ÉpiTalk, un outil générique pour la construction de systèmes conseillers.

Gilbert Paquette, François Pachet, Sylvain Giroux

ÉpiTalk, un outil générique pour la construction de systèmes conseillers.

par Gilbert Paquette[1], François Pachet[2] et Sylvain Giroux[1]

[1] LICEF, Télé-université,
1001 rue Sherbrooke est,
Montréal H2X 3M4 Canada
gilbert_paquette@teluq.uquebec.ca

[2]LAFORIA, Université Paris 6,
4, Place Jussieu, 75252
Paris, cedex 05, France
pachet@laforia.ibp.fr

Résumé

Nous présentons l'architecture d'un outil générateur de systèmes conseillers. Le système ÉpiTalk sert à développer un conseiller qui se greffe à un environnement d'apprentissage ou un système d'aide à la tâche existant sans en perturber le fonctionnement. Après un inventaire des formes que prend le conseil dans les systèmes d'apprentissage à base de connaissances et les tutoriels intelligents, nous présentons des exemples de systèmes conseillers et une première architecture qui a fait l'objet de travaux antérieurs. Puis nous présentons l'architecture et l'implantation d'un nouveau système générateur de systèmes conseillers multi-agents appelé ÉpiTalk. Ensuite, nous présentons trois applications d'ÉpiTalk qui ont été implantées ou sont en voie d'être complétées: AGD, un atelier de génie didactique, COPERNIC-2 un environnement d'apprentissage sur la démarche scientifique, et HyperGUIDE, un environnement pour la formation à distance. En conclusion, nous discutons les résultats sur quatre plans: la faisabilité d'un générateur de systèmes conseillers, la versatilité du générateur quant aux formes de conseil, son applicabilité à des conseils sur des activités de groupe et la qualité des interfaces visant à minimiser l'usage de la programmation.

Mots-cles: systèmes conseillers, environnements d'apprentissage, systèmes d'aide à la tâche, systèmes multi-agents, systèmes d'apprentissage à base de connaissances, système tutoriels intelligents.

Abstract

We report on the architecture of a generator for advisor systems. ÉpiTalk is used to construct an advisor component to an existing learning environment or task support system, without disturbing its operation. After a survey of advisory components in knowledge-based learning environments and intelligent tutoring systems, we present implemented examples of such systems based on a previous generic architecture. Then we introduce a new multi-agent framework of a generator of advisor systems called ÉpiTalk. Finally, we outline three ÉpiTalk advisors that have been programmed or are in progress: AGD, a course design engineering workbench, COPERNIC-2, a learning environment on scientific discovery and HYPERGUIDE, a telelearning environement. We end by discussing our results on four aspects: the feasability of such an advisor generator, the versatiliy of the ÉpiTalk generator with regards to the types of advices, its applicability to collaborative work and learning and, finally, the quality of the designer's interface aiming at reducing the programming load.

Keywords: advisor systems, learning environments, task-support systems, multi-agent systems, knowledge-based learning systems, intelligent tutoring systems

On trouvera dans [Wenger 1987], et plus récemment dans [Jonassen 1991] et [Winkels 1992] trois solides revues du conseil et de l'aide intelligente dans les environnements d'apprentissage et les tutoriels intelligents. Dans ce contexte, nos travaux se concentrent sur une méthodologie de développement de systèmes conseillers se greffant à une application existante, capable de conseiller l'usager sur le contenu de ses travaux ou les méthodes qu'il utilise, et de le faire selon différents modes déterminés par le concepteur du système.

A la section 2, nous présenterons les travaux ayant mené à une première architecture d'un outil générique de conseil intelligent, puis à la section 3, nous présentons une nouvelle architecture d'un tel système, appelée ÉpiTalk. La section 4 est consacrée à trois applications de ce système et la section 5 à la discussion des résultats obtenus jusqu'à présent.

1. Conseils dans les logiciels de formation

Nous discutons ici quatre dimensions du conseil dans les environnements d'apprentissage: l'importance de l'aide du système, l'initiative de l'intervention, le support aux activités de groupe et la nature du conseil: contenu du domaine ou méthodologie.

1.1 Importance de l'aide du système

Nous distinguons d'abord quatre niveaux de système, selon l'importance de l'aide apportée par le système.

1- A une extrémité se trouvent les systèmes qui se contentent de donner une simple *rétroaction* à l'apprenant. C'est le cas notamment de la plupart des progiciels et des hypermédia. C'est également le cas des systèmes experts, lorsqu'ils ne contiennent pas de véritable module d'explication. L'apprenant utilise les outils du système pour réaliser une tâche ou résoudre un problème. Le système affiche certaines conséquences des actions de l'apprenant d'une façon qui devrait l'aider à cheminer vers une solution. Pour maximiser le rôle "enseignant" de tels systèmes, les concepteurs devront y implanter des outils dont les interfaces sont transparentes relativement aux connaissances méthodologiques qui leur servent de support.

2- Un niveau interventionniste moyen consiste à prévoir une *aide interactive intelligente*. Un tel mécanisme affichera, sur demande de l'apprenant, une explication relative à la

composante de l'interface où il se trouve ou à l'outil qu'il utilise, le plus possible en relation avec la tâche qu'il est en train de réaliser. Un bel exemple d'une telle approche se retrouve dans le module d'explication d'un système expert. Celui-ci peut être plus ou moins sophistiqué, mais il offre toujours une explication directement reliée aux interventions de l'usager relativement au cas présentement à l'étude. On peut aussi greffer des systèmes d'aide interactive à des progiciels ou à des micro-mondes pour éviter de laisser l'apprenant trop démuni dans l'univers des actions possibles.

3- Un niveau interventionniste plus élevé est présent dans les systèmes *"coach"* ou les *conseillers actifs*. Dans des systèmes tels que SOPHIE-I [Brown 1975] ou STEAMER [Steven 1983], non seulement l'apprenant obtient sur demande une aide ciblée sur ses activités, mais le système peut également décider d'intervenir pour afficher un conseil lorsqu'il lui semble que l'apprenant a de trop grandes difficultés avec la tâche en cours. Cependant, le conseil n'est pas ici impératif, il peut être suivi ou non par l'apprenant.

4- Enfin, dans les systèmes tutoriels intelligents, entre autres dans le "LISP TUTOR" [Anderson 1984], l'initiative est entièrement laissée au système qui exerce un guidage de l'apprenant de type *tutorat*. Ici les interventions du système sont impératives, les erreurs sont soulignées et demandent correction de la part de l'apprenant.

1.2 Initiative de l'intervention

On peut aussi classifier les systèmes conseillers selon le rôle que jouent les divers agents dans le processus d'apprentissage. Dans la plupart des systèmes mentionnés plus haut, il n'y a que deux agents: l'apprenant et le système. Dans ce contexte, plus l'apprenant est actif, plus le conseiller est passif, moins il intervient, comme c'est le cas dans les systèmes d'aide intelligente. Par contre, plus le conseiller est actif ou carrément interventionniste, comme dans le cas du tutorat, moins l'apprenant a de marge de manoeuvre pour exercer son activité cognitive.

En règle générale, il s'agit de trouver le bon dosage en fonction des besoins de formation des apprenants. C'est par une coopération apprenant/système que l'apprentissage sera le mieux favorisé. Trop de conseils directifs et l'apprenant n'a plus d'espace pour apprendre. Trop peu de conseils et la plupart des apprenants piétineront dans des voies peu productives. Voilà pourquoi de plus en plus de systèmes adoptent un mode mixte de conseil dans lequel c'est

tantôt l'apprenant qui consulte un conseiller passif, tantôt le conseiller pro-actif qui intervient pour donner un conseil ou suggérer un élément de solution.

C'est le cas, par exemple, de ERMA [Brahan 1992], un système d'apprentissage combinant tous les types d'aide. Un système conseiller est greffé à l'outil de modélisation des données informatiques SILVERRUN, lequel permet à un apprenant de construire des modèles de données. En mode *réactif* ou *passif*, le système conseiller fournit sur demande une aide à l'apprenant sur ses constructions. En mode *pro-actif*, le système observe l'interaction de l'apprenant avec le système et offre un conseil pour améliorer ses chances de solutions. Enfin, en mode *tutoriel*, le système utilise des problèmes dont il possède la solution pour guider l'apprenant. Ce module est particulièrement utile auprès d'apprenants débutants.

Support au travail de groupe

Sur un autre plan, on commence à prendre en compte des contextes de collaboration qui tiennent compte de la présence d'autres agents que l'apprenant et le système, notamment le formateur et les co-apprenants [Vivet 1991]. Il faut alors partager les rôles de conseil et d'intervention pour tenir compte de la dimension coopérative de l'apprentissage. Par exemple, dans le concept de classe virtuelle élargie [Paquette 1993a], il est prévu que le système conseiller sera surtout utilisé de façon passive ou active, pour donner des conseils sur la démarche individuelle de l'apprenant, ainsi que sur ses interactions avec le groupe. Des interventions plus globales sont réservées au formateur, qui disposera d'un portrait des démarches individuelles et collectives fourni par le système conseiller.

1.3 Conseils sur le contenu

Nous distinguons ici deux types de systèmes conseillers, selon que les conseils portent sur le contenu du domaine d'étude ou sur des méthodes permettant de résoudre une classe de problèmes. La plupart des systèmes conseillers passés en revue plus haut sont du premier type. Ils opèrent souvent dans le cadre d'une approche de découverte guidée et d'un enseignement de type inductif. Après avoir considéré un certain nombre d'exemples, l'apprenant a souvent tendance à sur-généraliser, c'est-à-dire à énoncer une assertion qui s'applique à tous les exemples qu'il connaît, mais qui est fausse dans certains cas. Le système conseiller intervient alors à l'aide d'un contre-exemple qui incite l'apprenant à modifier son assertion. Dans d'autres cas où l'apprenant s'éloigne trop du but, le système enseignant fournira des indices dévoilant une partie des connaissances.

WHY [Collins 1977], un des premiers systèmes tutoriels intelligents, nous offre une bonne illustration de cette stratégie. Voici un exemple de dialogue[1] de WHY avec un apprenant sur le sujet de la géographie de l'Amérique du sud.

WHY:	Pensez-vous que la jungle amazonienne reçoit des pluies abondantes ou légères?
Apprenant:	Pluies abondantes
WHY:	Pourquoi croyez-vous que la jungle amazonienne reçoit des pluies abondantes?
Apprenant:	Parce qu'elle est à proximité de l'océan Atlantique
WHY:	La région amazonienne est en effet proche de l'océan Atlantique. Comment cela affecte-t-il l'abondance des pluies?
Apprenant:	L'eau a une température élevée
	••••••
WHY:	Pensez-vous que tout endroit montagneux reçoit des pluies abondantes?
Apprenant:	Oui
WHY:	La Californie du sud est montagneuse. Pourquoi ne reçoit-elle pas de pluies abondantes?
	•••••

Comme on peut le constater, les interventions du système sont très liées au contenu du domaine. A la fin, le système présente un contre-exemple. Ici on ne veut pas "donner" la solution mais la faire "découvrir" par l'apprenant au moyen de questions bien planifiées.

Conseils méthodologiques

A l'opposé, les systèmes qui sont présentés à la section suivante peuvent être qualifiés de *conseillers méthodologiques* ou *heuristiques*. Cette caractéristique est imposée par une stratégie pédagogique constructiviste qui suppose une activité cognitive maximale de l'apprenant. Celui-ci explore une situation ou un domaine de connaissance et énonce des assertions, construit des procédures, définit des concepts, élabore un modèle ou une taxonomie, sans être guidé de près par un système enseignant qui fournirait exemples, contre-exemples et indices.

Contrairement au dialogue socratique, le guidage est de type méthodologique ou heuristique plutôt que spécifique au domaine de connaissances. Le système conseiller ne donne pas d'exemples, ni de connaissances spécifiques au domaine, mais il suggère des méthodes à utiliser pour faire progresser la démarche.

[1] Adapté de [Wenger 87], p. 42

Voici l'allure que prendrait un tel dialogue heuristique si on l'appliquait au même sujet que WHY , la géographie de l'Amérique du Sud:

Enseignant:	Pourquoi croyez-vous que la jungle amazonienne reçoit des pluies abondantes?
Apprenant:	Parce qu'elle est à proximité de l'océan Atlantique
Enseignant:	Pour vérifier votre hypothèse, énoncer divers facteurs susceptibles d'influencer le climat; construire un tableau ou un schéma synthèse
Apprenant:	••••• Mon tableau n'indique aucun facteur caractérisant la jungle amazonienne
Enseignant:	Vérifier si vous avez bien inclus tous les facteurs influençant le climat, puis examinez-les pour les régions ayant des pluies abondantes et d'autres non.

Au départ, l'enseignant pose un problème et amène l'apprenant à poser une première hypothèse. Puis il lui suggère de la vérifier en lui proposant un moyen général pour le faire. L'étudiant s'engage dans la construction d'un tableau synthèse qui ne donne pas les résultats escomptés. Il demande alors conseil au système. En examinant le tableau construit par l'apprenant, le système est en mesure de lui faire une suggestion plus précise. Le deuxième conseil est aussi de nature méthodologique et suggère une meilleure dispersion des données dans le tableau. Le conseil est habillé de certains mots du domaine, mais il ne livre aucune connaissance du domaine. Il communique plutôt une métaconnaissance à propos de la nécessité de varier les cas.

Un autre exemple de conseiller méthodologique est fourni par SMITHTOWN [Shute 1986]. Cet environnement concerne la simulation d'une ville fictive dont l'apprenant peut varier les paramètres comme la quantité et les prix de certaines marchandises pour examiner leur effet sur d'autres paramètres. Un conseiller intégré à l'environnement fournit des noms pour les lois découvertes par l'apprenant ou suggère des expériences lorsque l'activité piétine. Sur le plan métacognitif, il offre des conseils sur certains aspects de la méthode scientifique comme la cueillette de données et leur examen systématique.

1.4 Exemples de systèmes conseillers dans la formation

Dans cette section, nous présentons trois exemples de systèmes conseillers qui partagent les caractéristiques suivantes. Ils sont passifs, affichant leurs conseils uniquement sur demande de l'apprenant de façon à maximiser son activité cognitive. Ils sont conçus pour un usage individuel où un apprenant seul interagit avec le système. Ce sont des conseillers méthodologiques, axés sur des tâches génériques comme l'induction de lois scientifiques

(COPERNIC), la planification de projets respectant certaines contraintes (PLANIF) ou la construction d'une taxonomie (LINNÉ).

L'objectif de ces systèmes est d'offrir des conseils méthodologiques relatif à une tâche générique, valables quel que soit le domaine particulier à l'intérieur duquel cette tâche s'exerce. Par exemple COPERNIC a été développé d'abord dans le domaine de l'astronomie, puis utilisé dans d'autres domaines de connaissance comme la chimie des gaz parfaits, la cinématique ou l'électricité.

Ces systèmes ont tous été réalisés au moyen d'extensions apportées au système PRISME/LOUTI qui permet d'assembler divers outils pour créer des environnements d'apprentissage à base de connaissances.

1.4.1 COPERNIC, un conseiller en induction de lois

Dans COPERNIC [Paquette 92a] (Cf. figure 3a), l'apprenant dispose d'outils tels que tableaux, graphiques, réseau de sous-ensembles pour examiner les objets d'une base de connaissances regroupant des données sur le système solaire ou sur la loi des gaz parfaits. Après plusieurs activités d'exploration des connaissances à l'aide de ces outils, l'apprenant est invité à s'intéresser par exemple au problème suivant:

> "quel est le lien entre la période de révolution d'un astre et sa distance de son centre de rotation?"

Des outils d'aide à l'induction utilisent des métaconnaissances relatives à l'induction de lois. Les éducateurs scientifiques y reconnaîtront certains conseils heuristiques qu'ils prodiguent à leurs étudiants pour l'analyse des résultats de laboratoires tels que:

1) garder tous les termes constants sauf deux X et Y;

2) comparer les deux termes variables X et Y; s'ils sont inversement proportionnels, calculer le produit X*Y; s'ils sont directement proportionnels, calculer le quotient X/Y;

3) comparer comme en (2) chaque nouveau terme avec les précédents jusqu'à ce qu'il soit constant ou qu'il varie linéairement en fonction d'une des variables;

4) lorsqu'une expression constante ou linéaire par rapport aux deux variables est obtenue, faire varier d'autres éléments qui étaient maintenus constants, de façon à généraliser cette loi.

Ces heuristiques sont adaptées à partir de celles utilisées par Les programmes BACON, réalisés à l'université Carnegie-Mellon dans le but de modéliser l'activité des chercheurs scientifiques qui induisent des lois à partir de données expérimentales.

Un menu donne accès aux outils permettant d'obtenir du système des conseils sur les couples de variables intéressants à considérer, sur l'opération à effectuer entre deux variables et sur le caractère constant ou linéaire de la dernière variable, à une approximation près. Le message obtenu du conseiller est fonction du contenu de la base des connaissances de l'apprenant au moment où le conseil est demandé. La figure 1b nous montre le menu des agents conseillers, ainsi que trois conseils différents donnés par l'outil "Constante?", pour un même ensemble d'observations à différents moments de la démarche.

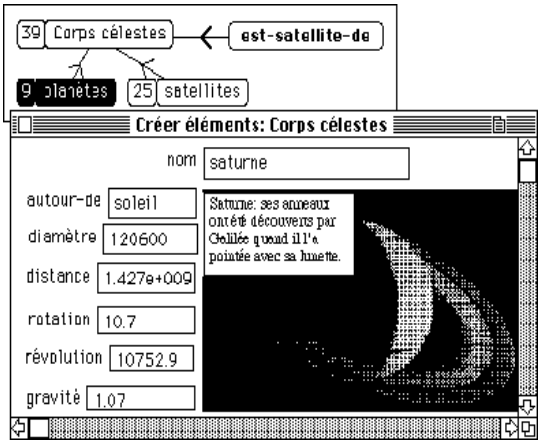

Figure 1a - Outils de travail dans COPERNIC

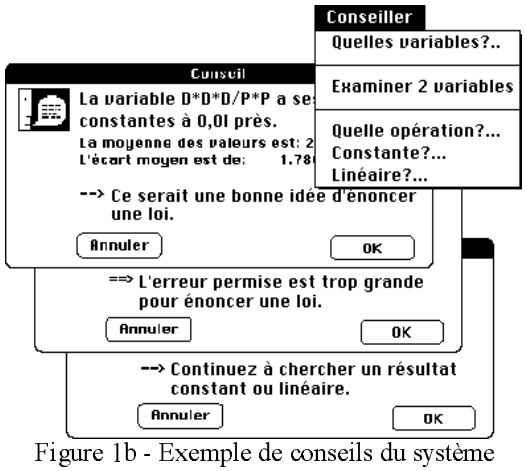

Figure 1b - Exemple de conseils du système

À l'aide de ces outils et de ces agents conseillers, un apprenant peut construire, à partir des observations initiales contenues dans la base de connaissance, des lois aussi différentes que la loi des gaz parfaits ($P*V/n*T$ = constante) ou la troisième loi de Képler (D^3/P^2 = constante)

1.4.2 PLANIF, un conseiller en planification

PLANIF [Paquette 1992b] a été développé à l'été 1991 dans le but de former les agents de planification d'emploi qui doivent analyser les informations sur le marché du travail d'une région et identifier les priorités d'emploi, en respectant les contraintes budgétaires des programmes de création d'emploi et leurs critères d'admissibilité, ainsi que les critères légaux d'équité en emploi. Finalement, les agents doivent construire un plan regroupant les priorités, les budgets des programmes et des projets sélectionnés parmi ceux soumis par différents employeurs.

Différentes bases de connaissances peuvent être construites selon les particularités d'une région ou selon différents degrés de difficulté. Ainsi, un apprenant débutant s'attaquera à une base regroupant des priorités et des projets présentant des solutions faciles, alors qu'une base plus avancée pourrait contenir plusieurs projets non admissibles ou des contraintes budgétaires ou sociales plus difficiles à remplir.

L'apprenant construit son plan en utilisant des interfaces comme celle de la figure 4a. Il édite ainsi les priorités, choisit les programmes et les projets et leur attribue des budgets. Pour l'aider dans sa tâche de planification, il dispose de fichiers d'informations hypermédias sur le marché du travail, la définition des programmes d'emploi, les directives du ministère, etc. Quatre agents conseillers sont à sa disposition pour lui faire des recommandations sur ses choix de priorités, la répartition des budgets, l'admissibilité des projets et, finalement, sur la validité d'ensemble de son plan. Pour faire ces recommandations, le conseiller compare le plan défini par l'apprenant (sa solution) avec des plans experts intégrés à la base de connaissances par le concepteur, mais invisibles à l'apprenant.

À titre d'exemple, un conseil sur la validité du plan, indiquera à l'apprenant qu'il a choisi des projets en dehors des priorités de secteurs économiques et que, de plus, les projets ne permettent pas d'atteindre certains pourcentages d'équité sociale.

1.4.3 LINNÉ, un conseiller en taxonomie

Dans LINNÉ [Paquette 1991] la tâche générique consiste à construire une liste d'ensembles qui forment une partition d'une classe d'objets. Pour construire cette taxonomie, l'apprenant dispose d'une base des connaissances regroupant une description des objets de la classe, définis par les valeurs de leurs attributs. Comme dans COPERNIC, ces objets peuvent être présentés sous forme de fiche, de tableau ou de graphique et ils peuvent être triés de différentes façons.

Contrairement aux environnements précédents, on tient compte, dans les conseils, de l'historique de l'apprenant. Un outil de traçage enregistre les actions importantes de l'apprenant : définition des classes, des attributs ou des objets, utilisation des outils de présentation. Ces actions ou événements, sont enregistrées dans une "base événementielle". Le système conseiller dans LINNÉ utilise cette base d'événements dans deux de ses cinq agents conseillers. Des agents entrent en action sur demande de l'apprenant. L'agent "Constructions" conseille l'apprenant quant à ses opérations sur la base des connaissances, tandis que l'agent "Outils" le conseille sur sa façon d'utiliser les outils relativement à la tâche de construire une taxonomie.

Deux autres agents conseillers comparent la base construite par l'apprenant et une base experte intégrée à l'environnement par le concepteur pour le conseiller sur "l'étendue" de l'un ou l'autre des ensembles de la partition ou pour lui donner une rétroaction sur une "prédiction" qu'il peut faire sur l'appartenance d'un objet à un des ensembles de la partition. Le cinquième agent conseiller vérifie si une liste d'ensembles définis par l'apprenant forme bien une partition: les ensembles sont-ils disjoints deux à deux et couvrent-ils tous les éléments de la classe d'objets? Voici un exemple d'interaction d'un apprenant avec LINNÉ:

APPRENANT:	se présente au poste de travail et choisit la tâche "Charger la base de connaissances sur les vertébrés et définir une partition en cinq sous-ensembles: mammifères, oiseaux, reptiles, poissons et amphibiens". Puis il parcourt plusieurs fiches présentant les caractéristiques des animaux contenus dans la base. Ensuite, il demande un conseil à l'agent "Constructions"
LINNÉ	"Vous n'avez encore défini aucun ensemble. Continuez à regarder des animaux qui sont semblables pour certains attributs. Quand vous aurez trouvé des similitudes consistantes, définissez un premier ensemble."
APPRENANT:	Il définit un premier ensemble "Mammifère: Sang (x) = chaud" et demande un avis à l'agent "Étendue".

LINNÉ: "Votre définition de "Mammifère", lorsque comparée à mes connaissances, regroupe les mammifères et les oiseaux. Vous devriez restreindre la définition".

.....

APPRENANT: Il demande un avis à l'agent "Outils".

LINNÉ: Vous n'avez pas utilisé tous les outils utiles. Il est important de faire des tris de différentes façons pour découvrir leurs similitudes et leurs différences. Utilisez la fonction de tri de l'outil "Tableau".

.....:

2. Une première architecture de système conseiller générique

À partir des exemples qui précèdent se dégage une architecture commune pour les environnements de formation à base de connaissances munis de systèmes conseillers.

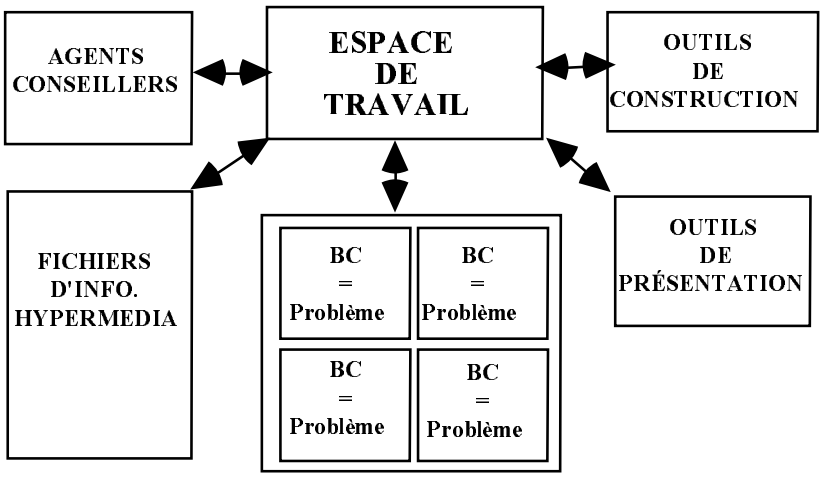

Figure 2 - Une architecture d'environnement de formation avec agents conseillers

2.1 Architecture générale

La Figure 2 met en évidence les caractéristiques de cette architecture générale:

—*Environnements centrés sur la tâche*

Les environnements d'apprentissage présentés à la section précédente sont centrés sur un ensemble de tâches similaires plutôt que sur un domaine de connaissance particulier. Le contenu théorique relatif à ces tâches n'est pas présenté avant que l'apprenant ne commence à travailler. Ces informations sont disponibles en tout temps, au moment où les apprenants en ont besoin, en consultant des fichiers hypermédias

regroupant les informations pertinentes, ou en naviguant dans une des bases de connaissances.

—*Outils génériques de construction des connaissances*

Dans chaque environnement, l'apprenant dispose d'un espace de travail où il construit ses connaissances en utilisant des outils pertinents à la tâche générique. Certains outils tels que fiches, tableaux, graphiques ou réseaux relationnels sont généraux, au sens ou ils sont utilisés dans plusieurs environnements pour visionner les connaissances existantes ou en construire de nouvelles. D'autres outils tels que la sélection de variables dans COPERNIC, la sélection de priorités ou de projets dans PLANIF ou la vérification de partition dans LINNÉ sont particuliers à la tâche générique d'un environnement: induction de lois, planification ou la définition de classifications. En utilisant les outils dans son espace de travail, l'apprenant construit des connaissances dans un domaine particulier, en même temps qu'il utilise et construit les métaconnaissances de la tâche générique incarnées dans les outils.

—*Base de connaissances et résolution de problèmes*

Dans chacun des environnements d'apprentissage, il y a une correspondance entre une base des connaissances et un ensemble de problèmes du domaine correspondant. Par exemple, dans COPERNIC, la même base de connaissance en astronomie peut donner lieu à plusieurs problèmes concernant des lois différentes, mais une autre base peut être sélectionnée sur un autre sujet comme la loi des gaz. Ces bases de connaissances pourront être sélectionnées par le formateur ou l'apprenant pour fournir à ce dernier des problèmes plus ou moins difficiles en fonction de sa démarche. Par exemple dans COPERNIC, un problème d'induction concernant la base sur le système solaire pourra être précédé d'un problème concernant une base plus simple sur la cinématique ou l'électricité. De même, dans PLANIF, une base de connaissances présentant un problème plus difficile pourra être créée en restreignant les contraintes budgétaires ou de priorités et en incluant un nombre plus ou moins grand de projets qui ne respectent pas ces contraintes.

—*Agents conseillers méthodologiques*

Les environnements d'apprentissage intègrent un système conseiller méthodologique qui joue un rôle passif; les conseils s'affichant uniquement sur demande de l'apprenant. La métaphore utilisée ici est celle de l'étudiant levant la main et recevant un conseil spécialisé de l'agent qu'il a choisi. Par exemple, dans LINNÉ, chaque agent conseiller

est compétent pour un aspect du travail: étendue d'un ensemble, définition d'une partition ou prédiction de la classe d'appartenance d'un objet. Cette forme de conseil permet de maintenir une stratégie pédagogique constructiviste, tout en augmentant les chances que l'apprenant converge vers des résultats productifs. Le processus se caractérise par une alternance entre des activités d'exploration, de construction des connaissances et de consultation des agents conseillers.

2.2 Source des conseils

Dans les applications présentées plus haut, les agents conseillers déterminent leurs conseils en utilisant une ou plusieurs de trois sources: la base construite par l'apprenant, une base experte intégrée par le concepteur, et la base événementielle.

Dans COPERNIC, tous les conseils sont déclenchés en examinant uniquement la base de l'apprenant. Dans PLANIF on utilise des comparaisons entre la base de l'apprenant et la base experte pour déterminer les conseils. LINNÉ est le seul des environnements qui fait appel aux trois bases. Deux des agents conseillers déterminent leur conseil en fonction de la base événementielle. Ils examinent d'une part la nature des modifications effectuées par l'apprenant à la base de connaissance et, d'autre part, les outils qu'il a utilisés pour ce faire.

2.3 Traçage et analyse de la base événementielle

L'implantation du concept de base événementielle a été facilitée par l'architecture de LOUTI[2]. Une méthode "set-trace" a été insérée dans certains objets de traitement de la base de connaissances et également dans les objets implantant les outils à la disposition de l'apprenant. Cette méthode retourne les modifications apportées par l'apprenant à la base de connaissances ou les outils utilisés par l'apprenant et les objets auxquels ils ont été appliqués.

Un outil TRACE a ensuite été construit et ajouté aux outils de LOUTI, pouvant ainsi s'intégrer dans tout environnement de formation. Cet outil fait appel aux méthodes "set-trace" pour construire un événement chaque fois qu'une méthode de ce type est déclenchée par un objet de l'environnement. Chacun de ces événements est un objet dont les attributs sont l'identification de la session, le numéro d'ordre de l'événement, le temps écoulé depuis le

---

[2] LOUTI est un atelier de réalisation d'environnements d'apprentissage à base de connaissances. Dans LOUTI, le concepteur choisit des modes de représentations, développe une ou plusieurs bases de connaissances, sélectionne et répartit des outils de traitement destinés, puis génère l'application destinée à l'apprenant. Pour plus de détails, voir [Paquette 1991,1992b]

début, la structure de la base de connaissance qui a été créée, modifiée, détruite, affichée ou cachée, l'outil utilisé et l'action particulière que l'outil a permis de faire. Par exemple, lors d'un tri dans un tableau, l'événement correspondant contient notamment l'ensemble qui a été trié, le nom de l'outil (tableau) et l'action particulière (le tri).

Cette base événementielle est globale, regroupant en une seule classe tous les événements survenus depuis le moment où l'outil TRACE a été activé. Cependant, elle pourra être décomposée pour une analyse plus fine à l'aide des mêmes outils présents dans l'environnement d'apprentissage (tableaux, réseaux, graphiques...). Cette décomposition pourra être faite par l'apprenant, le formateur, ou un agent conseiller intégré au système.

La figure 3 montre l'état de la base événementielle à la fin d'une démarche comme celle présentée à la fin de la section 1 sur une tâche de classification des vertébrés en cinq groupes. Il y a 66 événements. Sur le premier graphique, on note deux temps d'arrêt, vers 300 secondes et 500 secondes. La deuxième graphique montre que les temps d'arrêt coïncident assez exactement avec l'usage de l'outil "Classement". Cette hypothèse est illustrée en définissant d'abord l'ensemble des événements "Action=Classement", puis en affichant le tableau de ces événements. La troisième partie, à droite de la figure 7 montre l'ensemble des 4 événements "Structure=Poisson". On constate que le concept de Poisson a posé certaines difficultés à l'apprenant. Il y a eu deux usages de la création ou de la modification d'ensembles sur ce thème, soit vers 400 secondes, puis vers la fin de la démarche.

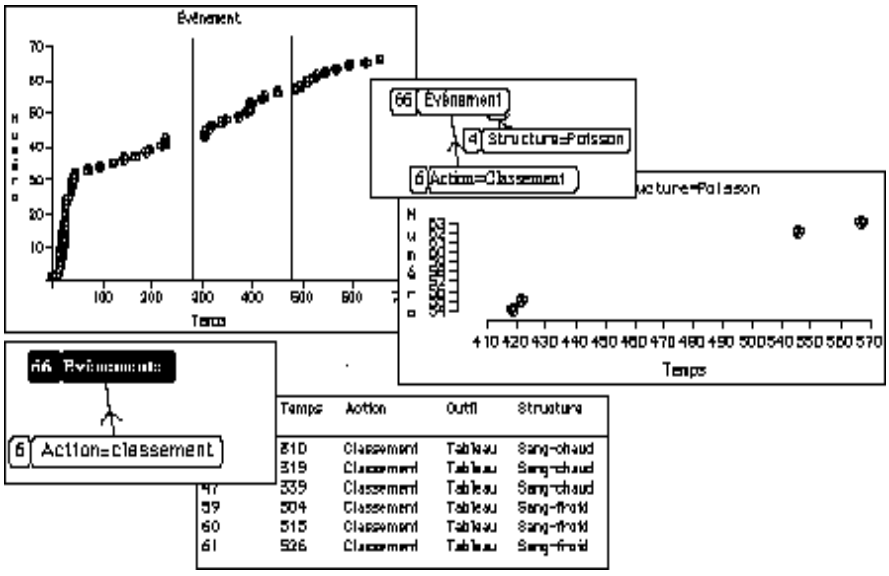

Figure 3 - Un exemple de base événementielle et son analyse

Caractéristiques principales de la première architecture

L'architecture commune aux exemples qui précèdent se résume ainsi:

- un mécanisme de traçage est intégré aux principaux outils d'un environnement de formation, captant les interactions jugées importantes en général;
- un outil TRACE peut être configuré par le concepteur pour construire une base événementielle ne conservant que les événements jugés importants pour une application donnée;
- cette base événementielle est globale; elle peut toutefois être particularisée en définissant des sous-ensembles d'événements;
- un objet "outil-conseil" a été créé; cet objet permet de définir deux composantes principales de chaque outil héritant de l'objet: une base de règles permettant de déclencher une action comme l'affichage d'un message à l'usager, et une liste de "données-préalables" qui sont des méthodes calculant des indicateurs pouvant apparaître dans les conditions de règles.

3. L'Architecture d'ÉpiTalk

A partir des travaux qui précèdent, nous avons entrepris la construction d'un système conseiller générique dont l'architecture permettrait:

1- De greffer un système conseiller à toute application dotée d'outils d'aide à la tâche, sans perturber le fonctionnement de l'application, plutôt qu'intimement intégré à une application comme dans le projet LOUTI.

2- De faciliter à un concepteur la réalisation de conseillers, passifs ou actifs, pouvant donner des conseils de toute nature, méthodologiques ou de contenu, au choix du concepteur.

3- D'offrir des conseils autant dans un scénario individuel que dans les divers scénarios d'apprentissage collaboratif inhérents au concept de classe virtuelle [Paquette 1993a]

4- De fournir au concepteur des interfaces graphiques lui permettant de définir les points d'insertion et le contenu des conseils, en minimisant l'usage de la programmation.

3.1 Le concept d'écosystème informatique

Les idées développées dans ReActalk [Giroux 1993] introduisent la notion de *programmation par écosystème* que nous allons reprendre dans notre architecture. Un système informatique est vu comme un *écosystème* qui évolue en fonction de ses *interactions* avec son environnement. La structure de cet écosystème est réifiée (représentée explicitement) à un *métaniveau*, dans lequel sont spécifiées les lois régissant le niveau inférieur.

Plus précisément, deux types d'interactions nous intéressent. Dans le premier, un apprenant est en interaction individuelle avec un environnement d'apprentissage. Dans le second, le système informatique doit supporter les interactions entre plusieurs individus (apprenants, professeurs et/ou conseillers). Si l'on utilise la programmation par écosystèmes telle que développée dans ReActalk, on peut utiliser la même architecture dans les deux cas.

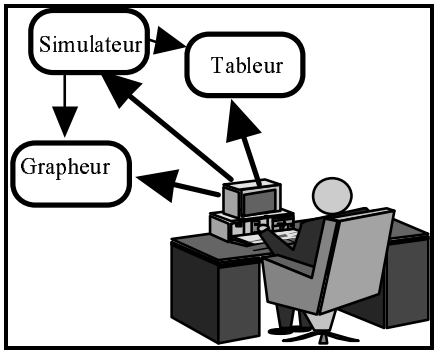

Figure 4a - Interaction d'un apprenant avec des outils

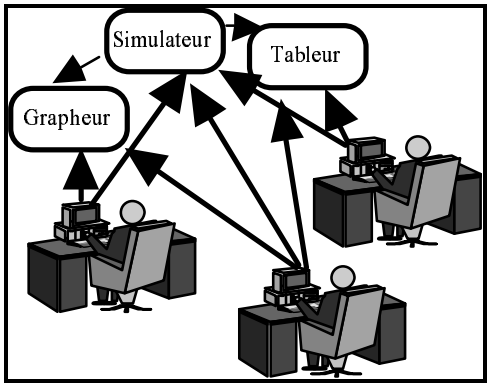

Figure 4b - Interaction de plusieurs apprenants entre eux et avec des outils

Dans le premier cas de figure, l'environnement de l'écosystème est constitué principalement du seul apprenant. Ainsi l'observation des stimuli et des interactions entre les outils utilisés par l'apprenant fournit la base sur laquelle un conseiller (interactif ou humain) pourra opérer.

La figure 4a illustre la situation d'un étudiant qui cherche à induire des lois de la physique à l'aide d'un certain nombre d'outils supportant la démarche scientifique : un simulateur pour faire ses expérimentations, des tableurs et grapheurs pour organiser ses résultats. Pour donner des conseils sur la démarche scientifique, il s'agit alors de surveiller l'utilisation que fait l'apprenant de ces outils et de surveiller les interactions entre ces outils. L'utilisation des outils correspond aux stimuli que reçoit l'écosystème de son environnement. A partir de ces informations, le système peut effectivement donner des conseils sur la démarche scientifique.

Si après un certain temps, l'apprenant n'a pas utilisé le simulateur, on pourra lui suggérer que l'expérimentation est le moyen usuel pour découvrir une hypothèse. De même, on pourra surveiller les interactions entre le simulateur et le tableur pour analyser les données transmises du premier au second et conseiller des formes d'équations mathématiques. Par

exemple, “comparer les variables X et Y, si elles sont inversement proportionnelles, calculer le produit X*Y; si elles sont directement proportionnelles, calculer le quotient X/Y”

Si nous considérons le second cas de figure, celui d'un travail coopératif entre plusieurs apprenants, nous obtenons une configuration analogue (figure 4b). Cette fois, l’écosystème informatique à observer est formé des outils disponibles aux acteurs de la classe virtuelle pour réaliser une tâche et des outils de communication entre eux. L’environnement de la classe virtuelle est composé de tous les apprenants et, s’il y a lieu, du ou des pédagogues en fonction. La structure des interactions va refléter la structure du groupe.

Ainsi, il devient possible de tenir compte, non seulement des tâches réalisées par chaque individu, mais aussi des interactions entre les individus et les ressources documentaires à des fins de télétravail, de télédiscussion, de téléconsultation ou de télétutorat.

Ainsi la programmation par écosystème capture le processus d’aide autant au plan individuel qu’au plan collectif. Il s’agit alors d’observer/espionner les interactions entre une personne et des outils, ou bien entre plusieurs personnes à travers des outils. Les intervenants[3] ne sont jamais observés directement. Le système raisonne alors à partir de ces interactions réifiées. Son raisonnement dirige ses interactions (les conseils) avec son environnement. Les interactions entre les outils eux-mêmes ou encore la structure du système peuvent aussi intervenir dans son raisonnement.

3.2 Une architecture multi-agents en trois niveaux

L’architecture proposée est réflexive et comporte essentiellement trois niveaux. Le premier niveau interagit directement avec l’usager. C'est l'environnement d'apprentissage proprement dit. Le second niveau observe le premier et peut donner des conseils sur la démarche ou adapter certains éléments de l'environnement d'apprentissage. Le troisième niveau contrôle le second niveau et permet notamment d'adapter ses actions selon un modèle de l'usager construit au second niveau.

—*Le premier niveau* comprend des outils génériques d’apprentissage et les outils spécifiques au domaine d’application. Ces outils permettent à un apprenant ou à un groupe d’apprenants de résoudre un problème ou d’accomplir une tâche donnée en exercice. Leur utilisation cristallise l’apprentissage et l’utilisation des connaissances. Par exemple, dans un

3 Dans la suite de ce document, le terme “apprenant” fera tout aussi bien référence à un individu pris isolément qu’à un groupe d’individus puisque cette architecture permet d’exprimer les deux aspects.

environnement d'apprentissage pour la chimie, nous retrouverons à ce niveau les informations sur les éléments chimiques, les connaissances sur la composition des éléments et des outils génériques tels un simulateur et un tableur permettant de conduire et d'analyser une expérimentation.

— *Le second niveau* réifie (modélise, représente) la structure du premier niveau considéré comme un écosystème. Cette réification sert de base pour organiser la cueillette des faits sur lesquels le système conseiller travaille. Entre autres, elle permet d'observer indirectement un apprenant à travers les actions et paramètres qu'il transmet au système. Ce procédé présente le net avantage de dissocier l'implantation des outils de première ligne et le raisonnement sur l'utilisation de ces outils. Un système conseiller s'insère donc tout naturellement à ce niveau. Ainsi on pourra déduire si un apprenant se conforme à la démarche scientifique en étudiant une trace de l'activation des outils.

Pour ce faire, on aura besoin d'un système à base de règles qui fonctionne par chaînage avant. Les règles mises en oeuvre raisonnent sur deux bases de faits, d'une part le modèle de l'apprenant et d'autre part les interactions réifiées. La base des interactions est alimentée par le système de surveillance mis en place lors de la réification. Le modèle de l'apprenant est mis à jour par les règles et trace l'évolution de l'apprenant en fonction de ses interactions avec le système.

—*Le troisième niveau* de l'architecture s'intéresse à la multitude des manières de dispenser des conseils. Par analogie, nous pouvons dire qu'il va donner des conseils sur la manière de donner des conseils. Il s'agit de rendre opérationnels des principes tels que

- "il ne faut pas submerger l'apprenant de conseils";
- "à telle étape, il faut suivre de manière serrée le cheminement de l'apprenant";
- "il vaut mieux ne pas répéter le même conseil de la même manière".

Cette structuration en trois niveaux est abstraite. Il reste à déterminer comment structurer les agents conseillers entre eux, étant donné la diversité des types de conseils, la complexité de l'application sous-jacente, et surtout comment le faire sans introduire de nouveaux concepts orthogonaux, qui augmenteront encore la complexité du système.

3.3 Définition d'un système conseiller

Dans notre optique, un système conseiller est vu de manière générale comme une *extension* d'une application (système d'aide à la tâche, environnement d'apprentissage, système

tutoriel), quelle qu'elle soit. Afin d'éviter les confusions de vocabulaire, nous proposons d'utiliser les termes suivants dans un sens relativement précis.

On appellera ici **outil** tout élément logiciel avec lequel un utilisateur interagit, dans le cadre de l'application, par exemple, le Browser Smalltalk, un grapheur, un tableur, une interface pour définir des graphes de concepts, etc.

Une **interaction** est une action de l'utilisateur avec un outil. Les interactions sont le plus souvent des actions de la souris (cliquer, sélectionner dans une liste, accepter un texte dans un champ, etc.). Mais d'autres types d'interactions peuvent être envisagées (parole, vidéo). Une interaction est localisée dans le temps (heure, minute, seconde, voire milliseconde).

Les **applications** considérées ici sont constituées essentiellement d'un ensemble *cohérent* mais *non structuré linéairement* d'outils. Les outils sont cohérents car ils présentent à tout instant un état d'une situation quelconque. Ils sont non structurés linéairement car l'utilisateur peut se promener librement dans cet univers, et aucune contrainte n'est imposée sur l'ordre de parcours ou d'utilisation des outils ni sur la nature des actions qu'il doit effectuer.

Par exemple, un navigateur ("browser") comme celui de Smalltalk peut être vu comme une application à observer dont les fenêtres sont reliées entre elles pour présenter un état cohérent de l'ensemble des classes. Mais c'est un système non structuré linéairement: l'utilisateur est libre d'utiliser le navigateur comme bon lui semble. De même, le système "Induction de lois", comporte des outils pour définir des classes de contraintes, un simulateur de lois physiques, des outils graphiques (grapheur, tableur) etc. Ces outils sont cohérents entre eux mais aucun ordre particulier n'est imposé pour l'utilisation de ces outils.

Un **système conseiller** est un système informatique qui produit des conseils à partir de l'espionnage de l'interaction entre un utilisateur et un environnement d'apprentissage.

Un **conseil** est une action informatique quelconque, ne perturbant pas le déroulement du système étudié (l'application). Le plus souvent un conseil est un *texte,* affiché dans une fenêtre réservée à cet effet, sur décision du système conseiller ou à la demande de l'utilisateur. Dans d'autres cas un conseil pourra être une action informatique plus significative, comme l'ouverture d'un nouvel outil, mais notre approche privilégie la vue d'un

conseil comme une action neutre. Un contre-exemple de conseil serait donc la non-validation par le système conseiller d'une action de l'utilisateur.

En d'autres termes, les conseils ne doivent pas être considérés comme des contraintes d'intégrité, ni le système conseiller comme un résolveur de contraintes, au sens informatique du terme comme dans Prolog III ou ThingLab.

3.4 Propriété "épiphyte" du système conseiller

Le système conseiller qui est construit au-dessus d'un environnement de formation ou d'aide à la tâche doit posséder des propriétés d'indépendance. Cette indépendance est à comprendre dans deux sens :

> Au sens conceptuel, l'architecture du système conseiller est indépendante de celle de l'environnement d'apprentissage. Leur architectures ne sont aucunement semblables ou isomorphes. En particulier les liens entre les outils ne ressemblent en rien aux hiérarchies des agents conseillers chargés d'espionner ces outils et de raisonner dessus.
>
> Au sens du génie logiciel : les concepteurs des outils ne doivent pas concevoir ces derniers en fonction du système conseiller qui va être greffé dessus. Cette contrainte est importante, car elle est le gage de la réutilisabilité des outils. Tout outil programmé en fonction d'un système conseiller quelconque perd ses propriétés de réutilisabilité.

Toutes ces propriétés sont résumées en un seul adjectif que nous donne la botanique : épiphyte. Cet adjectif décrit le comportement de certaines plantes qui vivent par-dessus des plantes existantes, sans en troubler le fonctionnement normal[4].

> épiphyte : (botanique, du grec "épi", sur et "phyte", plante) Qui croît sur d'autres plantes, sans en tirer sa nourriture. Opposé à parasite. Le lierre, les lianes, sont des plantes épiphytes.
>
> parasite : (biologie, du grec parasitos, de "sitos", nourriture). Organisme animal ou végétal qui vit aux dépends d'un autre, appelé hôte, lui portant préjudice, mais sans le détruire (à la différence d'un prédateur)...
>
> (Dictionnaire Petit Robert)

[4] Et non pas simplement parasites, car le but d'un système conseiller est d'observer sans modifier, ce qui n'est pas le cas des plantes parasites.

3.5 Principes de base

La multiplicité des niveaux d'expertise nous a conduits à choisir naturellement une architecture multi-agents [Gasser 1991]. La vision multi-agents est bien adaptée à la nature des interactions dans les environnements ouverts, à plus forte raison s'ils doivent supporter le travail de groupe. Le système est vu comme une collection d'agents conseillers, chargés chacun d'une tâche particulière. Certains agents seront chargés de "conseiller" sur une tâche bien délimitée, d'autres sur un ensemble de tâches. Il reste à spécifier comment ces agents conseillers sont organisés. C'est l'objet des quatre principes de base suivants:

(1) Un *graphe des tâches* , décrivant de manière hiérarchique les tâches effectuées par l'utilisateur est l'élément principal de toute l'architecture.

(2) Les conseillers sont organisés en un graphe hiérarchique. Ce *graphe des agents conseillers est isomorphe au graphe des tâches* fourni par le concepteur.

(3) les interactions entre un utilisateur et les outils sont recueillies par des objets informatiques appelés *"espions", qui s'insèrent dans le système sans en troubler le fonctionnement*. Ces espions peuvent espionner toute interaction entre l'utilisateur et un outil, et transmettre les messages espionnés aux agents conseillers "terminaux". C'est le concepteur qui doit spécifier quelles interactions (et donc quels *messages*) particulières chaque conseiller terminal est chargé d'analyser.

(4) Les conseillers se *transmettent l'information de bas en haut*. Seuls les conseillers terminaux (associés à des outils) reçoivent les interactions de l'utilisateur avec les outils. Chaque agent traite ces informations puis retransmet des informations au niveau supérieur, et ainsi de suite.

Ces quatre principes soutiennent toute l'architecture qui s'y conforme systématiquement. Les deux premiers décrivent comment les conseillers sont créés et organisés entre eux. Le troisième spécifie comment les interactions sont capturées, ainsi que la structure des espions, et est relativement indépendant des trois autres. Le quatrième principe spécifie comment les informations sont transmises d'un agent conseiller à un autre.

3.6 Espionnage des interactions

L'espionnage des interactions est fondé sur les hypothèses suivantes :

1- On espionne les *messages transmis aux objets* après une action de l'utilisateur (et non pas directement les actions de l'utilisateur).

2- On espionne les messages *à la réception* (et pas à l'émission).

Par exemple, on ne recueillera pas directement un *clic* de sélection d'un item dans une fenêtre, mais plutôt le message envoyé après ce clic, par la fenêtre elle-même, à l'objet du modèle pour le prévenir que tel item a été sélectionné.

Ces hypothèses peuvent avoir certaines conséquences non intuitives. Les plus importantes sont les suivantes :

—On ne recueille, par définition que les interactions qui se traduisent (en termes informatiques) par un envoi de message à un objet. Ceci est trivial, mais on ne pourra pas espionner les clics ou actions de l'utilisateur qui ne sont pas interprétés par l'application, par exemple un bouton qui ne fait rien.

—On ne pourra pas espionner toute une catégorie d'envois de messages, en particulier les messages qu'un objet s'envoie à lui-même. Ceci est cohérent avec la notion d'espion, qui intercepte des communications entre agents différents. On ne peut pas espionner l'activité *interne* d'un objet.

3.7 Graphes de l'application, des tâches, des conseillers

L'architecture d'ÉpiTalk est basée sur la manipulation de trois graphes parfaitement distincts les uns des autres. Dans l'implantation actuelle, trois outils permettent de visualiser et d'éditer ces graphes, puis d'éditer les classes qui définissent les espions et les agents conseillers.

Le graphe de l'application représente l'état courant de l'application. On y voit les outils actuellement ouverts. Chaque fois qu'un outil est ouvert, un représentant de cet outil est ajouté à ce graphe. Les noeuds de ce graphe sont les outils instanciés, les "vrais outils". Les connections entre les noeuds représentent les connections entre les outils. Pour l'instant, tous les liens entre ces noeuds ne sont pas indiqués car ils n'interviennent pas du tout dans la construction du système conseiller. la fenêtre de ce graphe sert uniquement à consulter l'état de l'application.

L'utilisation projetée de l'application est représentée par un graphe des tâches. Ce graphe sert à spécifier l'architecture du système conseiller de manière abstraite. C'est en effet à partir de ce graphe que le système conseiller sera construit. Il décrit le *point de vue* de l'application de l'auteur du système conseiller (le graphe de tâche considéré comme matérialisation du point

de vue en EpiTalk est discutée plus en détail dans [Pachet & al 94]). On y distingue trois sortes de noeuds:

—*Les noeuds "terminaux" ou feuilles.* Ces noeuds n'ont pas de fils. Ils ont un statut particulier car ce sont à ces noeuds que l'on associera les outils du système étudié. Seuls les noeuds terminaux du graphe de description se voient associer un ou plusieurs outils. Par convention, on représente les noeuds terminaux entre crochets, [].

—*Les noeuds non terminaux simples.* Ce sont les noeuds qui ne sont pas des feuilles et qui représentent des objets dont on connaît le nombre de composants.

—*Les noeuds non terminaux étoilés.* Ce sont les noeuds qui ne sont pas des feuilles et qui représentent des objets dont on ne connaît pas a priori le nombre de composants (de 1 à n). Par convention, on représente ces noeuds en les suffixant par une étoile "*".

La figure 5 présente l'écran d'édition du graphe des tâches. Cette fenêtre permet d'ajouter ou de supprimer des noeuds et des arcs à la description, ainsi que d'éditer les noeuds. On y remarque une tâche "*", ce qui signifie qu'on pourra naviguer simultanément dans un ou plusieurs "Browser". Les tâches entre crochets sont les tâches terminales ou "feuilles".

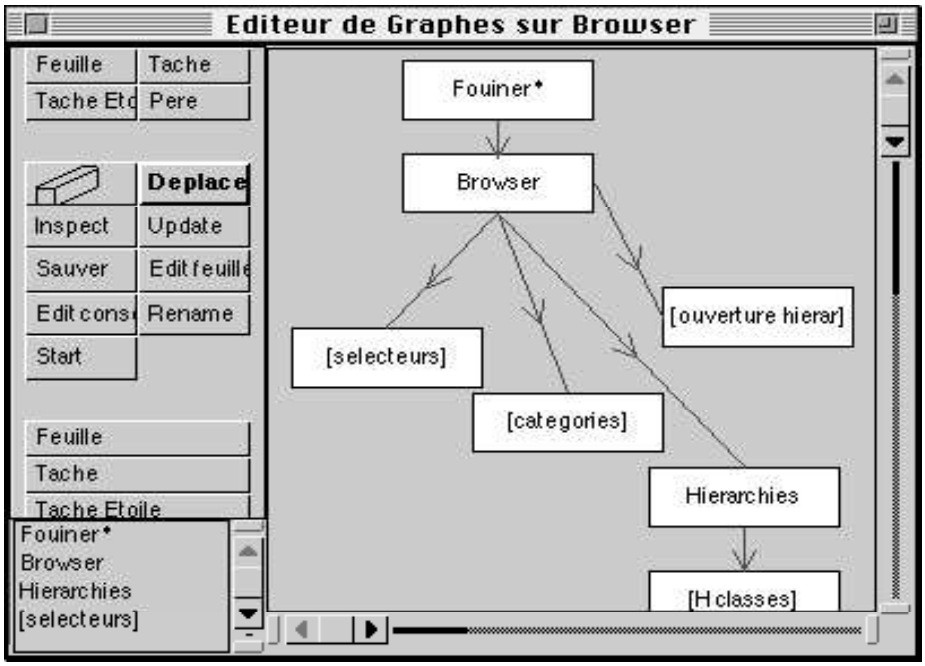

Figure 5. L'éditeur de graphes de description de tâche

Par la suite, l'édition des feuilles permet de spécifier quelles classes seront espionnées par le conseiller correspondant, et pour chacune d'elles, quels messages seront interceptés. Cette

fenêtre présente trois listes, tel qu'indiqué sur la figure 6: une liste des classes Smalltalk devant être espionnées (on peut en ajouter ou en supprimer); pour chaque classe une liste des messages que cette classe comprend et qui peuvent être espionnés et, finalement, une liste des messages sélectionnés pour la classe sélectionnée.

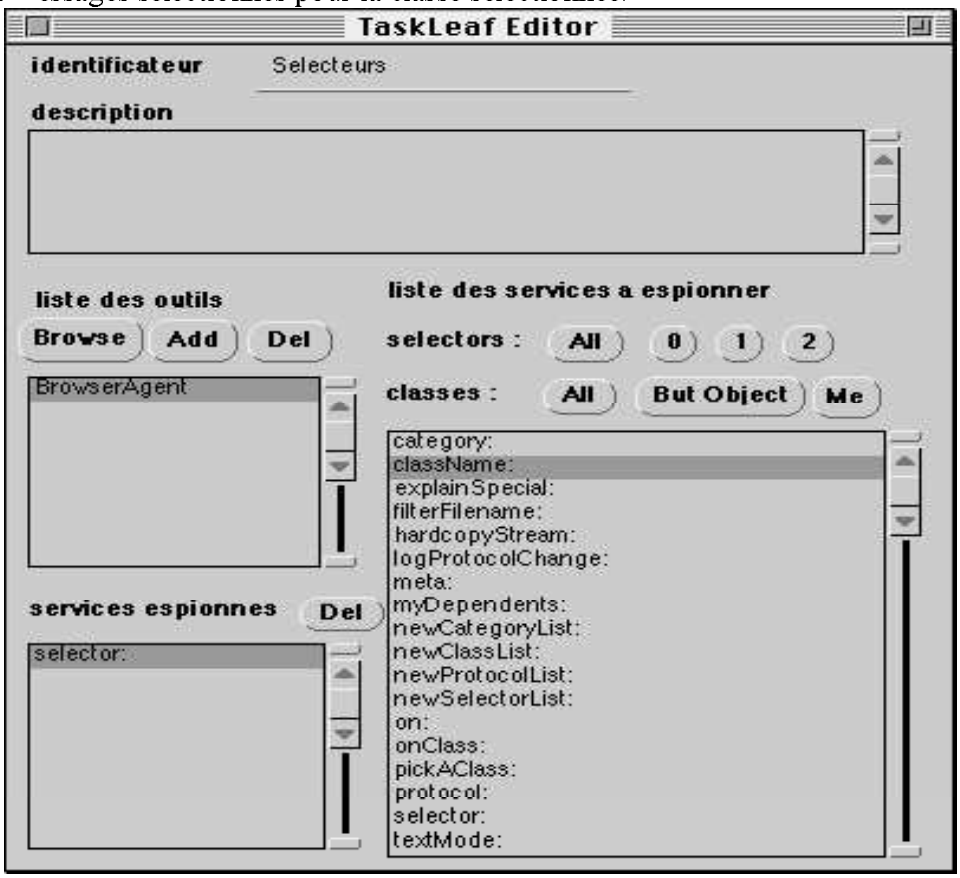

Figure 6. La fenêtre d'édition des tâches terminales

Le terme "graphe des tâches" est volontairement général pour pouvoir inclure toutes les sortes de descriptions qu'un expert conseiller peut avoir à faire sur une application existante. Cette description peut être une description des *tâches* effectives à effectuer par l'utilisateur, ou bien une description des *résultats* que celui-ci doit produire.

Ce graphe ne contient qu'un seul type de lien, celui de hiérarchie de partie, ou d'agrégation. Bien sûr, il peut exister d'autres types de liens entre les tâches, comme les liens de précédence temporelle (telle tâche doit être effectuée avant telle autre), ou des liens conditionnels (cette tâche sera faite si telle condition est remplie). Tous ces autres liens pourront être intégrés comme partie anatomique des agents conseillers. Par exemple, pour traduire une contrainte de précédence entre deux tâches t1 et t2, on définira un

comportement particulier du conseiller correspondant à la tâche englobant t1 et t2. Ce comportement pourra par exemple vérifier que les deux tâches sont effectuées dans le bon ordre et émettre un message si ce n'est pas le cas.

Le graphe des agents conseillers est isomorphe au graphe des tâches. Ce graphe est engendré automatiquement par le système, en même temps que les objets "espions", à partir du graphe des tâches et des outils, en fonction des résultats de l'édition des tâches terminales.

L'hypothèse fondamentale de l'architecture est que le système des agents conseillers est isomorphe au graphe des tâches. L'isomorphisme est réalisé simplement en associant à chaque noeud du graphe des tâches un et un seul agent conseiller. Si le noeud est terminal, l'agent conseiller sera terminal. Si le noeud est non terminal l'agent conseiller correspondant sera non terminal. De plus, les liens de hiérarchie du graphe des tâches sont transposés dans le graphe des conseillers.

La sémantique du graphe des conseillers est entièrement déterminée par le graphe des tâches spécifié par le concepteur. En effet, c'est à partir du graphe des tâches que le système conseiller sera construit, au moment du lancement de l'application. Plus précisément, le graphe des tâches a deux fonctions :

—organiser les agents conseillers dans une hiérarchie,

—spécifier, pour chaque conseiller terminal, quels sont les outils qu'il doit surveiller, et pour chaque outil quels sont les messages que l'on désire capturer.

3.8 Que fait un conseiller ?

Nous avons décrit comment les agents conseillers sont organisés entre eux. Nous allons maintenant décrire le fonctionnement d'un conseiller en particulier. Le fonctionnement des conseillers est régi par les deux principes suivants :

1. Seuls les conseillers terminaux reçoivent les interactions espionnées.
2. Les informations sont transmises de bas en haut dans le graphe des conseillers

La spécification des messages à analyser pour un conseiller terminal se fait par l'éditeur de conseillers terminaux. Chaque conseiller terminal peut recevoir des informations de plusieurs outils, et pour chaque outil on peut spécifier quels sont les messages à intercepter. A la réception d'une information, l'agent, qu'il soit terminal ou non, traite l'information

localement, puis transmet un signal (la même information, ou une autre, plus abstraite) à son supérieur hiérarchique.

Le traitement local de l'information se fait suivant un schéma simple fondé sur une métaphore: chaque agent est doté à sa création de "parties anatomiques", spécialisées chacune dans un type particulier de traitement. Le traitement local consiste simplement *à transmettre l'information aux parties anatomiques existantes*.

L'information transmise au supérieur hiérarchique n'est pas nécessairement l'information reçue. Ce peut être une information plus abstraite. L'idée est que plus un conseiller est haut dans la hiérarchie, plus il traite des informations abstraites (et plus le volume d'informations qu'il a à traiter doit être réduit). Idéalement, les interactions en tant que telles ne devraient être manipulées qu'au niveau des conseillers terminaux.

On obtient donc un graphe d'interprétation dans lequel les informations sont envoyées aux feuilles (les agents terminaux) qui, pour les traiter les transmettent à leur parties anatomiques (vers le bas). Ensuite les informations sont transmises vers le haut, et de chaque noeud vers ses parties anatomiques (vers le bas), et ainsi de suite.

3.9 Partie anatomique d'un agent

La notion de partie anatomique permet de donner au concepteur un moyen simple et efficace de définir les comportements des agents conseillers. L'idée est de proposer une palette de parties anatomiques standards, représentées par des classes. Chacune des classes d'une partie anatomique est capable d'effectuer une petite tâche particulière, comme mémoriser une information, trouver des régularités, gérer la destruction d'un objet, déclencher une base de règles, etc. Le concepteur *n'a qu'à* brancher sur le conseiller qu'il est en train de définir les parties anatomiques dont il a besoin. Évidemment, les parties existantes ne pourront pas couvrir tous les cas possibles, et il sera nécessaire, pour les systèmes conseillers sophistiqués, de définir de nouvelles classes de parties anatomiques.

La figure 7 illustre les parties anatomiques d'un conseiller qui peuvent être spécifiées via une interface d'édition disponible dans ÉpiTalk. Par défaut, les agents sont créés sans aucune partie anatomique. Ils ne font donc rien. Mais on peut simplement redéfinir le comportement d'un agent en sélectionnant une classe de parties anatomiques existantes, ou bien en définissant de nouvelles. Pour l'instant, nous n'avons défini qu'un petit nombre de parties

anatomiques: une mémoire, une base de règles (dans le formalisme NéOpus, intégrant des règles d'ordre un en Smalltalk [Pachet 92, Pachet 95]) et un collecteur.

—Pour la **mémoire**, deux classes sont disponibles : une mémoire qui enregistre tout ce qu'elle reçoit (AgentMemory) et une qui ne fait rien (AgentNoMemory). D'autres sous-classes pourraient être envisagées, qui conserveraient les informations seulement un certain temps, ou des informations de manière sélective. Une classe de mémoire pourrait aussi tenter de détecter des "régularités" dans la mémoire.

—Pour les **bases de règles,** l'interface permet de spécifier directement une nouvelle base de règles NéOpus, ou de consulter/modifier les bases de règles existantes.

—Pour les **collecteurs**, une seule classe existe, chargée de la création et de la connexion de nouveaux objets. Ce collecteur est typiquement utilisé pour les tâches terminales chargées de l'espionnage de la création/ouverture de nouveaux outils à partir d'outils existants.

Figure 7
La fenêtre d'édition des conseillers

4. Applications d'ÉpiTalk

Nous allons maintenant présenter trois applications qui utilisent ÉpiTalk dans le but d'adjoindre un système conseiller à un environnement d'apprentissage ou d'aide à la tâche.

4.1 L'atelier de génie didactique

L'atelier de génie didactique (AGD) [Paquette 1993b] est un système d'aide à la tâche destiné aux concepteurs pédagogiques. L'atelier repose sur des connaissances conceptuelles, procédurales et stratégiques propres au domaine du design pédagogique.

Les tâches d'ingénierie didactique, telles que le choix des connaissances ou la formulation d'un objectif d'une unité d'apprentissage, représentent des situations pour lesquelles le concepteur est susceptible d'avoir besoin d'aide. Le système conseiller d'aide à la tâche doit donc être greffé sur les outils correspondant à ces tâches, et plus précisément sur certaines actions dans ces outils comme la création d'un objectif ou la modification de la description générale d'un cours, que l'on nommera *points d'insertion* du système conseiller. A ces points d'insertions correspondent des agents conseillers au sens d'ÉpiTalk.

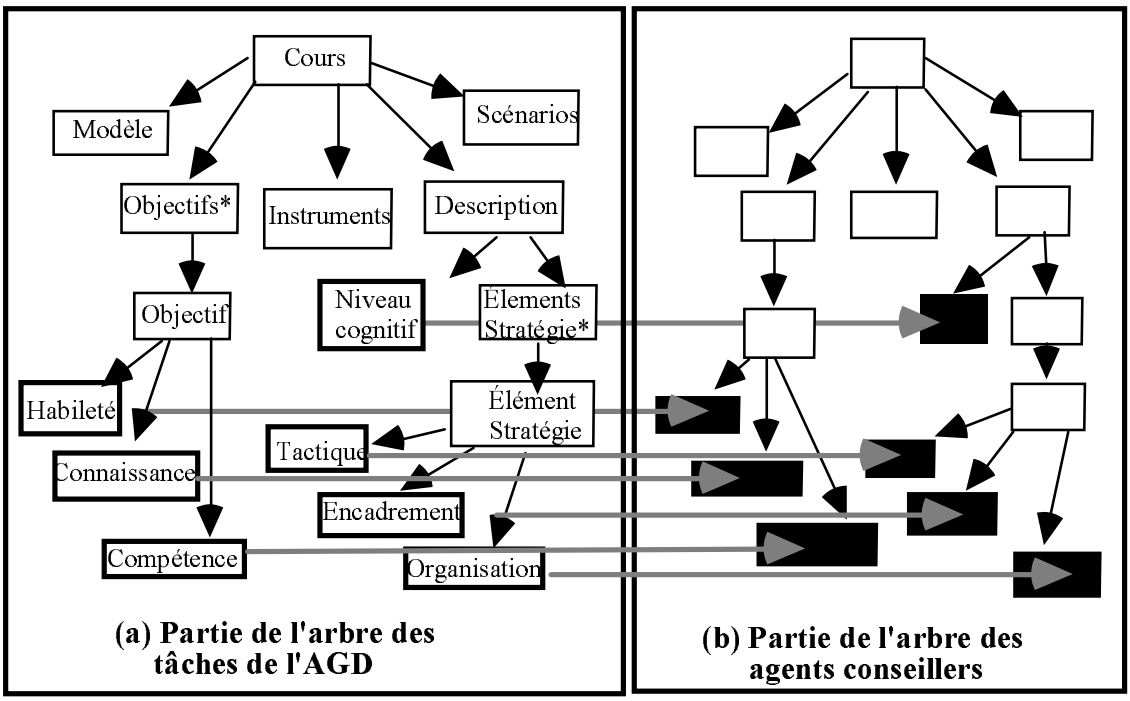

Figure 8 - Une partie de l'arbre des tâches et des agents conseillers dans l'AGD

La figure 8a, présente une partie du graphe des tâches de l'AGD. La tâche de conception d'un cours se décompose en cinq sous-tâches: construire le *modèle* des connaissances, énoncer les *objectifs* d'apprentissage, faire la *description* de la stratégie pédagogique, définir les *instruments* didactiques et définir les *scénarios* d'apprentissage. La tâche "énoncer les objectifs" est une tâche étoile, car on doit définir une liste d'objectifs. Cette tâche a pour sous-tâche "énoncer un objectif", laquelle se décompose en trois sous-tâches: "choisir une habileté", "choisir une connaissance", "choisir une compétence". Ces trois dernières tâches sont terminales.

La figure 8b, présente l'arbre des agents conseillers. Les conseillers terminaux (en noir sur la figure) conseillent sur une seule tâche du graphe des tâches. Les conseillers situés à un

niveau plus élevé dans l'arbre des tâches, par exemple "concevoir le devis d'un cours", peuvent conseiller sur une tâche non terminale, grâce aux conseillers inférieurs qui leur transmettent des messages à propos des composantes de cette tâche. Le conseiller situé à la racine de l'arbre des tâches conseille sur l'ensemble du projet, notamment sur la cohérence entre les analyses, le devis du cours et le plan de réalisation. Actuellement, quelques deux cents règles conseillent l'usager sur les principales tâches de conception. Pour chaque point d'insertion, on détermine les attributs qui influencent la décision en ce point et on décide de la façon dont la règle va se déclencher.

Par exemple, pour la formulation des objectifs, il y a actuellement 52 règles qui se manifestent de trois façons différentes:

—spontanément à l'initiative du système; c'est le cas des règles qui concernent le trop grand nombre d'objectifs dans une structure pédagogique:

> **Si** le nombre d'objectifs <n> d'une UF est > 6 et si public-cible = hétérogène,
> **alors** afficher le message "Cette UF comporte déjà <n> objectifs, une ampleur au dessus de la moyenne. Puisque le public-cible est hétérogène, plutôt que d'augmenter le nombre d'objectifs, il vaut mieux concevoir des versions différentes de cette UF destinées à différents publics-cible".

—lorsque l'usager actionne le bouton "Vérifier la complétude" dans l'outil d'édition des objectifs; c'est le cas des règles qui s'assurent que les principales connaissances du modèle sont couvertes par au moins un objectif d'apprentissage;

—lorsque l'usage actionne le bouton "Valider l'objectif" dans la même interface; c'est le cas des autres règles concernant par exemple la cohérence des niveaux taxinomiques des habiletés ou le choix des habiletés en fonction des besoins de formation:

> **Si** un <objectif d'UF> a une habileté de niveau taxinomique inférieur à celui d'un <objectif activité>
> **et si** <objectif activité> est une composante de la connaissance de <objectif d'UF>,
> **alors** afficher le message "L'objectif d'une activité doit être une spécialisation des objectifs de son unité de formation. Or <objectif activité> a un niveau taxinomique supérieur à celui de <objectif UF>. Vous pouvez corriger la situation en augmentant le niveau taxinomique de <objectif UF> ou en diminuant celui de <objectif activité>".

4.2 COPERNIC revisité

L'environnement d'apprentissage COPERNIC [Paquette 1992a], présenté à la section 2, a été reconstruit en Smalltalk, dans le but de développer un système conseiller plus sophistiqué

sur l'induction de lois scientifiques. COPERNIC-2 contient actuellement des outils pour planifier une expérience et générer des observations, analyser des observations en tableau et en graphique, rechercher des invariants, formuler une hypothèse, la valider, puis la généraliser.

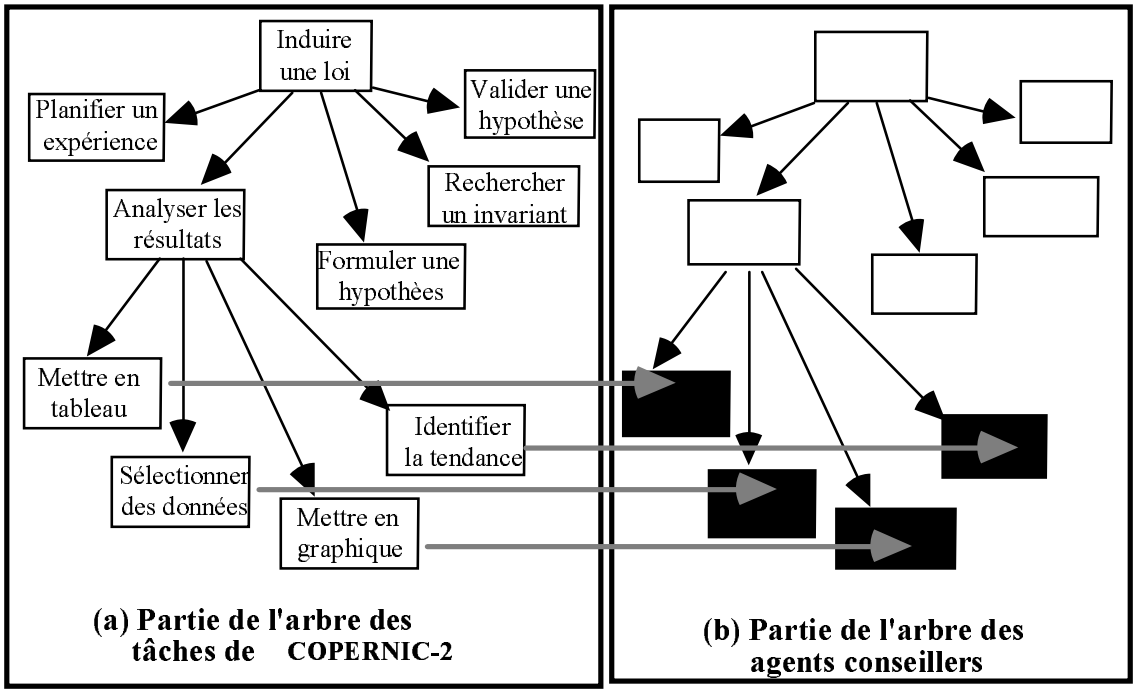

Figure 9 - Une partie de l'arbre des tâches et des agents conseillers dans COPERNIC-2

La figure 9 présente une partie seulement de l'arbre des tâches qui sera utilisé par le système conseiller. On y présente quatre tâches terminales d'analyse des résultats pour lesquelles des espions seront définis.

Par exemple, pour la tâche "Analyser les résultats", des conseils doivent être donnés sur la sélection des données à mettre en graphique quant à leur "bonne" dispersion. Également, un choix des variables à mettre en graphique pourra être recommandé en fonction d'un calcul de tendance. Voici trois exemples de règles parmi celles qui sont développées actuellement.

> **Si** une sélection de données est telle qu' hormis deux attributs variables, tous ses autres attributs sont constants, à l'exception d'un petit (< 10%) pourcentage de données <D>
> **alors** afficher le message "Vous devriez éliminer les données <D> de votre sélection car, en les conservant, il y a plus de deux attributs qui sont des variables. Avec un tel échantillon, il est difficile de conclure quoi que ce soit".
>
> **Si** le nombre de données dans une sélection est > 10, que celle-ci a deux attributs variables et les autres constants, et que l'un des attributs variables a des valeurs bien réparties dans son domaine,

alors afficher le message "Votre échantillon est bien choisi. Il est bien réparti et comporte suffisamment de données pour conclure. Utiliser l'outil d'examen des tendances."

Si le nombre de données dans l'ensemble est > 50 et que celles-ci sont bien distribuées,
alors afficher le message "Les données que vous avez obtenues des expériences sont suffisamment nombreuses et différentes pour que nous puissions vous suggérer les couples <C> de variables à examiner".

4.3 HyperGUIDE

HyperGUIDE [Paquette 1993a] est un logiciel hypermédia disponible au poste de travail de l'apprenant dans un environnement de formation à distance. Il constitue la feuille de route interactive de l'apprenant qui décrit les différentes activités d'apprentissage du cours. Il permet un accès intégré aux différentes ressources pédagogiques disponibles: personnes-ressources, co-apprenants, documents multimédias, conseil interactif. Il offre enfin diverses formes de rétroaction à l'apprenant, lui permettant d'évaluer sa démarche et de la réorienter au besoin.

Dans HyperGUIDE, les activités d'apprentissage constituent les noeuds du réseau hypermédia. À partir d'une activité, l'apprenant peut choisir entre plusieurs activités subséquentes, les choix disponibles étant parfois limités par des contraintes telles que des préalables entre concepts ou entre les étapes d'une méthode.

Chaque activité d'apprentissage est définie comme une transformation sur l'ensemble des documents d'un cours. Au départ, l'apprenant dispose des documents mis à sa disposition par l'équipe de conception. Chaque activité l'amènera, par la suite, à produire de nouveaux documents et à les valider en interaction avec le système et un formateur-tuteur, étendant ainsi la base d'information et de connaissances à sa disposition.

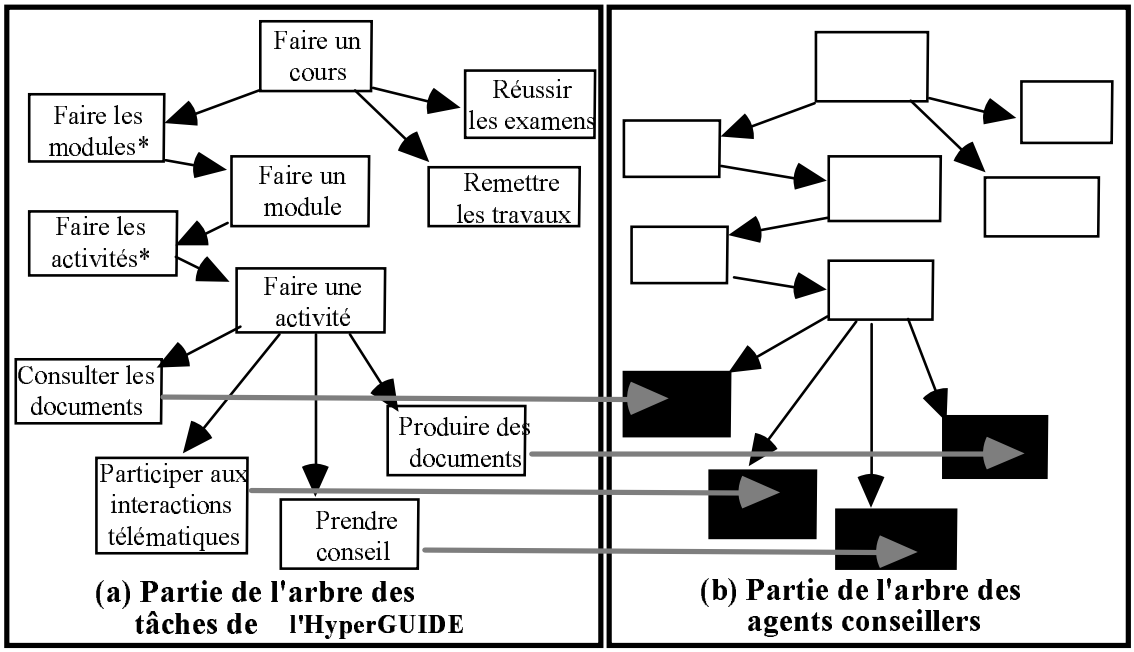

Figure 10 -Une partie de l'arbre des tâches et des agents conseillers dans HyperGUIDE

La figure 10 met évidence la structure hiérarchique des tâches dont les principaux niveaux sont: (1) COURS -> (2) MODULE -> (3) ACTIVITÉ-> (4) CONSULTATIONS & PRODUCTIONS. Les tâches terminales sont soit la consultation de documents, la consultation de personnes (tuteur, co-apprenant, équipe, groupe), ou des productions à l'aide d'outils intégrés à HyperGUIDE ou appelés par celui-ci.

Pour chaque tâche terminale, on doit préciser ce que l'on espionne, quels messages de l'interface du document ou de l'outil de production seront interceptés et transmis aux agents conseillers correspondants.

On peut *construire un arbre des tâches pour chaque point de vue adopté sur l'environnement d'apprentissage*. On peut examiner uniquement la nature des interactions des apprenants entre eux et avec le formateur, les connaissances consultées et produites par un apprenant ou sa démarche générale dans le réseau des activités, des modules et des cours.

Nous avons entrepris le développement d'un tel conseiller sur la démarche, ce qui permettra de l'intégrer à tout HyperGuide, quel qu'en soit le contenu particulier. Dans ce cas, on ne peut qu'espionner chaque entrée/sortie dans l'écran décrivant chaque activité, et également d'évaluer le progrès des apprenants dans les activités, basé sur l'ordre des actions de l'apprenant et des facteurs quantitatifs comme le nombre de caractères des textes, le nombre

de champs remplis et le temps consacré à chaque séjour dans une tâche terminale. On peut également préciser certaines hypothèses en posant des questions (en petit nombre) à l'apprenant.

Ces informations sont transmises par chaque espion à l'agent conseiller correspondant pour chaque tâche terminale. Celui-ci détient sa portion de la trace de l'apprenant dans sa partie "mémoire" et utilise une base de règles qui peut afficher un message à l'usager spontanément ou sur demande et décider des informations transmises à son agent conseiller supérieur: l'activité. Ici l'activité est intéressée à recevoir des données regroupées sur ses tâches terminales: consultation de documents, consultation de personnes, productions; à la fois sur le chemin de parcours, la quantité de travail affectée à chaque tâche et la quantité de productions réalisées.[5] Au besoin certaines questions posées à l'usager permettront de préciser une hypothèse avant le conseil. Puis les différentes activités d'un module transmettront leurs informations groupées au module, et les différents modules au cours dont ils font partie.

5. Discussion des résultats

Au début des travaux sur le projet ÉpiTalk, nous nous étions fixés un certain nombre d'objectifs qu'il convient d'évaluer, de façon à identifier les lignes directrices des travaux à poursuivre.

5.1 Faisabilité de l'architecture

Nous avons atteint, pour l'essentiel, les objectifs énoncés au début de la section 3:

> 1- Greffer un système conseiller à toute application existante dotée d'outils d'aide à la tâche, sans en perturber le fonctionnement, plutôt qu'intimement intégré à l'application.

Pour le moment, nous devons nous limiter aux applications programmées en Smalltalk. Une analyse des possibilités du système Windows et du transfert d'ÉpiTalk en langage C^{++} est en cours. Elle pose plusieurs difficultés, mais elle pourrait permettre d'élargir le champ d'application du système.

---

[5] En ce qui concerne la QUALITÉ des productions, il faut tenir compte du contenu propre à chaque HyperGuide. Par exemple, si l'HyperGuide est un cours sur le design pédagogique, des règles analogues à celles de l'AGD serviront de base aux agents conseillers. De même, un HyperGuide sur l'induction de lois pourra intégrer l'environnement et le système conseiller de COPERNIC-2.

5.2 Versatilité du conseil

> 2- Faciliter à un concepteur la réalisation de conseillers, passifs ou actifs, pouvant donner des conseils de toute nature, méthodologiques ou de contenu, au choix du concepteur.

L'implantation de l'AGD a démontré que l'architecture permet de réaliser des agents conseillers passifs ou actifs, pouvant donner des conseil méthodologiques ou sur le contenu. En fait, une même application peut recevoir plusieurs systèmes conseillers adoptant chacun un point de vue différent sur l'application.

5.3 Conseils individuels et coopératifs

> 3- Offrir des conseils autant dans un scénario individuel que dans les divers scénarios d'apprentissage collaboratif inhérents au concept de classe virtuelle.

Le conseil coopératif ne devrait pas poser de problème théorique, comme l'ont démontré certains essais sur une application de jeu de Scrabble coopératif. Cependant, il faudra résoudre des problèmes pratiques d'espionnage d'application qui ne sont pas programmées en Smalltalk et qui sont utilisés dans le travail coopératif, comme les logiciels de courrier électronique, de téléconférence assistée ou les collecticiels ("*groupware*").

5.4 Minimiser l'usage de la programmation

> 4- Fournir au concepteur d'application des interfaces graphiques lui permettant de définir les points d'insertion et le contenu des conseils, en minimisant l'usage de la programmation.

Des interfaces minimisant l'usage de la programmation ont été développées et présentées plus haut. Une assez bonne connaissance de la programmation Smalltalk est encore nécessaire pour les utiliser. De plus, l'intégration des bases de règles nécessite un connaissance du formalisme de NéOpus. Il apparaît difficile d'éliminer complètement l'usage de la programmation, toutefois il serait possible d'en réduire davantage l'usage dans les cas les plus usuels, notamment en programmant plus de classes relatives aux parties anatomiques d'un agent. D'autre part, l'interface usager offerte au concepteur devrait pouvoir être améliorée sur le plan ergonomique. Il serait également intéressant d'offrir des outils permettant de manipuler et fusionner des graphes de description des tâches.

5.5 Améliorations sur le plan technique

Sur le plan technique, un certain nombre d'améliorations seraient utiles:

—Pour l'instant tout le système est synchrone. Il serait utile de désynchroniser les agents au fur et à mesure de leur développement.

—La destruction des agents est un problème compliqué en général. Un agent peut être connecté à plusieurs outils et il est difficile de savoir a priori si tous les outils qu'un agent est censé espionner sont fermés. Pour le moment, on a choisi de ne pas supprimer les agents conseillers créés au cours d'une session. En effet, un agent conseiller qui ne reçoit aucune information ne fera rien et ne perturbera donc pas le déroulement du système. Seule la fermeture de la fenêtre générale du système (la fenêtre dite de l'écosystème) déclenche la destruction des agents et des espions.

En conclusion, l'architecture ÉpiTalk s'avère un outil puissant pour développer rapidement des systèmes espions en général, et pas simplement des systèmes conseillers. Des applications en dehors du contexte de la formation sont envisagées, en particulier le deverminage de programmes dans les langages d'acteurs [Giroux & al 94].

Bibliographie

[Anderson 1984] J.R. Anderson, C.F. Boyle, R.G. Farrell and B.J. Reiser. *Cognitive principles in the design of computer tutors.* Proceedings of the Sixth Cognitive Science Society Conference, Boulder Colorado, 1984.

[Brahan 1992] J.W Brahan, B. Farley, R.A. Orchard, A. Parent, C.S. Phan. *A Designer's Consultant*, Technical Paper #NRCC33234, Institute for Information Technology, National Research Council, Ottawa, 1992.

[Briot 1989] J.P. Briot *Actalk: A test Bed for Classifying and Designing Actor Languages in the Smalltalk-80 environment.* ECOOP '89, pp. 109-130.

[Brown 1975] J.S. Browns, R.R. Burton, A.G. Bell. *SOPHIE: a step towards a reactive learning environment.* Int Jrnl Man-machine studies, vol. 7, pp 675-696.

[Gasser 1991]. L. Gasser. *Social Conceptions of Knowledge and Action: DAI Foundations and Open Systems Semantics.* Artificial Intelligence, 47, pp. 107-138, 1991.

[Giroux 1993] Sylvain Giroux. *Systèmes et agents : une nécessaire unité.* Thèse de doctorat de l'Université de Montréal. Août 1993.

[Giroux & al 94] S. Giroux, F. Pachet & J. Desbiens. *Debugging multi-agent systems: a distributed approach to events collection and analysis*. Canadian Workshop on Distributed Artificial Intelligence - CWDAI '94. Banff, Alberta, Canada, mai 1994.

[Jonassen 1991] D.H. Jonassen, S. Grabinger. Instructional Design and Development Adviser: Intelligent Job Aid, Help Sytem, and Intelligent Authoring System. Division of Instructional Technology, University of Colorado, Denver, 1991.

[Pachet 1992] François Pachet. *Représentation de connaissances par objets et règles : le système NéOpus.* Thèse de l'Université Paris 6. Paris, Septembre 1992.

[Pachet 1995] François Pachet. *On the Embeddability of Production Rules in Object-Oriented Languages*. Journal of Object-Oriented Programming, 1995, à paraître.

[Pachet & al 1994] François Pachet, Sylvain Giroux, Gilbert Paquette. *Pluggable Advisors as Epiphyte Systems*. Conférence CALISCE '94, Paris, 31 août- 2 septembre 1994.

[Paquette 1991] Gilbert Paquette. *Métaconnaissance dans les environnements d'apprentissage*. Thèse de doctorat de l'Université du Maine (Le Mans), Octobre 1991.

[Paquette 1992a] G. Paquette. *Metaknowledge in the LOUTI developement system.* Proceedings of CSCSI-92, Canadian Society for Computational Study of Intelligence, Vancouver, mai 1992.

[Paquette 1992b] G. Paquette. *An Architecture for Knowledge-based Learning Environments.* Expersys-92, Paris, octobre 1992.

[Paquette 1993a] G. Paquette, G. Bergeron et J. Bourdeau, *The Virtual Classroom revisited*, Conference TeleTeaching'93, Trondheim, Norvège, août 1993.

[Paquette 1993b] G.Paquette , C. Aubin, J. Bourdeau, F. Crevier, C. Paquin, D. Ruelland. *Modélisation des connaissances de design pédagogique dans un atelier de génie didactique (AGD),* Rapport de recherche du LICEF, Télé-université, décembre 1993.

[Shute 1986] VJ. Shute and J.G. Bonar. *An intelligent tutoring system for scientific inquiry skills.* Prodeedings of the Eighth Cognitive Science Society Conference, Amherst , Massachusetts, 1986.

[Stevens 1977] A.L.Stevens and A. Collins. *The goal structure of a Socratic Tutor,* Proceedings of the national ACM Conference, Seattle, Washington, 1977.

[Steven 1983] A.L. Stevens, B. Roberts, L. Stead. *The use of a sophisticated interface in computer-assisted instruction.* IEEE Computer Graphics and Applications, vol. 3, March/April, pp. 25-31, 1983.

[Vivet 1991] M. Vivet, *Knowledge Based Systems for Educations Taking in account the Learner's Context*, Proceedings of PEG-91, Gène, May 1991.

[Wenger 1987] Etienne Wenger, *Artificial Intelligence and Tutoring Systems, Computationnal and Cognitive Approaches to che Communication of Knowledge*, Morgan Kaufmann, Los Altos, USA, 1987, 486 pages.

[Winkels 1992] R. Winkels. *Explorations in Intelligent Tutoring and Help.* IOS Press, Amstedam 1992, 227 pages.

Notices biographiques

Gilbert Paquette, est Professeur à la Télé-Université à Montréal où il dirige le Laboratoire de recherche en informatique cognitive et environnements de formation (LICEF). Il a soutenu une thèse de doctorat au LIUM de l'Université du Maine: "Métaconnaissances dans les environnements d'apprentissage". Pionnier du mouvement LOGO au Québec, il a animé les projets LOUPE et LOUTI en IA et éducation et, plus récemment, le projet CAMPUS VIRTUEL portant sur les méthodes et les outils de télé-apprentissage.

François Pachet est Maître de conférences à l'Université Pierre et Marie Curie et poursuit des recherches au LAFORIA en intelligence artificielle et programmation par objets. Il a soutenu une thèse de doctorat à cette université: "Représentation de connaissances par objets et règles: le système NéOpus".

Sylvain Giroux est chercheur au LICEF où il poursuit des recherches sur la programmation par éco-systèmes et ses applications au conseil sur les activités coopératives. Il a soutenu une thèse de doctorat à l'Université de Montréal: "Systèmes et agents: une nécessaire unité".