PEGASE : un système tutoriel intelligent générique et adaptatif en environnement virtuel

C. Buche — R. Querrec — P. De Loor — P. Chevaillier — J. Tisseau

Université Européenne de Bretagne/École Nationale d'Ingénieurs de Brest
Laboratoire d'Informatique des Systèmes Complexes
Centre Européen de Réalité Virtuelle
25 rue Claude Chappe, F-29280 Plouzané
[buche,querrec,deloor,chevaillier,tisseau]@enib.fr

RÉSUMÉ. *Ce travail se situe dans le cadre de la réalisation d'environnements d'apprentissage humain utilisant la réalité virtuelle. Dans ce contexte, nous proposons d'intégrer un système tutoriel intelligent générique et adaptatif (appelé* PEGASE*) dans un environnement virtuel dont l'objectif est de fournir une aide pédagogique à l'apprenant et une assistance pédagogique au formateur. Le système proposé est réalisé par un système multi-agent. Il dégage un ensemble de connaissances (actions réalisées par l'apprenant, connaissances du domaine...) qui sont utilisées par* PEGASE *pour prendre des décisions pédagogiques. Notre étude se focalise sur la représentation des connaissances de l'environnement et sur l'agent pédagogique adaptatif qui propose des assistances pédagogiques.*

ABSTRACT. *This work takes place in the framework of the design of a training environment using virtual reality. In this context, we defend the thesis that it is possible to integrate a generic and adaptive intelligent tutoring system (called* PEGASE*) in a virtual environment, in order to provide pedagogical aid for the learner and pedagogical assistance for the trainer. Our proposal is a multi-agent system. It highlights a set of information (actions realized by the learner, domain knowledge...) used by* PEGASE *to take pedagogical decisions. Our study is focused (1) on knowledge representation for the environment and (2) on an adaptive pedagogical agent suggesting pedagogical assistances.*

MOTS-CLÉS : *environnement virtuel d'apprentissage humain, système multi-agent, tuteurs intelligents, système de classeurs*

KEYWORDS: *virtual environment for training, multi-agent system, intelligent tutoring system, classifiers system*

RSTI - TSI. Volume 28 – n° 8/2009, pages 1051 à 1076

1. Introduction

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. Au centre européen de réalité virtuelle, nous développons des environnements virtuels (EV) qui immergent les apprenants dans de telles situations. La figure 1 représente l'application ARÉVIROAD destinée à la sécurité routière (Herviou *et al.*, 2006), SÉCURÉVI pour la sécurité civile (Querrec *et al.*, 2003) et GASPAR pour la logistique sur porte-avions (Buche, 2005; Marion *et al.*, 2007).

Figure 1. *De gauche à droite : captures d'écran des applications* ARÉVIROAD*,* SÉCURÉVI *et* GASPAR

Le travail présenté ici a été réalisé dans le contexte de ces EV et vise plus particulièrement la formation à la prise de décision destinée au travail procédural et collaboratif. Ces EV sont des systèmes complexes. Nous souhaitons disposer d'une collection d'exercices pour chaque application et il n'est pas question de redévelopper de multiples environnements (un EV par exercice).

Classiquement, les formations visent à l'apprentissage de *savoir*. Pour favoriser l'acquisition de savoir, il s'agit de manipuler des connaissances. Dans ce cadre, nous proposons l'utilisation de systèmes tutoriaux intelligents (en anglais ITS : *Intelligent Tutoring System*) qui utilisent les connaissances liées à la situation de formation. Ces connaissances seront manipulées, par exemple, pour interroger l'apprenant automatiquement. Dans notre cas, il s'agit d'acquérir une compétence. Or, être compétent, c'est non seulement acquérir un savoir mais également un *savoir-faire*. Pour favoriser l'acquisition de savoir-faire, il faut mettre en situation l'apprenant. Pour cela, nous proposons l'utilisation de systèmes interactifs permettant d'immerger les apprenants dans des environnements virtuels où ils peuvent essayer, prendre des initiatives, échouer et recommencer (là où des environnements réels ne le permettent pas forcément). La simulation fournit alors un environnement commun à l'apprenant, à la compétence à acquérir et au formateur. Elle médiatise la relation d'apprentissage (apprenant-compétence) et la relation pédagogique (apprenant-formateur). Ainsi, les simulations informatiques, combinées avec un ITS, ont la possibilité d'améliorer les compétences des apprenants puisqu'ils associent savoir et savoir-faire.

Les ITS ont déjà été utilisés indépendamment de la réalité virtuelle. Comme le montre (Wenger, 1987), ils utilisent généralement quatre modèles. Le premier, appelé

modèle du domaine, contient une représentation des connaissances liées à la compétence à acquérir. Les ITS utilisent également un modèle de l'apprenant qui précise ses caractéristiques personnelles et qui établit l'état de ses connaissances à un instant donné. Utilisant le modèle du domaine et de l'apprenant, l'ITS est capable d'évaluer les connaissances acquises par l'apprenant, en comparant ses activités et les informations sur le domaine. Mais l'objectif principal de l'ITS est de fournir une assistance adaptée à la situation pour l'apprenant ou le formateur (suivi d'activités ou propositions d'assistances). Dans ce cadre, le modèle pédagogique permet d'effectuer des choix concernant la formation et visant à favoriser l'apprentissage. Enfin, un modèle d'interface, permet l'échange d'informations entre le système et l'utilisateur. Ce modèle n'est pour l'instant pas réifié dans les EV existants destinés à l'apprentissage.

Dans le cadre de nos EV, nous considérerons un ITS comme étant un système qui fait partie d'un environnement virtuel d'apprentissage humain (EVAH). Nous proposons d'évaluer le niveau d'intégration des ITS dans les EVAH existants que nous avons regroupés selon trois catégories (Buche *et al.*, 2006a) :

1) *L'EVAH est un simulateur classique*
Cette première catégorie regroupe les applications n'intégrant aucun des quatre modèles, comme dans l'application d'aide à la maintenance et à la conduite des ponts roulants présentée dans (Levesque, 2003). Dans ce type d'EV, le système ne peut pas fournir des explications sur la tâche à réaliser, ce qui nécessiterait un modèle du domaine. Il ne peut pas s'adapter à un apprenant, ce qui nécessiterait un modèle de l'apprenant. Enfin, la pédagogie est de la responsabilité du formateur. Le système ne peut pas prendre des décisions relatives à des interventions pédagogiques. Toutefois, un tel système permet le renforcement de compétences préalablement acquises.

2) *L'EVAH contient des modèles du domaine et/ou de l'apprenant*
Cette deuxième catégorie d'EV est constituée des applications intégrant un modèle du domaine et/ou un modèle de l'apprenant. L'exemple le plus connu est STEVE, un personnage qui a pour objectif la formation à des tâches procédurales (Rickel *et al.*, 1999). Utilisant le modèle du domaine, il sait montrer la procédure, l'expliquer et surtout valider les actions de l'apprenant. Cependant, STEVE intervient à la demande. Il est incapable de savoir quand, comment et pourquoi intervenir, ce qui nécessiterait un modèle pédagogique. Dans un tel système, l'acquisition de compétence est possible, mais les interventions pédagogiques restent du ressort du formateur.

3) *L'EVAH contient des modèles du domaine, de l'apprenant, et pédagogique*
Cette dernière catégorie rassemble les EV qui présentent non seulement les modèles du domaine et de l'apprenant, mais également un modèle pédagogique. Prenons pour illustration l'agent pédagogique HAL utilisé dans le système FIACRE (Lourdeaux, 2001). L'application est destinée à la formation individuelle des agents de conduite de train à grande vitesse (TGV) à l'aide de réalité virtuelle (intervention sur les voies ferrées). Outre les capacités de STEVE, HAL aide les formateurs à construire un discours pédagogique. Concrètement, pour chaque comportement attendu, les assistances pédagogiques sont décrites (mise en transparence d'objet, ajout d'information...). Le formateur doit lister de manière exhaustive les erreurs pour chaque connais-

sance. De plus pour chacune de ces erreurs, il doit préciser la manière dont les stratégies pédagogiques sont réalisées par des assistances pédagogiques et cela pour chaque exercice. L'intérêt principal réside dans l'aide apportée au formateur dans la relation pédagogique qui le lie à l'apprenant et dans la relation didactique qui le lie à la compétence. Toutefois, le formateur doit expliciter l'ensemble des connaissances pour chaque exercice.

Ainsi, la plupart des EV n'incorpore 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. HAL nous semble être le système le plus abouti. Néanmoins, le formateur doit lister les erreurs et préciser les stratégies pédagogiques pour chaque exercice. Bien plus, l'impact des assistances pédagogiques sur l'apprenant n'est pas considéré. Concrètement, une assistance pédagogique ne faisant pas progresser un apprenant sera remise en place à chaque fois que la situation en question apparaît.

Pour palier ces limitations, notre proposition consiste à intégrer un système tutoriel intelligent au sein d'un EV. Ce système doit proposer un modèle pédagogique modulable, *i.e.* permettant d'intégrer, de modifier ou de supprimer des concepts pédagogiques facilement. De plus, il doit être *générique* dans le sens où le modèle pédagogique devra être utilisable indépendamment de l'exercice à réaliser. Enfin, 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 : le système devient *adaptatif*. Le modèle proposé se nomme PEGASE (*PEdagogical Generic and Adaptive SystEm*).

Dans la section 2, nous décrivons l'architecture globale de PEGASE. Ensuite, nous présentons notre modèle du domaine (cf. section 3). Ceci nous amène à exposer notre modèle pédagogique (cf. section 4). Nous discutons par la suite de l'apport de notre proposition (cf. section 5). Notons que la proposition développée ici s'applique dans le cadre de l'*apprentissage de tâches procédurales et collaboratives* et n'est pas utilisable dans les situations d'apprentissage d'ordre général.

2. Proposition d'un système tutoriel intelligent : PEGASE

Notre proposition consiste à réifier les quatre modèles classiques d'un ITS au sein d'un EV (domaine, apprenant, pédagogie, interface). Comme (Py, 1998), nous considérons les erreurs comme des informations cruciales. Par conséquent nous avons décidé d'introduire un modèle appelé « modèle des erreurs ». C'est grâce à ce modèle que nous allons pouvoir généraliser (là où HAL ne le pouvait pas). De plus, nous ajoutons un « modèle du formateur » qui définit les connaissances sur l'exercice à réaliser précisées par le formateur. Il fixe les consignes, qui définissent le(s) procédure(s) à effectuer, et le(s) rôle(s) à jouer par tel apprenant (et par conséquent ceux qui seront joués automatiquement).

Ces modèles devront répondre aux limitations des systèmes existants que nous avons dégagées et par conséquent présenter deux propriétés : généricité et adaptativité. Ainsi, nous soutenons qu'il est possible, à partir d'un environnement virtuel, de greffer un ITS générique et adaptatif en réifiant les 6 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.

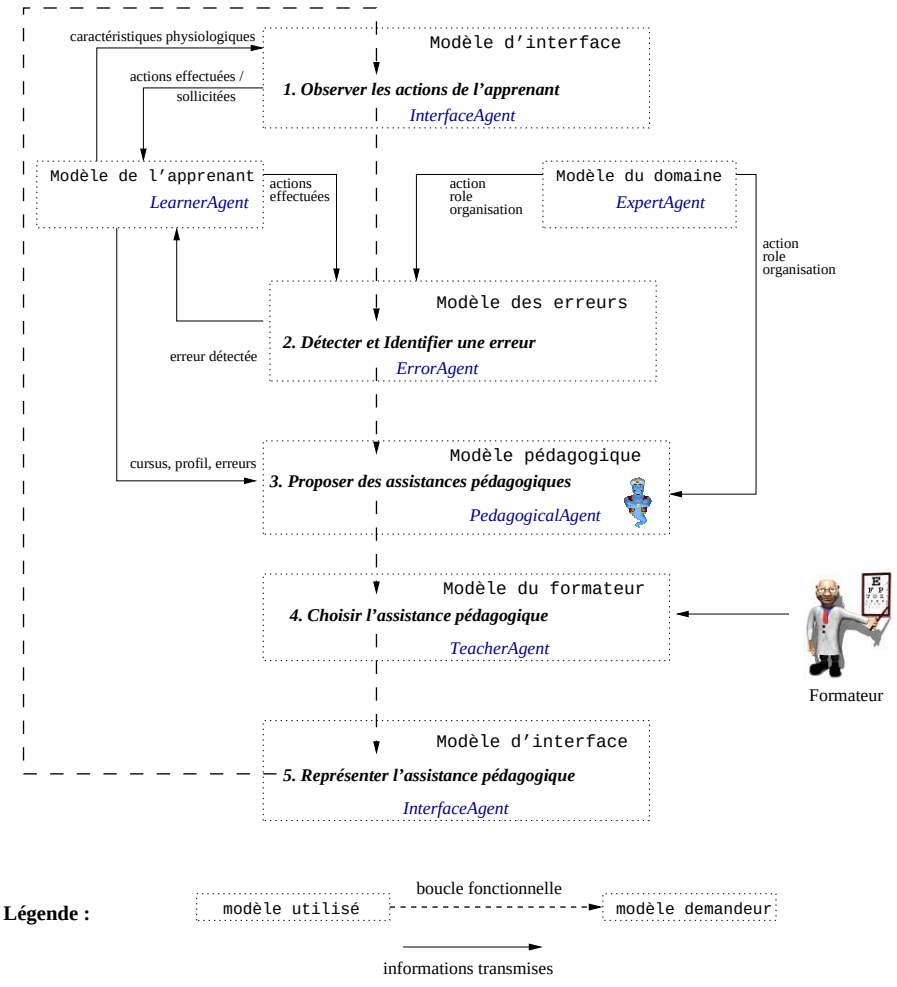

Figure 2. *Processus pédagogique de notre système constitué de cinq étapes*

Les agents interagissent en s'échangeant des messages contenant des données (cf. figure 2). Elles peuvent être extraites de la simulation ou déduites d'un raisonnement interne à l'agent à partir de ses connaissances (modèle auquel il est lié) :

Étape 1. *Observer*
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 les actions de l'apprenant, les éléments observables par l'apprenant, les déplacements de l'apprenant.

Étape 2. *Détecter et identifier une erreur*
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).

Étape 3. *Proposer des assistances pédagogiques*
Utilisant le modèle de l'apprenant (caractéristiques, activités, erreurs, 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. *Sélectionner une assistance pédagogique*
Le formateur peut sélectionner une assistance pédagogique parmi les propositions.

Étape 5. *Représenter l'assistance pédagogique*
L'assistance pédagogique sélectionnée est présentée dans l'environnement virtuel.

Pour utiliser les informations de l'EV, nous devons informer l'environnement afin d'obtenir des connaissances manipulables : nous obtenons un environnement virtuel informé (cf. section 3). Nous devons alors *réifier l'environnement*. Ces connaissances sont complétées par des informations supplémentaires contenues dans les 6 modèles de l'ITS. Ces données forment une base de connaissances pour le modèle pédagogique que nous appelons *situation pédagogique*. Ces connaissances alimentent le moteur de prise de *décision pédagogique* de l'ITS (cf. section 4). Un exemple de spécification des règles qui régissent ce moteur est présenté (cf. section 4.3).

3. Modèle du domaine

Pour la réification, nous définissons le métamodèle `Veha` (*Virtual Environment for Human Activity*) (Chevaillier *et al.*, 2009). L'objectif de `Veha` est de fournir un métamodèle pour décrire l'environnement virtuel (EV), non pas en tant qu'espace géométrique mais en fournissant une sémantique permettant à des agents artificiels (ITS, personnages autonomes) ou humains (apprenants et formateurs) de s'en construire une représentation et d'y agir conjointement pour atteindre leurs buts. Le métamodèle `Veha` s'appuie sur `Uml` et permet de construire des modèles métiers d'environ-

nements virtuels et les environnements virtuels concrets correspondants (cf. tableau 1). Toutefois, le métamodèle ne permet pas d'expliciter les concepts spécifiques à la réalité virtuelle. Dans Veha, nous proposons une extension d'Uml pour représenter ces concepts. Veha est destiné à réifier les EV réalistes dans lesquels des humains réalisent des activités. Il n'est pas adapté à tous les EV (métaphoriques, phénoménologiques...).

M4	Mof (restriction d'Uml)				
M3	métamodèle Uml	métamodèle Veha			
M2	*user model* Uml	modèle EV_1		...	
M1	*user object*	EV_{1a}	EV_{1b}	...	...

Tableau 1. *Couches de modélisation (Mi) : positionnement de Veha dans le référenciel Mof, parallèle avec Uml*

3.1. *Le métamodèle Veha*

L'ITS a besoin de savoir de quels objets est composé l'environnement virtuel, d'en connaître l'accessibilité, les propriétés, le comportement, et ainsi de savoir comment interagir avec eux. Trois types de connaissances peuvent être exprimés à l'aide de Veha :

– les concepts du domaine. Il s'agit de la description sémantique des concepts du domaine d'activité abordé. Cela représente en partie les connaissances que l'apprenant doit acquérir (section 3.1.1) ;

– les possibilités d'interaction et la structuration de l'environnement. Ces notions s'apparentent à celles proposées dans les *smart objects* (Kallmann *et al.*, 1998) qui réifient des propriétés nécessaires aux interactions. Il s'agit d'expliciter les moyens dont l'apprenant ou l'ITS dispose pour modifier l'environnement (section 3.1.2) ;

– les comportements des entités. Dans le cadre d'un EVAH, il faut simuler les réactions de l'environnement aux actions de l'apprenant. Le comportement des entités est également une partie de la connaissance à transmettre et doit être exécutable (section 3.1.3).

Dans la suite de cette section, nous décrivons comment Veha permet d'exprimer ces trois types de connaissances.

3.1.1. *Les concepts du domaine*

Les connaissances sur le domaine sont exprimées au niveau du modèle (les concepts) et au niveau des instances de ces concepts (objets concrets peuplant l'environnement). En Veha comme en Uml, ces connaissances sont représentées par des classes (Class) et des instances (InstanceSpecification).

En Veha, la notion de classe permet de définir un type d'objet (figure 3) issu de l'ontologie du domaine. Son objectif est de pouvoir donner une sémantique aux diffé-

rents concepts métiers, qu'ils aient ou non une représentation tangible dans l'environnement virtuel (concepts *vs.* objets concrets). Toutes les classes héritent de la classe `Element`. Cette classe permet d'identifier chaque élément d'un modèle métier par son nom et d'ajouter un commentaire textuel, ceci est utile pour fournir des explications à l'utilisateur sur la signification d'un objet.

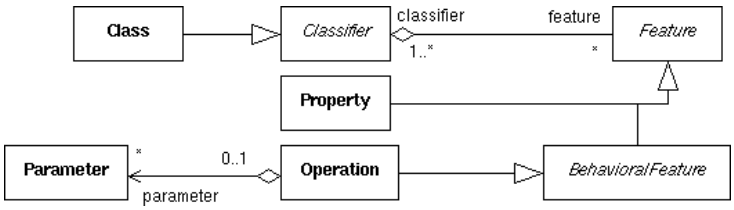

Figure 3. *Diagramme de classes du métamodèle `Veha: Features` d'un `Classifier`*

Les propriétés structurelles (`Property`) et comportementales (`BehavioralFeature`) des classes sont associées à `Classifier`[1] *via* la classe `Feature`. La classe `Property` permet de représenter la composante structurelle (aussi bien les attributs que les relations avec d'autres concepts métiers) du `Classifier`. Comme en `Uml`, la classe `Operation` est la seule sous-classe concrète de `BehavioralFeature`. Elle permet d'exprimer ce qu'un objet ou un utilisateur peut effectuer sur un autre objet. Il s'agit d'une spécification du comportement de l'objet et non de la méthode utilisée pour le réaliser. La manière de modéliser les comportements associés à l'`Operation` est spécifiée à l'aide des modèles comportementaux (cf. section 3.1.3).

Après celle de `Class`, la notion d'`Instance`, synonyme d'objet, est le deuxième concept clé de `Veha`. Les classes `InstanceSpecification`, `Slot` et `AssociationInstance` représentent respectivement l'instantiation de `Class`, `Property` et `Association`. Le terme `InstanceSpecification` indique que l'on représente ici une entité du niveau M1 (cf. tableau 1), indépendemment de son implémentation.

L'ensemble des connaissances sur l'environnement explicité en `Veha` est accessible pour l'ITS et pour les utilisateurs (apprenants ou formateurs). L'ITS peut, par exemple, proposer à l'apprenant une liste d'opérations à exécuter sur un type d'objet et le formateur peut modifier l'environnement en cours de simulation en modifiant les valeurs des attributs d'un objet concret.

3.1.2. *Les possibilités d'interaction et la structuration de l'environnement*

Dans le cadre des environnements virtuels, la plupart des objets concrets de l'environnement ont une représentation géométrique et sont positionnés dans l'environnement. L'apprenant doit pouvoir les observer, les reconnaître et les manipuler. L'ITS a

1. Classe du méta modèle UML qui généralise la notion de classe.

également besoin de les manipuler dans le cadre des assistances pédagogiques qu'il met en œuvre (mise en transparence, recentrage du point de vue de l'apprenant...). Les connaissances sur la géométrie de ces objets doivent donc également être explicitées pour que l'ITS puisse recontextualiser ses assistances dans l'environnement virtuel. Ces objets sont des entités et ont toutes les propriétés des instances de `veha::Class` ainsi que des propriétés topologiques et géométriques (cf. figure 4).

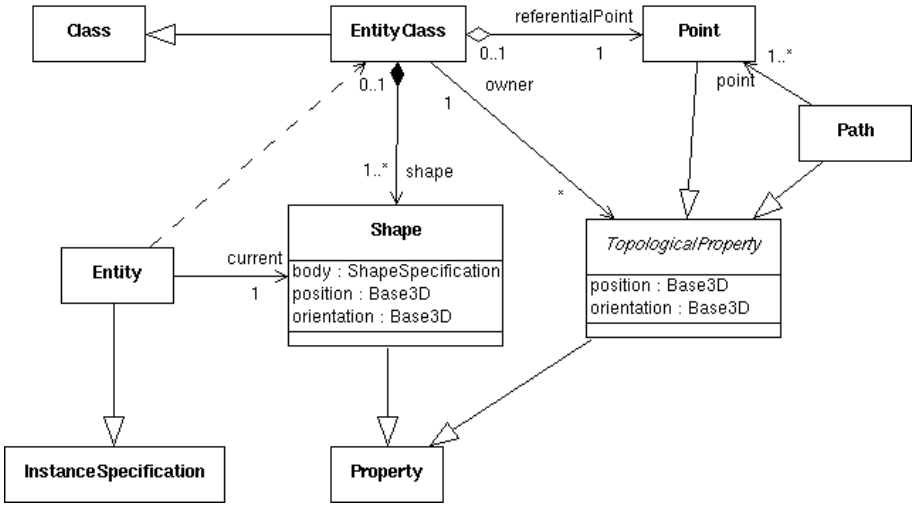

Figure 4. *Diagramme de classe du métamodèle* `Veha : EntityClass`

Chaque entité est localisée dans un repère global. La classe `Shape` permet de doter une instance d'`EntityClass` d'une représentation graphique dans l'environnement virtuel. Il est possible d'attacher plusieurs formes à une classe d'entités. L'ITS peut se servir de cette connaissance pour mettre en évidence un objet ou au contraire le faire disparaître. La classe `TopologicalProperty` supporte la notion de localisation (position et orientation) et permet d'exprimer des propriétés topologiques sur les éléments de l'environnement virtuel (cf. figure 4). Il est possible d'associer à une entité des points informés (`Point`) qui sont utilisables pour la réalisation d'interaction. Ces informations servent à l'ITS pour, par exemple, positionner le point de vue de l'apprenant sur l'objet important.

Une entité de l'environnement virtuel est une instance de `EntityClass` : la classe `Entity` dérive de `Kernel::InstanceSpecification`. Les valeurs des propriétés d'une entité sont définies par ses `slots`. Il s'agit donc des propriétés sémantiques (apportées par `InstanceSpecification`), morphologiques, géométriques et topologiques des objets de l'environnement virtuel.

3.1.3. *Les comportements des entités*

Lorsque l'apprenant effectue une action dans l'environnement, il faut que celui-ci réagisse de manière réaliste pour que l'apprenant puisse comprendre l'effet de ses actions. L'apprenant se fait alors une représentation du comportement des entités. Pour que l'ITS puisse contrôler cette représentation, la connaissance sur le comportement des entités doit donc également être explicite comme les deux précédents types de connaissances. Elle doit également être exécutable.

Le package `Behavior` a pour rôle de modéliser les comportements possibles des entités de l'environnement virtuel, l'objectif étant que le modèle soit interprétable en temps réel par un contrôleur comportemental et introspectable en ligne. Comme pour les aspects structuraux, l'introspection porte à la fois sur le modèle comportemental (M2) et sur son « instanciation », c'est-à-dire son exécution (M1). Les deux classes qui supportent ces notions sont `Behavior` et `BehaviorExecution` (figure 5). Les entités `Veha` ont des comportements réactifs qui sont déclenchés sur occurrence d'un événement qui peut être produit par une action de l'apprenant ou par une autre entité de l'environnement virtuel.

Classiquement, on associe à un comportement, des préconditions et post-conditions qui portent sur les entités de l'environnement. La modélisation comportementale repose sur les machines à états et le modèle d'activité d'`Uml`. Enfin, il peut s'agir d'une fonction écrite dans un langage de programmation et invocable en ligne (`OpaqueBehavior`). Les deux premières méthodes sont introspectables, l'ITS peut alors décrire ou vérifier l'exécution du comportement.

Le tuteur peut ainsi analyser, expliquer ou vérifier le contexte d'exécution du comportement d'une entité sollicitée par l'apprenant. Mieux, si ce comportement a été décrit de manière explicite (machine à états ou activités) il peut également en expliquer l'exécution.

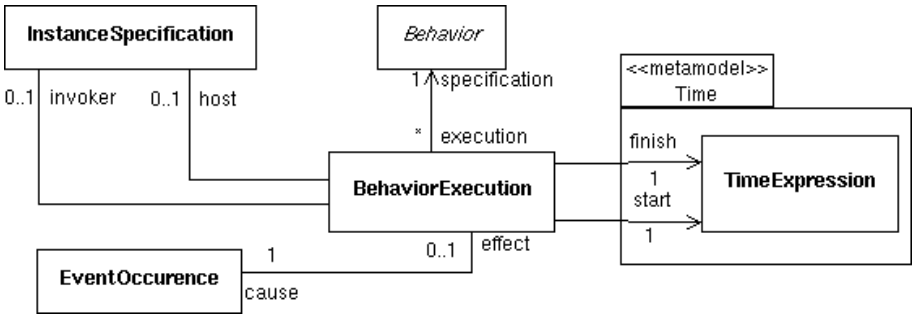

Figure 5. *Diagramme de classes du métamodèle `Veha` : package Behavior : :Common, la classe `BehaviorExecution`*

3.2. *Exemple d'environnement en* `Veha`

Le métamodèle `Veha` permet d'interpréter automatiquement un modèle décrit en `Uml`. La figure 6 présente le diagramme de classe d'un exemple d'environnement virtuel en `Veha`. Il s'agit d'un exemple issu d'une application réalisée en `Veha`, mais fortement simplifiée pour les besoins de la démonstration. L'application (GASPAR, (Marion *et al.*, 2007)) est constituée d'une cinquantaine de classes et de plus de mille entités. Ce modèle décrit les classes `Déflecteur` et `CabineCatapulte` (fenêtre de gauche). Une cabine catapulte permet d'abriter des opérateurs sur la piste de catapultage d'un porte-avions. Une cabine peut s'ouvrir (s'élever sur la piste) ou se fermer (s'enfoncer dans la piste). Le modèle métier décrit l'ensemble de propriétés d'une cabine (`hauteur`, `vitesse`...). Le comportement réactif d'une cabine est décrit par une machine à états (fenêtre en haut à droite). Cette machine à états est sensible aux signaux `Ouvrir` et `Fermer`. Ainsi lorsque la cabine est dans l'état `Fermée`, si elle reçoit le signal `Ouvrir`, elle passe dans l'état `SOuvre` et réalise l'opération `Ouvrir()`. Dans le cadre de cette application, cette opération est décrite par un `OpaqueBehavior`, un code C++ qui réalise le déplacement visuel de la cabine en fonction de l'attribut vitesse et met à jour l'attribut hauteur.

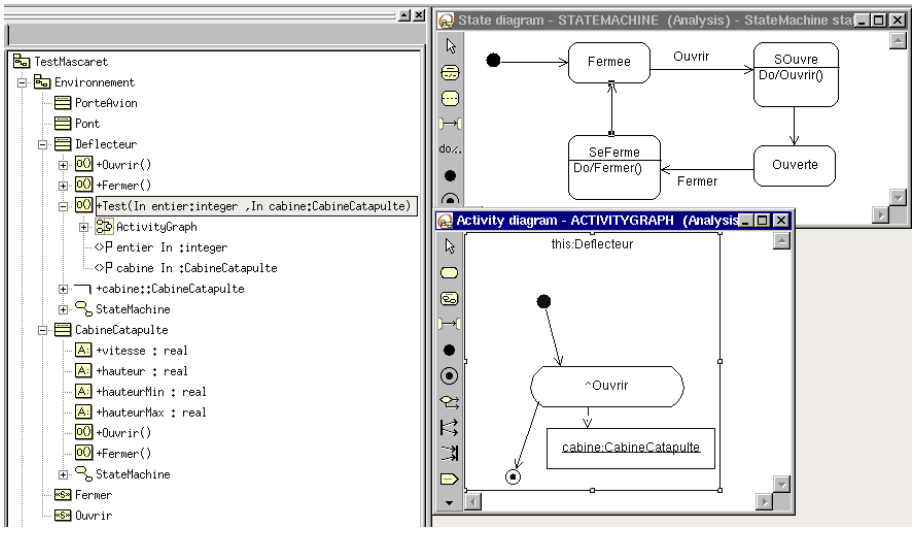

Figure 6. *Diagramme de classes d'un déflecteur et d'une cabine catapulte*

Un déflecteur agit sensiblement de la même manière. Pour les besoins de cette démonstration, nous ajoutons une opération de test (`Test`). Cette opération prend en paramètre une cabine catapulte. Le comportement de cette opération est décrit par un diagramme d'activité (fenêtre en bas à droite). Ainsi lorqu'une opération `Test` est invoquée sur une instance de la classe `Déflecteur`, l'opération envoie le signal `Ouvrir` à la cabine passée en paramètre.

Ce modèle est décrit en utilisant le logiciel de modélisation *Objecteering*, il est ensuite exporté dans un fichier au format XMI. Une première implémentation d'un interpréteur pour le métamodèle `Veha` a été proposée en utilisant l'atelier de réalité virtuelle *AReVi*. L'interpréteur lit le fichier XMI et pour chaque classe du métamodèle `Uml` crée une instance de la classe correspondante dans le métamodèle `Veha`. Ainsi pour chaque classe métier décrite dans le fichier XMI, l'interpréteur crée une nouvelle instance de la classe `class` du métamodèle `Veha`. Dans le cadre de notre exemple, une instance de la classe `Class` est créée pour la classe `Deflecteur` et une autre pour la classe `CabineCatapulte`. L'interpréteur permet de réifier le modèle métier et fournit un ensemble de méthodes permettant l'introspection de ce modèle. Il est alors possible de demander à l'interpréteur l'ensemble des propriétés d'une classe, le signal qui permet de faire transiter d'un état à un autre et l'opération qui sera alors réalisée, cela indépendemment de tout objet concret.

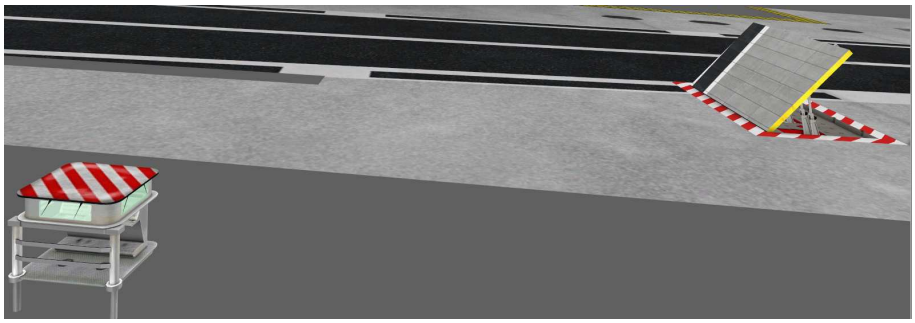

Figure 7. *Visualisation d'instances de `CabineCatapulte` et de `Deflecteur`*

L'environnement virtuel est peuplé d'entités, instances de la classe `Entity` du métamodèle `Veha`. D'un point de vue technique, ces instances sont décrites dans un fichier XML. `Uml` permet également de décrire les instances des classes et de les exporter dans le fichier XMI. Toutefois, aucun modeleur `Uml` ne permet de facilement attribuer une géometrie et une position à ces instances. La conception géométrique de l'environnement virtuel est en général issue de modeleurs spécialisés tels que *3DS MAX* ou *Blender*. Nous proposons donc un `plugin` d'export pour *3DS Max* qui permet de générer le fichier d'instance lu par l'interpréteur. La figure 7 montre le résultat visuel de l'interpétation du fichier décrivant le modèle (XMI) et du fichier d'instance (XML) dans une application implémentée à l'aide d'ARéVi. L'interpréteur fournit également des méthodes pour interroger et manipuler les entités. Il est alors possible de demander à une entité les valeurs de ses propriétés, d'exécuter une opération ou de lui envoyer un signal pour la faire changer d'état.

3.3. *Procédure et collaboration*

Nous nous intéressons ici à l'acquisition de compétences. Le modèle du domaine ne contient pas uniquement les connaissances sur l'environnement manipulé, mais également les connaissances sur la tâche qui doit y être réalisée par les apprenants. Dans le cadre de ce travail et des exemples cités en introduction, les activités sont décrites par des procédures qui agencent les `Actions` à réaliser par plusieurs intervenants dont les rôles sont bien définis. Nous utilisons le même principe que pour l'environnement et nous proposons un métamodèle s'appuyant sur le langage `Uml` pour décrire ces activités ; ainsi, les procédures sont décrites par des diagrammes d'activités. Ce type de diagramme dispose des possibilités classiques d'agencement des `Actions` (parallélisme, séquence, jonction, conditionnel...). S'agissant de représenter des activités humaines, nous considérons que l'enchaînement des activités s'effectue de manière asynchrone.

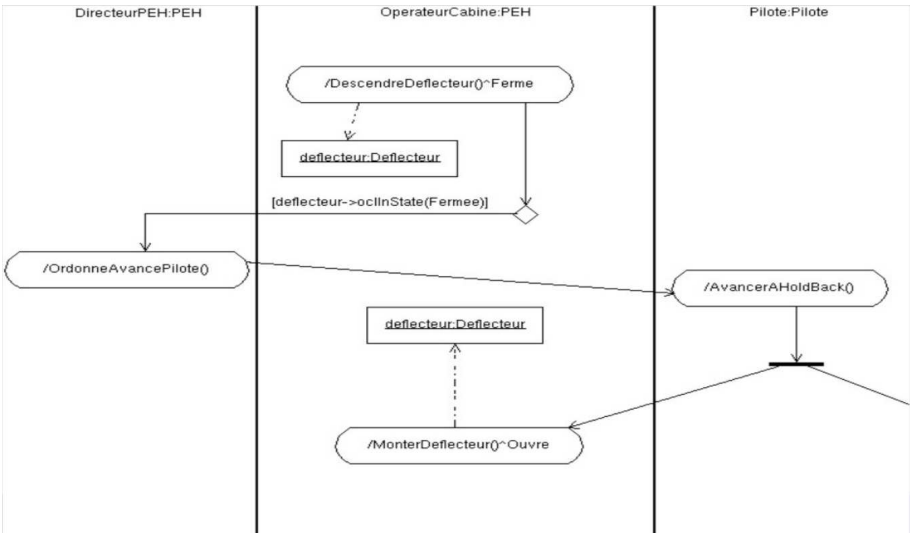

Figure 8. *Exemple de procédure écrite à l'aide d'un diagramme d'activité*

Les rôles de l'organisation sont représentés par les couloirs d'activités. Le nom du couloir identifie le rôle et le type du couloir, le type d'agent autorisé à prendre ce rôle. Comme en `Uml` 2.1, les activités sont de plusieurs types. Il peut s'agir de l'exécution d'opérations de l'agent, d'une action de base de réalité virtuelle (jouer une animation, atteindre un point...) ou de l'envoi d'un signal à une ressource. Les ressources sont jouées par les entités de l'environnement et représentées par des objets en `Uml`. Les conditions sont exprimées en `Ocl` et portent sur les rôles et les ressources intervenant dans la procédure. La figure 8 montre l'exemple d'une procédure exprimée grâce à un diagramme d'activité. Cette procédure fait intervenir 3 rôles (`OpérateurCabine` par exemple) qui doivent être joués par des personnages d'un type défini (`PEH` par

exemple). Ce sont les personnages effectivement instanciés dans l'environnement qui jouent ces rôles. Cette procédure vise à faire avancer l'avion à catapulter jusqu'à un point précis en manipulant le déflecteur (sorte de plaque de protection). L'exemple de la procédure de la figure 8 montre la complémentarité entre les machines à états utilisées pour décrire le comportement réactif des objets de l'environnement et les diagrammes d'activités servant à décrire une procédure. Une action de la procédure peut se traduire par l'envoi d'un événement à l'objet manipulé et les conditions de passage à l'action suivante peuvent dépendre de l'état actuel de l'objet.

Nous avons implémenté un comportement d'agent utilisant la connaissance sur les procédures pour choisir leurs actions. L'apprenant joue un ou plusieurs rôles dans le cadre de ces procédures. L'ITS s'appuie également sur cette connaissance pour choisir les assistances. Comme pour l'environnement, il existe 2 niveaux de modélisation accessibles aux agents (dont l'ITS) et aux utilisateurs (formateurs), la structure organisationnelle et les instances organisationnelles. Ainsi le tuteur intelligent est capable de connaître l'enchaînement des actions indépendemment de toute organisation. Il peut également connaître plus spécifiquement à quel endroit est rendue l'exécution de la procédure en cours de réalisation dans l'équipe dans laquelle l'apprenant joue un ou plusieurs rôles. Il est ainsi capable de détecter des erreurs de l'apprenant sur l'ordre des actions et sur le non respect des conditions exprimées dans la procédure (Trinh *et al.*, 2009).

4. Modèle pédagogique

Les connaissances sur l'environnement (les entités et la tâche à réaliser) sont représentées grâce au modèle `Veha`. Notre ITS peut donc les manipuler pour construire ses propres connaissances comme nous le montrerons (section 4.1) et simuler un raisonnement pédagogique (section 4.2). Ensuite, une mise en œuvre concrète de l'ITS est proposée en section 4.3 (spécification des règles du raisonnement pédagogique simulé).

4.1. *Situation pédagogique*

Rappelons que nous nous positionnons dans le cadre d'apprentissages « en situation ». Dans ce cadre théorique, les éléments du contexte sont considérés prépondérants dans la prise de décision de l'ITS (Turner, 1993; Pomerol *et al.*, 2001). Dans notre cas, nous appelons contexte la situation pédagogique qui sert de base pour la prise de décision. 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 spécifique à réaliser. Pour cela, nous séparons les connaissances sur la tâche à effectuer (cf. section 4.1.1) des connaissances sur l'apprenant (cf. section 4.1.2).

4.1.1. *Informations concernant la tâche à effectuer*

Nous nous plaçons dans le cadre d'une formation au travail procédural. L'objectif de l'ITS 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.

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, c'est-à-dire la réalisation de la procédure, il faut effectuer un ensemble de sous-buts liés causalement. Il s'agit alors d'étudier la procédure en considérant la distance au but de la procédure d'un point de vue *causal*, 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 causaux 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 (actions 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 proche 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. Techniquement, ceci est réalisé en effectuant une reconnaissance de plan en se basant sur le diagramme d'activité `Veha` présenté en section 3.3. 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.

2) *Les liens causaux entre sous-buts*

La procédure peut être considérée comme un graphe représentant l'enchaînement des sous-buts causaux. 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...). 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. Techniquement, il s'agit des liens entre postconditions et préconditions évoqués en section 3.1.3.

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, nous considérons les connaissances du tableau 2. Toutes les actions identifiées précédemment (liens d'ordonnancement et causaux) constituent la situation pédagogique. Plus précisément nous nous intéressons aux informations liées aux actions sélectionnées. A ce stade, il devient nécessaire de préciser les connaissances relatives à la notion d'action. Dans cette optique, le « contexte d'action » est constitué des connaissances liées directement à l'`Action` (description, ressources...), des connaissances liées à l'`Operation` cible de l'`Action`, ainsi que les connaissances liées à l'agent qui a sollicité l'action puisqu'il est le protagoniste. Ainsi, nous utilisons les *contextes d'action* pour représenter les connaissances associées aux actions (sous-ensemble de l'environnement constitué d'entités et d'agents considéré comme pertinent dans le contexte de l'action).

Connaissances	Nature	Description
① Contexte d'action précédente	Ordonnancement	La dernière action correcte à avoir été réalisée. Cette action fait référence et permet de se positionner dans la procédure.
② Contexte d'action sollicitée	Ordonnancement	Il s'agit de l'action sollicitée. Cette action peut être correcte, comme erronée. Elle n'est pas forcément réalisée, conformément au modèle pédagogique.
③ Contexte d'action(s) correcte(s) sans considération sur les rôles	Ordonnancement	En considérant la dernière action correcte, nous déterminons les actions à effectuer selon la procédure courante.
④ Contexte d'action(s) correcte(s)	Ordonnancement	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)	Ordonnancement	Pour chaque action correcte, nous déterminons les actions qui suivent selon la procédure courante.
⑥ Contexte d'action(s) liée(s)	Causale	En considérant les actions à réaliser à la suite de la dernière action correcte, nous récupérons les liens « causaux » entre actions indépendamment de la procédure. Nous obtenons les actions qui sont liées.

Tableau 2. *La situation pédagogique : connaissances sur la tâche à réaliser*

La construction de cet ensemble de connaissances est à la charge de l'agent pédagogique. L'agent pédagogique récupère ou construit les connaissances sur le travail à réaliser lorsqu'il reçoit un message de l'agent d'interface qui précise qu'une action vient d'être sollicitée. Ce choix peut être discuté. En effet, une autre solution est de mettre à jour les connaissances sur erreur. Nous avons préféré 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 (exemple : affirmer des règles fausses qui entrent en contradiction avec ses choix).

4.1.2. *Informations concernant l'apprenant*

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 (exemple : âge) que dynamiques (exemple : éléments de la mémoire instantanée).

Notons que les erreurs effectuées par l'apprenant sont stockées, c'est-à-dire le type et éventuellement d'autres connaissances sur l'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. En outre, il est nécessaire de montrer pourquoi l'apprenant a fait une erreur. Pour ce faire, nous avons modélisé la méthode CREAM (*Cognitive Reliability and Error Analysis Method*). Nous avons intégré le mécanisme d'analyse rétrospective de CREAM pour que le système puisse indiquer les liens causaux les plus probables expliquant l'occurrence des erreurs. Pour plus de détails sur le modèle des erreurs le lecteur pourra se référer à (Trinh *et al.*, 2009).

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.

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.

4.2. *Agent pédagogique*

La situation pédagogique (section 4.1) 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. Il s'agit à présent de définir un modèle simulant le comportement décisionnel de l'agent pédagogique qui fournit les propositions d'assistance pédagogique, c'est-à-dire un modèle qui fait le lien entre les connaissances et les propositions d'assistances. Rappelons que nous nous plaçons dans le cadre de l'apprentissage de tâches procédurales et collaboratives. Nous devons considérer :

– l'hétérogénéité et la nature des connaissances impliquées (connaissances issues de la pédagogie fondamentale jusqu'à la réalité virtuelle) ;

– les capacités d'adaptation (le raisonnement doit s'automodifier pour prendre en compte les expériences passées) ;

– le raisonnement doit pouvoir être spécifié *a priori* (la spécification initiale peut alors être réalisée par un pédagogue).

Les critères qui en découlent sont : expressivité, hiérarchisation, modularité, réactivité et adaptabilité.

Après nous être intéressés aux familles d'architectures comportementales existantes (connexionistes, à base d'automates, à base de règles), nous avons opté pour celles qui utilisent des règles qui répondent au mieux aux critères précédents. Plus précisément, nous avons choisi les systèmes de classeurs (Buche *et al.*, 2006b). Il s'agit d'une architecture réactive et adaptative, fondée sur des règles conditionnelles.

Nous proposons 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 fondées 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.

La figure 9 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.

Simuler un raisonnement pédagogique présente un double intérêt :

1) les formateurs ne sont pas toujours des pédagogues, nous leur proposons alors une assistance pédagogique ;

2) les formateurs ne sont pas des experts dans les logiciels de simulation, l'agent pédagogique va proposer à l'apprenant des aides qui exploitent au mieux les capacités de la réalité virtuelle.

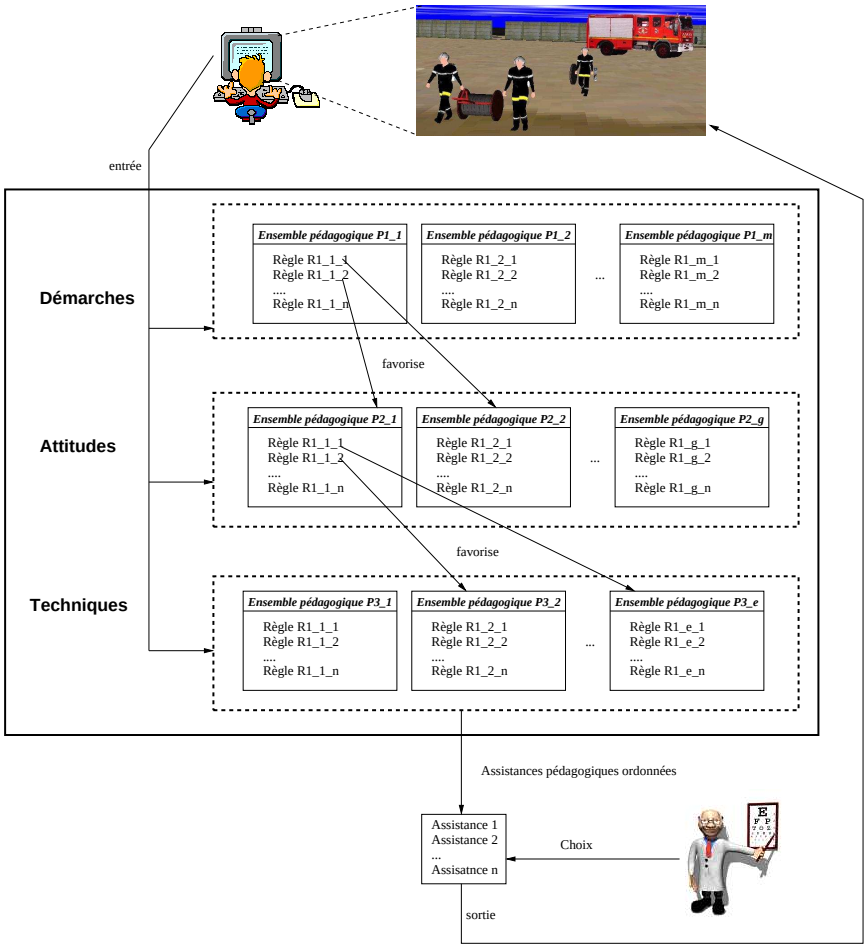

Figure 9. *Représentation complète du modèle pédagogique*

4.3. *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.

4.3.1. *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 3, 4 et 5 correspondant respectivement aux trois niveaux (démarches, attitudes et techniques). Ces informations représentent une possibilité de spécification des ensembles de règles des trois niveaux (cf. figure 10).

4.3.2. *Spécifier les règles pédagogiques*

Les ensembles de règles pédagogiques étant définis, 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.

Dans l'exemple suivant, nous nous plaçons au niveau d'abstraction `Démarches Pédagogiques`, nous avons un ensemble de règles appelé `Active`. Une première règle de cet ensemble est satisfaite si l'apprenant est novice (`Apprenant.Niveau==novice`), si il vient d'effectuer une erreur de type agencement (`Apprenant.Erreur.type==procédural`) et si l'action effectuée est différente de l'action correcte (`! Travail.ActionSollicitée in Travail.ActionCorrectes`). Dans ce cas, la règle favorise l'ensemble `Expliquer` du niveau qui suit.

```
if (Apprenant.Niveau == novice &&
    Apprenant.Erreur.type==procédural &&
    ! Travail.ActionSollicitée in Travail.ActionCorrectes)
then (Expliquer)
```

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 au 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 3. *Exemples de définition des ensembles pour le niveau d'abstraction « Démarches pédagogiques », en se basant sur (Lourdeaux, 2001; Burkhardt* et al.*, 2003)*

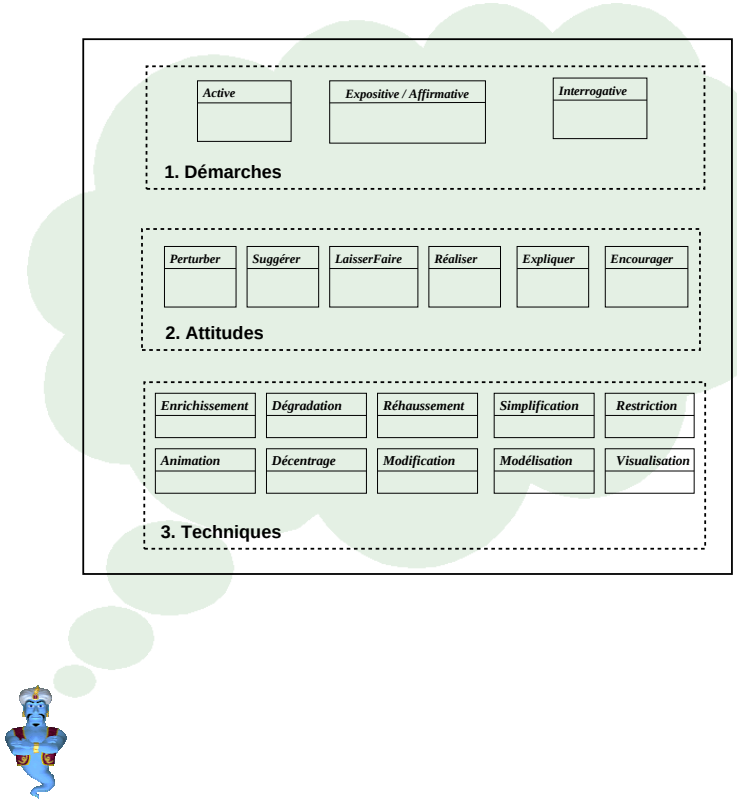

Figure 10. *Spécification des 3 niveaux du modèle de décision pédagogique*

4.3.3. *Utilisation*

Un modèle pédagogique particulier a été créé à partir de la structure précédemment décrite et des références (Lourdeaux, 2001; Burkhardt *et al.*, 2003). Les ensembles sont ceux décrits figure 10, chaque ensemble contient en moyenne 5 règles. Ce modèle pédagogique a été appliqué à deux EV distincts destinés à l'apprentissage de procédures collaboratives. Pour ces deux applications complétement différentes, mais qui portent sur le même type d'apprentissage, aucune modification du modèle pédagogique (ensembles et règles) n'est nécessaire. Dans le cas d'un autre type d'apprentissage (par exemple un travail pratique de sciences), ces règles seraient probablement à revoir. Cependant, nous faisons l'hypothèse que des modifications ne seraient à apporter que sur le niveau intermédiaire. Une étude est en cours dans laquelle les auteurs du modèle pédagogique sont des spécialistes des sciences de l'éducation.

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

4.4. *Apprentissage*

L'apprentissage va permettre de raffiner les poids des règles pour s'adapter aux habitudes du formateur et imiter ainsi son expertise.

L'algorithme d'apprentissage est inspiré du *Bucket Brigade* (Holland, 1986; Wilson, 1987). Ce dernier distribue les rétributions aux règles qui ont permis de les obtenir. Il est initialement adapté aux systèmes de classeurs (Buche *et al.*, 2006b) possédant une file de règles dont l'enchaînement produit une action. Dans notre cas, l'enchaînement correspond au passage entre les niveaux hiérarchiques. La rétribution est simplement reflétée par le choix du formateur : la technique pédagogique qu'il sélectionne définit les règles du niveau trois qui vont être rétribuées. Par chaînage arrière, les règles du niveaux 2 et 1 sont également rétribuées. Les règles qui s'apparient avec la *situation pédagogique*, mais qui participent à l'activation d'une autre technique pédagogique que celle choisie par le formateur, voient leur poids diminuer. L'algorithme effectue en fait un partage de la rétribution, incluant une taxe permettant de ne pas désavantager les règles rarement appariées et pénalisant plus intensément les règles fortes pour préserver l'adaptativité du système.

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-apprenant, prendre le relais temporairement et appliquer directement les assistances qu'il a choisies.

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

5. Discussion

Avant de conclure, il convient de discuter des apports de notre proposition. L'étude proposée dans cet article 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). Nous pensons qu'il est possible de répondre aux problèmes de généricité et de modularité du modèle pédagogique.

Sans revenir sur chaque point de nos travaux, nous pouvons montrer en quoi notre proposition répond aux difficultés des modèles existants. 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, « si l'apprenant peut observer l'avion 2 alors... », mais utilisera plutôt des connais-

sances 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, « faire clignoter le pompier », manipulent également des connaissances génériques indépendantes de l'exercice (« Faire clignoter les protagonistes 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 : apprentissage de procédures collaboratives sur porte-avions (GASPAR) (Marion *et al.*, 2007) et sur site SEVESO pour les pompiers (SÉCURÉVI) (Querrec *et al.*, 2004). 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). Enfin, notons que PEGASE est fondé sur la prise en compte du couple apprenant-formateur.

Ainsi, notre proposition apporte des solutions 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. Envisager ce type de formation impose de repenser les éléments qui sont étroitement liés à la notion de procédure, *i.e.* les connaissances de la situation pédagogique.

6. Bibliographie

Buche C., Un système tutoriel intelligent et adaptatif pour l'apprentissage de compétences en environnement virtuel de formation, Thèse de doctorat, Université de Bretagne Occidentale, Centre Européen de Réalité Virtuelle, novembre, 2005.

Buche C., Querrec R., Chevaillier P., Kermarrec G., « Apports des systèmes tutoriaux intelligents et de la réalité virtuelle à l'apprentissage de compétences », *In Cognito – Cahiers Romans de Sciences Cognitives*, vol. 2, n° 2, p. 51-83, 2006a.

Buche C., Septseault C., De Loor P., « Les systèmes de classeurs. Une présentation générale », *Revue des Sciences et Technologies de l'Information, série Techniques et Sciences Informatiques (RSTI–TSI)*, vol. 25, p. 963-990, Décembre, 2006b.

Burkhardt J. M., Lourdeaux D., Mellet d'Huart D., *Le traité de la réalité virtuelle*, deuxième edn, Les Presses de l'Ecole des Mines, chapitre La conception des environnements virtuels pour l'apprentissage, 2003.

Chevaillier P., Querrec R., Septseault C., « VEHA : Un méta-modèle d'environnement virtuel informé et structuré », *Revue des Sciences et Technologies de l'Information, série Techniques et Sciences Informatiques (RSTI-TSI)*, vol. 28, n° 6-7, p. 715 - 740, 2009.

Herviou D., Maisel E., « ARéViRoad : a virtual reality tool for traffic simulation », *Proceedings of Urban Transport*, p. 297-306, 2006.

Holland J. H., « Escaping Brittleness : The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems », *in* R. S. Michalski, J. G. Carbonell, T. M. Mitchell (eds), *Machine Learning : An Artificial Intelligence Approach*, vol. 2, Kaufmann, Los Altos, CA, p. 593-623, 1986.

Kallmann M., Thalmann D., « Modeling Objects for Interaction Tasks », *Proceedings of Computer Animation and Simulation'98*, p. 73-86, 1998.

Levesque P., « Creation and Use of 3D As-built Models at EDF », *FIG Working Week 2003*, 2003.

Lourdeaux D., Réalité virtuelle et formation : conception d'environnements virtuels pédagogiques, Thèse de doctorat, Ecole des mines de Paris, 2001.

Marion N., Septseault C., Boudinot A., Querrec R., « GASPAR : Aviation management on an aircraft carrier using virtual reality », *Cyberworlds 2007 proceedings*, p. 15-22, October, 2007.

Pomerol J., Brézillon P., « About Some Relationships between Knowledge and Context », *in* V. Akman, P. Bouquet, R. Thomason, R. A. Young (eds), *Modeling and Using Context : Third International and Interdisciplinary Conference, Context 2001*, Springer-Verlag, Berlin, p. 461-464, 2001.

Py D., « Quelques méthodes d'intelligence artificielle pour la modèlisation de l'élève », *Sciences et Techniques Educatives*, vol. 5, n° 2, p. 123-140, 1998.

Querrec R., Buche C., Maffre E., Chevaillier P., « SécuRéVi : virtual environments for fire-fighting training », *in* S. Richir, P. Richard, B. Taravel (eds), *5^{th} virtual reality international conference*, Laval, France, p. 169-175, 2003.

Querrec R., Buche C., Maffre E., Chevaillier P., « MultiAgents Systems for Virtual Environment for Training. Application to fire-fighting », *Special issue "Advanced Technology for Learning" of International Journal of Computers and Applications (IJCA)*, vol. 1, n° 1, p. 25-34, juin, 2004.

Richard J. F., *Les activités mentales : Comprendre, Raisonner, Trouver des solutions*, Armand Colin, Paris, 1990.

Rickel J., Johnson W. L., « Animated Agents for Procedural Training in Virtual Reality : Perception, Cognition, and Motor Control », *Applied Artificial Intelligence*, vol. 13, p. 343-382, 1999.

Trinh T., Buche C., Querrec R., Tisseau J., « Modeling of Errors Realized by a Human Learner in Virtual Environment for Training », *International Journal of Computers, Communications and Control*, vol. IV, n° 1, p. 73-81, March, 2009.

Turner R. M., « Context-sensitive reasoning for autonomous agents and cooperative distributed problem solving », *In Proceedings of the IJCAI Workshop on Using Knowledge in its Context*, p. 141-151, 1993.

Wenger E., *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann, Los Altos, California, 1987.

Wilson S. W., « Hierarchical Credit Allocation in a Classifier System », *10^{th} International Joint Conference on Artificial Intelligence (IJCAI'87)*, Milan, Italy, p. 217-220, 1987.

Article reçu le 10 juillet 2008
Accepté après révisions le 14 mai 2009

Cédric Buche *est maître de conférence à l'Ecole Nationale d'Ingénieurs de Brest (ENIB). Ses activités de recherche ont lieu au Centre Européen de Réalité Virtuelle (CERV). Elles*

concernent les environnements virtuels de formation et les comportements adaptatifs. Il est le fondateur de PEGASE.

Ronan Querrec *est maître de conférence à l'ENIB, membre du LISyC (Laboratoire Informatique des SYstèmes Complexes). Ses travaux de recherche ont lieu au CERV. Ils portent sur la réalisation de modèles d'environnements virtuels pour l'apprentissage humain. Il s'intéresse plus particulièrement à l'apprentissage de tâches procédurales et collaboratives.*

Pierre De Loor *est maître de conférence HDR à l'ENIB et il dirige l'équipe "Atelier de Réalité Virtuelle" (ARéVi) du CERV. Ses travaux portent sur les liens entre sciences cognitives, réalité virtuelle et intelligence artificielle. Il s'interesse particulièrement à la notion "d'autonomie dans l'interaction".*

Pierre Chevaillier *est maître de conférence HDR à l'ENIB, membre du LISyC, et il dirige le CERV. Ses travaux portent sur la conception d'environnements de réalité virtuelle pour l'apprentissage humain à travers le projet MASCARET. Il s'intéresse au caractère adaptatif et collaboratif du processus de formation en se fondant sur le paradigme des systèmes multi-agents.*

Jacques Tisseau *est Professeur des Universités en Informatique à l'ENIB. Directeur de l'ENIB et fondateur du CERV, ses recherches portent sur l'autonomisation des entités virtuelles, l'interaction avec ces entités autonomes et l'épistémologie de la réalité virtuelle.*