

# Un agent pédagogique pour les environnements virtuels de formation

Cyrille BAUDOUIN

CERV : Centre Européen de Réalité Virtuelle

EA3883 – LISYC

Laboratoire d'Informatique des SYstèmes Complexes

Rapport de stage de Master 2 Recherche Informatique IFSIC

Encadrant : Pierre CHEVAILLIER

12 juillet 2005



## Résumé

Ce rapport présente les travaux effectués au cours du stage de M2RI<sup>1</sup> intitulé : "Un agent pédagogique pour les environnements virtuels de formation" qui a pour objectif la réalisation d'un agent pédagogique indépendant s'intégrant dans un environnement virtuel de formation tel que ceux développés et utilisés au CERV. Cet agent utilise un modèle proche du modèle BDI et exploite l'architecture cognitive SOAR. Il possède plusieurs rôles pédagogiques -professeur, tuteur et entraîneur- qui orientent sa stratégie didactique. Il est capable d'adapter ses interventions au contexte et à l'apprenant.

MOTS-CLÉS : environnement virtuel de formation, *intelligent tutoring system*, réalité virtuelle, Système Multi-Agents, agent pédagogique, tuteur.

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Etat de l'art et enjeux</b>	<b>5</b>
2.1	Quelques Agents Pédagogiques . . . . .	5
2.2	Synthèse . . . . .	11
2.3	Enjeu . . . . .	12
<b>3</b>	<b>Modèle et méthode</b>	<b>13</b>
3.1	Représentation et exploitations des connaissances . . . . .	13
3.2	La stratégie pédagogique adaptative . . . . .	17
3.3	Modèle de l'agent . . . . .	23
3.4	SOAR : approche et méthode de fonctionnement . . . . .	24
3.5	L'utilisation de SOAR dans notre approche . . . . .	30
<b>4</b>	<b>Résultats</b>	<b>35</b>
4.1	Contexte et domaine d'apprentissage . . . . .	35
4.2	L'environnement réel . . . . .	35
4.3	L'environnement virtuel . . . . .	35
4.4	Implémentation de l'agent . . . . .	37
<b>5</b>	<b>Discussion</b>	<b>38</b>
5.1	Hypothèses cognitives et docimologiques . . . . .	38
5.2	Modélisation ITS . . . . .	39
5.3	Utilisation de Soar et maintenance des modèles ITS . . . . .	40
5.4	Rôles de l'agent . . . . .	40
5.5	Collaboration entre les agents pédagogiques . . . . .	41
<b>6</b>	<b>Conclusion</b>	<b>42</b>

---

<sup>1</sup>Master 2 Recherche en Informatique

<b>7</b>	<b>Annexe I : quelques agents pédagogiques</b>	<b>45</b>
7.1	Steve . . . . .	45
7.2	AutoTutor . . . . .	45
7.3	ADELE . . . . .	45
7.4	abits . . . . .	46
7.5	HAL . . . . .	46
7.6	Baghera . . . . .	46
7.7	EduAgent . . . . .	47
7.8	LuCy . . . . .	47
7.9	COSMO . . . . .	47
7.10	COLER . . . . .	48
7.11	Vincent . . . . .	48
7.12	Agneta and Frida . . . . .	48
<b>8</b>	<b>Annexe II : quelques rôles d’agents pédagogique</b>	<b>49</b>
<b>9</b>	<b>Annexe III : le TP de physique réel</b>	<b>50</b>
9.1	Résultats de l’étude . . . . .	50
9.2	Le plan de travail réel . . . . .	51
9.3	Consignes pour le Protocole 1 . . . . .	51
9.4	Consignes pour le Protocole 2 . . . . .	53
9.5	Consignes pour le Protocole 3 . . . . .	53
9.6	Consignes pour le protocole 4 . . . . .	55
<b>10</b>	<b>Annexe IV : Docimologie et erreurs d’évaluation</b>	<b>56</b>

## Table des figures

1	STEVE montre un voyant pendant une explication . . . . .	7
2	Fonctionnement global de HAL . . . . .	8
3	Architecture externe (à gauche) et interne (à droite) d’ABITS . . . . .	9
4	Exemple de graphe conceptuel sous ABITS : les Opérations Algébriques . . . . .	10
5	Les trois types d’éléments primaires du domaine . . . . .	13
6	Liens possibles entre les éléments primaires . . . . .	14
7	Diagramme d’activité : Activités générique et spécifiques . . . . .	17
8	Diagramme d’activité : ensure learn (stratégie globale) . . . . .	18
9	Choix d’un chemin : abstraction ou décomposition . . . . .	19
10	Diagramme d’activité : explain (explication) . . . . .	20
11	Schématisation d’un espace de problème . . . . .	25
12	Architecture d’exécution de SOAR . . . . .	25
13	Vue schématisée de la mémoire de travail . . . . .	26
14	Cycle d’exécution de SOAR . . . . .	27
15	Création d’un nouvel espace de travail suite à une impasse . . . . .	29
16	Environnement virtuel . . . . .	36
17	Le domaine <i>speed computation</i> exprimé la mémoire de travail . . . . .	37
18	Photo du TP de physique . . . . .	51
19	Vue schématisée de l’environnement . . . . .	52

# 1 Introduction

Ce stage de M2RI <sup>2</sup> a pour thème général les environnements interactifs d'apprentissage humain (EIAH) et plus particulièrement les agents pédagogiques. Un agent pédagogique est un agent informatique <sup>3</sup> spécifique, intégré dans un EIAH et impliqué dans le processus pédagogique. Il est souvent personnifié et interagit directement avec l'apprenant.

Il existe de nombreux agents pédagogiques dont la plupart sont développés dans le cadre de projets de recherche. Une étude bibliographique préalable a permis la mise en valeur de critères caractéristiques de ces agents (modèles d'agents, domaine d'apprentissage, rôles pédagogique...) grâce auxquels nous avons pu analyser et classer différents agents existants. En se basant sur ces critères nous élaborons une approche regroupant les points forts des travaux existants.

Nous nous fixons ici comme objectif la réalisation d'un agent pédagogique multi-rôle et indépendant du domaine et de la plate-forme d'apprentissage pouvant être intégrable dans un environnement virtuel de formation tel que MASCARET [24]. L'un des points important pour permettre et améliorer l'apprentissage est que l'agent adopte une bonne stratégie d'intervention. Le point central de notre recherche est donc de permettre une intervention plus souple et efficace que celles des agents existants. Nous proposons pour cela de mieux adapter cette intervention à l'apprenant et au contexte. Du point de vue pédagogique, notre agent se base sur une approche behavioriste de l'apprentissage de part la description des connaissances qui peuvent être décomposés en d'autres connaissances plus simples, en particulier pour les comportements. Nous avons intégré cet agent dans un environnement virtuel de formation en cours de réalisation au CERV, dont le domaine d'apprentissage est la mesure de la vitesse de la lumière dans le cadre de travaux-pratiques en Physique.

Dans la première partie, nous présentons un bref état de l'art et exposons plus en détail les enjeux du projet. Puis nous décrivons nos choix et orientations pour notre propre agent. Nous présentons ensuite nos résultats à travers l'intégration de l'agent dans un EVF. Enfin, nous proposons une discussion de ces résultats et des nouvelles perspectives de recherche.

---

<sup>2</sup>Master 2 Recherche en Informatique

<sup>3</sup>entité informatique autonome et communicante

## 2 Etat de l'art et enjeux

Les agents pédagogiques que nous avons étudiés sont principalement mis en oeuvre dans des Systèmes Tutoriels Intelligents (ITS)<sup>4</sup> ou dans des environnements virtuels de formation (EVF).

Les Environnements Virtuel de Formation (EVF) sont construits autour de la réalité virtuelle : ils immergent l'apprenant dans un environnement virtuel où sont reproduites les situations d'apprentissage. Les EVF peuvent avoir les mêmes fonctionnalités qu'un ITS (c'est le cas de STEVE par exemple, cf. 2.1.1).

### 2.1 Quelques Agents Pédagogiques

Etant donné le nombre d'agents existant, il semblait impossible de réaliser une étude exhaustive. Nous avons sélectionné un ensemble de douze agents représentatifs, au vu des critères que nous souhaitions mettre en valeur. L'annexe I présente brièvement chacun de ces agents.

Nous présentons ici plus en détail STEVE, HAL et ABITS qui sont les trois sources d'inspiration principales pour notre travail.

#### 2.1.1 steve

STEVE (Soar Training Expert for Virtual Environment) [25] est un agent autonome et animé qui évolue dans un environnement virtuel de formation : VET - *Virtual Environment for Training* (littéralement : environnement virtuel d'entraînement). Il a été conçu et développé par Jeff Rickel et W. Lewis Johnson à l'*Information Sciences Institute (University of Southern California)* en coopération avec le *Lockheed AI Center*. STEVE est utilisé pour la formation à la maintenance d'équipements navals complexes.

VET est implémenté en C, Tcl/Tk et en VRML (environnement 3D). STEVE est implémenté en C, VRML (avatar) et SOAR (cf. 3.4).

STEVE est basé sur un modèle d'agents cognitifs et est composé de trois modules exécutés en parallèle :

- le module de perception,
- le module de cognition,
- le module de contrôle moteur.

Le module de perception permet à STEVE d'être informé de tous les événements qui se déroulent dans l'environnement et de posséder à tout instant une représentation cohérente de l'état global de l'environnement (*snapshot*).

Le module de cognition, implémenté principalement sous forme de règles de production SOAR, interprète les perceptions (*snapshot* + liste d'événements récents), choisit des buts appropriés puis construit des plans pour atteindre ces buts. Il exécute ensuite les plans en envoyant les tâches à jouer au module de contrôle moteur.

Le module de contrôle moteur transforme les ordres de haut-niveau reçus par le module de cognition en tâches élémentaires exécutables par l'avatar (paroles, gestes, et

---

<sup>4</sup>Les *intelligent tutoring systems* sont basés sur les techniques de l'intelligence artificielle et ont pour principal objectif de simuler le formateur

déplacements). Pour pouvoir créer des gestes, le module de contrôle moteur utilise certaines informations données par les éléments de l'environnement :

- la position de l'objet,
- la sphère englobante de l'objet,
- le vecteur d'orientation qui indique la face avant de l'objet,
- le vecteur de prise qui définit la manière de prendre l'objet,
- le vecteur de pression qui définit la manière de presser l'objet,
- le vecteur de positionnement qui permet à l'avatar de se positionner devant l'objet sans gêner la vue de l'apprenant (optionnel).

Le déplacement de STEVE est géré sur un graphe dont chaque noeud est une position possible, c'est-à-dire la place indiquée par le vecteur de positionnement d'un élément (s'il n'a pas été défini, il est calculé automatiquement).

STEVE est un agent pédagogique qui agit de manière autonome (sans l'action du formateur pendant une séquence). Son rôle principal est celui de tuteur mais on peut aussi l'utiliser comme un simple élève (sans stratégie pédagogique associée) exécutant des ordres pour permettre à l'apprenant d'exécuter une action collaborative nécessitant une tierce personne. Par exemple, pour une simulation où il faut ouvrir deux vannes simultanément, on peut avoir un apprenant, un agent tuteur et un agent élève. Lorsque le tuteur demandera d'ouvrir les vannes, l'élève simulé se dirigera vers la sienne et l'ouvrira.

Les stratégies pédagogiques de STEVE sont davantage centrées sur l'action que sur les concepts puisqu'il s'agit d'un apprentissage de procédures. STEVE peut :

- expliquer une tâche,
- montrer comment faire en commentant,
- superviser l'élève et intervenir pour signaler ou pour prévenir erreurs,
- observer l'élève en le laissant faire des erreurs et en expliquant ensuite,
- répondre aux questions (simples).

L'aptitude qui nous semble la plus intéressante chez STEVE est sa capacité cognitive qui lui permet d'élaborer des plans d'actions (séquence arborescente de tâches) et d'expliquer ces plans (en parcourant l'arbre hiérarchique des tâches). Les créateurs de STEVE mettent également en avant deux autres aptitudes remarquables : sa représentation et sa capacité de communication. L'avatar de STEVE est un modèle 3D stylisé d'un homme (cf figure 1) qui permet de simuler de nombreuses actions et situations avec un rendu fluide et assez naturel.

Le personnage peut se déplacer, montrer des éléments de l'environnement, désigner des zones du doigt ou du regard, utiliser des actionneurs (appuyer sur un bouton) ; tout cela en temps réel et sans gêner l'apprenant dans sa tâche.

STEVE dispose d'un moteur d'analyse et de synthèse de langage naturel et vocal. Il peut donc comprendre quelques questions (prédéterminées) très simples à l'oral (pourquoi ? *-why ?-* et comment ? *-how ?-*), et des questions plus compliquées à l'écrit (pendant la simulation pour l'oral et hors simulation pour l'écrit). Il donne une réponse en parcourant son domaine d'apprentissage : en montant pour répondre à la question *-pourquoi ?-* et en descendant pour répondre à la question *-comment ?-*.

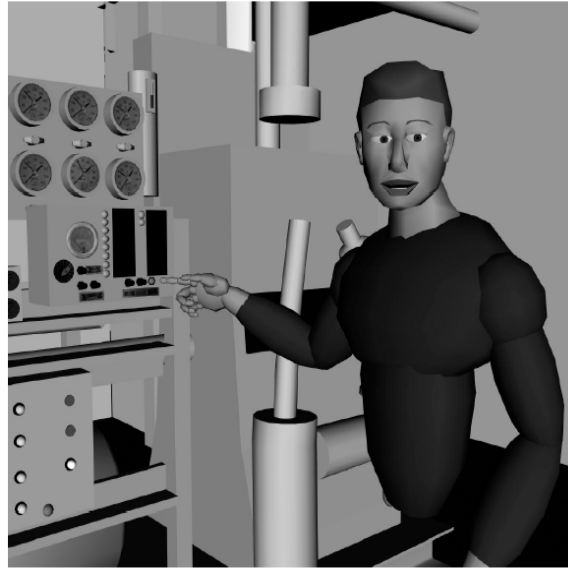


FIG. 1 – STEVE montre un voyant pendant une explication

### 2.1.2 hal

HAL [20] (*Help Agent for Learning*) a été développé dans le cadre du projet SOFI. SOFI (Simulateur Opérationnel de Formation Individuelle) est un environnement virtuel de formation qui immerge l'apprenant au coeur d'un simulateur d'entraînement destiné à former les agents de la SNCF aux tâches de maintenance en bordure de voie ferrée.

HAL est un agent cognitif et possède trois modèles :

1. le modèle du formé (modèle de l'apprenant),
2. le modèle de référence (modèle du domaine),
3. le modèle pédagogique.

Le modèle du formé contient des informations sur les niveaux de connaissances atteints et sur le profil de l'apprenant. Ce profil est utilisé par HAL mais doit être fabriqué manuellement (auto-évaluation par l'apprenant). Le modèle du formé utilise deux sous-modèles de représentation des connaissances de l'apprenant : le modèle d'expertise partiel (overlay) et le modèle différentiel.

Le modèle de référence contient les données du domaine (tâches procédurales) sous forme hiérarchique : une tâche peut être décomposée en d'autres tâches de niveau inférieur et il existe des relations entre les tâches d'un même niveau. Par exemple **ouvrir la porte** contiendra la tâche **prendre la clef** puis **insérer la clef dans la serrure** puis **tourner la clef**. Si l'apprenant tourne la clef avant de l'insérer dans la serrure, HAL détectera une erreur, avertira le formateur qui pourra intervenir (ou faire intervenir HAL).

Le modèle pédagogique associe les connaissances (et particulièrement les erreurs relatives) à des stratégies et des fonctions (activités) pédagogiques. HAL est alors capable de rajouter du sens dans l'environnement lorsque cela est nécessaire (décentrage, animation, enrichissement...).

HAL est un agent pédagogique non-personnifié et un assistant puissant pour le formateur. Il a le rôle d'un tuteur/gêneur/facilitateur mais n'est pas complètement autonome : ceci n'est pas dû à une quelconque limitation cognitive, mais au choix d'assister la formation sans "remplacer" le formateur.

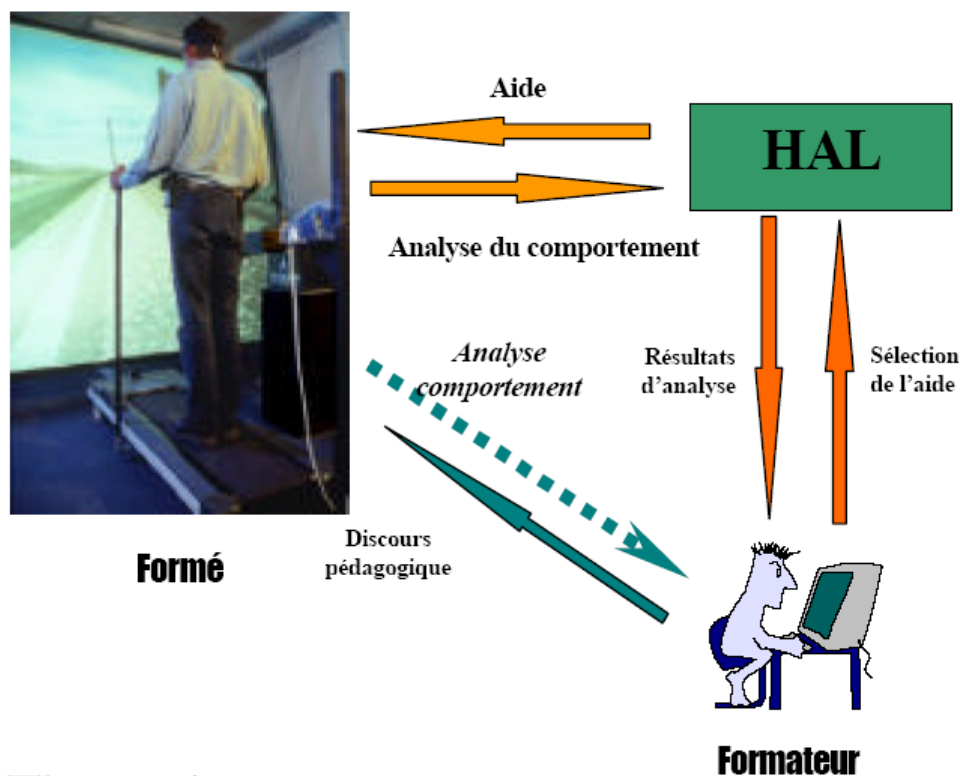


FIG. 2 – Fonctionnement global de HAL

HAL analyse le comportement de l'apprenant et propose des activités mais laisse le pouvoir décisionnel au formateur. HAL dispose de deux stratégies : une méthode active où le formé exécute les tâches et une méthode explicative où le formé reste passif. De plus HAL possède plusieurs niveaux d'intervention pour chaque mode. En mode actif il peut laisser faire le formé sans l'assister, lui dire quoi faire ou faire à sa place. De même, pour le mode explicatif, HAL pourra suggérer la connaissance, suggérer l'erreur ou expliquer. Selon le niveau de l'apprenant et les choix du formateur, HAL peut modifier son scénario ou l'environnement pour introduire des problèmes ou les inhiber.

L'apprenant dispose d'une interface immersive (écran géant, tapis roulant et gant muni de capteurs) et peut évoluer dans l'environnement en 3D (cf. figure 2). Dans la même pièce, le formateur observe directement les actions de l'élève. Il supervise l'apprentissage indirectement, lisant des comptes-rendus fournis par HAL, et en choisissant les stratégies pédagogiques.

### 2.1.3 abits

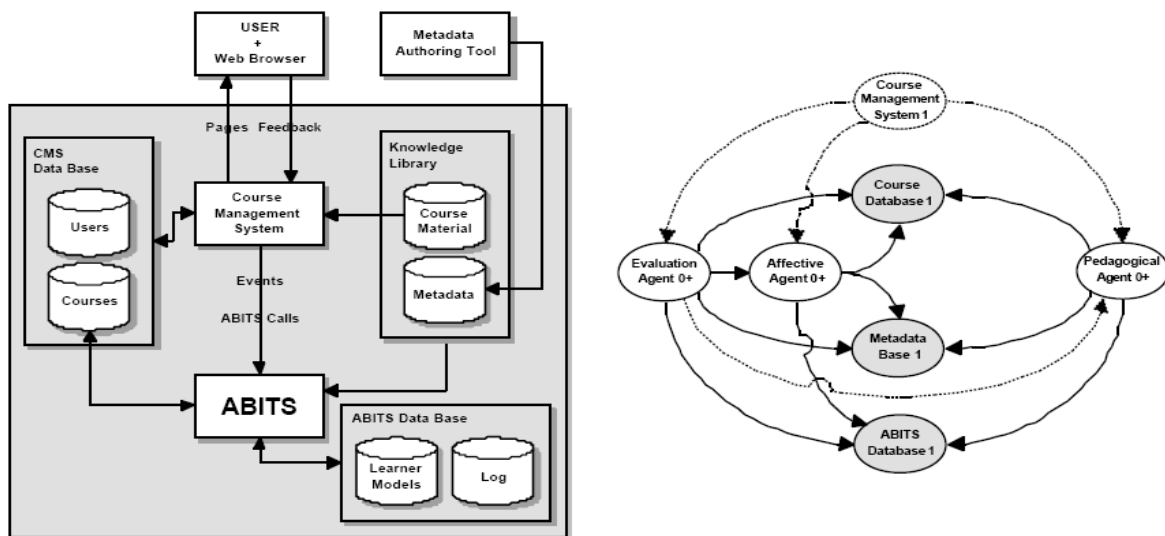
ABITS [7] (*Agent Based Intelligent tutoring system*) a été réalisé dans le cadre du projet *InTraSys* (*Intelligent Training System in Technical Assistance*). C'est un ITS basé sur une approche multi-agents, créé pour l'apprentissage à distance qui permet d'étendre un système de gestion de cours classique (*Course Management System* ou CMS).

ABITS respecte les normes IEEE LTSC sur les *Learning Object Metadata* (LOM) définissant les propriétés minimales nécessaires à la gestion, l'organisation et l'évaluation des ressources pédagogiques et des méta-data associées.

ABITS est constitué de trois ensembles d'agents fonctionnels (cf. figure 3 – droite) :

1. Les agents d'évaluation, responsables de l'évaluation de l'état cognitif de l'apprenant,
2. Les agents affectifs, qui évaluent le profil de l'apprenant,
3. Les agents pédagogiques, qui génèrent le programme d'étude.

La partie gauche de la figure 3 (vue externe) montre les interactions entre ABITS, le CMS et les apprenants.



#### Légende :

- *Course Material Database* : contient les ressources sous forme de fichiers délivrables sur le Web (HTML, VRML, etc.).
- *Metadata Base* : contient tous les schémas de meta-data (XML/RDF).
- *Log database* : contient la liste de toutes les activités de l'apprenant (pages visitées, temps, résultats des tests, etc.).
- *Learner Models Database* : contient l'état cognitif et le profil de l'apprenant.
- *User Database* : contient toutes les informations à propos des utilisateurs.
- *Courses Database* : contient les objectifs d'apprentissages, les cours et les programmes d'étude.

FIG. 3 – Architecture externe (à gauche) et interne (à droite) d'ABITS

Pour représenter les relations hiérarchiques entre les concepts du domaine, ABITS utilise des graphes conceptuels (cf. figure 4).

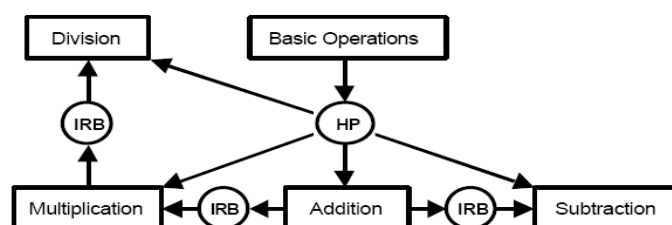


FIG. 4 – Exemple de graphe conceptuel sous ABITS : les Opérations Algébriques

La relation "IRB" (pour "Is Required By") définit un pré-requis tandis que la relation "HP" (pour "Has Part") définit une relation d'inclusion entre les concepts.

ABITS possède de nombreux agents fonctionnels qui permettent son fonctionnement interne mais il peut être vu comme un seul agent d'un point de vue extérieur. Il peut alors être vu comme un tuteur pour l'apprenant et un assistant personnel pour le formateur : ce dernier bénéficie d'une aide à la décision et peut automatiser certaines tâches. Par exemple, ABITS peut adapter le matériel didactique en fonction des préférences et des capacités de l'apprenant.

ABITS 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 connaissances atteints par l'apprenant. Un nombre flou (*fuzzy number*) est utilisé pour représenter le niveau dans chaque concept, ce qui permet plus de souplesse. ABITS applique par exemple une fonction d'oubli (*forgetting function*) 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 (*learning preferences*) décrites dans le tableau suivant :

Champ	Valeurs Possibles
Modalité	Texte, Image, Présentation, Hyper-texte, Vidéo, Simulation, Réalité virtuelle
Approche	Inductive, Déductive, Explorative
Niveau d'interaction	Très bas, Bas, Moyen, Haut, Très haut
Densité sémantique	Très basse, Basse, Moyenne, Haute, Très haute
Difficulté	Très basse, Basse, Moyenne, Haute, Très haute

TAB. 1 – Profil de l'apprenant dans ABITS

L'évaluation des préférences utilise également des nombres flous et est réalisée en mettant en rapport les degrés de connaissances lors d'étapes clefs (*milestones*). 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.

ABITS peut également générer un programme d'études (*curriculum*) spécifique à l'élève. Le programme d'étude est une suite ordonnée de ressources qui vont être utiles pour mener à bien un cours, c'est à dire assurer l'apprentissage de concepts clefs (*learning goals*).

La modularité et le respect des normes font d'ABITS un *framework* qui peut être adapté à de nombreux CMS<sup>5</sup>.

## 2.2 Synthèse

Nous avons pu constater la richesse des thèmes de recherche dans ce domaine mais également le manque de recoupement entre ces travaux. En effet, pour beaucoup d'équipes de recherche, le travail prospectif s'est concentré sur un thème précis tel que l'expression du domaine et la représentation des connaissances, l'interactivité et la communication avec l'apprenant, le travail collaboratif et l'apprentissage social.

Les agents ITS comme ABITS, HAL ou VINCENT<sup>6</sup> [21], nous donnent beaucoup de pistes différentes pour la formalisation, l'expression et l'exploitation des connaissances nécessaires à un agent pédagogique. L'expression du domaine dans STEVE et ABITS semble convenir partiellement à notre approche. La manière de représenter les connaissances sur l'apprenant dans ABITS en particulier semble être pertinente pour pouvoir adapter au mieux les activités didactiques : la prise en compte du profil et la dynamique du cursus<sup>7</sup> donne plus de souplesse qu'un modèle statique tel que celui de STEVE2.1.1.

En ce qui concerne l'aspect réalité virtuelle, les meilleurs agents sont sans conteste STEVE et COSMO<sup>8</sup> [17] dont l'incarnation assure une présence plus naturelle. Quant à HAL, il offre une bonne immersion de l'apprenant dans l'environnement, diminuant ainsi les biais d'apprentissage ajoutés par la réalité virtuelle. En effet, l'utilisateur reproduit des gestes proches de ceux qu'il devra produire en situation réelle, ce qui lui permet d'une part d'acquérir plus facilement des automatismes et d'autre part de favoriser l'action de transfert. Le transfert est l'action de réutiliser des connaissances apprises dans un contexte particulier dans un contexte différent [3]. Il est, pour les behavioristes comme pour les cognitivistes, l'un des indices nécessaires à l'évaluation de l'apprentissage.

Certains agents étudiés ici ont d'autres capacités intéressantes mais qui ne seront pas prises en compte pour notre agent, soit parce qu'elles ne sont pas applicables, soit parce qu'elles traitent d'un point non étudié ici. AUTOTUTOR<sup>9</sup> [12], par exemple, possède une maîtrise du langage naturel intéressante, mais son modèle sous-jacent ne semble pas convenir à notre approche : l'ensemble des connaissances est décrit par une matrice relationnelle entre des mots du langage et ne permet pas les traitements que notre approche nécessite.

Un autre système très intéressant est BAGHERA [26] (cf. 7.6) qui met en place une structure collaborative et coopérative entre des agents pédagogiques portant chacun un

---

<sup>5</sup>Pour être compatibles, ces derniers doivent seulement pouvoir être étendus par un langage de script supportant l'invocation RMI et pouvant accéder à des données externes

<sup>6</sup>VINCENT est un agent ITS dans un environnement basé-web. cf. 7.11

<sup>7</sup>ensemble des niveaux de connaissances et de compétences de l'apprenant

<sup>8</sup>COSMO est un agent mi-insecte mi-humain intégré dans un EVF. cf. 7.9

<sup>9</sup>AUTOTUTOR est un agent conversationnel qui simule un formateur. cf. 7.2

rôle, comme par exemple tuteur ou assistant personnel. Notre étude envisage cette collaboration mais ne prévoit pas pour le moment sa mise en place.

### 2.3 Enjeu

Notre but n'est pas ici la réalisation d'un agent qui conviendrait pour toutes les situations d'apprentissage ou tous les domaines. Il ne s'agit pas non plus de faire le plagia d'un agent existant. Nous voulons ici mettre en commun plusieurs principes que nous jugeons intéressants afin de voir s'ils peuvent cohabiter. Nous désirons un agent incarné capable d'interagir avec l'environnement virtuel et de communiquer avec l'apprenant tel que STEVE, et qui adapte ses interventions au contexte, au niveau de l'apprenant et à son profil (comme HAL et ABITS). Cet agent doit être générique vis à vis du domaine d'apprentissage. L'enjeu fixé est la conception d'un modèle d'agent permettant une meilleure adaptation à l'apprenant et au contexte, afin de rendre les interventions didactiques plus naturelles. Notre agent doit également pouvoir tenir plusieurs rôles prédéfinis qui appliqueront chacun une stratégie pédagogique différente.

## 3 Modèle et méthode

Nous présentons ici d'abord la méthode de représentation des connaissances. Puis nous décrirons la stratégie pédagogique employée et le modèle conçu pour notre agent. Nous présenterons ensuite SOAR et la manière dont nous l'utilisons.

### 3.1 Représentation et exploitations des connaissances

L'aptitude à enseigner efficacement découle directement de la manière de représenter et d'exploiter la base des connaissances (*Knowledge Base* ou KB) de l'agent. Elle repose donc sur les connaissances appartenant au domaine d'apprentissage et aux connaissances pédagogiques. Les travaux sur les Systèmes Tutoriels Intelligents (*ITS*) ont mis en évidence quatre aspects de modélisation<sup>10</sup> décrits par Woolf [27] : le modèle du domaine, le modèle pédagogique, le modèle de l'interface, le modèle de l'apprenant.

Nous préservons ici cette modélisation en quatre modèles car elle permet, entre autres choses, de bien séparer les connaissances et de faciliter la modifications de chaque modèle indépendamment des autres.

#### 3.1.1 Modèle du domaine

Ce modèle contient tous les éléments de connaissance sur l'environnement et sur le domaine d'apprentissage. Il est structuré sous forme de plusieurs sous-domaines, possédant chacun trois ensembles de données (cf. 5), dits éléments primaires :

- les concepts (*concepts*), qui sont les connaissances déclaratives et conceptuelles,
- les comportements (*behaviors*), qui représentent les savoir-faire liés au domaine (connaissances procédurales),
- les objets (*objects*), qui sont les éléments physiques dont l'observation et/ou la manipulation a un sens dans le domaine.

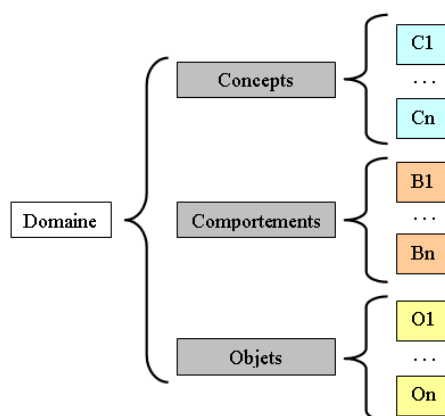


FIG. 5 – Les trois types d'éléments primaires du domaine

---

<sup>10</sup> Remarque : le découpage n'est pas toujours explicite et les modèles peuvent être présents sans pour autant être facilement dissociables.

Ces éléments primaires sont complétés par des éléments secondaires qui peuvent être des attributs ou des éléments pédagogiques. On peut par exemple ajouter une information sur la couleur d'un objet **cube** ou une primitive textuelle (de description, de consigne ou de question) au comportement **prendre un cube**.

Les éléments primaires se référencent entre eux par des liens unidirectionnels de différents types. Ces liens peuvent lier des éléments appartenant à des sous-domaines différents.

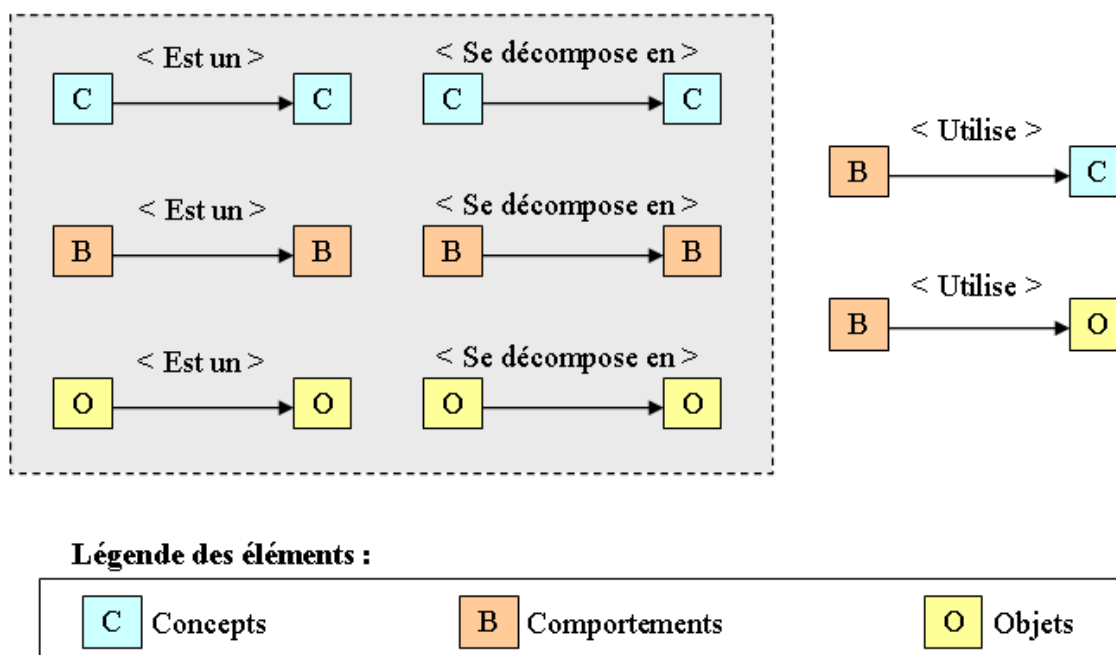


FIG. 6 – Liens possibles entre les éléments primaires

Comme l'illustre la figure 6, il existe trois types de liens :

- le lien **-Est un-** (*Kind-of*), qui lie toujours deux éléments de même type. Il exprime soit la notion d'instanciation entre deux éléments, par exemple

O -> [medor] -Est un-> [chien]

B -> [aller vers la porte] -Est un-> [aller vers un objet]

C -> [distance terre-lune] -Est un-> [distance]

soit la notion de classification par exemple

O -> [chien] -Est un-> [animal]

B -> [aller vers un objet] -Est un-> [se déplacer]

C -> [distance] -Est un-> [quantité]

- le lien **-Se décompose en-** (*Include*) montre l'inclusion, l'agrégation ou la composition d'éléments. Il lie toujours deux éléments de même type. Par exemple :

O -> [couteau] -Se décompose en-> [lame]

[couteau] -Se décompose en-> [manche]

B -> [aller dans le salon] -Se décompose en-> [aller vers la porte]

[aller dans le salon] -Se décompose en-> [ouvrir la porte]

- [aller dans le salon] -Se decompose en-> [entrer par la porte]
- C -> [vitesse] -Se decompose en-> [distance]
- [vitesse] -Se decompose en-> [durée]
- le lien **-Utilise-** (*Uses*) montre une relation d'utilisation entre un comportement et un objet ou concept. Par exemple :
  - B-0 -> [Jouer au football] -Utilise-> [terrain de football]
  - [Jouer au football] -Utilise-> [ballon]
  - B-C -> [prendre le pouls] -Utilise-> [pouls]
  - [prendre le pouls] -Utilise-> [durée]

### 3.1.2 Le modèle de l'apprenant

Le modèle de l'apprenant comporte d'une part un module de diagnostic qui permet de connaître l'état des connaissances de l'apprenant (par rapport au domaine) et qui représente ses objectifs courants et d'autre part un module comportemental qui décrit le profil de l'apprenant. Ce profil, qui décrit ses préférences pédagogiques et ses aptitudes cognitives, est généralement statique et n'évolue pas au cours d'une séance.

Pour notre modèle, les connaissances de l'apprenant sont décrites comme un sous-ensemble de celles du domaine. Le module de diagnostic référence chaque élément primaire du domaine et lui associe deux attributs.

1. D'une part, l'attribut **-présenté-** (*presented*) qui détermine le nombre de fois où l'apprenant a été confronté à cet élément dans l'environnement. Il est incrémenté par le modèle pédagogique lors des phases d'explication et de supervision (cf. 3.2). Cette notion de présentation est emprunté à l'ITS WEST [5].
2. D'autre part, l'attribut **-niveau-** (*level*) qui détermine, sur une échelle de quatre (-1= non-évalué, 0=novice, 1=débutant, 2=confirmé, 3=expert), le niveau correspondant à cet élément. Ce niveau est assimilable à un niveau de connaissance pour un concept, un niveau de savoir faire pour un comportement et un niveau de maîtrise pour un objet. Il est calculé par une évaluation (cf. 3.2.3).

Les éléments de comportement possèdent un troisième attribut, **-exercé-** (*exercised*), qui représente le nombre de fois où l'apprenant a réalisé cette tâche avec succès. Cet attribut est incrémenté à chaque évaluation positive (cf. 3.2.3).

Notre modèle d'apprenant possède également une information sur le profil de l'apprenant : sa capacité à conceptualiser. Elle doit être fournie avant les séances d'apprentissage et peut être obtenue grâce à des tests psychologiques sur l'apprenant tel que le *Passalong* [1].

### 3.1.3 Le modèle pédagogique

Ce modèle décrit les activités pédagogiques exécutables et simule les décisions pédagogiques du formateur. Le modèle pédagogique apporte la réponse aux trois questions suivantes :

- Quand et pourquoi intervenir ?
- Sur quel sujet intervenir ?

– Comment intervenir ?

Notre agent pédagogique utilise ce modèle pour adapter sa stratégie pédagogique en fonction de son rôle, du contexte et de l'apprenant : c'est à dire pour déterminer les activités didactiques qu'il va mettre en place. Concernant les interventions nous optons pour une méthode d'intervention dynamique, commandée par la stratégie pédagogique globale.

Pour notre agent, une intervention peut avoir pour objectif de donner une consigne, signaler une erreur potentielle, proposer une remédiation (explication), encourager l'apprenant, donner un conseil.

Le tableau donné en annexe II (cf. 8) présente quelques rôles mis en place dans les travaux de recherches existants et les activités pédagogiques qui leur sont associées. Nous avons, pour le moment, retenu trois rôles : professeur, tuteur et entraîneur.

Lorsqu'il a le rôle de professeur, l'agent présente des connaissances théoriques à l'apprenant sous la forme d'un cours organisé. Il ne prend pas en compte les interruptions potentielles de l'apprenant.

Le rôle d'entraîneur se base sur la théorie de l'apprentissage par renforcement (Behaviorisme [4]) et favorise l'apprentissage par la répétition et la mise en place de contraintes de performance.

L'agent tuteur est plus souple : quand il joue ce rôle, l'agent propose des travaux pratiques tout en donnant des explications comme remédiation aux erreurs détectées ou suite à une demande de l'apprenant.

Ces trois rôles sont matérialisés dans l'agent par la manière différente d'appliquer les activités pédagogiques. Lors d'une séance d'apprentissage, l'agent conserve l'un de ces trois rôles, et peut donc appliquer les activités tel que son rôle le définit.

Le modèle pédagogique est divisé en deux parties : d'une part la stratégie globale (assurer l'apprentissage) et les activités pédagogiques génériques, et d'autre part, les activités spécifiques aux différents rôles.

Notre agent possède trois activités didactiques principales :

1. l'explication (*explain subject*), qui génère des messages à caractère descriptif, explicatif ou démonstratif pour l'apprenant,
2. l'évaluation (*evaluate*), qui permet de mesurer le niveau de l'apprenant pour un élément primaire du domaine,
3. la supervision (*supervise*), qui permet d'accompagner l'apprenant (en l'évaluant) pendant qu'il tente de réaliser un comportement.

Ces activités sont atomiques et peuvent être exécutées indépendamment. En les combinant selon sa stratégie, l'agent exécute des activités pédagogiques plus complexes, comme - Enseigner- (*teach*) ou -Donner un cours- (*give course*). Comme le montre le diagramme 7, l'application de ces activités est générique ou spécifique à l'un des rôles.

D'autres actions didactiques secondaires sont incluses dans les activités principales comme la relance, lorsque l'agent détecte une attente trop longue ou l'encouragement qui vient en général ponctuer la supervision et compléter l'évaluation.

Nous détaillons la stratégie globale et quelques activités dans la partie suivante (cf. La stratégie pédagogique adaptative 3.2).

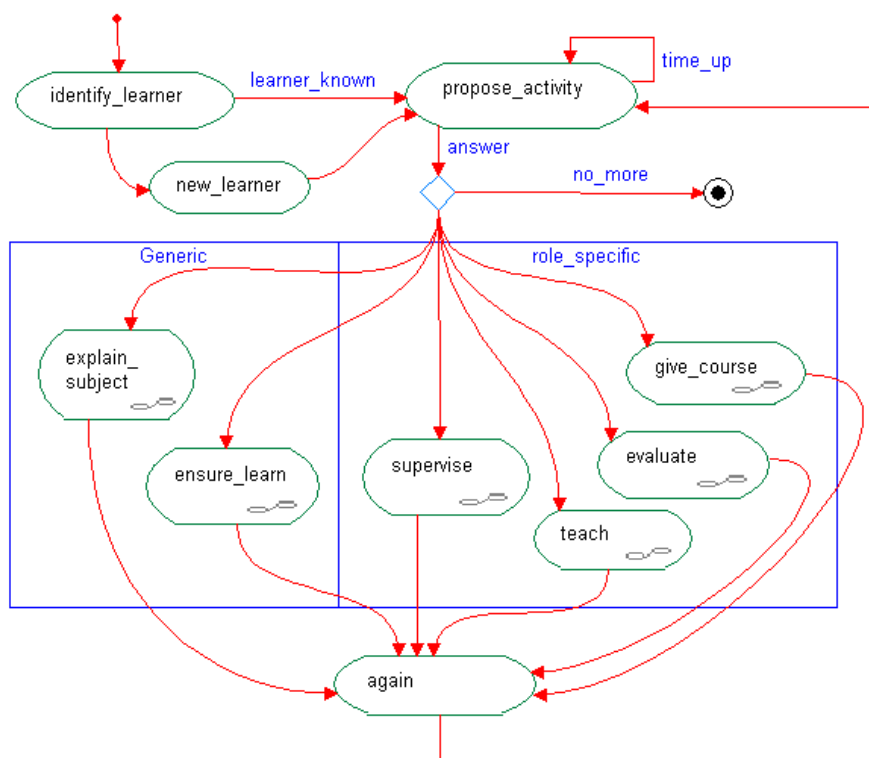


FIG. 7 – Diagramme d'activité : Activités générique et spécifiques

### 3.1.4 Le modèle de l'interface

Ce modèle régit l'interface entre le système et les utilisateurs (principalement l'apprenant). Il est responsable de la forme finale de toutes les communications vers l'apprenant et de la présentation des objets pédagogiques. Il est donc en charge de l'ergonomie de l'interface qu'il doit adapter en fonction de ses préférences (par exemple le choix de la modalité ou de la précision). Les interactions entre l'agent pédagogique et l'apprenant passent également par ce modèle, qui peut donc contenir les données nécessaires à la compréhension et la production de langage naturel, la reconnaissance vocale et la synthèse vocale ou la production de langage non-verbal (gestes, expressions du visage).

Notre agent ne prend pas en compte les différentes modalités et le modèle de l'interface n'a pas vraiment été utilisé ici. Il ne comporte que les éléments ayant trait au formatage et à la concaténation de primitives textuelle. Pour l'affichage du texte, l'agent se repose sur les ressources mises à disposition par l'environnement.

## 3.2 La stratégie pédagogique adaptative

La stratégie globale vise l'apprentissage d'un élément primaire du domaine. Cette stratégie est l'activité de plus haut niveau de notre agent : *ensure learn*. Le diagramme d'activité de la figure 8 montre l'algorithme utilisé pour réaliser ce comportement.

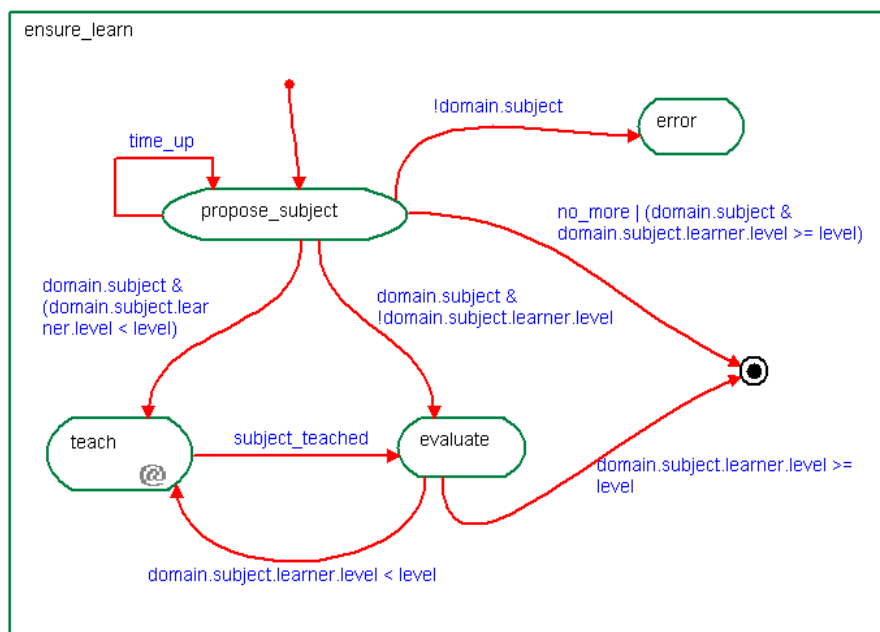


FIG. 8 – Diagramme d'activité : ensure learn (stratégie globale)

On cherche à vérifier que l'apprenant maîtrise bien l'élément visé, et, lorsque ce n'est pas le cas, on apporte une remédiation, grâce à l'activité *teach*), dont la forme dépend du rôle de l'agent.

L'entraîneur concentre son action sur l'apprentissage des comportements. Pour faire apprendre un comportement il fait apprendre les sous-comportements inclus. Pour cela, il supervise chaque sous-comportement en imposant une contrainte de performance grandissante puis il demande à l'apprenant de réaliser l'ensemble de la tâche. Pour l'apprentissage d'un objet ou d'un concept, l'agent entraîneur parcourt le domaine à la recherche de comportements utilisant l'objet ou le concept visé, puis fait apprendre ces comportements. Ce type de recherche peut aboutir à une explosion combinatoire : dans le cas où le nombre de comportements retournés dépasse cinq, l'agent sélectionne un échantillon de cinq comportements en favorisant les comportements non-présentés puis ceux où le niveau est le plus faible (sélection probabiliste).

Le professeur lui est plus orienté sur les données conceptuelles et présente comme remédiation un cours magistral (activité *give course*) sur l'élément visé. Cette activité correspond à l'enchaînement de l'activité d'explication *explain subject* sans intervention de l'apprenant (sauf pour terminer le cours).

Pour un agent tuteur, cette remédiation consiste premièrement à donner une explication sur l'élément courant, puis, faire apprendre (*ensure learn*) un élément lié à l'élément courant. Une activité secondaire permet le choix d'un élément primaire : **-choisir sujet-**

### 3.2.1 Le choix d'élément

L'activité **-choisir sujet-** (*choose subject*) permet de choisir un élément primaire en fonction du contexte, du profil et des niveaux de l'apprenant. Cette activité secondaire

est utilisée dans l'activité d'explication (*explain subject* et dans l'activité de remédiation et d'enseignement *teach*).

L'objectif est de choisir l'un des éléments liés à l'élément courant par l'un des deux liens : **-Est un-** ou **-Se décompose en-**.

Lorsque les deux types de liens sont présent, tel que le montre la figure 9, l'agent sélectionne et conserve tous les liens d'un type. Ce choix repose sur l'utilisation du niveau de conceptualisation de l'apprenant (profil). Si l'apprenant conceptualise facilement, il est plus pertinent de choisir le lien d'abstraction **-Est un-**, sinon, il vaut mieux choisir le lien **-Se décompose en-** qui pointe vers un sous-élément.

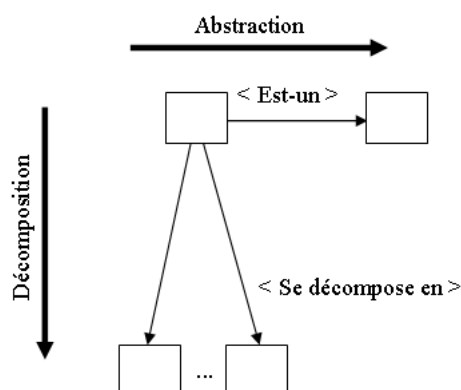


FIG. 9 – Choix d'un chemin : abstraction ou décomposition

Une fois l'un des deux type de lien choisis ou lorsque seul l'un des deux types de lien est présent, l'agent sélectionne un élément parmi les éléments qui n'ont pas encore été présentés à l'apprenant (attribut **-présenté-**). Si tous les éléments ont été présentés, il choisit en priorité l'élément dont le niveau est le plus faible.

### 3.2.2 L'explication

Les consignes et les questions sont générés simplement par l'assemblage de la consigne en elle même et de la description des éléments liés. Par exemple, si on démarre un exercice pratique où l'apprenant doit mettre en oeuvre le comportement intitulé `lancer_balle`, l'agent assemble la primitive de consigne `Vous devez [elem]` et la description associée à l'élément comportemental `lancer la balle`.

Les explications qui sont adaptées au contexte, au profil et au niveaux de l'apprenant sont générées par l'activité `explain subject`

Le diagramme d'activité de la figure 10 montre l'algorithme utilisé.

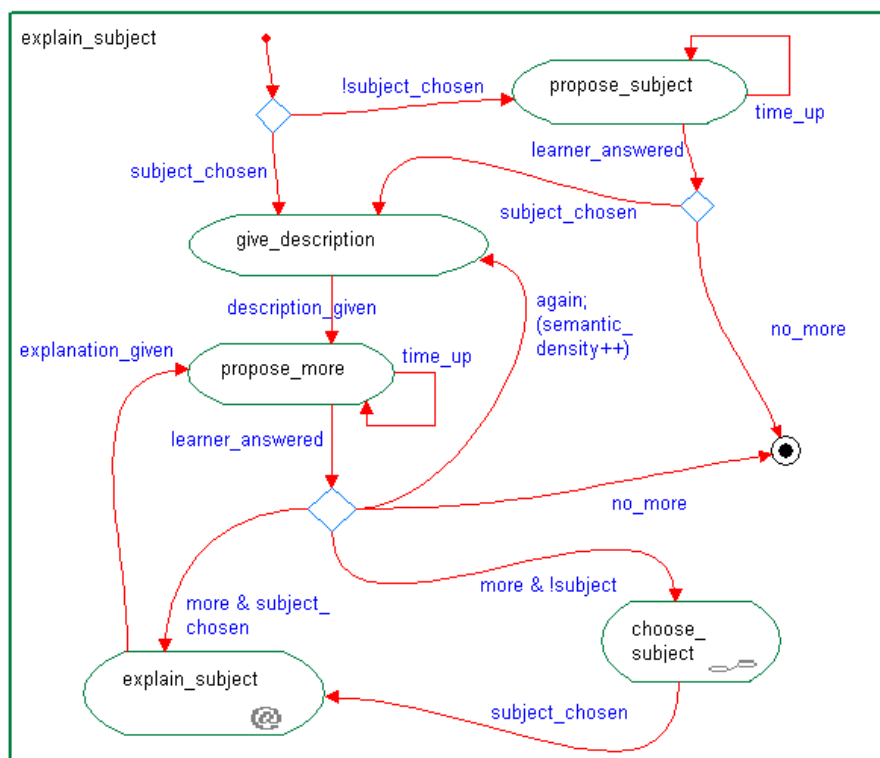


FIG. 10 – Diagramme d'activité : explain (explication)

L'explication est donnée pas à pas à l'apprenant : chaque élément étant décrit sous forme textuelle. Après chaque description, l'apprenant est invité à intervenir : il peut alors demander soit un rejeu de l'explication (reformulée avec plus de détail) *-again-*, soit des précisions sur un point précis de l'explication (c'est à dire un élément du domaine) *-subject-*, soit un complément d'informations *-more-*. Dans le second cas, l'agent débute une nouvelle phase d'explication en partant de l'élément primaire du domaine sélectionné par l'apprenant, dans le troisième cas, c'est l'agent qui va déterminer le pas suivant. Il va alors choisir l'élément racine de l'explication suivante en appliquant l'activité secondaire *-Choisir sujet-*.

### 3.2.3 L'évaluation

Cette activité permet d'évaluer le niveau de l'apprenant sur un élément primaire donné. Les attributs "niveaux" décrits dans le modèle de l'apprenant sont modifiés au cours de la simulation grâce aux évaluations.

Le modèle de l'élève est modifié en comparant la performance de l'apprenant avec celle que l'expert aurait produite dans les mêmes circonstances. [10]. Pour les comportements, on évalue le déroulement et le résultat (s'il est mesurable dans la simulation) d'un exercice pratique. Cette méthode s'apparente au **Model Tracing** dans l'ITS LISP ([2]).

Pour les concepts, on évalue une série de réponses à un QCM (Questionnaire à Choix Multiples). Ces évaluations sont soit prédéfinies (écrite à l'avance pour un élément) soit

composés grâce à d'autres évaluations. Par exemple s'il existe des liens de dépendances (uses) entre un concept et plusieurs comportements mais pas d'évaluation prédéfinie pour ce concept, on peut déduire un niveau de connaissance pour ce concept en sommant les résultats des évaluations des comportements qui en dépendent.

### 3.2.4 La supervision

L'activité de supervision (*supervise*) consiste à observer l'agent pendant qu'il exécute un comportement. L'agent reconnaît des comportements grâce au module de perception (assemblage d'actions primitives) et vérifie à chaque comportement exécuté (reconnu ou non) qu'il est valide dans le contexte de la tâche. Il utilise pour cela des informations de contraintes associées à chaque comportement dans le domaine. Les contraintes informent sur l'ordre d'exécution de groupes de comportements parmi les comportements liés à la tâche par le lien -se décompose en-.

Le tableau suivant (2) montre les choix pédagogiques effectués par un agent tuteur et un agent entraîneur dans le cas de la supervision d'un exercice pratique. Le but de l'exercice pour l'apprenant est de réaliser une tâche 1, composé de trois comportements A, B, C, dont l'ordre d'exécution n'a pas d'importance. Un comportement D est également possible dans cette situation.

Tâche réalisées par l'apprenant	Intervention de l'agent tuteur	Intervention de l'agent entraîneur
premier cas		
A	"Vous devez réaliser la Tâche 1"	"Vous devez réaliser la Tâche 1"
B	-	-
C	"C'est bien, vous avez réussi l'exercice"	"C'est bien, vous avez réussi l'exercice"
second cas		
B	"Vous devez réaliser la Tâche 1"	"Vous devez réaliser la Tâche 1 en 1 min"
-	-	-
-	<b>"Désirez-vous un rappel de la consigne"</b>	<b>"Désirez-vous un rappel de la consigne"</b>
réponse "oui"	"Vous devez réaliser la Tâche 1"	"Vous devez réaliser la Tâche 1 en 1 min"
C	-	-
A	"C'est bien, vous avez réussi l'exercice"	"C'est bien, vous avez réussi l'exercice"
troisième cas		
B	"Vous devez réaliser la Tâche 1"	"Vous devez réaliser la Tâche 1"
D	<b>"Cette action ne semble pas nécessaire, voulez vous continuer?"</b>	<b>"Cette action est incorrecte. Nous reprenons l'exercice au départ"</b>
A	-	-
C	"C'est bien, vous avez réussi l'exercice"	"C'est bien, vous avez réussi l'exercice"
B	-	-

TAB. 2 – Exemple de scénario avec un agent tuteur ou un agent entraîneur

L'entraîneur impose une contrainte de temps globale sur la supervision. Si le temps se termine avant que l'apprenant n'ait achevé la tâche, l'agent considère la tâche comme échouée et reprend l'exercice au départ.

### 3.3 Modèle de l'agent

Au regard des besoins, nous décidons de créer un agent basé sur un modèle cognitif capable d'exploiter tous les modèles cités. Nous faisons le choix d'un modèle inspiré du modèle *Belief-Desire-Intention* (BDI) [23].

Les croyances (*Belief*) d'un agent sont les informations que l'agent possède sur l'environnement et sur d'autres agents qui existent dans le même environnement. Les croyances peuvent être incorrectes, incomplètes ou incertaines et, à cause de cela, elles sont différentes des connaissances de l'agent, qui sont des informations toujours vraies. Les croyances peuvent changer au fur et à mesure que l'agent recueille plus d'information, par sa capacité de perception ou par l'interaction avec d'autres agents.

Les désirs (*Desire*) d'un agent représentent les états de l'environnement, et parfois de lui-même, que l'agent aimerait voir réalisés. Un agent peut avoir des désirs contradictoires. Le sous-ensemble cohérent des désirs réalisable est parfois assimilé aux buts de l'agent. A chaque désir correspond un objectif (l'état attendu) et un plan générique pour atteindre cet objectif.

Les intentions (*Intentions*) représentent les plans d'actions concrets que l'agent fabrique et met en place pour atteindre les buts (en général un seul but) qu'il a sélectionné.

Dans notre cas, les désirs de l'agent sont toujours de favoriser l'apprentissage d'un certain élément du domaine. Pour arriver à satisfaire ce désir, l'agent va mettre en oeuvre des activités pédagogiques, en fonction de ses croyances sur l'environnement et sur l'apprenant. L'agent ne fabrique pas à l'avance des plans d'actions : il respecte un plan d'action générique et ré-évalue ses intentions à chaque cycle de décision afin de garder une meilleure réactivité face au changement du monde et au comportement imprédictible de l'apprenant.

L'utilisation de ce modèle permet de simuler le comportement global d'un formateur tout en restant réactif pour permettre une adaptation en temps réel à l'apprenant. L'agent possède une architecture interne modulaire composée de trois modules, similaires à ceux de STEVE 2.1.1 : le module de décision, le module de perception, le module d'action.

Notre agent exécute un comportement cyclique qui alterne l'exécution des 3 modules :

- la phase de perception pendant laquelle les croyances de l'agent sont mises à jours,
- la phase de décision qui permet l'élaboration d'états internes et la sélection d'actions.
- la phase d'action pendant laquelle l'agent commande l'exécution des actions sélectionnées.

Le module de perception qui récupère les informations dans l'environnement et les envoie au module de décision. Il permet de percevoir le monde passivement ou activement. La perception active répond à une action observationnelle de l'agent, comme par exemple l'inspection d'un objet du monde ou la lecture d'une valeur. La perception passive est la mise à jour constante des informations sur le monde, elle correspond à une perception humaine inconsciente.

Le module d'action exécute les actions commandées par le module décisionnel. Il possède les informations permettant la réalisation d'actions primitives (par exemple **prendre**) et peut obtenir des informations sur le monde (par exemple **la position de l'objet** qui doit être pris).

Le module principal est le module de décision qui analyse la situation et sélectionne les actions à effectuer. Il possède nécessairement un algorithme permettant l'exploitation des modèles du domaine, de l'apprenant et pédagogique. De plus, la description des connaissances décrites dans les différents modèles nécessite un langage et une syntaxe particulière qui permet cette exploitation. Nous avons donc choisi d'utiliser SOAR [19], [18]<sup>11</sup>, à la fois comme moteur cognitif de notre agent, mais aussi pour l'expression des différents modèles ITS (domaine, apprenant, pédagogique). Les informations liées à l'interface, aux géométries et aux primitives textuelles sont décrites dans des fichiers XML<sup>12</sup> externes.

### 3.4 Soar : approche et méthode de fonctionnement

SOAR est une architecture cognitive générale basée sur un modèle humain, développé à l'université de Carmellon. C'est un moteur de règle de production dont on peut résumer l'approche en une phrase : tout comportement visant à réaliser un but peut être exécuté par l'application d'opérateurs sur un état de l'environnement. SOAR permet également l'apprentissage par une méthode de *chunk*. L'objectif premier des concepteurs de SOAR était de créer une architecture capable de résoudre des problèmes d'assez haut niveau. Aujourd'hui, SOAR est utilisé pour créer des agents ayant un comportement humain le plus crédible possible. Plusieurs applications différentes tel que TacAirSoar [15] ou Quake-Soar [16] [14] ont prouvé l'efficacité de SOAR, en particulier pour la résolution de problème de haut-niveau et l'exécution de tâches en temps réel (avec ou sans planification).

TacAirSoar est une simulation de combat aérien dont les pilotes sont des agents SOAR capables de planifier leur plan d'actions et de communiquer en pseudo langage naturel. Ils sont également capables de s'adapter aux conditions dynamiques de l'environnement et de collaborer.

Quake-Soar est un joueur virtuel pour les jeux Quake 2 et Quake 3<sup>13</sup>, il est capable de découvrir et d'exploiter un niveau, d'évaluer ses adversaires et d'élaborer des stratégies d'attaque et de défense en temps réel.

#### 3.4.1 Opérateurs et espace de problème

Chaque problème donne lieu à la mise en place d'un espace de problème (*problem space*) qui décrit le but à atteindre sous la forme de conditions à vérifier (sur l'état du monde ou sur des prédicats). La figure 11 montre un espace de problème, quelques états possibles du monde et les opérateurs à appliquer. Ces états du monde ne sont pas décrits explicitement dans SOAR ou dans l'agent mais peuvent être déduits : ils représentent en fait la combinaison d'un ensemble de valeurs ou de conditions donnant une représentation simplifiée du monde.

---

<sup>11</sup>voir aussi [15], [16], [14], et [25]

<sup>12</sup>eXtensible Markup Language

<sup>13</sup>jeux de tir à la première personne ou *First-Person-Shooter* (FPS)

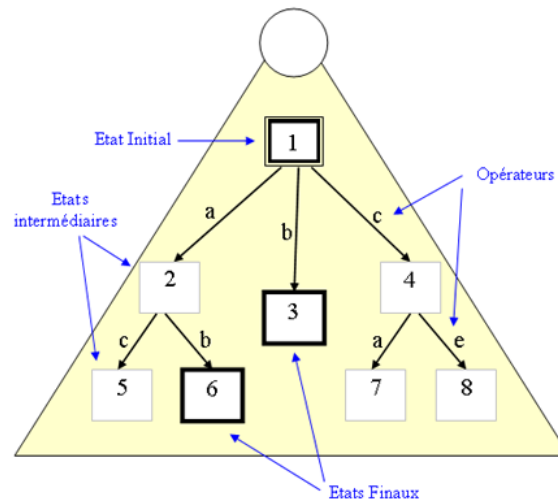


FIG. 11 – Schématisation d'un espace de problème

Les opérateurs ont deux rôles, d'une part faire évoluer les représentations internes de l'agent (croyances et intentions) et d'autre part commander des actions réelles sur l'environnement pour permettre son évolution.

### 3.4.2 Architecture de SOAR

Comme le montre la figure 12 la mémoire d'exécution est divisée en deux parties : d'une part la mémoire de production formée par le "programme" (règles de production) et d'autre part, la mémoire de travail qui contient les éléments liés au contexte courant.

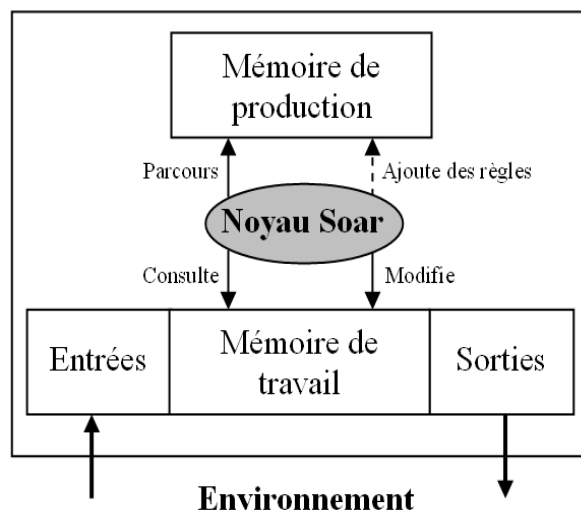


FIG. 12 – Architecture d'exécution de SOAR

Le noyau de SOAR utilise les règles contenues dans la mémoire de production pour instancier les opérateurs et les appliquer. Pour cela il consulte et modifie les éléments

contenus dans la mémoire de travail. Le noyau SOAR est également doté d'un mécanisme d'apprentissage qui permet de créer de nouvelles règles (cf. 3.4.7).

### 3.4.3 La mémoire de travail

La mémoire de travail contient les éléments courants de l'espace de problème, c'est à dire les différents éléments (*Working Memory Element* ou WME) sur lesquels vont porter les conditions des règles. Ces informations sont stockées sous forme d'un graphe orienté (cf. fig 13) dont la racine est l'état initial. Chaque élément, représenté par un disque sur la figure, possède des liens appelés "augmentations" (flèches) qui peuvent pointer vers un autre élément ou vers une constante, élément terminal sans augmentation (représenté par un rectangle).

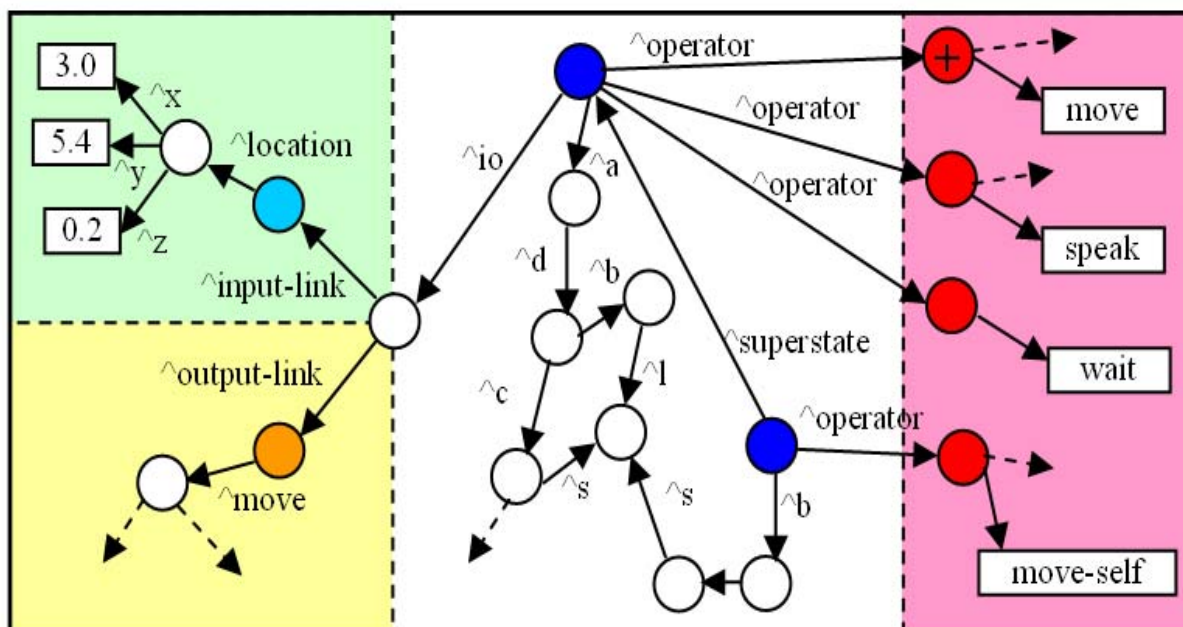


FIG. 13 – Vue schématisée de la mémoire de travail

Les éléments peuvent avoir un type spécial, qui leur donne des propriétés différentes des éléments normaux. C'est par exemple le cas pour les éléments "états", les éléments "opérateurs" ou les éléments d'entrées/sorties qui sont liés à l'état principal par les augmentations  $\hat{io}$ ,  $\hat{input-link}$  et  $\hat{output-link}$ . Les états sont des éléments représentatifs d'un espace de problème. SOAR ajoute par exemple un nouveau sous-état lors de la création d'une impasse (cf. 3.4.6). Les opérateurs sont des éléments qui ont deux niveaux de présence dans la mémoire : ils peuvent être proposés et l'un d'entre eux peut être sélectionné. Une condition peut porter sur un opérateur proposé ou sur l'opérateur sélectionné.

### 3.4.4 La mémoire de production et les règles

Cette mémoire constitue, d'après les concepteurs de SOAR la "mémoire à long terme" de l'agent. Elle contient le "programme" SOAR qui est essentiellement formé de règles de production. Ces règles peuvent être soit chargées par l'utilisateur (avant ou pendant l'exécution), soit créées automatiquement par SOAR (apprentissage). Chaque règle de production est de la forme "si conditions alors actions" et comporte la même syntaxe dont voici une description simplifiée.

1. le Mot clef "sp" (pour "SOAR *Production*"),
2. Une accolade ouvrante ("{"
3. Un nom de règle,
4. Une liste non-vide de conditions,
5. le symbole "-- >" (pour "then"),
6. Une liste non-vide d'actions,
7. Une accolade fermante ("}").

Les règles peuvent avoir plusieurs rôles :

- la proposition de nouveaux opérateurs en fonction de l'état interne et des perceptions,
- la comparaison entre plusieurs opérateurs proposés et l'ajout de préférences,
- l'application d'un opérateur,
- l'élaboration de nouveaux éléments dans la mémoire de travail.

### 3.4.5 Le cycle d'exécution

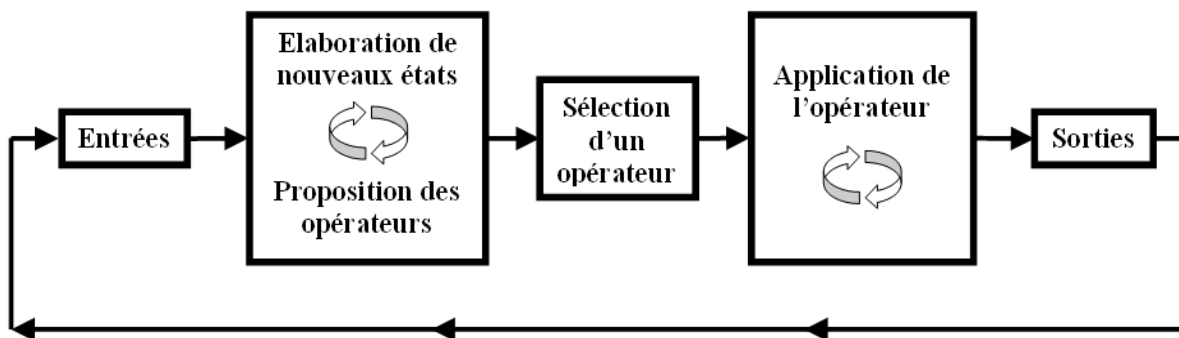


FIG. 14 – Cycle d'exécution de SOAR

La première phase du cycle correspond à la lecture des informations d'entrée fournies à l'agent. Lors de la seconde phase, les règles dont les conditions sont valides par rapport au contexte courant et aux nouvelles entrées sont tirées parallèlement, ce qui peut permettre la création de nouveaux WME, qui peuvent être des opérateurs (ils sont alors créés et proposés). Cette seconde phase est itérative et sera répétée tant qu'une règle est tirable.

Voici par exemple une règle permettant la proposition d'un opérateur d'attente dans le cas où aucun choix n'a été fait.

```
sp {top-ps*propose*wait
  (state <s> ^attribute state
  ^choices none
  -^operator.name wait)
-->
  (<s> ^operator <o> +,=)
(<o> ^name wait)
}
}
```

C'est durant cette seconde phase que les règles appliquant des préférences aux opérateurs sont tirées. La règle suivante applique une préférence (inférieure) sur l'opérateur d'attente lorsqu'il est proposé. En langage SOAR cela signifie que cet opérateur ne sera sélectionné que s'il est seul ou qu'un autre opérateur a une préférence explicite plus faible.

```
sp {top-ps*prefer*all*operator*to-wait
  (state <s> ^name spag
  ^operator <op>)
  (<op> ^name wait)
-->
  (<s> ^operator <op> < )
}
```

La phase suivante est très simple : le noyau SOAR compare toutes les préférences sur les opérateurs proposés et sélectionne celui qui possède la meilleure préférence. En cas d'égalité de préférences, un opérateur sera sélectionné aléatoirement, si et seulement si les opérateurs possèdent (entre autres) la préférence "indifférente", représentée par le signe "=". Dans le cas contraire, SOAR ne peut sélectionner d'opérateur et provoque une impasse (cf. 3.4.6). La règle ci-dessous correspond à l'application de l'opérateur d'attente précédent :

```
sp {top-ps*apply*wait*random
  (state <s> ^operator <o>)
  (<o> ^name wait)
-->
  (<o> ^random elaboration)
  (write (crlf) |spag : I am waiting - status complete|)
}
```

La quatrième phase est la phase d'application de l'opérateur sélectionné. Durant cette phase, les règles d'application correspondant à l'opérateur sélectionné seront tirées en parallèle. Comme la seconde phase, cette phase est itérative et continuera tant qu'une règle est tirable.

Enfin, la dernière phase donne lieu à l'écriture des sorties, ce qui permettra éventuellement de répercuter les changements sur le monde.

### 3.4.6 Les impasses

Lorsque la sélection d'un opérateur est impossible, Soar se trouve dans une **impasse**. Il existe quatre sortes d'impasse :

1. l'impasse d'égalité (*tie*) : elle survient quand Soar ne peut pas choisir d'opérateur car les préférences sont égales et ne permettent pas de trancher.
2. l'impasse de conflit (*conflict*) : cette impasse est levée lorsque deux opérateurs ou plus sont meilleurs entre eux (A meilleur que B et B meilleur que A) et qu'il ne sont pas dominé par un autre opérateur.
3. l'impasse d'échec de contrainte (*constraint-failure*) : elle survient lorsque les préférences sont incohérentes.
4. l'impasse d'inactivité (*no-change*) : l'impasse d'inactivité est levée lorsque que la phase de proposition se termine (plus de règle tirable) sans qu'un opérateur sélectionnable soit proposé.

Chaque impasse entraîne la création d'un sous-état et d'un nouvel espace de problème. Ce phénomène permet la décomposition des problèmes en sous-problèmes plus simples à résoudre. La figure suivante 15 présente ce mécanisme : un agent a pour but de s'échapper mais il rencontre une échelle, il n'a ici pas de préférence entre l'opérateur qui le ferait monter et celui qui le ferait descendre. Une impasse d'égalité est levée et un nouvel espace de problème est créé.

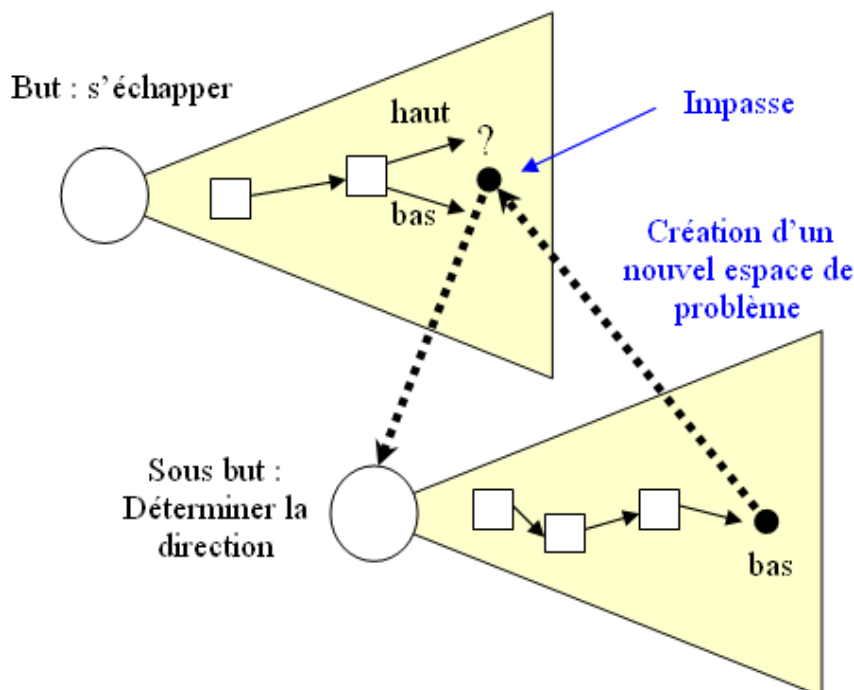


FIG. 15 – Création d'un nouvel espace de travail suite à une impasse

Lorsque le nouvel espace de problème ne peut être résolu, il lève une nouvelle impasse, qui crée un troisième espace de problème, et ainsi de suite, ce qui peut causer une avalanche d'impasses et de nouveaux états.

### 3.4.7 L'apprentissage

SOAR possède un mécanisme interne d'apprentissage : le "*chunking*". Cette méthode crée des nouvelles règles de productions appelées "*chunks*" pendant l'exécution. Un *chunk* résume le processus nécessaire à la résolution d'un sous-espace de problème. Les *chunks* sont créés automatiquement lorsqu'un sous-problème est résolu, permettant ainsi "d'apprendre" un comportement et de l'appliquer directement lorsque un sous-problème exactement équivalent survient. Cette fonction d'apprentissage consomme des ressources et peut-être désactivée si elle n'est pas nécessaire afin de gagner en performance. Un troisième mode est possible où la création des *chunks* est limitée à la résolution de buts terminaux.

## 3.5 L'utilisation de Soar dans notre approche

### 3.5.1 Implémentation des modèles ITS en Soar

Les connaissances sont représentées en grande partie dans la mémoire de production (domaine, pédagogique) et en partie moindre dans la mémoire de travail (apprenant). Les **connaissances procédurales** sont simples à mettre en oeuvre puisqu'elles sont cohérentes avec la vue buts-opérateurs de SOAR. La représentation des connaissances déclaratives reste également peu complexe, mais il faut néanmoins garder à l'esprit que ces informations doivent pouvoir être utilisées par l'agent.

Le domaine d'apprentissage est codé sous la forme de règles de production associées à un opérateur d'initialisation. Ces règles ne sont pas génériques et dépendent du domaine. Elles correspondent à l'application de l'opérateur `init-domain` et sont toujours de la forme :

```
sp {apply*initialize-domain*domain-name
  (state <s> ^operator <op>)
  (<op> ^name init_domain
    ^domain_name domain-name)
-->
  (write |init domain : domain-name| (crlf))
  (<s> ^domain <dom>)
  (<dom> ^name domain-name)
  (<dom> ^behaviors <behav>
    ^objects <objs>
    ^concepts <conc>)
#-----objets-----
(<objs> ^concept ...)
#-----concepts-----
(<conc> ^concept ...)
```

```
#-----behaviors-----  
(<behav> ^concept ...)  
}
```

Nous présentons un exemple de règle d'initialisation du domaine dans la partie Résultats 4.3.1.

Pour le modèle de l'apprenant, seul le profil est connu à la première séance : il est donné dans une règle d'initialisation semblable à la règle précédente. Voici la règle permettant l'initialisation du profil d'un apprenant nommé John.

```
sp {apply*initialize-learner*john  
  (state <s> ^operator <op>)  
  (<op> ^name init_learner  
    ^learner_name john)  
-->  
  (write |init learner : john| (crlf))  
  (<s> ^learner <lear>)  
  (<lear> ^name john  
    ^profile.conceptualisator 40)  
}
```

Les autres informations, c'est à dire les références sur les éléments du domaine et les attributs, sont initialisées par la règle suivante à leur valeur par défaut. Cette règle va être tirée autant de fois qu'il existe d'éléments primaire dans le domaine.

```
sp {apply*initialize-learner_level*john  
  (state <s> ^operator <op>  
    ^learner <lear>)  
  (<lear> ^name john  
    ^elements <elems>)  
  (<delem> ^type primary_element)  
  (<elems> -^element.domain-elem <delem>)  
  (<op> ^name init_learner_level  
    ^learner_name john)  
-->  
  (<elems> ^element <el>)  
  (<el> ^domain-elem <delem>)  
  (<el> ^presented 0  
    ^level -1)  
}
```

Ces attributs seront modifiés par les règles pédagogiques au cours de la séance. Le modèle de l'apprenant ne peut pas être sauvegardé pour le moment : pour ne pas perdre les données recueillies durant la séance, il est tout de même possible de sauver l'intégralité de la mémoire de travail pour pouvoir recharger l'agent dans le même état.

Le modèle pédagogique est implémenté sous forme de règle de production et utilise le concept d'opérateur.

### 3.5.2 Expressions des opérateurs pédagogiques

Les activités présentées précédemment sont en général représentée par un opérateur SOAR. Par exemple l'activité **Faire apprendre** est réalisé par l'opérateur *ensure learn*. La règle suivante permet la sélection de l'opérateur lorsque cette activité est commandée sur l'entrée de l'agent et que l'élément visé (<elem-name>) existe dans le domaine.

```
sp {propose*ensure-learn*on_demand
  (state <s> ^io.input-link.command <cmd>)
  (<cmd> ^name |ensure learn|
    ^subject <elem-name>
    ^level <level>)
  (<pelem> ^type primary_element ^name <elem-name>)
-->
  (<s> ^operator <o> +, =)
  (<o> ^name ensure_learn
    ^element <pelem>
    ^level <level>
    ^op-type pedagogical)
}
```

Cet opérateur ne possède pas de règles d'application : si SOAR sélectionne cet opérateur, il ne peut pas l'appliquer et produit une impasse de type *operator no-change* (cf. 3.4.6). On considère donc *ensure learn* comme un super-opérateur puisqu'il provoque la création d'un sous-problème qui sera résolu lorsque l'activité prendra fin, c'est à dire lorsque le niveau de l'apprenant sur l'élément visé aura atteint le niveau requis. Un nouvel état est donc créé et nous allons l'utiliser pour résoudre le sous-problème, c'est à dire réaliser l'activité faire apprendre *ensure learn*. La règle générique suivante nomme les sous-états pour permettre la sélection des sous-opérateurs.

```
sp {elaborate*state*name
  (state <s> ^superstate.operator.name <name>)
-->
  (<s> ^problem-space-name <name>)
}
```

On obtient alors un sous-état nommé *ensure learn* qui va permettre l'application d'opérateur tel que *evaluate* ou *teach*. Voici par exemple la règle proposant l'opérateur *teach* quand l'agent à le rôle de tuteur.

```
sp {tutor*propose*teach
  (state <s> ^name ensure_learn)
  ^superstate.operator.element <elem>
  ^superstate.operator.level <require-level>
  ^current-learner <learner>)
  (<learner> ^element <delem>)
```

```
(<delem> ^domain-elem <elem>)
(<delem> ^level <level> <> -1)
(<delem> ^level <level> < <require-level>)
-->
(<s> ^operator <o> +, =)
(<o> ^name teach
    ^element <elem>
    ^op-type pedagogical)
}
```

Cet opérateur est proposé si l'agent à connaissance du niveau de l'apprenant pour cet élément (<level> < <require-level>) et si le niveau requis par le super opérateur *ensure learn* n'est pas atteint (<level> < <require-level>). Si l'agent n'a pas connaissance du niveau, c'est l'opérateur *evaluate* qui est proposé. L'opérateur *teach* est lui aussi un super-opérateur : il ne possède pas de règle d'application et il est réalisé par des sous-opérateurs.

Lors de l'activité d'explication (cf. 3.2.2), si l'apprenant demande plus d'information mais ne précise pas de sujet, les deux règles suivantes vont permettre de proposer une explication pour tous les éléments liés par un lien -Est un- ou -Se décompose en-.

```
sp {tutor*propose*sub-explain*kind-of
    (state <s> ^name explain
        ^superstate.operator.element <elem>
        ^description_given
        ^semantic_density 2
        ^more
        -^subject_given)
    (<elem> ^kind-of <selem>)
-->
(<s> ^operator <o> +, =)
(<o> ^name explain
    ^link |kind-of|
    ^element <selem>
    ^op-type pedagogical)
}
```

```
sp {tutor*propose*sub-explain*include
    (state <s> ^name explain
        ^superstate.operator.element <elem>
        ^description_given
        ^semantic_density 2
        ^more
        -^subject_given)
    (<elem> ^include <selem>)
-->
```

```
(<s> ^operator <o> +, =)
(<o> ^name explain
  ^link |include|
  ^element <selem>
  ^op-type pedagogical)
}
```

L'activité -choisir sujet- ne correspond pas à un l'application d'un opérateur mais utilise la méthode de sélection d'opérateur en SOAR. Le niveau de conceptualisation est donné sur une échelle de 0 à 100 et va pondérer le choix du lien à suivre, ce qui est réalisé par la règle de préférence suivante :

```
sp {tutor*prefer*sub-explain
  (state <s> ^name explain
    ^operator <o1> +
    ^operator <o2> +
    ^superstate.learner.profile.conceptualisator <prob>)
  (<o1> ^name explain
    ^link |kind-of|)
  (<o2> ^name explain
    ^link |include|)
-->
  (<s> ^operator <o1> +, = <prob>)
  (<s> ^operator <o2> +, = ( - 100 <prob>))
}
```

Une autre règle de préférence favorisera les éléments qui n'ont pas été présenté puis pour lesquels l'apprenant à un faible niveau.

## 4 Résultats

### 4.1 Contexte et domaine d'apprentissage

Afin de tester notre agent nous l'avons intégré dans un environnement virtuel de formation. L'EVF choisi est un environnement dédié à la manipulation de matériel de physique dans le cadre de séances de travaux pratiques. L'implémentation actuelle concerne un TP de physique sur le thème de la vitesse de la lumière dans différents matériaux.

Cet environnement est développé au CERV dans le cadre d'un autre projet, dont l'un des objectifs est la comparaison entre une séance réelle de travaux pratique et une séance identique dans un environnement virtuel.

Nous avons choisi cet environnement car Beney et Guignard [3] ont déjà utilisé ce domaine au cours d'une expérience d'apprentissage, dans le cadre d'une étude portant l'efficacité des différents modes de guidage. Pour cette étude, présentée brièvement en Annexe III, ils ont mis en place :

- des tests cognitifs pour classer les apprenants,
- une analyse des comportements pertinents vis à vis de domaine,
- des protocoles de guidages (et des consignes),
- un protocole d'évaluation quantitatif.

### 4.2 L'environnement réel

L'apprenant peut manipuler des objets translucides et les placer sur le chemin d'un laser, entre sa source et un miroir, qui renvoie le rayon vers un récepteur (cf. annexe III figure 19). Un oscilloscope branché sur le système lumineux permet la visualisation de deux courbes, la première correspondant à la lumière à la sortie de la source et l'autre à l'entrée du récepteur. Le déphasage entre ces deux courbes permet le calcul du temps mis par la lumière pour parcourir le chemin. Une règle graduée est placée entre les miroirs et le dispositif lumineux (source+récepteur), permettant ainsi de mesurer les distances entre les objets. Lors de cette expérience, l'oscilloscope est calibré à l'avance et l'apprenant ne doit pas manipuler les réglages de ce dernier. Il peut tout de même jouer sur un paramètre pour aider l'étude des courbes : le dispositif lumineux possède un bouton déphaseur, qui décale la courbe correspondant au récepteur.

### 4.3 L'environnement virtuel

L'environnement virtuel 3D présente ce matériel de travaux pratiques sur une table au centre d'une pièce, comme le montre l'image de la figure 16.

Les interactions sont limitées pour minimiser la phase d'adaptation à l'environnement virtuel. Les miroirs sont mobiles sur un axe et peuvent être translattés le long du banc. L'apprenant peut cliquer sur un élément (cube, tube d'eau ou aquarium) pour le placer sur le banc. Il peut ensuite déplacer cet objet dans l'axe du banc.

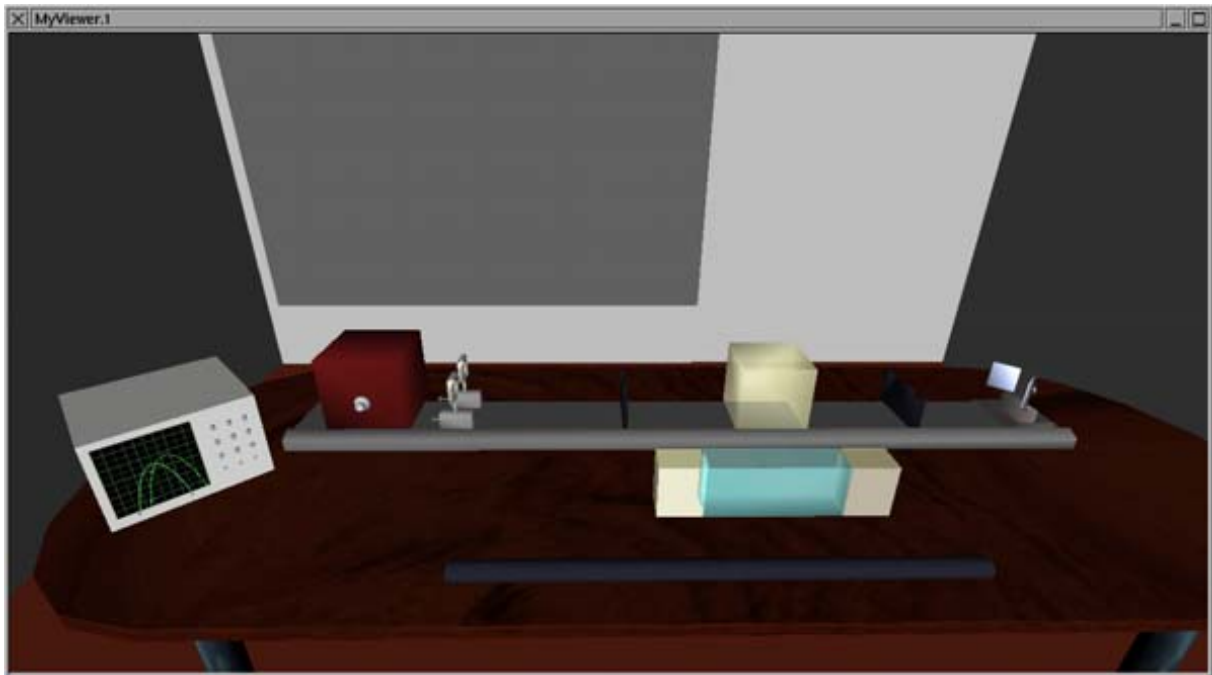


FIG. 16 – Environnement virtuel

#### 4.3.1 Modèle du domaine

Le modèle du domaine est décrit par des règles de production SOAR (cf. 3.5.1). Voici une règle qui permet la création en mémoire de travail d'un sous domaine concernant le calcul de la vitesse (*speed-computation*). Cette règle correspond à l'application d'un opérateur `init-domain` (portant le nom du domaine) et va ajouter des éléments dans la mémoire de travail (cf. figure 17). Les éléments sont créés les uns à la suite des autres en leur donnant une variable (par exemple `<speed>`) pour pouvoir les référencer. On ajoute également à chaque élément primaire le type `primary-element` qui permettra aux règles pédagogiques de les retrouver directement.

```
sp {apply*initialize-domain*speed_computation
  (state <s> ^operator <op>)
  (<op> ^name init_domain
    ^domain_name speed_computation)
-->
  (write |init domain : speed_computation| (crlf))
  (<s> ^domain <dom>)
  (<dom> ^name speed_computation)
  (<dom> ^behaviors <behaviors>
    ^objects <objs>
    ^concepts <conc>)
  #-----objects-----
  #-----concepts-----
  (<conc> ^concept <speed> <duration> <distance>)
```

```

(<speed> ^name speed
  ^include <duration>
  ^include <distance>
  ^type primary-element)
(<duration> ^name duration
  ^type primary-element)
(<distance> ^name distance
  ^type primary-element)
#-----
(<behaviors> ^behavior <compute_speed>)
(<compute_speed> ^name compute_speed
  ^uses <speed>
  ^uses <duration>
  ^uses <distance>
  ^type primary-element)
}
    
```

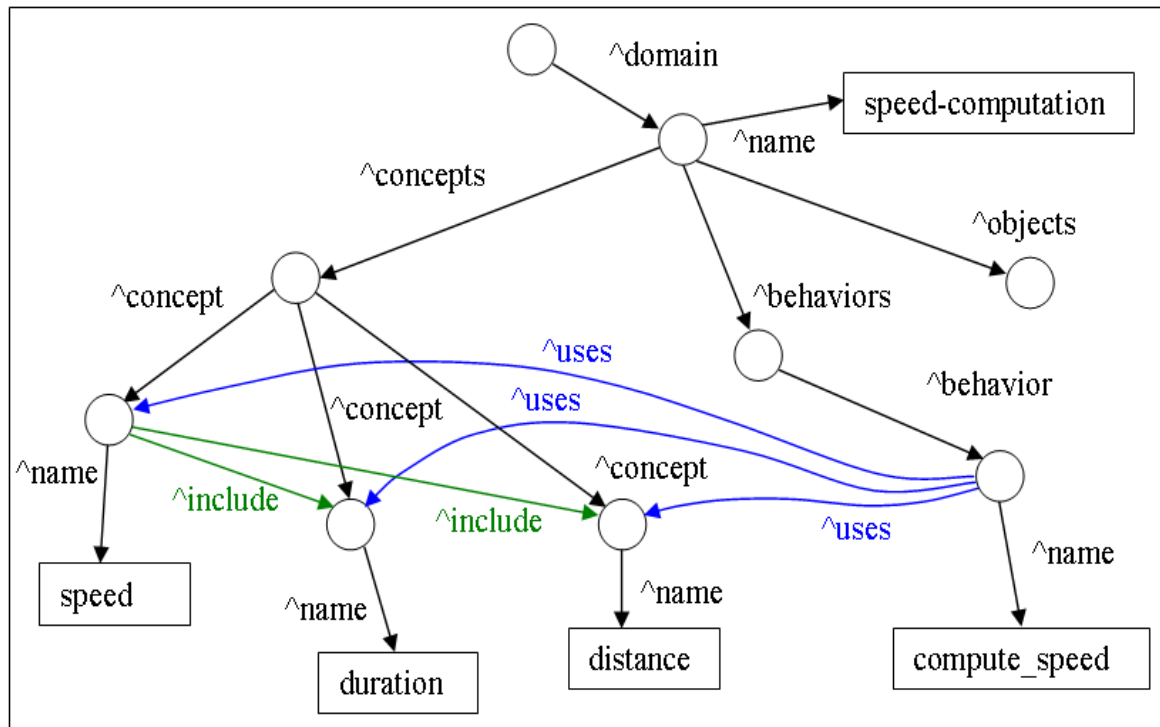


FIG. 17 – Le domaine *speed computation* exprimé la mémoire de travail

#### 4.4 Implémentation de l'agent

L'implémentation de l'agent a été réalisé dans les langages SOAR (module de décision) et C++ (modules de perception et d'action). L'interface avec l'environnement utilise la

bibliothèque **ARéVi**.

Le module de décision est exprimé par les règles de production correspondant au modèle pédagogique (cf. 3.5.2).

Si la sauvegarde des **connaissances épisodiques** <sup>14</sup> concernant les propres actions de l'agent est également assez simple, ce n'est pas le cas des situations extérieures qui ne dépendent pas de l'agent. Le module de perception permet de percevoir les actions élémentaires effectuées dans le monde et de créer ou mettre à jour des éléments sur le lien d'entrée (`^io.input-link`). Dans notre cas, certaines actions ne peuvent être perçues par l'agent : par exemple lorsque l'apprenant regarde l'oscilloscope et mesure le décalage, ou lorsqu'il réalise un calcul mental. C'est le formateur qui doit manuellement informer l'agent lorsqu'il perçoit ces actions en envoyant une commande de simulation à l'agent (ce dernier considérera alors avoir perçu ces actions de lui même).

## 5 Discussion

### 5.1 Hypothèses cognitives et docimologiques

Nous faisons ici l'hypothèse que l'apprentissage d'un savoir-faire "composé" d'autres savoir-faire est dépendant de l'apprentissage de ces derniers. Cette hypothèse behavioriste semble convenir dans des cas simples mais se révèle réductrice pour le cas général. Les facultés d'adaptation d'un correcteur humain sont grandes par rapport à celle de notre agent : ce dernier n'est pas capable d'évaluer un comportement qu'il ne connaît pas et qui n'est pas la composition de comportements connus. En effet, un certain comportement n'utilisant pas les mêmes comportements subordonnés est considéré comme incorrect dans un contexte ou il déclenche tout de même le résultat attendu. Ce problème nous oblige à spécifier le plus grand nombre possible de comportements (y compris les comportements erronés) pour éviter les échecs de la procédure d'évaluation. Cela renvoie au problème de l'évaluation des comportements : comment reconnaître exactement un comportement ? Et plus difficile encore, lorsqu'aucun comportement n'est reconnu, comment reconnaître un comportement équivalent ? S'il est possible de raisonner sur la perception des actions élémentaires pour déduire la perception de comportements composés simples (du point de vue du nombre d'actions élémentaires et/ou de leur enchaînement), il nous semble nécessaire de raisonner aussi sur les effets attendus et constatés pour résoudre ces problèmes sur des comportements complexes.

L'autre problème que l'on peut généralement soulever à propos de l'évaluation est sa validité. Elle est mise en défaut par plusieurs phénomènes déjà identifiés en situations d'apprentissage réelles dans les études docimologiques <sup>15</sup> telles que les effets présentés en Annexe IV 10. Ces phénomènes disparaissent presque tous lorsque l'évaluation est automatisée (sauf l'effet de flou), puisqu'ils sont principalement liés à des défauts comportementaux humains : le manque d'objectivité, la fatigue... En contrepartie, l'évaluation automatique des performances de l'apprenant pose d'autres problèmes comme l'accentuation de l'effet de flou. En effet, si la difficulté à formaliser les critères et les objectifs est

---

<sup>14</sup>mémoire des événements passés qui est acquise durant l'exécution

<sup>15</sup>La docimologie est la science de l'évaluation

du même ordre que dans le réel, leur implémentation est un problème supplémentaire.

## 5.2 Modélisation ITS

Le principal défaut de notre agent est certainement son manque de présence. Le modèle de l'interface n'a pas été mis en place et l'agent est représenté par un humanoïde fixe non-animé qui peut se déplacer dans l'environnement. Il est prévu d'utiliser ce modèle dans la suite du projet pour donner plus de présence à l'agent. L'objectif à atteindre serait d'arriver à un rendu 3D et à des possibilités d'animation au moins égales à celles de STEVE (cf. fig. 1). De plus, la communication avec l'apprenant se limite à l'échange de phrases textuelles et pourra elle aussi être améliorée par ce modèle.

On peut comparer les liens de notre modèle du domaine avec les liens reliant les concepts dans le modèle de référence (équivalent au modèle du domaine) d'ABITS (cf. figure 4). Le lien **-Has Part-** se traduira pour nous en lien **-Se décompose en-**. Le lien **-Is required by-** correspond au lien **-include-** dans le cas d'éléments de même type et au lien **-uses-** sinon. Cette différence s'explique par l'absence de différenciation entre objets, concepts et comportement dans le modèle d'ABITS.

Le domaine d'apprentissage utilisé est assez limité et n'est pour le moment pas très complexe. Le modèle du domaine nous a permis d'exprimer une partie suffisante des connaissances liés au TP pour permettre l'apprentissage.

Il serait pertinent d'évaluer notre méthode sur un domaine plus large, afin de vérifier l'utilisabilité de notre modèle du domaine (du point de vue de l'expression et de l'exploitation en temps réel). Néanmoins, pour valider la généralité de ce modèle, il faut nécessairement implémenter d'autres domaines et vérifier le comportement de notre agent dans un autre environnement. On pourra par exemple utiliser un domaine plus théorique, moins centré sur les comportements.

De plus, il serait peut-être intéressant de réifier des notions transverses à certains domaines. Une notion de criticité et/ou de sécurité pourrait, par exemple, être ajoutée sur des comportements. Ainsi une action dangereuse sera immédiatement interrompue quelque soit le rôle. L'ajout d'une telle notion nécessiterait des modifications assez simple du modèle du domaine (ajout d'un concept `securite`) mais également des modifications du modèle pédagogique.

L'avantage de notre modélisation du domaine est sa forme, proche de certaines ontologies : on peut imaginer un système de traduction d'ontologies existantes pour créer des règles de production SOAR, en particulier pour la partie concernant les concepts. Un autre moyen de générer une partie du modèle serait l'utilisation d'un agent observateur, parcourant l'environnement pour acquérir les informations sur les objets. Pour les comportements, le formateur humain pourrait exécuter les actions pendant que l'agent en retient l'enchaînement pour fabriquer les comportements.

Le modèle de l'apprenant possède des informations limitées sur le profil. On pourrait imaginer un profil prenant en compte des préférences d'apprentissage, tel que celle présente dans le modèle de l'apprenant d'ABITS 2.1.3. Ces informations pourraient être utilisés par le modèle pédagogique pour enrichir l'adaptation de la stratégie pédagogique, mais également par le modèle de l'interface.

Le modèle pédagogique pourra être enrichi par de nouveaux rôles et par de nouvelles activités.

### 5.3 Utilisation de Soar et maintenance des modèles ITS

L'utilisation de SOAR entraîne quelques effets secondaires négatifs comme la visibilité réduite de la base de connaissance et la difficulté de maintenir les modèles. De plus, l'approche conseillée par SOAR pour interfacier l'agent avec son environnement est parfois gênante : il est en effet difficile d'associer des éléments réels à leur représentation dans la mémoire de travail. Cette limitation n'est pas liée à SOAR mais à SGIO : l'interface "officielle" fournie pour interfacier SOAR avec une application C++. Néanmoins, ces aspects négatifs ne sont pas incontournables : on pourra par exemple réaliser une application dédiée à la maintenance graphique des modèles qui générera le code SOAR comme on pourra ajouter les fonctionnalités manquantes à SGIO dans notre propre interface agent/environnement.

Un autre point important est qu'il n'existe ni mécanisme d'oubli ni possibilité de sélection partielle et/ou aléatoire de mémoire dans SOAR et il faut être prudent quant à la sauvegarde des informations : une mémoire de travail trop grande et mal organisée peut conduire à un grand nombre de règles tirées en parallèle et aboutit inévitablement à la saturation des ressources de la plate-forme. Il est possible de ne garder qu'une partie de la mémoire pour réaliser un traitement, mais l'intégralité de la mémoire est tout de même parcourue.

L'utilisation du mécanisme d'apprentissage contenu dans SOAR entraînerait des modifications des règles de production et compliquerait la structure du programme dans son ensemble. De plus, le gain potentiel nous semble faible face à la quantité de ressources, en terme de puissance de calcul et de mémoire, nécessaire à son fonctionnement. Nous décidons donc de ne pas utiliser ce mécanisme.

### 5.4 Rôles de l'agent

L'approche multi-rôle mise en place permet, pour l'atteinte d'un même objectif, c'est à dire faire apprendre, de proposer plusieurs stratégies différentes qui donneront chacune lieu à une expérience individualisée.

Seul trois rôles sont utilisables pour le moment : le professeur, le tuteur et l'entraîneur. Chacun de ces rôles possède une base de connaissances pédagogique spécialisée (règles SOAR) et il est possible de changer dynamiquement de rôle à l'exécution. Pour le moment l'agent ne change pas de rôle de façon autonome et il serait intéressant que ce changement soit le résultat d'une réelle action pédagogique. Cela n'a pas été implémenté pour le moment car ce changement pose un problème d'intégrité sur les données qui n'a pas été résolu : le changement de la stratégie équivaut à un changement de processus, il faut donc créer des règles pour transformer l'état interne de l'agent et le garder cohérent.

De plus, pour montrer la capacité de l'agent à jouer un rôle quelconque, il faudra certainement implémenter un rôle différent des trois déjà fonctionnels, comme par exemple un compagnon d'apprentissage élève ou critique (cf. 3).

## 5.5 Collaboration entre les agents pédagogiques

On peut déjà faire cohabiter plusieurs agents pédagogiques indépendants dans l'environnement (par exemple un pour chaque apprenant), mais aucune confrontation des informations n'est réalisé pour le moment. Il n'y a pas de collaboration ou de mise en commun des informations et des interventions entre ces agents. L'une des perspectives futures de ce projet est la collaboration entre différents agents pédagogiques semblable à celle qui existe dans le système BAGHERA. SOAR ne concerne que le côté interne (base de connaissance et module décisionnel) de l'agent. L'avantage de cette approche est l'indépendance et l'autonomie des agents : chaque agent a son comportement et ses propres représentations du monde et il s'exécute dans un processus séparé. Le désavantage est que notre approche, et plus particulièrement l'utilisation de SOAR, semble insuffisante pour mettre simplement en place une stratégie collaborative entre les agents pédagogiques. On pourra mettre en place une collaboration faible par un système de relais. Par exemple, on pourrait remplacer un agent tuteur par un couple entraîneur/professeur : si un agent entraîneur réalise deux fois un cycle d'évaluation plus supervision, il peut décider de s'écarter momentanément et de laisser la place à un agent professeur qui donnera un cours sur le comportement problématique.

Pour parvenir à une réelle coopération entre des agents pédagogiques comme le notre, il semble pertinent de les intégrer dans une architecture multi-agent prenant en charge les problèmes liés à la collaboration et à la cohabitation entre les agents telle que BRAHMS [8]. Une telle intégration imposerait certainement une nouvelle implémentation du modèle global de l'agent et également de revoir le modèle pédagogique pour prendre en compte l'aspect collaboratif.

BRAHMS est à la fois une plate-forme multi-agent et une suite d'outils logiciels dédiés à cette plate-forme. A l'instar de SOAR, BRAHMS est basé sur un système de règles, possède son propre langage et est centré sur l'expression des activités. BRAHMS possède des similarité avec le modèle BDI et se fonde sur une théorie de cognition situé. Le langage de BRAHMS permet de représenter des activités d'agent situés (dans un modèle géographique du monde) dont le déclenchement est lié au contexte. L'exécution de ces actions dépend :

- des croyances de l'agent sur le monde,
- de l'état du monde (à cette position et ailleurs),
- des activités des autres agents,
- des communications inter-agents,

L'objectif de BRAHMS est de permettre la représentation de connaissances et d'actions informelles, de la collaboration, d'activités multi-tâches, de l'arrêt et de la reprise de procédures, etc.

## 6 Conclusion

Après avoir étudié divers agents pédagogiques et leur modèles, nous avons conçu une approche personnelle en nous inspirant principalement des agents STEVE, HAL et ABITS. Une observation de l'ensemble des données nécessaires à un comportement pédagogique souple et adaptable nous a amené à modéliser les trois modèles ITS que nous jugeons essentiels : le modèle du domaine, le modèle de l'apprenant et le modèle pédagogique. Nous avons déduit de cette approche un modèle d'agent pédagogique basé sur le modèle BDI que nous avons implémenté en utilisant l'architecture cognitive SOAR.

L'intégration dans l'EVF sur le TP de physique nous a permis de tester le caractère opérationnel et les fonctionnalités de notre agent afin de vérifier la cohérence des modèles et la conformité de son comportement vis à vis de l'approche. L'agent adapte sa stratégie pédagogique et donc ses interventions à l'apprenant et au contexte en s'appuyant sur les trois modèles ITS et sur les mécanismes de SOAR.

La mise en pratique de l'agent dans une situation d'apprentissage réelle permettrait une validation partielle de l'approche. On prévoit de simuler grâce à notre agent les protocoles de guidages mis en place par Beney et Guignard : les résultats ainsi obtenus pourraient être comparés à ceux obtenus dans l'expérience réelle et ceux de l'expérience virtuelle (avec un formateur humain). Une autre étude comparative pourra porter sur les différentes stratégies pédagogiques employées par notre agent.

L'un des points que nous n'avons pas traité est la présence de l'agent dans l'environnement, qui est en rapport avec le modèle de l'interface. Cet aspect n'est pas secondaire, encore moins dans un environnement utilisant la réalité virtuelle. Il reste à implémenter le modèle de l'interface pour donner à l'agent des facultés de communication et d'interaction plus proches de celles de STEVE.

Nous avons ouvert plusieurs perspectives d'amélioration du modèle dans la partie discussion dont l'ajout de nouveaux rôles et de nouvelles activités. Un autre axe de recherche intéressant est la mise en place d'une équipe d'agents pédagogiques collaboratifs, telle que dans BAGHERA (cf Annexe I 7.6).

## Références

- [1] William Picken Alexander. *Une échelle de performance pour la mesure de l'intelligence pratique*. Centre de Psychologie Appliquée, 1969.
- [2] John R. Anderson and Brian J. Reiser. The lisp tutor : it approaches the effectiveness of a human tutor. *BYTE*, 10(4) :159–175, 1985.
- [3] Michel Beney and Jean-Yves Guignard. L'évaluation de l'efficacité du guidage dans les travaux pratiques de deug : un problème méthodologique complexe. *Didaskalia*, pages 1–36, 2004.
- [4] Eric Bruillard. *Les machines à enseigner*. Hermès, 1997.
- [5] R. R. Burton and J. S. Brown. An investigation of computer coaching for informal learning activities. In D. Sleeman and J. S. brown, editors, *Intelligent Tutoring Systems*. Academic Press, 1982.
- [6] Nicola Capuano, Marco Marsella, and Saverio Salerno. Abits : An agent based intelligent tutoring system for distance learning. In *proceeding of the ITS 2000 (unpublished ?)*, 2000.
- [7] Nicola Capuano, Massimo De Santo, Marco Marsella, Maria Molinar, and Saverio Salerno. A multi-agent architecture for intelligent tutoring. In *Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, 2000.
- [8] William J. Clancey, Patricia Sachs, Maarten Shierhuis, and Ron Van Hoof. Brahms : simulating practice for work systems design. *International Journal Human-Computer Studies*, 49 :831–865, 1998.
- [9] A. Constantino and D. Suthers. A coached collaborative learning environment for entity-relationship modeling. In G. Gauthier, editor, *Intelligent Tutoring Systems, Proceedings of the 5th International Conference (ITS 2000)*, pages 325–333. Berlin : Springer-Verlag, 2000.
- [10] I.P. Goldstein. The genetic graph : a representation for the evaluation of procedural knowledge. *Intelligent Tutoring Systems*, pages 51–78, 1982.
- [11] Bradley Goodman, Amy Soller, Frank Linton, and Robert Gaimari. Encouraging student reflection and articulation using a learning companion. *International Journal of Artificial Intelligence in Education*, 9 :237–255, 1998.
- [12] Arthur C. Graesser, Katja Wiemer-Hastings, Peter Wiemer-Hastings, and Roger Kreuz. Autotutor : a simulation of human tutor. *Journal of Cognitive Systems Research*, 1 :35–51, 1999. Tutoring Reserch Group, University of Memphis.
- [13] W. Lewis Johnson, Erin Shaw, and Rajaram Ganeshan. Pedagogical agents on the web. In *Proceedings of ITS 98*, 1998.
- [14] J. Laird and J. Duchi. Creating human-like synthetic characters with multiple skill levels : A case study using the soar quakebot.
- [15] J. Laird, R. Jones, and P. Nielsen. Coordinated behavior of computer generated forces in tacair-soar. In *Fourth Conference on Computer Generated Forces and Behavioral Representation*, 1994.
- [16] J. E. Laird. It knows what you're going to do : Adding anticipation to a quakebot. In *AAAI Symposium Series : Artificial Intelligence and Interactive Entertainment*, 2000. AAAI Technical Report.
- [17] J. Lester, J. Voerman, S. Towns, and C. Callaway. Cosmo : A life-like animated pedagogical agent with deictic believability. In *Proc. of the IJCAI97 Workshop on Animated Interface Agents : Making them Intelligent*, 1997.
- [18] R.L. Lewis. Cognitive modeling, symbolic. In R. Wilson and F. Keil, editors, *The MIT Encyclopedia of the Cognitive Sciences*. MIT Press, 1999.
- [19] R.L. Lewis. Cognitive theory, soar. In *International Encyclopedia of the Social and Behavioral Sciences*. Pergamon (Elsevier Science), 2001.
- [20] Domitile Lourdeaux. *Réalité virtuelle et formation : conception d'environnement virtuel pédagogique*. Phd dissertation, École des Mines de Paris, October 2001.

- [21] Ana Paiva and Isabel Machado. Lifelong training with vincent, a web-based pedagogical agent. *International Journal of Continuing Engineering Education and Lifelong Learning*, 12(1/2/3/4) :254–266, 2002.
- [22] Per Persson. Agneta and frida : Browsing, narrative and emotion. Persona Project, 1999.
- [23] Munindar P.Singh, Anand S.Rao, and Michael P. Georgeff. *Multiagent Systems*, chapter 8, pages 331–337. MIT Press, 1998.
- [24] Ronan Querrec, Pierre De Loor, and Pierre Chevaillier. Environnement virtuel pour la formation des officiers sapeurs-pompiers. In A. Seghrouchni and L. Magnin, editors, *Neuvièmes Journées Francophones d’Intelligence Artificielle et Systèmes Multi-Agents (JFIADSMA’01)*, pages 311–314, Montréal, Québec, Canada, 12–14 November 2001. Hermès Science Publications.
- [25] Jeff Rickel and W. Lewis Johnson. Animated agents for procedural training in virtual reality : perception, cognition and motor control. *Applied Artificial Intelligence*, 13 :343–382, 1999.
- [26] Carine Webber, Sylvie Pesty, and Nicolas Balacheff. A multi-agent and emergent approach to learner modelling. In F. van Harmelen, editor, *Proceedings of the 15th European Conference on Artificial Intelligence*, Amsterdam, 2002. IOS Press.
- [27] B.P. Woolf. Building knowledge based tutors. In I. Tomek, editor, *Fourth International Conference of Computer Assisted Learning*, pages 44–60, 1992.

## 7 Annexe I : quelques agents pédagogiques

Nous présentons ici de manière succincte quelques agents éducationnels étudiés, qui possèdent des particularités ou des aptitudes intéressantes.

### 7.1 Steve

STEVE (Soar Training Expert for Virtual Environment) [25] est un agent autonome et animé qui évolue dans un environnement virtuel de formation : VET - *Virtual Environment for Training* (littéralement : environnement virtuel d'entraînement). STEVE est utilisé dans le cadre de travaux pratiques sur des équipements navals complexes [25].

STEVE est un agent pédagogique qui agit de manière autonome (sans l'action du formateur pendant une séquence). Son rôle principal est celui de tuteur mais on peut aussi l'utiliser comme un simple élève (sans stratégie pédagogique associée) exécutant des ordres pour permettre à l'apprenant d'exécuter une action collaborative nécessitant une tierce personne. Les stratégies pédagogiques de STEVE sont plus centrées sur un apprentissage des comportements que sur un apprentissage des connaissances.

### 7.2 AutoTutor

AutoTutor [12] est développé par le *Tutoring Research Group* pour l'*Office of Naval Research* (en collaboration avec la *National Science Foundation*). Ce tuteur a été utilisé et testé dans deux domaines : des cours sur le vocabulaire en informatique et en physique.

AutoTutor est un agent pédagogique indépendant mono-utilisateur, et est basé sur le modèle des agents conversationnels. Le domaine d'apprentissage est représenté par une matrice de mots à  $n$  dimensions (entre 100 et 300) qui est construite en fournissant à AutoTutor des textes à étudier sur le sujet (et en dialoguant avec lui). AutoTutor est capable de suivre le fil de la conversation et d'évaluer les réponses de l'apprenant, grâce à son analyse statistique du langage (*Latent Semantic Analyse* ou LSA).

### 7.3 ADELE

ADELE [13](Agent for Distance education - Light edition) a été développée par le *Center for Advanced Research in Technology for Education (CARTE)* et a été appliquée dans le domaine de la médecine pour deux projets :

1. la résolution de problèmes sur des diagnostics cliniques,
2. l'entraînement au traitement des traumatismes en salle d'urgence.

ADELE est un agent pédagogique mono-utilisateur ayant le rôle de professeur / tuteur / évaluateur basée sur le modèle des agents cognitifs. Elle possède deux interactions pédagogiques principales : elle peut proposer un exercice et offrir l'accès à des informations en rapport avec le problème en cours (conseil), ou analyser les réponses de l'exercice et interroger l'utilisateur pour vérifier ses connaissances.

## 7.4 abits

ABITS [6] (*Agent Based Intelligent tutoring system*) a été réalisé dans le cadre du projet *InTraSys* (*Intelligent Training System in Technical Assistance* ESPRIT). C'est un ITS basé sur une approche multi-agent, créé pour l'apprentissage à distance qui permet d'étendre un système de gestion de cours classique (*Course Management System* ou CMS).

ABITS possède de nombreux agents fonctionnels qui permettent son fonctionnement interne mais il peut être vu comme un seul agent d'un point de vue extérieur. Il peut alors être vu comme un assistant personnel pour le formateur : celui ci pourra bénéficier d'une aide à la décision ou bien automatiser certaines tâches. Par exemple, ABITS pourra adapter le matériel didactique en fonction des préférences et des capacités de l'apprenant.

ABITS évalue à chaque instant le profil de l'apprenant et son état cognitif pour mettre à jour le modèle de l'apprenant (qui utilise une représentation par nombres flous).

## 7.5 HAL

HAL [20] (*Help Agent for Learning*) a été développé par l'équipe dans le cadre du projet SOFI. SOFI est un environnement virtuel de formation (réalité virtuelle) qui immerge l'apprenant au coeur de l'environnement. Ce projet est un simulateur d'entraînement destiné à former les agents de la SNCF aux tâches de maintenance en bordure de voie ferrée.

HAL est un agent pédagogique cognitif non-personnifié qui a le rôle d'un tuteur / gêneur / facilitateur. Il n'est pas complètement autonome puisqu'il a pour but d'assister la formation sans "remplacer" le formateur. HAL analyse et propose des solutions mais laisse le pouvoir décisionnel au formateur. HAL dispose de deux stratégies : une méthode active où le formé exécute les tâches et une méthode explicative où le formé reste passif. De plus HAL possède plusieurs niveaux d'intervention pour chaque mode. En mode actif il pourra laisser faire le formé sans l'assister, lui dire quoi faire, ou faire à sa place. De même, pour le mode explicatif, HAL pourra suggérer la connaissance, suggérer l'erreur ou expliquer. Selon le niveau de l'apprenant et les choix du formateur, HAL pourra modifier son scénario ou l'environnement pour introduire des problèmes ou les inhiber.

L'apprenant dispose d'une interface immersive (écran géant, tapis roulant et gant muni de capteurs) et peut évoluer dans l'environnement en 3D (cf. figure suivante). Dans la même pièce, le formateur observe directement les actions de l'élève. Il supervise l'apprentissage indirectement, lisant des comptes-rendus fournis par HAL, et en choisissant les stratégies pédagogiques.

## 7.6 Baghera

Baghera [26] est un EIAH orienté Web créé au laboratoire Leibniz (IMAG). Ce projet a pour but la réalisation d'une plate-forme d'enseignement générique et multi-utilisateurs basée sur une architecture multi-agent. Le domaine d'apprentissage choisi pour l'implémentation est la résolution de problèmes de géométrie pour de jeunes élèves. L'une des thèses défendues par ce projet est la nature sociale et émergente de la fonction éducative : la responsabilité de l'apprentissage ne peut pas être centralisée dans un seul composant

pédagogique ou module. Elle est le résultat des interactions sociales entre l'apprenant, le formateur et tous les acteurs de leur environnement et l'environnement lui même.

Baghera utilise l'environnement de développement multi-agent JATlite. Les agents de Baghera sont de type cognitif et fonctionnent selon une architecture qui s'inspire du modèle d'agent BDI.

Chaque utilisateur se voit attribuer les services de plusieurs agents dédiés (non-personnifiés). Le processus éducatif est le produit de la collaboration entre tous ces agents et les utilisateurs.

## 7.7 EduAgent

Le projet EduAgent a pour objectif d'étudier différents rôles et stratégies pédagogiques (limités aux compagnons d'apprentissages). Le domaine d'étude choisi est l'apprentissage des équations. Ici plusieurs agents pédagogiques ayant tous le rôle de camarade sont proposés à l'utilisateur : deux camarades "forts" qui répondent de manière correcte aux questions et deux "faibles" qui peuvent parfois donner des réponses erronées (choix aléatoire). Les deux agents faibles simulent une progression au fur et à mesure de leur apprentissage commun avec l'apprenant. L'un des résultats de cette étude concerne les préférences des apprenants sur les différents agents pédagogiques proposés.

## 7.8 LuCy

LuCy est un agent pédagogique inséré dans l'ITS PROPA [11], utilisé pour améliorer les compétences d'analyse explicative dans le domaine du contrôle d'activité satellite. C'est à dire le processus de formulation d'explications scientifiques des SAAs (*satellite activity analysts*) concernant les capacités des satellites artificiels en orbite autour de la terre et leur comportement passé présent et à venir. PROPA est basé sur une approche argumentative de l'apprentissage et contient également un autre agent pédagogique ayant le rôle de tuteur. LuCy joue le rôle de compagnon d'apprentissage critique qui peut insérer des informations bonnes ou erronées (en fonction du modèle de l'apprenant) dans le but de provoquer un dialogue argumentatif. Cet approche oblige l'apprenant à se remettre en question et permet de lui apprendre à justifier ses positions.

## 7.9 COSMO

Cosmo [17] est un agent pédagogique personnifié qui agit dans un environnement virtuel de formation nommé *Internet Advisor* (Le domaine d'application de cet environnement est le routage de paquets sur Internet). Il joue le rôle de tuteur et peut conseiller l'apprenant pendant la simulation ou lui démontrer une solution. Cosmo est représenté par un petit robot humanoïde et possède un éventail assez large de comportements (par exemple se déplacer, cligner des yeux, applaudir, bouger ses antennes) qu'il utilise pendant ses interventions. Il dispose également de plus de deux cents expressions verbales.

## 7.10 COLER

COLER [9] (*Collaborative Learning environment for Entity-Relationship*) est un environnement d'apprentissage collaboratif basé sur internet. Dans cet environnement, les apprenants doivent résoudre des problèmes de modélisation de base de données en petit groupe. Chaque apprenant peut communiquer avec les autres tout en travaillant sur son interface personnelle. L'agent pédagogique de COLER a pour objectif de favoriser l'apprentissage collaboratif : il supervise les actions individuelles puis compare les solutions. Il peut ainsi encourager des dialogues entre certains apprenants et maintenir l'équilibre dans la participation des apprenants.

## 7.11 Vincent

Vincent, [21] est un agent pédagogique animé développé par l'institut INESC. Il a été créé avec l'ITS TEMAI mais peut être adapté à d'autres systèmes. Vincent joue le rôle de professeur/incitateur. Il assiste l'apprenant pendant les exercices en lui apportant des conseils (lorsqu'il détecte une difficulté) et le félicite pour l'encourager. Vincent est composé de deux modules : le module de l'esprit (*Mind Module*), responsable du choix des actions pédagogiques à effectuer et le module du corps (*Body Module*) qui exécute ces actions. Les choix d'actions sont réalisés grâce à la mise en relation des données du modèle de l'apprenant (résultats des exercices, temps passés pour les résoudre et erreurs commises) avec la perception de l'environnement (sous forme d'inférences).

## 7.12 Agneta and Frida

Agneta et Frida [22] sont des assistants personnels pour la navigation Internet. Elles n'ont pas de réelle influence pédagogique, mais présentent quand même deux points intéressants : leur capacité à lire et à interpréter le contenu des pages Web en temps réel et leur comportement qui évolue en fonction du contenu des pages visitées qu'elles commentent (parfois ironiquement).

## 8 Annexe II : quelques rôles d'agents pédagogique

Ce tableau présente quelques rôles pédagogiques qui peuvent être tenus

Rôles	activités
professeur	offre l'accès aux connaissances théoriques. organise des cours.
tuteur	propose des exercices ou des travaux pratiques. donne les éléments théoriques associés. montre et corrige les erreurs. apporte une remédiation.
entraîneur	supervise les actions de l'apprenant. renforce les acquisitions par la répétition.
évaluateur	pose des contraintes de performance propose des sessions d'exercices (examens). élabore des situations critiques pour l'élève (en fonction de ses faiblesses). évalue les progrès de l'élève. vérifie les acquis en posant des questions.
camarade	collabore avec l'apprenant (réalisation de tâches à plusieurs ou entraide). répond aux questions (correctement ou pas).
élève	apprend (ou simule l'apprentissage) en même temps que l'apprenant (observation/rejeu et questions). sollicite l'apprenant en lui demandant de l'aider à comprendre ( <i>teaching by learning</i> )
perturbateur	distrait l'apprenant pour le déconcentrer. exécute des actions parasites pour complexifier l'environnement.
gêneur	provoque des situations critiques pour la résolution du problème. empêche l'apprenant d'exécuter des actions pour l'obliger à revoir son plan. oblige l'apprenant à changer ses buts.
facilitateur	simplifie les problèmes posés pour aider à la compréhension. modifie l'environnement pour aider l'apprenant à percevoir les éléments clefs.
incitateur	motive l'apprenant en le félicitant. encourage l'apprenant à essayer à nouveau en cas d'échec.
critique	pose des questions pour obliger l'apprenant à formaliser ses intentions. encourage l'apprenant à remettre en question son plan.

TAB. 3 – Rôle des agents pédagogiques et activités associées

## 9 Annexe III : le TP de physique réel

Pour évaluer l'efficacité des différents modes de guidage, Beney et Guignard [3] ont mis en place une expérience d'apprentissage sur un groupe d'étudiants de niveau homogène. Ces étudiants ont d'abord subi des tests visant à tracer leur profil cognitif (par exemple le Passalong [1]) puis ils ont participé individuellement à l'expérience d'apprentissage. Cette expérience alterne deux phases, une phase d'apprentissage, puis une phase évaluation. Cinq protocoles, ont été mis en place pour la phase d'apprentissage. Durant cette phase, l'étudiant devra résoudre les trois situation-problèmes suivantes : mesurer la vitesse de la lumière dans l'air, le cube et le tube d'eau.

- Le protocole 0 est un protocole témoin, sans guidage. L'étudiant expérimente seul et doit résoudre le problème en autonomie.
- Dans le premier protocole de guidage, noté 1, on met en place un guidage serré pour les deux premières mesures (air et cube) puis on demande à l'étudiant de résoudre le problème avec le tube d'eau (cf 9.3).
- Le protocole 2 est centré sur la construction d'une représentation interne pertinente liée à la situation-problème. L'étudiant est d'abord laissé en autonomie comme dans le protocole 0. Puis, une solution générale est énoncée verbalement sous forme de règles d'actions. L'étudiant doit ensuite appliquer cette solution aux trois exemples sans autre aide que la règle (cf 9.4).
- Dans le protocole 3, l'étudiant est guidé comme pour le protocole 1, mais sur les trois problèmes. A la fin de l'activité, Il doit répondre à des questions concernant les méthodes qu'il a utilisées et la conscience qu'il a du problème. Cette phase a comme objectif de favoriser la construction d'une représentation du problème à partir de la représentation des solutions (cf 9.5).
- Le protocole 4 est semblable au précédent, mais la phase de questionnement est remplacée par une activité où il doivent dessiner plutôt que verbaliser dans le but de leur faire construire une représentation imagée du problème (cf 9.5).

La phase d'évaluation est identique pour les cinq protocoles : l'étudiant doit calculer la vitesse de la lumière dans un aquarium rempli d'eau. Pour vérifier que l'étudiant a réellement appris, il faut vérifier qu'il s'agit bien d'un transfert et qu'il n'applique pas seulement des automatismes. L'environnement est donc équivalent mais différent : on change la table de position, on place la source lumineuse à droite et on remplace la règle graduée par un mètre ruban.

### 9.1 Résultats de l'étude

L'analyse psychologique et statistique des résultats de cette expérience tendent à montrer que le contexte nécessite bien une tâche de transfert et de conceptualisation pour résoudre le problème de l'évaluation.

- Pour pouvoir comparer les modes de guidages étudiés, Beney et Guignard ont réalisé :
- la mise au point d'une mesure de performance objective et contrôlée (étude quantitative),
  - le contrôle de variable objectivable (étude clinique) susceptibles d'expliquer la progression,

- la mise en relation des performances obtenues et des comportements observés et catégorisés à partir de chaque modèle d'apprentissage.

L'expérience globale a mis en évidence qu'un non guidage ne permettait pas de réaliser des apprentissages et qu'un guidage serré en suivant des instructions pas à pas n'est pas inefficace contrairement à un préjugé répandu. Cependant, l'accompagnement de ce guidage par une verbalisation concernant les problèmes à résoudre et les démarches favorise l'apprentissage, en particulier pour les étudiants qui conceptualisent le mieux.

## 9.2 Le plan de travail réel

La photographie ci-dessous (cf. fig 18), donnée en annexe 1 de l'article de M. Michel Beney et M. Jean-Yves Guignard [3], montre le plan de travail et le matériel mis à disposition des apprenants pour les travaux pratiques de physique réel.

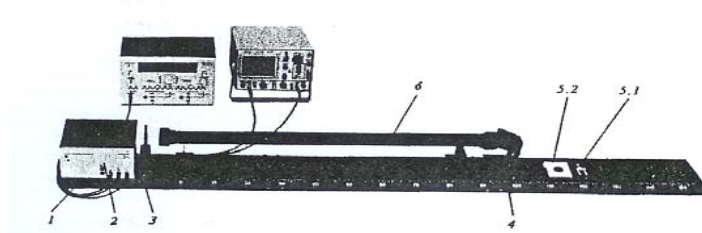


FIG. 18 – Photo du TP de physique

### Légende :

1. Banc d'optique.
2. Boutons de réglage sur la source lumineuse (dont déphasage).
3. Lentilles sur leur pieds magnétiques (2pièces).
4. Miroirs de renvoi (2pièces).
5. Cuvette tubulaire.
6. Prisme en verre acrylique.

Le schéma suivant représente symboliquement les éléments et le parcours de la lumière.

La partie suivante reprend l'annexe 5 du même article [3]. Elle décrit les consignes des 4 protocoles de guidages étudiés et appliqués.

## 9.3 Consignes pour le Protocole 1

D'une façon générale, la vitesse de la lumière dans un milieu transparent (notée  $c$ ) se calcule par la formule  $c = L/\Delta t_1$  où  $L$  représente la distance parcourue par la lumière dans ce milieu et  $\Delta t_1$ , le temps de parcours mesuré à l'oscilloscope.

Mais ici le problème est que les positions de l'émetteur et du récepteur ne sont pas connues avec précision, d'autre part la forme particulière des miroirs ajoute une distance supplémentaire pour le trajet de la lumière.

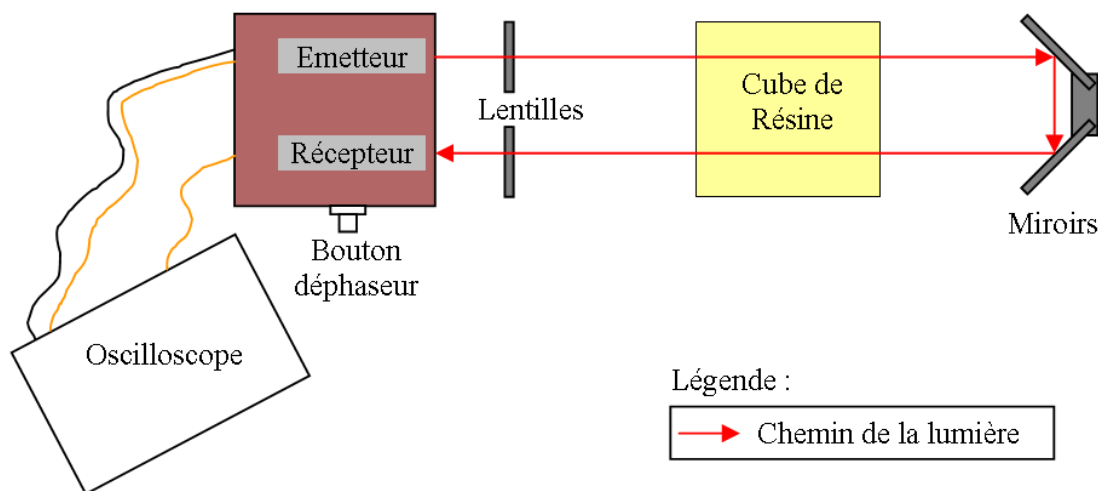


FIG. 19 – Vue schématisée de l'environnement

Pour mesurer le temps que met la lumière à parcourir une distance donnée dans un milieu transparent, on vous propose la manipulation suivante :

### 9.3.1 Mesure de la vitesse de la lumière dans l'air

1. Vous placez les miroirs sur la position zéro de la règle graduée.
2. Vous tournez le bouton appelé « déphaseur » qui est sur le boîtier et ce jusqu'à ce que les deux courbes soient en phase.
3. Vous placez les miroirs à 0,5 mètre.
4. Vous relevez la valeur du décalage en temps entre les deux courbes qui sont visualisées sur l'écran de l'oscilloscope. On notera  $\Delta t_1$ , ce décalage. Le temps effectivement mis par la lumière pour parcourir  $2 * 0,5$  mètre dans l'air est  $\Delta t_1$ .

On peut alors calculer la vitesse de la lumière notée  $c$  par la formule  $c = \frac{2*0,5}{\Delta t_1}$ ,

### 9.3.2 Mesure de la vitesse de la lumière dans la résine

1. Refaites les consignes 1 et 2 de l'expérience précédente.
2. Placez les miroirs à une distance de 0,29 mètre.
3. Placez le bloc de résine dont la longueur est de 0,29 mètre sur le trajet de la lumière.
4. Mesurez la valeur du décalage en temps. On notera  $\Delta t_r$ , ce décalage.

La vitesse  $c$  de la lumière dans la résine est donnée par la formule  $c = \frac{2*0,29}{\Delta t_r}$

### 9.3.3 Mesure de la vitesse de la lumière dans l'eau

Mesurez le temps que met la lumière à traverser le tube rempli d'eau dont la longueur est de 1 mètre. En déduire la vitesse de la lumière dans l'eau.

## 9.4 Consignes pour le Protocole 2

### 9.4.1 Partie 1

D'une façon générale, la vitesse de la lumière dans un milieu transparent (notée  $c$ ) se calcule par la formule  $c = L/\Delta t_1$  où  $L$  représente la distance parcourue par la lumière dans ce milieu et  $\Delta t_1$  le temps de parcours mesuré à l'oscilloscope. Mais ici le problème est que les positions de l'émetteur et du récepteur ne sont pas connues avec précision, d'autre part la forme particulière des miroirs ajoute une distance supplémentaire pour le trajet de la lumière.

Essayer de mesurer cette vitesse dans l'air par une méthode de votre choix et essayez de formuler alors une méthode générale qui permette d'avoir une mesure précise du temps mis par la lumière pour parcourir une distance donnée dans un milieu transparent.

### 9.4.2 Partie 2

Énoncé de la règle : ici pour mesurer le temps mis par la lumière pour parcourir une **longueur donnée** il faut faire une mesure par **différence** entre 2 positions des miroirs distantes de cette longueur. D'une façon générale, on peut dire que quand on ne connaît pas avec précision le zéro d'une mesure on peut malgré tout faire une mesure précise par une méthode de différence. Appliquer ici cette méthode à la mesure du temps de parcours de la lumière :

1. Dans un mètre d'air. En déduire la vitesse de la lumière dans l'air.
2. Dans un bloc de résine d'une longueur de 0,29 m. En déduire la vitesse  $v$  de la lumière dans la résine.
3. Dans un tube d'eau d'une longueur de 1 mètre. En déduire la vitesse  $v$  de la lumière dans l'eau.

## 9.5 Consignes pour le Protocole 3

D'une façon générale, la vitesse de la lumière dans un milieu transparent (notée  $c$ ) se calcule par la formule  $c = L/\Delta t_1$  où  $L$  représente la distance parcourue par la lumière dans ce milieu et  $\Delta t_1$  le temps de parcours mesuré à l'oscilloscope.

Mais ici le problème est que les positions de l'émetteur et du récepteur ne sont pas connues avec précision, d'autre part la forme particulière des miroirs ajoute une distance supplémentaire pour le trajet de la lumière. Pour mesurer le temps que met la lumière à parcourir une distance donnée dans un milieu transparent, on vous propose trois manipulations ; pour les deux premières vous aurez le détail de ce qu'il faut faire sous forme de consignes et vous aurez à réaliser sans aide une troisième expérience.

### 9.5.1 Mesure de la vitesse de la lumière dans l'air

1. Vous placez les miroirs sur la position zéro de la règle graduée.
2. Vous tournez le bouton appelé " déphaseur " qui est sur le boîtier et ce jusqu'à ce que les deux courbes soient en phase

3. Vous placez les miroirs à 0,5 mètre.
4. Vous relevez la valeur du décalage en temps entre les deux courbes qui sont visualisées sur l'écran de l'oscilloscope. On notera  $\Delta t_1$ , ce décalage. Le temps effectivement mis par la lumière pour parcourir  $2 * 0,5$  mètre dans l'air est  $\Delta t_1$ .

On peut alors calculer la vitesse de la lumière notée  $c$  par la formule  $c = \frac{2*0,5}{\Delta t_1}$

### 9.5.2 Mesure de la vitesse de la lumière dans la résine

1. Refaites les consignes 1 et 2 de l'expérience précédente.
2. Placez les miroirs à une distance de 0,29 mètre.
3. Placez le bloc de résine dont la longueur est de 0,29 mètre sur le trajet de la lumière.
4. Mesurez la valeur du décalage en temps. On notera  $\Delta t_r$ , ce décalage.

La vitesse  $c$  de la lumière dans la résine est donnée par la formule  $c = \frac{2*0,29}{\Delta t_r}$

### 9.5.3 Mesure de la vitesse de la lumière dans l'eau

1. Refaites les consignes 1 et 2 de la première expérience.
2. Placez les miroirs à une distance de 1 mètre.
3. Notez la valeur du décalage en temps  $\Delta t_1$ .
4. Placez le tube d'eau dont la longueur est de 1 mètre sur le trajet de la lumière
5. Mesurez la valeur du décalage en temps  $\Delta t_2$ .
6. Le temps de parcours de la lumière dans l'eau est  $\Delta t = \left( \Delta t_2 - \frac{\Delta t_1}{2} \right)$
7. La vitesse  $c$  de la lumière dans l'eau est donnée par la formule  $c = \frac{1}{\Delta t}$

**Répondez aux questions suivantes :** Les mesures de distances et de temps, telle que vous les avez faites en suivant les protocoles expérimentaux, sont des mesures exactes (ii n'y a pas de système particulier dans le boîtier ou l'oscilloscope) pourtant quand vous avez fait les mesures :

- la lumière parcourt une distance entre les deux miroirs,
- il y avait une distance entre le zéro de la règle et l'émetteur de la lumière (ou le récepteur),
- il y avait une distance entre les miroirs et l'extrémité du bloc de résine ou du tube d'eau,
- le bloc de résine ou le tube d'eau dépassaient le zéro de la règle.

Dites en quoi tous ces facteurs n'ont pas eu d'effet sur les mesures (telles que vous les avez faites en suivant les protocoles) des distances et des temps. Maintenant à partir des trois protocoles et des questions précédentes, essayez d'en tirer **une méthode générale** qui permette d'avoir une mesure précise du temps mis par la lumière pour parcourir une distance donnée dans un milieu transparent.

## 9.6 Consignes pour le protocole 4

D'une façon générale, la vitesse de la lumière dans un milieu transparent (notée  $c$ ) se calcule par la formule  $c = L/\Delta t_1$  où  $L$  représente la distance parcourue par la lumière dans ce milieu et  $\Delta t_1$  le temps de parcours mesuré à l'oscilloscope. Mais ici le problème est que les positions de l'émetteur et du récepteur ne sont pas connues avec précision, d'autre part la forme particulière des miroirs ajoute une distance supplémentaire pour le trajet de la lumière. Pour mesurer le temps que met la lumière à parcourir une distance donnée dans un milieu transparent, on vous propose trois manipulations; à partir des trois expériences que vous allez réaliser, vous aurez à essayer de formuler **une méthode générale** qui permet de mesurer **précisément** le temps mis par la lumière pour parcourir une distance donnée dans un milieu transparent.

### 9.6.1 Mesure de la vitesse de la lumière dans l'air

1. Vous placez les miroirs sur la position zéro de la règle graduée.
2. Vous tournez le bouton appelé " déphaseur " qui est sur le boîtier et ce jusqu'à ce que les deux courbes soient en phase. **Faites un schéma en rouge** du trajet parcouru par la lumière correspondant à cette mise à zéro.
3. Vous placez les miroirs à 0,5 mètre.
4. Vous relevez la valeur du décalage en temps entre les deux courbes qui sont visualisées sur l'écran de l'oscilloscope.

**Sur un même schéma tracez en vert** le trajet parcouru par la lumière correspondant au décalage en temps que vous avez mesuré. On notera  $\Delta t_1$ , ce décalage. Le temps effectivement mis par la lumière pour parcourir 2\*0.5 mètre dans l'air est  $\Delta t_a = \Delta t_a/1000$ . On peut alors calculer la vitesse de la lumière notée  $c$  par la formule  $c = \frac{2*0,5}{\Delta t_a}$

### 9.6.2 Mesure de la vitesse de la lumière dans la résine

1. Refaites les consignes 1 et 2 de la première expérience.
2. Placez les miroirs à une distance de 0.29 mètre.
3. Placez le bloc de résine dont la longueur est de 0.29 mètre sur le trajet de la lumière. Sur un même schéma, tracer en vert le trajet parcouru par la lumière correspondant au décalage en temps que vous avez mesuré.
4. Mesurez la valeur du décalage en temps. On notera  $\Delta t_1$  ce décalage.

La vitesse  $c$  de la lumière dans la résine est donnée par la formule  $c = \frac{2*0,29}{\Delta t_1}$

### 9.6.3 Mesure de la vitesse de la lumière dans l'eau

1. Refaites les consignes 1 et 2 de la première expérience.
2. Placez les miroirs à une distance de 1 mètre.
3. Notez la valeur du décalage en temps  $\Delta t_1$ .
4. Placez le tube d'eau dont la longueur est de 1 mètre sur le trajet de la lumière

5. Mesurez la valeur du décalage en temps  $\Delta t_2$ .

Sur un même schéma tracez en vert le trajet parcouru par la lumière correspondant au décalage en temps que vous avez mesuré.

6. Le temps de parcours de la lumière dans l'eau est  $\Delta t = \left(\Delta t_2 - \frac{\Delta t_1}{2}\right)$

7. La vitesse  $c$  de la lumière dans l'eau est donnée par la formule  $c = \frac{1}{\Delta t}$

*À partir des trois expériences, essayez de formuler une méthode générale qui permet de mesurer **précisément** le temps mis par la lumière pour parcourir une distance donnée dans un milieu transparent.*

## 10 Annexe IV : Docimologie et erreurs d'évaluation

- L'effet de fatigue ou d'ennui : il peut engendrer laxisme ou sur-sévérité.
- L'effet de halo : le professeur, influencé par des caractéristiques de présentation (soin, écriture, orthographe) surestime ou sous-estime la note.
- L'effet de relativation : plutôt que de juger intrinsèquement d'un travail, les professeurs jugent ce dernier en fonction des travaux dans lesquels il est inséré.
- L'effet de contamination : les notes attribuées successivement aux différents aspects d'un même travail s'influencent mutuellement.
- L'effet de tendance centrale : par crainte de surévaluer ou de sous-évaluer un élève, le professeur groupe ses appréciations vers le centre de l'échelle.
- L'effet de l'ordre de correction : devant un nouveau travail ou un nouveau candidat à évaluer, un juge se laisse influencer par la qualité du candidat précédent. Un travail moyen paraîtra bon s'il suit un travail médiocre.
- L'effet de stéréotypie : le professeur maintient un jugement immuable sur la performance d'un élève, quelles que soient ses variations effectives.
- L'effet de flou : les objectifs poursuivis et les critères de notation ne sont pas toujours définis avec précision.
- L'effet de trop grande indulgence et de trop grande sévérité : certains juges sont systématiquement trop indulgents ou trop sévères dans toutes leurs évaluations.