Vers une acquisition coopérative des connaissances acquises par l'expérience

Serge Tadié Guepfu[1], Claude Frasson[1], Bernard Lefebvre[2].

[1]Université de Montréal , Case postale 6128, succursale Centre-Ville, Montréal (Québec), H3C 3J7 CANADA

[2]Université du Québec à Montréal, Case postale 8888, succursale Centre-Ville, Montréal (Québec), H3C 3P8, CANADA

Adresse électronique : tadie@iro.umontreal.ca, frasson@iro.umontreal, lefebvre.bernard@uqam.ca.

Résumé: *L'acquisition des connaissances est le processus le plus important dans la construction d'un système à base de connaissances. Dans les STI (Systèmes Tutoriels Intelligents) il est l'un des éléments de base dans la réussite de la formation. Cet article met l'accent sur l'acquisition des connaissances expérimentales. Pour ce faire, nous faisons coopérer un agent (social) appelé co-expert avec l'expert humain dans la construction d'un scénario d'une tâche suivant le modèle MONACO_T (connaissances structurelles). L'agent co-expert utilisé intègre des capacités de réactions face aux actions de l'expert et de raisonnement sur ses propres interventions. Il connaît l'expert grâce au modèle de l'usager et est donc capable de faire des interventions appropriées à son interlocuteur. Les consensus résultant du dialogue entre les deux experts vont constituer les connaissances expérimentales qui seront couplées aux connaissances structurelles pour former une base de connaissances riche et adaptée pour les STI*
Mots clés: agent social, connaissances expérimentales, connaissances structurelles, STI.

Abstract: *Knowledge acquisition is the most important part in the building of a knowledge base system. In the ITS(Intelligent Tutoring System), it is a basic element of a good training. In this paper, our aim is the acquisition of experimental knowledge. To do this, we use a social agent called the co-expert who uses the execution of scenario to build experimental knowledge. The task model we use is MONACO_T (structural knowledge). Our co-expert hase three levels of behavior . He is a reactive agent, a cognitive agent and a social agent. The discussion betwen the two agents gives us the way to collect experimental knowledge which is not included in a model of structured knowledge like MONACO_T. This two kind of knowledge give a powerful knowledge base to an ITS.*
Keys words: social agent, expérimentals knowledges, structural knowledge, ITS

1. Introduction.

L'acquisition des connaissances peut-être définie comme la transformation et le transfert des connaissances qui se trouvent dans le monde réel vers l'ordinateur. C'est l'une des phases clés dans le cycle de vie d'un système à base de connaissances. De nombreux systèmes ont déjà été développés: KADS (Wielinga et al., 1993), MOLE (Eshelman et McDermott, 1988), SALT (Marcus et al., 1985), ACTE (Charlet, 1992). Ces différents outils permettent de construire les connaissances du domaine, les structures de tâches, des scénarios d'exécution. Ces différentes connaissances sont surtout adaptées pour les systèmes experts et les environnements de simulations.

Lorsque la base de connaissances est destinée à un STI, les connaissances dont on a besoin vont au-delà de celles produites par les systèmes d'acquisition classique. Dans un STI, le tuteur doit être capable d'évaluer les solutions que lui proposerait tout apprenant. Pour cela, les connaissances que possède le STI doivent inclure plusieurs scénarios d'exécution, des

exceptions dans les exécutions et les raisons de ces exceptions. Ces connaissances sont généralement des connaissances acquises par l'expérience.
Dans la suite nous allons nous situer dans le cas des connaissances de type tâche. Dans ce cas, nous allons appeler connaissances structurelles l'ensemble composé des connaissances du domaine, de la structure de la tâche et le scénario type proposé par l'expert. Par contre les connaissances acquises par l'expérience seront, les scénarios alternatifs, les exceptions de comportement et leurs raisons, les scénario-erreur (scénarios avec quelques erreurs classiques). Les connaissances acquises par l'expérience sont généralement implicites dans l'esprit de l'expert. L'expert doit donc faire des introspections du processus mental afin de fournir ces connaissances, ce qui est très difficile (Nisbett & Wilson 1977). L'explicitation des connaissances implicites nécessite des analyses profondes et le succès de l'acquisition de ces connaissances dépend de l'habilité de communication du cogniticien comme c'est le cas lors d'une acquisition avec un cogniticien humain (LaFrance, 1987).
Le travail que nous avons effectué va dans la même lancée que ceux de Boose et Gruber (Boose et Bradshow, 1987). Le but de ce travail est de trouver un moyen qui permet à l'expert d'expliciter les connaissances qu'il a acquises par l'expérience. Pour atteindre ce but, nous allons faire coopérer l'expert humain avec un co-expert simulé. Ce choix est motivé par le fait que, dans la réalité, lorsque deux experts d'un même domaine discutent, ils sont amenés à faire des introspections qui leur permettent de justifier leurs points de vue. Le résultat obtenu est donc plus riche que si l'on avait un seul expert.
Dans une discussion d'experts, on peut avoir le dialogue suivant:

- *Expert A: Pour résoudre le problème P1, il faut exécuter l'action A1.*
- *Expert B: Oui c'est bien vrai mais, en exécutant l'action A2, on obtient de meilleures solutions pour le problème P1 car...*
- *Expert A: Effectivement, ça me revient, il faut effectivement exécuter l'action A2 pour avoir de meilleurs résultats pour le problème P1.*

Cette discussion entraîne donc les experts dans un processus d'introspection, ce qui leur permet de ressortir le maximum de connaissances implicites qu'ils possèdent
Le co-expert que nous mettons dans notre outil d'acquisition de connaissances doit donc nous permettre de créer cet environnement de discussion déclencheur des processus d'introspection chez l'expert humain. Nous allons trouver un moyen de le construire afin qu'il soit crédible vis-à-vis de l'expert humain sans pour autant nécessiter une quantité importante de connaissances.
Les connaissances acquises par l'expérience s'appuyant sur des connaissances structurelles, nous allons modéliser les connaissances structurelles suivant le modèle MONACO_T (Tadié et al., 1996). Ce modèle permet de représenter les tâches coopératives. Pour simplifier le problème, nous allons supposer que le scénario coopératif est construit par un expert qui spécifie le comportement au niveau de chaque poste de travail.
Dans cet article, l'agent co-expert et l'expert vont construire des scénarios d'exécution de manière coopérative. Avant de décrire le processus de coopération, nous allons faire une brève présentation de MONACO_T.

2. Bref aperçu de MONACO_T

MONACO_T est un modèle construit pour les tâches coopératives concernant la manipulation de concepts (et non la manipulation de systèmes). Il utilise une approche "orientée tâche" (par opposition à orienté agent), ce qui lui permet d'être adapté pour l'enseignement du travail coopératif (Tadié et al., 1996).
Notre approche de modélisation considère deux couches qui distinguent d'une part les propriétés d'une tâche coopérative et qui constituent ses caractéristiques intrinsèques, et d'autre part le comportement de cette tâche lors de son exécution. Ces deux couches sont respectivement la statique et la dynamique de la tâche (Figure 1):

- la statique représente la structure arborescente de la décomposition de la tâche en sous-tâches; on parle de décomposition statique ou physique,
- la dynamique représente le comportement de la tâche en phase d'exécution avec la génération et l'utilisation d'événements se produisant au niveau de chaque sous-tâche.

Nous précisons dans ce qui suit les caractéristiques de ces deux couches.

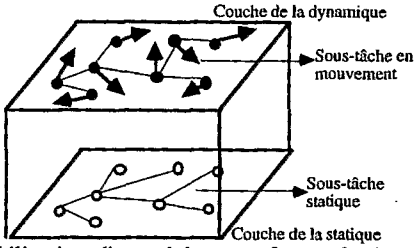

Figure 1 Modélisation d'une tâche sous forme de deux couches

2.1. Description de la statique de la tâche

La plupart des modèles de tâche construite qui permettent de générer des explications sont des cas particuliers du modèle GOMS (Goals, Operators, Methods, and selection Rules). Nous citerons par exemple la structuration des tâches dans SHERLOCK (Lesgold et al., 1992), le modèle de tâche de Kieras ou encore le graphe de tâche dans le projet SAFARI (Djamen,1995). Au niveau de la statique, cette approche considère que:

- une tâche est décomposée en sous-tâches,
- le résultat de la décomposition est une arborescence,
- seules les feuilles de cette arborescence sont les opérations de la tâche (sous-tâches opérationnelles), les autres noeuds de l'arborescence représentant des tâches abstraites (buts ou sous-tâches abstraites)

Dans MONACO_T nous empruntons au modèle GOMS la décomposition d'une tâche en une arborescence de sous tâches. Les différences que notre approche a avec le modèle GOMS sont les suivantes:

- Une tâche n'est pas un but abstrait à atteindre, mais représente une action à réaliser. Lorsque cette tâche est décomposable, son action consiste en la composition des résultats produits par ses sous-tâches.
- Il n'y a pas de notion de sous-tâche abstraite et de sous-tâche opérationnelle (toutes les sous-tâches sont opérationnelles car représentent une action).
- Il y a une prise en compte de la possibilité d'une exécution coopérative de la tâche (par l'introduction de postes de travail au niveau de chaque sous-tâche).

Chaque sous-tâche au niveau de la statique possède un ensemble de propriétés: nom de la sous-tâche, description de la sous-tâche, méthode de réalisation, liste de résultats possibles, méthode d'évaluation, ressource (poste de travail).

Un exemple de décomposition d'une tâche coopérative de diagnostic est présenté à la figure 2. Les sous-tâches dans cette représentation sont des ellipses, et les ressources sont des rectangles.

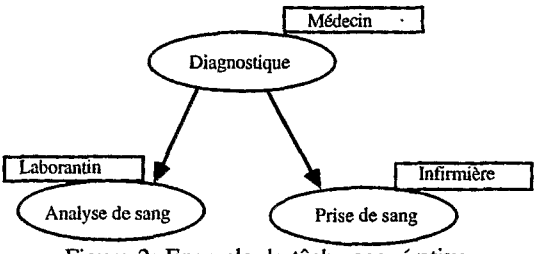

Figure 2: Exemple de tâche coopérative

Cet exemple de tâche coopérative est structuré autour de trois postes de travail: le médecin, l'infirmière et le laborantin. Au niveau de la statique l'une des informations les plus importantes dans le cadre de l'explication est la méthode de réalisation de chaque sous-

tâche. Dans ce cas, nous allons considérer que cette méthode de réalisation est un texte décrivant comment l'action de la sous-tâche est réalisée.

2.2. Description de la dynamique d'une tâche

La dynamique d'une tâche représente la manière dont la tâche évolue en phase d'exécution par un apprenant. Dans les modèles proches de GOMS, cette dynamique est représentée par les règles de sélection. Dans MONACO_T cette dynamique est construite au-dessus de la statique de la tâche et définit comment les conditions dans lesquelles l'action de composition de chaque sous-tâche est réalisée.
Les actions dans MONACO_T sont divisées en trois étapes:

- une étape de *DÉCLENCHEMENT* qui permet de commencer la réalisation de l'action de la tâche,
- une étape de *COMPOSITION (synthèse de résultats)* qui permet d'exécuter l'action proprement dite de la tâche en mettant en oeuvre la méthode de réalisation de la tâche,
- Une étape de *TERMINAISON* qui permet de clore l'exécution de la tâche.

Pour définir la dynamique, trois bases de règles ont été implantées au niveau de chaque sous-tâche de l'arbre de décomposition de la tâche. Une base spécifiant les conditions de déclenchement, une base les conditions de composition et une base les conditions de terminaison. Ces trois bases ressemblent aux préconditions, invariants et postconditions qu'on trouve dans le domaine des preuves de programmes. Ces conditions utilisent des termes qui sont des tests sur certains états caractéristiques des sous-tâches de la tâche. Les variables définissant l'état caractéristique d'une sous-tâche sont les suivantes: *état* de la sous-tâche, *résultat* de la sous-tâche, *conditions de déclenchement*, *conditions de composition*, *conditions de terminaison.*

Exemple

Nous reprenons l'exemple de la figure 2, et nous y introduisons les règles définissant la dynamique de la sous-tâche diagnostique.

Dynamique de la sous-tâche Diagnostic.
Condition de déclenchement: État diagnostic = initial
Condition de synthèse: État diagnostic = déclenché ET
État analyse de sang = terminé .
Condition de terminaison: État diagnostic = réalisé.

Chaque base de règles définissant une condition est enregistrée sous forme de disjonction de conjonction de termes.
Le graphe de la dynamique ainsi construit est isomorphe au graphe de la statique. En pratique les deux graphes théoriques n'en forment qu'un où les sous-tâches contiennent les informations de la statique et les règles qui définissent la dynamique.

3. Architecture du module d'acquisition coopérative des connaissances.

L'architecture du module d'acquisition des connaissances expérimentales que nous mettons en oeuvre est orientée agent. Ce choix implique que tous les composants sont considérés comme étant des agents. Le schéma de notre architecture (figure 3) est composé de trois agents qui dialoguent dans le but de construire les connaissances acquises par l'expérience.
Les agents principaux de cette architecture sont: l'agent tâche, l'agent co-expert et l'expert humain. Nous avons en plus un agent modèle de l'usager qui permet de représenter le profil de l'expert humain afin que le co-expert puisse adapter ses interventions.

3.1. Rôle des différents composants

- L'agent co-expert: il est chargé de jouer le rôle d'un expert et doit initier des discussions avec l'expert réel, afin d'amener ce dernier à faire des introspections nécessaires pour enrichir les connaissances expérimentales devant être rattachées à la tâche courante.
- L'agent tâche: il est chargé de présenter la tâche à réaliser à l'expert. Au fil de l'exécution de cette tâche par l'expert, il sélectionne les actions disponibles ou réalisables à

chaque étape de l'exécution. Cette sélection est basée sur les connaissances structurées par MONACO_T. Il vérifie également que les choix faits par l'expert ne sont pas en contradiction avec les connaissances structurelles de la tâche.

- L'agent expert: Il est l'humain qui possède les connaissances acquises par l'expérience. Cette personne est chargée tout au long du processus d'acquisition de résoudre une tâche et d'expliciter éventuellement ses choix.
- Base de Connaissances expérimentale: Elle comprend le résultat des différents consensus d'amélioration des connaissances structurelles qu'il y a eu entre l'expert et le co-expert. Les consensus ainsi récoltés sont ceux qui ont un impact réel sur le résultat de l'exécution de la tâche.
- L'agent modèle de l'usager: il comprend les informations permettant d'individualiser les interventions auprès des experts humains. Il dialogue principalement avec l'agent co-expert. Ce dialogue consiste à transmettre au co-expert le profil de l'expert humain.

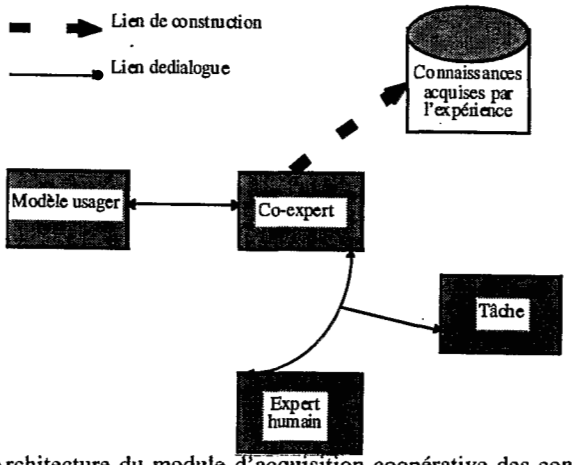

Figure 3: Architecture du module d'acquisition coopérative des connaissances

3.2. Technique d'acquisition utilisée.

La technique d'acquisition fait référence à celle utilisée par le co-expert pour acquérir les connaissances de l'expert. Pour cela , le co-expert doit d'une part poser des questions et d'autre part fournir les moyens de réponses. La technique d'acquisition mise en place ici est constituée de deux phases: exécution de la tâche et explicitation des choix. L'exécution de la tâche se fait par l'activation d'une succession d'actions. Lorsque le co-expert detecte une contradiction entre ses pensées et le choix de l'expert il déclenche le dialogue d'explicitation. Les étapes du processus d'acquisition mise en oeuvre par le co-expert sont les suivantes:

- mettre l'expert en situation de résolution de tâche,
- contrôler les actions de l'expert,
- Dès qu'il y a une divergence de point de vue, déclencher le processus de consensus qui doit permettre à l'expert de faire l'introspection nécessaire pour explicitation.
- transformer le résultat du consensus obtenu sous forme computationnelle (construction de la base expérimentale).

4. Architecture du co-expert.

Le co-expert doit posséder des aptitudes lui permettant d'une part d'analyser de manière intelligente les actions de l'expert et d'autre part de conduire un dialogue cohérent permettant à l'expert d'expliciter ses connaissances. Pour permettre au co-expert de

manifester ces comportements intelligents, nous l'avons structuré en trois modules (Figure 4) : une mémoire à long terme, un module de raisonnement et un module de dialogue.
La mémoire à long terme contient toutes les règles qui permettent au co-expert de réagir aux actions de l'expert.
Le module de raisonnement permet au co-expert de controler son propre raisonnement et celui de l'expert.
Le module de dialogue permet au co-expert de converser avec l'expert et d'extraire les explicitations utiles dans le cadre de STI.

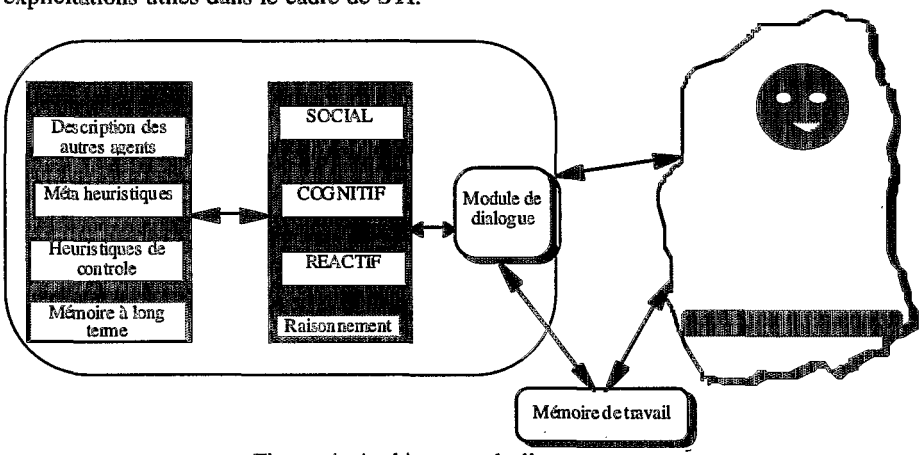

Figure 4: Architecture de l'agent co-expert

À partir de cette structure, le comportement du co-expert se divise en trois processus distincts (Figure 5): le méta-contrôle, le contrôle et le consensus.

4.1. Le processus de méta-contrôle

Il est mis en oeuvre grâce au niveau cognitif du raisonnement. Ce processus permet au co-expert de raisonner sur son propre comportement. Le co-expert peut ainsi améliorer ses performances au cours du dialogue avec l'expert. Les connaissances exploitées pour mettre en place ce processus sont les méta-heuristiques. Les méta-heuristiques sont des heuristiques sur des heuristiques. Ils exploitent les facteurs de rentabilités attribués aux heuristiques de contrôle. Ces facteurs de rentabilités permettent de définir des contraintes d'ordre explicites parmi les heuristiques de contrôle. L'idée d'utiliser des caractéristiques de contrôle abstraites pour raisonner sur le contrôle a été développée dans la littérature [Clancey, 1983a; Clancey & Bock, 1988; Hayes-Roth, 1985; Wesley, 1983].
Une méta-heuristique peut être: « utiliser les heuristiques dont le facteur de rentabilité est le plus élevé ».
Le concept de méta-Heuristique dont nous parlons est similaire au concept de méta-règle que Clancey a développé dans son travail sur le système NEOMYCIN [Clancey, 1988; Shortliffe, 1984].

4.2. Le processus de contrôle

Il est mis en oeuvre grâce au niveau réactif du raisonnement. Il permet à l'agent de détecter des failles dans le raisonnement de l'expert et de déclencher le processus de consensus. Ce processus est régulé par la base des heuristiques de contrôle que possède le co-expert. Les heuristiques de contrôle sont définies à partir des connaissances que l'on a sur le domaine visé et sur le modèle utilisé. Dans notre cas, le domaine représente les tâches dans une salle de soins intensifs et le modèle est MONACO_T.
Un exemple d'heuristique basé sur le modèle : « si on a le choix entre déclencher un noeud A ou synthétiser un noeud B, il est préférable de commencer par synthétiser le noeud B ».

Plus généralement on peut classer les trois types d'actions de MONACO_T par ordre de priorité d'exécution de la manière suivante: « déclencher < synthétiser < termine ».
Un exemple d'heuristique basé sur le domaine : « lorsque le médecin et l'infirmière peuvent exécuter une action, il est préférable de commencer par celle du médecin ». C'est le processus de méta-contrôle qui choisit l'heuristique qui est utilisée pour contrôler l'activité de l'expert

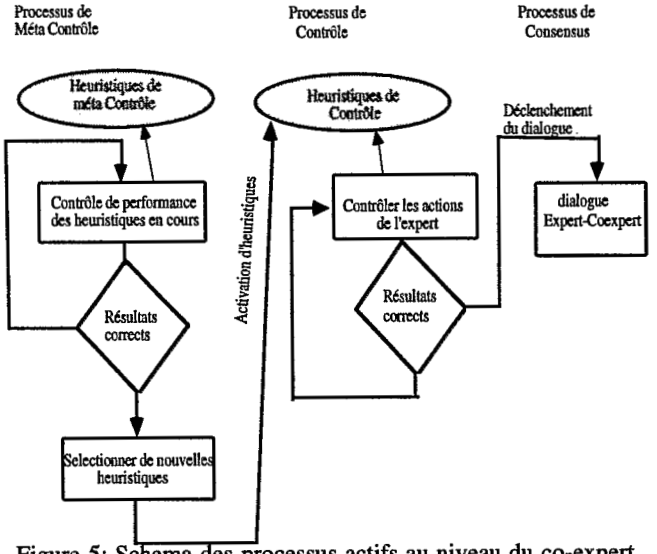

Figure 5: Schema des processus actifs au niveau du co-expert

4.3. Le processus de consensus

Il est mis en place par le module de dialogue et permet au co-expert de discuter avec l'expert dans le but d'arriver à un consensus. Ce consensus permet d'enrichir la base de connaissances expérimentales. Ce processus est l'un des plus importants car c'est lui qui permet à l'expert d'expliciter ses connaissances. Ce processus est basé sur un dialogue expert co-expert. Ce dialogue a deux formes: une forme non productive et une forme productive.
Dialogue non productif: Un dialogue est non productif lorsque le consensus ne permet pas d'enrichir la base de connaissances expérimentale. Dans notre cas le dialogue est non productif si l'expert juge que l'alternative que lui propose le co-expert est équivalente à la sienne. Dans ce cas le dialogue est constitué d'une seule étape (Proposition du co-expert)
Dialogue productif: Le dialogue est productif lorsque l'expert juge que l'avis du co-expert n'est pas équivalent au sien. Dans ce cas, le dialogue est structuré en trois étapes (Proposition du co-expert, justification de l'expert et contexte de validation)

Le processus démarre dès que le co-expert détecte une violation par rapport à l'heuristique qui guide son niveau réactif. L'initiative dans ce dialogue est laissée au co-expert. Le co-expert commence par déclencher l'étape « proposition du co-expert ». Suivant la réaction de l'expert, le co-expert peut enclencher l'étape « justification de l'expert », ensuite il déclenche l'étape « contexte de validation ».

a) Proposition du co-expert

La proposition du co-expert consiste à présenter à l'expert une alternative à l'action qu'il vient d'effectuer (Figure 6). ensuite le co-expert demande l'avis de l'expert par rapport à cette proposition. Afin que la réponse de l'expert puisse être exploitable nous l'avons limité

à quatre choix possibles (Meilleur, équivalent, moins bon, inacceptable). C'est le choix que va faire l'expert qui va permettre d'évaluer l'heuristique en cours et de déclencher éventuellement les autres étapes de l'explicitation.
Si l'expert dit que le choix est équivalent alors le dialogue s'arrête pour que l'expert continue à réaliser la tâche. Dans le cas contraire, l'expert engage l'étape de justification et celle du contexte de validation.

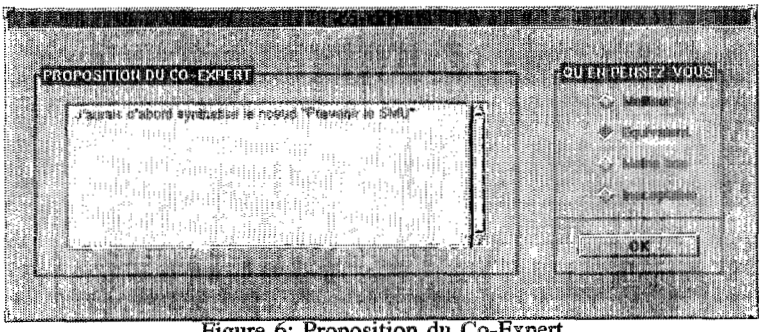

Figure 6: Proposition du Co-Expert

C'est également au niveau de cette étape que le co-expert donne des facteurs de rentabilité aux heuristiques qu'il utilise. Cette notation est basée sur la production des dialogues. Un dialogue est productif lorsque la proposition du co-expert n'est pas équivalente à celle de l'expert. Au départ, toutes les heuristiques de contrôle ont un facteur de rentabilité qui vaut zéro. Lorsqu'une heuristique permet de déclencher une discussion productive avec l'expert, elle gagne des points de rentabilité et elle en perd si la discussion déclenchée n'est pas productive.

b) Justification de l'expert

L'expert se justifie lorsqu'il est en désaccord avec le co-expert. La justification qu'il donne est enregistrée sous forme de texte (Figure 7). Ce texte sera utilisé tel quel lorsqu'on exploitera les connaissances acquises pour faire un enseignement

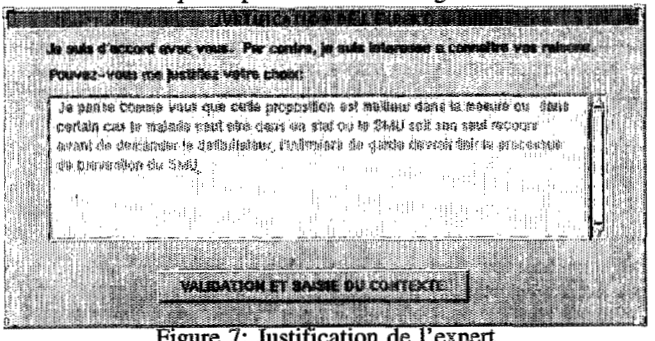

Figure 7: Justification de l'expert

Dans la mesure où cette remarque peut être valable uniquement dans certains cas, le co-expert demande à l'expert de préciser le contexte dans lequel cette appréciation est valable.

c) Contexte de validation

Le contexte ici est un élément très important dans le processus de consensus entre l'expert et le co-expert. Il définit les états de la tâche où l'avis de l'expert est valable.
Les paramètres permettant de définir ce contexte sont les suivants:

- Les noeuds d'un graphe à la MONACO_Tâche.
- Les différents états possibles de ses noeuds.

• Les états par lesquels la tâche est passée avant que l'action soit exécutée. Ce paramètre représente l'évolution temporelle de la tâche depuis son état initial.

La définition du contexte permet d'élargir le champ d'action du résultat du consensus qu'il y a eu entre l'expert et le co-expert. Initialement le contexte de validité est l'état de la tâche lorsque le consensus est survenu. Ce contexte peut-être ensuite élargi à d'autres états de la tâche.

Dans le cadre du prototype que nous avons mis en place, nous avons uniquement pris en compte les deux premières dimensions du contexte. Ce contexte est donc saisie (Figure 8) sous forme d'une base de règle. L'expert saisit les conditions qui doivent être vérifiées pour que l'expertise donnée soit valable.

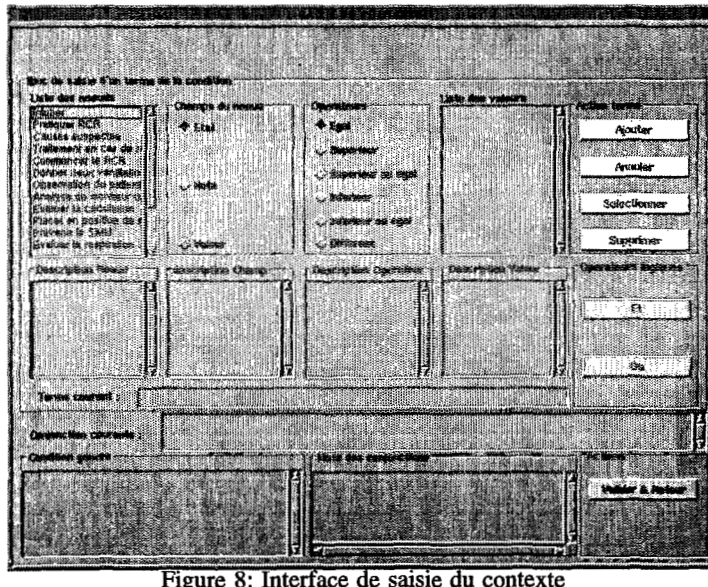

Figure 8: Interface de saisie du contexte

Ce contexte représente les conditions de validité des connaissances acquises par l'expérience que nous avons enregistrées. Cette condition de validité est utilisée comme déclencheur lorsqu'un apprenant se trouve en phase de résolution de tâche.

Conclusion:

L'outil d'acquisition des connaissances que nous avons construit, basé sur l'utilisation d'un agent co-expert, fournit un moyen de communication entre l'expert et le co-expert permettant à l'expert de décrire ou de démontrer l'expertise et au co-expert de construire une base de connaissances expérimentales pour réaliser la tâche désirée. Puisque les connaissances sont souvent tacites et difficiles à obtenir, le moyen par lequel on extrait les connaissances de l'expert peut déterminer de façon significative le succès du module d'acquisition des connaissances. Notre outil ne demande pas à l'expert d'écrire des procédures compliquées, mais simplement de résoudre un problème et de discuter des éventuelles alternatives qui pourraient se présenter.

L'agent co-expert qui est chargé du principal rôle dans cet outil est un agent social dans la mesure où:

il est réactif: déclenche un dialogue en fonction des actions de l'expert,

il est cognitif: analyse son propre comportement afin d'améliorer ses interventions,

Enfin il possède une certaine connaissance sur l'expert grâce au modèle de l'usager.

Un bon outil d'acquisition des connaissances est un programme interactif qui extrait de l'expert les connaissances dont il a besoin, en fournissant un cadre qui permette à ce dernier

de répondre facilement et de rester attentif. Le concepteur de l'interface de l'assistant doit balancer deux objectifs complémentaires:

- contraindre la façon dont l'expert introduit l'information et
- Permettre à l'expert d'avoir le contrôle sur le système.

Pour réduire les erreurs et pour être certain que le système comprenne la réponse, nous avons décidé de donner le contrôle du dialogue au co-expert. On ne laisse donc pas à l'expert la possibilité de prendre l'initiative malgré le fait qu'il soit le mieux placé pour apprécier l'importance d'une information. Sachant que l'intérêt de l'expert peut diminuer, nous lui donnons le dernier mot dans le processus de consensus que le co-expert déclenche. Cette caractéristique correspond à la réalité qui veut que l'homme soit meilleur dans les propositions et l'ordinateur dans la critique.
Les connaissances acquises par cet outils permettrons à un STI de donner de riches conseils et critiques à tout apprenant participant à un processus d'enseignement.

Remerciement: Ce travail est un volet du projet SAFARI et est financé par le Ministère de l'industrie de la Science du Commerce et de la Technologie du QUEBEC que nous remercions.

Bibliographie.

Boose, J. H. & Bradshaw, J. M. (1987). Expertise transfer and complex problems: Using AQUINAS as a knowledge acquisition workbench for expert sysyems. *International Journal of Man-Machine studies, 26(1)*: 21-25.

Carlet, J. (1992) ACTE: acquisition des connaissances par l'interprétation d'un modèle causal. *Revue d'intelligence artificielle*, vol. 6: 1-2,

Chandrasekaran, B. (1987). Towards a functional architecture for intelligence based on generic information processing tasks. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 1183-1192, Milan, Italy .

Clancey, W. J. & Bock, C. (1988). Representing control knowledge as abstract tasks and metarules. *Expert System Applications*, pages 1-77. New York: Springer-Verlag.

Clancey, W. J. (1988). Acquiring, representing and evaluating a competence model of diagnosis. In M. T. H. Chi, R. Glaser & M. J. Farr (Eds.), *The Nature of Exertise*, pages 343-418. Hillsdale, NJ: Lawrence Erlbaum.

Djamen J.Y. (1995). Une architecture de STI pour l'analyse du raisonnement d'un apprenant. Thèse de Ph.D., DIRO, Université de Montréal, Montréal, CANADA, Juin 1995.

Eshelman, L., Ehret, D., McDermott, J., et Tan, M. (1993) MOLE A tenacious knowledge Acquisition tool, Reading in *Knowjedge Acquisition and Learning*, pp. 253-261, Edited by B.G. Buchman & D.C. Wilkins, Morgan Kaufmann Publishers, 1993

Hayes-Roth, F., (1985). A blackboard architecture for control. *Artificial Intelligence, 26*: 251-321.

LaFrance, M. (1987). The knowledge acquisition grid: A method for training knowledge engineers. *International Journal of Man-Machine Studies*, 26(2): 245-236.

Marcus, S. et McDermott (1993) SALT: A Knowledge Acquisition language for "propose-and-revise" system, Reading in *Knowjedge Acquisition and Learning*, pp. 253-261, Edited by B.G. Buchman & D.C. Wilkins, Morgan Kaufmann Publishers, 1993

Nisbett, R. E. & Wilson, T. D. (1977). Telling more then we can know: Verbal reports on mental processus. *Psychological Review, 84*: 231-259.

Tadié, S., Lefebvre, B., Frasson, C. (1996) MONACO_Tâche un modèle à nature coopérative pour la modélisation de tâche. À apparaître dans les proceeding de la 3ème conférence internationale sur les systèmes tutoriels intelligents.

Wesley, L. P. (1983). Reasoning about control: The investigation of an evidential approach. *Proceedings of the Eighth International Joint Conference On Artificial Intelligence*, pages 203-206, Karlsruhe, Federal Republic of Germany.

Wielinga, B.J., Schreiber, A.T., Breuker, J.A., (1993) KADS: A modelling approach to knowledge engineering, Reading in Knowledge Acquisition and Learning, pp. 92-117, Edited by B.G. Buchman & D.C. Wilkins, Morgan Kaufmann Publishers