <Organisation>Pont</Organisation>
    <Affect type = "Role">
      <Role>Pilote</Role>
        <Agent>Laverdure</Agent>
    </Affect>
 ...
```

Ici, il s'agit simplement d'affecter les rôles aux agents, les fichiers XMI ayant définis les rôles et les actions par rôle (informations contenues dans la structure organisationnelle de type `Pont`). Ces actions restent à être définies (*cf.* section suivante).

5.2.5 Implémentation des actions

Nous avons vu, en section 5.2.3, comment les instances de rôles sont définies et comment les actions sont associées aux rôles. Il reste à définir l'implémentation des actions et le lien entre les actions (liées aux rôles) et les opérations (liées aux compétences des entités). Ces informations sont codées par le programmeur ou extraites de bibliothèques existantes.

5.2.5 - A Les opérations

Définir les opérations est un travail conséquent, néanmoins nous avons développé une bibliothèque d'opérations couramment utilisées par une entité virtuelle (avancer, reculer,

presser, allerA, *etc.*). Cette bibliothèque peut être augmentée si besoin est. Pour définir une opération, il suffit d'implémenter une classe fille de la classe `Operation` et de sur-définir la méthode qui permet de réaliser son exécution. Le prototype est le suivant :

```
bool  Operation::exec(ArRef<Entity> performer,              // entité exécutante
                      ArRef<Entity> stimulus,               // entité qui commande l'exécution
                      map< string, ArRef<Attribute> > args, // paramètres
                      double dt);                           // fréquence d'appel
                                                               pour les actions cycliques
```

L'exemple suivant montre comment nous avons défini l'opération « ouvrir ». Elle peut s'appliquer à toute entité disposant d'un attribut `vitesse` et d'un attribut `angleOuverture`.

```
bool ouvrir::exec(ArRef<Entity> performer,
                  ArRef<Entity> /*stimulus*/,
                  StlMap< StlString, ArRef<Attribute> > /*args*/,
                  double dt)
{
  double vitesse, angleOuverture;
  double x,y,z;

  vitesse = performer->getAttribute("vitesse")->getValueDouble();
  angleOuverture = performer->getAttribute("angleOuverture")->getValueDouble();

  performer->pitch(vitesse*dt);                      // méthode ARéVi
  performer->extractOrientation(x,y,z);              // méthode ARéVi
  if(y >= angleOuverture)
  {
    performer->setOrientation(x,angleOuverture,z);  // méthode ARéVi
    return false;
  }

  return true;
}
```

5.2.5 - B Lier actions et opérations

Les actions sont utilisées dans la procédure. Elles utilisent les opérations des entités et précisent les attributs de leur méthode `exec`. Dans l'exemple suivant, l'action `AllerAuTrainAvantAvion` utilise l'opération `AllerA` du `Performer` de l'action, et précisent la destination en utilisant la ressource `TrainAvantAvion` qui est contenue dans l'organisation. Les conditions, telles que nous les avons préalablement définies en figure 3.8 page 51, sont également précisées.

```
bool AllerAuTrainAvantAvion::activate(ArRef<Activity> /*act*/, double /*dt*/)
{
   if (_state == 0)
   {
         ArRef<Agent> actionPerf = getPerformer();

         ArRef<Organisation> organisation = actionPerf->getOrganisation("Pont");

         ArRef<PhysicalEntity> train;
         train = organisation->getRessource("TrainAvant")->getEntity();
         double tx,ty,tz;
         ArRef<Base3D> base = new_Base3D();
         base->setLocation(train->getShape3D());
         base->translate(3,0,0);
         base->getPosition(tx,ty,tz);

         ArRef<Operation> operation = actionPerf->getOperation("AllerA");
           // OperationExecution exécute une opération
         ArRef<OperationExecution> _execution  = new_OperationExecution();
         _execution->setOperation(operation);
         _execution->setPerformer(actionPerf);

         StlMap <StlString , ArRef<Attribute> > parameters;
         ArRef<Attribute> x = new_Attribute();
         x->setType(Double);
         x->setValueDouble(tx);
         parameters["x"] = x;
         ArRef<Attribute> y = new_Attribute();
         y->setType(Double);
         y->setValueDouble(ty);
         parameters["y"] = y;
         _execution->start(parameters);
         _state ++;
   }
   else
   {
         if ( (_execution->getStatus() == FAILED) ||
              (_execution->getStatus() == COMPLETED) )
         {
            _state = 0;
            stop();
         }
   }
   return true;
}
```

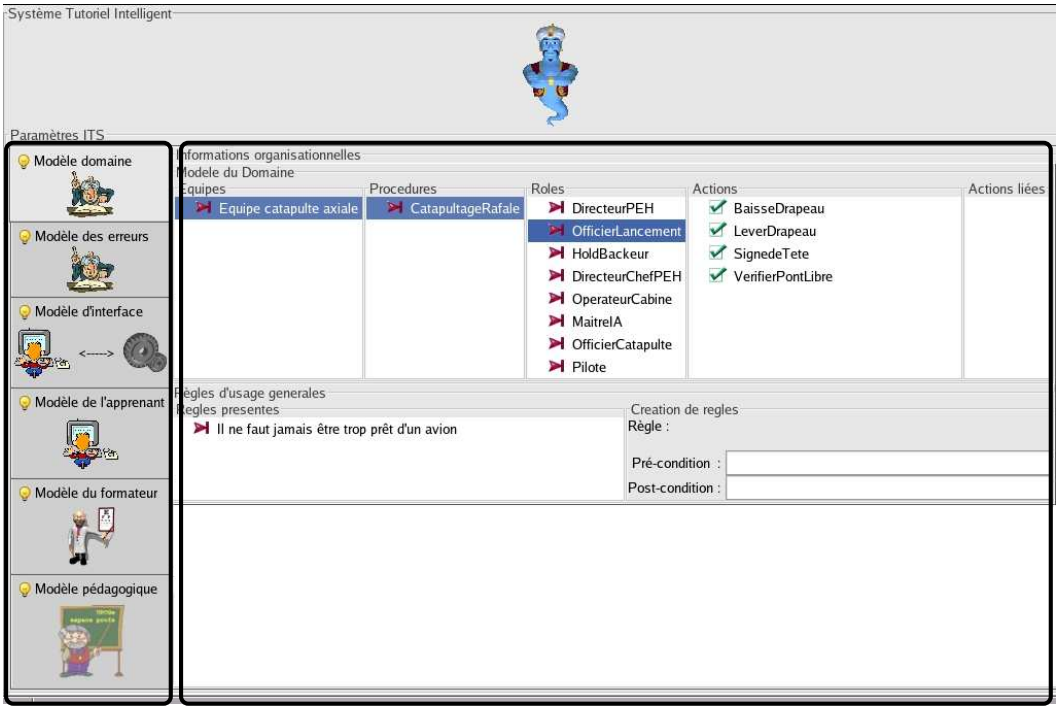
5.3 Intégrer l'ITS

L'intégration de notre ITS s'établit comme l'ajout d'un module, *i.e.* un programme tiers venant se greffer à l'application principale afin de lui apporter de nouvelles fonctionnalités. La seule hypothèse concernant l'ITS est l'utilisation du modèle Veha pour définir l'application.

Dans cette section, nous présentons, dans un premier temps, les possibilités de paramétrage des modèles des ITS. Ensuite, nous montrons les solutions offertes au formateur concernant le suivi de chaque apprenant.

5.3.1 Paramètres des modèles

Afin de pouvoir facilement paramétrer les modèles des ITS, une interface graphique interactive est proposée. Elle propose un onglet par modèle (domaine, erreurs, interface, apprenant, formateur et pédagogique). En cliquant sur un onglet, une page permettant de paramétrer le modèle correspondant apparaît sur la partie principale. La figure 5.9 représente cette interface et montre la page relative au modèle du domaine.



Onglet précisant le modèle sélectionné

Page principale

Figure 5.9 – Interface permettant de paramétrer l'ITS. Ici l'onglet sélectionné correspond au modèle du domaine.

5.3.1 - A Modèle du domaine

Pour paramétrer le modèle du domaine, l'expert sélectionne les organisations concernées. Dans le cas de Gaspar, il s'agit d'équipes. Pour chacune, il peut consulter les procédures, les rôles et les actions qui sont à chaque fois associés. Il peut également inspecter les liens « logiques » entre actions (liaisons indépendantes de la procédure). Enfin, il peut définir et consulter les règles d'usages générales.

Dans l'exemple de la figure 5.9, l'organisation sélectionnée est `Equipe catapultage axiale`. Elle contient la procédure `CatapultageRafale`. L'utilisateur a choisi de consulter le rôle `OfficerLancement`. Les actions associées à ce rôle sont alors affichées (`BaisseDrapeau`, `LeverDrapeau`, `SignedeTete`, *etc.*).

5.3.1 - B Modèle des erreurs

Pour paramétrer le modèle des erreurs, il faut définir les erreurs classiques du domaine (`Concept_on_Error` *cf.* chapitre 3). Plus précisément, le formateur spécifie l'ensemble des règles représentant ces erreurs pour un exercice donné. Rappelons qu'une telle règle est composée de deux parties :

1. Condition : les éléments permettant de déterminer les conditions de l'erreur.
2. Ressources : les informations supplémentaires considérées comme des ressources pédagogiques privilégiées si la règle est satisfaite :
   - ▷ un texte explicatif,
   - ▷ les entités physiques associées (`physicalEntity`),
   - ▷ les éléments externes (vidéo, texte, *etc.*).

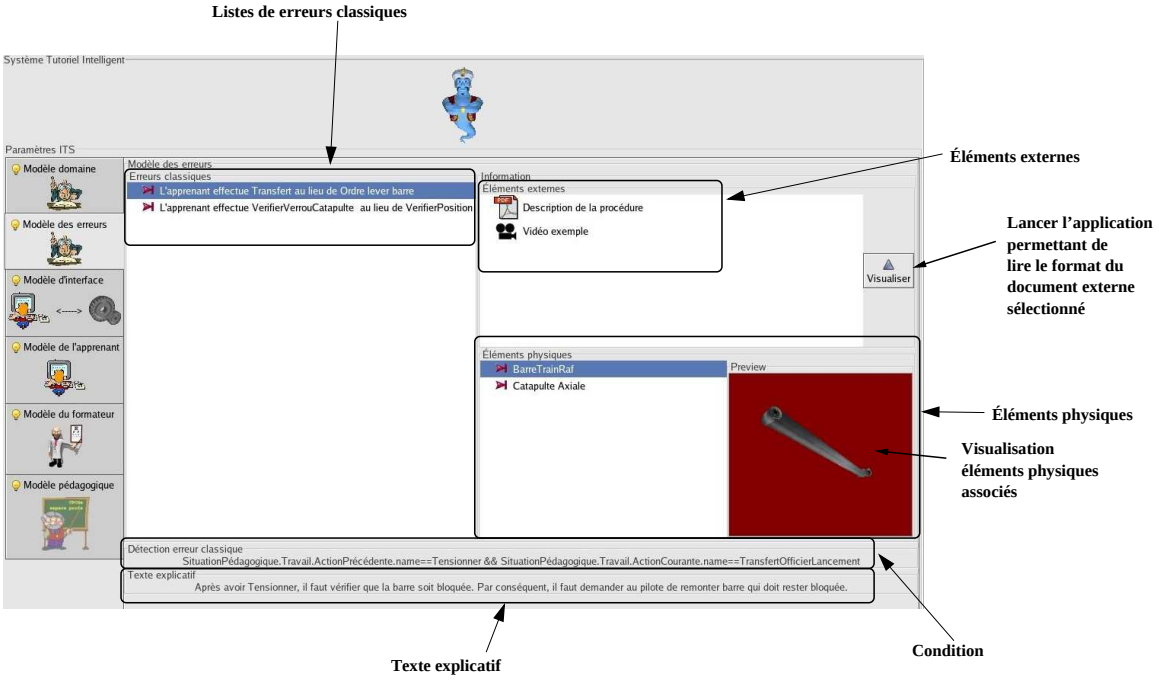


Figure 5.10 – Paramétrage du modèle des erreurs : les erreurs classiques.

Dans l'exemple de la figure 5.10, des erreurs classiques sont listées. Le formateur visualise, ici, l'erreur intitulée « L'apprenant effectue Transfert au lieu de Ordre Lever Barre ». Il s'agit d'une référence à des actions présentes dans la procédure de catapultage. Plus précisément, nous nous intéressons, ici, à une sous-partie de la procédure qui fait intervenir trois rôles (`Maître IA`, `Officier Lancement` et `Pilote`) selon l'ordonnancement suivant :

Maître IA	Officier Lancement	Pilote
① Ordonner Tensionner		
	② Tensionner	
③ Ordonner Lever Barre		
		④ Lever Barre
⑤ Transfert Officier Lancement		

La condition de la règle précise que l'erreur est identifiée si l'action précédente était `Tensionner`, et si l'action courante est `Transfert Officier Lancement`. Dans ce cas, l'agent erreur identifie l'erreur comme faisant partie de sa base d'erreurs classiques. Les ressources associées sont ici :

- ▷ Le texte explicatif : « Après avoir Tensionner, il faut vérifier que la barre soit bloquée. Par conséquent, il faut demander au pilote d'essayer de remonter la barre qui doit rester bloquée ».
- ▷ La barre du train du Rafale (qui est visualisée dans l'interface 3D) et la catapulte axiale. En effet, ces éléments sont des objets clefs dans cette phase de la procédure puisqu'il s'agit de vérifier que la barre de catapultage est bien placée dans le croc de la catapulte pour que le catapultage soit réalisable. Le formateur peut choisir de solliciter des assistances pédagogiques sur ces deux objets (*e.g.* les faire clignoter, les agrandir, *etc.*).
- ▷ Un document écrit rappelant la procédure à suivre et une vidéo illustrant les dégâts occasionnés par cette erreur.

Ces ressources sont des éléments à considérer pour la prise de décision pédagogique lorsque cette erreur classique est identifiée.

Spécifier les erreurs classiques du domaine est optionnel. Rappelons que le modèle des erreurs contient un mécanisme générique, qui est capable de détecter des erreurs et de leur affecter un type (*cf.* chapitre 3). Ce mécanisme ne nécessite aucun paramétrage.

5.3.1 - C Modèle de l'apprenant

Nous nous plaçons dans un apprentissage multi-apprenants. Dans ce cadre, il existe un modèle d'apprenant par étudiant. Le formateur peut ici consulter les informations relatives à chaque apprenant (*cf.* figure 5.11). Ces données renseignent sur les composantes épistémiques et non-épistémiques (*cf.* chapitre 3). Les mises à jour des informations sont automatiques par l'agent apprenant.

Dans l'exemple de la figure 5.11, l'apprenant sélectionné est `Querrec Gabriel`. La sous-partie épistémique indique qu'il a réalisé deux actions correctes et que sa troisième action a provoqué une erreur de type `TeamProceduralError`.

5.3.1 - D Modèle de l'interface

Le modèle de l'interface concerne les PCV de déplacement, d'observation et d'action (*cf.* figure 5.12).

Le PCV de déplacement ne nécessite aucun paramétrage. En effet, les vitesses et directions sont calculées à partir de la dernière action. Les vitesses permettent de déterminer si un apprenant est plus ou moins passif. Les directions peuvent être utilisées pour déterminer si l'apprenant se dirige vers les zones concernées par l'action (vers l'agent à solliciter par exemple). De plus, les zones sont déjà définies lorsqu'on spécifie les instances de classes. Dans l'exemple de la figure 5.12 (c), l'apprenant se positionne dans la `Zone Catapulte Axiale`.

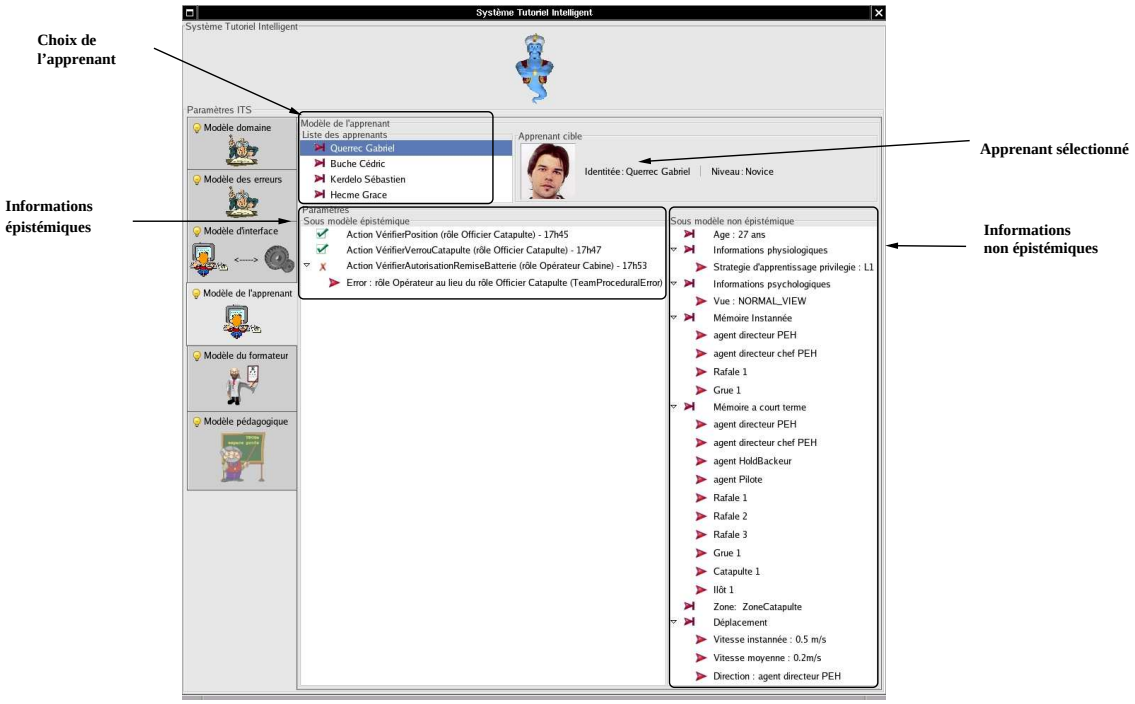
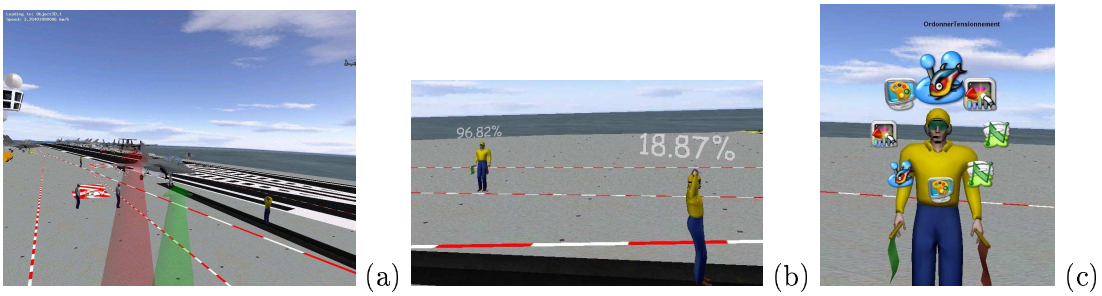


FIGURE 5.11 – Interface permettant de consulter les informations du modèle de l'apprenant préalablement sélectionné.

Cette information peut être prise en compte dans la prise de décision pédagogique, par exemple en comparant la zone de l'apprenant et celle où se place l'agent à solliciter.

Le PCV observation ne nécessite également aucun paramétrage. En effet, les éléments physiques à sélectionner pour être stockés dans les images mémorielles des apprenants, sont définis au niveau du modèle du formateur lorsqu'il spécifie l'exercice (*cf.* section suivante).

Le PCV action doit gérer les interactions entre l'apprenant et l'application. Lorsqu'il sollicite une action, l'apprenant choisit l'entité exécutante. Le PCV action récupère les actions de l'ensemble des rôles mis en jeu dans les équipes participant à l'exercice. Un menu apparaît représentant les possibilités d'actions. Paramétrer le PCV action revient à définir les icônes associées aux actions qui sont utilisées dans le menu.



(a) (b) (c)

FIGURE 5.12 – PCV déplacement (a), observation (b) et action (c).

5.3.1 - E Modèle du formateur

Le formateur doit définir les éléments physiques pertinents liés à l'exercice. Seuls ceux-ci sont stockés dans les images mémorielles du modèle de l'apprenant. Concrètement, le formateur choisit les classes d'entités qui vont être considérées. Il doit également préciser, pour chaque apprenant, les informations suivantes (*cf.* figure 5.13) :

- ▷ L'équipe dans laquelle l'apprenant intervient.
  Il suffit de sélectionner l'équipe concernée parmi celles qui sont présentes dans le modèle du domaine. Dans l'exemple il s'agit de l'`Equipe catapultage axiale`.
- ▷ La procédure à réaliser.
  Une fois que l'équipe est choisie, l'interface propose l'ensemble des procédures connues qui peuvent être réalisées. Le formateur en choisit une.
- ▷ Le(s) rôle(s) à jouer
  De la même manière, le formateur sélectionne les rôles à jouer pour chacun des apprenants.

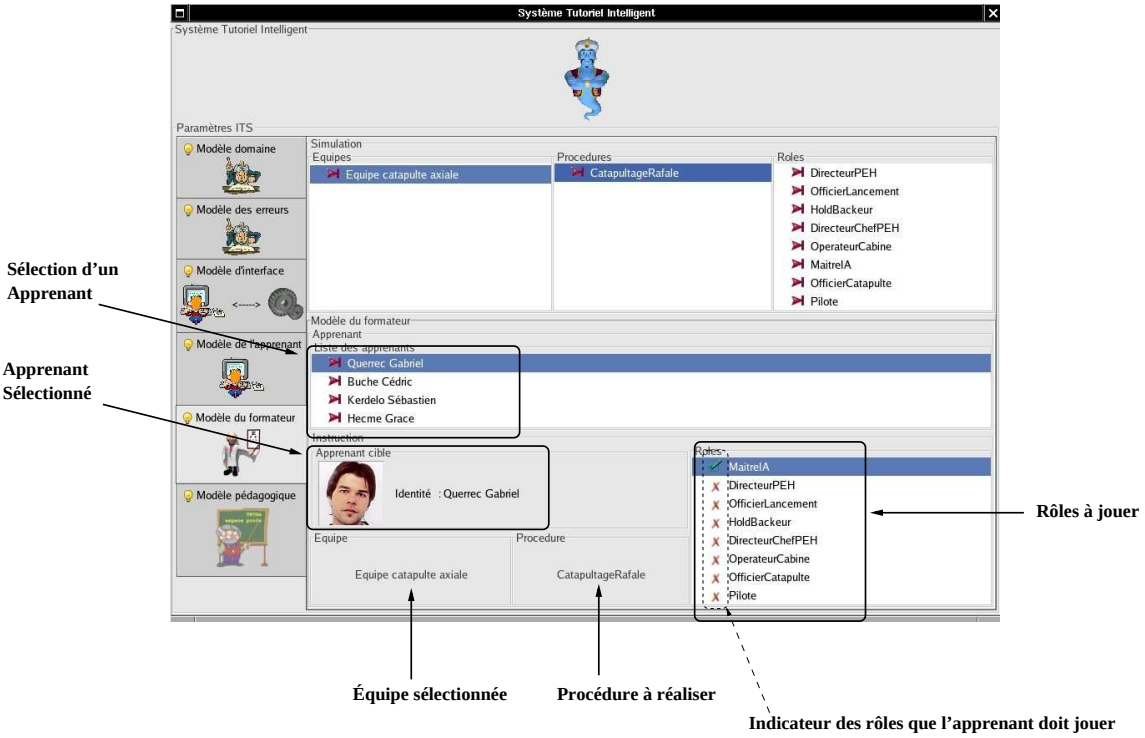


Figure 5.13 – Interface du modèle du formateur.

5.3.1 - F Modèle pédagogique

La partie pédagogique, illustrée en figure 5.14, est composée de trois sous-parties :

1. Les règles qui définissent la « liberté d'action » de l'apprenant.
   Il s'agit de déterminer les actions à proposer à l'apprenant par l'intermédiaire du PCV action, lorsqu'un apprenant sollicite les actions réalisables par un agent. Dans l'exemple de la figure, les apprenants experts disposeront de l'ensemble des actions connues, en

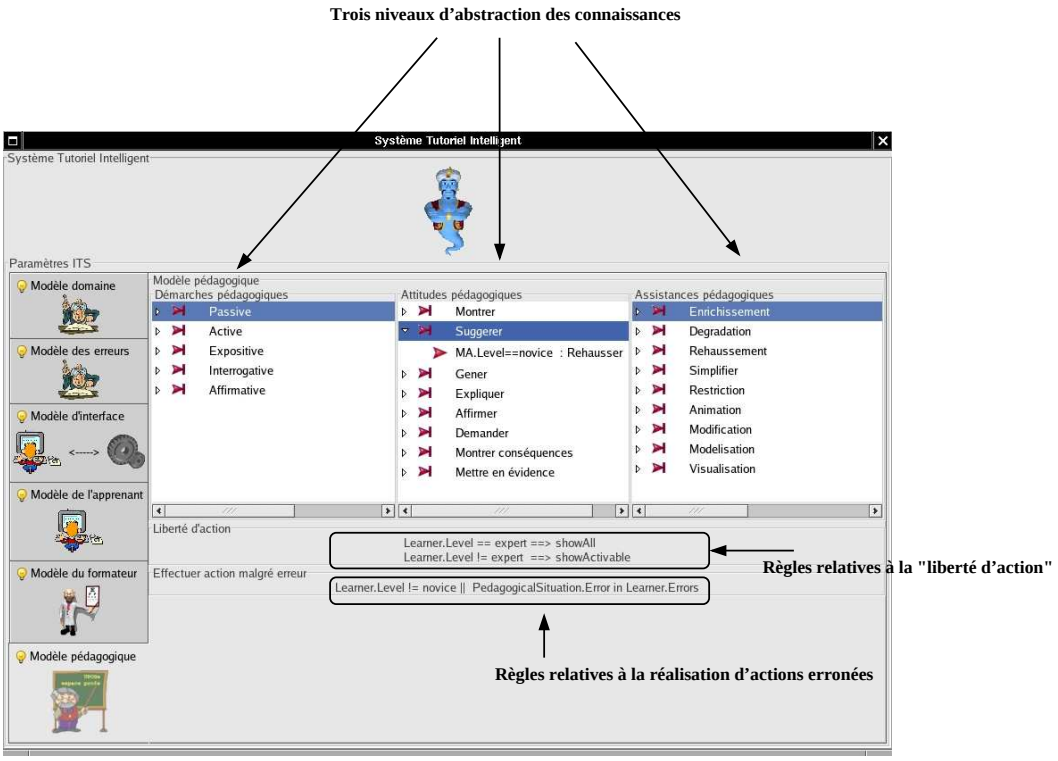


Figure 5.14 – Interface du modèle pédagogique.

fonction des rôles joués par l'agent sollicité. Les autres apprenants obtiendront un sous-ensemble de ces actions, en considérant celles dont les pré-conditions liées à l'opération sont satisfaites.

2. Les règles qui définissent la réalisation d'actions erronées.
   En effet, quelle que soit sa liberté d'action, l'apprenant peut solliciter une action qui va provoquer une erreur. Dans ce cas, il s'agit de déterminer si l'action demandée doit effectivement être réalisée ou non. Dans l'exemple de la figure 5.14, l'action qui engendre une erreur sera réalisée si l'apprenant n'est pas novice, ou s'il a déjà effectué la même erreur.

   Nous pouvons remarquer que, le fait que les actions erronées peuvent être réalisées, nous amène à porter une réflexion sur la cohérence du monde par rapport à la procédure à réaliser. En effet, prenons l'exemple d'un apprenant qui effectue une action provoquant l'explosion d'un Rafale. Imaginons que l'action à réaliser utilise ce Rafale, l'action est irréalisable et donc l'apprenant ne peut pas effectuer la bonne action. L'environnement est devenu incohérent par rapport à la procédure. Pour palier à ce problème, qui sera discuté en perspectives, nous effectuons l'hypothèse simplificatrice que chaque action possède une action inverse qui garantit la cohérence (ce qui n'est pas nécessairement toujours le cas).

3. Les règles du modèle pédagogique permettant d'effectuer des propositions d'assistances pédagogiques au formateur.
   Ces règles sont spécifiées selon les trois niveaux définis dans le chapitre précédent. Elles portent sur les éléments de la situation pédagogique. Le pédagogue peut ajouter, modifier ou supprimer des règles ou des ensembles de règles. Les informations sont

sauvegardées au format XML. Dans l'exemple suivant, nous nous plaçons au niveau d'abstraction `Méthodes Pédagogiques`, nous avons un ensemble de règles (un système de classeurs) appelé `Active`. Une première règle de cet ensemble est satisfaite si l'apprenant est novice (`Apprenant.Niveau==novice`), et si il vient d'effectuer une erreur de type agencement (`Apprenant.Erreur.type==procédural`). Dans ce cas, la règle favorise l'ensemble `Montrer` du niveau qui suit.

```
<PedagogicalModel>
    <ClassifierStage name="Méthodes Pédagogiques">
        <classifierSystem name="Active">
            <classifier>
                <conditions>
                    <item>Apprenant.Niveau==novice</item>
                    <item>Apprenant.Erreur.type==procédural</item>
                </conditions>
                <actions>
                    <item>Montrer</item>
                </actions>
            </classifier>
            ...
        </classifierSystem>
        ...
    </ClassifierStage>
    ...
</PedagogicalModel>
```

Le modèle pédagogique manipule des connaissances qui sont complètement indépendantes de l'application.

5.3.2 Suivi des apprenants

Le formateur a la possibilité de suivre les apprenants dans leur apprentissage en obtenant les données restituant l'analyse de l'ITS. L'interface graphique de suivi est représentée en figure 5.15. Dans cet exemple, nous nous plaçons dans le cas ou deux apprenants sont connectés. L'ITS analyse les actions de chacun. Après avoir sélectionné un apprenant cible, le formateur peut consulter les informations qui restituent l'analyse de l'ITS sur cet étudiant. Ces informations concernent :

- ▷ l'avancement dans la procédure,
- ▷ des statistiques représentant des performances,
- ▷ la situation pédagogique,
- ▷ des propositions d'assistances pédagogiques résultant de la simulation d'un raisonnement de l'agent pédagogique.

Nous détaillons ici chacun de ces points.

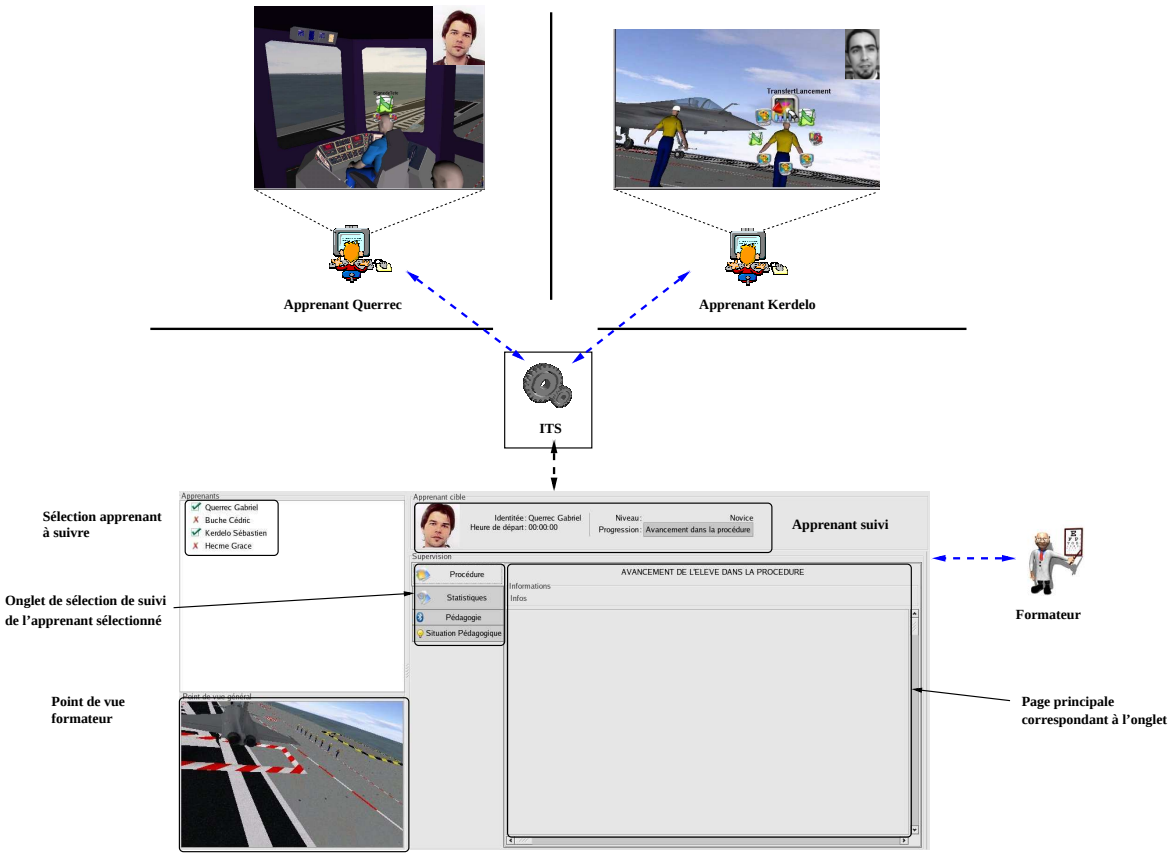


Figure 5.15 – Interface de suivi des apprenants pour le formateur.

5.3.2 - A Procédure

L'interface de suivi de procédure représente l'ordonnancement des actions à effectuer selon les rôles à jouer dans la procédure en cours. Elle combine les informations des modèles :

- ▷ du formateur : il précise la procédure à effectuer et le rôle à jouer par l'apprenant,
- ▷ du domaine : il récupère la structure de la procédure,
- ▷ de l'apprenant : il fournit les actions effectuées,
- ▷ d'interface : il fournit l'action sollicitée.

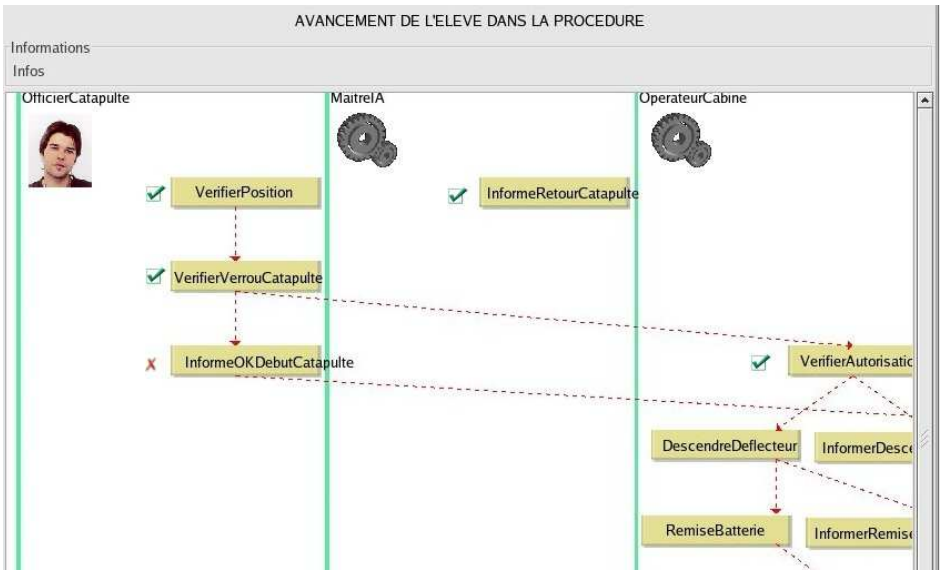


Figure 5.16 – Suivi procédure.

Dans l'exemple de la figure 5.16, l'apprenant joue le rôle `OfficierCatapulte`. Le système joue automatiquement les autres rôles. Des icônes indiquent quelles sont les actions qui ont été réalisées. L'interface précise celles qui n'ont pas été effectuées correctement. En cliquant sur une action, le formateur obtient des informations (état, explication d'une éventuellement erreur, *etc.*), et peut éventuellement solliciter son exécution.

5.3.2 - B Statistiques et performances

L'interface de suivi des statistiques et des performances est représentée en figure 5.17. Elle est composée d'informations générales (durée de la séance, avancement dans la procédure, nombre d'assistances pédagogiques, nombre d'actions effectuées correctement, temps moyen d'inactivité) et de données mesurant les performances (taux de réussite, nombre d'actions erronées).

Le choix des indicateurs de performances n'a pas fait l'objet d'étude particulière. Même si la pertinence de ces critères restent à être démontrée pour être directement pris en compte dans la prise de décision pédagogique, ils font actuellement office de fiche-bilan donnant une bonne idée quant à l'avancée de l'apprenant dans l'exercice.

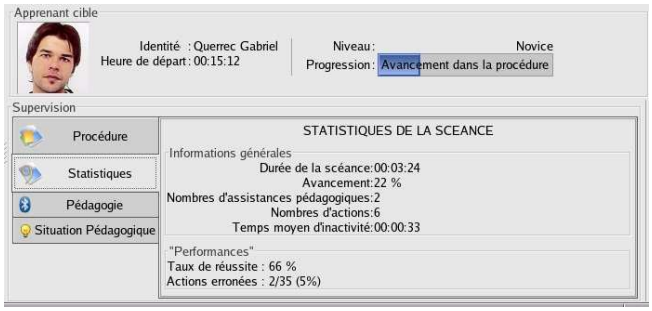


Figure 5.17 – Suivi des statistiques et des performances

5.3.2 - C Situation pédagogique

L'interface de suivi de la situation pédagogique est représentée en figure 5.18. Elle résume les principales informations de la situation pédagogique telle qu'elle a été définie dans le chapitre 3, *i.e.* les données concernant l'apprenant cible (niveau, éventuelle erreur, *etc.*) et les contextes des différentes actions définis par la situation pédagogique : action précédente, action sollicitée, action(s) correcte(s), action(s) correcte(s) sans considération sur les rôles, action(s) suivante(s) et action(s) liée(s).

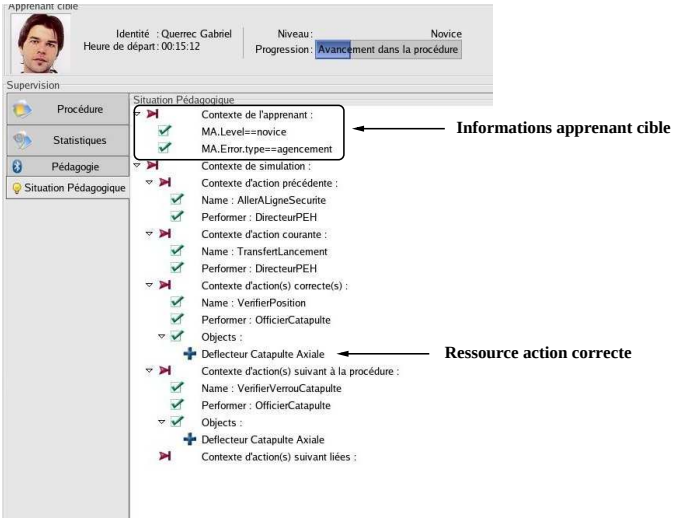


Figure 5.18 – Suivi de la situation pédagogique.

Dans l'exemple, l'action précédente était `AllerALigneSecurite` effectuée par l'agent `DirecteurPEH`. L'action qui vient d'être effectuée par l'apprenant est `TransfertLancement` par le même agent. Or, la seule action correcte est `VerifierPosition` par l'agent `OfficierCatapulte` qui utilise la ressource `Deflecteur Catapulte Axiale`. Par conséquent, c'est une erreur de type agencement. Il n'y a pas d'action liée indépendamment de la procédure. Il n'y a qu'une possibilité d'action suivante : l'action `VerifierVerrouCatapulte`.

5.3.2 - D Activités pédagogiques

Le formateur dispose d'un point de vue personnel de la scène qu'il peut utiliser pour se déplacer dans l'environnement (*cf.* figure 5.19) et pour solliciter des opérations ou des actions. Il dispose également d'un point de vue qui est identique à celui de l'apprenant cible.

De plus, le formateur dispose d'une interface qui lui propose des possibilités ordonnées d'assistances pédagogiques. Ces propositions sont le résultat de la simulation du comportement pédagogique de l'agent pédagogique. Le formateur peut éventuellement visualiser les règles pédagogiques impliquées dans le raisonnement selon les trois niveaux d'abstraction des données (les règles appariées par rapport à la situation sont sur-lignées). Visualiser cette interface n'est pas nécessaire. Néanmoins, elle constitue une source de compréhension qui peut servir au formateur souhaitant comprendre le raisonnement pédagogique utilisé.

Le résultat de ce raisonnement fournit des propositions d'assistance qui utilisent un formalisme « générique ». Dans l'exemple de la figure, un des résultats est d'encadrer la ressource des actions correctes. Pour que ce résultat soit directement exploitable, les propositions effectuées au formateur bénéficient d'une interprétation (nous profitons ici de la réification des connaissances). Suivant l'exemple précédent, nous proposons au formateur la solution consistant à encadrer le déflecteur de la catapulte axiale.

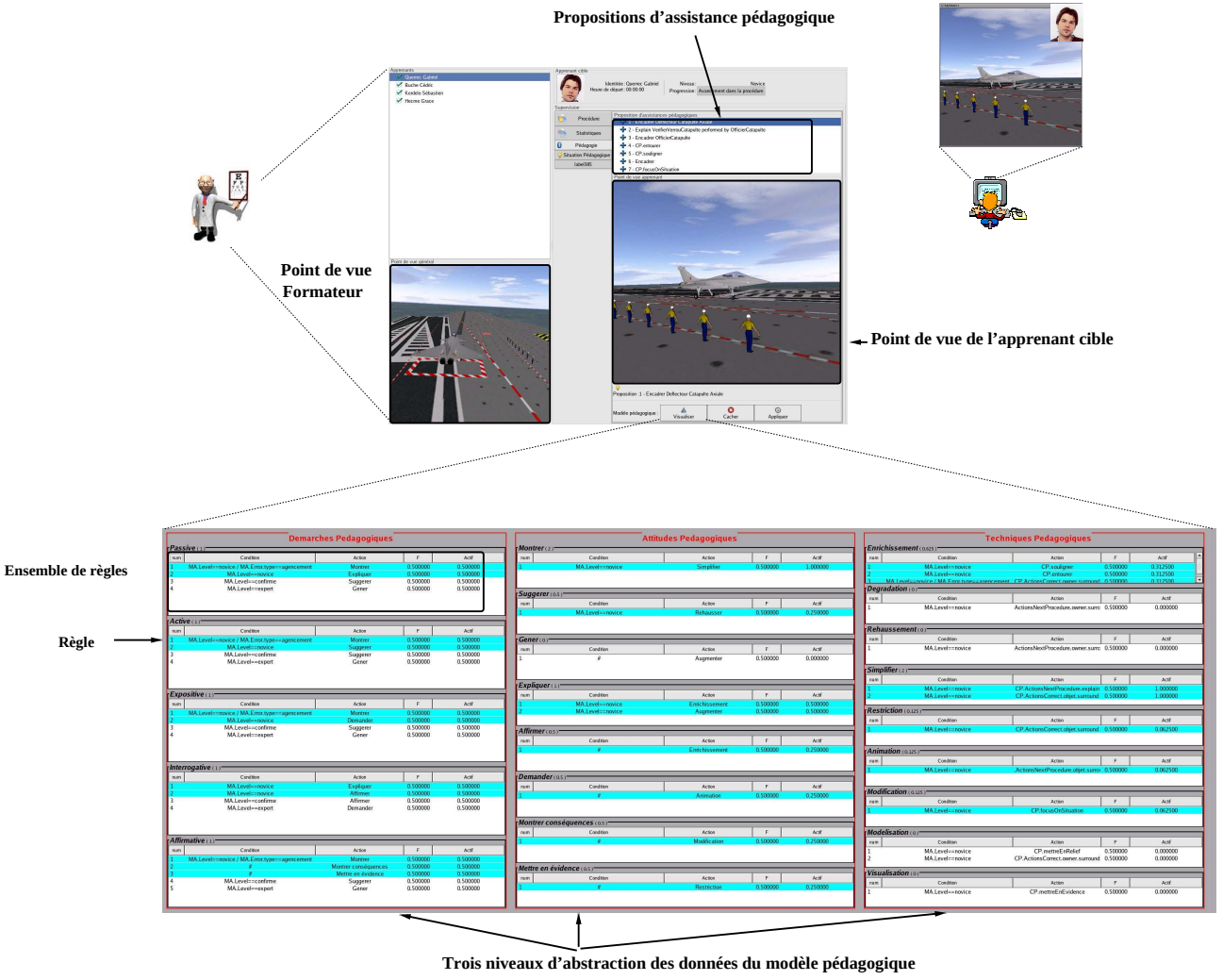


FIGURE 5.19 – Suivi des activités pédagogiques.

5.3.3 Cas d'utilisation

Lors de la simulation, le rôle de l'apprenant est de participer à la réalisation de la procédure cible. En parallèle, le rôle du formateur est de suivre les apprenants et de les faire progresser dans la procédure. Enfin, l'ITS doit assister le formateur dans sa tâche, *i.e.* présenter les connaissances acquises dans l'ITS (activités apprenant, analyse des erreurs, situation pédagogique, *etc.*) et proposer des assistances pédagogiques (résultat du raisonnement de l'agent pédagogique). Pour illustrer les capacités de nos modèles, nous avons choisi un support permettant de montrer au mieux l'aspect dynamique et interactif de la simulation : une bande dessinée. Dans cette bande dessinée, l'image située dans le coin haut à gauche de chaque vignette, renseigne sur la personne[12] qui visualise la scène représentée. Cette bande dessinée nous permet de mettre en évidence la présentation de l'environnement, les possibilités d'action, les erreurs de l'apprenant et les activités pédagogiques.

[12] Il s'agit du formateur  , de l'apprenant GQ  ou l'apprenant SK  .

Figure 5.20 – Présentation de l'environnement.

5.3.3 - A Environnement

La première utilisation de l'application est la navigation dans la scène, permettant de se familiariser avec l'environnement. La présentation du site peut alors être réalisée par le formateur. Une autre possibilité est de laisser chaque apprenant découvrir l'environnement par lui-même, comme c'est le cas dans les vignettes ❶ et ❷.

Dans cet exemple, deux apprenants (nommés GQ et SK) sont en formation et participent à l'exercice. Le formateur doit alors attribuer les rôles de chacun au sein d'équipes, afin de réaliser une procédure particulière. Comme nous pouvons le voir dans la vignette ❸, le formateur choisit la procédure `CatapultageRafale`. Il affecte le rôle `MaitreIA` à l'apprenant GQ au sein de l'`Equipe catapultage axiale`.

Le formateur peut, à tout moment, consulter les informations statiques (*e.g.* âge) ou dynamiques (*e.g.* erreurs effectuées) portant sur les apprenants. Ces connaissances renseignent le formateur et constituent une vision du « profil » de l'apprenant. Dans l'exemple de la vignette ❹, l'apprenant sélectionné est GQ. Il n'a réalisé aucune action jusqu'à présent.

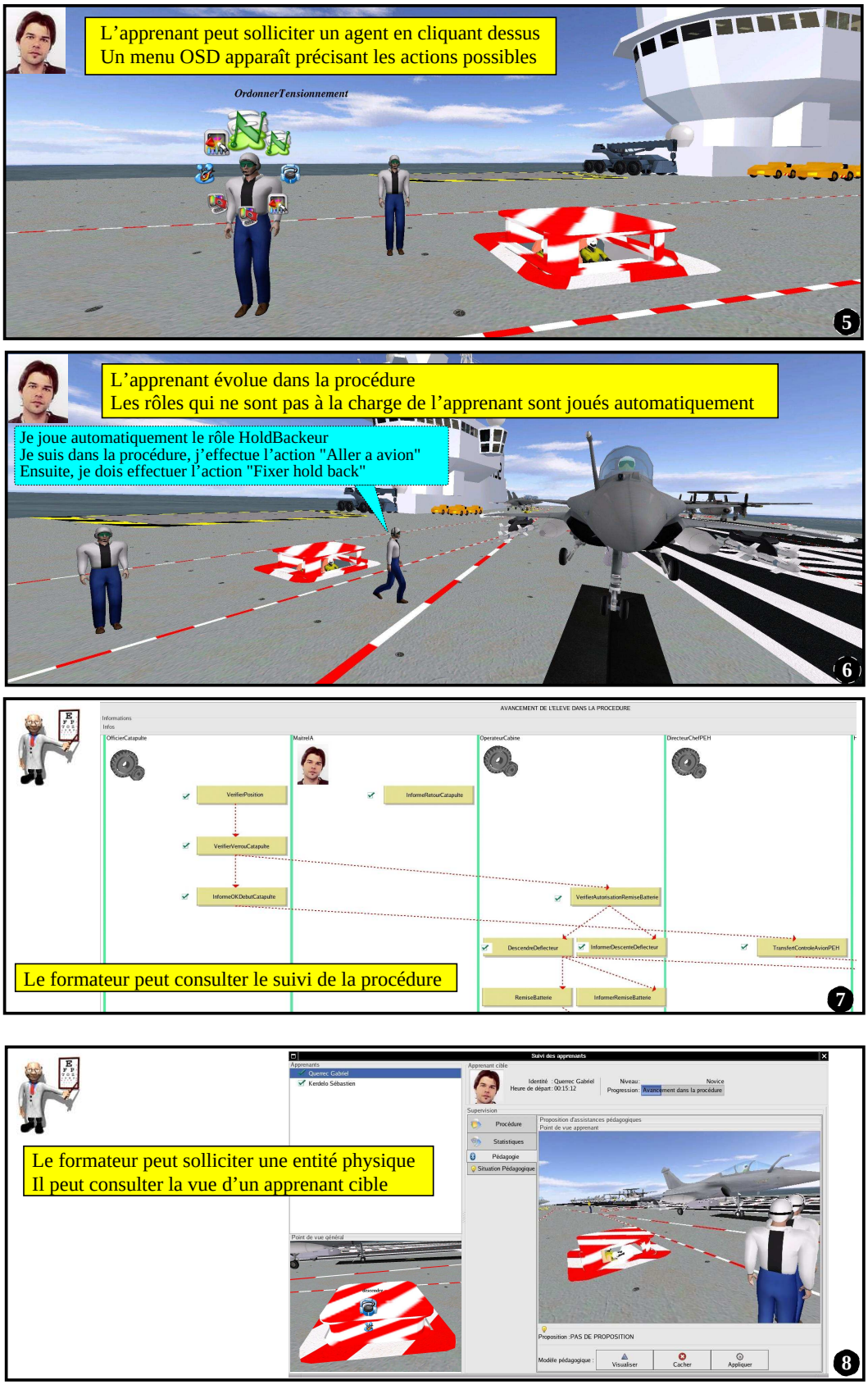


Figure 5.21 – Les actions dans l'environnement.

5.3.3 - B Action

L'apprenant GQ décide de solliciter un agent. Pour cela, il lui suffit de pointer l'entité représentant l'agent et de cliquer. Un menu OSD représentant les possibilités d'actions offertes à l'apprenant par cet agent apparaît. La vignette ❺ montre que l'apprenant dispose de huit actions pour l'agent sélectionné. En tournant la molette de la souris, il fait défiler les possibilités. Ici, s'il valide par le bouton droit, l'apprenant sollicitera l'action `OrdonnerTensionnement`.

Les apprenants évoluent dans la procédure en jouant les rôles sélectionnés par le formateur. Les autres rôles sont joués automatiquement par le système. Dans la vignette ❻, nous pouvons observer l'agent jouant le rôle `HoldBackeur` qui réalise l'action `Aller a avion`. Suivant la procédure, il doit ensuite effectuer l'action `Fixer hold back`.

Pendant ce temps, le formateur peut consulter l'interface de suivi de la procédure. Il peut alors suivre en ligne la progression et se repérer pour déterminer quelles sont les actions à venir. Comme l'indique la vignette ❼, pour l'instant la procédure se déroule sans erreur.

Le formateur dispose d'une interface lui permettant de visualiser le point de vue de l'apprenant cible (ici l'apprenant GQ). Ainsi, il suit ses déplacements en temps réel et peut observer à tout moment le point de vue que l'apprenant a choisi. Le formateur dispose également d'une vue personnalisée, dans laquelle il peut se déplacer librement. Ce point de vue peut, par exemple, lui permettre de suivre les activités de l'ensemble des agents qui agissent en parallèle. Le formateur utilise également ce point de vue pour agir dans l'environnement. En effet, il bénéficie de possibilités d'interactions privilégiées, il peut ainsi solliciter des opérations ou des actions. Dans l'exemple de la vignette ❽, le formateur sollicite la cabine de catapultage. Il peut, par exemple, aider l'apprenant en agrandissant la cabine si cette dernière est une ressource de l'une des actions que l'apprenant doit réaliser. Le même type d'interaction permet de gêner ou perturber un apprenant expérimenté.

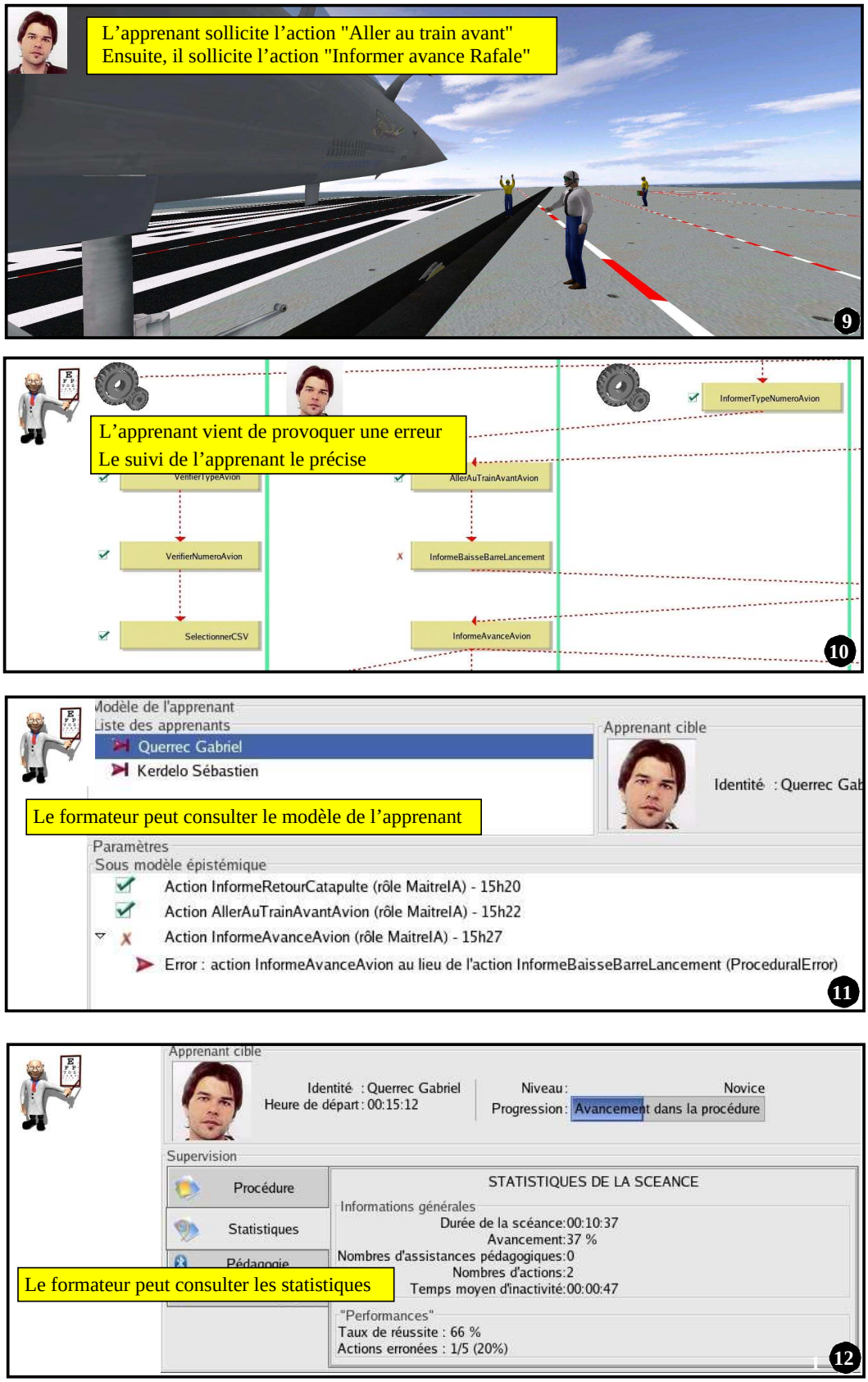


Figure 5.22 – L'apprenant effectue une action qui provoque une erreur.

5.3.3 - C Erreur

En vignette ❾, la réalisation de la procédure est bien avancée. L'apprenant effectue l'action `Aller au train avant`. Il sollicite ensuite l'action `Informer avance Rafale`. Il s'agit d'une erreur puisque cette action ne fait pas partie des actions correctes à la suite de l'action `Aller au train avant`.

Le modèle de l'apprenant indique les actions sollicitées et, ici, explicite qu'il s'agit d'une erreur. Plus précisément, le résultat de l'analyse de l'agent erreur est récupéré. Le formateur peut ainsi remarquer que l'erreur est de type `ProceduralError`. En effet, cette action ne fait pas partie des actions à réaliser, néanmoins elle est de sa responsabilité (c'est une action contenue dans les rôles qu'il doit jouer). Comme nous pouvons le voir en vignette ❿, le formateur peut observer sur l'interface de suivi de procédure qu'une erreur est survenue (symbole ✗ au lieu de ✔). Cette information est également disponible au niveau du modèle de l'apprenant (vignette ❶❶).

Le formateur peut consulter la situation pédagogique qui lui indique les informations prises en compte pour la prise de décision de l'agent pédagogique. Rappelons que ces connaissances portent sur l'apprenant et sur le travail à réaliser. Il s'agit notamment de contextes d'actions (*cf.* chapitre 3). Dans notre cas, nous obtenons les connaissances illustrées dans le tableau 5.1 de la page 160. Le formateur dispose d'une interface qui lui indique toutes ces informations.

Le formateur remarque que l'apprenant est en difficulté puisqu'il n'a pas réussi à progresser dans la procédure depuis son erreur. Le formateur choisit de considérer le raisonnement simulé de l'agent pédagogique. Cependant, avant de choisir parmi les assistances pédagogiques proposées, il souhaite consulter les statistiques portant sur l'apprenant cible (vignette ❶❷).

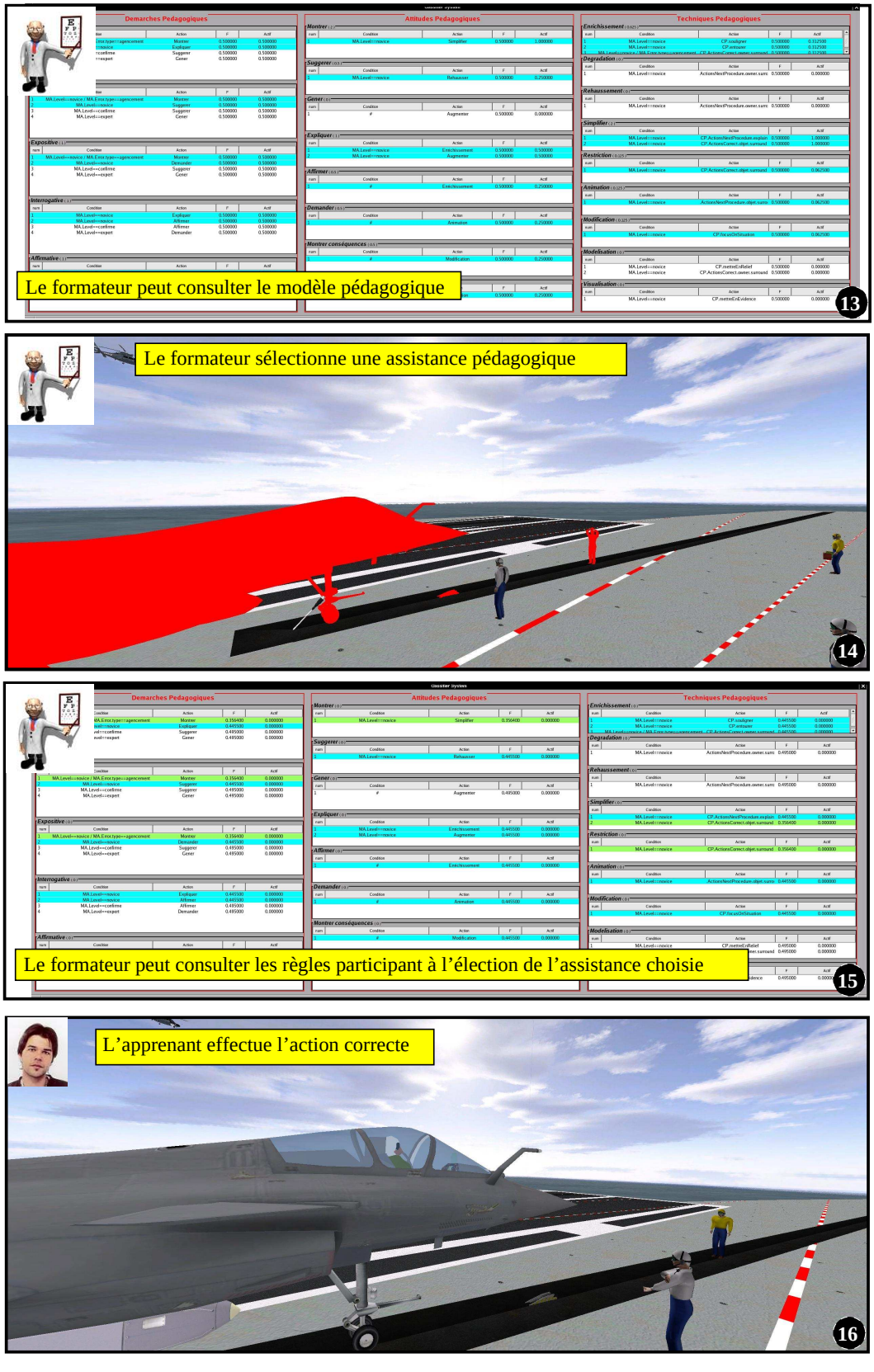


Figure 5.23 – La pédagogie dans l'environnement.

5.3.3 - D Assistances pédagogiques

Le formateur peut éventuellement consulter le modèle pédagogique. Dans ce cas, il visualise les trois niveaux correspondant aux démarches, attitudes et techniques pédagogiques spécifiées par le pédagogue. Chaque niveau est constitué d'ensemble de règles, chacune étant soumis à des conditions. Dans la vignette ❶❸, les règles appariées sont sur-lignées. Ainsi le formateur peut comprendre le raisonnement pédagogique simulé.

L'agent pédagogique propose une liste ordonnée d'assistances pédagogiques. Le formateur en choisit une. Dans le cas de la vignette ❶❹, le formateur a sélectionné l'assistance qui met en rouge clinquant la barre de lancement, le Rafale, la catapulte et le directeur PEH. En fait, le raisonnement pédagogique simulé avait abouti à l'assistance consistant à mettre en rouge clinquant les ressources des actions correctes. La proposition générique a été transformée pour s'adapter à la situation et pour que le formateur puisse choisir une assistance concrète.

L'agent pédagogique prend en compte le choix du formateur. En effet, il détermine automatiquement l'ensemble des règles qui ont participé à l'élection de l'assistance pédagogique choisie (couleur différente sur le vignette ❶❺). Il applique alors l'algorithme d'apprentissage artificiel.

Enfin, comme le montre la vignette ❶❻, fort de l'aide pédagogique, l'apprenant sollicite l'action correcte.

Contextes d'Actions	Actions	Opérations associées
précédente	`Aller au train avant` ▷ Ressources ⇨ Train avant, Rafale ▷ Conditions ⇨ Rafale arrêté ▷ Objectif ⇨ être proche du train avant ▷ Résultat ⇨ position (performer - rafale) $<$ seuil	`Aller a` ▷ Nom ⇨ Aller A ▷ Post-conditions ⇨ distance à la cible $<$ seuil ▷ Arguments ⇨ cible, vitesse
sollicitée	`Informer avance Rafale` ▷ Ressources ⇨ Rafale, directeur PEH ▷ Objectif ⇨ le directeur PEH ordonne au pilote du Rafale d'avancer ▷ Résultat ⇨ le directeur PEH reçoit l'autorisation d'effectuer le geste ordonnant au pilote d'avancer	`Informer` ▷ Nom ⇨ Informer ▷ Post-conditions ⇨ cible a reçu le message ▷ Arguments ⇨ cible, message
correcte(s) / correcte(s) sans considération sur les rôles	`Informer Baisse Barre Lancement` ▷ Ressources ⇨ Barre de lancement, Rafale, catapulte, directeur PEH ▷ Objectif ⇨ le directeur PEH ordonne au pilote du Rafale de baisser sa barre de lancement ▷ Résultat ⇨ le directeur PEH reçoit l'autorisation d'effectuer le geste ordonnant au pilote de baisser sa barre de lancement	`Informer` ...
suivante(s)	`Ordonner Baisse Barre Lancement` ▷ Ressources ⇨ Barre de lancement, Rafale, catapulte, pilote ▷ Objectif ⇨ le pilote fait baisser la barre de lancement ▷ Résultat ⇨ le pilote reçoit l'ordre de descendre la barre de lancement du Rafale à hauteur de la catapulte	`Ordonner` ▷ Nom : Ordonner ▷ Post-conditions ⇨ cible a reçu l'ordre ▷ Arguments ⇨ cible, ordre

Tableau 5.1 – Partie des connaissances de la situation pédagogique.

5.4 Bilan

Nous venons de voir l'intégration des modèles que nous avons proposés dans l'environnement GASPAR. Dans cette application, l'ITS manipule les connaissances sur l'environnement au travers des organisations. En effet, ce concept permet d'accéder à la représentation des connaissances portant sur les entités, les équipes et les procédures. Par conséquent, les organisations constituent la seule connaissance qui doit être fournie à l'ITS.

Dans le cas de l'application exemple GASPAR, le modèle d'ITS et l'environnement ont été implémentés en parallèle. Les développements ne sont liés que par le code qui suit, ce qui témoigne du caractère générique de notre modélisation :

```
ArRef<Scheduler> simulationInit(void)
{
  ArRef<Scheduler> sched=new_RealTimeScheduler(1e-3);                  // simulation temps réel

  /****************** Application ********************/
  ArRef<Gaspar> appli = new_GASPAR("gaspar.xmi");                      // chargement fichier XMI
  appli->createInstances("gaspar.xml");                                // chargement fichier XML

  /******************** ITS ************************/
  ArRef<IntelligentTutoringSystem> its = new_IntelligentTutoringSystem();
  its->create( appli->getMessageService(),                             // communication par messages
               appli->getTeams() );                                    // organisations de GASPAR

  return(sched);
}
```

De plus, nous pouvons rappeler que les modèles utilisés pour définir les entités de l'environnement (modèle VEHA) et pour spécifier la partie organisationnelle, *i.e.* les équipes et les procédures (`Modèle organisationnel`), sont définis en UML. Par conséquent, l'application GASPAR est pratiquement générée en totalité. Ainsi, la généricité des modèles présentés permet le développement rapide d'applications.

Bien plus, la modélisation de l'ITS ne considère pas les connaissances sur une application particulière[13], son utilisation est donc générique. Pour illustrer la généricité de notre proposition, nous présentons l'intégration de notre modélisation d'ITS dans d'autres applications.

Nous avons recréé un environnement destiné à la formation qui avait déjà fait l'objet d'un cadre illustratif pour le projet MASCARET : l'application SÉCURÉVI [14] (Querrec et al., 2003b) (*cf.* figure 5.24). Elle permet d'aider à la formation des officiers sapeurs-pompiers à la gestion opérationnelle et au commandement. 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.

Dans cette application, une équipe de sapeurs-pompiers nommée FPT (Fourgon Pompe Tonne) a pour responsabilité d'attaquer l'incident (éteindre un foyer, ralentir la progression d'une fuite de gaz, *etc.*). Une équipe FPT est organisée autour de cinq rôles : le chef d'agrès,

[13] Seules les règles d'usage, les erreurs classiques et les objets à considérer dans l'image mémorielle du modèle de l'apprenant sont à définir pour chaque application.

[14] SÉCURÉVI : Sécurité et Réalité Virtuelle.

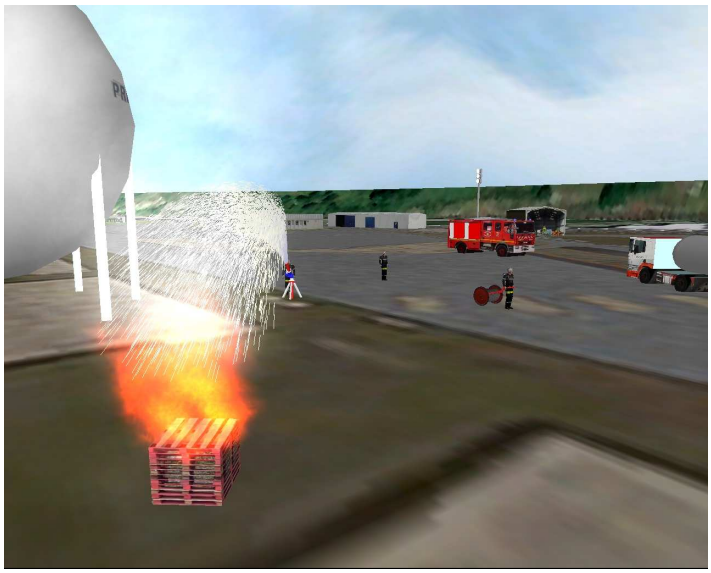


Figure 5.24 – Exemple de scène dans l'application SécuRéVi.

le chef du binôme d'alimentation, l'équipier du binôme d'alimentation, le chef du binôme d'attaque et l'équipier du binôme d'attaque. L'exercice simulé représente 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.

Gaspar et SécuRéVi sont des applications très similaires. En effet, elles sont destinées à l'apprentissage procédural et collaboratif. Bénéficiant de la représentation des connaissances fournie par le modèle Veha et le `Modèle organisationnel`, l'intégration de notre ITS n'a demandé aucun travail supplémentaire. Seules les règles d'usage, les erreurs classiques et les objets à considérer dans l'image mémorielle du modèle de l'apprenant ont été précisés. Notons que ces informations ne peuvent en aucun cas être génériques.

Gaspar et SécuRéVi sont destinés à l'apprentissage procédural dans un cadre professionnel. Nous disposons de l'application Eve (Popovici et al., 2004) qui diffère, dans le sens où elle propose d'apprendre la lecture dans un milieu scolaire. Nous travaillons actuellement à l'intégration de nos modèles dans cet environnement. L'aspect non procédural entraîne des modifications de notre modélisation, qui se situent au niveau de la caractérisation des erreurs et des connaissances contenues dans la situation pédagogique. Fort du caractère générique de notre proposition, le reste de la modélisation est directement utilisable. Nous pouvons également souligner que nous conservons une grande partie des règles du modèle pédagogique qui ne sont pas liées à la notion de procédure.

Cette section profite de ces trois applications pour rappeler le processus complet permettant d'obtenir un environnement virtuel de formation (*cf.* figure 5.25).

① Conception de l'EVF

La conception de l'EVF s'effectue en trois étapes :

1. L'expert du domaine et le concepteur de l'EVF spécifient en UML les modèles utilisés dans les applications (GASPAR, SÉCURÉVI et EVE). Ces modèles sont ensuite exportés au format XMI.
2. S'appuyant sur ces modèles, le concepteur de l'EV et le formateur construisent les instances en utilisant un logiciel dédié. L'ensemble de ces instances forme l'environnement qui sert de base aux exercices. L'environnement est alors décrit dans un fichier au format XML.
3. Les données contenues dans les fichiers XMI et XML permettent la création des applications.

② Conception pédagogique

En parallèle, le pédagogue spécifie les règles contenues dans le modèle pédagogique. Le formateur spécifie l'exercice à réaliser en sélectionnant les organisations et les procédures à réaliser. Il effectue également l'affectation des différents rôles.

③ Simulation

Les apprenants évoluent dans la simulation en sélectionnant les actions qu'ils souhaitent réaliser. L'ITS offre alors au formateur un suivi des procédures, des performances, des activités et caractéristiques de l'apprenant, et de la situation pédagogique. L'ITS effectue des propositions d'assistances pédagogiques au formateur (mode manuel) ou les applique directement dans l'environnement (mode automatique).

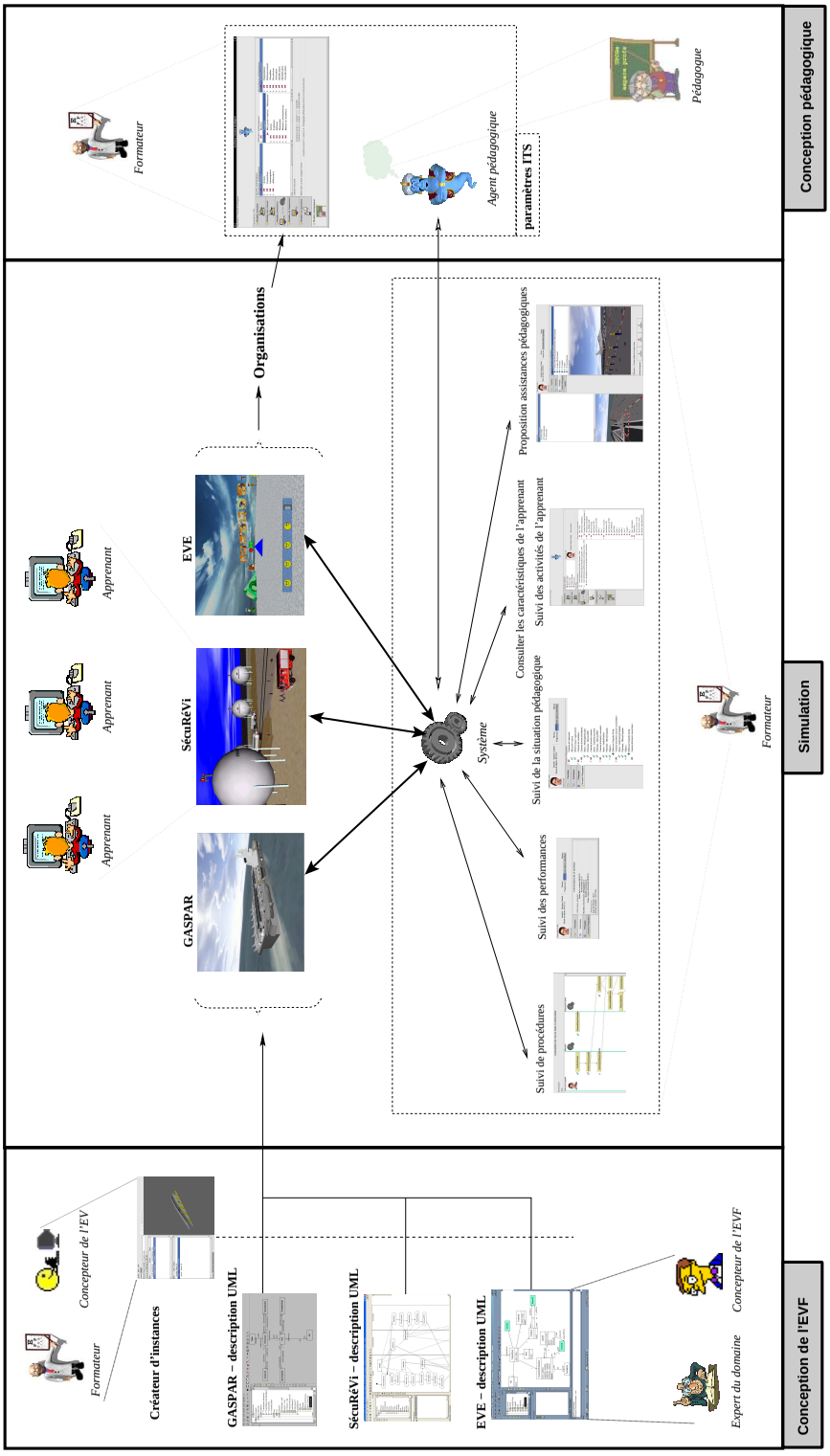


FIGURE 5.25 – Conception et simulation d'un environnement virtuel de formation.

Chapitre 6

Bilan et perspectives

DANS ce manuscrit de thèse, nous avons proposé un modèle d'ITS destiné à l'apprentissage de compétences en environnement virtuel. Ce chapitre est l'occasion de dresser un bilan de nos travaux, de discuter des apports de notre proposition et d'envisager certaines perspectives.

6.1 Bilan

L'étude que nous avons présentée constitue une contribution aux environnements virtuels de formation. Plus particulièrement, nous avons proposé une approche fondée sur les *systèmes tutoriels intelligents*.

Le chapitre 2 s'intéresse aux systèmes informatiques destinés à la formation. Nous proposons de montrer l'intérêt de la Réalité Virtuelle (RV) et des Systèmes Tutoriels Intelligents (ITS) pour l'apprentissage de compétences. Les ITS permettent de représenter, sous la forme de modèles, les informations sur un domaine d'apprentissage, l'apprenant, la communication entre l'apprenant et le système, et les méthodes d'enseignement. Ils analysent les activités de l'apprenant et proposent des assistances pédagogiques individualisées. La RV est définie selon trois axes : autonomie, interaction et immersion. L'autonomie permet la variabilité des contextes, ce qui constitue une condition nécessaire à la construction de compétences transférables. Nous évaluons ensuite l'intégration des ITS dans les Environnements Virtuels de Formation (EVF) existants. Nous montrons que cette intégration est actuellement limitée.

Dans le chapitre 3, nous proposons une modélisation du système tutoriel intelligent. La représentation des connaissances portant sur l'environnement simulé et sur le travail à réaliser (procédural et collaboratif) offre la possibilité de manipuler les informations nécessaires à la prise de décision pédagogique. Un modèle d'entité virtuelle, un modèle organisationnel et les diagrammes d'activités UML nous permettent de réifier ces informations. Ces connaissances sont utilisées par l'ITS au travers de plusieurs modèles (domaine, apprenant, erreur, interface, formateur et pédagogique) implémentés sous la forme d'agents. Une analyse des rôles de chacun définit leur base de connaissances et leurs comportements participant à la prise de

décision. Ils analysent les activités de l'apprenant dans l'univers 3D et caractérisent, de manière générique, les erreurs effectuées. Les interactions entre agents produisent un ensemble de connaissances, appelé situation pédagogique, considéré pour simuler un raisonnement pédagogique. Elle fournit les connaissances permettant d'identifier des situations particulières et extrait de l'environnement des ressources privilégiées pour déclencher des assistances pédagogiques.

Dans le chapitre 4, nous présentons le modèle de l'agent pédagogique. Plus précisément, nous décrivons le comportement de prise de décision qui effectue les propositions d'assistance pédagogique au formateur. Une étude des architectures comportementales existantes, tenant compte des besoins en terme de spécification du modèle, de hiérarchisation, de modularité, de réactivité et d'adaptabilité, nous amène à choisir les systèmes de classeurs. L'agent pédagogique organise ses connaissances selon une structure hiérarchisée, composée de trois niveaux représentants différentes abstractions des connaissances : les méthodes, les attitudes et les techniques pédagogiques. Chaque niveau contient des ensembles de règles représentant les différentes manières d'aborder une démarche, une attitude ou une technique. Les règles effectuent le lien entre les différents niveaux, chaque règle favorisant un ensemble du niveau inférieur. Une description générale des mécanismes, des principaux paramètres et l'introduction de hiérarchie mis en jeu dans les systèmes de classeurs, est présentée. Nous montrons alors l'adaptation de ces systèmes pour le comportement de l'agent pédagogique. Dans un premier temps, le modèle propose un mécanisme de diffusion au travers des trois niveaux, pour finalement ordonner les propositions d'assistances pédagogiques effectuées au dernier niveau. Dans un second temps, le modèle intègre un mécanisme d'apprentissage artificiel de type *bucket brigade*, basé sur un renforcement provenant des choix du formateur, permettant de personnaliser les aides pédagogiques.

Le chapitre 5 est consacré à la mise en application de notre proposition d'ITS. Nous décrivons un processus générique permettant de créer des environnements virtuels de formation. Dans un premier temps, le concepteur spécifie les modèles UML qui représentent les entités de l'environnement, les agents et les organisations. Ils sont ensuite transformés en données au format XMI et par conséquent deviennent manipulables. Le modèle Veha, implémenté en ARéVi, interprète le fichier XMI et génère la structure des éléments de l'environnement. Enfin, les instances sont précisées par des fichiers XML, dont certains peuvent être générés à l'aide d'utilitaires logiciels. L'intégration de notre modèle d'ITS utilise la connaissance sur les organisations impliquées, et bénéficie de la représentation des connaissances effectuées par le modèle Veha pour effectuer ses traitements. Les différentes parties de notre modèle d'ITS sont paramétrables à l'aide d'une interface graphique interactive. L'ITS propose également des interfaces de suivi des apprenants, où le formateur peut visualiser l'analyse portant sur la procédure, les statistiques, la situation pédagogique et les activités pédagogiques. L'exemple support de ce chapitre est l'environnement Gaspar, qui simule l'activité aviation sur un porte-avions. Ce chapitre s'achève en soulignant le caractère générique de notre proposition, idée qui est appuyée par la mise en œuvre de nos modèles dans d'autres environnements.

6.2 Discussion

Avant de conclure cette thèse, il convient de discuter des apports de notre proposition. L'étude proposée dans ce manuscrit débute par un état de l'art qui analyse les utilisations des ITS existants au sein des environnements virtuels de formation. Nous avons alors montré que le système HAL est le plus abouti, nous avons également souligné les points qui pouvaient être améliorés. En effet, dans ce système, le modèle pédagogique est en partie dépendant de l'exercice. Plus précisément, les erreurs et les stratégies pédagogiques doivent être définies. De plus, le formateur ne peut choisir qu'entre deux méthodes pédagogiques (explicative ou active). Le travail réalisée dans cette thèse doit alors répondre aux problèmes de généricité et de modularité du modèle pédagogique.

Dans cette thèse, nous avons défini un ITS complet (modèle du domaine, modèle des erreurs, modèles de l'apprenant, modèle d'interface, modèle du formateur et modèle pédagogique). Sans revenir sur chaque point de nos travaux, nous pouvons montrer en quoi notre proposition répond aux difficultés des modèles existants. Dans un premier temps, nous pouvons remarquer que nous avons proposé un mécanisme qui réalise une caractérisation générique des erreurs. Le modèle pédagogique peut alors raisonner sur le type de l'erreur sans pour autant l'avoir préalablement précisé pour un exercice particulier. Plus généralement, les connaissances utilisées dans le raisonnement pédagogique ne portent pas sur une particularité de l'exercice à réaliser. Ainsi, une règle pédagogique ne considère pas des informations spécifiques, *e.g.* « si l'apprenant peut observer le Rafale alors ... », mais utilisera plutôt des connaissances génériques indépendantes de l'exercice (« si les ressources des actions correctes sont visibles alors ... »). De la même manière, les assistances pédagogiques, bien que proposant des solutions concrètes au formateur, *e.g.* « faire clignoter le pompier », manipulent également des connaissances génériques indépendantes de l'exercice (« Faire clignoter les *performer* des actions suivantes »). Ainsi, la généricité de notre proposition est une caractéristique forte qui est illustrée par l'intégration de notre ITS au sein de plusieurs applications. Bien plus, le modèle pédagogique de notre ITS présente une forte modularité, puisqu'il offre la possibilité d'ajouter, de supprimer ou de modifier chacune des composantes du modèle pédagogique participant à la prise de décision pédagogique (règle ou ensemble de règles).

Ainsi, notre proposition répond aux problèmes soulignés dans notre état de l'art. De surcroît, le mécanisme d'apprentissage artificiel adapte les propositions d'assistances pédagogiques au couple apprenant-formateur. Toutefois, nous pouvons nous interroger sur les possibilités liées à l'utilisation de notre ITS dans le cadre d'un apprentissage non-procédural[1]. Envisager ce type de formation impose de repenser les éléments qui sont étroitement liés à la notion de procédure, *i.e.* la caractérisation des erreurs et les connaissances de la situation pédagogique.

[1] Rappelons au lecteur que notre proposition d'ITS avait posé pour hypothèse le cadre de l'apprentissage procédural.

6.3 Perspectives

Les réflexions menées au cours de cette étude ouvrent de nombreuses perspectives. Nous envisageons d'orienter nos travaux futurs vers les sujets suivants.

Application du modèle d'ITS à d'autres applications

Le modèle proposé dans cette étude peut être intégré facilement dans des applications utilisant le modèle Veha. La prochaine étape consiste à élargir l'utilisation de notre modèle à des environnements autres que Gaspar et SécuRéVi, notamment dans l'application Eve (Environnement Virtuel pour Enfants). Le projet Eve vise l'apprentissage de la lecture par les enfants de classes primaires. Nous effectuons actuellement les modifications nécessaires pour que Veha soit le support de cette application. Cette révision nous permettra d'intégrer notre modèle d'ITS et, par conséquent, de simuler un raisonnement pédagogique permettant d'effectuer le suivi des élèves. Au delà de l'aspect non procédural, cette étude nous permettra de tester notre modèle dans un environnement virtuel qui n'est plus destiné à la formation professionnelle, mais à l'apprentissage en milieu scolaire. Les apprenants ne sont plus des adultes mais des enfants. Nous devrons alors mesurer l'impact des différences liées aux objectifs et au public concerné, et en tirer les conséquences pour raffiner notre modèle.

Représenter l'environnement physique

Dans cette étude, nous avons utilisé une représentation des connaissances sur l'environnement simulé et sur l'environnement social (travail procédural et collaboratif). Nos travaux ne proposent pas de représentation des connaissances liées aux phénomènes physiques. En effet, bien que les phénomènes physiques soient simulés, nous n'avons pas fourni une modélisation permettant de les représenter. Les connaissances liées aux phénomènes ne sont donc pas manipulables et ne peuvent pas être exploitées dans un but pédagogique. Des travaux futurs devront s'attacher à la représentation de ces connaissances, déjà abordée dans (Querrec, 2002), et à leur utilisation pour la prise de décision pédagogique.

Cohérence du monde

Quelle que soit l'application utilisée, il faudra résoudre le problème lié à la cohérence du monde suite à une action erronée. En effet, comme nous l'avons souligné dans le chapitre 5, l'apprenant peut réaliser des actions qui sont différentes de celles qu'il aurait dû effectuer et par conséquent, il peut modifier l'état du monde. La cohérence entre l'état du monde et les actions à réaliser n'est dès lors plus garantie. Prenons l'exemple d'une action erronée qui a pour conséquence l'explosion d'une citerne de gaz, son exécution implique qu'on ne pourra plus réaliser l'action qui consiste à remplir la cuve de la citerne. Ce problème est un sujet de recherche en soi. Il ne s'agit pas seulement de détecter une incohérence entre le scénario à réaliser et l'état du monde. Il faut aller plus loin et proposer un modèle de l'environnement qui proposerait les fonctionnalités d'un « magnétoscope dynamique » : retour arrière sur action, avance rapide, possibilité de débuter un exercice en milieu de procédure, rejeu, *etc.*

Agents pédagogiques

L'agent pédagogique que nous avons défini, effectue des propositions d'assistances pédagogiques au formateur. Le rôle de notre système est donc d'aider le formateur. Pour aller plus loin, notre système pourrait jouer d'autres rôles dans le processus d'apprentissage. Ainsi, nous nous intéressons non plus à un agent pédagogique, mais à plusieurs agents pédagogiques qui jouent le rôle de compagnon, assistant, tuteur, gêneur, *etc.* (Payr, 2003). Ces agents forment une organisation particulière. Ils peuvent collaborer pour prendre des décisions pédagogiques, comme dans BAGHERA (Webber et al., 2002).

Utiliser les normes et standards éducatifs

Les travaux futurs devront raffiner les modèles proposés dans cette thèse en prenant en compte des approches de normalisation qui décrivent les objets d'apprentissage. L'idée est de représenter les connaissances liées aux ressources pédagogiques afin de pouvoir les manipuler dans diverses applications. Pour cela, nous devrons considérer les standards existants concernant la gestion des systèmes d'information (LOM : Learning Object Metadata[2] (Forte et al., 1999)), l'ingénierie des composants logiciels (SCORM : Sharable Content Object Reference Model[3]) et l'ingénierie pédagogique (EML : Educational Modeling Language[4]). De plus, pour aller plus loin, nous devrons considérer la formation comme une progression au travers de plusieurs exercices que nous pourrons représenter selon des formes normalisées en étudiant les langages tels que IMS-LD[5] (Learning Design) (IMS, 2003).

Validation expérimentale

Dans ce manuscrit, nous n'avons pas proposé de méthode permettant de valider notre modèle. En effet, la démarche généralement utilisée consiste à comparer les performances entre des groupes d'apprenant qui ont été formés selon une méthode traditionnelle et ceux qui ont bénéficié de notre simulateur. Le problème de l'environnement GASPAR réside dans le fait qu'il est impossible d'effectuer des mesures de performances en condition réelle pour plusieurs raisons. La principale porte sur l'indisponibilité des ressources et des personnels mis en jeu pour effectuer les mesures de performances. En effet, prendre possession du porte-avions, et expérimenter des manœuvres mettant en jeu des personnels ainsi que des matériels coûteux et dangereux (Rafale, missiles, *etc.*), ne nous seront probablement jamais permis. Nous devons dans l'avenir proposer d'autres méthodes de validation permettant de mesurer l'impact de notre système en terme de formation.

Évaluation

Plus généralement, des expérimentations sur les résultats de l'utilisation de la RV et des ITS sur l'acquisition de compétences devront être effectuées. En effet, si l'intérêt de l'utilisation

[2] `http ://ltsc.ieee.org/wg12/`

[3] ADL/SCORM, ADL Sharable Content Object Reference Model Version 1.3, Working draft 0.9, 2002.

[4] `http ://eml.ou.nl/introduction/`

[5] `http ://www.imsglobal.org/learningdesign/`

des assistances fournies par les ITS dans un environnement virtuel de formation pour l'apprentissage de compétences transférables a pu être vérifié au travers de quelques applications (Loftin et Kenney, 1995; Thibault, 2002), les résultats des quelques évaluations réalisées à ce jour (*cf.* section 2.3.3) témoignent de la difficulté d'identifier des conditions d'apprentissage optimales. Par exemple, une étude de Wilson (1999a) cherche à évaluer l'efficacité respective de deux conditions d'apprentissage en RV. Dans une première condition, des sujets doivent mémoriser des objets et leur localisation dans un environnement, tout en se déplaçant au sein de cet environnement virtuel (condition active). Dans une condition passive, les sujets observent simplement l'écran, à coté d'un sujet « actif ». Alors que l'auteur attendait des performances plus élevées dans des tests de rappel et de reconnaissances pour les sujets « actifs », les résultats de l'analyse de variance ne montrent aucune différence significative entre les deux groupes (conditions). Cette étude démontre toute la prudence avec laquelle nous pouvons envisager de futures formations à l'aide d'environnements virtuels. Non seulement, il est nécessaire de tester leur efficacité réelle dans la perspective de transférer des acquis vers le monde réel, mais il reste encore à définir et à évaluer des conditions d'apprentissage optimales face à un ordinateur, ou des conditions d'apprentissage alternant des situations de simulations et des situations réelles. Cette variabilité des contextes au sein et autour de la situation simulée reste actuellement, et d'un point de vue théorique, une condition *sine-qua-non* du transfert de compétences.

Glossaire

ABITS	Agent Based Intelligent Tutoring System
ACS	Anticipatory Classifier System
ADELE	Agent for Distance Education - Light Edition
ALECSYS	A LEarning Classifier SYStem
ARéVI	Atelier de Réalité Virtuelle
CERV	Centre Européen de Réalité Virtuelle
CS	Classifiers System
EA	Equipe d'Accueil
EAO	Environnement Assisté par Ordinateur
EDF	Electricité De France
EIAH	Environnement Informatique ou Interactif pour l'Apprentissage Humain
EIAO	Environnement Interactif ou Intelligement Assisté par Ordinateur
EML	Educational Modeling Language
ENIB	École Nationale d'Ingénieurs de Brest
EV	Environnement Virtuel
EVE	Environnement Virtuel pour les Enfants
EVF	Environnement Virtuel de Formation
FFH	Free-Flow Hierarchy
FPT	Fourgon Pompe Tonne
GASPAR	Gestion de l'Activité aviation et des Sinistres sur Porte-avions par la Réalité virtuelle
HAL	Helpful Agent for Learning
HGCS	Hierarchical Generalized Classifiers System

IA	Intelligence Artificielle
ITS	Intelligent Tutoring System
LISYC	Laboratoire d'Informatique des SYstèmes Complexes
LOM	Learning Object Metadata
Macs	Modular Anticipatory Classifier System
Mascaret	Multi Agent System for Collaborative and Adaptive Realistic Environment for Training
Mhics	Modular HIerarchical Classifier Systems
Ocs	Organizational Classifier System
OMG	Object Management Group
OSD	On Screen Display
PCV	Primitives Comportementales Virtuelles
RV	Réalité Virtuelle
SCORM	Sharable Content Object Reference Model
SécuRéVi	Sécurité et Réalité Virtuelle
SNCF	Société Nationale des Chemins de Fer
SPI	équipe Simulation Participative et Immersive du CERV
Steve	Soar Training Expert for Virtual Environments
TGV	Train Grande Vitesse
UBO	Université de Bretagne Occidentale
UML	Unified Modeling Language
Veha	Virtual Environment supporting Human Activity
WTA	Winner-Takes-All
Xcs	eXtended Classifier System
XMI	XML Metadata Interchange
XML	eXtensible Markup Language
Yacs	Yet Another Classifier System
Zcs	Zeroth level Classifier System

Publications

Cette section énumère les articles écrits pendant cette thèse.

Revues internationales avec comité de lecture

Buche, C., Querrec, R., Chevaillier, P., et Kermarrec, G. (2005). Apports des systèmes tutoriaux intelligents et de la réalité virtuelle à l'apprentissage de compétences. *In Cognito – Cahiers Romans de Sciences Cognitives (CRSC)*, 2(2) :53–87. ISSN 1267–8015.

Buche, C., Querrec, R., De Loor, P., et Chevaillier, P. (2004). MASCARET : A pedagogical multi-agent system for virtual environment for training. *International Journal of Distance Education Technologies (JDET)*, 2(4) :41–61. ISSN 1539-3100.

Querrec, R., Buche, C., Maffre, E., et Chevaillier, P. (2004). Multiagents systems for virtual environment for training. application to fire-fighting. *International Journal of Computers and Applications (IJCA)*, 1(1) :25–34. ISSN 1710-2251.

Revues nationales avec comité de lecture

Buche, C., Septseault, C., et De Loor, P. (2006). Proposition d'un modèle générique pour l'implémentation d'une famille de systèmes de classeurs. *Revue des Sciences et Technologies de l'Information, série Intelligence Artificielle (RSTI–RIA)*, 20(1). A paraître.

Tisseau, J., Parenthoën, M., Buche, C., et Reignier, P. (2006). Comportements perceptifs d'acteurs virtuels autonomes. une application aux cartes cognitives floues. *Revue des Sciences et Technologies de l'Information, série Techniques et Sciences Informatiques (RSTI–TSI)*. A paraître.

Groupe de travail

Buche, C. (2004). Comportement pédagogique dans un environnement virtuel de formation. Communication au groupe de travail *Environnement Interactif pour l'Apprentissage Humain (EIAH)* du GdR I3 – 4.3.

Conférences internationales avec comité de lecture et publication des actes

Buche, C., Parenthoën, M., et Tisseau, J. (2002). Learning by imitation of behaviors for autonomous agents. Dans Medhi, Q., Gough, N., et Cavazza, M., éditeurs, *3rd International Conference on Intelligent Games and Simulation (GAME-ON'02)*, pages 89–93, London, United Kingdom. SCS. ISBN : 90-77039-10-4.

Buche, C. et Querrec, R. (2005a). Intelligent tutoring system for MASCARET. Dans Richir, S. et Taravel, B., éditeurs, *7th virtual reality international conference (VRIC'05)*, pages 105–108, Laval, France. ISBN : 2-9515730-4-9.

Buche, C. et Querrec, R. (2005b). Simulate pedagogical reasoning in a virtual environment for training. Dans *International Conference on Computers and Advanced Technology for Education (CATE'05)*, pages 183–187, Aruba. ISBN : 0-88986-522-1.

Buche, C., Querrec, R., et De Loor, P. (2005a). Système tutoriel intelligent pour l'apprentissage de travail procedural et collaboratif. Dans Herzig, A., Lespérance, Y., et Mouaddib, A., éditeurs, *Troisièmes journées francophones Modèle Formels de l'Interaction (MFI'05)*, pages 205–210, Caen. Cépaduès. ISBN : 2.85428.697.9.

Buche, C., Querrec, R., De Loor, P., et Chevaillier, P. (2003a). MASCARET : Pedagogical multi-agents system for virtual environment for training. Dans Kunii, T., Soon, S., et Sourin, A., éditeurs, *International Conference on Cyberworlds (CW'03)*, pages 423–430, Singapore. School of Computer Engeneering, Nanyang Technological Unviversity, IEEE Computer Society. ISBN : 0-7695-1922-9.

Buche, C., Querrec, R., et Le Gall, C. (2005b). Intelligent tutoring system for procedural and collaborative training. Dans *8th World Conference on Computers in Education (WCCE'05)*, University of Stellenbosch, Cape Town, South Africa. ISBN : 1-920-01711-9.

Buche, C., Querrec, R., Maffre, E., Chevaillier, P., et De Loor, P. (2003b). MASCARET : multiagent system for virtual environment for training. Dans Richir, S., Richard, P., et Taravel, B., éditeurs, *5th virtual reality international conference (VRIC'03)*, pages 159–164, Laval, France. ISBN : 2-9515730-2-2.

Parenthoën, M., Buche, C., et Tisseau, J. (2002). Action learning for autonomous virtual actors. Dans Mayorga, R. V. et Segovia-De Los Ríos, A., éditeurs, *3rd International Symposium on Robotics and Automation (ISRA'02)*, pages 549–554, Toluca, Mexico.

Popovici, D. M., Buche, C., Querrec, R., et Harrouet, F. (2004). An interactive agent-based learning environment for children. Dans Nakajima, M., Hatori, Y., et Sourin, A., éditeurs, *International Conference on Cyberworlds (CW'04)*, pages 233–240, Tokyo, Japan. IEEE Computer Society. ISBN : 0-7695-2140-1.

Querrec, R., Buche, C., Maffre, E., et Chevaillier, P. (2003a). Multiagents systems for virtual environnement for training. Dans Uskov, V., éditeur, *International Conference on Computers and Advanced Technology in Education (CATE'03)*, pages 647–652, Rhodes, Greece. ACTA Press. ISBN : 0-88986-361-X.

Querrec, R., Buche, C., Maffre, E., et Chevaillier, P. (2003b). SécuRéVi : virtual environments for fire-fighting training. Dans Richir, S., Richard, P., et Taravel, B., éditeurs, *5^{th} virtual reality international conference (VRIC'03)*, pages 169–175, Laval, France. ISBN : 2-9515730-2-2.

Références bibliographiques

Aka, M. et Frasson, C. (2002). ASIMIL : Overview of a Distance Learning Flight-Training System. Dans Cerri, S. A., Gouardères, G., et Paraguaçu, F., éditeurs, *Intelligent Tutoring Systems*, pages 484–495, Biarritz, France. Springer. ISBN 3-540-43750-9.

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Journal of the Association for Computing Machinery*, 26(11) :832 – 843.

Anderson, J. R. (1988). The expert module. Dans Polson, M. C. et Richardson, J. J., éditeurs, *Foundations of Intelligent Tutoring Systems*, pages 21–53. Erlbaum, Hillsdale, NJ.

Anderson, J. R. (1990). Analysis of student performance with the LISP tutor. Dans Fredericksen, N., Glaser, R., Lesgold, A., et Shaffo, M., éditeurs, *Diagnostic Monitoring of Skill and Knowledge Acquisition*, pages 27–50, Hillsdale, NJ. Erlbaum.

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, NJ : Erlbaum.

Anderson, J. R. et Reiser, B. J. (1985). The lisp tutor. *Byte*, 10 :159–175.

Arruarte, A., Fernandez-Castro, I., Ferrero, B., et Gree, J. (1997). The IRIS shell : How to build ITSs from pedagogical and design requisites. *International Journal of Artificial Intelligence in Education*, 8(3-4) :341–381.

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

Aïmeur, E., Frasson, C., Labonde, et M. (2001). The role of conflicts in the learning process. *ACM Journal of SIGCUE Outlook*, 27(2) :12–27.

Balacheff, N. (1998). Éclairage didactique sur les EIAH en mathématiques. Dans *Colloque annuel du Groupe de Didactique des Mathematiques du Québec*, pages 11–42, Université de Concordia, Montréal, Canada.

Barry, A. (1993). The Emergence of High Level Structure in Classifier Systems - A Proposal. *Irish Journal of Psychology*, 14(3) :480–498.

Barry, A. (1996). Hierarchy Formulation Within Classifiers System – A Review. Dans Goodman, E. G., Uskov, V. L., et Punch, W. F., éditeurs, *Evolutionary Computation and Its Applications (EVCA'96)*, pages 195–211, Moscow. The Presidium of the Russian Academy of Sciences.

Baylor, A. L. (1999). Intelligent agents as cognitive tools for education. *Educational Technology*, 39(2) :36–40.

Bellman, R. (1957a). *Dynamic Programming*. Princeton University Press. ISBN : 069107951X.

Bellman, R. (1957b). A markov decision process. *Journal of Mathematical Mechanics*, 6 :679–684.

Bliss, J. P., Tidwell, P. D., et Guest, M. A. (1997). The effectiveness of virtual reality for administering spatial navigation training to firefighters. *Presence*, 6 :73–86.

Blumberg, B., Downie, M., Ivanov, Y., Berlin, M., Johnson, M. P., et Tomlinson, B. (2002). Integrated learning for interactive synthetic characters. Dans *29th annual conference on Computer graphics and interactive techniques (SIGGRAPH'02)*, pages 417–426, New York, NY, USA. ACM Press.

Bonelli, P., Parodi, A., Sen, S., et Wilson, S. (1990). NEWBOOLE : A Fast GBML System. Dans *International Conference on Machine Learning*, pages 153–159, San Mateo, California. Morgan Kaufmann.

Brooks, R. A. (1986). A Robust Layered Control System For A Mobile Robot. *IEEE Transactions on Robotics and Automation*, RA-2(1) :14–23. ISSN 0882-4967.

Brown, J., Burton, R., et Bell, A. (1975). SOPHIE : a step towards a reactive learning environment. *International Journal of Man-Machine Studies*, 7 :675–696.

Brown, J. S. et Burton, R. R. (1978). A paradigmatic example of an artificially intelligent instructional system. *International Journal of Man-Machine Studies*, 10 :323–339.

Bruillard, E. (1997). *Les machines à enseigner*. Hermès, Paris. ISBN : 2-86601-610-6.

Bruner, J. (1983). *Le développement de l'enfant. Savoir dire, savoir faire*. PUF, Paris.

Burkhardt, J. M., Lourdeaux, D., et Mellet d'Huart, D. (2003). *Le traité de la réalité virtuelle*, chapitre La conception des environnements virtuels pour l'apprentissage. Les Presses de l'Ecole des Mines, deuxième édition.

Burns, H. L. et Capps, C. G. (1988). *Foundations of Intelligent Tutoring Systems*, chapitre Foundations of Intelligent Tutoring Systems : An Introduction, pages 1–19. Erlbaum, Hillsdale, NJ.

Burton, R. R. (1982). *Intelligent Tutoring Systems*, chapitre Diagnosing bugs in a simple procedural skill, pages 157–183. Academic Press, London, England.

Burton, R. R. et Brown, J. S. (1982). *Intelligent Tutoring Systems*, chapitre An investigation of computer coaching for informal learning activities, pages 79–98. Academic Press, London, England.

Butz, M. et Pelikan, M. (2001). Analyzing the evolutionary pressures in xcs. Dans Spector, L., Goodman, E. D., Wu, A., Langdon, W., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., et Burke, E., éditeurs, *Genetic and Evolutionary Computation Conference (GECCO'01)*, pages 935–942, San Francisco, California, USA. Morgan Kaufmann.

Butz, M., Sastry, K., et Goldberg, D. (2003). Tournament selection : stable fitness pressure in XCS. Dans Cantu-Paz, E., éditeur, *Genetic and Evolutionary Computation Conference (GECCO'03)*, pages 1857–1869. Springer.

Butz, M. et Wilson, W. S. (2002). An algorithmic description of XCS. *Journal of Soft Computing*, 6(3–4) :144–153.

Capuano, N., Marsella, M., et Salerno, S. (2000). ABITS : An agent based intelligent tutoring system for distance learning. Dans *International Workshop in Adaptative and Intelligent Web-based Educational Systems. Fifth International Conference on Intelligent Tutoring Systems (ITS'00)*, Montreal, Quebec, Canada.

Card, S. K., Moran, T. P., et Newell, A. (1983). *The psychology of human-computer interaction*. Lawrence Erlbaum, London.

Cazeaux, E., Devillers, F., et Saint-Romas, C. (2005). GIAT Virtual training. Formation à la maintenance. Dans *Laval Virtual, First International VR-Learning Seminar, virtual reality international conference (VRIC'05)*.

Chan, T. W. et Baskin, A. B. (1990). *Intelligent Tutoring Systems : At the Crossroads of Artificial Intelligence and Education*, chapitre Learning companion systems, pages 6–33. Norwood, NJ : Ablex.

Chou, C. Y., Chan, T. W., et Lin, C. J. (2003). Redefining the learning companion : the past, present, and future of educational agents. *Computers & Education*, 40(3) :255–269.

Clancey, W. J. (1983). GUIDON. *Journal of Computer-Based Instruction*, 10(1) :8–14.

Cliff, D. et Ross, S. (1995). Adding Temporary Memory to ZCS. Rapport technique CSRP347, School of Cognitive and Computing Sciences, University of Sussex.

Collins, A., Warnock, E. H., et Passafiume, J. (1975). *The Psychology of Learning and Motivation*, volume 9, chapitre Analysis and Synthesis of Tutorial Dialogues, pages 49–87. Academic Press, New York.

Constantino, A. et Suthers, D. (2000). A coached collaborative learning environment for entity-relationship modeling. Dans Gauthier, G., Frasson, C., et VanLehn, K., éditeurs, 5^{th} *International Conference on Intelligent Tutoring Systems*, volume 1839 of *Lecture Notes in Computer Science*, pages 325–333, Montréal, Canada. Springer.

Coutaz, J. (1994). Evaluation techniques : Exploring the intersection of HCI and software engineering. Dans Taylor, R. N. et Coutaz, J., éditeurs, *Software Engineering and Human-Computer Interaction, Workshop on SE-HCI : Joint Research Issues*, volume 896 of *Lecture Notes in Computer Science*, pages 35–48, Sorrento, Italy. Springer.

Crowder, N. A. (1959). *Automatic Teaching : The State of the Art*, chapitre Automatic tutoring by means of intrinsic programming, pages 109–116. Wiley, NY.

Davies, J. R., Gertner, S. A., Lesh, N., Rich, C., Sidner, C. L., et Rickel, J. (2001). Incorporating tutorial strategies into an intelligent assistant. Dans *6^{th} international conference on Intelligent User Interfaces*, pages 53–56, Santa Fe, New Mexico, United States. ACM Press New York, NY, USA. ISBN :1-58113-325-1.

De Boer, B. et Cañamero, D. (1999). *Learning Sites : Social and Technological Resources for Learning*, chapitre Situated Learning in Autonomous Agents, pages 236–248. Pergamon Press, Amsterdam.

De Montmollin, M. (1984). *L'intelligence de la tâche. Eléments d'ergonomie cognitive.* Peter Lang.

De Terssac, G. (1996). *Savoirs théoriques et savoirs d'action*, chapitre Savoirs, compétences et travail, pages 223–248. PUF, Paris.

Delestre, N. (2000). *METADYNE, un Hypermèdia Adaptatif Dynamique pour l'Enseignement.* Thèse de doctorat, Université de Rouen, France. Informatique.

Demazeau, Y. (1995). From interactions to collective behaviour in agent-based systems. Dans *European Conference on Cognitive Science*, pages 97–132.

Depover, C., Giardina, M., et Marton, P. (1998). *Les environnements d'apprentissage multimédia.* L'Harmattan.

Deutsch, D. (2003). *L'étoffe de la réalité.* Cassini, Paris. Traduction F. Balibar.

Dillenbourg, P. (1994). The role of artificial intelligence techniques in training software. Dans *European Conference and Specialist Trade Fair for Educational and Information Technology (LEARNTEC'94)*, pages 295–308.

Donikian, S. (2001). HPTS : a behaviour modelling language for autonomous agents. Dans *the fifth international conference on Autonomous agents (AGENTS '01)*, pages 401–408, New York, NY, USA. ACM Press.

Donikian, S. (2004). Modèlisation, contrôle et animation d'agents virtuels autonomes évoluant dans des environnements informés et structurés. Habilitation à diriger les recherches, Université de Rennes I.

Dorigo, M. (1995). Alecsys and the AutonoMouse : Learning to Control a Real Robot by Distributed Classifier Systems. *Machine Learning*, 19 :209–240.

Dorigo, M. et Bersini, H. (1994). A Comparison of Q-Learning and Classifier Systems. Dans *From Animals to Animats 3 : Third International Conference on Simulation of Adaptive Behavior (SAB'94)*, pages 248–255.

Fahlman, S. E. (1979). *NETL : A System for Representing and Using Real-World Knowledge.* MIT Press, Cambridge, MA.

Flavell, J. H. (1985). Développement métacognitif. Dans Bideaud, J. et Richelle, M., éditeurs, *Psychologie développementale, problèmes et réalités*, pages 29–41, Bruxelles : Mardaga.

Forte, E., Haenni, F., Warkentyne, K., Duval, E., Cardinaels, K., Vervaet, E., Hendrikx, K., Wentland Forte, M., et Simillion, F. (1999). Semantic and pedagogic interoperability mechanisms in the ARIADNE educational repository. *SIGMOD Record*, 28(1) :20–25.

Fuchs, P. et Burkhardt, J. M. (2003). *Le traité de la réalité virtuelle*, chapitre Approche théorique et pragmatique de la réalité virtuelle. Les Presses de l'Ecole des Mines, deuxième édition.

Fuchs, P. et Moreau, G. (2003). *Le traité de la réalité virtuelle*. Presses de l'école des mines, Paris, seconde édition.

George, C. (1983). *Apprendre par l'action*. PUF.

Gibson, J. J. (1958). Visually controlled locomotion and visual orientation in animals. *British Journal of Psychology*, 49(3) :182–194.

Gick, M. L. et Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology*, 15 :1–38.

Girard, B., Robert, G., et Guillot, A. (2001). Jeu vidéo et intelligence artificielle située. *In Cognito*, 22 :57–72.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley.

Goodman, B., Soller, A., Linton, F., et Gaimari, R. (1998). Encouraging student reflection and articulation using learning companion. *International Journal of Artificial Intelligence in Education*, 9 :237–255.

Graesser, A. C., Wiemer-Hastings, K., Wiemer-Hastings, P., et Kreuz, R. (1999). Autotutor : a simulation of human tutor. *Journal of Cognitive Systems Research*, 1 :35–51.

Gruau, J. Y., Doireau, P., et Poisson, R. (1998). Conception et usage de la simulation. *Le Travail Humain*, 61 :361–385.

Gérard, P. (2002). *Systèmes de classeurs : étude de l'apprentissage latent*. Thèse de doctorat, Université Paris VI.

Halff, H. M. (1988). Curriculum and instruction in automated tutors. Dans Polson, M. C. et Richardson, J. J., éditeurs, *Foundations of Intelligent Tutoring Systems*, pages 79–108. Erlbaum, Hillsdale, NJ.

Harrouet, F., Tisseau, J., Reignier, P., et Chevaillier, P. (2002). oRis : un environnement de simulation interactive multi-agents. *Revue des Sciences et Technologies de l'Information, série Techniques et Sciences informatiques (RSTI–TSI)*, 21(4) :499–524.

Heguy, O., Sanza, C., Berro, A., et Duthen, Y. (2002). GXCS : A generic classifier system and its application in a real time cooperative behavior simulations. Dans *International Symposium and School on Advanced Distribued System (ISADS'02)*, Guadalajara, Mexique.

Hoc, J. M. (1990). *Traité de Psychologie Cognitive 2*, chapitre Les connaissances concernant les procédures, pages 46–50. Bordas.

Holland, J. (1976). Adaptation. Dans Rosen, R. et Snell, F. M., éditeurs, *Progress in theoretical biology*. New York : Plenum.

Holland, J. et Reitman, J. (1978). Cognitive systems based on adaptive algorithms. Dans Waterman, D. A. et Hayes-Roth, F., éditeurs, *Pattern-directed inference systems*. New York : Academic Press. Reprinted in : Evolutionary Computation. The Fossil Record. David B. Fogel (Ed.) IEEE Press, 1998. ISBN : 0-7803-3481-7.

Holland, J. H. (1975). *Adaptation in Natural and Artifcial Systems*. University of Michigan Press, Ann Arbor, MI.

Holland, J. H. (1980). Adaptive algorithms for discovering and using general patterns in growing knowledge bases. *International Journal of Policy Analysis and Information Systems*, 4(3) :245–268.

Holland, J. H. (1985). Properties of the bucket brigade algorithm. Dans *International Conference on Genetic Algorithms and Their Applications*, pages 1–7, Pittsburgh, PA.

Holland, J. H. (1986). Escaping brittleness : The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. Dans Michalski, R. S., Carbonell, J. G., et Mitchell, T. M., éditeurs, *Machine Learning : An Artificial Intelligence Approach*, volume 2, pages 593–623. Kaufmann, Los Altos, CA.

Houssaye, J. (1988). *Théorie et pratiques de l'éducation scolaire I : Le triangle pédagogique*. Peter Lang, Paris.

Howard, R. A. (1960b). *Dynamic Programming and Markov Process*. MIT Press, Cambridge, MA, 7^{th} édition.

IMS (2003). *IMS Learning Design Information Model*. IMS Global Learning Consortium, Inc. Disponible sur http ://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html (accédé le 12/09/05).

Johnson, D. (1998). Virtual environments in army aviation training. Dans 8^{th} *Annual Training Technology Technical Group Meeting*, pages 47–63.

Johnson, D. M. et Wightman, D. C. (1995). Using virtual environments for terrain familiarization. Rapport technique ARI Research Report 1668, U.S. Army Research Institute for the Behavioral and Social Sciences, Alexandria.

Kermarrec, G. (2002). *Stratégies d'apprentissage et autorégulation en EPS, une recherche descriptive en contexte scolaire*. Thèse de doctorat, Université de Rennes 2.

Kermarrec, G., Todorovitch, J., et Fleming, D. (2004). Investigation of the self - regulation components students employ in the physical education setting. *Journal of Teaching in Physical Education*, pages 71–95.

Klopf, A. H. (1972). Brain function and adaptive systems - a heterostativ theory. Rapport technique AFCRL72 -0164, Air Force Cambridge Research Laboratories.

Kodratoff, Y. et Michalski, R. S., éditeurs (1990). *Machine Learning : An Artificial Intelligence Approach*, volume 3. Kaufmann, San Mateo, CA.

Kovacs, T. (1996). Evolving Optimal Populations with XCS Classifier Systems. Habilitation à diriger les recherches, School of Computer Science, University of Birmingham, Birmingham, U.K.

Kovacs, T. (1997). XCS Classifier System Reliably Evolves Accurate, Complete, and Minimal Representations for Boolean Functions. Rapport technique CSRP-97-19, School of Computer Science, University of Birmingham, Birmingham, U.K.

Kowalski, R. (1974). Predicate logic as a programming language. Dans *IFIP'74*, pages 569–574. Reprinted in Computers for Artificial Intelligence Applications, (Wah, B. and Li, G.J., éditeurs), IEEE Computer Society Press, Los Angeles, 1986, pages 68-73.

Kozak, J. J., Hancok, P. A., Arthur, E. J., et Chrysler, S. T. (1993). Transfer of training from virtual relality. *Ergonomics*, 36(7) :777–784.

Lanzi, P. L. (1997). A Study of the Generalization Capabilities of XCS. Dans Bäck, T., éditeur, *the Seventh International Conference on Genetic Algorithms*, pages 418–425, San Francisco, CA. Morgan Kaufmann.

Lanzi, P. L. et Colombetti, M. (1999). An extension to the XCS classifier system for stochastic environments. Dans Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., et Smith, R. E., éditeurs, *Genetic and Evolutionary Computation Conference (GECCO'99)*, pages 353–360, Orlando, Florida, USA. Morgan Kaufmann.

Le Bodic, L. (2005). *Approche de l'évaluation des systèmes interactifs multimodaux par simulation comportementale située.* Thèse de doctorat, Université de Bretagne Occidentale, Brest, France.

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

Lesgold, A. (1988). *Learning Issues for Intelligent Tutoring Systems*, chapitre Toward a theory of curriculum for use in designing intelligent instructional systems, pages 114–137. Springer-Verlag New York, Inc., New York, NY, USA.

Lester, J., Voerman, J., Towns, S., et Callaway, C. (1997). Cosmo : A life-like animated pedagogical agent with deictic believability. Dans *Working Notes of the IJCAI Workshop on Animated Interface Agents : Making Them Intelligent*, pages 61–69, Nagoya, Japan.

Levesque, P. (2003). Creation and use of 3d as-built models at EDF. Dans *FIG Working Week 2003.*

Loftin, R. B. et Kenney, P. (1995). The use of virtual environments for training the hubble space telescope flight team. Disponible sur http ://www.vmasc.odu.edu/vetl/html/Hubble/virtel.html (accédé le 12/09/05).

Lourdeaux, D. (2001). *Réalité virtuelle et formation : conception d'environnement virtuels pédagogiques.* Thèse de doctorat, Ecole des mines de Paris.

Maes, P. (1991). The Agent Network Architecture (ANA). *SIGART Bulletin*, 2(4) :115–120.

Major, N., Ainsworth, S., et Wood, D. (1997). REDEEM : Exploiting symbiosis between psychology and authoring environments. *International Journal of Artificial Intelligence in Education*, 8(3-4) :317–340.

Major, N. P. et Reichgelt, H. (1992). COCA : A shell for intelligent tutoring systems. Dans Frasson, C., Gauthier, G., et McCalla, G. I., éditeurs, *Intelligent Tutoring Systems*, pages 523–530, Berlin. Springer-Verlag.

Matteucci, M. (1999). Fuzzy learning classifier system : Issues and architecture. Rapport technique 99.71, Dipartimento di Elettronica e Informazione - Politecnico di Milano.

McCulloch, W. et Pitts, W. (1943a). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5 :115–133.

Mellet d'Huart, D. (2004). *De l'intention à l'attention, contribution à une démarche de conception d'environnements virtuels pour apprendre.* Thèse de doctorat, Ecole Normale Supérieure des Mines de Paris, Paris, France.

Mendelsohn, P. (1994). Le transfert des connaissances. Dans Meirieu, P. et Develay, M., éditeurs, *Colloque international sur le transfert des connaissances en formation initiale et continue*, Université Lyon II.

Meyer, B. (2000). *Conception et Programmation orientées objet.* Eyrolles, première édition. Collection Technologies objet. ISBN : 2-212-09111-7.

Miller, J. R. (1988). The role of human-computer interaction in intelligent tutoring systems. Dans Polson, M. C. et Richardson, J. J., éditeurs, *Foundations of Intelligent Tutoring Systems*, pages 143–189. Erlbaum, Hillsdale, NJ.

Minsky, M. (1975). A framework for representing knowledge. Dans Winston, P. H., éditeur, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York.

Murray, T. (1996). Special purpose ontologies and the representation of pedagogical knowledge. Dans *International. Conference for the Learning Sciences (ICLS'96)*, Charlottesville, VA. AACE.

Murray, T. (1999). Authoring intelligent tutoring systems : An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10 :98–129.

Newell, A. (1990). *Unified theories of cognition.* Harvard University Press, Cambridge, MA, USA.

Nguyen-Xuan, A. (1995). Les mécanismes cognitifs d'apprentissage. *Revue Française de Pédagogie*, (112) :57–67.

Nielsen, J. (1993). *Usability Engineering.* Academic Press Inc, Boston, MA.

Norman, D. A. (1980). Welves issues for cognitive science. *Cognitive Science*, 4 :1–32.

Ong, J. et Ramachandran, S. (2000). Intelligent tutoring systems : The what and the how. *On line magazine Learning Circuits.* Disponible sur http ://www.learningcircuits.org/2000/feb2000/ong.htm (accédé le 12/09/05).

Papert, S. (1981). *Jaillissement de l'esprit.* Flammarion.

Patrick, J. (1992). *Training : research and practice.* Academic Press, London.

Pavlov, I. P. (1927). *Conditionned Reflexes.* Oxford University Press, New York.

Payr, S. (2003). The virtual university's faculty : An overview of educational agents. *Applied Artificial Intelligence*, 17(1) :1–19.

Perlin, K. et Goldberg, A. (1996). Improv : a system for scripting interactive actors in virtual worlds. Dans *23rd annual conference on Computer graphics and interactive techniques (SIGGRAPH '96)*, pages 205–216, New York, NY, USA. ACM Press.

Perrenoud, P. (1996). *Enseigner, agir dans l'urgence, décider dans l'incertitude.* ESF.

Piaget, J. (1974). *La prise de conscience.* PUF, Paris.

Pomerol, J. et Brézillon, P. (2001). About some relationships between knowledge and context. Dans Akman, V., Bouquet, P., Thomason, R., et Young, R. A., éditeurs, *Modeling and Using Context : Third International and Interdisciplinary Conference, Context 2001*, pages 461–464, Berlin. Springer-Verlag.

Popovici, D. M., Gerval, J. P., Chevaillier, P., Tisseau, J., Serbanati, L. D., et Gueguen, P. (2004). Educative distributed virtual environments for children. *Journal of Distance Education Technologies*, 2(4) :18–40.

Py, D. (1998). Quelques méthodes d'intelligence artificielle pour la modèlisation de l'élève. *Sciences et Techniques Educatives*, 5(2) :123–140.

Querrec, R. (2002). *Les systèmes Multi-Agents pour les Environnements Virtuels de Formation. Application à la sécurité civile.* Thèse de doctorat, Université de Bretagne Occidentale.

Quillian, M. (1968). Semantic memory. Dans Minsky, M., éditeur, *Semantic Information Processing*, pages 227–270, Cambridge, Mass. MIT press.

Rasmussen, J. (1986). *Information Processing and Human and Machine Interaction : An approch to cognitive engineering.* Elsevier Science Publishers, New York, NY.

Regian, J. W., Shelbilske, W. L., et Monk, J. M. (1992). Virtual reality : An instructional medium for visual spatial tasks. *Journal of Communication.*, 42 :136–149.

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

Richard, J. F. (1986). The semantics of action : its processing as a function of the task. Rapport de recherche 542, INRIA.

Richard, J. F. (1990). *Les activités mentales : Comprendre, Raisonner, Trouver des solutions.* Armand Colin, Paris. ISBN 2-200-2187-0.

Richard, J. F. (1996). Attention, contrôle et gestion des ressources. Dans Mellice, D. et Vom Hole, A., éditeurs, *Attention et contrôle cognitif : mécanisme, développement des habiletés, pathologies*, pages 5–15. Publications de l'université de Rouen.

Richard, J. F. (1998). *Les bases des fonctionnements cognitifs*, chapitre Cours de psychologie 1 : Origines et bases. Dunod, Paris.

Richards, R. A. et Sheppard, S. D. (1996). A Learning Classifier System for Three-dimensional Shape Optimization. Dans Voigt, H. M., Ebeling, W., Rechenberg, I., et Schwefel, H. P., éditeurs, *4th International Conference on Parallel Problem Solving from Nature (PPSN'96)*, pages 1032–1042, London, UK. Springer-Verlag.

Rickel, J. et Johnson, W. L. (1999). Animated agents for procedural training in virtual reality : Perception, cognition, and motor control. *Applied Artificial Intelligence*, 13.

Robert, G. (2005). *MHiCS, une architecture de sélection de l'action Motivationnelle et Hiérarchique à Systèmes de Classeurs pour Personnages Non Joueurs adaptatifs.* Thèse de doctorat, LIP6 - AnimatLab, Université Pierre et Marie Curie.

Robert, G. et Guillot, A. (2003). MHiCS, a modular and hierarchical classifier systems architecture for bots. Dans Mehdi, Q., Gough, N., et Natkin, S., éditeurs, *4th International Conference on Intelligent Games and Simulation (GAME-ON'03)*, pages 140–144, London, United Kingdom.

Robert, G., Portier, P., et Guillot, A. (2002). Classifier systems as 'animat' architectures for action selection in MMORPG. Dans Medhi, Q., Gough, N., et Cavazza, M., éditeurs, *3rd International Conference on Intelligent Games and Simulation (GAME-ON'02)*, pages 121–125, London, United Kingdom. SCS.

Rose, F. D., Atree, E. A., Perslow, D. M., et Penn, P. R.and Ambihaipahan, N. (2000). Training in virtual environments : transfer to real word tasks and equivalence to real task training. *Ergonomics*, 43(4) :494 – 511.

Rosenblatt, J. K. et Payton, D. W. (1989). A fine-grained alternative to the subsumption architecture for mobile robot control. Dans *the IEEE International Conference on Neural Networks*, volume 2, pages 317–324, Washington, DC. IEEE Press.

Rouse, W. B. (1982). Models of human problem solving : Detection, diagnosis and compensation for system failures. *Automatica*, 19 :613–625.

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

Russell, D. M., Moran, T. P., et Jordan, D. S. (1998). *Intelligent Tutoring Systems. Lessons Learned*, chapitre The Instructional-Design Environment, pages 203–228. Lawrence Erlbaum.

Rust, J. (1996). *Handbook of Computational Economic*, volume 1, chapitre Numerical dynamic programming in economomics, pages 614–722. Elsevier, North-Holland.

Sanchez, S. (2004). *Mecanismes evolutionnistes pour la simulation comportementale d'acteurs virtuels.* Thèse de doctorat, Université des Sciences Sociales Toulouse I.

Sanza, C. (2001). *Evolution d'Entités Virtuelles Coopératives par Système de Classifieurs.* Thèse de doctorat, Université Paul Sabatier.

Schiffrin, R. M. et Schneider, W. (1977). Controlled and automatized human information processing : perceptual learning, automatic attending, and a general theory. *Psychological Review*, 84 :127–190.

Schulenburg, S. et Ross, P. (2001). Strength and money : An LCS approach to increasing returns. Dans *Third International Workshop on Advances in Learning Classifier Systems (IWLCS'00)*, pages 114–137, London, UK. Springer-Verlag.

Self, J. (1988). *Artificial intelligence tools in education*, chapitre Student models : what use are they ?, pages 73–85. Elsevier Science Publishers B.V.

Shaw, E., Johnson, W., et Ganeshan, R. (1999). Pedagogical agents on the web. Dans *Third Annual Conference on Autonomous Agents (AGENTS '99)*, pages 283–290, New York, NY, USA. ACM Press.

Shute, V. J. et Reigian, J. W. (1990). Rose garden promises of intelligent tutoring systems : Blossom or thorn ? Dans *Space operations and Research (SOAR) Symposium*, Albuquerque, New Mexico.

Simon, H. A. et Hayes, J. R. (1976). *Cognition and Instruction*, chapitre Understanding Complex Instructions, pages 271–285. Lawrence Erlbaum Associates, Hillsdale, NJ.

Skinner, B. F. (1938). *The behavior of organims.* Appleton-Century-Crofts, New York.

Skinner, B. F. (1958). Teaching machines. *Science*, 128 :969–977.

Skinner, B. F. (1974). *About behaviorism.* Knopf, New York.

Smith, R. E., Dike, B. A., Ravichandran, B., El-Fallah, A., et Mehra, R. K. (1999). The fighter aircraft LCS : A case of different LCS goals and techniques. Dans *Second International Workshop on Learning Classifier Systems (IWLCS'99)*, pages 282–289.

Smith, S. F. (1980). *A Learning System Based on Genetic Adaptive Algorithms.* Thèse de doctorat, University of Pittsburgh.

Sowa, J. F. (1984). *Conceptual Structures : Information Processing in Mind and Machine.* Addison-Wesley, Boston, MA, USA.

Stansfield, J., Carr, B., et Goldstein, I. P. (1976). Wumpus advisor 1 : a first implementation of a program that tutors logical and probabilistic reasoning skills. Rapport technique AIM-381, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Steib, D., Da Dalto, L., et Mellet d'Huart, D. (2005). Apprendre le geste du soudage avec Cs-Wave : l'expérimentation de l'Afpa. Dans *Laval Virtual, First International VR-Learning Seminar, virtual reality international conference (VRIC'05).*

Stevens, A., Collins, A., et Godin, S. E. (1982). Misconceptions in students' understanding. Dans Sleeman, D. H. et Brown, J. S., éditeurs, *Intelligent Tutoring Systems*, pages 13–49, London. Academic Press.

Stolzmann, W. (1998). Anticipatory classifier systems. Dans *Third Annual Genetic Programming Conference*, pages 658–664. Morgan Kaufmann.

Sutton, R. S. (1984). *Temporal Credit Assignement in Reinforcement Learning.* Thèse de doctorat, University of Massachusetts.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3 :9–44.

Sutton, R. S. (1991). Reinforcement learning architectures for animats. Dans Meyer, J.-A. et Wilson, W. S., éditeurs, *From Animals to Animats : First International Conference on Simulation of Adaptive Behavior (SAB'91)*, pages 288–296.

Sutton, R. S. (1992). Reinforcement learning architectures for animats. Dans Meyer, J., Roitblat, H., et Wilson, W. S., éditeurs, *From Animals to Animats 2. The second International Conference on Simulation of Adaptive Behavior*. MIT Press.

Takadama, K., Terano, T., et Shimohara, K. (2001). Learning classifier systems meet multiagent environments. Dans *Third International Workshop on Advances in Learning Classifier Systems (IWLCS '00)*, pages 192–212. Springer-Verlag.

Takadama, K., Terano, T., Shimohara, K., Hori, K., et Nakasuka, S. (1999). Can multiagents learn in organization ? analyzing organizational-learning oriented classifier systems. Dans *Agents Learning about, from and with other Agents*.

Tardif, J. (1999). *Le transfert des apprentissages*. Éditions Logiques.

Tchounikine, P., Baker, M., Balacheff, N., Baron, M., Derycke, A., Guin, M., Nicaud, J. F., et Rabardel, P. (2004). Platon-1 : quelques dimensions pour l'analyse des travaux de recherche en conception d'EIAH. Rapport technique, Rapport de l'action spécifique « fondement théorique méthodologique de la conception d'EIAH » département STIC du CNRS.

Thibault, G. (2002). EDF : 5 ans d'expérience de formation avec la réalité virtuelle. Communication présentée lors de la session « Réalité virtuelle et formation : Une autre approche du réel ». *Virtual reality international conference (VRIC'02)*.

Thorndike, E. L. (1932). *The Fundamentals of Learning*. Teachers College.

Tinbergen, N. (1951). *L'étude de l'instinct*. Payot.

Tisseau, J. (2001). Réalité virtuelle - autonomie in virtuo. Habilitation à diriger les recherches, Université de Rennes I.

Tisseau, J. et Harrouet, F. (2003). *Le traité de la réalité virtuelle*, chapitre Autonomie des entités virtuelles. Les Presses de l'Ecole des Mines, Paris, deuxième édition.

Tolman, E. et Honzik, C. (1930). Insight in rats. *University of California Publications in Psychology*, 4 :215–232.

Tomlinson, A. et Bull, L. (1999a). A Corporate XCS. Dans *Proceedings of International Workshop on Learning Classifier Systems*, pages 298–305.

Tomlinson, A. et Bull, L. (1999b). A zeroth level corporate classifier system. Dans *Second International Workshop on Learning Classifier Systems (IWLCS'99)*, pages 306–313.

Tulving, E. (1976). *Recall and Recognition*, chapitre Ecphoric processes in recall and recognition, pages 37–73. Wiley.

Turner, R. M. (1993). Context-sensitive reasoning for autonomous agents and cooperative distributed problem solving.

Tyrrell, T. (1993). *Computational Mechanisms for Action Selection*. Thèse de doctorat, University of Edinburgh, Centre for Cognitive Science.

Van de Panne, M. V. et Fiume, E. (1993). Sensor-actuator networks. Dans *SIGGRAPH'93 : Computer Graphics*, pages 335–342, Anaheim, CA.

Van Lehn, K. (1988). Student modeling. Dans Polson, M. C. et Richardson, J. J., éditeurs, *Foundations of Intelligent Tutoring Systems*, pages 55–78. Erlbaum, Hillsdale, NJ.

Van Lehn, K. et Brown, J. S. (1980). Planning nets : a representation for formalizing analogies and semantic models of procedural skills. *Aptitude, Learning and Instruction*, 2 :95–138.

Van Marcke, K. (1998). GTE : An epistemological approach to instructional modeling. *Instructional Science*, 26 :147–191.

Vergnaud, G. (1995). Dossier compétences : introduction. *Performances Humaines et Techniques.*, pages 7–12.

Vygotski, L. (1985). *Pensée et langage.* Editions Sociales.

Wall, A. E. (1986). A knowledge-based approach to motor skill acquisition. *Motor development in children : aspect of coordination and control*, pages 33–50.

Webber, C. (2003). *Modélisation informatique de l'apprenant. Une approche basée sur le modèle cK¢ et la théorie de l'émergence.* Thèse de doctorat, Université Joseph Fourier - Grenoble I, Grenoble, France.

Webber, C. et Pesty, S. (2002). Emergence de diagnostic par formation de coalitions - Application au diagnostic des conceptions d'un apprenant. Dans Muller, J. et Mathieu, P., éditeurs, *Actes des Journées Francophones pour l'Intelligence Artificielle Distribuée et les Systèmes Multi-Agents (JFIADSMA'02)*, pages 45–57. Hermes.

Webber, C., Pesty, S., et Balacheff, N. (2002). A multi-agent and emergent approach to learner modelling. Dans Van Harmelen, F., éditeur, *European Conference on Artificial Intelligence*, pages 98–102. IOS Press.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems.* Morgan Kaufmann, Los Altos, California.

Wilson, P. N. (1999a). Active exploration of a virtual environment does not promote orientation or memory for encountered objects. *Environment and Behavior*, 31(6) :752–763.

Wilson, S. W. (1987). Hierarchical credit allocation in a classifier system. Dans *10^{th} International Joint Conferences on Artificial Intelligence (IJCAI'87)*, pages 217–220, Milan, Italy.

Wilson, W. S. (1994). ZCS : A zeroth level classifier system. *Evolutionary Computation*, 2(1) :1–18.

Wilson, W. S. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation*, 3(2) :149–175.

Wilson, W. S. (1996). Explore/exploit strategies in autonomy. Dans *From Animals to Animats 4 : the Fourth International Conference on Simulation of Adaptive Behavior (SAB'96)*, pages 325–332.

Wilson, W. S. (2000a). Get Real! XCS with continuous-valued inputs. Dans *Learning Classifier Systems, From Foundations to Applications*, pages 209–222.

Wilson, W. S. (2000b). State of XCS classifier system research. Dans *Learning Classifier Systems, From Foundations to Applications*, pages 63–82, London, UK. Springer-Verlag.

Winn, W. (1993). A conceptual basis for educational applications of virtual reality. Rapport technique 93-9, Washington Technology Center, University of Washington.

Winn, W. (2002). What can students learn in artificial environments that they cannot learn in class ? Dans *First International Symposium Open Education*, Anadolu Unversity, Turkey.

Witmer, B. G., Bailey, J. H., Knerr, B. W., et Parsons, K. C. (1996). Virtual spaces and real world places : transfer of route knowledge. *International Journal of Human Computer Studies.*, 45(2) :413–428.

Woolf, B. P. (1992). Building knowledge based tutors. Dans Tomek, I., éditeur, *Computer Assisted Learning : the 4^{th} International Conference (ICCAL'92)*, pages 46–60. Springer, Berlin, Heidelberg.

Zampa, V. (2003). *Les outils dans l'enseignement : conception et expérimentation d'un prototype pour l'acquisition par expositions à des textes.* Thèse de doctorat, Université de Grenoble II. Sciences de l'éducation.

Zelter, D. (1992). Autonomy, interaction and presence. *Presence*, 1(1) :127–132.

Zorola, R. (1995). *L'évaluation des IHMs Multi-utilisateurs dans le Travail Coopératif.* Thèse de doctorat, Université de Toulouse 1.

Annexe A

Apprentissage artificiel

L'apprentissage artificiel (ou automatique) est la notion qui englobe toute méthode permettant de construire un modèle de la réalité à partir de données, soit en améliorant un modèle partiel ou moins général, soit en créant complètement le modèle. Cette annexe s'intéresse à la notion d'apprentissage artificiel, précise la distinction entre apprentissage numérique et apprentissage symbolique, et décrit les différents paradigmes.

A.1 Qu'est ce qu'apprendre ?

Apprendre, c'est chercher à acquérir un ensemble de connaissances par un travail intellectuel et/ou par expérience. Bien que la plupart des personnes soit d'accord sur ce concept, il existe plusieurs communautés qui abordent le problème de l'apprentissage selon différents angles et qui utilisent des termes différents. En effet, la communauté mathématique considère le problème de l'apprentissage comme un problème d'*optimisation*, *i.e.* la recherche d'un état considéré comme le plus favorable pour atteindre un but déterminé. D'un autre côté, la communauté IA parle d'*adaptation* ou d'*évolution*, *i.e.* le passage par une série de transformations afin de progresser. Dans tous les cas, même si les termes différent, l'objectif reste l'*amélioration* du système.

L'apprentissage est défini comme « la construction de nouvelles connaissances ou l'amélioration de connaissances déjà existantes » (Kodratoff et Michalski, 1990). Philippe Beaune nous présente un modèle général de l'apprentissage automatique (*cf.* figure A.1). Globalement, l'enseignant (ou l'environnement) agit sur le module d'apprentissage de l'élève. Cela a pour effet de consulter et de modifier la base de connaissances pour arriver à une exécution. Un retour est attendu de cette exécution pour savoir si le résultat est bien celui escompté.

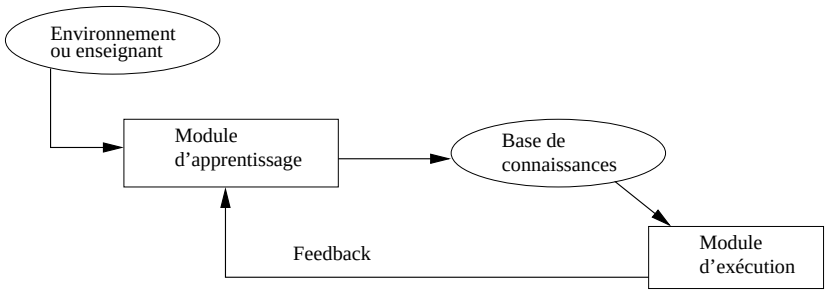


Figure A.1 – Représentation de l'apprentissage automatique.

A.2 Apprentissage numérique ou symbolique ?

Il existe deux tendances principales en apprentissage, celle issue de l'IA qualifiée de symbolique, et celle issue des statistiques qualifiée de numérique.

1. *Apprentissage numérique :*
   L'apprentissage numérique met en œuvre des données quantitatives, et peut ainsi permettre de résoudre des problèmes numériques complexes. Les méthodes d'apprentissage numérique consistent à établir des relations numériques entre les entrées (données des exemples à apprendre) et les sorties (comportement observé des exemples). Deux exemples qui utilisent des variables intermédiaires dont les valeurs peuvent fluctuer au cours de l'apprentissage, sont l'apprentissage des poids des connexions pour les réseaux de neurones, et des probabilités conditionnelles pour les réseaux bayésiens.

2. *Apprentissage symbolique :*
   Dans cet apprentissage, la manière de représenter les données qui s'intègrent dans une structure plus ou moins complexe, avec des entités définies (ensembles, listes, objets, n-uplets, *etc.*), est primordiale. Aussi, chaque type d'entité est associé à des relations d'appartenance et de liens spécifiques. L'apprentissage symbolique élabore des méthodes permettant d'extraire des connaissances structurelles ou décisionnelles à partir d'un ensemble d'instances peu structurées. Un exemple d'apprentissage symbolique est montré par les arbres de décision. Ils permettent de classer des objets ayant des attributs de nature discrète.

A.3 Paradigmes d'apprentissage artificiel

En apprentissage artificiel, le rôle de l'enseignant (le maître) est primordial. Trois grandes catégories d'approches peuvent être distinguées dans le domaine de l'apprentissage artificiel selon le type d'informations disponibles :

- ▷ l'apprentissage non-supervisé.
- ▷ l'apprentissage supervisé (*cf.* figure A.2).
- ▷ l'apprentissage renforcé (*cf.* figure A.3).

A.3.1 Apprentissage non-supervisé

En apprentissage *non-supervisé*, le système ne reçoit aucune information lui indiquant quelles devraient être ses sorties ou même si celles-ci sont correctes. Il doit donc découvrir par lui-même les corrélations existantes entre les patrons d'apprentissage. Pour permettre un tel apprentissage, il est nécessaire que les données contiennent un certain degré de redondance. Cette méthode propose de classifier des exemples.

Exemple : Construire un profil client. Imaginons un produit d'assurance. Une base de données contient des informations sur chacun des clients. L'apprentissage consiste à regrouper les ensembles d'exemples en classes. Le système est en mesure de prédire le risque d'accident de son client en utilisant sa classe. Le programme d'apprentissage n'a pas besoin d'un maître puisque l'expertise est présente dans les données de manière implicite, l'apprentissage est non-supervisé.

A.3.2 Apprentissage supervisé

En apprentissage *supervisé*, le maître fournit soit l'action qui devrait être exécutée, soit un gradient sur l'erreur commise. Dans les deux cas, le maître fournit au contrôleur une indication sur l'action qu'il devrait générer afin d'améliorer ses performances (*cf.* figure A.2). L'utilisation d'une telle approche présuppose l'existence d'un expert capable de fournir un ensemble d'exemples formés de situations et d'actions correctes associées. Ces exemples doivent être représentatifs de la tâche à accomplir.

Exemple : Reconnaissance de caractères manuscrits. L'apprentissage consiste à identifier les formes. Dans ce cas, on présente aux systèmes des échantillons manuscrits d'une lettre de l'alphabet connue. Il s'agit d'être capable de généraliser à partir d'exemples en connaissant la solution, l'apprentissage est supervisé.

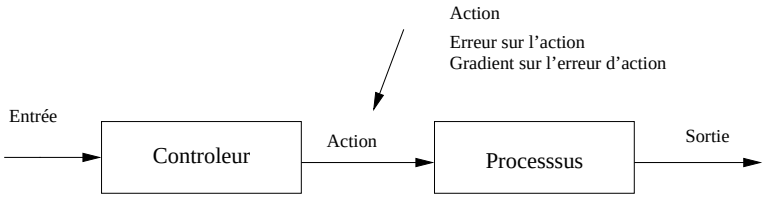


FIGURE A.2 – Représentation de l'apprentissage supervisé, d'après (Reignier, 1994).

A.3.3 Apprentissage renforcé

Par opposition, le maître en apprentissage *renforcé* a un rôle d'évaluateur et non pas d'instructeur. Il est, en général, appelé critique. Le rôle du critique est de fournir une mesure indiquant si l'action générée est appropriée ou non. Il laisse le contrôleur déterminer seul

comment il doit modifier ses actions de manière à obtenir une meilleure évaluation dans le futur (*cf.* figure A.3). Contrairement donc à l'apprentissage supervisé, le critique connaît les objectifs visés mais ne sait pas comment les atteindre. Dans le cas de l'apprentissage renforcé, le contrôleur doit découvrir seul l'action qu'il doit générer afin d'obtenir la sortie désirée.

Exemple : Contrôle d'un robot perdu dans un labyrinthe. Le robot obtient une récompense à chaque fois qu'il trouve la sortie. Aucun exemple n'est à fournir au système, il ne s'agit donc pas d'apprentissage supervisé. Le mécanisme apprend l'interaction entre un système et son environnement. Le système doit trouver l'action qu'il doit effectuer lorsqu'il est dans une situation donnée. L'apprentissage se base sur la métaphore de la carotte et du bâton. L'environnement peut "punir" ou "récompenser" le système en fonction des actions que celui-ci a fait.

Figure A.3 – Représentation de l'apprentissage renforcé, d'après (Reignier, 1994).

Une partie de la communauté considère que l'apprentissage par renforcement ne constitue pas un paradigme d'apprentissage et qu'il se classe parmi les apprentissages non-supervisés.

Annexe B

Les systèmes de classeurs, une présentation générale

Les systèmes de classeurs sont des architectures de contrôle et d'apprentissage non supervisé d'interactions entre un système artificiel et son environnement. Ils sont apparus pour pallier les problèmes de complexité rencontrés lors de la mise en œuvre d'algorithmes d'apprentissage par renforcement (Sutton, 1984). Rappelons que l'apprentissage par renforcement consiste à rechercher une action à effectuer selon l'état dans lequel se trouve un système afin de maximiser des récompenses (ou renforcement) distribuées lors de l'arrivée du système dans certains états. Une des difficultés de ce type d'apprentissage réside dans la représentation des états : soit on est capable de connaître chacun d'eux sans ambiguïté, ce qui implique des mécanismes de perception de l'environnement irréalistes et une exhaustivité augmentant considérablement les vitesses d'apprentissage ; soit on considère des mécanismes de perception restreints et il faut trouver un moyen de distinguer des situations différentes perçues de façon semblable (perception non markovienne). Les systèmes de classeurs proposent d'associer des algorithmes d'apprentissage par renforcement à des représentations généralisantes. Ils apprennent à la fois les actions à choisir en fonction des perceptions ainsi qu'un filtrage de ces perceptions permettant de définir le nombre d'états généralisés minimum, pertinents pour l'apprentissage.

Les systèmes de classeurs sont également des outils de prise de décision à part entière. Ils sont utilisés, par exemple, pour effectuer des diagnostics médicaux (Bonelli et al., 1990) ou élaborer des stratégies dans le combat aérien ce qui, entre autre, a permis de découvrir des manœuvres qui ont été testées en situation réelle par l'US AirForce (Smith et al., 1999). En économie, ils ont permis d'étudier des scénarios de négociation (achat/vente d'actions) et d'apprendre des stratégies économiques (Schulenburg et Ross, 2001). Mais c'est dans la problématique de l'apprentissage de comportements en situation autonome qu'ils sont les plus pertinents. Dans Girard et al. (2001), ils constituent le mécanisme décisionnel de vaisseaux spatiaux. Sanza (2001) les utilise pour contrôler la sélection d'actions d'entités virtuelles coopératives, appliquées dans un jeu de football virtuel où chacun des 22 acteurs possède un système de classeurs. Enfin, Robert et al. (2002) les implique dans la sélection d'action de personnages de jeu en ligne persistant et massivement multi-joueurs. Cette liste d'exemples, non exhaustive, permet d'évaluer le large champ d'application de ces systèmes. Elle explique

en partie les raisons pour lesquelles il existe une multitude de modèles et d'algorithmes au milieu desquels il n'est pas aisé de se retrouver. Cette annexe propose une vue d'ensemble au travers d'une présentation progressive identifiant les différents problèmes rencontrés par les différentes versions de systèmes de classeurs. L'annexe est articulée comme suit.

La section B.1 présente les principes utilisés dans les systèmes de classeurs, en proposant une formalisation de leur structure et de leur dynamique interne. Cette formalisation sert de base à l'introduction des mécanismes d'apprentissage. La section B.2 montre différentes versions de système de classeurs : le Zcs, le Xcs, les systèmes à anticipation et les systèmes hiérarchiques et hétérogènes. Enfin, la section B.3 dresse un bilan et propose un récapitulatif permettant d'aider au choix d'un modèle en fonction de l'analyse du problème à traiter.

B.1 Principes de base

Nous allons introduire les principes de base des systèmes de classeurs au travers des premiers modèles, proposés par Holland (1976). La première implémentation, appelée CS1, apparaît en 1978 (Holland et Reitman, 1978) : une situation ou « état perçu » est stockée dans une mémoire perceptive de taille limitée, assimilée à la « mémoire à court terme » des sciences cognitives (appelée liste des messages). Une base de règles (« condition-action ») constitue l'équivalent de la mémoire à long terme. A chaque cycle de fonctionnement, un mécanisme d'appariement entre les conditions des règles et les perceptions permet d'identifier la ou les actions pouvant être appliquées (action(s) stockée(s) dans la liste des messages). L'exemple support proposé porte sur l'exploration d'un labyrinthe au sein duquel se trouve de la nourriture qui rapporte des points. L'objectif est d'apprendre en ligne (ou par expérimentation) l'association de conditions et d'actions maximisant la récolte de points. Pour éviter l'explosion combinatoire du nombre de règles possibles, Holland utilise des algorithmes génétiques recherchant des règles généralisantes. Le système devient adaptatif grâce à l'ajout d'un paramètre préférentiel dynamique (souvent appelé force) à chacun des classeurs. Les forces des règles sont alors modifiées dynamiquement par le biais d'un apprentissage par renforcement. Les règles de force importante ont plus de chance d'être choisies par le système, tandis que les règles de faible force sont détruites par l'algorithme génétique.

La figure B.1 synthétise les interactions entre un système de classeur et l'environnement. Nous introduisons ici deux interfaces qui seront utilisées lors de notre formalisation : l'interface d'entrée, qui permet de représenter une perception de l'état de l'environnement, et celle de sortie qui permet d'exécuter l'action choisie et donc d'agir sur l'environnement. Elle montre également les flux d'information : l'opération d'appariement, l'élection d'action, le renforcement et l'adaptation par algorithmes évolutionnistes.

B.1.1 Formalisation

Dans cette partie, nous allons proposer une formalisation incrémentale et générique des systèmes de classeurs.

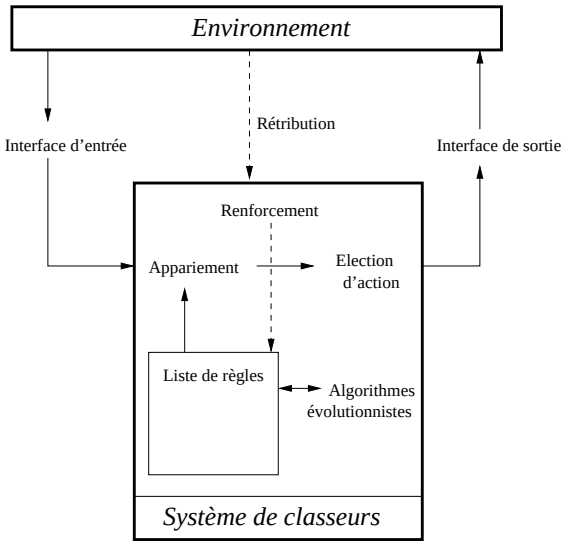


FIGURE B.1 – Interactions entre l'environnement et le système de classeurs.

B.1.1 - A Structure de base

La structure d'un système de classeurs, présentée en figure B.2, est constituée de différents ensembles jouant un rôle dans le choix de l'action à réaliser relative à une situation donnée. Formellement, un système de classeurs est un 7-uplet (`Ie`, `[P]`, `[M]`, `[A]`, `Comparaison`, `Sélection`, `Io`) :

- ▷ `Ie` est l'interface d'entrée, faisant correspondre à toute `Perception` de l'environnement un code binaire.
- ▷ `[P]`, appelé *population*, est l'ensemble des classeurs du système, codés par une succession de n bits[1]. Les représentations généralisantes contiennent des symboles # correspondant à une valeur indéterminée. Une règle est un couple (C, A) avec $C \cup A \in \{0, 1, \#\}^n$ avec :
  - C : la condition d'application de la règle.
  - A : la ou les actions associées à l'application de la règle.

  Prenons l'exemple d'un robot qui possède quatre capteurs tout ou rien, et une action. L'interface d'entrée transforme l'état des capteurs en une valeur binaire, et l'interface de sortie déclenche l'action en fonction de la valeur du bit d'action. Ainsi, une règle $\{011\#, 1\}$ signifie que la règle est applicable si le premier capteur est inactif et les deux suivants actifs. L'état du quatrième capteur n'a pas d'influence, et l'application de la règle déclenche l'action.
- ▷ `[M]` $\subseteq$ `[P]` est l'ensemble des classeurs dont la partie condition s'apparie avec les informations perçues de l'environnement pour un cycle de sélection. Il est appelé *Match-set*.
- ▷ `[A]` $\subseteq$ `[M]` est l'ensemble des classeurs représentant l'action sélectionnée. Il est appelé *Action-set*.
- ▷ `Comparaison` est le mécanisme permettant de passer de `[P]` à `[M]`. Il s'agit généralement

[1] Même si certains systèmes travaillent sur d'autres alphabets (Matteucci, 1999; Wilson, 2000a; Heguy et al., 2002).

d'une règle d'appariement entre `C` et l'information provenant de `Ie`. Cette règle sait interpréter les symboles de généralisation composant les conditions des classeurs.

- ▷ `Sélection` est le mécanisme permettant de passer de `[M]` à `[A]`. Il détermine, en fonction de critères spécifiques aux différentes versions de systèmes de classeurs, l'action choisie.
- ▷ `Io` est l'interface de sortie faisant correspondre à un code binaire l'activation d'`Action(s)`.

Chaque type de système de classeurs définit ses propres mécanismes de `Comparaison` et de `Sélection`.

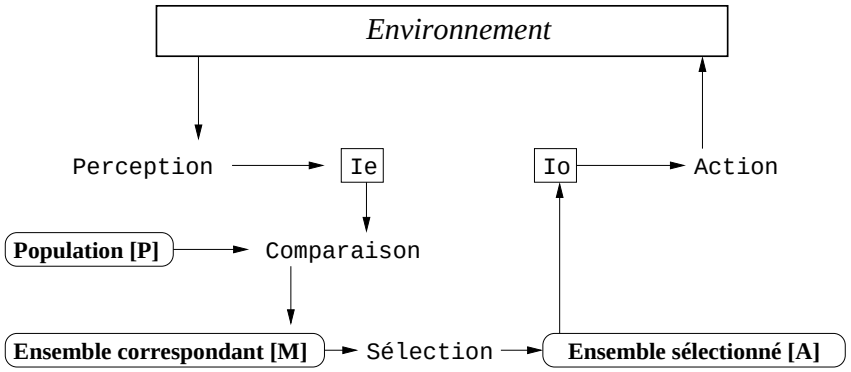


Figure B.2 – Structure et dynamique d'un système de classeurs.

B.1.1 - B Dynamique d'un système de classeurs

Les différents mécanismes impliqués dans la dynamique d'un système de classeurs, s'enchaînent cycliquement selon l'ordre `Perception` / `Comparaison` / `Sélection` / `Action`. Ce cycle est répété et peut être soumis à un critère de terminaison (temps maximal de résolution du problème par exemple). Un pseudo code équivalent est défini en figure B.3, où la variable t, représentant le numéro de l'itération courante, est introduite pour faciliter l'abord de l'apprentissage.

```
t := 0 ;                                  // compteur d'itérations
initClassifierPopulation( P(t) ) ;        // population de classeurs initiale
while not done do                         // critère de terminaison (temps ...)
  t := t +1 ;
  // Perception : 1. codage de la perception
  Ie(t) := readDetectors(t) ;
  // Comparaison : 2. comparaison de [P] et Ie et sauvegarde les appariements dans [M]
  M(t) := matchClassifiers( Ie(t),P(t),Comparaison ) ;
  // Sélection : 3. sélection des règles dans [A]
  A(t) := selectClassifiers( M(t),Sélection ) ;
  // Actions : 4. envoie de l'action via Io
  Io(t) := sendEffectors( A(t) ) ;
```

Figure B.3 – Pseudo code représentant les quatre étapes du cycle d'un système de classeurs.

B.1.2 Introduction de l'apprentissage

L'apprentissage des systèmes de classeurs repose sur les algorithmes d'apprentissages par renforcement (Sutton, 1991). Cet apprentissage est élaboré par essai-erreur et se classe donc parmi les apprentissages dits « non supervisés » (Reignier, 1994), *i.e.* qui ne nécessitent ni une série d'exemples à apprendre comme dans les réseaux de neuromimétiques de type *Perceptron* (McCulloch et Pitts, 1943a) ni une élaboration d'une fonction de *« fitness »* indiquant précisément la *qualité* d'une solution, comme c'est le cas avec les algorithmes génétiques (Goldberg, 1989). Plus précisément, les systèmes de classeurs utilisent des algorithmes d'apprentissage par différence temporelle (Klopf, 1972; Sutton, 1984, 1988) qui unifient les théories de l'apprentissage par renforcement pour découvrir un modèle de l'environnement[2] et les résultats de la programmation dynamique qui optimisent une stratégie de contrôle d'un système interagissant avec un environnement « *dont on connaît le modèle* »[3] (Bellman, 1957a,b; Howard, 1960b), (*cf.* également (Rust, 1996) pour une synthèse de ces travaux). L'apprentissage par différence temporelle s'intéresse aux environnements « *dont le modèle est inconnu* ». L'objectif est de retrouver les qualités définies en programmation dynamique. L'algorithme « apprend » l'environnement par essai-erreur, au fur et à mesure de l'exploration, en considérant les récompenses fournies par l'environnement et de l'estimation faite de ces récompenses aux instants précédents, reprenant ainsi les considérations des psychologues. Il élabore le comportement à adopter qu'aurait trouvé un algorithme de programmation dynamique. Concrètement, une qualité initiale $Q(s,a)$ correspondant à un nombre *a priori* quelconque, est fixée pour chaque couple *état-action*. L'algorithme interagit avec l'environnement et reçoit parfois des récompenses. La qualité d'une action effectuée à un instant t va être corrigée par la valeur de celle de l'action exécutée à l'instant $t+1$ et du renforcement éventuel fourni par l'environnement. Cette technique est mise en œuvre au travers d'algorithmes tels que le Q-learning ou le Sarsa.

Les systèmes de classeurs peuvent être vus comme un algorithme d'apprentissage d'interaction par renforcement, incluant des mécanismes de généralisation et de mémorisation permettant de traiter des environnements de grande taille, non déterministes, dynamiques et perçus de façon non markovienne[4]. L'un des objectifs du mécanisme d'apprentissage est de

[2] La notion de renforcement a été introduite par les psychologues dans des expériences d'apprentissage animal (Skinner, 1938). Ces expériences, poursuivant les travaux de Pavlov (1927) sur le conditionnement, montraient que si dans une situation donnée, un animal recevait une récompense ou une punition dépendant de l'action qu'il entreprenait, il apprenait progressivement à choisir les actions apportant le maximum de récompenses. Il arrivait donc à associer situation et action par mémorisation et l'amélioration de ses performances était fonction de l'intensité des récompenses et de leur répétitivité. Les psychologues Rescola et Wagner ont proposés une équation relatant ces mécanismes. Tolman et Honzik (1930) nuancèrent l'effet de la récompense en montrant que les animaux mémorisaient leur environnement de façon latente en l'absence de récompense et pouvaient exploiter cette mémorisation à posteriori.

[3] En programmation dynamique, le modèle de l'environnement est probabiliste : une probabilité de gain est associée à chaque action en fonction de chaque état et une autre probabilité caractérise l'état résultant d'une action (exécutée à partir d'un état donné). Le calcul de la commande est basé sur la définition d'une qualité associée aux couples *état-action*. La définition de cette qualité dépend de la nature de l'optimisation (optimiser des gains moyens, sur le long terme, sur un horizon fini ou infini). Il est possible de prouver l'optimalité de la commande calculée. De plus, l'environnement étant entièrement défini, aucune simulation n'est nécessaire.

[4] La capacité de *perception de l'environnement* d'une entité se scinde selon deux types :

1. les informations extraites de la perception immédiate de l'entité fournissent toutes les informations nécessaires pour choisir la meilleure action dans toutes les situations, ou, en d'autres termes, chaque

favoriser l'enchaînement de classeurs. Concrètement, on fusionne les systèmes de classeurs et l'apprentissage par différence temporelle en assimilant un ensemble de couples *état-action* à un classeur. Si l'on préserve la notion de rétribution faite épisodiquement par l'environnement, il est possible d'associer à chaque classeur une fonction d'utilité similaire à celle de l'apprentissage par renforcement et d'obtenir des apprentissages d'enchaînement d'actions. L'utilisation du symbole # permet d'accéder à la généralisation. De même, la représentation des classeurs sous la forme d'une chaîne de taille a priori quelconque, permet d'y introduire la mémorisation des historiques. Reste à régler le problème de la définition des historiques et des généralisations pertinentes. En effet, l'objectif de réduction de l'explosion combinatoire par codage généralisant implique que tous les classeurs *possibles* ne sont pas représentés. Il faut donc des mécanismes de `Sélection` et de `Génération` de classeurs. Ces mécanismes n'existent pas en apprentissage par différence temporelle et peuvent être élaborés en fonction de différentes hypothèses concernant la notion de qualité d'un classeur. Enfin, un classeur représente une multitude de couples *état-action*, plusieurs classeurs vont donc désigner une action identique. De façon réciproque, un état possible de l'environnement s'appariera avec plusieurs classeurs. Par conséquent, la chaîne de rétribution n'est plus linéaire comme en apprentissage par différence temporelle, mais arborescente. La technique de distribution des `Rétributions` la plus connue est l'algorithme du « *Bucket Brigade* », mais nous allons présenter d'autres variantes. L'introduction de la `Rétribution` et de la `Génération` modifie le cycle générique de fonctionnement des systèmes de classeurs qui est alors celui représenté en figure B.4.

B.1.3 Principaux paramètres et mécanismes

B.1.3 - A Paramètres des classeurs

Pour chaque classeur, des paramètres permettant de définir la qualité de la règle sont définis. Ils permettent de préciser les mécanismes de `Sélection`, de `Génération` et de `Rétribution`. On peut, par exemple, enrichir une règle avec une valeur de *Force* f identique à la création de tous les classeurs et évoluant au cours de l'apprentissage. Ainsi $R = (C, A, f)$. Dans ce cas, la fonction d'utilité définie en Q-learning, peut être assimilée à la force des classeurs. La qualité d'une règle influence deux aspects du système :

- ▷ la `Sélection` des classeurs de `[M]` vers `[A]`, et par conséquent le comportement à court terme du système ;
- ▷ l'extraction des classeurs de `[P]` servant de bases aux algorithmes de `Générations` (« *covering* » ou algorithmes génétiques, *cf.* ci-après) et par conséquent le comportement à long terme du système.

perception correspond à un état distinct. Dans ce cas, l'environnement est *markovien* du point de vue de l'entité. Celle ci n'a pas besoin de mémoriser l'histoire du système pour connaître l'état de l'environnement à un instant donné, seule sa perception immédiate y suffit ;

2. les informations extraites de la perception immédiate de l'entité fournissent des informations partielles sur l'environnement. Dans ce cas, il est possible qu'il existe différentes situations qui apparaissent comme identiques pour l'entité, nécessitant des actions optimales différentes. Par conséquent, l'entité ne peut pas choisir la meilleure action en ne considérant que ses informations sensorielles immédiates. L'environnement est dit *non markovien* du point de vue de l'entité. Afin de pouvoir différencier ces états, celle-ci doit prendre en compte son historique et peut utiliser une mémoire explicite (registre interne) ou implicite (chaînage entre les décisions).

```
t := 0 ;                                  // compteur d'itérations
initClassifierPopulation( P(t) ) ;        // population de classeurs initiale
while not done do                         // critère de terminaison (temps ...)
   t := t +1 ;
   // Perception : 1. codage de la perception
   Ie(t) := readDetectors(t) ;
   // Comparaison : 2a. comparaison de [P] et Ie et sauvegarde les appariements dans [M]
   M(t) := matchClassifiers( Ie(t),P(t),Comparaison ) ;
   // Génération (1) Covering : 2b. créer des règles s'appariant avec Ie (si M vide ou criterion1)
   if ( M.size < criterion0 || criterion1 ) then
                                              M(t) := cover( Ie(t),P(t),Covering ) ;
   // Sélection : 3. sélection des règles dans [A]
   A(t) := selectClassifiers( M(t),Sélection ) ;
   // Actions : 4. envoie de l'action via Io
   Io(t) := sendEffectors( A(t) ) ;
   // Rétribution (1) : 5. réception de la rétribution de l'environnement
   r := receivePayoff(t) ;
   // Rétribution (2) : 6. distribution de la rétribution aux classeurs
   P(t) := distributeCredit( r,P(t),P(t-1),Rétribution ) ;
   // Génération (2) A.E. : 7. éventuellement (selon t) un Algorithme Evolutionniste est utilisé sur [P]
   if ( criterion2 ) then
                              P(t+1) := reviseRules( P(t),Algorithme_Evolutionniste ) ;
```

FIGURE B.4 – Pseudo code représentant les sept étapes du cycle d'un système de classeurs apprenant classique.

D'autres critères tels que la spécificité (proportionnelle au nombre de 0 ou de 1 que contient le classeur) ou la prédiction de paiement, sont utilisés.

B.1.3 - B Mécanisme de `Sélection`

Le mécanisme de `Sélection` permet d'effectuer le passage de l'ensemble de classeurs appariés `[M]` à l'ensemble `[A]`. Les classeurs de `[M]` sont regroupés par paquets, chacun étant constitué de règles dont la partie *Action* s'apparie avec les autres règles du même paquet. Une combinaison des paramètres des classeurs de chaque paquet est utilisée pour définir leur sélectivité. En phase d'exploitation, ce critère est utilisé directement pour sélectionner l'ensemble `[A]`. En phase d'exploration, un mécanisme de roue de la fortune[5] est généralement appliqué, ce qui signifie que chaque paquet a une probabilité proportionnelle à sa sélectivité d'être sélectionné. De plus, des mécanismes de `Rétribution` et de `Génération` sont appliqués.

B.1.3 - C Mécanisme de `Rétribution`

La rétribution, appelée « *credit assignment* », modifie les forces des classeurs existants dans la population `[P]`. La répartition de la rétribution de l'environnement favorise ou défavorise les règles selon leurs efficacités. Le problème de la rétribution revient à décider

[5] Le mécanisme de roue de la fortune est une métaphore où il faut imaginer une roulette de casino sur laquelle est placé chacun des éléments sélectionnables, la place accordée à chacun étant proportionnelle à sa sélectivité. Ensuite la bille est lancée et l'endroit où elle s'arrête indique l'élément sélectionné.

quelles sont les règles qui, à un temps t, ont été nécessaires et suffisantes pour atteindre le but à un temps $t+n$, sachant que différentes règles ont été actives à chaque pas. Dans la recherche de but dans un labyrinthe, on peut se demander quels mouvements précédents ont permis d'atteindre le but. La plupart des algorithmes modifient la force des classeurs.

B.1.3 - D Mécanisme de `Génération`

Un des objectifs des systèmes de classeurs apprenant est de limiter le nombre de règles en supprimant les moins efficaces, mais également en cherchant des meilleures. Les meilleures règles sont celles qui sont les plus généralisantes possibles tout en ayant une force maximale. Pour cela deux mécanismes de génération, appelée *« rule discovery »*, sont utilisés ; il s'agit du *« covering »* et des *algorithmes génétiques*.

- ▷ Le *« covering »* permet de créer des règles lorsqu'aucun classeur ne s'apparie à la perception de l'environnement. Cela signifie que la population `[P]` possède un nombre insuffisant de règles permettant de proposer une action relative à la situation. Le *« covering »* peut également créer des règles lorsque les qualités des classeurs appariés sont considérées insuffisantes. Dans les deux cas, de nouvelles règles sont créées avec une partie condition(s) adéquate(s) plus ou moins généralisante. Dans ce cas, la probabilité d'obtenir un # est un paramètre à fixer. La partie action est choisie aléatoirement et la *Force f* peut être la moyenne des forces de `[P]`. Imaginons par exemple que le message d'entrée provenant de `Ie` soit 0111. La population `[P]` ne possède pas de règle correspondante. Le *« covering »* crée une règle dont la condition peut être 0111 ou #111 ou 0##1. Cette méthode permet notamment d'initialiser le système avec une population `[P]` vide et évite la génération de règles ne correspondant à aucun état possible de l'environnement.
- ▷ Les *algorithmes génétiques* (Holland, 1975; Goldberg, 1989) sont utilisés comme `algorithme évolutionniste` pour générer de nouvelles règles à partir de celles existantes. Ils utilisent des opérations évolutionnistes telles que le croisement ou la mutation[6]. La sélection des règles, servant de base pour les opérations génétiques, est généralement effectuée par un mécanisme de roue de la fortune. Les algorithmes génétiques peuvent intervenir sur les règles se situant dans `[P]` ou dans `[A]` selon les versions de système de classeurs (Wilson, 1994, 1995). Ce mécanisme permet au système de s'adapter plus rapidement aux environnements dynamiques. La fréquence d'utilisation des algorithmes génétiques, par rapport au cycle d'un système de classeurs, est un facteur important pouvant influencer sensiblement les performances d'apprentissage. La sélection par algorithmes génétiques peut encore être raffinée. Deux types de systèmes de classeurs se distinguent : le type « Michigan » (Smith, 1980) et le type « Pittsburgh » (Holland et Reitman, 1978). La méthode « Michigan » applique les algorithmes génétiques en utilisant un unique système où chaque règle est un individu, alors que celle de « Pittsburgh » considère un individu comme une population entière. Dans ce dernier, l'opérateur de croisement mêle les ensembles de règles. Une valeur sélective est attribuée à chacune des populations. La méthode « Pittsburgh » permet alors de les mettre en compétition, néanmoins la gestion d'un aussi grand nombre de

[6] Le croisement divise deux règles en un point de croisement. Le croisement génère une nouvelle règle qui est le résultat de la partie gauche d'une des règles et de la partie droite de l'autre règle. La mutation consiste à modifier un ou plusieurs bits d'une règle pour générer une nouvelle règle.

règles ne convient pas à un apprentissage incrémental d'un environnement dynamique. Le mécanisme de suppression est simple puisqu'il s'agit d'éliminer les individus les « moins bons ». La qualité d'un individu reste sujette à débat. Elle dépend des types d'applications.

B.2 Différentes versions de systèmes de classeurs apprenant

La complexité des premiers systèmes, tels que le CS1, n'a pas donné de résultats concluants. En effet, le CS1 peut s'envoyer lui-même des messages à l'aide de la liste des messages, cela crée des cycles d'inférences internes ayant tendance à développer et à maintenir des règles parasites. De plus, d'autres problèmes ont été mis en évidence (prolifération de règles surgénéralisées, *etc.*). Contrairement à certains travaux qui essayent de corriger les déficiences de performance des systèmes originaux, Wilson choisit une approche qui consiste à les simplifier et propose le Zcs. Il réduit les systèmes originaux aux éléments essentiels (suppression de la liste des messages), tout en conservant la même architecture.

B.2.1 Zcs

Le Zcs (Zeroth level Classifier System) a été présenté par Wilson comme un système de classeurs de type « Michigan » (Wilson, 1994). Son fonctionnement est représenté en figure B.5.

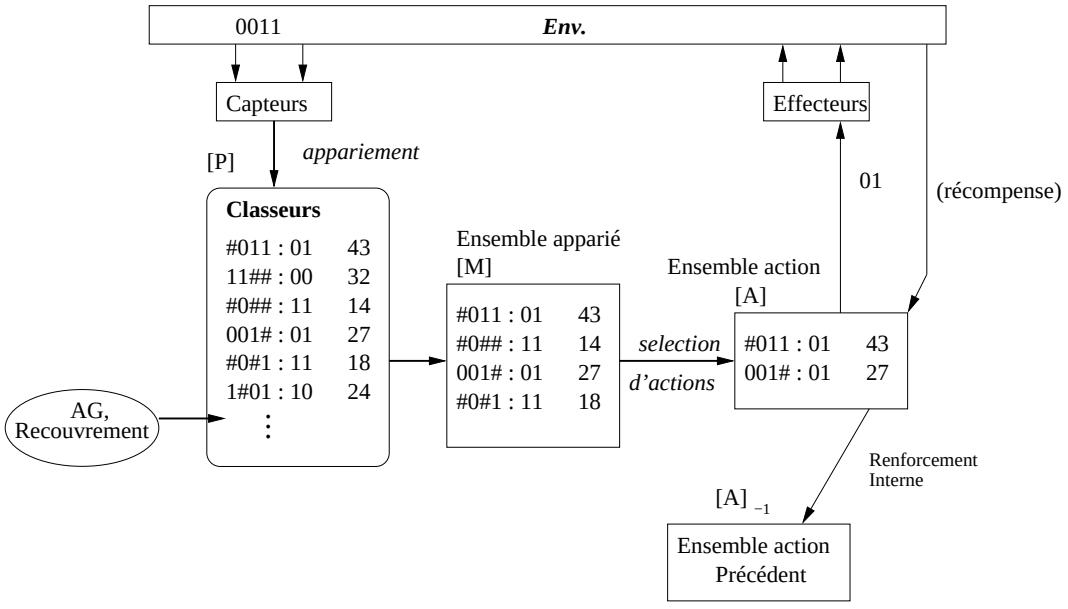


Figure B.5 – Système de classeurs de type Zcs d'après (Wilson, 1994).

Ce type de système de classeurs utilise des règles sous la forme classique du triplet $R = (C, A, f)$. Il précise les mécanismes suivants :

Sélection

La `Sélection` des règles relatives à une situation s'effectue en tenant compte de la *force* des classeurs de `[M]`. Lors de la phase d'exploitation, elle est généralement soumise au principe de *roue de la fortune*.

Rétribution

La `Rétribution` est soumise à un mécanisme proche du Q-Learning : le « *Bucket Brigade* ». La rétribution de l'environnement est l'origine de la chaîne de rétribution, qui passe par tous les classeurs qui ont participé aux déclenchements de l'action, et se termine par les classeurs qui sont à l'origine de toute l'action. L'algorithme (Holland, 1980, 1985) intègre un système économique dans la population de classeurs `[P]`. L'idée est d'assimiler les classeurs et l'environnement à des agents financiers. La force d'une règle peut alors être vue comme le capital de l'agent. Le mécanisme de marché met en place une compétition entre les agents concernés par la proposition de l'environnement pour avoir le droit de participer aux enchères. Pour chaque classeur de `[P]`, trois paramètres sont introduits : une *Enchère Cbid*, une *Taxe Ctaxe* et une *Rétribution Cr*. Ils sont initialisés avec une valeur nulle à chaque cycle. Nous pouvons considérer un classeur sous la forme $R = (C, A, f, s, Cbid, Ctaxe, Cr)$. Le mécanisme est constitué de trois étapes :

1. *La vente aux enchères* : chaque agent concerné donne une enchère pour participer. Elle consiste en un calcul d'une valeur *Cbid* pour les classeurs étant dans `[M]`. L'enchère proposée par chaque classeur est proportionnelle à sa *Force* f et à sa *Spécificité* s (donc de n). Elle est calculée en suivant : $Cbid = (r_{const} * s) * f$ avec $0 < r_{const} \leqslant 1$. La constante r_{const} peut être comparée au taux d'apprentissage d'un algorithme connexionniste. Par exemple, l'enchère est définie par $Cbid = k0 * (1 + 2 * n) * f$ avec $0 <$ k0 < 1 dans (Richards et Sheppard, 1996) ou par $Cbid = k0 * n' * f$ avec $0 <$ k0 < 1 et n' : % de # dans (De Boer et Cañamero, 1999).
2. *La compétition* : une compétition est mise en place pour déterminer les agents vainqueurs. La probabilité d'être victorieux est proportionnelle à l'enchère relative. Elle consiste à calculer la probabilité qu'un classeur de `[M]` puisse gagner (mécanisme de roue de la fortune). C'est le mécanisme de `Sélection`. Les classeurs non sélectionnés réinitialisent leur *Cbid* à 0.
3. *La répartition des rétributions* va modifier la force des classeurs de `[A]`, les transactions sont effacées et chaque agent :
   - ▷ reçoit une *rétribution de l'environnement* s'il est un des vainqueurs (les classeurs de `[A]`) : la rétribution *Cr* dépend de l'environnement fournissant r (r est en général divisé entre tous les classeurs) ;
   - ▷ doit *payer son enchère* s'il a été vainqueur (les classeurs de `[A]`) et reçoit une *rétribution des vainqueurs* s'il est à l'origine de la situation (précédent vainqueur : `[A]` au cycle précédent noté $[A]_{-1}$) : les classeurs de `[A]` vont payer leurs enchères et la somme sera distribuée aux classeurs à l'origine de la situation présente $[A]_{-1}$. Ce mécanisme permet de favoriser l'enchaînement de règles. Le mécanisme fonctionne comme une économie où des entreprises en compétition rémunèrent leurs sous-traitants, s'enrichissant ou s'appauvrissant selon que le contrat soit remporté ou non.

   La mise à jour de la force du classeur peut être soumise à un mécanisme de taxe. Celui-ci diminue la force des classeurs qui ne sont jamais choisis afin d'éliminer les règles qui ne

sont jamais invoquées. *Ctaxe* est prélevée proportionnellement à la force des classeurs de `[P]`. En résumé, la *Force* f de chaque classeur de `[P]` est mise à jour à chaque cycle de fonctionnement, suivant la formule : $f(t) = f(t-1) - Cbid * f(t-1) - Ctaxe * f(t-1) + Cr$.

Dorigo et Bersini (1994) montrent l'équivalence entre le « *Bucket Brigade* » et le Q-Learning et soulignent le fait que le mécanisme de taxes produit le même effet sur les forces que le facteur d'amortissement mis en place dans le Q-Learning.

Génération

La `Génération` s'effectue à l'aide du *« covering »* ou d'algorithmes génétiques s'appliquant au niveau de la population `[P]` en considérant la force des classeurs comme valeur sélective. Ces derniers agissent à fréquence constante. L'opérateur de mutation utilise des « parents » issus de la méthode de la roue de la fortune sur `[P]`. Les nouveaux classeurs remplacent les règles les plus faibles, afin de maintenir une population constante. La suppression sélectionne des règles en utilisant la méthode de la roue de la fortune sur `[P]` en considérant la valeur inverse de la force. La force des nouvelles règles est initialisée par la valeur moyenne des forces des « parents ». Dans le cas du « covering », elle est initialisée par la valeur moyenne des forces sur `[P]`.

Évolutions

Certains systèmes proposent d'étendre les capacités du Zcs :

- ▷ le Zcsm (Mémoire) (Cliff et Ross, 1995) ajoute une mémoire temporaire au Zcs. Chacune des parties (condition et action) des règles est étendue avec une sous-chaîne. La partie condition incorpore des bits, appelés registres, et la partie action est étendue avec un effecteur interne capable de modifier ces registres. Cette méthode permet d'augmenter les capacités des Zcs à des environnements non markoviens ;
- ▷ le Zccs (Corporation) (Tomlinson et Bull, 1999b) définit des liens entre les règles. Chacune possède un lien vers la règle précédente et un lien vers la règle suivante. Une corporation est un ensemble de classeurs liés entre eux. La méthode de croisement utilise des règles sélectionnées de la corporation.

Limites

Comme le souligne Gérard (2002), le Zcs est sujet au problème de la maintenance des longues chaînes d'actions. En effet, le mécanisme d'enchère permet de rétribuer la chaîne des règles qui ont permis d'arriver à l'action proposée. Les classeurs appariés à des situations éloignées de toute récompense sont donc rétribués, mais d'une façon moindre aux classeurs qui sont en fin de chaîne menant à la rétribution. Le Zcs utilise la force des classeurs comme valeur sélective pour l'algorithme génétique. Les classeurs qui associent une action optimale à une condition peuvent avoir une force moindre que des classeurs associant une action non-optimale à des situations plus proches d'une source de rétribution et par conséquent ils peuvent être supprimés. Ce problème augmente avec la taille de l'environnement, qui a pour effet de modifier la taille moyenne de la chaîne d'actions permettant d'atteindre un but.

B.2.2 Xcs

Le Xcs (Wilson, 1995) est une version de système de classeurs basée sur le Zcs. Son fonctionnement est représenté en figure B.6. Il a été introduit pour pallier au problème de la maintenance des longues chaînes d'actions. Wilson propose de ne plus considérer la force d'un classeur comme sa valeur sélective. Le Xcs propose de remplacer la force d'une règle, définie initialement par les gains qu'elle permet d'espérer, par la précision de ses prédictions de gain.

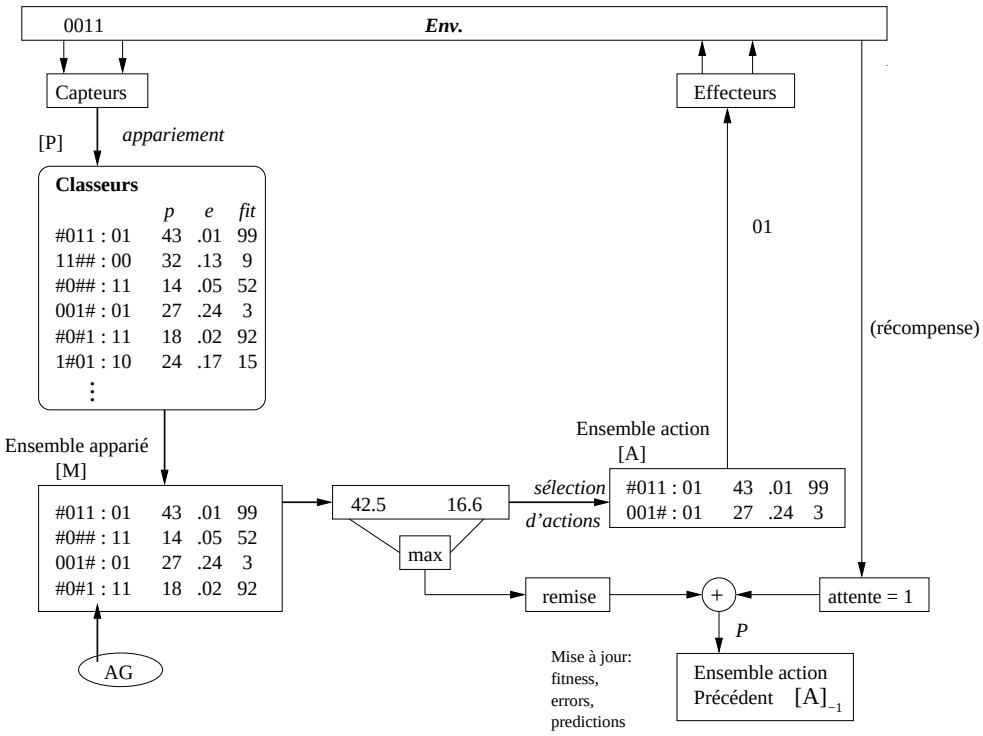


Figure B.6 – Système de classeurs de type Xcs d'après (Wilson, 1995).

Règles

Les règles sont composées de trois paramètres venant compléter les parties condition(s) et action(s). L'ancienne *Force f* du Zcs est ainsi décomposée :

- ▷ la *Prédiction de paiement p* représente la récompense attendue en effectuant l'action A dans les situations appariées par C,
- ▷ l'*Erreur sur la prédiction e* représente l'écart entre la *Prédiction de paiement p* et le paiement réel,
- ▷ la *« Fitness » fit* est la fonction inverse de e, elle représente donc l'exactitude de la *prédiction de paiement p*.

Les classeurs peuvent être définis par $R = (C, A, p, e, fit)$ avec C et $A \in \{0, 1, \#\}^n$ et $p, e, fit \in \Re$.

Sélection

La `Sélection` se base, non plus sur la *Force f* du classeur du ZCS, mais sur la précision de la *Prédiction de paiement p*. Cette méthode permet de sélectionner les classeurs n'ayant pas seulement une action optimale, mais plutôt ceux qui permettent de donner la qualité de l'action proposée avec précision. Elle nécessite un calcul des prédictions (*Prediction Array* `PA`). Pour chaque action a_i présente dans `[M]`, une prédiction $p(a_i)$ est calculée de la manière suivante : $p(a_i) = \sum p_i.(fit)_i / \sum (fit)_i$. Dans l'exemple de la figure B.6, quatre classeurs s'apparient à la situation de l'environnement et proposent les actions 01 et 11, la prédiction $p(01) = (43 * 99 + 27 * 3)/(99 + 3) = 42.5$ et $p(11) = (14 * 52 + 18 * 92)/(52 + 92) = 16.6$. Wilson (1996) propose de séparer explicitement les stratégies d'exploration et d'exploitation :

- ▷ pour l'exploitation, la sélection est déterministe et basée sur la plus haute prédiction ; les algorithmes génétiques sont inhibés ;
- ▷ pour l'exploration, la sélection est soumise à une probabilité permettant de sélectionner aléatoirement ou de sélectionner la plus haute prédiction.

Rétribution

La `Rétribution` n'est plus un « *Bucket Brigade* » suite à l'article de (Dorigo et Bersini, 1994). Elle est soumise à un mécanisme de « Q-Learning dérivé », la *Prédiction de paiement p* représente la fonction d'utilité $Q(s, a)$ du Q-Learning et la rétropropagation de la récompense utilise les équations de Bellman. Une fois la `Sélection` effectuée, l'ensemble des actions précédentes $\texttt{[A]}_{-1}$ (l'ensemble `[A]` lors du dernier cycle) est modifié en utilisant une combinaison de la dernière rétribution de l'environnement et la prédiction maximale de la `PA` actuelle (*cf.* figure B.6). La mise à jour des paramètres s'effectue de la manière suivante :

- ▷ $Cr = C_{-1} + \alpha * max(PA)$, α étant le taux de remise $\in [0, 1]$,
- ▷ $p = p + \beta * (Cr - p)$ avec β taux d'apprentissage $\in [0, 1]$,
- ▷ $e = e + \beta * (||Cr - p|| - e)$,
- ▷ *fit* peut s'obtenir[7] en trouvant la *Précision* :

  $k = \begin{cases} \gamma * [(e - e_0)/e_0] \; si \; e > e_0 \\ 1 \; sinon \end{cases}$ avec e_0 paramètre de seuil ;

  la *Précision relative* : $k' = k/(\sum^{[A]} k)$ d'où $fit = fit + \beta * (k' - fit)$. La *Fitness fit* augmente lorsque l'erreur de prédiction diminue.

Le système ne cherche plus à trouver des classeurs adaptés au problème, mais à trouver une approximation de la fonction d'utilité $Q(s, a)$ sans se limiter aux classeurs proposant une action optimale.

Génération

La `Génération` est effectuée à l'aide du *« covering »*. Il permet d'initialiser `[P]` avec un ensemble vide ou très réduit. Le processus d'algorithme génétique ne se situe plus au niveau de la population `[P]` mais au niveau de $\texttt{[A]}_{-1}$, et se base sur l'erreur de prédiction. La *Fitness fit* est la valeur sélective. Si l'erreur est grande alors la valeur sélective est faible et inversement. Le détail des opérations est présenté dans (Butz et Wilson, 2002). Butz et Pelikan (2001) ont

[7] Il existe de nombreuses possibilités de calcul de fitness dans diverses variantes successives de XCS.

montré que cette méthode favorise le maintien dans [P] des classeurs les plus généraux. Le classeur est conservé tant que son erreur est faible.

Le problème de la maintenance de la longueur de chaîne d'actions

Ce système résout le problème de la maintenance de la longueur de chaînes d'actions en ne cherchant pas directement à résoudre le problème de la maximisation de la récompense attendue. Le mécanisme sépare l'apprentissage de la sélection (*cf.* figure B.7). Il apprend un modèle de la fonction de récompense en utilisant l'*Erreur sur la prédiction e* et sélectionne les classeurs en utilisant la *Prédiction de paiement p*.

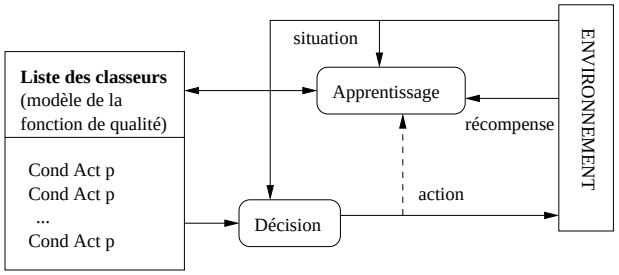


Figure B.7 – Architecture de Xcs d'après (Gérard, 2002).

Améliorations

Des modifications ont été proposées afin d'améliorer le système dans des cas précis :

- ▷ une classification des règles (Kovacs, 1997) selon la capacité de généralisation (nombre de #) : surgénéralisée, généralisée au maximum et généralisée sous optimale. Cette classification oriente la généralisation des règles ;
- ▷ le Pop-Xcs (optimisé) (Kovacs, 1996) propose d'étendre la généralisation au maximum. La méthode permet de converger vers une population réduite pour des problèmes markoviens ;
- ▷ le Xcss (spécifié) (Lanzi, 1997) est utilisé pour des environnements qui permettent peu de généralisation. Il introduit un nouvel opérateur « specify » qui permet de remplacer certains symboles # par des valeurs binaires et supprime les oscillations dues à une surgénéralisation ;
- ▷ le Xcsm (« messy ») (Lanzi et Colombetti, 1999) propose une modification de la partie condition des règles permettant d'avoir une longueur variable. Il supprime la correspondance entre les bits de la partie condition et ceux du message d'entrée. La condition est remplacée par une séquence de gènes en deux parties (un gène est constitué du type de capteur et d'une valeur binaire) ;
- ▷ le Xcsr (réel) (Wilson, 2000a) permet la prise en compte des données réelles. Il modifie l'interface d'entrée, la mutation, le « covering » et l'algorithme génétique ;
- ▷ le Cxcs (corporation) (Tomlinson et Bull, 1999a) introduit le principe de corporation dans les Xcs ;
- ▷ le Xcsts (« tournament selection ») (Butz et al., 2003) propose de répondre au problème de performance dû à la sélection proportionnelle des parents de l'algorithme génétique[8]. Les parents sont sélectionnés en considérant deux sous populations choisies

aléatoirement. Les vainqueurs sont ceux qui ont la plus grande *fitness*. Cette modification rend les Xcs plus efficace.

B.2.3 Systèmes de classeurs à anticipation

Le problème majeur des systèmes de classeurs vus jusqu'à présent, réside dans les temps de convergence de l'apprentissage. Il s'avère que l'exploration de l'environnement est sous-utilisée puisque c'est simplement lorsqu'une récompense est reçue qu'il y a un apprentissage. Or, durant cette exploration il est possible de faire un apprentissage latent[9] (*cf.* figure B.8). Le

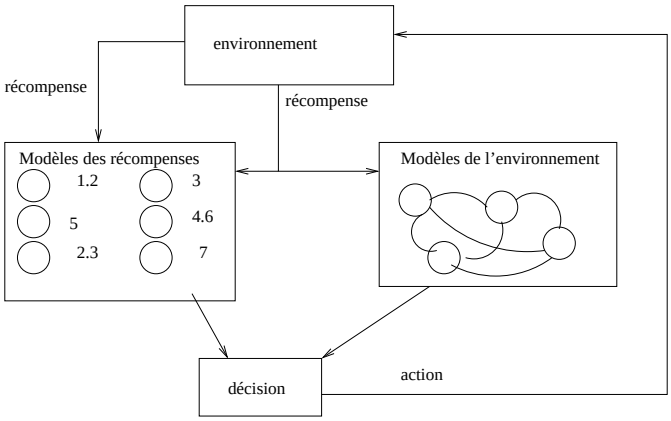


Figure B.8 – Apprentissage artificiel latent.

principe est de découvrir un modèle de l'environnement de façon découplée de l'apprentissage par renforcement et de rentabiliser ainsi les différentes explorations non exploitées en l'absence de renforcement avec des algorithmes tels que le Q-learning. Le modèle de l'environnement est une liste des probabilités de passage d'état à état (par le biais d'actions) apprise au fur et à mesure des évolutions du système par évaluation statistique. Les systèmes de classeurs à anticipation proposent une approche similaire, remplaçant l'énumération exhaustive des transitions entre états par une base de règles généralisantes, caractérisant les modifications apportées à l'environnement par les actions. Une telle généralisation permet de refléter en une seule règle des *régularités* de l'environnement. Par exemple, dans un environnement peuplé d'obstacles, une règle peut représenter le fait que, foncer sur l'un d'eux, ne permet pas de bouger. Les différentes versions de ce type de systèmes de classeurs se distinguent par les modèles de l'anticipation qui peuvent mettre en évidence différents types de régularités, ainsi que les règles de spécialisation et de généralisation qui en découlent (*cf.* figure B.9).

[8] Le lecteur trouvera le détail dans (Butz et al., 2003).

[9] L'apprentissage latent a été mis en évidence par Tolman et Honzik (1930). Il a montré que l'on pouvait apprendre sans motivation ou récompense et que les rats construisent, au cours de leurs déplacements, une *carte cognitive* du labyrinthe dans lequel ils se déplaçent. Pour l'apprentissage artificiel, c'est Sutton (1992) qui fut le premier à proposer une amélioration de l'apprentissage par renforcement avec l'architecture `Dyna` et l'algorithme `DynaQ+`.

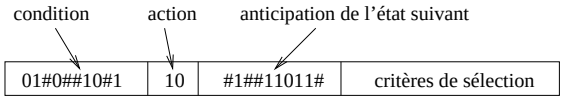


FIGURE B.9 – Modèle d'une règle d'un système de classeur anticipant.

B.2.3 - A ACS

Les ACS (Anticipatory Classifier System) (Stolzmann, 1998) sont destinés à une utilisation dans une architecture de type *Dyna*. Pour introduire l'apprentissage latent, une partie *effet* est ajoutée à chaque règle ainsi qu'une *qualité d'anticipation* notée `fa`. C'est cette qualité d'anticipation qui sera utilisée pour sélectionner les règles (conjointement à la prévision de renforcement). Un effet est une représentation similaire à celle de la partie condition mais reflétant l'état espéré du système suite à l'application de cette règle. Pour cette partie effet, une sémantique différente est associée au symbole # : il représente l'absence d'un changement d'attribut et, par conséquent, n'est pas à proprement parlé, un symbole de généralisation. La figure B.10 représente une règle d'un tel classeur et montre les états anticipés par celle-ci.

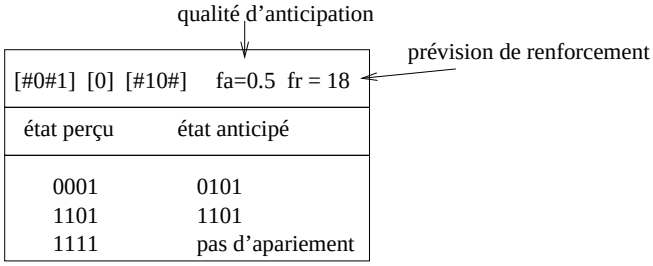


FIGURE B.10 – Fonctionnement d'une règle dans un ACS.

Pour accélérer le processus d'apprentissage, l'ACS utilise des heuristiques de spécialisation plutôt que les algorithmes génétiques. Ces heuristiques (issues des théories de Hoffman sur l'apprentissage latent dans le contrôle du comportement animal par anticipation), consistent à modifier une *qualité d'anticipation* d'une règle en fonction du résultat de l'application de l'action qui lui est associée et éventuellement à spécialiser celle-ci : si le résultat prédit par la règle n'est pas celui obtenu, sa *qualité d'anticipation* est diminuée, mais si l'erreur avait pu être évitée en spécialisant la règle, l'algorithme effectue cette spécialisation en remplaçant les # par les symboles adéquats. Cette spécialisation est cependant très brutale car elle concerne en bloc tous les attributs mal évalués à la fois dans la partie condition et dans la partie action. La modélisation proposée ne permet pas de distinguer les attributs pertinents à spécialiser. Pour pallier cette sur-généralisation, un algorithme génétique classique s'emploie à générer de nouveaux classeurs. Les classeurs qui anticipent mal sont supprimés.

B.2.3 - B YACS

le YACS (Yet Another Classifier System) (Gérard, 2002) utilise le même formalisme que le ACS avec des heuristiques différentes : il prend en compte un historique des erreurs de prédiction des classeurs ainsi qu'une mémorisation des valeurs des attributs lors du dernier

échec et du dernier succès de prédiction. Ceci permet d'élaborer des heuristiques permettant de modifier indépendamment les parties condition et action des règles et d'obtenir des spécialisations et des généralisations beaucoup plus fines, prenant en compte le *rôle* distinct de chaque attribut lors d'une erreur de prédiction. L'historique des erreurs de prédiction (sur un horizon fini) permet de détecter un classeur fiable que l'on préservera, d'un classeur oscillant, dont il faut spécialiser la partie condition. Enfin un classeur peu fiable, très souvent en échec, sera supprimé. La mémorisation des valeurs des attributs, lors des dernières situations d'échec et de succès, permet de définir une *estimation* de l'amélioration attendue en spécialisant chaque attribut généralisant (#) qui reflétera sa *chance* d'être modifié par généralisation : si un classeur a bien anticipé, on compare l'ancienne situation où il fut en échec de prédiction, à la situation qui vient de fonctionner : tous les attributs # qui permettent de distinguer ces deux situations, voient leur estimation augmenter (les autres la voient diminuer). A l'inverse, si l'anticipation n'a pas été correcte, on compare l'ancienne situation de succès à celle qui vient d'échouer et tous les attributs généralisant (#) qui distinguent ces situations, voient leur estimation augmenter. C'est l'*estimation* des attributs qui permet de choisir celui qui sera spécialisé. Notons également que, pour éviter des spécialisations abusives, un regroupement de classeurs similaires (possédant la même partie condition) est effectué, et que l'algorithme attend que ces classeurs soient tous oscillants pour les spécialiser. La généralisation est également effectuée à l'aide d'heuristiques similaires partant d'une estimation de l'amélioration attendue de la généralisation de chaque attribut spécialisé. Pour cela YACS s'intéresse aux classeurs qui n'ont pas été sélectionnés, à cause d'une partie condition trop spécialisée, alors qu'ils auraient fait une bonne anticipation (tout en ayant déclenché la bonne action). Comme pour la spécialisation, on attend d'avoir une bonne estimation de la fiabilité d'un classeur avant de le généraliser, et on opère par groupe de classeurs à effets similaires. Enfin, un mécanisme de couverture est introduit pour générer des règles s'appariant à une situation lorsqu'il n'en existe pas. YACS découple plus explicitement l'apprentissage par renforcement de l'apprentissage latent, puisqu'aucune force n'est associée aux règles : il y a une prévision de récompense apprise pour chaque état à l'aide d'un apprentissage par renforcement classique. Cette prévision est utilisée après les prédictions calculées par les règles pour choisir l'action à effectuer.

B.2.3 - C MACS

Le MACS (Modular Anticipatory Classifier System) (Gérard, 2002) est un formalisme pour l'apprentissage latent qui introduit les anticipations partielles : ses règles anticipent la valeur d'un seul attribut, pour les autres, on ne fait aucune hypothèse. Ce système offre de nouvelles possibilités de généralisation et d'apprentissage de régularités. En particulier, il permet d'apprendre des causalités entre attributs (la valeur de l'attribut n est répercutée sur celle de l'attribut $n+1$ à chaque fois que l'on fait une action a et quelque soit la valeur des autres attributs), très fréquentes dans le cas de perceptions partielles. La conséquence immédiate de cette approche est que plusieurs règles vont définir l'état, suivant l'exécution d'une action. Toutes les règles correspondant à l'exécution en cours (s'appariant avec l'état du système avant application de l'action), vont être évaluées : à chaque règle sont associées deux valeurs g et b comptant respectivement le nombre d'échecs et le nombre de succès de celle-ci. Ces nombres sont utilisés pour décider de la suppression de la règle. Un mécanisme similaire à YACS permet de définir les attributs devant être généralisés et ceux devant être spécialisés : on

spécialisera l'attribut d'un classeur oscillant discriminant au maximum les échecs des succès, et on généralisera l'attribut d'un classeur fiable ne discriminant pas les échecs des succès.

Au premier abord, toutes les améliorations effectuées sur ces systèmes améliorent les performances, mais il est toujours possible de trouver des contre-exemples. Par exemple, si l'environnement est stochastique, la plupart des systèmes ayant remplacés les algorithmes génétiques par des heuristiques sont pénalisant car ils peuvent tomber dans des minima locaux, dont le facteur aléatoire des algorithmes génériques permet de sortir.

B.2.4 Abstraction et hétérogénéité des données

Bien que l'utilisation des systèmes de classeurs ait produit des résultats intéressants, les modèles présentés précédemment souffrent d'un problème de granularité de représentation, lorsqu'il s'agit d'établir des connaissances mêlant des informations de portée sémantique différente (Barry, 1993). En effet, prenons l'exemple d'un système pour lequel certains attributs représentent la position d'une entité permettant de définir sa vitesse pour atteindre un but, et d'autres sa stratégie de défense à long terme face à une attaque. Cette hétérogénéité des données implique des niveaux de décisions différents avec éventuellement des échelles de temps différents. Si elle est *identifiable*, l'introduction de structures dans les systèmes de classeurs permet d'abstraire (tâches simples et tâches complexes), de décomposer (décomposer un problème complexe pour le résoudre plus facilement) et de réutiliser (solutions des sous-problèmes) les connaissances (Barry, 1996), pour améliorer leurs performances. C'est ce que proposent les systèmes hiérarchiques et les systèmes de classeurs hétérogènes.

B.2.4 - A Hgcs

Les Hgcs (Systèmes de Classeurs hétérogènes généralisés) (Sanchez, 2004) abordent l'hétérogénéité des données sous un autre angle : Il n'y a pas une hiérarchie de systèmes de classeurs, mais une classification des données en types distincts. Par exemple, il existe un type de données correspondant à un intervalle de valeurs possibles d'une entrée entière. La structure d'un Hgcs est donc très proche de celle d'un Zcs ou d'un Xcs (il existe d'ailleurs des Ghzcs et des Ghxsc), ce qui l'en distingue, c'est l'utilisation d'algorithmes génétiques hétérogènes pour sélectionner, généraliser ou spécialiser les règles. Ces algorithmes utilisent des opérateurs de mutation et de croisement, spécifiques à chaque ensemble de données. Ces opérateurs sont définis en fonction de la nature des données, de sorte que les *chromosomes* générés ne dépendent pas d'une loi aléatoire quelconque, mais introduisent un biais représentatif améliorant les performances du système.

B.2.4 - B Alecsys

Le Alecsys (A LEarning Classifier SYStem) (Dorigo, 1995) est une structure hiérarchique permettant de décomposer une tâche complexe en un ensemble de tâches simples. Un système de classeurs joue le rôle d'arbitre dans le choix de l'activation d'un des autres systèmes qui apprennent les comportements de bases. Le simulateur AutonoMouse permet d'évaluer le

modèle. Une souris doit suivre une source lumineuse mobile, et doit se rendre dans son terrier à l'apparition d'un prédateur. L'architecture est sur deux niveaux. A la base, deux systèmes de classeurs apprenant les deux comportements de base en parallèle : suivre la lumière et éviter le prédateur. Au sommet, un système de classeurs choisit le comportement qui va être activé.

B.2.4 - C OCS

Le OCS (Organizational Classifier System) (Takadama et al., 1999) est composé de différents agents possédant chacun un système de classeurs devant satisfaire une tâche commune. Les agents agissent collectivement pour résoudre un problème. Le mécanisme de spécialisation et la communication entre agents, permettent d'augmenter l'efficacité du système. Ce système a été appliqué à la conception de circuit électronique (Takadama et al., 2001). Chaque système de classeurs représente un composant électronique.

B.2.4 - D MHICS

Le MHICS (Modular HIerarchical Classifier Systems) (Robert et Guillot, 2003) est une architecture qui assemble des modules distribués sur plusieurs niveaux. Le premier niveau contrôle les motivations de l'entité (agressivité, faim, *etc.*). Il calcul une intensité pour chaque motivation en considérant une force relative à l'entité (PP) et une valeur de motivation (VM). La première correspond à la personnalité de l'entité, et la seconde évolue en fonction de la satisfaction obtenue préalablement. Un système de classeurs est associé à chaque motivation. L'objectif de chaque système est de satisfaire la motivation qui l'active. Les actions des règles des différents systèmes de classeurs du niveau I activent les classeurs du second niveau (pas dans l'implémentation actuelle) ou détermine directement des actions du niveau III (se déplacer vers la cible, tirer sur la cible, *etc.*). Le troisième niveau sélectionne les priorités entre les actions exécutables dépendantes des valeurs de motivations des niveaux I et II. Le niveau quatre fournit les ressources pour l'exécution des actions (angle, déplacement vers l'avant, *etc.*). L'action ayant la plus haute intensité (valeur provenant du niveau III) a la priorité pour utiliser les ressources. Cette architecture est utilisée pour contrôler des entités virtuelles pour des jeux en ligne massivement multi-utilisateurs.

B.3 Conclusion

Cette annexe a abordé une présentation globale des systèmes de classeurs. Nous avons pu voir, au travers des différentes versions existantes, les problèmes qui ont été rencontrés ainsi que les solutions proposées. Les systèmes que nous avons présentés permettent de trouver des solutions optimales dans des environnements markoviens ou non markoviens. Dans une étude récente, Sanchez effectue un bilan (*cf.* tableau B.1) sur les trois principaux systèmes (ZCS, XCS et ACS). Le lecteur pourra se rapporter à (Sanchez, 2004) pour des précisions sur les différences et correspondances entre les trois systèmes.

La nature heuristique des algorithmes sous-jacents aux systèmes de classeurs, implique

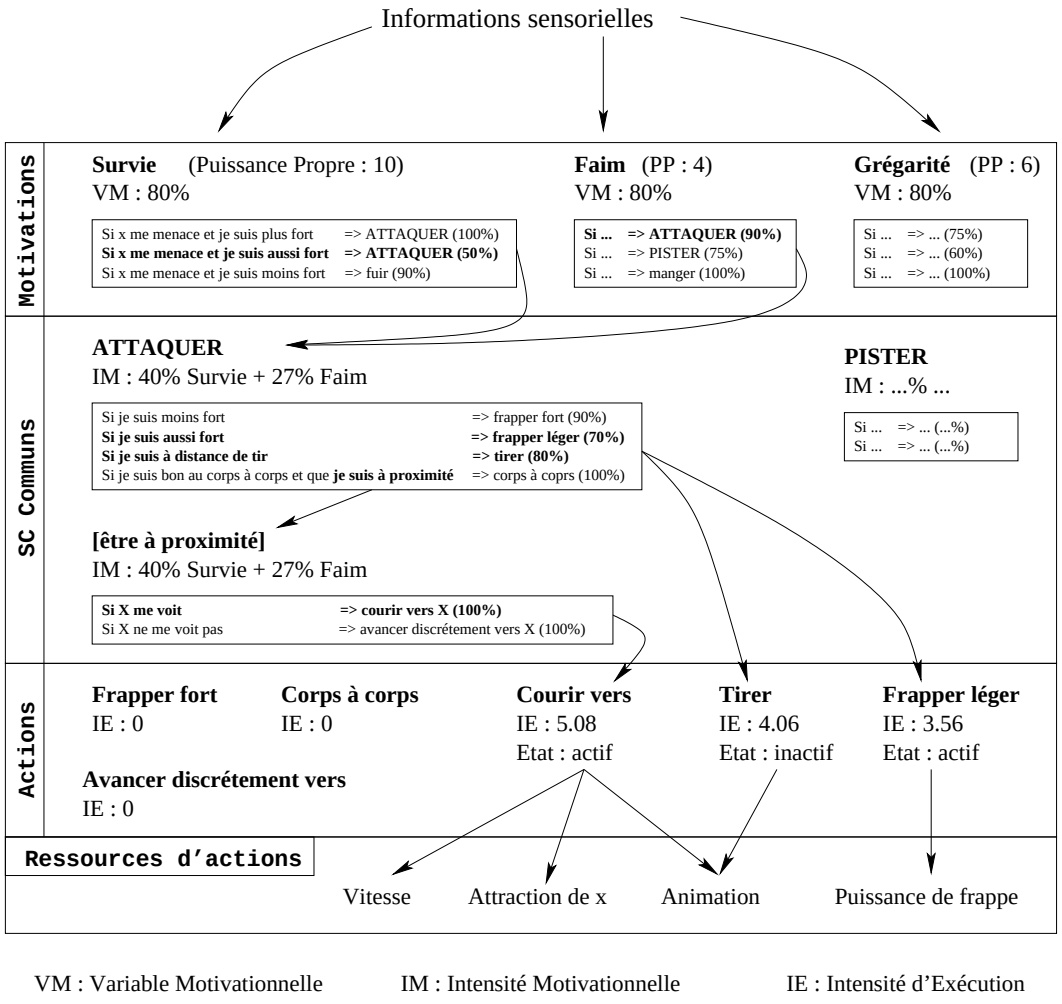


FIGURE B.11 – La sélection d'action à travers les niveaux dans MHICS.

qu'il n'y ait pas de système *idéal*, s'adaptant à tout problème. Comme le fait remarquer Sanza (2001) les systèmes sont adaptés à des cas précis, et aucun de ces modèles n'a vocation à apporter une solution globale pour tous les problèmes. Cependant, nous avons tenté de récapituler les considérations à effectuer pour guider le choix de l'un d'entre-eux sur le tableau B.2. Par exemple, il est difficile de définir si un MACS est adapté à des systèmes possédant des chaînes d'actions longues. Si la perception de l'environnement est markovienne, il faudra introduire des mémoires internes (XCSM au lieu de XCS), si les récompenses sont peu fréquentes, les chaînes d'actions sont donc longues et il serait préférable de privilégier un système dont la sélectivité est basée sur la précision de prédiction que sur la valeur des renforcements (XCS plutôt que ZCS). Le XCS n'est efficace que si les rétributions sont discrètes et en nombre fixe, le ACS n'est utile que si chaque action engendre une modification dans la perception du monde. A priori, les systèmes à anticipation apprennent plus rapidement en terme de nombre de pas de simulation à effectuer. Cependant, le temps de calcul de ces pas peut être très important, car, à chacun d'eux, on réévalue le modèle de l'environnement. Si l'on opte pour un apprentissage latent, l'étude des régularités induite par le couple (perception-environnement) orientera le choix. Enfin, l'hétérogénéité des données et la définition de niveaux d'abstraction de ces données feront opter pour une approche hiérarchique. Lorsqu'une case

contient un « *?* », c'est qu'il n'existe pas, à notre connaissance, de critères ou d'études permettant de conclure à l'adéquation du système de classeurs à la proposition faite.

Ces critères sont qualitatifs et constituent un guide et non une méthodologie. Cette étude appelle à une mise en œuvre formelle de critères de sélection de systèmes de classeurs. Cependant, celle-ci nous semble très difficile par le fait des liens entre ces critères potentiels : le choix ne peut se résoudre à un ensemble de règles (si-alors). L'écueil possible est alors d'obtenir des mécanismes de formalisation qui demandent un effort d'analyse dont on cherche justement à se passer lorsque l'on utilise des mécanismes d'apprentissage artificiel.

Systèmes / Caractéristiques	ZCS	XCS	ACS
Condition	✓	✓	✓
Action	✓	✓	✓
Conséquence			✓
Appariement	✓	✓	✓
Recouvrement	✓	✓	✓
Sélection de `[A]`	Fonction de la *force*	Fonction de la *fitness* et des *prédictions* des actions, utilisation d'un tableau de prédictions	Fonction des *prédictions* et des *qualités* des actions, utilisation d'un tableau de prédictions
Induction	AG sur `[P]`	AG sur `[A]`	AG sur `[A]`
Effacement	Fonction de la *force*	Fonction de la *fitness* et de l'*expérience*	Fonction de la *qualité* et de l'*expérience*
Renforcement	Mise à jour retardée de $[A]_{-1}$ en fonction de la récompense R_{-1} et du maximum des prédictions calculées pour les actions présentes dans `[M]`, taxation des règles non activées de `[M]`	Mise à jour retardée de $[A]_{-1}$ en fonction de la récompense R_{-1} et de la valeur maximale du tableau des prédictions	Mise à jour retardée de $[A]_{-1}$ en fonction de la récompense R_{-1} et de la valeur maximale du tableau des prédictions

Tableau B.1 – Caractéristiques des trois principaux systèmes de classeurs d'après (Sanchez, 2004).

Caractéristiques \ Systèmes		ZCS (ZCSM, ZCCS, *etc.*)	XCS (XCSM, CXCS, XCSTS, *etc.*)	ACS	YACS	MACS	Hierar-chiques	HGCS
Chaînes d'actions courtes (récompenses fréquentes)		oui	non	?	?	?	?	?
Chaînes d'actions longues (récompenses rares et stables)		non	oui	?	?	?	?	?
Chaque action provoque une modification des perceptions		moyen	moyen	oui	?	?	?	?
Problème markovien		ZCS	XCS	oui	?	?	?	?
Problème non markovien		ZCSM	XCSM	?	?	?	?	?
Problème stochastique = algorithmes génétiques		oui	oui	non	non	non	?	oui
Régularités liées aux attributs individuellement		?	?	non	oui	?	?	?
Régularités liées à une causalité entre attributs		?	?	non	non	oui	?	?
Données hétérogènes		non	non	non	non	non	non	oui
Données de niveaux sémantiques distincts		non	non	non	non	non	oui	non

TABLEAU B.2 – Comparaison entre les différents systèmes de classeurs.

Annexe C

Un modèle générique pour une famille de classeurs

Il existe de nombreux modèles de systèmes de classeurs. Partant de versions de bases, telles que les ZCS (Wilson, 1994) ou les XCS (Wilson, 1995) qui se distinguent par la nature des poids associés aux règles, des mémoires intermédiaires ont été introduites pour traiter les cas d'environnements non markoviens, comme dans le ZCSM (Cliff et Ross, 1995). D'autres encore intègrent la prédiction des états futurs (apprentissage latent) sans considérer la récompense, ACS (Stolzmann, 1998), YACS et MACS (Gérard, 2002). Le ZCCS (Tomlinson et Bull, 1999b) et le CXCS (Tomlinson et Bull, 1999a) utilisent le principe de corporation (ensemble de classeurs liés entre eux). Enfin, d'autres proposent d'optimiser des modèles existants, tels que le POP-XCS (Kovacs, 1996) , le XCSS (Lanzi, 1997) ou le XCSTS (Butz et al., 2003).

Les systèmes de classeurs sont donc multiples. Dans le cadre de nos travaux, il nous est nécessaire de pouvoir développer et tester facilement une grande variété de tels systèmes. L'analyse de la structure et de la dynamique effectuée en section B.1, nous permet d'aboutir à un modèle générique support pour une famille de systèmes de classeurs.

Ainsi, l'architecture que nous proposons, fait suite à l'analyse précédente sur la diversité des systèmes de classeurs. Notre architecture se veut générique, dans le sens où elle permet d'implémenter les systèmes de la famille des ZCS et XCS (ZCS, XCS, ZCSM, XCSM). Nous illustrons plus loin son utilisation par différentes expérimentations. Nous décrivons tout d'abord l'architecture ; nous montrons ensuite son utilisation.

C.1 Architecture

Notre architecture s'articule autour d'un assemblage de deux composantes (*interface avec l'environnement* et *système*), chacune étant détaillée par la suite. Elle est représentée en figure C.1 sous la forme d'un diagramme UML, auquel nous avons ajouté un système de classeurs de type ZCS par héritage.

C.1.1 Interface avec l'environnement

L'*interface avec l'environnement* détermine les interactions entre le système et l'environnement, communes aux différents systèmes de classeurs. Dans notre modèle, les différentes interfaces sont implémentées à l'aide de trois classes : `CS_II`, `CS_IO` et `CS_R` (respectivement interface d'entrée, interface de sortie et rétribution).

La communication entre les interfaces et l'environnement se fait sous la forme de messages, ce qui permet au système de classeurs d'avoir une exécution parallèle à celle de l'environnement (ce qui est nécessaire si on veut pouvoir l'utiliser autant dans des environnements simulés que réels).

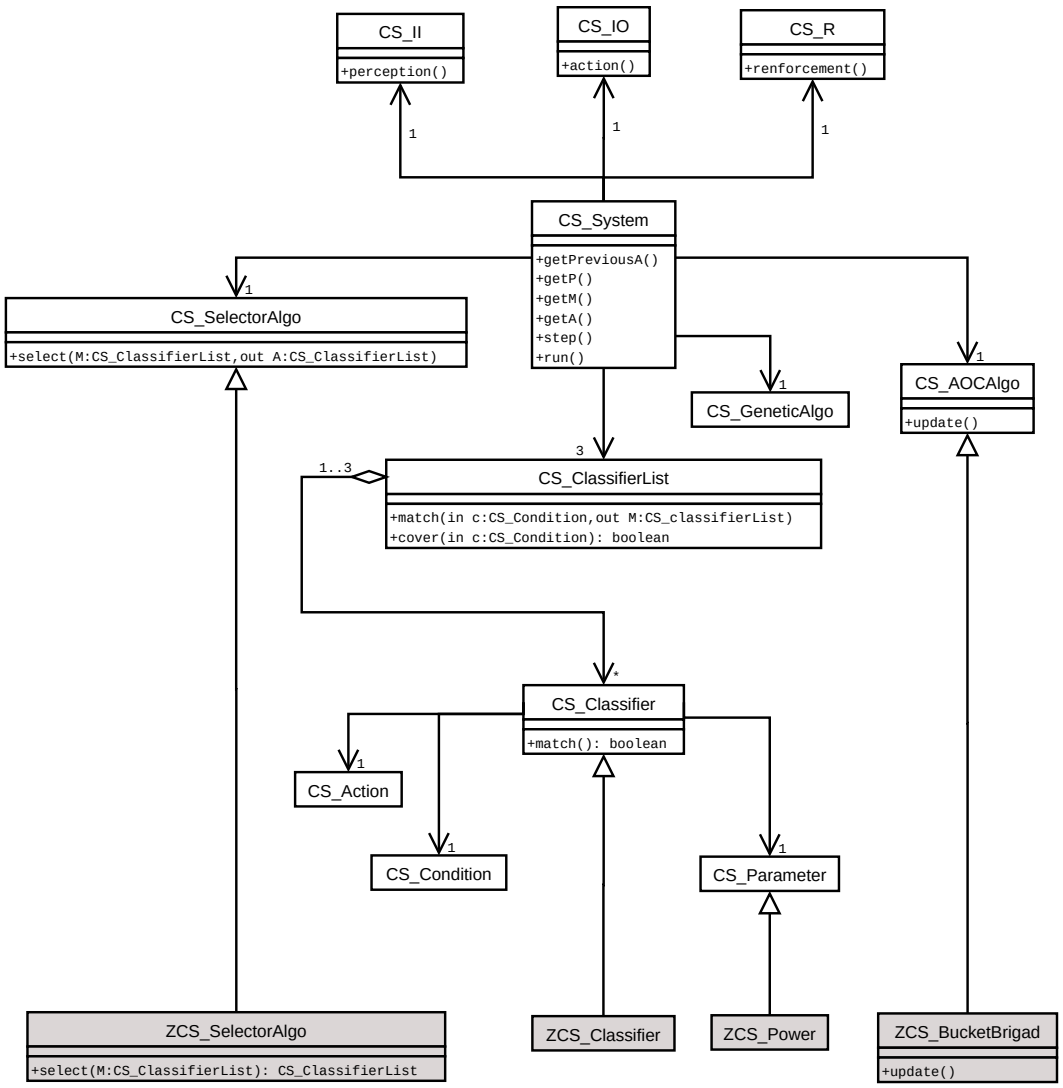


Figure C.1 – Diag. de classe UML de notre architecture, augmenté par un Zcs obtenu par héritage.

C.1.2 Système

La partie *système* définit, de façon réifiée, les éléments et les mécanismes de notre système de classeurs. Nous considérons les éléments suivants :

- ▷ un classeur (`CS_Classifier`) possède une partie condition (`CS_Condition`), une partie action (`CS_Action`) et une partie paramètre (`CS_Parameter`) ;
- ▷ les ensembles `[P]`, `[M]`, `[A]` et $[A]_{-1}$ sont des listes de classeurs, de type `CS_ClassifierList` ;

Nous proposons les mécanismes suivants :

- ▷ le mécanisme de `Comparaison`, qui permet d'extraire les classeurs dont la condition s'apparie aux informations provenant de l'environnement. Il est intégré dans `CS_ClassifierList` par la méthode `match()` ;
- ▷ le mécanisme de `Génération` *covering*, qui crée des règles suivant le contenu de `[M]` après `Comparaison`. Il est intégré dans `CS_ClassifierList` par la méthode `cover()` qui peut être paramétrée (nombre de # notamment) ;
- ▷ le mécanisme général (`CS_System`) représente le fonctionnement sur un cycle défini (méthode `step()`) par le pseudo code figure B.4 ;
- ▷ le mécanisme de `Sélection` des actions gagnantes (`CS_SelectorAlgo`) qui doit pouvoir être différent suivant l'apprentissage souhaité ;
- ▷ le mécanisme de `Rétribution` (`CS_AOCAlgo`) modifiant la partie paramètre des classeurs ;
- ▷ L'algorithme génétique de `Génération` (`CS_GeneticAlgo`), où différents opérateurs doivent être précisés tels que le croisement ou la mutation.

C.2 Utilisation

Nous pouvons spécialiser notre architecture[1] afin d'obtenir un ZCS (*cf.* figure C.1). Par héritage, nous définissons :

- ▷ les `Régles` (`ZCS_Classifier` fils de `CS_Classifier`) ayant un paramètre force (`ZCS_Power` fils de `CS_Parameter`) ;
- ▷ le mécanisme de `Sélection` de type roue de la fortune (`ZCS_RouletteWheel` fils de `CS_SelectorAlgo`) ;
- ▷ le mécanisme de `Rétribution` de type *Bucket Brigade* (`ZCS_BucketBrigad` fils de `CS_AOCAlgo`) ;
- ▷ l'algorithme génétique de `Génération` (`ZCS_GeneticAlgo` fils de `ZCS_GeneticAlgo`) spécifiant notamment que la valeur sélective est la force de la règle.

Le reste du système utilise les mécanismes par défaut déjà présents, tels que le *covering*. Nous avons mis en place le système de classeurs XCS en utilisant les mêmes techniques. Pour cela, il faut surdéfinir `CS_Parameter`.

Nous pouvons également très facilement ajouter de la mémoire au ZCS pour obtenir un ZCSM. Dans ce cas, il faut augmenter le résultat de la perception par un registre interne au

[1] L'implémentation de notre architecture se présente sous la forme d'une API *C++* basée sur ARÉVI (Harrouet et al., 2002), un atelier de réalité virtuelle fondé sur une approche multi-agents.

système de classeurs qui est modifié par une partie de la dernière action effectuée. Utilisant notre architecture, il suffit d'hériter de `CS_System` pour redéfinir la méthode `step()` (*cf.* figure C.2). Nous avons mis en place le système de classeurs Xcsm en utilisant les mêmes principes.

```
t := t +1 ;
// Perception : 1a. codage de la perception
Ie(t) := readDetectors(t) ;
// Perception : 1b. ajout du registre interne
Ie(t) := Ie(t) + Ir(t) ;
...
// Sélection : 3. sélection des règles dans [A]
A(t) := selectClassifiers( M(t),Sélection ) ;
...
// Mémoire : 8. Registre interne
Ir(t+1) := extractPart( getActionPart( A(t) ) ) ;
```

Figure C.2 – Pseudo code, représentant la modification de la figure B.4, qui met en place un registre interne afin d'ajouter de la mémoire à un système classique.

C.3 Validation

Un environnement d'évaluation simple et fréquemment utilisé est un multiplexeur. Considérons un multiplexeur avec pour entrée six bits $a_0, a_1, d_0, d_1, d_2, d_3$ et pour *sortie* un bit. a_0 et a_1 correspondent aux bits d'adresse. L'équation du multiplexeur est $sortie = \overline{a_0}.\overline{a_1}.d_0 + \overline{a_0}.a_1.d_1 + a_0.\overline{a_1}.d_2 + a_0.a_1.d_3$. La sortie sera la valeur de d_0, d_1, d_2 ou d_3 selon l'adresse. Le but est de retrouver cette fonction de multiplexage.

Utilisant notre architecture, il suffit de déterminer les détecteurs et les effecteurs s'interfaçant avec l'environnement, le renforcement à distribuer, et d'instancier un `CS_System` (*cf.* figure C.3). Les conditions et les actions des classeurs correspondent ici respectivement aux entrées et aux sorties du multiplexeur. Une récompense est fournie au système de classeurs lorsque la règle sélectionnée correspond à la fonction de multiplexage.

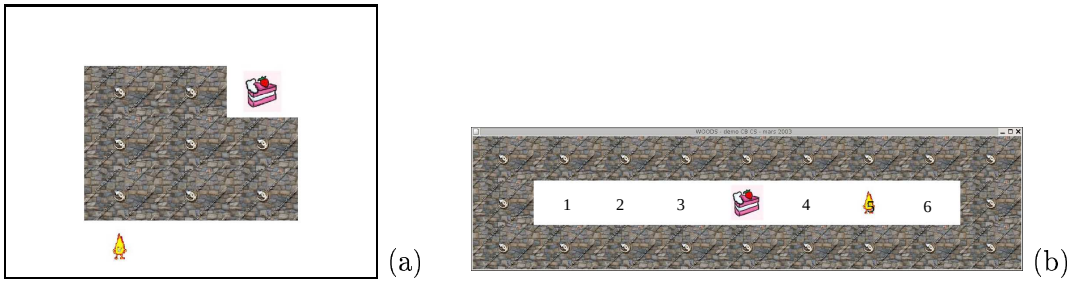
Un autre environnement d'évaluation intéressant est l'environnement *Woods*. Il correspond à une représentation graphique basée sur un motif qui est répété à l'infini. Il représente une forêt, où le but est de trouver de la nourriture. Dans cette forêt, il y a des obstacles infranchissables (arbres) et des cases de libre circulation. La perception de l'environnement correspond à une représentation des huit cases autour de la position occupée. Le plus simple de ces environnements est *Woods1* (*cf.* figure C.4 (a)). C'est un environnement déterministe markovien. Le *Woods100* (*cf.* figure C.4 (b)), quant à lui, est non markovien. En effet, le déplacement optimum de la case numéro deux est dirigé vers la droite alors qu'il est vers la gauche pour le numéro cinq, bien que ces deux cases sont perçues identiquement.

Le système apprend en alternant une itération en mode exploration (sélection de l'action par mécanisme de roue de la fortune) et une itération en mode exploitation (meilleure action choisie). Les courbes ne prennent en compte que les résultats en mode exploitation, en considérant la moyenne des dix dernières itérations.

Notre système converge pour le Multiplexeur (*cf.* figure C.5 (a)) et pour *Woods1* (*cf.* figure

```
/* Environnement */
     env = new EnvironmentMultiplexer(nbEntries,nbOut) ;
/* Relation avec l'environnement */
     detector = new CS_II_Boolean(env) ;
     effector = new CS_IO_Boolean(env) ;
     reinforcement = new CS_R(env) ;
/* Système */
     system = new CS_System() ;
     system->setDetector(detector) ;
     system->setEffector(effector) ;
     system->setReinforcement(reinforcement) ;
/* Activité */
     system->run() ;
```

FIGURE C.3 – Mise en œuvre de notre architecture générique pour un environnement de type multiplexeur.



(a) (b)

FIGURE C.4 – Environnements markovien Woods1 (a) et non markovien Woods100 (b).

C.5 (b)). Pour le multiplexeur, nous obtenons 100% de réussite. En ce qui concerne *Woods1*, nous obtenons des solutions proches du nombre minimum de déplacements. Les résultats sont concluants : dans les deux cas, nous obtenons des performances comparables aux résultats décrits par (Wilson, 1994).

Nous avons également comparé les résultats du ZCS et du ZCSM dans l'environnement non markovien *Woods100* (*cf.* figure C.6). Nos résultats mettent en évidence les difficultés du ZCS pour obtenir des règles optimales en environnement non markovien. Ils confirment également que notre architecture permet d'étendre les capacités du ZCS à des environnements non markovien.

C.4 Bilan

Dans cette annexe, nous avons décrit un modèle de système de classeurs qui recouvre les systèmes classiques. Nous en avons proposé une implémentation qui permet de modéliser et d'étendre les systèmes classiques ainsi que leurs variantes. Cette implémentation est assez souple pour être utilisée pour différents problèmes, puisqu'elle propose une interface entre l'environnement et le système de classeurs (entrées/sorties/renforcement). Elle a pu être

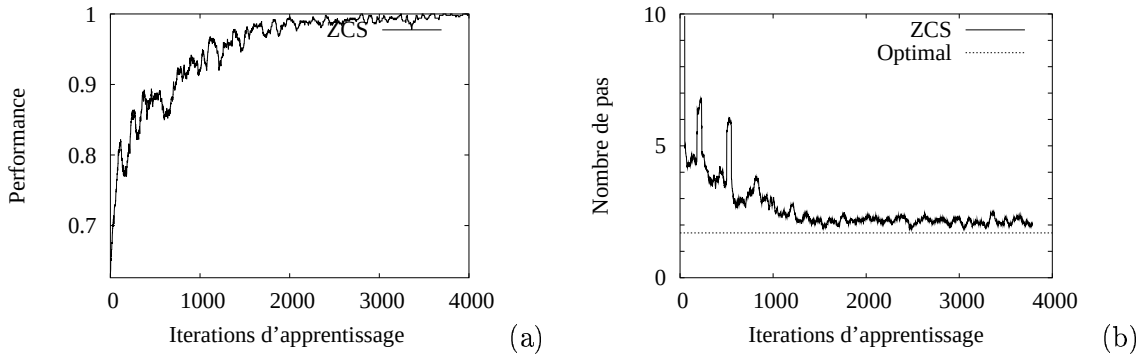


Figure C.5 – Apprentissage d'un multiplexeur (a) et dans Woods1 (b) d'un Zcs. A partir d'un nombre suffisant d'itérations, notre système réalise la fonction de multiplexage et obtient le minimum de déplacements dans le cas de Woods1.

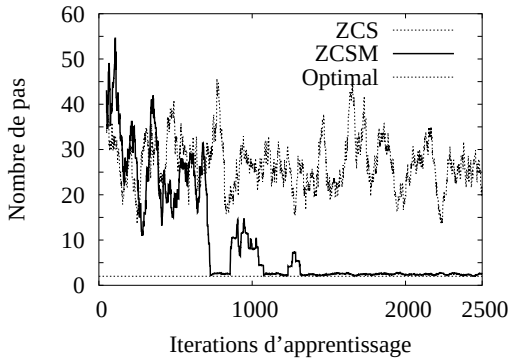


Figure C.6 – Apprentissage dans Woods100 d'un Zcs et d'un Zcsm.

utilisée sur différents problèmes et a permis de tester rapidement plusieurs types de systèmes de classeurs, différentes hypothèses conceptuelles, ainsi que d'obtenir des résultats comparatifs intéressants.

Une présentation du modèle proposé dans cette annexe est disponible dans l'article (Buche et al., 2006).

Un système tutoriel intelligent et adaptatif pour l'apprentissage de compétences en environnement virtuel de formation

Ce travail se situe dans le cadre de la réalisation d'environnements de formation utilisant la réalité virtuelle. Plus précisément, il s'intègre dans le projet MASCARET (Multi Agent System for Collaborative and Realistic Environment for Training) dont l'objectif est de développer un environnement virtuel, pour la formation au travail procédural et collaboratif.

Dans ce contexte, nous soutenons la thèse qu'il est possible d'intégrer un système tutoriel intelligent (ITS) générique et adaptatif dans un environnement virtuel, afin de fournir une aide pédagogique à l'apprenant et une assistance pédagogique au formateur.

Cette thèse débute par une étude montrant l'intérêt mutuel de la réalité virtuelle et des ITS pour l'apprentissage de compétences, et identifie les difficultés de leur intégration. Plus précisément, elle souligne la nécessité d'une représentation abstraite, indépendante de l'exercice à réaliser, manipulable pour la prise de décision pédagogique, et liée à la représentation d'un univers 3D.

Notre proposition est un système multi-agents permettant d'analyser l'action réalisée par l'apprenant par le biais d'un environnement virtuel informé. Le système dégage un ensemble d'informations, appelé situation pédagogique, considéré pertinent pour la prise de décision pédagogique. Notre étude se focalise alors sur un agent pédagogique, qui propose des assistances au formateur en utilisant la situation pédagogique. L'abstraction utilisée permet des assistances concrètes, liées au domaine, à l'exercice et à l'environnement virtuel. Le modèle comportemental de l'agent pédagogique se base sur un système de classeurs hiérarchique. Grâce à ce modèle, l'agent s'adapte au couple apprenant-formateur, en modifiant son comportement pédagogique par le biais d'un mécanisme d'apprentissage artificiel, basé sur un renforcement fourni par le formateur.

Ces travaux sont appliqués dans le cadre du projet GASPAR (Gestion de l'Activité aviation et des Sinistres sur Porte-avions par la Réalité virtuelle). L'application simule l'activité aviation sur un porte-avions.

Mots clefs : environnement virtuel de formation, système multi-agents, système tutoriel intelligent, système de classeurs.

Intelligent and adaptative tutoring system for the learning of competences in virtual environment for training

This work takes place in the framework of the processing of environment for training using virtual reality. More particularly, it integrates the MASCARET project (Multi Agent System for Collaborative and Realistic Environment for Training). The objective is to develop a virtual environment for training dedicated to collaborative and procedural work.

In this context, we defend the thesis that it is possible to integrate a generic and adaptive Intelligent Tutoring System (ITS) in a virtual environment, in order to provide pedagogical aid for the learner and pedagogical assistance for the trainer.

This thesis begins with a study showing the interest of virtual reality and ITS for the acquisition of competences and identifies their difficulties. More particularly, it shows up the necessity of an abstract representation, independent of the exercise to realize, manipulable for pedagogical decision making and connected to the representation of a 3D universe.

Our proposal is a multi-agent system allowing to analyse the action realized by the learner using an informed virtual environment. The system highlights a set of information, called pedagogical situation, considered relevant to make pedagogical decision. Then, our study is focused on a pedagogical agent. It suggests assistance to the trainer using the pedagogical situation. The abstraction used allows concrete assistance linked to the domain, to the exercise and to the virtual environment. The behavioural model of the pedagogical agent is based on a hierarchical classifiers system. Thanks to this model, the agent adapts itself to the trainer-learner pair, modifying its pedagogical behaviour by the way of an artificial learning mechanism based on a reinforcement provided by the trainer.

This work is applied to the GASPAR project. The application simulates the plane activity on an aircraft-carrier.

Key words : virtual environment for training, multi-agent system, intelligent tutoring system, classifiers system.

— Résumé —

Ce travail se situe dans le cadre de la réalisation d'environnements de formation utilisant la réalité virtuelle. Plus précisément, il s'intègre dans le projet MASCARET (Multi Agent System for Collaborative and Realistic Environment for Training) dont l'objectif est de développer un environnement virtuel, pour la formation au travail procédural et collaboratif.

Dans ce contexte, nous soutenons la thèse qu'il est possible d'intégrer un système tutoriel intelligent (ITS) générique et adaptatif dans un environnement virtuel, afin de fournir une aide pédagogique à l'apprenant et une assistance pédagogique au formateur.

Cette thèse débute par une étude montrant l'intérêt mutuel de la réalité virtuelle et des ITS pour l'apprentissage de compétences, et identifie les difficultés de leur intégration. Plus précisément, elle souligne la nécessité d'une représentation abstraite, indépendante de l'exercice à réaliser, manipulable pour la prise de décision pédagogique, et liée à la représentation d'un univers 3D.

Notre proposition est un système multi-agents permettant d'analyser l'action réalisée par l'apprenant par le biais d'un environnement virtuel informé. Le système dégage un ensemble d'informations, appelé situation pédagogique, considéré pertinent pour la prise de décision pédagogique. Notre étude se focalise alors sur un agent pédagogique, qui propose des assistances au formateur en utilisant la situation pédagogique. L'abstraction utilisée permet des assistances concrètes, liées au domaine, à l'exercice et à l'environnement virtuel. Le modèle comportemental de l'agent pédagogique se base sur un système de classeurs hiérarchique. Grâce à ce modèle, l'agent s'adapte au couple apprenant-formateur, en modifiant son comportement pédagogique par le biais d'un mécanisme d'apprentissage artificiel, basé sur un renforcement fourni par le formateur.

Ces travaux sont appliqués dans le cadre du projet GASPAR (Gestion de l'Activité aviation et des Sinistres sur Porte-avions par la Réalité virtuelle). L'application simule l'activité aviation sur un porte-avions.

Mots clefs : environnement virtuel de formation, système multi-agents, système tutoriel intelligent, système de classeurs.

— Abstract —

This work takes place in the framework of the processing of environment for training using virtual reality. More particularly, it integrates the MASCARET project (Multi Agent System for Collaborative and Realistic Environment for Training). The objective is to develop a virtual environment for training dedicated to collaborative and procedural work.

In this context, we defend the thesis that it is possible to integrate a generic and adaptive Intelligent Tutoring System (ITS) in a virtual environment, in order to provide pedagogical aid for the learner and pedagogical assistance for the trainer.

This thesis begins with a study showing the interest of virtual reality and ITS for the acquisition of competences and identifies their difficulties. More particularly, it shows up the necessity of an abstract representation, independent of the exercise to realize, manipulable for pedagogical decision making and connected to the representation of a 3D universe.

Our proposal is a multi-agent system allowing to analyse the action realized by the learner using an informed virtual environment. The system highlights a set of information, called pedagogical situation, considered relevant to make pedagogical decision. Then, our study is focused on a pedagogical agent. It suggests assistance to the trainer using the pedagogical situation. The abstraction used allows concrete assistance linked to the domain, to the exercise and to the virtual environment. The behavioural model of the pedagogical agent is based on a hierarchical classifiers system. Thanks to this model, the agent adapts itself to the trainer-learner pair, modifying its pedagogical behaviour by the way of an artificial learning mechanism based on a reinforcement provided by the trainer.

This work is applied to the GASPAR project. The application simulates the plane activity on an aircraft-carrier.

Key words : virtual environment for training, multi-agent system, intelligent tutoring system, classifiers system.