Université de Montréal

Étude et réalisation d'un système d'aide à la modélisation des tâches

par
Daniela Suzana Lupascu

Département d'informatique et de recherche opérationnelle

Faculté des Arts et des Sciences

Mémoire présenté à la Faculté des Études Supérieures
en vue de l'obtention du grade de
Maître ès Sciences (M.Sc.)
en informatique

Août, 1997

Université de Montréal
Faculté des Études Supérieures

Ce mémoire intitulé:

Étude et réalisation d'un système d'aide à la modélisation des tâches

présenté par:

Daniela Suzana Lupascu

A été évalué par un jury composé des personnes suivantes:

Marc Kaltenbach	président-rapporteur
Claude Frasson	directeur de recherche
Esma Aïmeur	codirectrice
Yoshua Bengio	membre

Mémoire accepté le: 23 janvier 1998

Sommaire

Dans la réalisation des Systèmes Tutoriels Intelligents (STI), l'acquisition des connaissances est un problème central. Ceci comprend deux activités indissociables: la modélisation et l'explicitation des connaissances. La problématique particulière que nous abordons dans ce travail se concentre sur la manière d'expliciter des connaissances par coopération avec un expert du domaine. Plus précisément, nous nous focaliserons sur le problème de l'explicitation d'une base de tâches initiale, base qui pourra ultérieurement servir de support au recueil d'autres types de connaissances. Ainsi, nous avons défini le système COSI (COgniticien SImulé) qui est un outil d'aide à l'explicitation des connaissances de type tâche. Il utilise un modèle de tâches générique qui sert de fil conducteur pour guider le processus d'explicitation des connaissances. Pour faciliter le travail de l'expert, COSI sait adapter le dialogue compte tenu des caractéristiques et des préférences de l'individu qui l'utilise. COSI offre ainsi à l'expert un environnement de travail convivial. Ce travail a été mené dans le cadre du projet SAFARI consacré à l'étude et à la réalisation de systèmes tutoriels intelligents. Le document se divise en deux grandes parties. La première effectue une étude des idées majeures en acquisition des connaissances. La seconde décrit l'outil COSI et son intégration au sein de l'environnement de

développement de logiciels éducatifs SAFARI. COSI a été expérimenté sur plusieurs exemples. Son utilisation dans le cadre d'une application médicale est présentée en fin de mémoire. Pour terminer, nous présentons quelques possibilités de développement futur pour améliorer le logiciel implémenté.

MOTS-CLÉS : *acquisition des connaissances, modélisation, explicitation, systèmes tutoriels intelligents, modèle de tâche, dialogue humain-machine.*

Remerciements

Je remercie mon directeur, Monsieur le professeur Claude Frasson, ainsi que mon codirecteur, Madame la professeure Esma Aïmeur. Leur soutien et leurs conseils précieux ont constitué une aide indispensable dans la réalisation de ce travail. En outre, Monsieur Claude Frasson m'a offert un excellent cadre de travail tout au long des deux années d'études. Mes remerciements s'adressent également au Ministère de la Science et de la Technologie pour l'aide financière qu'il m'a accordée.

Je remercie toute l'équipe: chercheurs, professeurs, employés et étudiants impliqués dans les projets SAFARI et TELELEARNING, pour le climat chaleureux qui y règne. Ces deux ans de travail au sein de cette équipe ont constitué pour moi une riche école. En particulier, je remercie les personnes qui se sont intéressées de près à mon travail, notamment Serge, Thierry, Roger et Carmen.

J'accorderai une mention particulière à Thierry pour l'aide qu'il m'a apportée durant la rédaction de ce mémoire. Son amitié et son effort pour corriger les versions de ce mémoire m'ont aidé à arriver au bout. Je ne peux m'empêcher de penser à Serge pour l'aide qui m'a apporté et son soutien depuis le début de ce travail qui n'aurait jamais

pu être réalisé sans lui. Je tiens à remercier Roger pour être toujours disponible pour donner un coup de main quand on en a le plus besoin.

Je remercie Madame le médecin Naîla Zalila avec qui j'ai travaillé plusieurs mois. Son intérêt pour l'informatique et les nombreux entretiens que nous avons eus m'ont permis d'améliorer mon travail.

Je remercie Martine pour la disponibilité dont elle a fait preuve chaque fois que j'ai eu besoin de son aide.

Je tiens à remercier ma mère et mon père qui, malgré la distance, m'ont soutenu de leur confiance durant mes études.

Je remercie tous mes amis de près et de loin: ceux qui ont partagé avec moi les bons et les mauvais moments.

Enfin, je remercie Ciprian pour tout.

À mes parents...

Table des matières

Sommaire ..... **i**
Remerciements ..... **iii**
Tables des matières ..... **vi**
Liste des figures ..... **ix**
Liste des tableaux ..... **xi**

Chapitre 1: Introduction **1**

Chapitre 2: Les Systèmes Tutoriels Intelligents **5**
2.1 Genèse des STI 5
2.2. Pourquoi est-il important d'aborder le thème de l'acquisition dans le domaine des STI? 7
2.3 Un approche de production de STI: le projet SAFARI 9

Chapitre 3: Acquisition des connaissances **10**
3.1 Qu'est ce qu'une expertise ? 10
3.2 L'évolution des recherches : "transfert" des connaissances vers modélisation des connaissances ... 11
3.3 Le "knowledge level" et les notions en découlant 13
3.3.1 Le "knowledge level" ou "niveau connaissance" 13
3.3.2. Les notions qui découlent du terme knowledge level: modèle, tâche, méthode 15
3.4 L'explicitation des connaissances 17

Chapitre 4: Modélisation du processus de résolution de problèmes **20**
4.1 L'école "choisir pour suivre" 21
4.1.1 L'approche "tâches génériques" 22
4.1.2 Les méthodes à limitation de rôles 24
4.1.3. L'approche "de la méthode à la tâche" 28
4.1.4. La méthodologie KADS 29
4.2 L'école "blocs de construction" 32
4.2.1. Les Composants de l'expertise 32

4.2.2.PROTEGE II 34
4.3. Une synthèse des systèmes d'aide à la modélisation 35
4.3.1 TERESIAS: un "pionnier" des systèmes d'aide à l'acquisition 35
4.3.2. MOLE: un outil basé sur la méthodologie "à limitation de rôles" 36
4.3.3 Open-KADS: un atelier pour rendre opérationnelle l'approche KADS 37
4.3.4 La boîte à outils de "Tâches Génériques" 38
4.3.5 COMMET: un outil de modélisation et d'explicitation des connaissances basé sur l'approche "Composants de l'expertise" 39

Chapitre 5: L'acquisition des connaissances dans le domaine des STI 41
5.1 Contexte de l'acquisition des connaissances dans les STI 42
5.1.1 Les étapes de la construction d'un STI 42
5.1.2 La pluralité de l'expertise dans un STI 43
5.1.3 Le besoin d'assistance à l'utilisateur 44
5.2 Expériences en acquisition des connaissances pour la production de STI 46
5.2.1 L'acquisition des connaissances pour le système DIAPASON 47
5.2.2 GEOMUS: un système de résolution de problèmes de géométrie élémentaire 50
5.2.3 L'acquisition dans les STI: discussion 55
5.3 Comment l'acquisition des connaissances peut-elle résoudre les difficultés rencontrées lors de la réalisation des STI? 56

Chapitre 6: COSI: un assistant d'explicitation de tâches en coopération avec l'expert 59
6.1 L'acquisition des connaissances dans le projet SAFARI 60
6.1.1 Structures des connaissances du curriculum 60
6.1.2 De l'acquisition des connaissances du curriculum 65
6.1.3 Position de COSI au sein de l'architecture SAFARI 68
6.2 COSI: principes de conception 70
6.2.1 Les apports de la psychologie cognitive et de l'ergonomie cognitive à la conception de l'outil COSI 70
6.2.2 COSI - L'approche d'acquisition 72
6.2.3 Principes de conception de COSI : résumé 73
6.3 Conception de COSI 74
6.3.1 La nature des connaissances à expliciter et la méthode de résolution 74
6.3.2 Les stratégies d'explicitation 75
6.3.3 Heuristiques 83
6.4 COSI: Architecture détaillée et exemple de dialogue 84
6.4.1 Architecture de COSI 84

6.4.2 Structure du dialogue 88
6.5 Contribution de COSI 92
6.6 Conclusion 96

Chapitre7: Implantation de l'outil COSI et expérimentation 97
7.1 Implantation du prototype COSI 97
7.1.1 Le modèle MVC: principe de base pour la construction de toute interface SmallTalk 98
7.1.2 Les classes utilisées et développées pour COSI 98
7.1.3 Les écrans d'aide à l'utilisation du système 101
7.2 Une session d'explicitation de connaissances avec COSI 102
7.2.1 Les principales étapes du raisonnement clinique 103
7.2.2 Comment COSI aide-t-il l'expert à analyser le problème "Examen clinique d'un patient" 103

Chapitre 8: Conclusion 115

Bibliographie 119

Table des figures

Figure 2.1: Architecture classique des STI 7
Figure 3.1: Les trois niveaux à travers lesquels se déroule l'acquisition des connaissances 14
Figure 3.2: Modèle du domaine pour une application médicale 16
Figure 4.1: Modélisation des tâches génériques 23
Figure 4.2: La clasification hierarchique. 23
Figure 4.3: Schéma général d'une "méthode à limitation de rôles". 26
Figure 4.4: Réseau causal partiel maladies-symptômes 27
Figure 4.5: KADS: suite de modèles [Wielinga et al., 92] 31
Figure 4.6: L'approche Tâches-Méthodes-Modèles 34
Figure 5.1: Les étapes de conception d'un STI 43
Figure 5.2 : Figure géométrique 51
Figure 5.3: Les sous-figures pour la figure 5.2. 51
Figure 5.4: Configurations caractéristiques et étiquettes de problèmes 52
Figure 5.5: Hiérarchie des étiquettes 52
Figure 5.6: La communication entre l'acquisition des connaissances et la formation de l'élève 57
Figure 6.1: Architecture de CREAM 61
Figure 6.2: Modélisation sur deux niveaux dans SAFARI 62
Figure 6.3: Graphe des tâches pour le traitement de l'embolie pulmonaire (TE). 65
Figure 6.4: Le système d'acquisition dans l'architecture SAFARI 69
Figure 6.5: L'approche d'acquisition de COSI 72
Figure 6.6: Les liens expert-stratégie-tâche 76
Figure 6.7: Architecture de COSI 87
Figure 6.8: Les phases d'explicitation d'une tâche 90
Figure 6.9: La structure du dialogue entre COSI et Expert 91
Figure 7.1: Modèle objet (une partie) de l'application 102
Figure 7.2: Les écrans d'aide attachés à l'interface concernant les informations de l'expert 103
Figure 7.3: Menu principal de l'application 106
Figure 7.4: Identification de l'expert 107
Figure 7.5: Identification de la tâche 108

Figure 7.6: Conseil de COSI concernant la stratégie d'explicitation 108
Figure 7.7: Choix de politique pour la stratégie descendante 109
Figure 7.8: Identification de sous-tâche 110
Figure 7.9: Comportement de la tâche "examen cardio-vasculaire". 111
Figure 7.10: Le graphe pour la tâche "examen cardio-vasculaire" 112
Figure 7.11: L'arbre de scénarios possibles dans l'utilisation de COSI 113

Table des tableaux

Tableau 2.1: Formalismes de représentation utilisés par quelques STI **8**
Tableau 4.1: Les trois "écoles" de l'approche modélisation **21**
Tableau 6.1: Modèles de représentation des connaissances **63**

Chapitre 1

Introduction

"...la connaissance est une capacité de comportement adaptatif dans un environnement; elle ne peut pas être réduite à des représentations de comportement ou de l'environnement"

Clancey, 1989

Comment peut intervenir l'informatique dans une discipline comme l'enseignement qui demande des qualités humaines exceptionnelles? En effet, être pédagogue ne signifie pas seulement maîtriser un savoir mais principalement avoir le talent de se faire comprendre, être capable de transmettre ce savoir à des gens avec des aptitudes différentes qui utilisent des modes de pensée divers.

Depuis la parution des premiers systèmes destinés à l'enseignement individualisé, le champ des recherches consacrées à l'informatique éducative a beaucoup évolué. Nous avons assisté à un changement de philosophie d'ensemble, dû à la convergence de plusieurs disciplines: éducation, intelligence artificielle, psychologie cognitive, etc. Dans ce contexte interdisciplinaire l'appellation de "système tutoriel intelligent" (STI) s'est imposée peu à peu, remplaçant le terme de système d'EAO (Enseignement Assisté par Ordinateur) ou de didacticiel.

Bien que les STI aient, comme les didacticiels, l'ambition commune de mettre en œuvre un programme permettant à un élève d'apprendre, ils s'inscrivent dans une approche plus réaliste. En effet, après la déception causée par l'inadéquation des didacticiels aux enseignements réels, les STI ont pris pour point de départ les hypothèses suivantes:

- un logiciel de formation ne peut remplacer intégralement un pédagogue humain mais il peut jouer un rôle important au sein du processus d'apprentissage. En effet, même si ces systèmes ont pour objectif de simuler (en partie) l'activité pédagogique, ils n'ont pas la prétention de reproduire exhaustivement le processus mental du pédagogue;
- en tant qu'instrument d'enseignement, un logiciel de formation doit satisfaire certains critères pour être utilisable. Cet instrument est "intelligent" dans la mesure où il est capable de satisfaire les besoins de ses utilisateurs. Ceci implique des qualités comme la flexibilité ou encore la possibilité d'adaptation à des contextes d'utilisation différents.

Apprendre à un logiciel à enseigner consiste à doter celui-ci avec suffisamment de connaissances afin de pouvoir notamment répondre aux demandes des élèves. Celles-ci sont de natures diverses: connaissances sur le domaine, sur la pédagogie, sur l'apprenant. Aussi, l'acquisition de toutes ces connaissances est un problème vital pour le succès d'un STI.

Au-delà d'un lexique parfois différent, les deux disciplines que sont l'acquisition des connaissances et l'ingénierie de la construction de STI se rejoignent face au même problème: comment réaliser un système capable de simuler une activité humaine?

L'acquisition des connaissances consiste à extraire et à organiser les connaissances (faits, concepts, heuristiques, méthodes) en vue de construire des systèmes à base de connaissances.

Comme dans les systèmes experts de première génération, dans les STI, l'intérêt sur l'acquisition a surtout visé la représentation des connaissances et plus précisément, la transcription des ces dernières dans un formalisme (règles, frames, etc.) permettant au

système de raisonner. Ceci n'est pas un problème facile à résoudre, étant donné le décalage entre un "savoir", tant qu'il est exprimé par un expert, et ces formalismes de représentation trop informatiques (trop proches du code).

Face à ce problème, les recherches dans l'acquisition considèrent que la solution est de fournir une modélisation adéquate du problème à résoudre, pour réduire l'écart entre la réalité et la représentation de celle-ci en machine.

Aussi, notre objectif est la construction d'un outil d'acquisition pour un STI. Il s'agit essentiellement d'adapter certaines techniques d'acquisition des connaissances pour réaliser une base de connaissances suffisamment riche pour supporter les besoins d'un STI.

Par conséquent, notre approche dans ce mémoire consistera :

1. Dégager un modèle du problème à résoudre portant particulièrement sur les tâches qui correspondent aux étapes de résolution de ce problème.
2. Construire un outil d'acquisition des connaissances relatives à la résolution de problèmes qui permette d'extraire les connaissances de l'expert et de les organiser selon les propriétés du modèle.

Organisation du mémoire

Pour traiter de la problématique précédente, le mémoire est organisé comme suit.

Le prochain chapitre pressente les caractéristiques d'un STI et la manière dont nous avons envisagé la réalisation de ce type de logiciel au sein du projet SAFARI.

Les chapitres trois et quatre introduisent le lecteur au domaine de l'acquisition des connaissances. L'un passe en revue les principales difficultés rencontrées lors de l'explicitation des connaissances; l'autre décrit l'approche que nous jugeons la plus satisfaisante pour résoudre ces problèmes: la modélisation des connaissances.

Le chapitre cinq présente différentes tentatives de mise en œuvre de techniques d'acquisition des connaissances dans les STI. Les apports portent essentiellement sur la modélisation des connaissances dans le cadre d'un tuteur intelligent.

Le chapitre six traite de la conception de notre outil d'explicitation dénommé COSI (COgniticien SImulé). Le chapitre sept présente les détails liés à son implémentation.

Enfin, la conclusion dresse un bilan des apports de nos travaux et énonce différentes perspectives pour les développements futurs.

Chapitre 2

Les Systèmes Tutoriels Intelligents

2.1 Genèse des STI

Un *Système Tutoriel Intelligent* (STI) [Frasson & Gauthier, 90] est un système à base de connaissances dédié à l'enseignement et à la formation d'apprenants.

Les STI ont été conçus pour tenter de pallier deux limites des systèmes d'Enseignement Assisté par l'Ordinateur (EAO), systèmes dont l'ambition était également la réalisation d'un enseignement de qualité comparable à l'enseignement humain. Même si l'ordinateur est un instrument très efficace pour l'activité de formation, la déception suscitée par les systèmes d'EAO, appelés aussi didacticiels, est liée à deux aspects: ces systèmes n'atteignent pas les performances réelles d'un pédagogue, d'une part, et, d'autre part, ils sont dans la plupart des cas mals adaptés aux besoins des apprenants considérés.

Afin d'identifier les causes de l'inefficacité des système d'EAO, regardons tout d'abord la manière dont ceux-ci ont été construits. Les concepteurs de ces systèmes ont longtemps procédé de façon empirique et souvent approximative [Péninou, 93] en utilisant des méthodes de conception par "petits pas". Plus précisément, ces systèmes sont des programmes qui présentent le contenu d'un cours comme étant structuré en plusieurs leçons. Ces dernières sont découpées en scénarios, chacun contenant des scènes

pédagogiques sous forme de d'enchaînements information-question-réponse [Courouble, 83].

L'utilisation des didacticiels a fait ressortir leurs limites, dont deux nous paraissent essentielles [Tadié, 96] [Péninou, 93]:

- la première, est le *manque d'explicitation des connaissance (connaissances du domaine et connaissances pédagogiques) et de leur structuration.* Ceci explique l'incapacité d'un didacticiel à répondre aux problèmes d'un apprenant;
- la deuxième réside dans *la non prise en compte de certaines caractéristiques de l'élève,* aussi bien au niveau cognitif qu'affectif .

Un système capable d'assurer un "enseignement individualisé" (c'est-à-dire un enseignement adapté du mieux possible à chaque apprenant) implique l'introduction de techniques d'intelligence artificielle ainsi que l'utilisation de théories provenant des sciences de l'éducation (puisqu'il s'agit d'enseignement), de la psychologie (adaptation aux traits psychologiques,...) et de l'ergonomie cognitive(aspect interface,...). Ainsi, contrairement à la démarche suivie en EAO, la réalisation d'un STI nécessite notamment une représentation explicite de toutes les connaissances impliquées dans ce processus (connaissances relatives au domaine, à l'apprenant, à l'expert, à la pédagogie, etc.) ainsi que des raisonnements manipulant celles-ci [Mengelle, 95].

L'architecture classique d'un STI doit inclure au moins trois modèles essentiels et une interface de communication avec l'apprenant (figure 2.1) [Nicaud & Vivet, 88]:

- le *modèle du domaine* qui contient les connaissances du domaine à enseigner,
- le *modèle pédagogique* ou *modèle tuteur* décrivant les stratégies d'apprentissage nécessaires pour planifier le travail de l'élève, pour répondre de façon appropriée à ses questions et pour corriger les erreurs de l'apprenant,
- le *modèle de l'apprenant* qui représente l'état des connaissances de l'étudiant.

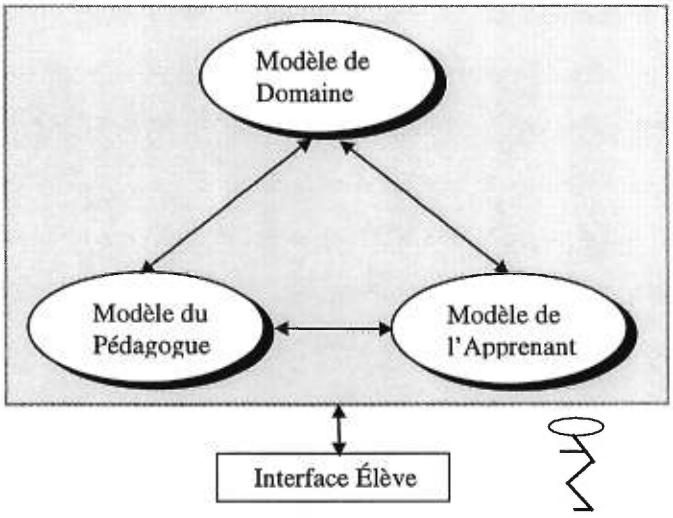


figure 2.1: Architecture classique des STI [Polson & Richardson, 88].

2.2. Pourquoi est-il important d'aborder le thème de l'acquisition dans le domaine des STI?

Dans le cadre de l'intelligence artificielle, les STI constituent des systèmes à base de connaissances (SBC) particuliers dédiés à l'enseignement par l'ordinateur. Puisque, pour la conception des SBC, l'acquisition est un des problèmes les plus ardus, cet aspect doit également être pris en compte dans les recherches sur la construction de STI.

Pour ce faire, précisons que le problème en acquisition [Krivine & David, 91].ne se résume pas à obtenir un maximum des données concernant un certain domaine. Même ci cela est réalisable, il reste encore deux tâches difficiles à résoudre:

1. Détecter les connaissances pertinentes (parmi les connaissances recueillies auprès des experts) par rapport au processus étudié,
2. Organiser ces connaissances pour les rendre utilisables par le système. Pour cela des structures des connaissances doivent être définies.

Lorsque le but d'un STI réside dans la mise en œuvre de cours complets, aborder le problème de l'acquisition des connaissances devient indispensable. En effet, le problème de plusieurs STI réside dans le fait qu'ils opèrent sur un domaine d'application très restreint, ceci limitant leur applicabilité et rendant difficile toute généralisation importante. Face à ce problème, les travaux en acquisition des connaissances peuvent être utiles pour la

réalisation d'un STI performant.

Concentrer les efforts de recherches sur l'acquisition de l'expertise n'est pas une idée nouvelle dans le domaine des STI. Plus précisément, une des différences essentielles entre les STI et les systèmes d'EAO est que les premiers visent l'intégration des connaissances expertes sur le domaine à enseigner aussi bien que celle de connaissances sur l'enseignement et l'apprentissage.

Jusqu'à présent, les travaux relatifs à l'expertise des STI se sont surtout focalisés sur la représentation des connaissances, négligeant quelque peu les aspects acquisition. Les STI utilisent souvent des formalismes de représentation comme les règles de production, les frames, ou encore les réseaux, pour ne citer que quelques exemples. Le tableau 2.1 fournit quelques exemples de formalismes utilisés dans des STI célèbres.

STI	Domaine	Formalisme de représentation
SCHOLAR	Géographie	réseaux
GUIDON	Maladies infectieuses	règles
SOPHIE	Détection de pannes	réseaux
WEST	Expressions arithmétiques	règles

tableau 2.1: Formalismes de représentation utilisés par quelques STI [Nkambou, 96].

Ces formalismes de représentation sont d'un trop bas niveau: ils sont trop proche de l'implémentation et ne résolvent pas le problème de l'interaction directe entre les différents sujets (apprenant, pédagogue, expert) et le système car généralement ceux-ci ne sont pas familiers des techniques utilisées par les concepteurs du système.

Pour conclure, soulignons deux points primordiaux:

- pour les domaines complexes il est indispensable d'organiser les connaissances d'un STI de manière explicite. Dans ce sens un curriculum est nécessaire pour structurer, planifier et contrôler l'enseignement et l'apprentissage;
- pour faciliter la mise en œuvre du premier point nous soulignons l'importance

d'intégrer des outils d'aide à l'acquisition des connaissances pour un STI. En effet, la majorité des STI utilise des techniques d'extraction des connaissances classiques: interviews, analyse des protocoles, questionnaires, etc. Contrairement à ces techniques d'explicitation classiques qui demandent aux experts de préciser tous les détails de leur savoir dès le début de la conception d'un système, nous considérons nécessaire de fournir des moyens de communication expert-système tout au long du processus.

2.3 Un approche de production de STI: le projet SAFARI

Les travaux qui font l'objet de ce document ont été effectués dans le cadre du projet SAFARI (Système d'Aide à la Formation par l'Analyse de Raisonnements Interactifs) [Gecsei & Frasson, 94]. SAFARI est un projet intégré dans le programme Synergie, soutenu par le ministère de l'Industrie, du Commerce, de la Science et de la Technologie du Québec qui s'appuie sur la coopération entre des partenaires universitaires (cinq universités sont impliquées dans ce projet) et des partenaires industriels (deux entreprises privées).

L'objectif principal du projet est la production de cours complets capables de supporter le processus de formation du personnel dans des entreprises industrielles. SAFARI est un atelier de développement de systèmes tutoriels intelligents s'appuyant sur l'utilisation de modèles pour guider la formation. Ainsi, plusieurs modèles ont été intégrés, notamment les modèles de représentation du curriculum (ces modèles sont présentés dans le chapitre 6, section 6.1.1) et le modèle de l'apprenant.

En guise de conclusion, nous pensons que, si la construction d'environnement permettant la coopération humain-machine est nécessaire pour la formation des apprenants, il n'en demeure pas moins qu'on doit assurer aux autres sujets impliqués (principalement l'expert du domaine et le pédagogue) des possibilités de communication avec le système pour favoriser l'acquisition des connaissances pour un STI.

Chapitre 3

Acquisition des connaissances

3.1 Qu'est ce qu'une expertise ?

Voilà une question qui, depuis environ deux décennies, a ouvert un nouveau champ de recherche dans les sciences cognitives, principalement en psychologie cognitive et en intelligence artificielle.

L'intelligence artificielle, qui se propose de rendre la machine intelligente, vise donc la réalisation de logiciels capables de reproduire certains aspects de l'activité intelligente humaine dans un domaine donné (médecine, gestion de production, etc.).

L'aptitude d'un tel système à montrer de l'intelligence est attribuée à la richesse des connaissances spécifiques qu'il peut contenir et manipuler [Lenat & Feigenbaum, 87]; le processus visant à obtenir les connaissances des experts et les mettre sous une forme utilisable par le système reste le problème fondamental dans le développement d'un Système à Base de Connaissances (SBC).

De façon traditionnelle, l'acquisition des connaissances est faite par un spécialiste en informatique - l'ingénieur des connaissances ou cogniticien - qui interroge l'(es) expert(s) du domaine et lit la documentation spécifique. L'ingénieur des connaissances implémente un prototype du système de connaissances pour quelques cas.

L'expert critique le prototype et l'ingénieur des connaissances modifie le système. Le prototype est testé et révisé jusqu'à ce qu'il atteigne un niveau de performance acceptable pour être livré aux usagers. La maintenance du système est assurée continuellement par l'équipe expert et ingénieur des connaissances. Cette méthodologie est lente, coûteuse et susceptible d'engendrer des erreurs [Gruber, 89].

En effet, "*il est généralement admis que la difficulté des experts à expliciter leur connaissances est une difficulté de communication, que la représentation adéquate est quelque part dans l'esprit de l'expert, mais inaccessible à l'introspection ou à la verbalisation. (...). Les experts n'ont pas besoin d'avoir des représentations formalisées pour agir*" [Winograd & Flores, 89].

Solliciter et formaliser la connaissance experte nécessite de faire référence à la pratique et à l'habilité de l'expert car ce dernier a acquis son savoir par l'expérience. L'expert n'applique pas forcément une démarche scientifique, il a une façon propre d'agir et une conception particulière de son travail. En effet, comme le souligne Le Roux [LeRoux, 94], "*son savoir articule des connaissances d'origines multiples dans un raisonnement spécifique*" qui lui permet d'agir sur le monde réel.

Nous voulons souligner que la diversité et la complexité de l'expertise, son caractère implicite et "sousconscient" transforme le processus d'acquisition en l'étape la plus difficile de la construction d'un SBC. La section suivante vise à montrer que l'idée "d'extraire" les connaissances pour ensuite les coder dans un programme, idée qui a marqué l'approche "transfert", n'est pas satisfaisante.

3.2 L'évolution des recherches : "transfert" des connaissances vers modélisation des connaissances

Dès l'origine des systèmes à base des connaissances, la complexité des savoirs experts ainsi que les difficultés rencontrées dans l'explicitation de ces savoirs ont été soulignées par de nombreux auteurs. Pour mieux comprendre pourquoi l'acquisition des connaissances a été qualifiée de "goulot d'étranglement" lors de la conception d'un SBC, étudions l'évolution de cette discipline.

Dans le domaine de l'intelligence artificielle, les années 70 ont soulevé beaucoup d'enthousiasme grâce, principalement, à l'apparition des systèmes experts. A partir de là, l'acquisition des connaissances - étape qui englobe le recueil, la structuration et la formalisation des connaissances - est vite apparue comme une des difficultés majeures lors de la conception d'un SBC.

Durant la période du début des années 70 au milieu des années 80, l'époque de l'approche "transfert", l'acquisition des connaissances était comprise comme une activité de *transcription* des connaissances d'un expert dans un formalisme de représentation (par exemple: la logique, les règles de production, les frames, les objets). Plusieurs auteurs parlent *d'acquisition des connaissances guidée par l'implementation* [Hickman et al., 89], car le but était d'extraire des connaissances et de les traduire dans des structures classiques de représentation. Par ailleurs, diverses expériences ont montré que ces *langages d'assemblage de connaissances* [Chandrasekaran, 87] sont insuffisants pour les problèmes complexes que l'intelligence artificielle se propose de résoudre, tels par exemple: la planification, le diagnostic, ... En effet, beaucoup de travaux se sont appuyés sur l'aspect "informatique", au détriment du problème essentiel - trouver le *bon* ensemble de connaissances car un système est d'autant plus performant que son niveau de connaissances est élevé.

Pendant cette période, l'acquisition des connaissances a soulevé deux difficultés majeures.

- La première difficulté est liée au niveau d'abstraction des formalismes de représentation classiques. Ces formalismes sont inadaptés face à la grande diversité et à la complexité de l'expertise, sans oublier les aspects dynamique et évolutif qui caractérisent les connaissances.
- Deuxièmement, même si les techniques d'explicitation [Aïmeur, 94], [Diaper, 89] sont très utiles pour obtenir une description en détail de l'expertise, il reste encore un problème important: l'organisation de l'ensemble de ces informations.

Face à ces problèmes, l'acquisition de connaissances se donne comme objectif la construction de modèles car, à partir des connaissances recueillies auprès de l'expert on

doit organiser la masse d'information pour réduire sa complexité en identifiant certaines caractéristiques compte tenu de buts visés. Voilà pourquoi le problème d'acquisition devient essentiellement un problème de modélisation.

3.3 Le "knowledge level" et les notions en découlant

3.3.1 Le "knowledge level" ou "niveau connaissance"

Les difficultés de l'approche "transfert" sont principalement dues au décalage entre la connaissance et sa représentation. En effet, les formalismes classiques de représentation - tels règles, frames, logique, etc. - possèdent un niveau d'abstraction trop bas pour permettre la représentation des connaissances de manière satisfaisante.

Pour éclaircir les rapports entre connaissance et représentation, Newell [Newell, 82] considère que le problème clé est d'identifier la nature des connaissances qu'un système doit posséder. Dans ce sens, il défini le terme *knowledge level (niveau connaissance)* pour désigner un niveau supérieur de description des systèmes informatiques situé au dessus du niveau symbolique (le programme) comme le montre la figure 3.1.

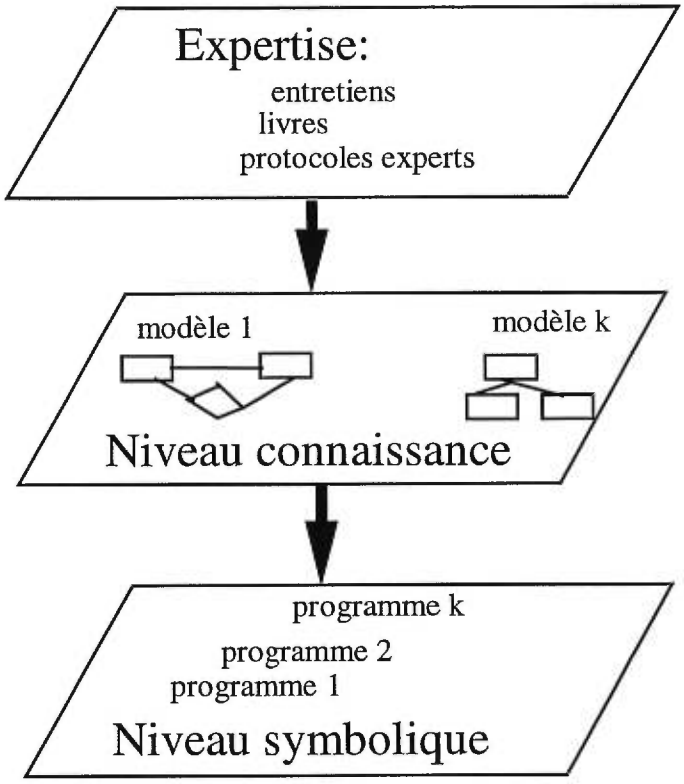


figure 3.1: Trois niveaux à travers lesquels se déroule l'acquisition des connaissances.

Le knowledge level décrit le comportement d'un système et les connaissances utilisées indépendamment du formalisme de représentation. Dans son article [Newell, 82], Newell caractérise le système comme un agent qui dispose de connaissances, qui doit atteindre des buts et qui a des possibilités d'exécuter des actions. De plus, cet agent est rationnel car il fonctionne en respectant le "principe de rationalité":

"Si un agent a la connaissance qu'une de ses actions peut l'amener à l'un de ses buts, alors il va sélectionner cette action" [Newell, 82].

On remarque que cette notion ne donne pas le réponse face au problème d'incertitude quand l'agent doit choisir parmi plusieurs actions potentielles. Les descriptions au knowlwdge level sont donc des théories de fonctionnement d'un système permettant de spécifier son comportement sans expliquer *comment* le système réalise ce comportement [Causse, 94].

Le concept de knowledge level a généré un changement de philosophie d'ensemble dans les recherches relatives à la conception des SBC. Cette notion a été reprise par plusieurs chercheurs qui en ont donné leur propre interprétation. Ainsi, selon Clancey:

"Une description au knowledge level est une spécification d'un système en termes de connaissances qu'il doit posséder pour résoudre la tâche qui lui est assignée. De plus cette spécification doit rendre compte de tout ce que le système peut savoir. Pour répondre à cette exigence, il est nécessaire de caractériser la méthode de raisonnement suivie par le système ainsi que la structure des connaissances qui supporte ce raisonnement" [Clancey, 85].

Nous remarquons déjà dans les mots de Clancey, la différence entre le *knowledge level* idéal tel que défini par Newell qui nous interdit de parler de "comment-faire" et le *knowledge level* adopté par la plupart des chercheurs en acquisition des connaissances qui considèrent que cette notion doit se traduire dans un modèle adapté à la tâche à résoudre en précisant également la méthode de raisonnement, ce sans toutefois faire référence à l'implémentation.

L'adoption d'une perspective au knowledge level est clairement un pas dans la bonne direction car elle concentre l'analyse de l'expertise sur une description abstraite de haut niveau des systèmes, en se focalisant sur les *structures* et la *nature des connaissances* et sur la *nature du raisonnement*. Comme souligne d'ailleurs [Aussenac et al, 93] il s'agit de définir un langage (semi-formel ou formel) de modélisation pour pouvoir exprimer les différents composants de la connaissances. .Plus précisément, à ce niveau nous abordons des problèmes du type: quels sont les buts (le pourquoi), quelles sont les tâches, leurs caractéristiques et quelles connaissances demande la tâche (le quoi), quel type de modèle propose l'expert du domaine [Steel, 90], [Aussenac et al., 96].

3.3.2. Les notions qui découlent du terme knowledge level: modèle, tâche, méthode

En acquisition des connaissances, l'hypothèse du knowledge level a donné naissance à un nouvel objectif: appuyer la conception d'un système à base de

connaissances sur une spécification générale permettant de décrire l'activité de l'expert en termes de structures abstraites.

Dans la pratique, cet objectif se traduit essentiellement par une activité de construction de modèles. Selon Karbach [Karbach et al., 90], un *modèle* est *"une abstraction voulue et orientée, qui nous permet de réduire la complexité réelle d'un problème en se concentrant sur certains de ses aspects."*

Ainsi, le rôle d'un modèle est de guider l'acquisition vers ce qui est essentiel. Ces connaissances de principe ont été concentrées en trois concepts: le *modèle du domaine*, la *tâche* et la *méthode de résolution.*

Le *modèle du domaine* sert à préciser les objets du monde réel et leurs caractéristiques, ainsi que leurs relations. Par exemple, pour une application médicale, le domaine peut être conçu sous forme d'un modèle "entités-relations", comme dans la figure 3.2, impliquant quatre objets du domaine: les examens, les bilans, les maladies, les symptômes, etc. Dans ce cas une relation peut être celle qui associe un ensemble de symptômes à une maladie.

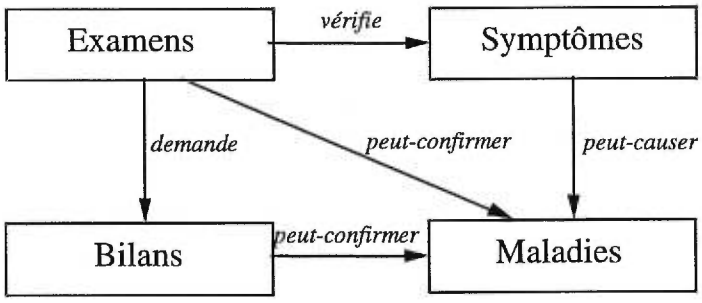


figure 3.2: Modèle du domaine pour une application médicale.

Les notions de *tâche* et de *méthode* sont essentielles pour définir le processus de résolution. Plus précisément, les tâches doivent spécifier le problème et les méthodes permettant de le résoudre.

Dans la littérature, le terme de "tâche" a été souvent employé par les chercheurs de manières différentes. Pour nous, une tâche représente une opération ou une séquence d'opérations visant à l'atteinte d'un but. Les tâches modélisent des problèmes et des stratégies (méthodes) de résolution. Par exemple, si nous considérons le calcul de la

superficie d'un triangle, obtenir le résultat constitue le but à atteindre alors que la formule géométrique constitue la méthode de résolution.

En résumé, en acquisition des connaissances, la perspective de modélisation comporte deux aspects: la modélisation du domaine [Wielinga & Breuker, 86], [Steels, 90] et la modélisation du processus de résolution [Wielinga & Breuker, 86], [McDermott, 88]. Ces deux aspects sont en effet totalement interdépendants: il n'est pas souhaitable de modéliser un domaine sans préciser le contexte (le problème) dans lequel ce domaine sera utilisé [Krivine & David, 91].

3.4 L'explicitation des connaissances

L'explicitation des connaissances consiste à recueillir les connaissances auprès des experts au moyen de techniques manuelles, semi-automatiques ou automatiques [Aïmeur, 94]. Ce processus a été l'objectif central de la recherche en acquisition des connaissances. Actuellement, puisque l'approche modélisation s'impose de plus en plus, de nombreux auteurs ont souligné le lien étroit existant entre modélisation et explicitation.

Sous l'optique de modélisation, nous considérons l'explicitation des connaissances comme un processus guidé par le(s) modèle(s) construit(s) auparavant, lors de la phase de modélisation. De plus, le processus d'acquisition n'est pas linéaire, du type modélisation puis explicitation. Le développement d'un système est un processus cyclique car la modélisation fait souvent appel à l'explicitation et vice-versa [Aïmeur, 94] [Aïmeur, 96].

Même si les travaux actuels font moins souvent allusion aux techniques d'explicitation, celles-ci sont essentielles pour l'acquisition des connaissances, notamment pour assurer le passage entre une théorie de l'expertise et la réalisation d'un système opérationnel.

Galloüin [Galloüin, 88] classifie les techniques d'explicitation en deux catégories:

- les interviews et l'observation directe;
- Les techniques issues de la psychologie cognitive: l'Analyse de protocole [Zacklad, 93a], la "Théorie des construction Personnelles" (Personal Construct

Theory) [Gaines, 93], l'Analyse des grilles-répertoires [Boose & Bradshow, 93], la Classification [Aïmeur, 94], [Ermine, 93], etc.

La première approche (intégrant les interviews et l'observation directe) et présente les inconvénients suivants:

1. l'information recueillie manque souvent d'objectivité,
2. on n'a pas de garantie de complétude de l'information,
3. le "coût" des séances d'interviews est trop élevé.

La seconde approche (psychologie cognitive) étudie les mécanismes du raisonnement humain. Les techniques issues de la psychologie cognitive garantissent l'objectivité des informations et, de plus, permettent l'accès rapide à certains traits de la connaissance, par exemple: les relations entre objets. Ces techniques permettent donc de réduire le temps de transfert d'expertise. Les limitations de cette approche sont:

1. qu'il existe peu de spécialistes capables de faire des analyses rigoureuses en raison de la durée importante de la formation aux techniques d'analyse de protocole,
2. que les psychologues ont eux-mêmes de difficultés à utiliser ces techniques dans un contexte réel.

Si on regarde plutôt la façon dont l'expert énonce son savoir, on peut diviser les techniques d'explicitation en méthodes contraintes et méthodes non-contraintes [Aïmeur, 94].

- Les *méthodes non-contraintes* ("non contrived methods") [Shadbolt & Burton, 90] appelées aussi *méthodes directes* [Aussenac, 89], [Firlej & Hellens, 91], ou *méthodes standards* [Major & Reichgelt, 90], permettent le transfert direct de la connaissance de l'expert, sans passer par la contrainte d'une structure préalable définie par le cogniticien (citons les interviews, l'observation directe, l'analyse de protocole, l'utilisation de questionnaires, les verbalisations rétrospectives, l'introspection, le "brainwriting", etc.).
- Les *méthodes contraintes* ou *méthodes indirectes* [Aussenac, 89], [Firlej & Hellens, 91], [Major & Reichgelt, 90], [Shadbolt & Burton, 90] contraignent l'expert à utiliser des structures établies au préalable (par exemple:

tableaux, grilles d'évaluation, arbres, réseaux, etc.). Dans cette catégorie on distingue les grilles-répertoires, le tri de cartes, le regroupement hiérarchique etc.

Il est à noter que dans le cas de méthodes non contraintes, le cogniticien dépend totalement de l'expert, qui est la seule source d'information, alors que dans les méthodes contraintes le cogniticien interagit avec l'expert. L'expert n'a, en général, aucune habitude des processus de transfert de connaissances, souvent il n'est pas capable de formuler et de structurer son savoir et dans ce sens les méthodes contraintes peuvent être très utiles [Aïmeur, 94].

Nous avons montré dans ce chapitre comment le processus d'acquisition perçu lors de la période des systèmes experts de première génération comme un transfert des connaissances vers l'ordinateur s'est imposé comme une activité de modélisation. Nous avons alors traité du concept de "Knowledge Level" qui a généré un changement radical dans les recherches en l'acquisition des connaissances. Celui-ci, ainsi que les notions qui en découlent (modèles, tâches, méthodes), forment le noyau de base qui nous permettrons d'aborder l'acquisition des connaissances sous l'angle de la modélisation. Des méthodologies importantes pour tout travail en acquisition des connaissances ont été proposées. Ces travaux fondateurs font l'objet du chapitre suivant.

Chapitre 4

Modélisation du processus de résolution de problèmes

Le chapitre précédent a abordé l'approche modélisation, qui constitue aujourd'hui le courant de recherche central en acquisition des connaissances, en introduisant les concepts clés sur lesquels s'appuie cette approche: les modèles, les tâches et les méthodes de résolution.

Il convient maintenant de préciser *comment modéliser,* c'est-à-dire, quels sont les travaux fondateurs qui ont fait progresser l'acquisition des connaissances abordée dans cette optique de modélisation.

Tous les travaux appartenant au courant de modélisation constructiviste ont pour caractéristique commune l'intention de dégager des éléments génériques dans les modèles construits. Ces éléments facilitent le processus d'acquisition car ils constituent des parties réutilisables par d'autres applications. Nous utilisons ici la classification réalisée par Causse qui regroupe les travaux majeurs de l'approche de *l'acquisition des connaissances dirigée par les modèles* en trois "écoles" [Causse et al., 92]:

L'école "choisir pour suivre"	L'approche "tâches génériques" Les méthodes "à limitation de rôles" L'approche "de la méthode à la tâche" La méthodologie KADS
L'école "blocs de construction"	Les "Composants de l'Expertise" de Steels PROTEGE II
L'approche modélisation du domaine	KOD : Modélisation de l'expertise La modélisation ontologique

tableau 4.1: Les trois "écoles" de l'approche modélisation [Causse et al., 92].

Dans ce chapitre consacré à l'approche modélisation du raisonnement, nous présenterons les travaux majeurs issus de ces écoles fondatrices, en privilégiant toutefois les écoles "choisir pour suivre" et "blocs de construction" qui nous semblent en effet plus adaptées à notre problématique.

4.1 L'école "choisir pour suivre"

L'école "choisir pour suivre" est à l'origine de la recherche de méthodes et de modèles réutilisables pour plusieurs applications.

L'idée est qu'en disposant d'"éléments" génériques on peut bâtir des nouveaux modèles plus rapidement tout en obtenant une meilleure qualité. Cette hypothèse constitue la base de départ de tout travail issu de cette approche. Ceci fait que, que même avec des mises en œuvre différentes, le "principe" reste le même: "choisir" la méthode adéquate et la "suivre" pour obtenir le modèle final (ce modèle est généralement obtenu par une instanciation d'un modèle générique de la méthode avec les connaissances spécifiques à un domaine d'application) [Causse et al., 92].

Nous présentons dans la suite les quatre approches issues de cette école: l'approche "tâches génériques", les méthodes "à limitation des rôles", la méthodologie KADS et l'approche "de la méthode à la tâche".

4.1.1 L'approche "tâches génériques"

Cette section est consacrée à la présentation de la théorie des "tâches génériques" ("generic tasks") développée par Chandrasekaran et son équipe [Chandrasekaran, 86], [Bylander&Chandrasekaran, 87].

Pour comprendre de quelle façon les convictions méthodologiques étayées par cette théorie apportent une réponse à la question de la modélisation, précisons d'abord deux concepts fondamentaux: le *problème d'interaction* et la notion de *tâche générique*.

Le *problème d'interaction* se résume dans la phrase suivante [Bylander & Chandrasekaran, 87]:
"la représentation de connaissances pour résoudre un certain problème est fortement influencée par la nature du problème et par la méthode (la "stratégie d'inférence") à appliquer aux connaissances". Ce principe implique *"qu'on aura besoin de méthodologies d'acquisition différentes pour des types de raisonnement différents, donc une méthodologie différente pour chaque tâche générique."*

Dans le même article [Bylander & Chandrasekaran, 87], les auteurs donnent deux raisons principales qui sous-tendent le problème d'interaction:

1. *Le choix des connaissances.* Le processus d'acquisition doit se focaliser sur les connaissances de "haute utilité" (les connaissances qui sont indispensables pour les problèmes à résoudre) et chercher à réduire la complexité en évitant ou en éliminant celles de "faible utilité".
2. *Les contraintes de la méthode de résolution.* Une *méthode* utilise et interprète des connaissances du domaine, celles-ci doivent être représentées en tenant compte de la manière dont elles seront traitées. La connaissance doit être adaptée à la méthode pour s'assurer que les inférences nécessaires puissent être faites.

Ainsi, il y a un lien étroit entre le *type de problème*, les *types de connaissances* et la *méthode* permettant de résoudre le problème. De plus, les recherches ont montré qu'il existe certains types de raisonnements applicables à des problèmes différents issus de domaines variés. En conséquence, pour décrire un ensemble de problèmes similaires, il est important d'identifier des *tâches génériques*, caractérisées par les éléments suivants:

1. le *type de problème* (types des entrées et des sorties). Quelle est la fonction de la tâche générique et à quoi sert-elle?
2. la *représentation des connaissances.* Comment structurer les connaissances pour accomplir la tâche? Comment sont-elles organisées en termes de concepts ?

3. la *méthode*. Quelle méthode peut on appliquer aux connaissances pour accomplir la tâche générique et comment cette méthode opère-t-elle sur les concepts ?

Ainsi, une *tâche générique* désigne un modèle élémentaire qui se focalise sur le type de problème à résoudre, sur la structure des connaissances du domaine (les concepts) et sur la méthode de résolution (selon la définition énoncée dans [Chandrasekaran & Johnson, 93], il s'agit d'une organisation d'inférence élémentaires). Ce modèle est représenté sur la figure 4.1 issue de [Chandrasekaran, 86].

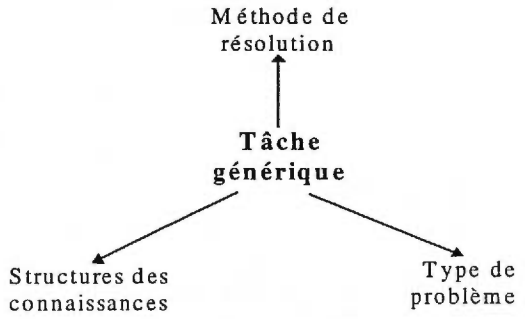


figure 4.1: Modélisation des tâches génériques [Chandrasekaran, 86].

Pour clarifier toutes ces idées, prenons pour exemple la première tâche générique définie par l'équipe de Chandrasekaran, la *classification hiérarchique*. Cet exemple est repris de [Chandrasekaran, 86].

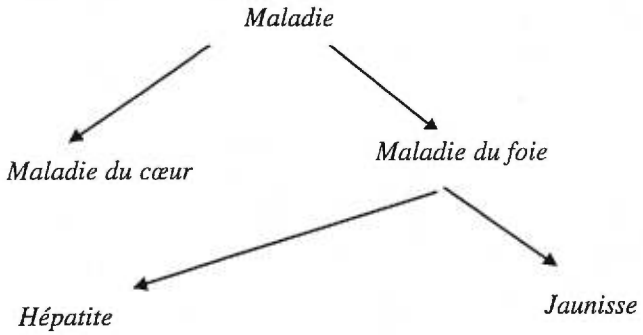


figure 4.2: La classification hiérarchique (exemple adapté de [Chandrasekaran, 86]).

La *classification hiérarchique* tell qu'illustrée dans la figure 4.2 est caractérisée par:

- le *problème*: étant donné la description d'une situation, déterminer quelles sont les catégories ou les hypothèses qui s'appliquent. Dans ce cas il s'agit de diagnostiquer la maladie,
- les *concepts importants*: les hypothèses, pour cet exemple les maladies potentielles.
- la *représentation des connaissances*: les hypothèses sont organisées en hiérarchie où les fils d'un nœud représentent des sous-hypothèses (dans l'exemple ci-dessus l'hépatite est

une sous-hypothèse de maladie du foix). Des connaissances pour calculer le degré de certitude sont nécessaires.

- la *méthode* à utiliser compte tenu du problème, par exemple ici: la méthode "*établir puis raffiner*" [Bylander, 86] qui utilise des connaissances additionnelles permettant d'accepter ou de rejeter une hypothèse. Si une hypothèse est confirmée ou semble l'être (partie à établir), alors ses sous-hypothèses doivent être considérées; sinon, l'hypothèse et ses sous-hypothèses doivent être rejetées (partie à raffiner).
- les *exemples*: cette tâche générique est applicable à tout problème de type diagnostic puisque le principe de résolution consiste à déterminer les hypothèses qui rendent le mieux compte d'une certaine situation, parmi un ensemble d'hypothèses potentielles.

La puissance de l'approche *tâches génériques* provient du fait que si un problème correspond à une tâche générique, alors la tâche générique offre une structure de connaissances et une stratégie d'inférence pour le résoudre. De plus, des "coquilles" ("shells") ont été implémentés pour chaque tâche générique. Ainsi, pour la résolution d'un problème complexe il faut identifier les tâches principales et sélectionner les "shells" adéquats [Krivine & David, 91].

En revanche, les méthodes proposées par cette approche ne peuvent pas être modifiées, donc un problème qui ne peut être résolu par la même décomposition et les mêmes mécanismes de base ne pourra être traité.

4.1.2 Les méthodes à limitation de rôles

Contrairement à l'approche *tâches génériques*, qui aborde la "généricité" à partir du type de problème à résoudre, *les méthodes à limitation de rôles* (Role Limiting Methods, RLM) [Mcdermott, 88] [Marcus & McDermott, 93] [Eshelman et al., 93] traitent cette problématique en se focalisant tout d'abord sur l'identification des connaissances selon leurs "rôles" dans une méthode de résolution.

L'objectif principal de cette approche est de faciliter le processus de résolution de problèmes en utilisant un ensemble de méthodes prédéfinies (moins générales que les méthodes développées pour les tâches génériques), chacune dédiée à un ensemble de tâches. Comme le travail de programmation est déjà fait, l'expert du domaine peut dialoguer directement avec l'outil d'acquisition et, de cette façon, il peut construire lui-même son application.

Selon McDermott, la résolution de problèmes comporte trois étapes: "*l'identification, la sélection, et l'implantation d'une séquence d'actions permettant de réaliser une tâche donnée dans un domaine spécifié*" [Mcdermott, 88]. Chaque étape peut être associée à une méthode de résolution décrivant la procédure à suivre. Pour une application particulière, ces méthodes opèrent sur des connaissances du domaine qui jouent les "rôles" demandés par la méthode.

McDermott a défini la notion de "rôle" comme étant un ensemble de connaissances spécifiques à une méthode de résolution de problèmes. Ainsi, une méthode à limitation de rôles est efficace pour les problèmes pour lesquels on dispose des connaissances adéquates (qui correspondent à leurs rôles).

L'explicitation des connaissances est réalisée afin que seules les connaissances impliquées dans la résolution soient identifiées et structurées de manière à pouvoir être appliquées au problème. La figure 4.3 représente graphiquement ce processus. Ainsi, dans une première phase, le cogniticien choisit l'outil approprié à la tâche. Ceci implique de déterminer la méthode adéquate et les types de connaissances jouant des rôles précis dans la méthode. Une fois la méthode de résolution "déclenchée" elle active le module d'acquisition qui réalisera la deuxième phase: collecter les connaissances du domaine via un dialogue avec l'expert.

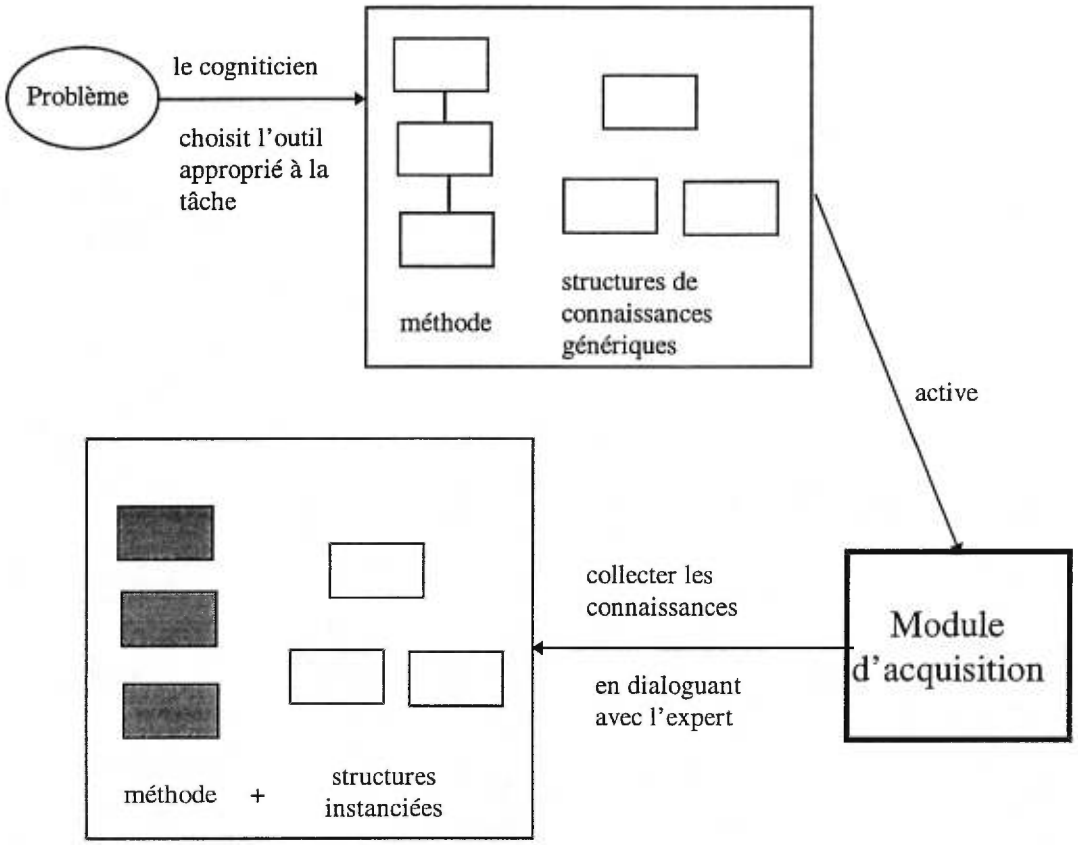


figure 4.3: Schéma général d'une "méthode à limitation de rôles".

Il est à noter que la notion de rôle est fondamentale et qu'on la trouve dans toutes les approches d'acquisition dirigée par les modèles. Par exemple, en KADS, nous retrouvons le concept de *rôle* sous l'appellation de *couche* ou *niveau.*

La réalisation pratique de cette approche a donné naissance à plusieurs méthodes à limitation de rôles.

- *Cover and Differentiate,* supportée par les outils MORE [Kahn, 88] et MOLE [Eshelman et al., 93].
- *Propose and Revise,* supportée par l'outil SALT [Marcus & McDermott, 93].
- *Extrapolate from a similar case,* supportée par l'outil SIZZLE [Offut, 88].
- *Acquire and Present,* supportée par l'outil KNACK [Klinker, 88].

Nous avons choisi de présenter ici la méthode "*couvrir puis différencier*" (*Cover and Differentiate)* qui peut être appliquée aux tâches de type diagnostic. Cette méthode s'appuie sur un raisonnement de type causal qui permet d'exprimer qu'un événement (une situation

réelle, par exemple une maladie) est la *"cause"* d'un autre événement (c'est-à-dire, elle permet d'expliquer, par exemple: une maladie cause des symptômes). Cette méthode consiste à développer un ensemble de solutions possibles et à choisir celle qui est la plus prometteuse. Ainsi, dans un premier temps, on applique la connaissance de "couverture" (*Cover*) pour expliquer un ensemble d'événements (signes, symptômes) par des hypothèses (maladies). Ensuite, on utilise la connaissance de "différenciation" (*Differentiate*) pour discriminer les hypothèses.

La connaissance de couverture est supportée par deux principes [Eshelman et al., 93] :

1. le principe d'*exhaustivité* - pour tout événement doit exister au moins une explication, donc au moins une hypothèse expliquant l'événement (il faut trouver, par exemple, l'ensemble de maladies qui peuvent expliquer une ou plusieurs symptômes);
2. le principe d'*exclusivité* - si pour expliquer un problème une seule hypothèse suffit, alors il ne faut retenir que celle-ci et éliminer les autres (s'il existe plusieurs hypothèses concurrentes il faut trouver la meilleure que toutes les autres) .

Les événements sont reliés à des hypothèses par un réseau causal d'explications. On dit qu'une explication couvre un événement quand il existe un chemin (dans le réseau causal) de l'hypothèse (qui est attachée à l'explication) à l'événement en cause. Dans la première étape, la méthode doit trouver toutes les explications possibles pour un ensemble d'événements en parcourant le graphe d'explications. Dans la deuxième étape, les explications seront différenciées pour retenir les "meilleures" et éliminer les "moins prometteuses" pour l'événement considéré. La "différenciation" peut être réalisée comme dans MOLE à partir des connaissances explicitées auprès de l'expert.

Par exemple, prenons le cas d'un patient souffrant d'une fièvre intermittente. Une explication possible est le fait que le patient souffre de paludisme (cf. le réseau causal ci-dessous)

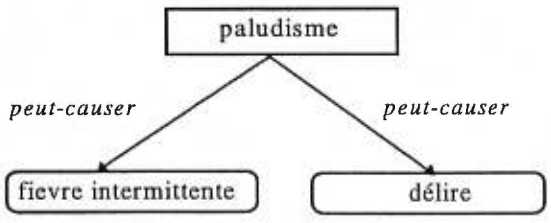


figure 4.4: Réseau causal partiel maladies-symptômes.

Par contre, les connaissances: "la personne n'a pas quitté le Québec depuis dix ans" et "le paludisme est transmis uniquement par la piqûre de l'anophèle" et "l'anophèle ne vit pas au Canada" éliminent cette hypothèse.

Les *méthodes à limitation de rôles* et les *tâches génériques* ont en commun la tendance d'offrir des modèles génériques qui réduisent l'acquisition des connaissances à l'instanciation de modèles avec des connaissances spécifiques à l'application.

En tant que théorie, la méthode RLM introduit pour la première fois la notion de rôle, en soulignant l'utilité de définir les types de connaissances impliquées dans la résolution d'un problème.

En pratique, il existe deux inconvénients:

1. peu d'informations permettent de déterminer l'adéquation d'une méthode à un problème réel [Causse, 94],
2. pour certaines applications, les structures de connaissances peuvent s'avérer trop simples pour pouvoir supporter toutes les connaissances nécessaires à l'étape de couverture. Dans ce sens, l'échec de MOLE pour construire un système de diagnostic relatif à une raffinerie de pétrole est significatif [Eshelman et al., 93].

4.1.3. L'approche "de la méthode à la tâche"

De manière générale, le cogniticien et l'expert travaillent ensemble sur les étapes de l'identification et de conceptualisation de l'acquisition des connaissances. Toutefois, ce contexte humain est délicat, l'expert et le cogniticien ont des compétences différentes (l'expert détient l'expertise du domaine et le cogniticien est censé organiser cette connaissance) et ceci pose un problème de communication.

L'idée originale de l'approche *"de la méthode à la tâche"* (*"method-to-task"*) [Musen, 89a] [Musen, 89b] est d'offrir pour chaque membre de l'équipe intervenant dans la conception d'un SBC, expert ou cogniticien, un outil adapté au rôle qu'il joue dans le processus d'acquisition. Cette division de la tâche globale d'une application en deux phases s'appuie sur l'hypothèse que de cette manière, chacun travaille mieux.

À notre connaissance, la seule réalisation supportant cette approche est l'architecture PROTEGE-OPAL de Musen [Musen, 89a] qui intègre deux outils: PROTEGE dédié au cogniticien et OPAL dédié à l'expert.

Cette approche comporte deux étapes :

1. le cogniticien construit le modèle de la tâche en utilisant le langage visuel offert par l'outil PROTEGE. Cet outil donne aussi les spécifications pour générer l'éditeur spécialisé OPAL, qui permet le dialogue avec l'expert dans la terminologie spécifique du domaine;
2. l'expert fournit la connaissance du domaine en utilisant OPAL qui vérifie les connaissances saisies par rapport à son modèle de la tâche.

L'outil PROTEGE utilise un modèle abstrait de la méthode "affinement de squelettes de plans" (*skeletal plan refinement*). Le cogniticien instancie ce modèle en définissant les termes et les structures des connaissances du domaine.

Par exemple, dans le domaine des tests cliniques, PROTEGE sert pour créer des modèles des classes des plans de traitement. Il génère ensuite des outils graphiques dédiés aux médecins pour définir en détail des tests cliniques particuliers. Les spécifications données par les médecins sont traduites automatiquement dans la base de données d'un système expert [Musen, 89a].

PROTEGE a été également utilisé pour la construction de p-OPAL, un outil d'acquisition pour la thérapie du cancer. HTN est un autre programme créé avec PROTEGE qui permet aux médecins d'introduire des plans de traitement pour les patients souffrant d'hypertension. La puissance de cet outil est telle que ces deux programmes ont été générés avec PROTEGE après seulement quelques jours de travail [Musen, 89b].

Le grand mérite de cette approche est sa capacité à générer automatiquement un outil permettant à l'expert de travailler dans son propre langage. En revanche, cette approche n'est applicable qu'à un nombre limité d'applications car on dispose d'une seule méthode de résolution. Le système PROTEGE-II [Puerta et al., 91] est conçu pour dépasser cette limitation. Nous en présenterons les principales caractéristiques dans la section 4.2.2.

4.1.4. La méthodologie KADS

Les travaux de KADS (Knowledge Acquisition Design Support) ont été élaborés par deux grands projets européens: le premier projet Esprit KADS (ou KADS-1) couvre la période 1985-1990 (travaux synthétisés par [Schreiber, 93]). Le deuxième, connu sous le nom de CommonKADS (1991-1994) complète les résultats de KADS-1.

Cette méthodologie est bâtie sur un ensemble de principes et sur le développement d'une succession de modèles. Les éléments-clé de la méthode sont: le "principe" de

modélisation, le modèle conceptuel de KADS et la bibliothèque de modèles d'interprétation génériques.

Selon KADS, la manière de traiter la complexité de l'expertise est d'utiliser plusieurs modèles relatifs à des aspects particuliers du système à construire. Cette idée centrale repose sur le principe de modélisation:

"Le développement d'un système à base de connaissances est considéré comme la construction d'une série de modèles vus dans leur contexte concret d'application et d'organisation. Un système à base de connaissances est une réalisation informatique concrète de ces modèles.

Le modèle de l'expertise est central dans la méthodologie CommonKADS, il modélise le comportement de résolution de problèmes en termes de connaissances qui sont appliquées pour résoudre certaines tâches." [Wielinga et al., 92].

Ainsi, pour simplifier la tâche de modélisation, KADS aborde ce problème de manière descendante, en la considérant comme la création d'une succession de modèles. Il existe sept types de modèles à distinguer [Wielinga et al., 92] (voir également la figure 4.5 qui présente ces modèles en respectant l'ordre de construction):

- *le modèle organisationnel* offre une analyse de l'environnement dans lequel le système devra fonctionner. Il comprend une description des fonctions et des tâches à réaliser;
- *le modèle de l'application* définit quel problème le système devrait résoudre en spécifiant la fonction du système au sein de l'organisation et les contraintes externes relatives au développement de l'application. Par exemple, pour le problème "détecter et réparer les pannes d'un système" la fonction du système est de "guider l'usager dans le processus de détection/réparation" et les contraintes externes à vérifier sont la vitesse du système, le matériel à utiliser, etc.;
- *le modèle de tâche* spécifie comment la fonction du système (spécifiée dans le modèle de l'application) est réalisée via un certain nombre de tâches que le système devra exécuter. De plus ce modèle précise la distribution des tâches entre différents agents (tels que le système et l'usager) ;
- *le modèle de coopération* spécifie la manière dont les agents (auxquels les différentes tâches ont été assignées) doivent coopérer afin de réaliser la fonction du système;
- *le modèle de l'expertise* permet de décrire les différentes tâches en précisant les connaissances nécessaires (organisées selon leur type) pour les accomplir;
- *le modèle conceptuel (modèle de l'expertise + modèle de coopération)* fournit des descriptions abstraites des objets et opérations. Le modèle conceptuel définit le

comportement du système à construire en mettant en relation ces descriptions avec les différents agents.

- *le modèle de conception* fournit, à partir du modèle conceptuel, une succession de transformations qui va permettre de spécifier l'implémentation.

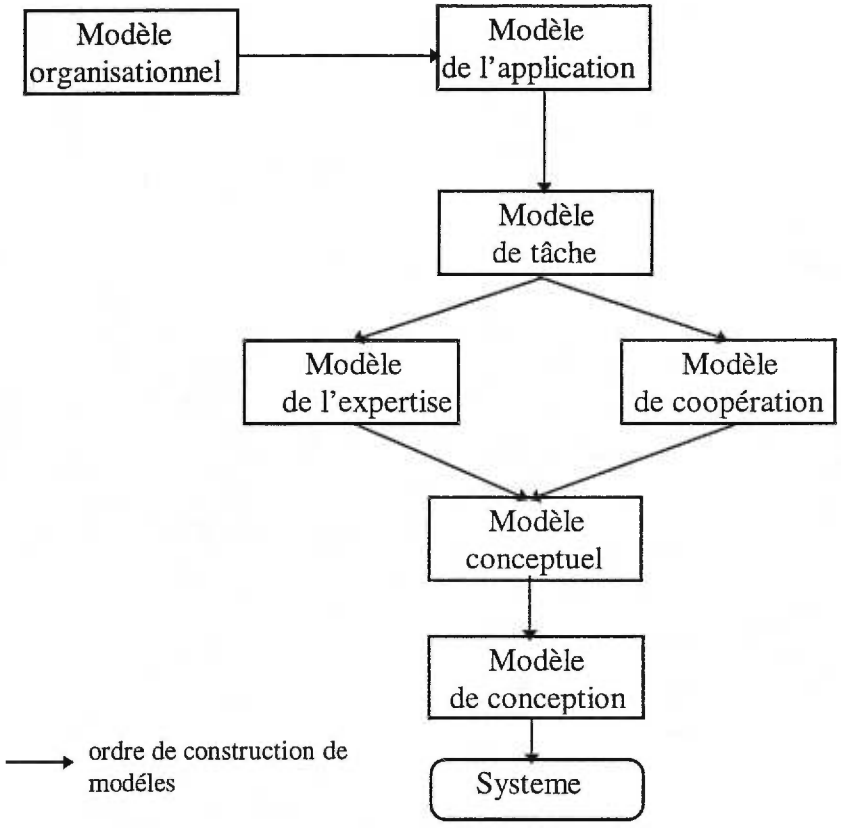


figure 4.5: KADS: suite de modèles [Wielinga et al., 92].

Le plus important, dans cette suite de modèles, est le célèbre *modèle de l'expertise* structuré en quatre couches:

Domaine : concepts, attributs, relations et structures du domaine représentés dans un formalisme de type: objet, attribut, valeur,

Inférence : inférences élémentaires faites sur les entités du domaine. Les modèles d'inférence sont appelés appellent sources de connaissances,

Tâches : comment combiner les inférences pour réaliser une tâche,

Stratégie[1] : comment enchaîner les tâches pertinentes pour résoudre un problème.

La *réutilisation des connaissances* constitue un autre point important dans la philosophie de KADS. Cet objectif est réalisé via la notion de modèle d'interprétation. Un

[1] Dans les travaux actuels concernant la méthodologie KADS cette couche a été éliminée.

modèle d'interprétation est un modèle d'expertise abstrait privé de sa couche domaine. KADS dispose d'une *bibliothèque de modèles d'interprétation génériques*. Ainsi, un modèle d'expertise peut être construit directement, ou par sélection d'un modèle d'interprétation complétée par un raffinement de ce modèle.

Le grand mérite de KADS réside dans sa complétude; cette méthodologie est considérée aujourd'hui comme l'approche d'acquisition la plus générale. En effet, elle couvre toutes les phases de développement d'un système à base de connaissances, de l'analyse jusqu'à la maintenance d'une application. Cependant, les efforts en KADS se sont focalisés plutôt sur l'aspect "conceptuel" que sur l'aspect "opérationnel" du système à développer. Plusieurs travaux, parmi lesquels le projet VITAL [LeRoux, 94] n'est qu'un exemple, ont eu comme but l'opérationalisation du modèle conceptuel de KADS.

4.2 L'école "blocs de construction"

La méthodologie KADS avec sa tendance d'utiliser une bibliothèque de modèles d'interprétation fait le passage vers l'école "blocs de construction". Dans cette nouvelle approche une méthode de résolution est une combinaison de "composants" de résolution prédéfinis [Causse, 92]. Nous présenterons ici deux méthodologies appartenant à cette école: les *Composants de l'expertise* [Steels, 90, 93] et l'approche *PROTEGE II* [Puerta et al., 91].

4.2.1. Les Composants de l'expertise

Nous constatons que les approches décrites dans la section 4.1. vont dans la même direction: appuyer le processus d'acquisition sur une analyse de l'expertise au niveau connaissances. Puisque chaque méthodologie a ses avantages et ses limites, l'idéal est de réaliser une combinaison de ces quatre approches en vue de garder les qualités de chacune: les capacités d'acquisition à base de rôles des RLM, la représentation explicite de "shells" opérationnels à instancier comme dans l'approche "tâches génériques", la flexibilité de KADS pour construire des modèles, la possibilité de manipuler des éditeurs adaptés à l'expert et au cogniticien, comme dans l'approche "de la méthode à la tâche" [Korbach et al., 90].

Les travaux de Steels [Steels, 90], [Steels, 93] s'inscrivent dans la même voie: ils unifient certaines idées présentées antérieurement en réalisant l'approche *Composants de l'expertise* qui impose plus de modularité entre les différents composants de l'expertise. Elle s'appuie notamment sur la considération des contraintes pragmatiques[2] relatives à une tâche.

La théorie de Steels suppose avant tout qu'il existe trois concepts de base pour décrire l'expertise: les *tâches*, les *modèles* et les *méthodes de résolution* [Steels, 90].

- La notion de *tâche*. Étant donné la complexité d'un problème réel, Steels souligne qu'il est important de procéder d'abord à une analyse détaillée de la tâche. La *tâche* de Steels est caractérisée par le type de problème à résoudre (par exemple, une tâche de diagnostic, de planification, etc.) comme dans l'approche "tâches génériques". De plus, l'analyse de la tâche doit prendre en compte les contraintes pragmatiques, c'est-à-dire tous les aspects relatifs à la réalisation d'un système opérationnel viable (un système avec une complexité raisonnable en temps et espace mémoire) [Steels, 90].

- Les *modèles*. Steels décrit sa théorie par la perspective de modélisation. Dans ce sens, l'analyse de la tâche doit se concrétiser dans la construction d'un ou plusieurs *modèles de la situation* (ou *modèles de cas)* pour chaque tâche, ainsi que des *modèles du domaine* nécessaires pour définir la tâche.

- Les *méthodes de résolution*. Selon leur rôle, y a deux types de méthodes de résolution:
  - les *méthodes de décomposition* de tâches en sous-tâches,
  - les *méthodes solution* pour résoudre la tâche.

  Pour une tâche, il existe plusieurs possibilités de décomposition ou de résolution. Prenons l'exemple donné par Steels [Steels, 90] pour une tâche de *classification.* Dans ce cas, il y a différentes méthodes-solutions: la recherche linéaire, l'association, la différentiation, l'affinement descendant, etc. De plus, chaque méthode demande un type précis de connaissances du domaine. Par exemple, l'affinement descendant demande une hiérarchie de classes tandis que la différentiation demande des connaissances sur les caractéristiques permettant de calculer les différences entre plusieurs classes possibles.

La méthodologie d'acquisition de Steels combine une approche descendante de décomposition de la tâche en sous-tâches avec une démarche cyclique qui défini pour chaque

[2] les contraintes pragmatiques sont imposées par l'environnement du travail comme par exemple les limitations de temps et d'espace (les modèles qui demandent de grands espaces mémoire ou trop de temps pour l'exécution ne peuvent être utilisés [Steels, 90]).

tâche (chaque sous-tâche de la tâche globale) le modèle du domaine et la méthode de résolution appropriés aux caractéristiques de la tâche considérée. Ainsi, en suivant le schéma de Steels (selon la flèche de la figure 4.6) il faut déterminer tout d'abord les caractéristiques majeures de la tâche (les aspects conceptuels et les contraintes pragmatiques de la tâche) puis examiner les méthodes ainsi que les modèles du domaine disponibles permettant pour chaque sous-tâche, soit de la décomposer, soit de la résoudre. La sélection de la méthode est déterminée par les contraintes pragmatiques de la tâche. Ensuite, le choix du modèle du domaine dépend de la méthode choisie, car chaque méthode demande un ensemble de types de connaissances qui doivent être instanciées par des connaissances spécifiques au domaine. Par exemple, la méthode de *différentiation* demande qu'on connaisse les caractéristiques de différentiation des différentes classes [Steels, 90].

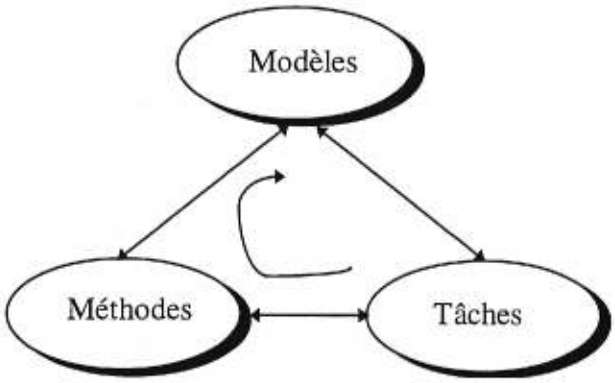


figure 4.6: L'approche Tâches-Méthodes-Modèles [Steels, 93].

Les travaux de Steels offrent une approche modulaire basée sur une combinaison de modèles de cas, modèles de domaine et méthodes de résolution de problèmes. Pour traduire cette approche en pratique, des environnements ("shells") ont été développés comme par exemple KAN [Rademaekers & VanWelkenhuysen, 90]. Ces environnements contiennent les méthodes nécessaires pour un type de problème. Ainsi, l'utilisateur a la possibilité de définir des applications opérationnelles en assemblant des méthodes prédéfinies.

4.2.2.PROTEGE II

Dérivée de son prédécesseur PROTEGE-OPAL (cf. la section 4.1.3), PROTEGE-II est une architecture plus générale conçue pour étendre l'applicabilité de l'approche "de la méthode à la tâche" à des applications plus diverses.

À la différence de PROTEGE, bâti sur une seule méthode - "affinement de squelettes de plans", dans PROTEGE-II il n'y a plus une seule méthode figée. Cette

nouvelle approche permet la modélisation de la méthode de résolution en combinant des procédures élémentaires appelées "mécanismes" capables de résoudre des tâches simples. Nous remarquons ici la ressemblance avec les méthodes-solutions (cf. la section précédente) de l'approche de Steels.

Il y a deux manières permettant de construire une méthode [Puerta et al., 91]:

- de manière descendante, par *configuration*: quand pour une décomposition de la tâche nous pouvons identifier les méthodes et les mécanismes à utiliser,
- de manière ascendante, par *assemblage* de plusieurs mécanismes.

Pour faciliter le travail du cogniticien, PROTEGE-II met à sa disposition une librairie de mécanismes (indexée selon les tâches génériques dont les mécanismes peuvent être utiles à la résolution) ainsi qu'un ensemble d'outils graphiques dédiés à la spécification du modèle de la méthode de résolution. Une fois la méthode définie, son modèle sera instancié avec le vocabulaire spécifique à l'application.

4.3. Une synthèse des systèmes d'aide à la modélisation

L'objectif de cette section est de montrer comment, à partir d'une théorie de l'expertise reposant sur un processus de modélisation, des systèmes d'aide à l'acquisition ont été développés. Leur rôle consiste notamment à faciliter la tâche du cogniticien et celle de l'expert ainsi qu'à améliorer leur communication. Les méthodes de conception de tels systèmes ont beaucoup évolué dans les dernières années en suivant les changements de principes dans le domaine de l'acquisition des connaissances. Nous voulons analyser ici quelques systèmes représentatifs de l'approche de modélisation.

4.3.1 TERESIAS: un "pionnier" des systèmes d'aide à l'acquisition

Un des premiers systèmes d'acquisition "basé sur un modèle" a été TERESIAS [Davis & Lenat, 82]. Il utilise les modèles de règles définis dans l'architecture EMYCIN [van Melle et al., 84] pour structurer le dialogue entre l'expert et l'outil d'acquisition. TERESIAS est capable d'automatiser quelques-unes des tâches effectuées auparavant par des cogniticiens [Gruber, 89]. Son rôle est d'aider l'expert à valider et compléter une base de connaissances en lui signalant les éventuelles incohérences et en lui proposant de nouvelles règles.

Lors d'une session avec TERESIAS, pendant que l'expert saisit les informations décrivant un cas précis, le système évalue ces données par rapport aux règles existantes dans

la base. Dans le cas d'une non-concordance entre la solution proposée par le système et celle proposée par l'expert, TERESIAS aide l'expert à identifier et corriger les erreurs. Néanmoins, un des problèmes de TERESIAS est d'expliquer son comportement par une trace d'exécution du programme. Ceci rend difficile le travail de l'expert qui doit avoir des connaissances pour comprendre les messages du système.

Même si TERESIAS est aujourd'hui quelque peu obsolète, son grand mérite est l'idée de contraindre le dialogue et de le structurer autour de l'interaction entre l'expert et un système imparfait. Au lieu de demander à l'expert "Comment choisissez-vous les tests de diagnostic?", le contexte d'un cas permet à l'assistant de demander: "Quels facteurs ont influencé votre préférence pour X dans ce cas?". Cette caractéristique de TERESIAS a influencé la conception de différents systèmes, comme par exemple l'outil ASK [Gruber, 89].

4.3.2. MOLE: un outil basé sur la méthodologie "à limitation de rôles"

Les outils les plus connus qui suivent la théorie "à limitation de rôles" sont MORE [Khan, 88], MOLE [Eshelman et al., 93] et SALT [Marcus & McDermott, 93]. Nous décrivons ici, le système MOLE qui est selon nous l'outil le plus représentatif mettant en pratique cette méthodologie.

MOLE, en tant que successeur du système MORE, a été conçu pour améliorer les aspects liés à la "qualité" du dialogue avec l'expert, entre autres le fait que l'expert soit obligé de manipuler et d'associer des valeurs numériques pour définir la certitude d'une hypothèse vis-à-vis d'un ensemble de symptômes. Ainsi, pour résoudre ce problème, MOLE interprète les réponses de l'expert et gère lui-même la traduction numérique de l'incertitude sur les connaissances.

L'utilisation de MOLE, pour une application donnée suppose quatre conditions [McDermott, 88]:

1. la possibilité d'identifier un ensemble fini d'événements (symptômes, malfonctionnements),
2. la capacité d'énumérer, pour chaque événement, une ou plusieurs explications possibles,
3. l'existence de connaissances de différentiation, qui permettent de choisir entre différentes possibilités en évaluant la capacité d'une hypothèse à expliquer un symptôme,
4. le postulat selon lequel il n'y a qu'une seule explication pour un événement.

MOLE interroge l'expert en lui demandant d'abord les événements dont on recherche la cause et les explications possibles pour chaque événement. Avec ces informations, MOLE instancie le réseau causal qui défini les connexions entre les événements et les explications. Ensuite, le système cherche à trouver les connaissances permettant d'éliminer certaines hypothèses. De cette manière, le réseau initial est affiné et complété au cours des différents tests (l'expert peut demander des explications pour valider les inférences composant le réseau causal) [Aussenac, 89].

MOLE a été utilisé pour la construction de plusieurs systèmes spécialisés dans les problèmes de diagnostic. Cependant, comme l'admet d'ailleurs Eshelman, dans certains cas, MOLE peut échouer. En effet, dans certaines situations, il peut s'avérer impossible d'énumérer toutes les solutions possibles, car une solution peut être une combinaison de solutions partielles viables [LeRoux, 94].

4.3.3 Open-KADS: un atelier pour rendre opérationnelle l'approche KADS

La méthodologie KADS a été longtemps critiquée comme étant la seule dont les modèles ne sont pas opérationnels [Karbach et al., 90]. Aujourd'hui, plusieurs outils, rendent cette théorie utilisable par des applications diverses; Open-KADS est probablement le plus connu d'entre eux.

Open-KADS est un atelier qui s'appuie sur le modèle de l'expertise de KADS. Il distingue deux types de connaissances: les *connaissances statiques* appartenant au niveau domaine et les *connaissances dynamiques* réparties sur les couches supérieures. Open-KADS permet la définition d'un modèle isomorphe au modèle conceptuel de KADS, structuré en quatre couches [Fuentes et al., 94]:

- la *couche domaine* décrit les connaissances statiques en termes de concepts et relations. Les concepts sont caractérisés par leurs attributs et structurés de manière hiérarchique.
- la *couche inférence* concerne les mécanismes d'inférence observés chez l'expert.
- la *couche tâche* modélise la manière dont les inférences s'enchaînent par rapport à une tâche. Dans Open-KADS une tâche est définie par: un nom, sa description, les liens avec les objectifs de la tâche et les inférences. Autrement dit, une tâche possède un but à atteindre par l'intermédiaire d'une procédure.
- la *couche stratégie* décrit le plan général de résolution à l'aide d'un graphe ET/OU.

Pour une application donnée, Open-KADS se focalise sur trois éléments de base: les *documents* de l'expertise, le *glossaire* et le *modèle conceptuel* décrit ci-dessus. Il intègre d'abord les documents élaborés comme la suite des entretiens d'experts dans un *Rapport de l'application* généré de manière automatique une fois tous les documents saisis [Fuentes et al., 94]. Ensuite, lors de la phase d'interprétation, Open-KADS réalise le glossaire contenant l'ensemble des concepts importants du domaine ainsi que leur définition. Enfin, par l'intermédiaire de l'interface graphique et de plusieurs éditeurs spécialisés (un éditeur pour chaque couche), Open-KADS permet de définir le modèle de l'expertise.

L'outil a été testé sur une application réelle baptisée REDRESSE dans le domaine juridique. Cette application s'inscrit dans un projet de développement d'un système d'aide à la gestion de créances des clients de la société BULL (France). Les résultats ont montré l'utilité de Open-KADS notamment lors de la phase d'explicitation-modélisation des connaissances du domaine (le recueil et la structuration de concepts). Plus précisément, cette étape a été réalisée entièrement par des juristes, sans aucune intervention du cogniticien. En revanche, la modélisation des couches supérieures a rencontré plusieurs difficultés, notamment en ce qui concerne la formalisation des connaissances recueillies en termes de stratégie, tâche, inférence car ces notions ne sont pas évidentes pour un non-spécialiste en acquisition des connaissances. Un autre problème majeur a été l'étape d'abstraction du modèle conceptuel de l'application REDRESSE, puisque la plupart des termes utilisés pour décrire les inférences ne sont pas réutilisables pour une autre application [Fuentes et al., 94].

4.3.4 La boîte à outils de "Tâches Génériques"

Plusieurs outils fondés sur l'approche "Tâches Génériques" ont été développés. Étant donné la nécessité de combiner des tâches génériques pour des problèmes complexes, il a fallu construire une boîte à outils (*Generic Task Toolset*) pour réaliser des SBC opérationnels. Chaque outil implémente une méthode particulière pour résoudre un type précis de tâche. Par exemple, l'outil CSRL [Bylander, 86] est dédié à une tâche de classification hiérarchique (cette tâche a été présentée dans la section 4.2.1) tandis que DSPL [Brown & Candrasekaran, 86] est spécialisé dans une tâche de type synthèse.

Chaque outil permet le dialogue avec l'expert, en lui posant des questions spécifiques à la tâche. Néanmoins, aucune aide n'est offerte à l'expert sur la manière de sélectionner une ou plusieurs tâches génériques pour une application donnée [Krivine & David, 91].

4.3.5 COMMET: un outil de modélisation et d'explicitation des connaissances basé sur l'approche "Composants de l'expertise"

La méthodologie "Composants de l'expertise" de Steels est mise en œuvre dans COMMET [Steels, 93] qui est à la fois un outil de modélisation des systèmes et un outil d'explicitation des connaissances.

Comme KADS, le but de COMMET est de couvrir le cycle complet de développement d'un système, cycle qui comporte quatre étapes: *analyse*, *conception*, *acquisition des connaissances* et *exécution de l'application* [Canãmero & Geldof, 93]. La phase d'*analyse* implique la décomposition progressive du problème à résoudre en sous-problèmes en vue d'obtenir une structure de la tâche sous la forme d'un arbre de tâches. Une fois la structure de tâche construite, l'étape de *conception* consiste à définir les modèles manipulés par la tâche: les modèles de cas destinés aux connaissances relatives à un exemple particulier, et les modèles du domaine prévus pour les connaissances statiques du domaine. Les liens entre les tâches faisant partie de la structure de tâche et les modèles sont réalisés par un *diagramme de dépendances des modèles*. Ensuite, lors de la phase d'*acquisition* les modèles du domaine sont instanciés. Enfin, lors de la phase d'*exécution*, les modèles de cas sont instanciés.

Par exemple, pour une application de surveillance du trafic d'un réseau d'autoroutes, un modèle du domaine serait un schéma contenant toutes les autoroutes d'une région particulière, tandis qu'un modèle de cas pourrait définir, par exemple, la surveillance d'une autoroute particulière et contenir un modèle de défaut apparaissant dans une situation bien précise, un modèle de réparation, etc.

COMMET, en tant qu'outil d'explicitation, facilite le dialogue avec l'expert en lui offrant de multiples aides et conseils.

L'intérêt d'utiliser COMMET est lié à deux caractéristiques essentielles:

- son double rôle d'outil de modélisation et d'explicitation permet à l'expert de construire une application du début jusqu'à la fin car c'est l'expert qui définit le problème et qui finalement l'exécute,
- COMMET dispose d'éditeurs graphiques pour chaque méthode. Ceci facilite le travail de l'expert et permet un prototypage rapide de l'application.

COMMET a été utilisé avec succès dans le cadre du système DIAPASON, un système de formation et d'entraînement [Moinard & Joab, 94]. Nous reviendrons sur ces aspects dans le chapitre 5.

Ce chapitre a abordé les travaux majeurs qui ont fondé l'approche de modélisation des connaissances. Ensuite, nous avons décrit le passage d'une théorie d'expertise à sa réalisation pratique en décrivant plusieurs outils que nous considérons représentatifs. Les idées présentées ici servent de support aux chapitres suivants.

Dans cette étude, nous nous intéressons à l'acquisition des connaissances pour la production de STI. Le projet SAFARI, cadre de cette étude, repose sur l'utilisation de modèles génériques. C'est donc tout naturellement que notre travail s'inscrit dans le cadre de l'école "choisir pour suivre". L'approche d'acquisition que nous avons retenue se rapproche de celle des "tâches génériques" et des "méthodes à limitations de rôles". Elle réutilise néanmoins l'idée de Steels relative à la nécessité de réaliser d'abord une analyse détaillée de la tâche avant de passer à sa résolution (résolution dans le sens "chercher la solution"). Mais avant de décrire plus précisément cette approche d'acquisition pour la production de STI au sein de SAFARI, ainsi que les trois types de méthodes proposées pour supporter l'analyse des tâches (méthode de décomposition, méthode de composition et méthode mixte), le chapitre suivant nous permettra de tirer quelques enseignements de différentes tentatives antérieures d'application de l'acquisition de connaissances à la production de STI.

Chapitre 5

L'acquisition des connaissances dans le domaine des STI

Nous avons présenté, dans le chapitre deux, les objectifs et les principes généraux des STI. Nous avons souligné, dans les chapitres trois et quatre, qu'en acquisition des connaissances convictions méthodologiques et outils d'acquisition convergent aujourd'hui vers une approche constructive de modélisation.

Ce chapitre se donne pour objectif de répondre à trois questions:

- quelles sont les spécificités de l'acquisition des connaissances dans le domaine des STI?
- de quelle manière les principes de base dans le domaine de l'acquisition ont influencé les recherches sur les STI jusqu'à présent?
- comment l'acquisition des connaissances peut résoudre les difficultés rencontrées lors de la réalisation des STI?

5.1 Contexte de l'acquisition des connaissances dans les STI

Comme nous l'avons vu dans le chapitre deux, la réalisation d'un STI doit s'intéresser aussi bien au domaine à enseigner qu'aux théories pédagogiques, ainsi qu'aux moyens de faciliter le processus de formation (par exemple, outils multimédia).

5.1.1 Les étapes de la construction d'un STI

Influencées par la psychologie cognitive, les recherches dans le domaine des STI ont été dirigées vers deux types de "pratiques pédagogiques" [Mengelle, 95]:

- les "*pratiques pédagogiques centrées sur le contenu de la formation*", qui reposent sur le courant béhavioriste, en se focalisant sur la description, l'analyse et l'organisation de la matière à enseigner;
- les "*pratiques pédagogiques centrées sur l'apprenant*", motivées par le courant cognitiviste, où l'on considère que l'apprenant joue un rôle actif dans le processus de formation et qu'il faut, en conséquence, adapter l'enseignement à certaines caractéristiques de l'élève.

Malgré les différences de principe entre ces courants de recherche, leur mise en œuvre nécessite de nombreuses connaissances, même si celles-ci sont de nature différente. D'ailleurs, les deux types de pratiques pédagogiques abordent la réalisation d'un STI comme étant un processus qui passe par deux étapes essentielles:

- la *phase d'acquisition des connaissances* de l'expert par le système,
- la *phase de "transmission"*, c'est à dire d'utilisation des connaissances acquises lors du processus de formation.

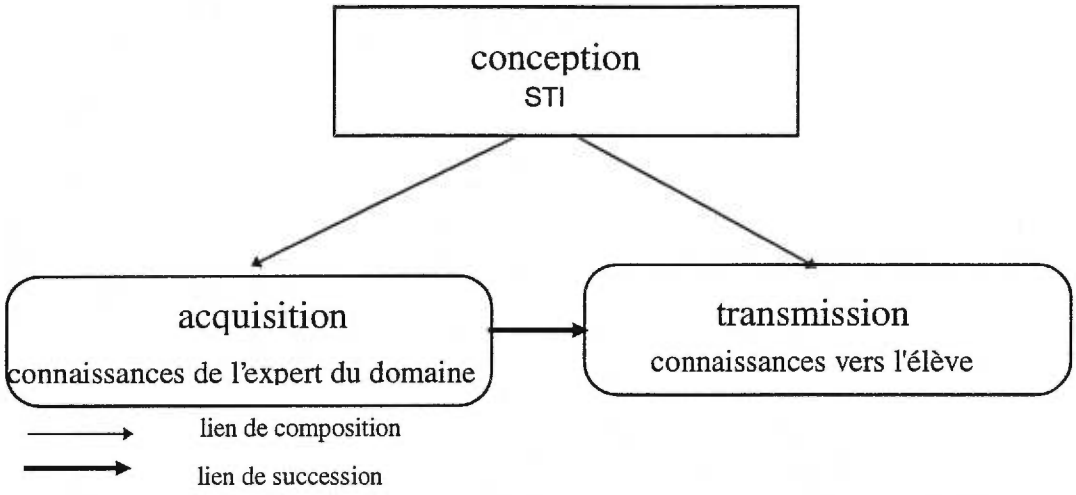


figure 5.1: Les étapes de conception d'un STI.

À notre connaissance, l'effort consacré pour ces deux activités, l'acquisition et la formation, n'a pas été distribué de manière équilibrée. En effet, le processus de formation à été l'objectif principal dans la plupart des recherches dans les tuteurs intelligents. Nkambou souligne d'ailleurs le fait que peu des travaux dans le domaine se sont focalisés sur la nécessité d'un curriculum dans un STI [Nkambou & Gautier, 96]. Ceci fait que le problème d'acquisition reste une des questions abordées mais non résolues dans ce domaine. S'il est vrai que l'objectif final d'un STI est l'enseignement (d'où l'intérêt pour la deuxième phase), il n'en demeure pas moins que la "qualité" de l'activité de formation dépend en grande partie du contenu de la base de connaissances.

5.1.2 La pluralité de l'expertise dans un STI

Lorsqu'on considère l'acquisition des connaissances dans les tuteurs intelligents il s'agit d'identifier et d'organiser au minimum deux expertises distinctes: celle de l'expert du domaine et celle du pédagogue. Cependant, les rôles de l'expert et de l'enseignant ne sont pas toujours clairement identifiés. On s'aperçoit souvent que plusieurs auteurs considèrent que le pédagogue est également capable de fournir les connaissances liées au domaine, de façon que l'apport de l'expert du domaine dans la construction d'un STI devient négligeable. Il nous semble important de critiquer ce point de vue simpliste: une méthodologie de conception rigoureuse doit viser d'abord la complétude et la validité du

système final et pour cela il faut obtenir toutes les connaissances dont ce dernier a besoin. Les expertises de l'expert du domaine et du pédagogue sont complémentaires et donc indispensables lors du processus de conception d'un STI.

Appliquer les principes de l'acquisition des connaissances dans le cadre de la conception d'un STI comporte deux avantages:

- Étant donné le fait que la plupart des recherches en acquisition des connaissances ont l'ambition de définir des *méthodes de conception* et de construire des *outils d'aide à l'acquisition*, celles-ci peuvent augmenter la productivité lors du développement d'un STI. Les méthodologies qui ont fondé l'approche modélisation comme KADS, COMMET, etc., peuvent permettre d'augmenter la productivité lors de la conception d'un STI. Ces méthodologies ont été conçues principalement pour servir comme guides de développement en fournissant les spécifications du système à base de connaissances à construire.
- Si, lors du processus d'acquisition, des connaissances "explicatives" (qui justifient le comportement de l'expert) sont recueillies, ces connaissances pourront servir comme aides et conseils pour l'apprenant, par exemple quand il sera en train de résoudre un problème. Dans ce cas, l'outil d'acquisition pourra jouer aussi le rôle d'un système conseiller (il dispose au moins des connaissances nécessaires pour un tel système) lors du processus de formation.

5.1.3 Le besoin d'assistance à l'utilisateur

Le fait que la performance d'un système puisse être accrue en augmentant sa capacité de coopérer avec ses utilisateurs est déjà reconnu. En effet, comme souligne Ferber

"*la coopération consiste à amplifier les capacités des agents pris isolément et à augmenter leurs performances*" [Ferbert, 94].

Étant donné qu'un STI performant doit s'appuyer sur l'interaction avec plusieurs intervenants (expert du domaine, pédagogue, élève), le problème de la coopération entre ces agents (humains ou informatiques) doit être pris en compte. Cet aspect a été d'ailleurs le point

central de plusieurs travaux dans le domaine des STI, notamment dans ceux de Murray, concernant le développement du système Cardiac Tutor [Murray & Woolf, 93].

L'assistance à l'utilisateur est un aspect important de la coopération homme-machine. L'assistance à l'utilisateur peut être abordée de différents façons lorsque on s'adresse à un concepteur de STI et, en conséquence, il existe plusieurs types de systèmes d'assistance:

- les systèmes d'aide en-ligne fournissant un aide standardisé à l'initiative de l'utilisateur,
- les systèmes dits "advice-giving" capables de répondre à la demande de l'utilisateur par un aide contextualisé,
- les systèmes dits systèmes"coochs" fournissant le même type d'aide à leur propre initiative,
- les systèmes dits critiques capables d'évaluer le travail effectué par l'expert (soit durant celui-ci, soit après sa validation).

Mais au delà de toutes ces aides logicielles, l'assistance concerne plus globalement le processus dans son ensemble : *comment structurer le dialogue expert-machine pour faciliter l'acquisition des connaissances?* C'est essentiellement sous cet angle que nous abordons l'assistance à l'expert dans ce mémoire.

En acquisition de connaissances, plusieurs travaux [Jacob & Jehl, 96] [Willamowski, 94] [Zacklad, 93b] se sont orientés vers la conception de systèmes coopératifs qui reposent essentiellement sur une participation conjointe du système et de l'utilisateur. Ceci implique d'une part que le système soit capable de résoudre le problème de manière autonome et d'autre part que l'utilisateur puisse contrôler (diriger) lui même le processus de résolution [Willamowski, 94]. Plus précisément, la conception d'un tel système doit prévoir deux modes de fonctionnement:

- le premier, qui est d'ailleurs caractéristique de la majorité des systèmes, dans lequel le système dirige l'activité de l'utilisateur;
- le second, qui, au lieu d'imposer à l'utilisateur une stratégie particulière, s'appuie sur le partage des tâches entre système et usager. Plus précisément, le système demande des

informations à l'utilisateur en conservant toutefois possibilités de suspension, d'abandon ou de changement d'activité au cours d'une session de consultation [Ermine, 93].

Dans la littérature sur les STI ce thème a été abordé seulement dans un contexte d'apprentissage: beaucoup de recherches ont eu comme objectif le problème de communication élève-système ou pédagogue-système. Un exemple est l'assistant pédagogique ROBOTEACH [Leroux, 96].

Nous pensons que, dans le domaine des STI, le problème d'assistance doit aussi être abordé sous un autre angle: il s'agit cette fois-ci de faire intervenir l'expert du domaine en vue d'expliciter son savoir. Pour cela il a besoin d'aide et de conseils, car, comme les travaux en acquisition des connaissances l'ont montré, trouver les "bonnes" connaissances et ensuite les organiser a été souvent un objectif délicat.

Étant donné que l'existence d'une base de données complète qui détient assez d'informations pour pouvoir répondre aux exigences des élèves est un critère primordial pour apprécier la qualité d'un STI, nous concluons cette section en soulignant à nouveau que pour la création d'un tel système doit s'appuyer sur la construction d'outils dédiés à l'expert qui peuvent lui offrir l'assistance nécessaire lors du processus d'acquisition des connaissances de la matière.

5.2 Expériences en acquisition des connaissances pour la production de STI

Depuis quelques années, plusieurs auteurs ont insisté sur une coopération entre les deux disciplines: l'acquisition des connaissances et l'E.I.A.O. Nous voulons montrer ici que l'utilisation des techniques d'acquisition dans le contexte de développement d'un STI peut s'avérer profitable pour ce dernier, surtout pour la modélisation des différentes expertises impliquées (au moins pour organiser les connaissances du domaine et celles de nature pédagogique). Pour cela, nous avons choisi de décrire deux outils: DIAPASON et GEOMUS car les recherches les concernant se situent dans le contexte de modélisation des connaissances.

La première partie de cette section traite de DIAPASON et s'intéresse donc aussi bien aux techniques d'explicitation des connaissances qu'à la modélisation de l'expertise en utilisant deux méthodologies d'acquisition: COMMET et KADS.

La deuxième partie présente GEOMUS et se focalise sur la description d'un mécanisme de résolution de problèmes de géométrie appuyé sur une analyse de la figure.

5.2.1 L'acquisition des connaissances pour le système DIAPASON

DIAPASON est le nom d'un projet dont l'objectif est la conception d'un système d'aide à la formation des chargés de conduite d'un réseau électrique moyenne tension utilisé pour alimenter la clientèle [Moinard & Joab, 94].

Le métier de chargé de conduite comporte principalement une activité de surveillance de l'alimentation continue du réseau. En cas d'interruption d'alimentation, le chargé de conduite a pour devoir prioritaire la réalimentation du maximum de clients le plus rapidement possible. Ceci implique tout d'abord une tâche de diagnostic du problème ayant provoqué la panne.

L'analyse de cette expertise dégage deux types de connaissances.

- *Les connaissances sur le fonctionnement du réseau* sont considérées comme connaissances de base pour analyser une situation de panne. Elles décrivent les différents composants du réseau ainsi que leur mode de fonctionnement. La surveillance du réseau est réalisée à l'aide de plusieurs systèmes appelés *protections* pour détecter les anomalies et pour interrompre l'alimentation dans la zone où s'est produit un défaut et d'*automatismes* dont le rôle est d'améliorer le service soit en évitant une coupure si l'anomalie a disparu, soit en réalimentant une zone par un autre poste.
- *Les connaissances opératoires pour la conduite* constituent l'ensemble des actions à effectuer par l'expert pour remédier à une situation défectueuse.

Le processus d'acquisition a été divisé en deux étapes: *explicitation des connaissances du domaine*, puis processus de *modélisation de l'activité experte*.

L'explicitation des connaissances est orientée surtout vers les connaissances relatives au fonctionnement du réseau. Selon les experts, qui doivent adapter leur travail en fonction de l'état du réseau, ces connaissances sont des "connaissances profondes" (alors que connaissances de conduite relèvent des "connaissances de surface"), puisqu'elles permettent de justifier leurs actions. De plus, ces connaissances sont essentielles lors du processus de formation, car elles permettent d'apprendre aux stagiaires la manière d'analyser tout problème lié au domaine. Cette analyse est indispensable avant toute intervention.

Plusieurs experts (des chargés de conduite ainsi que des formateurs) ont contribué à l'explicitation des connaissances du domaine. Le travail des chargés de conduite a été centré principalement sur une analyse rigoureuse des différentes situations de panne. Les formateurs ont été consultés en vue d'établir les liens entre les connaissances décrivant des situations réelles et celles dont les apprenants disposent, suite à une formation initiale.

Après plusieurs réunions de travail, une *grille d'analyse* a été conçue et validée par les experts. Cette grille associe pour chaque problème, un ensemble de symptômes, les réactions du réseau (par exemple, quelles alarmes ont été déclenchées), les actions de reprise qu'un chargé de conduite doit réaliser, ainsi que les manœuvres qui sont interdites.

L'intérêt de cette étape est d'expliciter les connaissances du domaine indépendamment de l'activité de l'expert (les tâches qu'il est censé faire).

La modélisation de l'activité experte a été effectuée en se basant sur deux méthodes: COMMET et KADS.

COMMET a été utilisée pour identifier les principales "composantes de l'expertise" selon la définition de Steels [Steels, 90], [Steels, 93] qui décrit le comportement de l'expert en termes de *tâches* (quoi faire), de *modèles* de connaissances nécessaires à la réalisation de la tâche (avec quoi faire) et de *méthodes* (comment faire). Le résultat de cette première étape de modélisation est une décomposition de la tâche globale en sous-tâches. Comme nous l'avons vu auparavant, la tâche de chargé de

conduite implique deux sous-tâches principales: une tâche de *diagnostic*, puis une tâche de *réparation*. Ensuite, la structure de tâche est complétée en évaluant par comparaison les situations de panne contenues dans la grille d'analyse suite à l'étape d'explicitation. Les problèmes produisant des symptômes similaires ont été regroupés dans le même ensemble. Pour chaque ensemble, les réactions du réseau sont analysées. Ensuite, les actions entreprises par les experts sont analysées en vue de distinguer les différents cas.

Une fois la structure de la tâche définie, des *modèles de cas* et des *modèles de domaine* ont été utilisés: les premiers pour décrire des tâches complexes (qui possèdent un niveau élevé d'abstraction) et les deuxièmes pour identifier les connaissances manipulées pour chaque tâche.

Lors d'une session avec DIAPASON, seule la phase d'*analyse* de COMMET a été utilisée. Les autres étapes (conception, acquisition et exécution) décrites dans la section 4.3.5 n'ont pas été abordées, car elles auraient conduit à choisir (ou écrire) des procédures implémentées en LISP pour les attacher aux différentes méthodes [Moinard & Joab, 94].

Par la suite, la méthodologie KADS s'est avérée utile pour affiner la structure de tâche définie avec COMMET. KADS offre une bibliothèque de modèles d'interprétation adaptés aux différents types de problèmes (section 4.1.4). Comme souligne [Duribreux & Houriez, 96], le choix du modèle d'interprétation approprié n'est possible qu'une fois le raisonnement de l'expert bien identifié. Ce choix est facilité dans le cas du système DIAPASON par la structure de tâche définie avec COMMET. En effet, le travail avec COMMET a permis de traiter le niveau tâche de KADS. De plus, la couche domaine a été structurée et implémentée en utilisant une méthode orientée objets [Shlaer & Mellor, 92]. Ensuite, KADS n'a servi qu'à élaborer le niveau d'inférence.

Après l'étape de modélisation, le modèle de l'expertise a servi de base pour la mise au point d'un *dossier de spécifications*, indispensable à la phase d'implémentation.

Cette expérience de travail constitue un bon exemple décrivant l'utilité des méthodes d'acquisition dans la réalisation d'un système de formation.

5.2.2 GEOMUS: un système de résolution de problèmes de géométrie élémentaire

Le système GEOMUS [Bazin, 93] a pour objet de modéliser la résolution des exercices de géométrie élémentaire. Ceci implique la construction d'un modèle de l'expert pour décrire son raisonnement pendant qu'il résout un problème de géométrie de quatrième.

En tant que "résolveur de problèmes", GEOMUS doit posséder des connaissances de nature diverse. Il s'agit d'abord d'un ensemble des connaissances sur les éléments de base de figures géométriques, par exemple, les théorèmes et les axiomes sur les triangles. De plus, le système doit connaître aussi les différentes stratégies appliquées par des spécialistes du domaine à des problèmes de géométrie. Les auteurs ont utilisé un processus d'acquisition manuel, basé sur l'observation des mathématiciens (les experts) au travail.

Lors de la séance d'observation, chaque expert reçoit quatre descriptions littérales de figures géométriques. Il doit alors tracer les figures et formuler des questions pertinentes à partir de ces figures.

En analysant les résultats de la séance d'observation, les professeurs de mathématiques ont proposé trois catégories de questions:

- la première catégorie demande aux élèves de reconnaître le type des différents objets faisant partie de la figure concernée. Une question de ce type est "quelle est la nature du triangle ABM ? (voir figure 5.2),
- un autre groupe de questions fait référence aux "transformations géométriques" (rotations, symétries, etc.) qu'on peut obtenir à partir du dessin initial, par exemple, "Comment peut-on, à partir d'un triangle ABC, construire un triangle EFD dont les sommets sont respectivement les milieux de [AB], [BC] et [CA] ?",
- enfin, la dernière catégorie se focalise sur l'introduction de nouveaux éléments sur la figure initiale. Par exemple, un professeur a demandé, entre autres, de trouver l'orthocentre d'un triangle faisant partie de la figure.

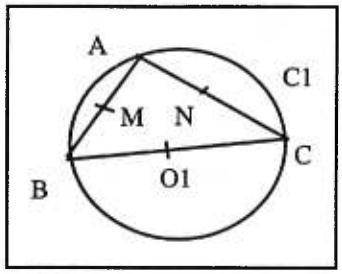


figure 5.2 : Figure géométrique.

Puisque l'analyse de figures géométriques est l'activité dominante chez l'expert, sa modélisation est un objectif important visé par le concepteur du système. Ainsi, lorsque GEOMUS analyse une figure, il extrait tout d'abord des sous-figures la composant. Par exemple pour la figure 5.2 il extrait les sous-figures suivantes (figure 5.3).

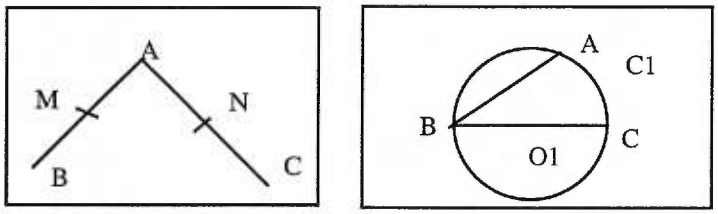


figure 5.3: Les sous-figures pour la figure 5.2.

Cette décomposition est basée sur la considération qu'un expert humain, lorsqu'il construit la figure appropriée à un énoncé particulier extrait d'abord des *configurations caractéristiques* de la figure. Ensuite, l'expert attribue une "étiquette" à chaque sous-figure. Une étiquette désigne l'objet d'une leçon ou le titre d'un chapitre (ou sous-chapitre) du livre comme par exemple, parallélogramme, triangle-rectangle, etc.

L'expertise mathématique est modélisée par le concepteur sous forme de règles qui associent aux sous-figures des étiquettes de problèmes (figure 5.4).

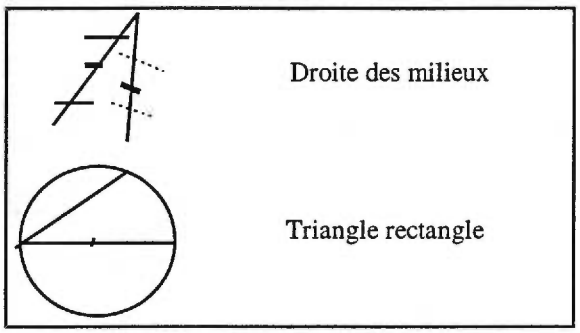


figure 5.4: Configurations caractéristiques et étiquettes de problèmes.

Une étiquette est donc une référence pour l'ensemble des connaissances qui lui sont associées. L'ensemble des étiquettes est organisé sous forme d'un graphe orienté où chaque nœud est une étiquette et le fils d'un nœud est une spécialisation de ce dernier (figure 5.5). Si les associations étiquettes-ensembles des connaissances permettent d'accéder aux connaissances spécifiques d'une étiquette, la structure hiérarchique offre la possibilité d'obtenir les connaissances relatives à tous ses ascendants dans l'arbre.

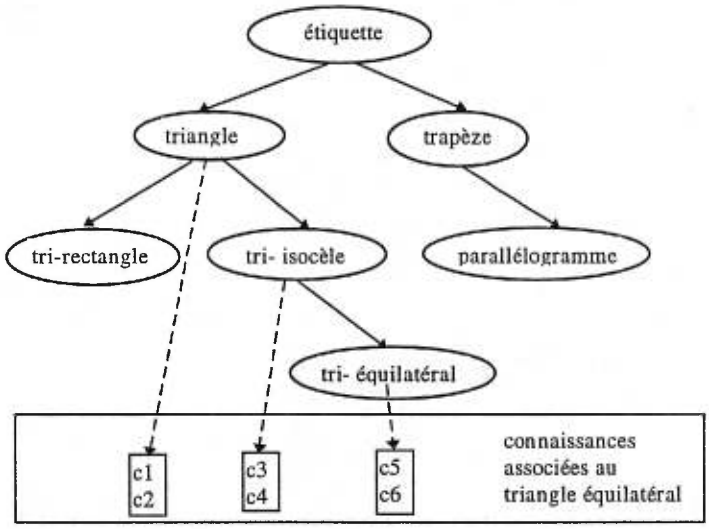


figure 5.5: Hiérarchie des étiquettes.

La recherche de la solution est un processus cyclique contenant deux étapes:

- En analysant la figure du problème initial, GEOMUS choisit un ensemble de connaissances (les théorèmes et le savoir-faire de l'expert attachés à une étiquette)

pour traiter le problème. Ceci conduit à la modification du problème ou à sa résolution.

- Supposons que le système ne soit pas capable de déduire la solution. En conséquence le problème initial est transformé en un nouveau problème, par l'ajout des nouvelles hypothèses et éventuellement par la modification de la figure. La phase d'analyse est invoquée de nouveau sur le problème enrichi. Le système a donc utilisé un nouvel ensemble de connaissances. Ce processus est se termine lorsque le système déduit la conclusion demandée.

Le comportement du système est modélisé en utilisant cinq types de règles qui manipulent la figure du problème:

- *Les règles de construction* décrivent la *représentation interne de la figure*, plus précisément, sa décomposition en sous-figures ainsi que tous les éléments nécessaires pour la décrire (le système manipule plusieurs listes contenant les sommets, les segments, etc. qui caractérisent une figure).
- *Les règles d'exploitation* recherchent les configurations caractéristiques à partir de la figure. L'application de ces règles permet au système de choisir certaines configurations et de proposer plusieurs étiquettes pour le problème.
- *Les règles d'étiquetage* manipulent le graphe des étiquettes en vue d'obtenir les étiquettes les plus spécialisées. À l'issue de cette phase les étiquettes moins spécialisées sont éliminées de la liste des étiquettes possibles. Par exemple, si cette liste a pour éléments: Triangle et Triangle isocèle, à la fin de cette étape, elle ne contiendra que l'élément Triangle isocèle.
- *Les règles d'appel de connaissances* activent les connaissances associées à une étiquette du graphe ainsi qu'à ses ascendants.
- *Les règles représentant les connaissances* du cours sont appelées par les règles d'appel de connaissances.

La simulation de l'activité de l'expert consiste en l'application de différents groupes de règles, l'ordre d'activation de ces règles étant établi par des métarègles.

Le grand mérite de GEOMUS réside dans son objectif d'élaborer un modèle de l'expert. En effet, il s'agit d'une notion peu abordé dans la communauté STI. Ce modèle est le résultat de l'observation directe de l'expert durant la réalisation d'une activité particulière (ici, la résolution d'une problème de géométrie) et d'une analyse cognitive de sa tâche. Plus précisément, cette analyse se focalise sur un mécanisme de décomposition de la figure du problème en sous-figures en vue d'identifier l'élément clé (l'étiquette de la sous-figure) dont les connaissances associées permettront au système de déduire la bonne conclusion.

En revanche, GEOMUS ne permet pas le dialogue direct avec l'utilisateur. De plus, l'ordre de déclenchement des différentes catégories de règles est figé par les métarègles. Un des objectifs de l'auteur est d'adapter cet ordre d'activation des règles compte tenu du problème à résoudre.

5.2.3 L'acquisition dans les STI: discussion

Le développement d'un STI relève, dans la majorité des cas, de l'approche utilisée en génie logiciel: un développement intégral divisé en une succession d'étapes. Les systèmes présentés auparavant ne sont que deux exemples. En effet la réalisation de ces systèmes passe par toutes les phases de développement d'un système [Hayes-Roth et al., 83]:

- identification du problème,
- conceptualisation,
- structuration des connaissances,
- implémentation,
- validation[1].

Si tel est le cas cette approche s'est heurtée aux mêmes problèmes que les systèmes experts de première génération. En effet, fabriquer pour chaque problème un système "à partir de rien" est inadapté aux contraintes pratiques.

En conséquence, une préoccupation majeure dans le domaine des STI doit être de raccourcir le processus de développement d'un système. Nous notons ici trois aspects permettant d'économiser une partie du travail d'acquisition des connaissances dans un STI:

- Le premier est la *réutilisation* des connaissances. Ceci s'appuie sur la définition des *modèles génériques* pouvant être utilisés au sein de plusieurs applications;
- Le deuxième vise la coopération entre les différents utilisateurs d'un STI. Dans SAFARI, nous abordons la coopération selon deux points de vue :
    1. le premier type se situe entre le système et son utilisateur,
    2. le deuxième a comme objectif la coopération de plusieurs expertises [Tadié, 96]. Pour cela il fait référence à l'approche multi-agents qui s'appuie sur une architecture logicielle englobant différentes sources de connaissances (dans cette approche les connaissances sont distribuées entre plusieurs

[1] Dans un STI la validation n'est pas seulement "informatique" elle doit être également "pédagogique

experts) et des moyens de communication entre ces dernières, soit par l'intermédiaire d'un tableau noir [Laâsri & Maître, 89], soit par envoi de messages.

Nos travaux ne traitent que la coopération système-utilisateur.

- Le troisième se focalise sur la *nécessité d'améliorer l'explicitation des connaissances.* De manière générale, l'explicitation dans le cadre des STI est réalisée par l'intermédiaire d'éditeurs dédiés au cogniticien ou à l'informaticien qui sont familiarisés avec ses outils, plutôt qu'à l'expert du domaine. Autrement dit, la plupart de ces éditeurs, même s'il font référence à un ou plusieurs modèles de l'expertise, ne disposent pas d'un langage de communication compréhensible par l'expert, capable de lui permettre de structurer ses connaissances par interaction directe avec le système.

5.3 Comment l'acquisition des connaissances peut-elle résoudre les difficultés rencontrées lors de la réalisation des STI?

Nous constatons que les deux disciplines, l'acquisition des connaissances et la réalisation des STI, se sont rejointes sur une préoccupation commune : comment augmenter la productivité de développement de nouveaux systèmes. Comme souligne Le Roux, une coopération entre ces deux disciplines peut s'avérer profitable pour ces deux domaines de recherche [LeRoux, 93].

A l'image de cette remarque nous considérons peu satisfaisante la façon linéaire de voir la conception d'un STI premièrement comme un processus d'acquisition en vue de fournir les éléments nécessaires à la construction du module curriculum et ensuite l'utilisation de ce dernier pour réaliser l'étape de formation.

Nous considérons que la réalisation d'un STI fiable demande d'abord un changement d'optique: au lieu de considérer l'acquisition et la formation comme deux phases successives, il faut plutôt les voir comme deux activités interconnectées (figure 5.6).

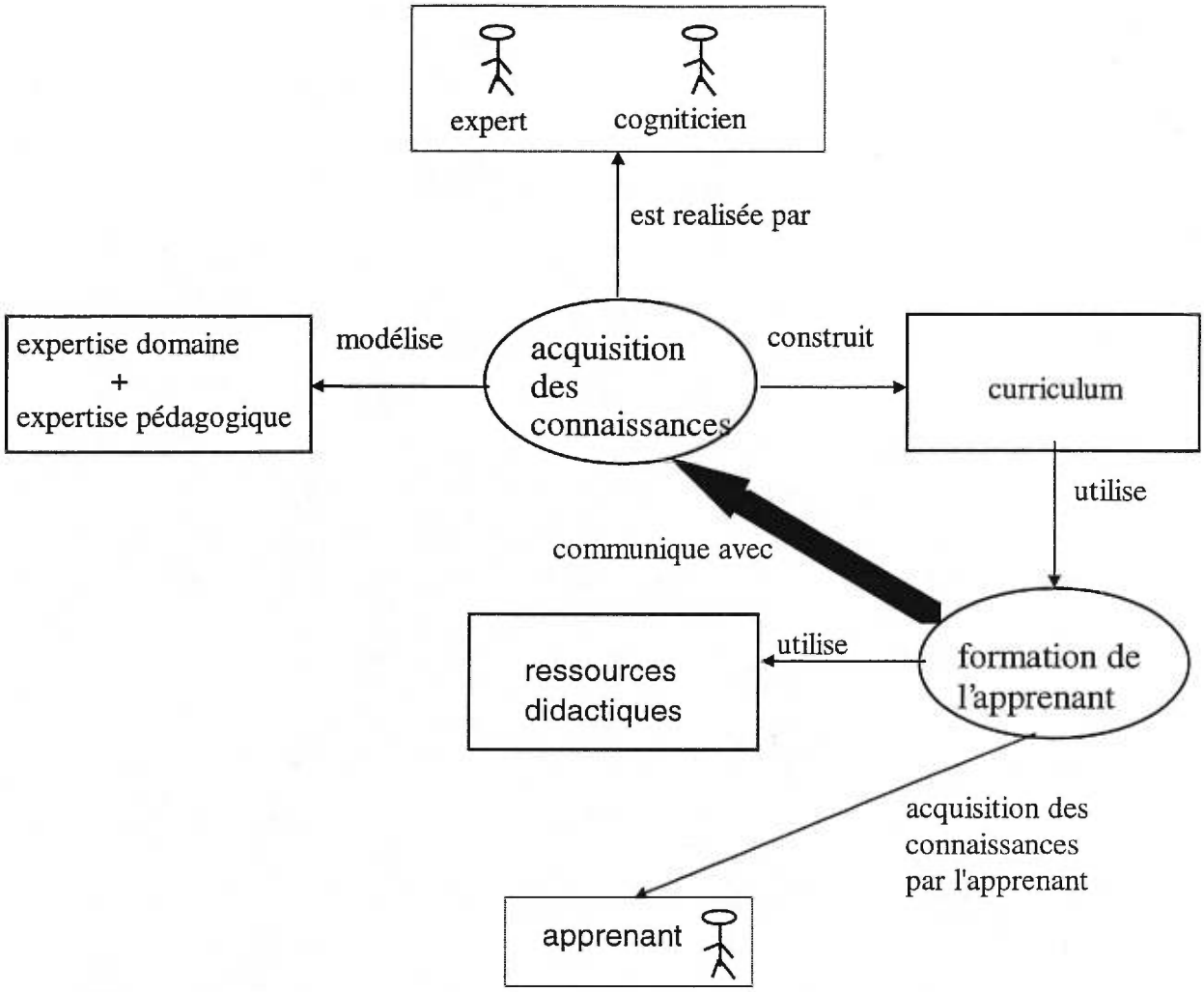


figure 5.6: La communication entre l'acquisition des connaissances et la formation de l'élève.

Ainsi, l'acquisition ne sert pas seulement à produire les connaissances pour le curriculum mais ce processus peut être utile lors du processus de formation même.

Selon cette nouvelle perspective l'expert est impliqué aussi bien dans la phase d'acquisition et validation des connaissances, que dans le processus de formation même. En réalité, comme nous l'avons souligné dans la section 5.1, la formation peut aussi demander du "feedback" qui nécessite encore une fois l'intervention de l'expert. Par exemple, il peut s'avérer nécessaire de fournir des explications ou justifications précises

de la part de l'expert qui n'ont pas été prévues initialement, quand la base de connaissances a été construite.

Nous voulons souligner ici que l'intégration des outils d'acquisition dans un STI peut faciliter le travail du concepteur et/ou de l'informaticien. Nos convictions se basent sur les idées suivantes.

- Certains outils de ce type permettent à des experts de domaine de construire eux-mêmes "sans programmation", donc sans aucun apport de la part du concepteur et/ou de l'informaticien, leurs propres applications [Aussenac, 96]. En effet, les outils d'acquisition peuvent automatiser une partie de la tâche du cogniticien ou du concepteur du système et en conséquence, ceci peut augmenter la productivité de développement du système final.
- Le fait que l'expert ait la possibilité de travailler directement avec le système, lui donne une motivation et simplifie l'étape de validation des connaissances saisies (ceci se base sur le constat que l'expert fait moins d'erreurs lors du processus d'explicitation qu'un intervenant qui ne connaît pas le domaine).
- Puisqu'un STI requiert toujours maintenance et développement, ces outils permettent aux experts d'intervenir à tout moment, soit pour fournir au système des nouvelles connaissances, soit pour corriger celles existantes dans la base de connaissances. Ceci peut entraîner une évolution du système par augmentation et validation de ses connaissances. De plus, de cette manière, on peut éviter la propagation des erreurs dans tout le processus de développement du système.

Nous concluons ce chapitre, en soulignant à nouveau que les travaux en acquisition de connaissances constituent des sources importantes permettant de résoudre les problèmes qui se posent dans le domaine des STI. En effet, comment pourrait-on, sans disposer d'une base de connaissances riche et bien structurée, expliquer à un élève toutes ses questions sur un domaine particulier?

Chapitre 6

COSI: un assistant d'explicitation de tâches en coopération avec l'expert

Notre travail s'inscrit dans le contexte décrit dans le chapitre précédent: l'acquisition des connaissances dans les tutoriels intelligents. Au sein de ce courant de recherche nous nous sommes intéressés a une problématique particulière: comment expliciter les connaissances de type tâche en coopération avec un expert. Après avoir présenté la manière dont l'acquisition des connaissances a été abordée dans le projet SAFARI, nous proposons notre approche d'acquisition. Notre démarche met l'accent sur l'implication de l'expert (qui travail directement avec le système) lors du processus d'explicitation d'une tâche. Cela nous a amené à considérer des principes provenant de la psychologie cognitive et de l'ergonomie cognitive lors de la conception du système COSI (COgniticien SImulé). Ce dernier est un outil d'acquisition qui utilise un modèle de tâches générique (section 6.1.1.3) pour guider le processus d'explicitation. Ensuite, nous précisons l'architecture du système d'acquisition. Le cœur de ce chapitre est constitué de la section 6.3 décrivant notre méthode de conception, les stratégies d'explicitation de tâches et les heuristiques que nous proposons. Enfin, la structure du dialogue fait l'objet de la section 6.4.

6.1 L'acquisition des connaissances dans le projet SAFARI

Comme déjà signalé dès le chapitre 2, le principal composant d'un STI développé dans SAFARI est le curriculum. Dans cette étude, nous nous focaliserons donc sur l'acquisition des connaissances du curriculum; nous tâcherons toutefois de proposer des solutions génériques réutilisables pour d'autres composantes du STI (par exemple, les Stratégies Pédagogiques). Nous structurons cette section en trois points. Nous présentons tout d'abord en détail la structure du curriculum en insistant en particulier sur le formalisme du graphe de tâches. Dans un second point, nous soulignons les limites des éditeurs actuellement disponibles pour la saisie des différentes connaissances impliquées et énonçons les motivations pour le développement d'outils d'acquisition plus performants. Enfin, le troisième point nous permet de spécifier la position de l'outil d'acquisition que nous nous proposons de développer par rapport au système SAFARI.

6.1.1 Structures des connaissances du curriculum

La notion de curriculum a fait couler beaucoup d'encre et plusieurs définitions ont été données [Chan, 92], [Halff, 88], [Nkambou, 96]. Toutes ces définitions qui se focalisent essentiellement sur la structuration de la matière à enseigner, en précisant que le curriculum doit s'intéresser à l'organisation des connaissances du contenu indépendamment de la manière dont elles seront enseignées (indépendamment donc de stratégies pédagogiques). En effet, comme Nkambou [Nkambou, 96] le souligne, la modélisation de l'activité d'enseignement peut être laissée à la charge d'autres composantes du STI notamment le *planificateur*, dont le rôle est de construire une leçon en s'appuyant sur le curriculum et en tenant compte de connaissances de l'étudiant, et le *tuteur* qui est censé gérer le processus d'apprentissage (superviser et évaluer l'activité de l'étudiant, etc.).

6.1.1.1 CREAM ou l'interconnexion de trois modèles

Pour rendre explicite la composante curriculum dans un STI, une approche de modélisation de connaissances, baptisée CREAM (Curriculum Representation and Acquisition Model) a été proposée [Nkambou, 96]. Cette approche organise les connaissances d'un curriculum en utilisant trois modèles interconnectés (figure 6.1).

- *Le modèle des capacités* sert à structurer les connaissances du domaine à enseigner (le terme de capacité traduit donc une connaissance du domaine à enseigner, comme par exemple un concept, une règle (procédure), etc).
- *Le modèle des objectifs d'enseignement* permet de définir et d'organiser les buts d'enseignement.
- *Le modèle des ressources didactiques* vise l'organisation des différentes ressources didactiques (problèmes, exercices, démonstrations, vidéo, etc.) qui supportent le processus d'enseignement.

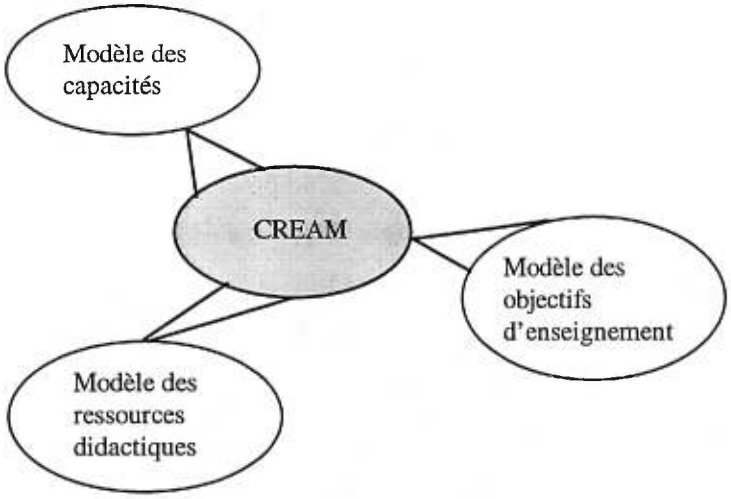


figure 6.1: Architecture de CREAM [Nkambou, 96].

6.1.1.2 Représentation des connaissances pour ces trois modèles

L'approche CREAM implique donc l'acquisition d'une variété de connaissances à enseigner. Ces connaissances peuvent être divisées en deux grandes catégories:

- les connaissances déclaratives (définition de concepts, etc.),

- et les connaissances procédurales (tâches, procédures, stratégies, etc.).

Il est utile d'organiser ces connaissances sur différents niveaux, selon la complexité des modèles impliqués dans la construction d'un Curriculum. Nous obtenons ainsi une hiérarchie entre les modèles, des plus complexes aux plus élémentaires. Cette hiérarchie constitue un support pour le processus d'explicitation des connaissances.

Dans ce sens, la définition du modèle CREAM repose sur un processus de modélisation correspondant à deux niveaux d'abstraction (figure 6.2). Ainsi, partant du besoin d'expliciter les connaissances pour les modèles définis ci-dessous (capacités, objectifs, ressources) trois modèles de représentation[1] des connaissances ont été construits dans SAFARI [Djamen, 95]:

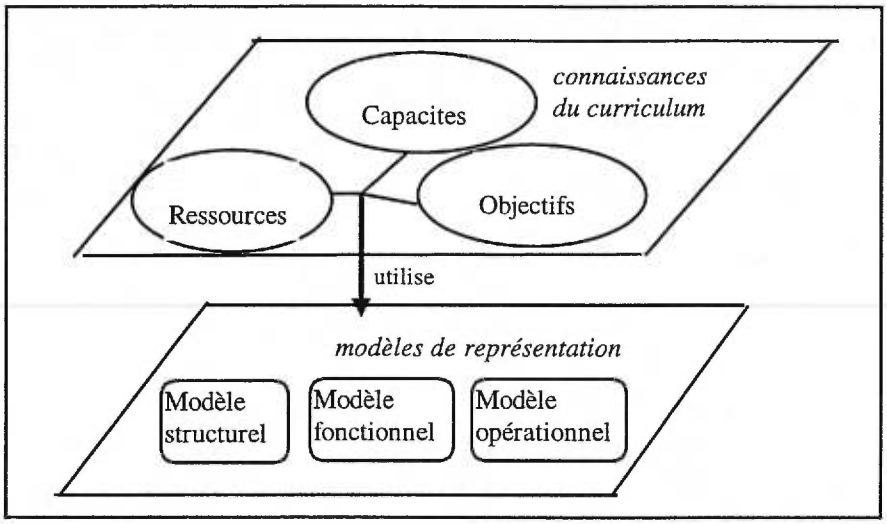


figure 6.2: Modélisation sur deux niveaux dans SAFARI.

1. *Le modèle structurel* décompose un environnement ou un système réel en vue d'identifier et de décrire ses composants.
2. *Le modèle fonctionnel* se focalise sur les relations existantes entre les différents constituants d'un système, pour décrire son fonctionnement.

[1] Il est à noter que chaque modèle de la couche supérieure peut utiliser n'importe lequel de ces modèles de représentation

3. *Le modèle opérationnel* permet de définir la structure d'une tâche (en identifiant l'ensemble de ses sous-tâches) ainsi que l'ordonnancement temporel à l'exécution de ses sous-tâches.

Chaque type de connaissances est modélisé selon un de ces modèles. Selon le type de connaissance considéré, un des ces modèle peut s'avérer approprié. Ainsi, on représente les connaissances déclaratives par le modèle structurel et/ou fonctionnel et les connaissances de type tâche par le *modèle opérationnel* (tableau 6.1).

Types de connaissances	**déclaratives** (définitions, descriptions)	**procédurales** (tâches, stratégies)
Modèle de représentation	• modèle structurel • modèle fonctionnel	• modèle opérationnel

tableau 6.1: Modèles de représentation des connaissances [Alexe, 96].

Quant au besoin d'instancier tous ces modèles, il est clair qu'il faut accorder une importance accrue à l'explicitation des connaissances. Parmi ces modèle de représentation, le modèle opérationnel[2] est celui qui s'est avéré le plus important au sein des prototypes de STI développés en SAFARI (STI globalement basés sur la manipulation des dispositifs simulés). Celui-ci est brièvement décrit dans le paragraphe suivant.

6.1.1.3 Représentation du modèle opérationnel: le graphe de tâches

Le modèle opérationnel [Pachet et al., 95] est une agrégation de nœuds, représentant les connaissances opératoires, et de liens entre ces connaissances. Ce modèle est lui-même le résultat d'une analyse cognitive des tâches [Lesgold et Lajoie, 94] Dans ce sens, une tâche est définie selon deux aspects, sa statique et sa dynamique.

- *Le modèle statique de la tâche* est une décomposition hiérarchique de la tâche en sous-tâches de moins en moins complexes. De cette manière, le problème à résoudre

[2] Il est à noter que le modèle structurel et le modèle fonctionnel ne sont valides que dans le contexte des enseignements lié à la manipulation des appareils physiques. Par contre le modèle opérationnel est toujours valable autant que la connaissance en jeu soit une connaissance procédurale.

est représenté sur différents niveaux d'abstraction. L'acquisition des connaissances pour la statique d'une tâche consiste à identifier toutes les sous-tâches qui la composent.

- *La dynamique de la tâche* définit les contraintes à respecter lors de l'exécution de la tâche. Le graphe de tâches utilise la notion de "*classe de comportement*" pour représenter les relations de précédence spécifiant l'ordre temporel des sous-tâches d'une tâche donnée. Une classe de comportement associée à une tâche, précise le nombre de sous-tâches requises pour accomplir la tâche ainsi que l'ordonnancement de ces sous-tâches lors de l'exécution. L'acquisition des connaissances pour la dynamique de la tâche consiste à identifier pour chaque tâche la classe de comportement adéquate.

Par exemple, considérons le "traitement d'une embolie pulmonaire" (TE) (figure 6.3). Cette tâche consiste à identifier d'abord les "bilans de l'embolie pulmonaire" (BE) et ensuite le "protocole de traitement" (PT). Supposons qu'il y ait deux bilans à faire pour l'embolie pulmonaire: le bilan tumoral (BT) et le bilan ganglionnaire (BG). À son tour, chaque bilan se décompose en une "collecte des données" (CDT, respectivement CDG) suivie de "l'analyse de ces données" (ADT, respectivement ADG). Le modèle opérationnel permet pour ce problème d'associer la classe "Et-strict"[3] à la tâche "traitement d'une embolie pulmonaire" (TE) ainsi qu'aux tâches "bilan tumoral" et "bilan ganglionnaire". Pour la tâche "bilan tumoral" (BT) cela veut dire que cette tâche doit s'effectuer en réalisant les sous-tâches "collecte des données" (CDT) et "analyse de données" (ADT) dans cet ordre strict. En ce qui concerne la tâche "bilans de l'embolie pulmonaire" (BE) la classe de comportement adéquate est "Et-libre" puisque les sous-tâches qui la compose, "bilan tumoral" et "bilan ganglionnaire", peuvent être réalisées dans n'importe quel ordre.

[3] Soit l'ensembles de tâches (A, B, C) La classe Et-strict(A,B,C) précise que l'ordre d'exécution de ces tâches est strictement A, B, C

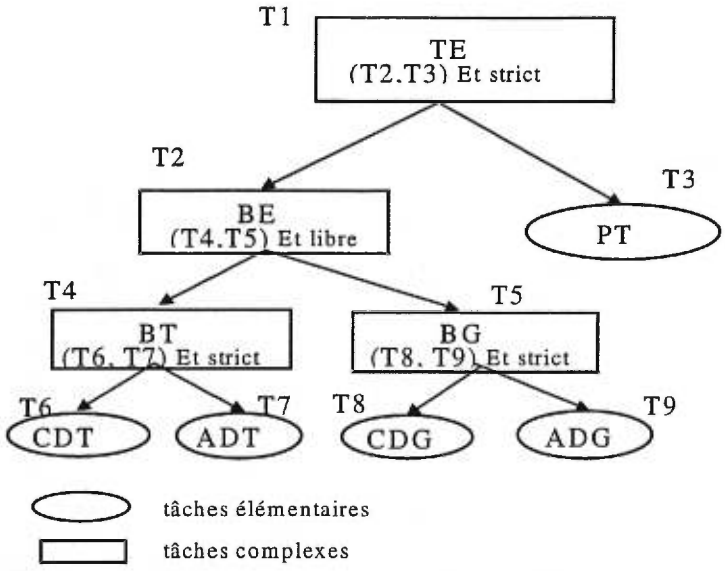


figure 6.3: Graphe des tâches pour le traitement de l'embolie pulmonaire (TE).

Actuellement le modèle opérationnel dispose de sept classes de comportements: (Et-libre, Et-strict, Ou-exclusif, Et-optionnel-strict, Et-optionnel-libre, N-sur-M-strict (N tâches sur M dans l'ordre indiqué), N-sur-M-libre (N tâches sur M dans n'importe quel ordre)). Les détails concernant les classes de comportement peuvent être trouvés dans [Pachet et al., 95].

Dans ce premier point nous avons détaillé la structuration du principal composant de l'architecture SAFARI: le curriculum. Comme nous venons de l'exposer, l'un des modèles généralement utilisé dans la représentation des connaissances du curriculum est le formalisme de graphe de tâches. C'est pourquoi, dans la suite de ce mémoire consacré globalement à l'acquisition des connaissances pour un STI, nous nous intéressons plus particulièrement à l'acquisition des connaissances de type tâche pour le curriculum.

6.1.2 De l'acquisition des connaissances du curriculum

6.1.2.1 De l'existence de plusieurs intervenants

Étant donné l'ambition d'obtenir l'ensemble des connaissances nécessaires à la réalisation d'un cours complet dans un domaine du monde réel (aéronautique, médecine,

etc.), plusieurs experts[4] doivent intervenir, parmi lesquels l'expert du domaine et le pédagogue. Nous critiquons ici l'une des faiblesses de plusieurs travaux dans le domaine des STI, qui considèrent qu'une seule source de connaissances, le pédagogue, pour créer la base de connaissances du système. Sans nier que l'expertise pédagogique puisse être suffisante dans certains cas, des connaissances décrivant le comportement d'un expert sont nécessaires (dans la plupart des applications). Prenons comme exemple le système GEOMUS présenté dans la section 5.2.2. Le système étant dédié aux élèves de quatrième, les concepteurs n'ont travaillé qu'avec des professeurs du premier et second degré. Pourtant, pour des problèmes plus complexes, l'observation de mathématiciens de haut niveau en train de résoudre des problèmes aurait pu s'imposer [Bazin, 93].

C'est pourquoi, nous insistons sur une distinction claire entre le rôle de l'expert du domaine et celui du pédagogue dans le cadre des STI. Dans un tutoriel intelligent, si l'expert du domaine est censé fournir les connaissances de contenu spécifiques à un domaine particulier (ou du moins une partie) et les valider en conformité avec les différents modèles (modèle de la tâche, modèle du domaine), le pédagogue doit enrichir la base de connaissances à l'aide d'une expertise permettant d'exploiter pédagogiquement ces connaissances en conformité avec les informations disponibles dans le modèle de l'apprenant. Il s'agit donc là d'une seconde validation: une validation pédagogique.

Compte tenu de ces idées il nous semble réaliste d'intégrer dans l'architecture du système plusieurs outils d'aide à l'explicitation de ces connaissances. La distinction entre ces outils doit se faire compte tenu de son utilisateur (l'expert du domaine ou l'enseignant), de la nature des connaissances à acquérir ainsi que des connaissances existantes déjà dans la base [Thomas & LeRoux, 96].

[4] Nous utilisons par la suite le terme expert pour désigner l'expert du domaine.

6.1.2.2 De l'insuffisance des éditeurs classiques

Comme dans la plupart des STI, l'explicitation des connaissances dans SAFARI est réalisée par l'intermédiaire d'éditeurs qui reflètent souvent la structure du modèle implémenté. Si cette démarche peut s'avérer profitable pour un utilisateur "expérimenté", qui connaît aussi bien le modèle de connaissances que l'outil informatique, il n'en demeure pas moins que dans le cas contraire ceci cause un problème de communication avec le système. Gaines explique d'ailleurs, qu'un des objectifs majeurs dans la conception des outils d'acquisition a été l'amélioration de l'environnement d'édition en vue d'offrir un cadre compréhensible par l'usager [Gaines, 93].

Ainsi, dans le cadre du projet SAFARI nous proposons un système d'explicitation des connaissances baptisé COSI (COgniticien SImulé) dont l'objectif est de capturer les connaissances décrivant la manière dont un expert définit un problème. Pour cela il repose sur un dialogue avec l'expert; une manière naturelle d'atteindre cet objectif est de l'aider à modéliser ses tâches.

Un système CREAM-TOOLS [Nkambou, 96] a été créé pour permettre à un concepteur d'enseignement de produire un curriculum selon l'approche CREAM. Ce système intègre plusieurs outils destinés à la construction d'un curriculum. La plupart sont des outils graphiques qui demandent une expérience minimale en informatique pour pouvoir être utilisés. Si l'expert n'est pas à l'aise avec la manipulation de ces éditeurs (l'éditeur de problèmes, par exemple), COSI sera très utile: il offre un cadre d'explicitation pour faciliter le travail de l'expert. Plus précisément:

- COSI dirige l'explicitation en simulant le dialogue avec l'expert qui n'est pas obligé de penser aux détails informatiques (exemples: quel objet de l'interface représente une tâche, comment spécifier les relations entre les différentes tâche, etc.) il doit que répondre aux questions,
- COSI essaye de déterminer certains besoins de l'expert. Des heuristiques ont été développées dans ce but (voir section 6.3.3)

En résumé, le but principal de COSI est de fournir à l'expert un cadre convivial d'explicitation de tâches. Un apport indirect est que la structure du graphe de tâches qui résultera de l'utilisation de COSI pourra constituer une aide précieuse pour un élève un situation d'apprentissage. Un graphe de tâches produit par l'équipe expert-COSI peut servir d'une part comme une source de savoir à consulter par l'élève (comme une explication ou comme un exemple de représentation d'une tâche) et, d'autre part, pour valider les réponses d'un apprenant en vérifiant la cohérence avec la solution de l'expert. En ce sens, le graphe de tâches résultant de l'utilisation de COSI peut être appréhendé comme un nouveau type de ressource didactique utilisable à des fins d'assistance.

Avant d'aborder plus en détail la conception de COSI, le point suivant résume la position de cet outil au sein du système SAFARI.

6.1.3 Position de COSI au sein de l'architecture SAFARI

COSI vise à faciliter l'explicitation des tâches auprès de l'expert. Replacé au sein de SAFARI, il permet donc la constitution d'une base de tâches organisée sous forme de graphes. Comme nous l'avons déjà signalé dans le paragraphe 6.1.1.3 de tels graphes de tâches peuvent être utiles pour modéliser la partie expertise opérationnelle.

Ainsi, au sein de SAFARI, le système d'acquisition dont l'architecture est présentée ci-dessous (figure 6.4) repose sur l'interaction entre l'expert et COSI.

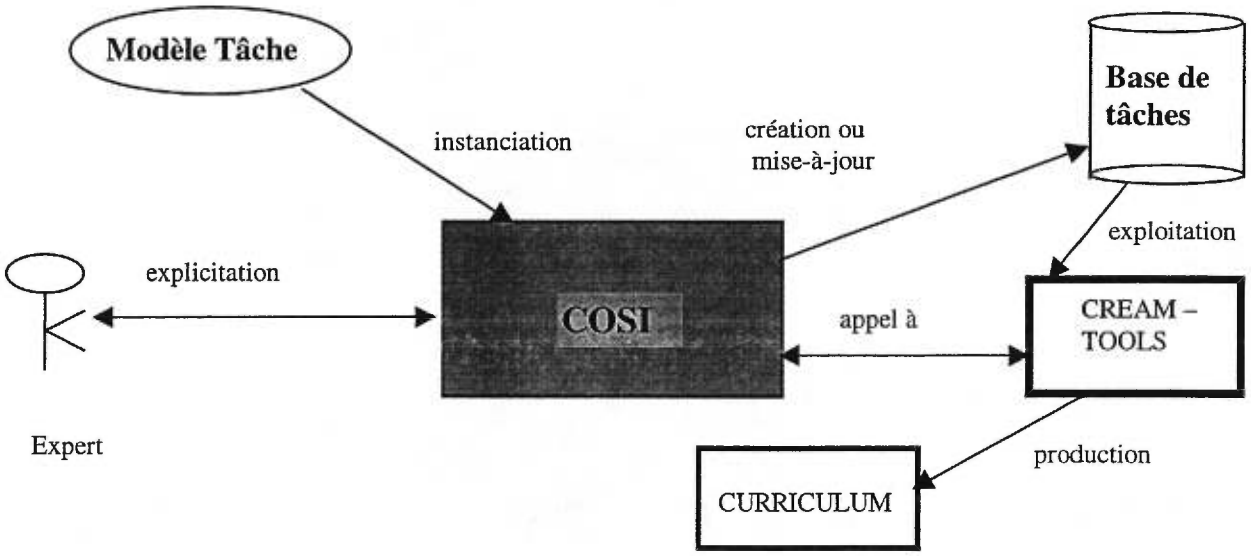


figure 6.4: Le système d'acquisition dans l'architecture SAFARI.

Ainsi, l'architecture du système d'acquisition comporte :

- *l'expert du domaine* qui joue un rôle fondamental lors de l'explicitation d'un modèle de tâches particulier. Il communique avec COSI par l'intermédiaire de l'interface.
- *le système COSI* qui va assister intelligemment l'expert au cours d'une résolution. Il va lui poser des questions pertinentes en vue de construire la base de tâches.
- *le modèle de la tâche* qui contient les connaissances relatives au modèle opérationnel décrit dans la section 6.1.1.3. Le graphe de tâches a été explicité selon une analyse cognitive de la tâche [Lesgold et Lajoie, 94]. Il est à noter que ce modèle peut contenir d'autres structures que le graphe de tâches défini dans la section 6.1.1.3. En effet, dans le projet SAFARI, il existe différents types de structures de tâches, notamment un graphe de tâches qui traite de l'aspect coopératif [Tadié, 96]. Avoir séparé explicitement le modèle de tâche de l'outil COSI, nous permet d'envisager de réutiliser cet outil pour d'autre types de structures de connaissances (par exemple d'autres graphes de tâches comme évoqué ci-dessus) en changeant simplement la spécification du modèle de tâches à utiliser.
- *la base de tâches* qui contient le résultat fourni par COSI. C'est-à-dire, elle comprend un ensemble de tâches structurées selon le modèle de tâches utilisé (par défaut, celui

de SAFARI décrit dans le paragraphe 6.1.1.3). Cette base de tâches est exploitée par CREAM-TOOLS pour produire un Curriculum.[5]

Abordons maintenant plus finement la conception de l'outil COSI.

6.2 COSI: principes de conception

6.2.1 Les apports de la psychologie cognitive et de l'ergonomie cognitive à la conception de l'outil COSI

6.2.1 Les apports de la psychologie cognitive

Plusieurs recherches en psychologie cognitive se sont concentrées sur une analyse de l'activité des experts pour mieux interpréter leur comportement.

Il y a deux aspects étudiés dans le cadre de cette discipline qui ont influencé la conception de l'outil COSI:

1. *L'analyse du problème.*

   La résolution de problèmes comprend deux étapes: l'analyse du problème à résoudre et la recherche de la solution. Selon Gagné [Gagné, 85], la phase d'analyse du problème est l'étape cruciale dans la résolution d'un problème puisque c'est lors de cette phase que l'expert "construit sa compréhension des données du problème". Il s'agit en fait de construire "une représentation du problème sans avoir immédiatement une solution" (une procédure) pour le résoudre [Hoc, 87]. Cette "représentation" est le résultat d'une analyse exhaustive du problème sans laquelle il est impossible de trouver la solution. L'une des faiblesses importantes dans les recherches concernant les systèmes à base de connaissances a été d'aborder la résolution de problèmes en insistant surtout sur l'aspect "comment trouver la solution". En revanche, peu d'intérêt a été accordé à l'étape d'analyse du problème. Un changement d'optique sur la notion de résolution de problèmes est proposé par Van de Velde qui considère que celle-ci consiste principalement à définir un modèle de ce problème [Van de Velde, 93].

[5] Dans le cadre de la conception d'un cours complet, CREAM-TOOLS , utilisé par le concepteur d'enseignement, peut faire appel à COSI pour intervenir dans la construction d'une nouvelle tâche.

2. *Le modèle de l'expert.*

Certains experts ne savent pas expliquer d'ou l'intérêt de détecter les difficultés et la pertinence des connaissances. Il est important de construire un modèle de l'expert puisqu'il permet d'adapter le système au profil de l'expert. Plusieurs modèles de l'expert ont été développés selon différents points de vue (social, comportemental, de résolution de problème). Pour nous, un tel modèle doit identifier les caractéristiques de l'expert qui lui permettent de résoudre un problème. Pour cela, plusieurs travaux de psychologie cognitive ont comparé le comportement des experts et des novices et ont démontré que la différence ne réside pas seulement dans la "quantité" de connaissances des experts, mais essentiellement dans leur manière de traiter le problème. Ainsi, l'expert cherche les caractéristiques d'un problème similaire pour lequel il dispose d'une représentation adéquate dans sa mémoire [Dieng, 93].

Les stratégies de résolution constituent le noyau du modèle de l'expert.

Nous précisons ici ce que doit contenir un modèle de l'expert. Puisque nous nous intéressons particulièrement aux préférences d'un expert pour analyser un problème, un tel modèle doit décrire la structure des différentes stratégies qu'il utilise lors de ce processus. Selon Tardif, il existe deux types de stratégies permettant d'analyser un problème [Tardif, 92]:

- suivant une *stratégie descendante*, les experts décomposent le problème en sous-problèmes moins complexes en adoptant une méthode de type "diviser pour régner";
- selon une *stratégie ascendante*, les experts ont aussi tendance à regrouper des problèmes dans une ou plusieurs catégories.

6.2.1.2 Les apports de l'ergonomie cognitive

L'ergonomie cognitive étudie principalement les moyens permettant d'adapter un système à son utilisateur en vue de lui permettre d'atteindre ses objectifs avec le moins d'effort possible [Dieng, 93]. Puisque l'outil COSI est dédié à un expert (un non-informaticien dans la plupart des cas), il est nécessaire de considérer plusieurs aspects:

- la conception d'un outil d'acquisition doit offrir à l'usager un langage de

communication compréhensible. Cet aspect doit se refléter en construisant des interfaces adaptées à l'utilisateur;

- le système doit disposer d'un module d'aide sur le fonctionnement du système;
- étant donné que généralement les outils d'acquisition dirigent l'activité de l'utilisateur (c'est le système qui demande des informations), ce dernier doit avoir des possibilités de suspension, d'abandon ou de changement d'activité ou cours d'une session de consultation [Ermine, 93].

6.2.2 COSI –Notre approche d'acquisition

Nous adoptons un principe général d'acquisition des connaissances par *raffinement successifs* car dans la communauté de recherche en acquisition des connaissances, cette manière d'aborder la construction d'un système à base de connaissances est de plus en plus utilisée.

Dans cette optique, nous proposons un processus d'acquisition qui s'appuie sur un modèle de tâche et l'instancie à la suite d'un dialogue avec l'expert (figure 6.5).

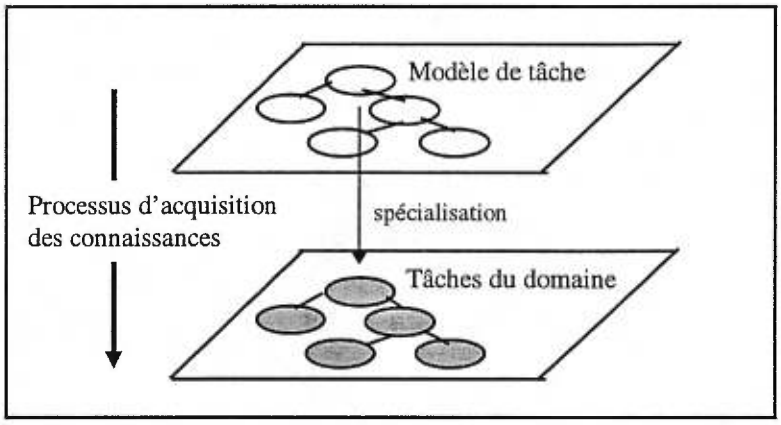


figure 6.5: L'approche d'acquisition de COSI.

Explicitation d'une base initiale de connaissances

Notre objectif de départ a été la conception d'un outil qui permette d'expliciter un noyau initial de connaissances pour décrire les principales étapes de l'activité de l'expert en vue de résoudre un problème. L'outil COSI commence son travail à partir d'une base de connaissances vide. Ainsi, notre travail couvre une première phase d'acquisition des

connaissances qui se focalise strictement sur les connaissances de type tâche.

Utilité de l'approche

La base initiale de connaissances que nous construirons ouvre des perspectives intéressantes pour une nouvelle étape d'acquisition consistant à expliciter les connaissances du domaine (les concepts et les relations entre ces concepts pour un domaine particulier) impliquées dans la réalisation des tâches ainsi définies. Cette nouvelle phase est de cette manière facilitée d'avantage, car le cogniticien (humain ou simulé) dispose déjà d'un ensemble de connaissances permettant de contraindre son dialogue avec l'expert (autrement souvent trop flou). Supposons qu'une des tâches de l'expert soit une tâche de diagnostic. Plutôt que de demander à l'expert "quels sont les objets dans ce domaine ?", le cogniticien focalisera son discours en vue d'obtenir des connaissances jouant un rôle précis dans la réalisation de la tâche, comme par exemple, "quels sont les connaissances ayant le rôle de symptôme ?". Ceci implique tout d'abord la définition d'un modèle du domaine adéquat pour un problème de diagnostic.

Notre démarche utilise une approche d'acquisition orientée tâches qui s'inscrit dans le contexte plus général de l'acquisition basée sur un modèle (ici, le modèle de tâche).

Notre objectif principal est d'offrir à l'expert un outil qui guide l'explicitation des connaissances, en tenant compte d'une part du modèle de l'expertise et d'autre part des caractéristiques de l'expert qui le manipule. Ainsi, l'outil COSI permet l'explicitation de tâches de l'expert en s'appuyant sur la coopération entre ce dernier et le système.

L'outil que nous proposons a pour caractéristique principale la généricité, car l'outil COSI n'est pas dédié à un type particulier de problème, ni à un domaine spécifique.

6.2.3 Principes de conception de COSI : résumé

Lors de la conception de l'outil COSI nous avons visé plusieurs aspects.

- COSI est un outil interactif d'acquisition des connaissances dédié à un expert. Un des

ses objectifs est de réduire le temps d'acquisition, en automatisant dans une certaine mesure la tâche du cogniticien.

- Nous utilisons un modèle de tâches générique qui sert de guide lors du processus d'explicitation des connaissances. Il permet à l'expert d'être concis, car il ne doit expliciter que les connaissances de type tâche.
- Le rôle de COSI est d'expliciter le noyau initial de connaissances. La base de tâches résultante représentera le support pour acquérir d'autres types de connaissances (connaissances du domaine, connaissances d'explication, etc.).
- COSI n'a pas été conçu pour donner la solution d'un problème mais pour analyser le problème en vue d'en obtenir une représentation explicite en termes de tâches.
- Il nous semble important d'offrir la possibilité d'expliciter une tâche de différentes manières, selon plusieurs stratégies. De cette façon, l'expert peut choisir sa stratégie préférée. On peut aussi déterminer la stratégie optimale pour une tâche donnée.
- Pour renforcer la coopération expert-COSI nous avons créé un modèle de l'expert. Ainsi, COSI doit guider le dialogue compte tenu des caractéristiques et des préférences de l'expert en lui offrant un environnement de travail convivial.

6.3 Conception de COSI

6.3.1 La nature des connaissances à expliciter et la méthode de résolution

De manière générale, l'explicitation des connaissances vise deux aspects importants:

- tout d'abord, l'identification des types de connaissances à expliciter;
- ensuite, la définition de la méthode utilisée pour mettre en œuvre ce processus, compte tenu de la nature des connaissances impliquées.

Le système COSI est spécialisé pour un type de connaissances (connaissances de type tâche) et utilise une méthode de résolution bâtie sur le principe de la planification hiérarchique.

La *planification hiérarchique* a été utilisée dans plusieurs travaux [Willamowski, 94], [Drummond & Tate, 90], [Stefik, 81]. Cette approche repose sur une *analyse de tâches* en vue d'obtenir une structure de tâches sur plusieurs niveaux d'abstraction, du plus général (la racine de l'arbre) au plus détaillé (le niveau feuilles). De cette manière, un sujet élabore un plan d'action en décrivant progressivement une activité par l'ensemble des sous-tâches qui la composent jusqu'à atteindre le niveau de détail désiré. La popularité de la méthode réside surtout dans le fait qu'elle permet de structurer une activité sous forme d'une hiérarchie et, ainsi, mieux la contrôler. Plus précisément, cette structure de tâches facilite la "compréhension" (description) d'un problème grâce à deux types de liens:

- *le lien de composition* entre une tâche et une de ses sous-tâches. Ainsi, en parcourant la hiérarchie de tâches du haut en bas, il est possible d'identifier les sous tâches à faire pour une certaine tâche;
- *le lien d'ordonnancement* définissant une relation d'ordre (partiel ou total) entre les sous-tâches d'une certaine tâche de façon à accomplir cette dernière.

Pour expliciter un problème particulier en respectant le principe de la planification hiérarchique nous avons développé trois stratégies d'explicitation. Ces stratégies ("descendante", "ascendante" et "mixte") sont décrites en détail dans la prochaine section.

6.3.2 Les stratégies d'explicitation

En acquisition des connaissances, on a souvent utilisé la notion de *stratégie* et plusieurs définitions parfois contradictoires lui ont été données.

Pour nous, une *stratégie* est une méthode permettant au système de générer une structure hiérarchique de tâche par l'intermédiaire d'un dialogue humain-machine. Autrement dit, le rôle d'une stratégie est d'instantier un modèle de tâche en coopérant avec un expert du domaine. Ceci nécessite qu'une stratégie soit définie au "niveau

connaissances" (le "knowledge level" présenté dans la section 3.3) uniquement par des interfaces demandant en langage naturel les informations dont on a besoin pour l'application de la méthode (la stratégie implémentée).

Une stratégie, telle que nous l'avons définie, possède plusieurs caractéristiques:

- une stratégie est une *méta-connaissance* (ou *méta-opération*) qui fonctionne comme un "générateur de plans de résolution" comme dans les systèmes conseillers proposés dans [Mengelle, 95];
- une *généricité* vis-à-vis du domaine de l'expert. Nous avons conçu des stratégies générales indépendantes des connaissances du domaine;
- une *nature heuristique* car une stratégie vise à faciliter le processus d'explicitation des connaissances de type tâche sans pour autant garantir le succès;
- une "*faisabilité cognitive*" car nous avons conçu une stratégie à l'image des "principes" d'analyse et de résolution de problèmes issus de la psychologie cognitive. En effet, comme nous l'avons indiqué dans la section 6.2.1, plusieurs chercheurs ont souligné que, même si la manière de traiter un problème varie beaucoup d'un sujet à un autre, les experts utilisent des stratégies générales ("ascendante", "descendante").

Nous abordons la notion de stratégie dans le contexte constitué de l'expert et de la tâche à résoudre. Nous considérons, en effet, les trois éléments (l'expert, la tâche et la stratégie d'explicitation) comme étant interdépendants lors du processus de résolution d'un problème (figure 6.6).

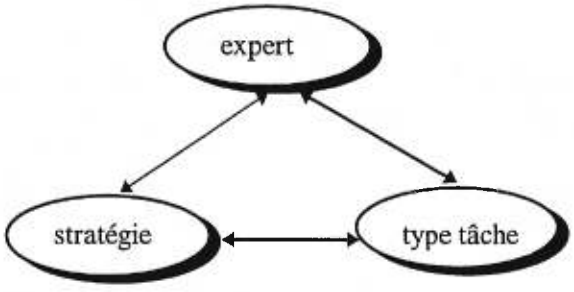


figure 6.6: Les liens expert-stratégie-tâche.

Plus précisément, sans exclure la possibilité qu'une tâche puisse être explicitée selon plusieurs stratégies, nous pensons qu'il existe une dépendance forte entre le type du problème et la stratégie appropriée à son explicitation. Comme souligné d'ailleurs par plusieurs recherches ([Breuker, 94], [Aussenac, 89]) l'intérêt de définir des types de problèmes réside dans le fait que cela nous permet "d'orienter" le processus d'explicitation (choisir la stratégie appropriée pour un type de problème particulier). D'autre part, un expert peut préférer une stratégie à une autre tout simplement en raison de ses habitudes de travail. COSI laisse à l'expert la possibilité de choisir sa stratégie préférée. Dans le cas d'une non-concordance entre le choix de l'expert et la stratégie que le système aurait choisi (compte tenu des heuristiques implémentées dans COSI qui associent à chaque type de tâches la stratégie appropriée), COSI donne des recommandations à l'expert, mais, c'est toujours l'expert qui prend la décision finale.

Nous avons prévu trois stratégies d'explicitation pour permettre aux experts de s'exprimer de différentes manières.

6.3.2.1 La stratégie descendante

La stratégie descendante permet la décomposition progressive de la tâche en sous- tâches de moins en moins complexes. Cette stratégie est surtout adaptée aux problèmes de type "analyse" (diagnostic, classification, etc.). Par exemple, pour changer une roue, il convient tout d'abord de lever la voiture, ensuite de retirer l'ancienne roue, de la remplacer par la nouvelle et, pour finir, de ne pas oublier de baisser la voiture.

La stratégie "descendante" que nous avons développée permet la décomposition progressive de la tâche selon deux politiques.

- La première politique permet l'explicitation d'une tâche en "largeur d'abord". En conséquence, le graphe est construit niveau par niveau, en partant de la racine jusqu'aux feuilles. Cette politique d'analyse d'une tâche impose la contrainte suivante: l'expert doit d'abord identifier toutes les sous-tâches d'une tâche donnée avant de passer à la décomposition de chacune de ses sous-tâches (les sous-tâches sont décomposées suivant une ordre "FIFO", c'est-à-dire la première sous-tâche saisie sera

la première décomposée). La politique "en largeur d'abord" permet une explicitation rapide de la tâche si les connaissances de l'expert sont bien structurées. En effet, cette politique est un "raccourci" pour un expert avec une vision bien structurée de la tâche. Par exemple, cette politique peut s'avérer utile pour planifier la rédaction d'un rapport (article, thèse, etc.): d'abord nous définissons les chapitres principaux, ensuite, pour chaque chapitre nous identifions les sections, etc. En revanche, cette stratégie peut contraindre l'expert surtout pour des problèmes complexes.

- Plutôt que d'implementer la politique en "profondeur d'abord", nous avons préféré une politique plus générale qui combine une politique d'explicitation "en largeur" avec une politique "en profondeur". Cette stratégie permet de définir une tâche à la fois. À chaque étape de l'explicitation, l'expert a deux possibilités:

  - décomposer la tâche courante,
  - ou saisir une nouvelle tâche.

La stratégie "descendante" que nous avons implémenté se base sur l'algorithme suivant:

```
Début { stratégie "descendante"}
Initialiser la mémoire de travail;
T ←Identifier la tâche à faire ;
RacineGraphe ← T;
TraitementTopDown(T);
Fin; { stratégie "descendante" }

Procédure TraitementTopDown(T);
Si choixAnalyse = 'largeurDAbord'          {l'expert a choisi de décrire toutes les sous-tâches
                                            d'un niveau }
                ExpliciterTâcheLargeurDAbord (T) ;
Si choixAnalyse = 'largeurOuProfondeur'    {l'expert a choisi de décrire une
                                            sous-tâche à  la fois }
                ExpliciterTâcheLargeurOuProfondeur (T );

Fin; { TraitementTopDown }

Procédure ExpliciterTâcheLargeurDAbord ( uneTâche: tâche )
{ pour la première politique descendante avec uneTâche comme paramètre }
TraiterToutesSousTâches (uneTâche);
Afficher la listeSous-tâchesSaisies pour la tâche uneTâche;
Définir la dynamique entre les sous-tâches de la tâche uneTâche;
Fusionner la listeTâchesATraiter avec la listeSous-tâchesSaisies;
Pour la liste TâchesATraiter do
                NoeudCourant←le premier élément de la liste TâchesATraiter
                TraiterToutesSousTâches (NoeudCourant);
```

```
                Effacer le NoeudCourant de la liste TâchesATraiter

Fin; { ExpliciterTâcheLargeurDAbord }

Procédure TraiterToutesSousTâches ( Parent: tâche )
{ permet d'identifier la structure de la tâche uneTâche ainsi que sa dynamique}
CreerNoeud pour la super-tâche
Afficher le nom de la super-tâche dans "Historique des tâches";
Si typeTâche pour la super-tâche = 'decomposable'
        IdentificationSousTâches pour la super-tâche;
        Tant qu'il existe des sous-tâches à saisir
                N←CreerNoeud pour chaque sous-tâche saisie;
                Ajouter arc du nœud Parent à N;
                Afficher le nom sous-tâche saisie dans "Historique des tâches";
        Fin Tant que;
Fin; { TraiterToutesSousTâches}

Procédure ExpliciterTâcheLargeurOuProfondeur( uneTâche: tâche )
{pour la deuxième politique descendante}
Parent ← uneTâche
Tant qu'il existe des sous-tâches à saisir pour la tâche uneTâche
        ST←IdentificationUneSousTâche pour la tâche uneTâche;
        Afficher le nom sous-tâche saisie dans "Historique des tâches";
        CreerNoeud (ST);
        NoeudCourant ← ST
        Ajouter arc du noeud Parent à NoeudCourant;
        TraiterUneSousTâche (NoeudCourant, Parent );
Si liste TâchesADecomposer non-vide
        Pour la liste TâchesADecomposer do
                NoeudCourant ← le premier élément de la liste TâchesADecomposer
                TraiterToutesUneSousTâche (NoeudCourant);
                Effacer le NoeudCourant de la liste TâchesADecomposer
Fin; { Expliciter TâcheLargeurOuProfondeur }

Procédure TraiterUneSousTâche (uneTâche : tâche, Parent : tâche )
Si typeTâche pour la super-tâche = 'decomposable'
        Si choixAnalyseTopDown = 'decomposer'
                STT ← IdentificationUneSousTâche pour la tâche uneTâche;
                Parent ← uneTâche ;
Si choixAnalyseTopDown = 'saisir'
        STT ← IdentificationUneSousTâche pour la super-tâche ;
        Ajouter Parent dans la liste TâchesADecomposer ;
NoeudCourant ← STT    ;
Ajouter arc du noeud Parent à NoeudCourant;
Afficher le nom sous-tâche saisie dans "Historique des tâches";
Si choixExpliciter = 'uneSousTâche'
        TraiterUneSousTâche (NoeudCourant);
Si choixExpliciter = 'toutesSousTâches'
        TraiterToutesSousTâches (NoeudCourant);

Fin; { TraiterToutesUneSousTâche}
```

Comparaison entre les deux politiques de la stratégie descendante

Du point de vue de l'expert, la deuxième stratégie, qui regroupe les deux politiques (en "largeur" et en "profondeur"), est plus flexible que la première. L'avantage qu'on peut en tirer est de permettre à l'expert de choisir au niveau de chaque tâche (dans le graphe) dans quelle "direction" continuer l'analyse de la tâche en cause. En effet, l'expert décompose les tâches quand il le souhaite, puisque l'ordre de décomposition de tâches n'est pas imposé. Toutefois, cette nouvelle stratégie est moins rapide que la première à cause de plusieurs "retours arrière" (pour revenir sur les tâches qui ont déjà décrites mais qui n'ont pas été décomposées). Elle demande plus de mémoire puisqu'on doit conserver une liste des tâches à décomposer (tâches définies par l'expert sans préciser leurs sous-tâches).

6.3.2.2 La stratégie ascendante

La stratégie ascendante quant à elle, a pour but d'expliciter la tâche en allant des tâches élémentaires vers les tâches plus abstraites. Cette stratégie est surtout adaptée aux problèmes de type "synthèse" (conception, abstraction, etc.). La stratégie ascendante implique souvent un travail de *création* d'expertise. Si un expert a choisi de définir un problème de manière ascendante, il est pertinent de penser qu'il est capable d'énumérer ses connaissances comme une suite d'actions, ce qui correspond à une vision plutôt linéaire que hiérarchique de la tâche. Un exemple type de cette stratégie est la construction d'un puzzle: la personne va commencer par assembler des petits morceaux entre eux, puis constituer peu à peu des zones plus importantes.

Puisque la représentation de la tâche que nous avons choisie impose la définition du problème sur plusieurs niveaux d'abstraction le rôle de COSI ne se résume pas à extraire les connaissances de l'expert mais aussi à l'aider à organiser hiérarchiquement ses connaissances (définir progressivement des sous-objectifs pour réaliser l'objectif global de la tâche). La stratégie "bottom-up" est donc une construction progressive d'une tâche; elle permet à l'expert de synthétiser son activité au sein d'une hiérarchie de tâches.

Afin de satisfaire les aspects mentionnés, une fois la saisie de toutes les sous-tâches qui composent la tâche principale terminée, COSI vérifie que le graphe

représentant la tâche soit un graphe connecté (que tout nœud soit lié à au moins un nœud Parent). S'il existe des tâches non-connectées au graphe, COSI propose deux modalités:

1. La liste de toutes les tâches saisies par l'expert (à l'exception de tâches élémentaires) est affichée. L'expert peut sélectionner un (ou plusieurs) élément(s) de la liste comme super-tâche (ceci inclut la possibilité de relier la tâche à la tâche principale représentant la racine du graphe).
2. L'expert peut définir une nouvelle tâche comme super-tâche pour la tâche considérée.

En tenant compte de ces idées nous avons implémenté la stratégie "ascendante" selon l'algorithme suivant :

```
Début { stratégie ascendante }
Identifier la tâche à faire;
Tant qu'il existe actions à saisir,
        Initialiser la mémoire de travail;
        Demander à l'expert de définir une à une les actions composant la tâche;
        Action ← IdentificationAction
        TraiterUneAction(Action)
Si il existe des noeuds sans Parent, alors
        Demander à l'expert de trouver un(des) Parent(s) pour ces nœuds ;
Pour liste NoeudsSansParent do
        TâcheSansParents ← le premier élément de la liste NoeudsSansParent
        TraiterTâchesSansParents(TâcheSansParents);
        Effacer TâcheSansParents de la liste NoeudsSansParent;
Fin { stratégie ascendante }

Procédure TraiterUneAction(Action : tâche)
Si l'action saisie est une action élémentaire,
        Créer noeud;
        Afficher le nom de l'action dans "Historique";
        Ajouter le noeud dans la liste de noeudsATtraiter;
Sinon
        Afficher la liste des actions saisies;
        Séléctionner les actions composant l'action concernée;
        Créer ListeActionsSélectionnées;
        Si (liste des noeuds à traiter non vide) ou (ListeActionsSélectionnées non vide),
                Créer l'instance du Modèle de la Tâche;
                Créer noeud Parent;
                Ajouter le noeud Parent dans liste des noeuds à traiter;
                Ajouter les arcs entre le noeud Parent et les noeuds fils;
                Définir la dynamique entre les noeuds fils;
                Effacer ListeActionsSéléctionnées de la liste des noeuds à traiter;
        Sinon
                Demander à l'expert de saisir les actions élémentaires puis celles composées;
Fin {TraiterUneAction};

Procédure TraiterTâchesSansParents ( uneTâche: tâche )
```

```
{ permet d'identifier le Parent de la tâche uneTâche }
Afficher la liste ParentsPossibles ; {cette liste contient les tâches décomposables};

Si choixExpert = 'selectionner un (des) Parent(s) dans la liste ParentsPossibles'
                Parents ← les éléments sélectionnes dans la liste ParentsPossibles;
Si choixExpert ='définir un nouveau Parent'
        IdentificationTâche pour un nouveau Parent;
        Ajouter Parent dans liste ParentsPossibles ;
TraiterTâchesSansParents (uneTâche);
Si choixExpert ='relier à la racine'
        Ajouter racineGraphe dans liste Parents;
Pour Parents do
                Ajouter arc du chaque élément de la liste Parents  à uneTâche;
Fin; { TraiterTâchesSansParents}
```

Difficultés et limites

- L'analyse ascendante d'une tâche est moins structurée que l'analyse descendante; en conséquence l'implémentation de cette stratégie est plus complexe.
- Un inconvénient de l'analyse ascendante est qu'elle peut s'avérer longue et coûteuse si l'expert n'a pas une vision très nette sur la possibilité de réutiliser des tâches existantes dans la compositions de nouvelles tâches. Il existe en ce sens le risque de définir plus de tâches que nécessaire, car dans la pratique l'expert a acquis des automatismes difficiles à synthétiser.

6.3.2.3 La stratégie mixte

La stratégie mixte combine les deux stratégies citées ci-dessus. C'est ce type de démarche qui est utilisé, pour ne citer qu'un exemple, lors de la démonstration d'un théorème: le mathématicien se fixe des objectifs à démontrer, puis il combine les axiomes jusqu'à démontrer un à un chacun des objectifs ainsi énoncés.

Le but est ici de permettre à l'expert de mélanger les deux stratégies définies et de réaliser une partie de son travail (la construction d'un graphe de tâches) selon une stratégie ascendante et une autre partie selon une stratégie descendante. Supposons que l'expert tente de décrire la démonstration d'un théorème. Il peut commencer par "décomposer" en sous-buts le théorème à démontrer, puis tenter de démontrer chacun des sous-buts à partir des axiomes. Le principal avantage de la stratégie mixte est de permettre à l'expert de se déplacer vers le "haut" (un niveau plus abstrait) ou vers le

"bas" du graphe à n'importe quel moment de la définition d'une tâche. Cette stratégie impose donc moins de contraintes à l'expert.

La stratégie "mixte" s'appuie sur l'algorithme suivant:

```
Début { stratégie "mixte"}
Initialiser la mémoire de travail ;
T ← Identifier la tâche à faire ;
RacineGraphe ←T ;
TâcheCourante ←T ;
Tant qu'il y a des sous-tâches à saisir
                Tant que choixAnalyseMixte= 'descendante'
                        TraitementTopDown(TâcheCourante);
                Tant que choixAnalyseMixte= 'ascendante'
                        TraiterUneAction (TâcheCourante);
Fin; { stratégie "mixte" }
```

6.3.3 Heuristiques

Selon Causse, une *heuristique* est une connaissance incertaine qui apporte des éléments à la solution recherchée [Causse, 94].

Pour nous, ce terme désigne une connaissance de COSI qui a pour rôle d'orienter (ou réorienter) le processus d'explicitation de tâches si cela est nécessaire. Plus précisément, les heuristiques visent les changements "tactiques" dans une stratégie.

Par exemple, nous avons deux heuristiques pour guider le choix à réaliser entre les différentes stratégies implémentées:

- La première associe à des types de problèmes les stratégies appropriées, par exemple:

  si type problème = 'diagnostic'

  alors stratégie = 'descendante'

- La deuxième vérifie si la stratégie choisie est appropriée pour l'expert considéré, par exemple:

  si (compteurAide[6] *>= nombreInterfacesUtilisées)*

  ou (compteurRetourArrière>=nombreInterfacesUtilisées)

  alors

[6] La variable compteurAide compte le nombre d'utilisations de l'aide disponible sur les interfaces. La variable compteurRetourArrière compte les retours aux étapes précédentes. Si l'expert n'est pas à l'aise avec une stratégie, on peut penser qu'il aura besoin de revoir les informations saisies antérieurement.

demander à l'expert s'il veut changer la stratégie choisie

6.4 COSI: Architecture détaillée et exemple de dialogue

6.4.1 Architecture de COSI

Pour simuler les aptitudes recherchées chez un cogniticien humain, COSI doit avoir à sa disposition des connaissances sur le modèle qui le guide (le graphe de tâches) et poser les bonnes questions dans le bon ordre pour que l'expert lui fasse confiance.

Pour satisfaire toutes ces contraintes, COSI dispose dans son architecture d'un modèle de l'expert, d'une base de connaissances, d'une mémoire de travail, d'un module de raisonnement et d'un gestionnaire de dialogue (figure 6.7).

Le modèle de l'expert

Contrairement au modèle de la tâche, le modèle de l'expert n'est pas facilement formalisable (à cause de la diversité des expert qui peut utilisé le système). Il sera construit au fur et mesure du processus d'explicitation de la tâche. Le modèle de l'expert est une notion nouvelle dans la communauté STI; nous l'avons introduite dans nos travaux à l'image du modèle étudiant qui lui est bien connu dans la littérature.

Définition: Le modèle de l'expert est l'*ensemble des connaissances* décrivant certaines caractéristiques de l'activité de l'expert. Ce modèle peut contenir trois types de connaissances :

- *Connaissances du domaine* étudié (termes de base du vocabulaire de l'expert) Exemple : Si l'expert est un médecin il utilise souvent des termes comme : maladies, bilans, diagnostic, etc.
- *Connaissances relatives à l'expérience de l'expert en explicitation des connaissances.* Celles-ci constituent un indicateur de l'expert quant à l'utilisation d'outils.
- *La stratégie d'explicitation préférée* pour décrire une tâche.

Nous disposions de quelques heuristiques permettant une adaptation de l'outil COSI à l'expert l'utilisant; celles-ci nous ont dont orienté quant au choix des informations à stocker dans le modèle de l'expert. Ces heuristiques se répartissent en trois catégories :

- *heuristiques générales* qui ne font pas référence ni à un domaine particulier ni à l'expérience de l'expert .

  Exemple : Si type problème = 'synthèse' alors

  Stratégie = ascendante

- *heuristiques concernant l'expérience de l'expert en utilisation d'outils d'acquisition*

  Exemple : Si Niveau_experience_ac ='faible' alors

  Conseiller à l'expert une stratégie à utiliser

  Si Niveau_experience_ac ='bon' alors

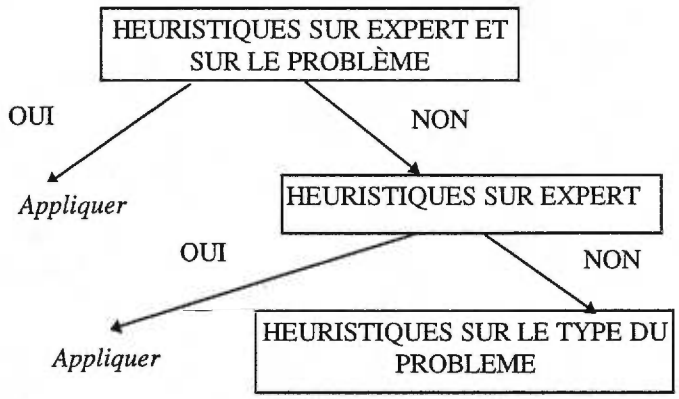
  Demander à l'expert quelle stratégie il préfère

- *heuristiques qui combinent des connaissances générales et des connaissances relatives à l'expérience de l'expert*

  Si type problème = 'diagnostic' et Niveau_experience_ac ='moyen'

  Alors Conseiller l'expert à choisir la stratégie descendante Largeur et profondeur

La séquence de déclenchement des heuristiques est la suivante:



Plus précisément, COSI vérifie s'il peut appliquer les heuristiques sur l'expert et sur le problème à analyser. Si ce n'est pas le cas, il cherche à appliquer les heuristiques sur

l'expert ou celles concernant le type du problème.

La base de connaissances contient :

- les connaissances sur la structure de la tâche à construire et le comportement de ses sous-tâches (le modèle de tâches);
- les connaissances relatives à l'expert (le modèle de l'expert).

Ces différentes connaissances de COSI lui permettent :

- de mémoriser les termes de base qu'utilise l'expert ainsi que sa stratégie préférée pour définir la tâche. Ceux-ci sont acquis par un questionnaire préliminaire de l'expert (figure 7.4);
- de guider le dialogue suivant le modèle de la tâche à construire.

La mémoire de travail contient toutes les variables manipulées lors de la résolution d'une tâche en vue de garder la trace de son explicitation.

Le module de raisonnement

Bien que notre recherche ne relève pas de la problématique des agents logiciels stricto sensu, COSI s'en rapproche sur trois points communs.

- *L'aspect réactif*: il permet à COSI de réagir de manière appropriée aux interventions de l'expert. L'expert a la possibilité d'interrompre COSI pour, par exemple, changer de stratégie.
- *L'aspect contrôle*: il permet à COSI de planifier son travail et de contrôler ses actions. Le rôle des heuristiques décrites dans 6.3.3. est d'adapter le comportement de COSI à l'expert. Grâce à ses heuristiques, COSI cherche à tenir compte de la stratégie préférée de l'expert (s'il existe une) ou de lui suggérer une compte tenu du type de problème lorsque nécessaire.
- *L'aspect social*: il permet à COSI de communiquer avec un agent humain qui est dans notre cas un expert du domaine. Cet aspect n'est pas très développé dans COSI ,mais

celui-ci devrait faire l'objet d'une attention toute particulière dès lors que nous déciderons de s'adresser à une société d'experts (chacun détenant une partie de l'expertise globale).

Les aspects réactif et social que nous évoquons sont similaires à ceux définis par [Tecuci, 96]. L'aspect contrôle respecte le caractère "autonome" défini par le même auteur. De plus, il exploite des heuristiques lui permettant de conseiller l'expert. Par exemple, l'heuristique "Si la tâche est de type diagnostic alors une stratégie descendante est la plus appropriée pour expliciter la tâche en question" peut aider l'expert à choisir la stratégie de dialogue.

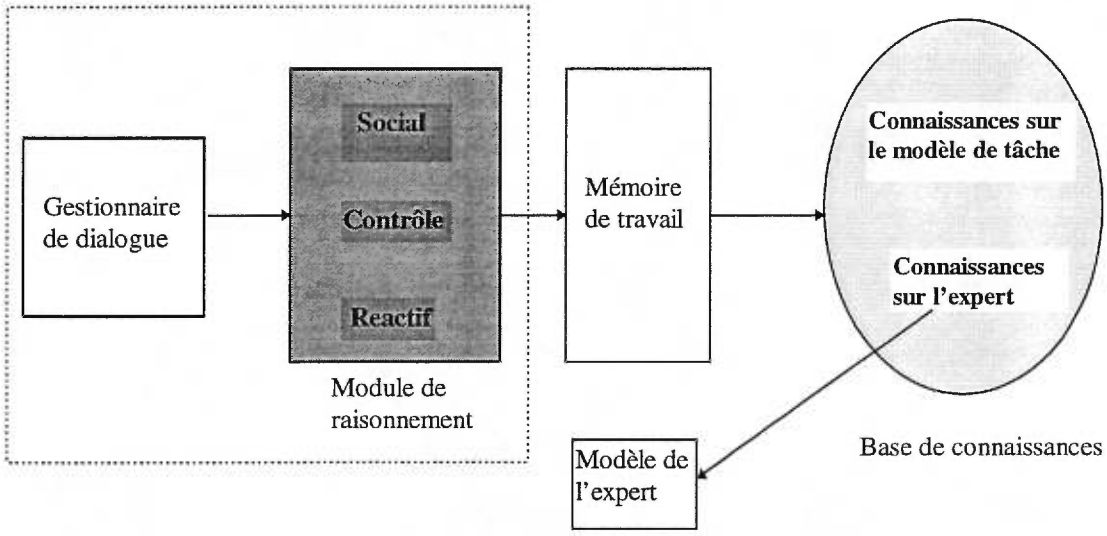


figure 6.7: Architecture de COSI.

L'ensemble des connaissances décrit ci-dessus permet à COSI de se comporter comme un agent dans l'acception générale que confère [Wooldridge & Jeannings, 95] à ce terme: *personne ou système informatique qui participe au processus de résolution de problèmes.*

Le gestionnaire de dialogue (figure 6.7) permet à COSI de structurer, d'enrichir et de rendre intelligent son dialogue avec l'expert. Cette structuration exploite les trois caractéristiques décrites plus haut. Pour cela, il manipule les stratégies et les heuristiques d'acquisition définies dans les sections 6.3.2 et 6.3.3. À l'issue du dialogue, le module

génère un graphe de tâche.

6.4.2 Structure du dialogue

Le dialogue est l'élément moteur dans le système que nous construisons. Dans la pratique, le dialogue humain-machine a été souvent confondu avec un monologue; dans la plupart des cas c'est le système qui "parle": il pose les questions et ne laisse à son interlocuteur que la possibilité de sélectionner la réponse parmi celles qu'il propose.

Puisque les deux partenaires possèdent des connaissances de nature différente (les connaissances de l'expert concernent un domaine particulier alors que celles du système sont constituées de stratégies d'explicitation générales), le langage utilisé par le système s'avère souvent inadapté car ne faisant pas usage des termes du domaine de l'utilisateur.

Sans prétendre que l'outil COSI soit capable de communiquer à la manière d'un humain on a tâché de prendre en compte certains aspects importants:

- On considère important d'associer à la tâche plusieurs stratégies de dialogue permettant à un expert d'expliciter une tâche en fonction de ses préférences ou de ses habitudes. COSI permet à l'expert de définir une tâche selon trois stratégies décrites dans la section 6.3.2.
- Nous avons introduit la notion de "synonyme" d'une tâche pour faciliter le travail de l'expert. Puisque deux experts dans le même domaine peuvent appeler la même tâche sous des noms différents, nous considérons utile d'informer l'expert qu'une tâche d'un nom synonyme a été déjà construite par un autre expert. De plus, attacher à une tâche plusieurs synonymes peut constituer un bon point de départ dans la réutilisation des connaissances existantes dans la base de COSI; l'expert pourra expliciter une tâche à partir des tâches existantes dans la base de COSI.
- Le dialogue humain - COSI doit être conçu de manière à ce que l'expert puisse suggérer une direction à prendre :
  1. l'expert peut interrompre le processus d'analyse quand il le souhaite,
  2. il a la possibilité de remplacer une stratégie d'analyse par une autre. Dans ce cas, il peut garder la partie de la tâche déjà définie, recommencer à définir la même tâche

ou définir une autre tâche.

- Suivant les caractéristiques de la tâche et du modèle de l'expert, COSI pourra anticiper la meilleure stratégie de dialogue compte tenu de la situation.

Afin de respecter les aspects mentionnés ci-dessus, COSI entretient le dialogue avec l'expert en demandant des informations et, en même temps, en lui laissant le contrôle. Le dialogue de COSI est structuré en suivant les phases d'explicitation d'une tâche (figure 6.8) :

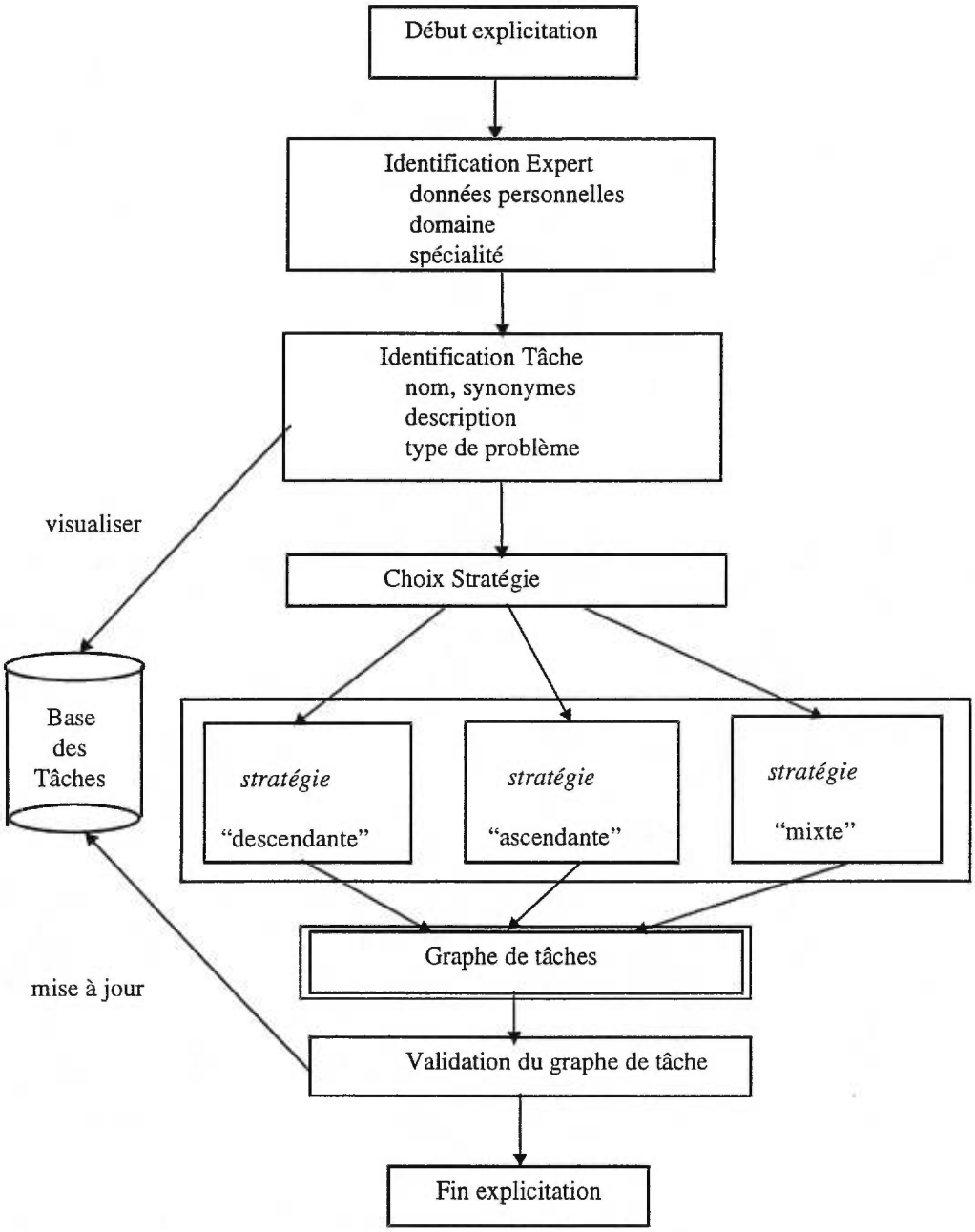


figure 6.8: Les phases d'explicitation d'une tâche.

Le dialogue implique le transfert d'informations entre l'expert et COSI. La figure 6.9 présente les négociations expert-COSI pendant une session de dialogue.

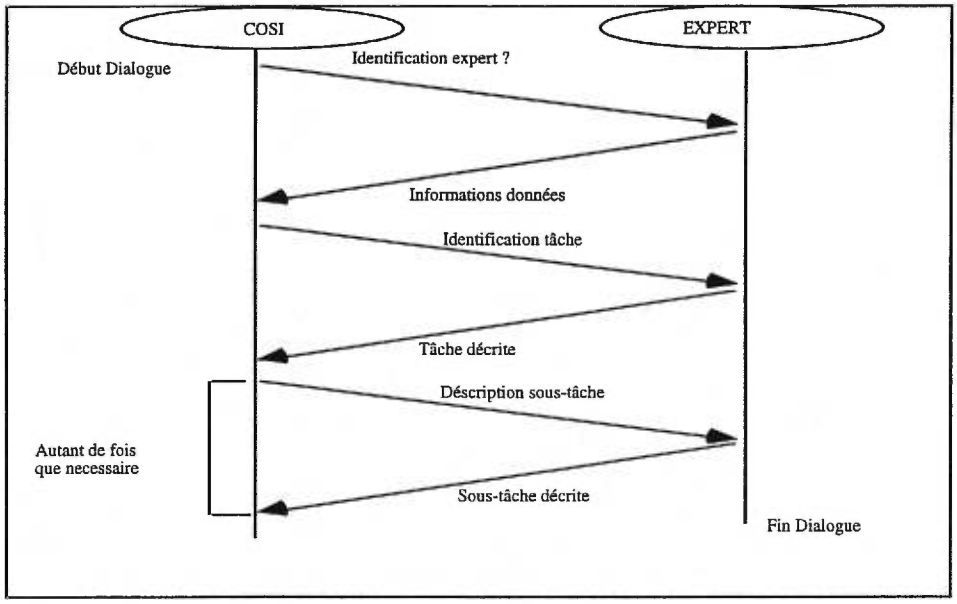


figure 6.9: La structure du dialogue entre COSI et Expert

Pour faciliter la lecture de ce chapitre, le dialogue entre COSI et l'expert peut traduit en langage naturel de la manière suivante:

....

COSI : Quelle tâche voulez-vous décrire ?

Expert : examen du système cardio-vasculaire

COSI : Pouvez-vous me dire s'il s'agit d'une tâche élémentaire ou d'une tâche décomposable ?

Expert : Quelle est la différence entre une tâche élémentaire et une tâche décomposable ?

COSI : Une tâche décomposable est identifiée par un ensemble des sous-tâches. Par contre, une tâche élémentaire est une simple action. Êtes-vous satisfait par cette explication ?

Expert : Oui. L'examen du système cardio-vasculaire est une tâche décomposable.

COSI : Bien. Comment procédez-vous pour réaliser un examen cardio-vasculaire?

Expert : Examen du cœur, examen des vaisseaux.

COSI : Décrivez maintenant chaque tache

Expert: nom de la tâche : examen du cœur

type : décomposable

description : ...

nom de la tâche : examen des vaisseaux

type : décomposable

description : ...

COSI : Précisez l'ordre d'exécution pour les tâches :

examen du cœur, examen des vaisseaux.

Expert : On peut réaliser ces tâches dans n'importe quel ordre.

COSI : Comment procédez-vous pour réaliser un examen du cœur ?

...

Cette section a présenté la communication expert-COSI comme un dialogue interactif entre ces deux entités. Ce dialogue est guidé par les trois stratégies d'explicitation décrites dans la section 6.3.2. Nous avons essayé de fournir à l'expert une assistance adéquate et de l'aider chaque fois qu'il en a besoin (en lui suggérant la stratégie appropriée à la tâche, en lui donnant la possibilité de changer la stratégie d'explicitation et en lui donnant des explications en ce qui concerne les interfaces qu'il manipule).

6.5 Contribution de COSI

Cette section a comme objectif d'évaluer l'outil COSI dans le cadre du projet SAFARI. Dans ce contexte, l'acquisition des connaissances a été abordée de deux manières différentes:

1. Acquisition manuelle des connaissances;
2. Acquisition semi-automatique avec COSI.

Ces deux approches ont pour but commun de construire un curriculum utilisable par SAFARI.

Notre intérêt pour l'acquisition semi-automatique est justifié par des besoins pratiques. En effet, les expériences sur l'acquisition manuelle des connaissances telle qu'elles ont été réalisées en SAFARI, nous ont guidé vers cette deuxième démarche.

Décrivons d'abord l'acquisition manuelle des connaissances en SAFARI:

Objectif: Il s'agit d'une méthode d'acquisition spécialisée pour le domaine médical. Le but est la génération de plusieurs scénarios (chaque scénarios décrit un cas particulier, comme ,par exemple, l'ensemble des informations concernant un dysfonctionnement d'une voiture).

Techniques utilisées: entretiens avec les experts,
observation de l'expert en situation de travail.

Déroulement du processus: Le cogniticien demande à l'expert de lui présenter un cas particulier. Ensuite, il raffine les informations recueilles pour déterminer celles qui peuvent être utilisées pour remplir les structures implémentées en SAFARI. Si le cogniticien constate qu'il y a des informations manquantes une autre séance de travail avec l'expert est nécessaire.

Problèmes:

- La qualité du processus d'acquisition manuelle dépend des habilités de communication de l'expert et du cogniticien
- La complexité de la tâche du cogniticien, En effet, ce dernier doit connaître:
- la technique d'acquisition qui favorise l'explicitation des connaissances par rapport à un expert et à une tâche donnée (un expert peut décrire verbalement une tâche ou, dans certain cas, il est préférable de l'assister lors d'une session de travail)
- les contraintes imposées par les structures implémentées pour éviter l'explicitation des connaissances inutiles (il doit connaître, par exemple, la notion de comportement attachée à une tâche, etc).
- S'il y a des incohérences de données (par exemple, il est difficile de s'assurer que toutes les relations entre les différentes tâches décrites ont été définies) ce processus peut demander plusieurs retours arrière nécessitant différentes interviews.

Solutions

Face à ces problèmes, nous avons tenté de réaliser une partie du processus d'explicitation à l'aide de l'outil COSI. Ainsi:

- Il est possible de mémoriser d'une session à l'autre toutes les connaissances explicitées auprès de l'expert. Ceci peut constituer un point de départ pour la réutilisation de connaissances acquises pour réduire le temps d'acquisition (par exemple, il est intéressant d'expliciter une tâche à partir d'une tâche existante dans la base de COSI).
- L'expertise (tout ce qu'a dit l'expert) est normalisée dans une structure unique: la tâche.
- La structure de tâche est plus conceptuelle que specifique à une implémentation particulière. Nous retrouvons ici le niveau connaissance (knowledge level) décrit dans le chapitre 3. En utilisant COSI, les structures implémentées sont transparentes pour l'expert (celui-ci ne doit pas penser en termes d'objets pour expliciter une tâche).
- COSI permet d'effectuer des contrôles de cohérence. COSI aide à la structuration des connaissances explicitées. Par exemple il peut signaler:
  - l'existence de cycles dans le graphe (une tâche se retrouve en tant que sous-tâche d'elle même),
  - les tâches restant à décomposer,
  - les tâches décrites de manière isolées des autres (non-connectées au reste du graphe de tâches).
- Grâce aux heuristiques implémentées, COSI aide l'expert à choisir sa technique (stratégie) d'explicitation. Il reste beaucoup de travail sur ce point. Par exemple, on pourrait étendre les heuristiques de COSI pour que ce dernier oriente le cogniticien vers le choix d'autres techniques d'acquisition appropriées lorsque nécessaire.
- COSI peut jouer le rôle d'un support permettant de focaliser la discussion expert-cogniticien. COSI ne demande que les connaissances de type tâche en évitant les discussions inutiles.

Comparaison avec les éditeurs de SAFARI

Regardons concrètement ce qu'aurait demandé l'éditeur traditionnel de SAFARI pour expliciter une tâche.

Une tâche est explicitée à l'aide des objets graphiques. Ceci ressemble beaucoup à l'utilisation de l'éditeur WORD quand on veut insérer des graphiques dans un document.

Lors d'une session de travail il n'y a aucun guide de manipulation de ces objets. De plus, pour utiliser l'éditeur de tâches de SAFARI l'expert doit connaître la signification de chaque objet de l'interface (par exemple, qu'un rectangle représente une tâche décomposable et qu'une ellipse une tâche élémentaire, etc).

L'explicitation des tâches avec l'éditeur traditionnel est lente et susceptible d'erreurs pour des tâches complexes. Ce processus s'appuie beaucoup sur l'expert qui doit assurer, par lui même, la cohérence de données saisies (par exemple, il doit connaître toutes les tâches saisies pour éviter les duplications, s'il existe des tâches isolées, etc.).

Expériences avec COSI

Les expériences faites avec COSI ont montré des différences importantes en temps d'explicitation et en nombre de spécialistes impliqués par rapport à la méthode d'acquisition manuelle. Par exemple, l'explicitation des connaissances concernant le fonctionnement de la pompe Baxter (appareil médical pour faire des infusions) a demandé trois mois de travail et quatre spécialistes : une infirmière (l'expert du domaine), un expert en psychologie cognitive (le cogniticien) qui effectue les entretiens avec l'infirmière, un étudiant en informatique dont le rôle est de structurer les connaissances recueilles par le cogniticien compte tenu du modèle de tâche de SAFARI. En revanche, COSI permet la réalisation d'un tel processus en quelques jours. De plus, étant donné que COSI effectue une partie de la tâche du cogniticien et de l'informaticien (il mémorise et organise les connaissances de l'expert) le seul spécialiste indispensable est l'expert du domaine.

COSI, quant à lui, propose des solutions originales aux problèmes ci-dessus:

- il guide le recueil des connaissances en simulant un dialogue avec l'expert,
- il assure la cohérence des données saisies,
- il permet de réduire le temps d'explicitation par rapport à la méthode d'acquisition avec l'éditeur graphique de SAFARI.

6.6 Conclusion

Nous nous intéressons à la façon dont un expert résout un problème. Nous considérons la tâche et l'expert comme deux composants inséparables. Par conséquent, notre but est d'expliciter les connaissances relatives à la tâche et à l'expert, car on ne peut pas spécifier une tâche sans connaître certaines caractéristiques de l'expert qui la définit, notamment sa stratégie d'explicitation préférée.

Notre approche soulève trois points d'intérêt majeurs:

- Premièrement, nos travaux vont dans le sens d'une meilleure insertion des idées de la psychologie cognitive et de l'ergonomie cognitive lors du processus de conception d'un système à base de connaissances (l'analyse de problème via plusieurs stratégies d'explicitation et le modèle de l'expert cf. 6.2.1.1).
- Deuxièmement, pour autoriser l'utilisation de l'outil COSI par plusieurs types d'experts de domaines différents, nous avons proposé un système offrant plusieurs stratégies d'explicitation. En l'état actuel, il intègre trois stratégies générales d'explicitation de tâches pour modéliser le dialogue expert-COSI. Ces stratégies ne sont pas nouvelles dans le domaine, toutefois, la plupart des outils d'acquisition sont spécialisés pour un seul type de stratégie (la stratégie descendante est la plus utilisé, par exemple, dans SCARP, etc.). Ce n'est pas le cas de COSI, qui intègre trois stratégies générales dans le même outil et par conséquent est utilisable dans plusieurs types de problèmes. D'autre part, ces stratégies sont très générales et ne sont utilisables que pour des tâches décomposables.
- L'exploitation de la modélisation de l'expert par des heuristiques générales permet de faciliter l'explicitation compte tenu de certaines caractéristiques de l'expert. Même si actuellement l'aide à l'expert vise surtout l'aspect interface et les conseils pour une stratégie d'explicitation, ajouter des heuristiques liées un domaine est un point à développer dans le cadre de ce travail.

Chapitre 7

Implantation de l'outil COSI et expérimentation

La mise en œuvre de l'outil COSI repose sur les principes de conception énoncés dans le chapitre 6. Notre application est développée en SmallTalk-80 [Goldberg & Robson, 89], [Dugerdil, 89] dans l'environnement VisualWorks 2.0 [VisualWorks, 92]. Pour illustrer le fonctionnement de l'outil COSI nous avons choisi comme application le problème "Examen clinique d'un patient". Un exemple de session d'utilisation de l'outil COSI est présenté dans la section 7.2.

7.1 Implantation du prototype COSI

Comme tout langage orienté objet, SmallTalk offre des bonnes primitives (*classes, héritage, relations, méthodes*) bien adaptées à la modélisation de certains domaines d'application. Le principal avantage de SmallTalk est d'offrir un environnement de programmation interactif avec des objets graphiques permettant le développement d'interfaces conviviales et facilement utilisables.

Puisque notre système est dédié à un expert supposé n'avoir aucune expérience de programmation nous avons insisté plutôt sur l'aspect fonctionnel que sur l'aspect "publicitaire" des interfaces (beaucoup de couleurs et d'icônes, etc). Ainsi nous avons développé des interfaces simples pour faciliter le dialogue expert-système.

7.1.1 Le modèle MVC: principe de base pour la construction de toute interface SmallTalk

Toute application développée en SmallTalk se décompose, selon le modèle MVC [Dugerdil, 89] en trois modules:

- le *modèle* pour stocker et manipuler les données de l'application,
- la *vue* qui est un objet permettant l'affichage de certaines caractéristiques du *modèle*. C'est une présentation à l'utilisateur d'une certaine perspective du modèle,
- le *contrôleur* attaché à une vue a pour rôle de gérer l'interaction entre l'utilisateur et la *vue*. Compte tenu des différentes manipulations de l'usager, le *contrôleur* doit lancer les actions demandées, ce qui entraîne la mise à jour du modèle.

Prenons un exemple simple. Un des buts de COSI est d'obtenir certaines informations concernant l'expert qui l'utilise. Pour cela nous avons créé un modèle, appelé ModèleExpert, qui va garder toutes données définissant le profil de l'expert (par exemple sa spécialité, son domaine de travail, etc.). La vue qui est affichée à l'écran est présentée sur la figure 7.4. Les données saisies par l'expert sont repérées par le contrôleur qui va initialiser le modèle (ses variables privées) avec les différentes informations fournies.

7.1.2 Les classes utilisées et développées pour COSI

Nous rappelons ici qu'après une session de travail avec l'expert, COSI a pour rôle de fournir un graphe contenant la solution de l'expert pour le problème posé. Ce graphe sera utilisé lors des sessions tutorielles par un algorithme, appelé algorithme de matching, qui évalue les réponses d'un apprenant en comparaison avec la solution de l'expert [Pachet et al., 95].

Puisque dans COSI l'acquisition est vue comme l'instanciation d'un modèle de tâches (la structure opérationnelle présentée dans la section 6.1.1.3), il s'agit en fait, d'une part, de *réutiliser* toutes les classes définissant ce modèle générique et, d'autre part, de *développer* des nouvelles classes permettant l'explicitation des connaissances pour une application particulière.

Cette section ne se veut pas une description exhaustive de toutes les classes réutilisées ou développées dans COSI. Nous allons nous restreindre à la présentation des trois classes importantes qui ont été réutilisées pour développer COSI. Ensuite nous nous concentrons sur quelques-unes des nouvelles classes que nous avons ajoutées.

7.1.2.1 Les principales classes réutilisées par COSI

- *La classe FPTask*

  Une FPTask représente une tâche définie par les informations suivantes:

  - son nom,
  - sa description,
  - la liste de ses sous-tâches,
  - la liste de ses tâches parents,
  - son comportement.

- *La classe ComportementTache*

  Une classe ComportementTache concerne les relations d'ordonnancement entre les sous-tâches d'une tâche donnée. Un "comportement" particulier correspond à une sous-classe de la classe ComportementTache. Ainsi, notre but est de créer pour chaque tâche une instance d'une des sous-tâches de la classe ComportementTache.

- La classe *EditeurGrapheDesTâches*

  Permet l'édition graphique des connaissances selon le formalisme des graphes. Les méthodes attachées à cette classe peuvent être divisées en deux groupes:

- les méthodes permettant la construction des graphes en manipulant les objets graphiques de l'éditeur,
- les méthodes facilitant la manipulation des graphes construits comme, par exemple: charger un graphe se trouvant dans un fichier, afficher horizontalement ou verticalement des graphes, attacher des explication à chaque tâche, etc.

7.1.2.2 Les nouvelles classes développées

- *La classe COSI*

La classe COSI est le noyau de l'application. Elle gère le déroulement d'une session d'explicitation avec COSI comprenant aussi bien la saisie de données que la visualisation et la validation des résultats (les graphes construits). Cette classe joue le rôle d'un planificateur du travail car elle doit savoir, compte tenu de l'étape du dialogue et des préférences de l'expert, quelle sera l'interface qu'elle doit afficher par la suite. Par exemple, si l'expert a choisi comme stratégie d'explicitation la stratégie descendante, COSI va afficher une suite d'interfaces pour décomposer une tâche en sous-tâches. Par contre, pour une stratégie ascendante, l'ordre de l'affichage des interfaces sera différent, pour identifier tout d'abord les tâches élémentaires et ensuite les regrouper en tâches plus complexes.
Ainsi, l'intérêt principal de la classe COSI est de lancer l'exécution des méthodes associées aux différentes classes de l'application.

- *La classe IdentificationExpert*

Cette classe permet d'instancier le modèle de l'expert (la classe ModeleExpert) à partir des informations saisies (son nom, son domaine de travail, sa spécialité, sa stratégie de raisonnement préférée).

- *La classe IdentificationTâche*
- Cette classe sert à initialiser la racine du graphe, la liste de nœuds et la liste d'arcs qui forment le graphe pour une tâche particulière.

- *La classe IdentificationSousTâches*

Si l'expert a choisi la stratégie d'explicitation descendante cette classe lui permet de décrire toutes les sous-tâches d'une tâche donnée avant de passer à la décomposition de chacune de ces sous-tâches.

- *La classe ComposerSousTâches*

Si l'expert a choisi la stratégie d'explicitation ascendante ou mixte, cette classe peut l'aider à définir une partie du graphe en précisant pour un sous-ensemble des tâches saisies quelle(s) est (sont) leur(s) tâche(s) parent(s). Ensuite les listes des sous-tâches et des tâches parents sont mises à jour pour chaque tâche.

- *La classe IdentificationComportementTâche*

Une fois toutes les sous-tâches d'une tâche saisies, cette classe permet d'afficher la liste des sous-tâches et d'attacher le "comportement" de l'ensemble des sous-tâches à la tâche parent.

- *La classe VisualiserGraphe*

Cette classe active l'éditeur des tâches et déclenche l'affichage du graphe des tâches construit lors d'une session de dialogue avec COSI.

Dans la figure 7.1 nous présentons un modèle partiel de l'application intégrant les principales classes décrites dans cette section et leurs relations.

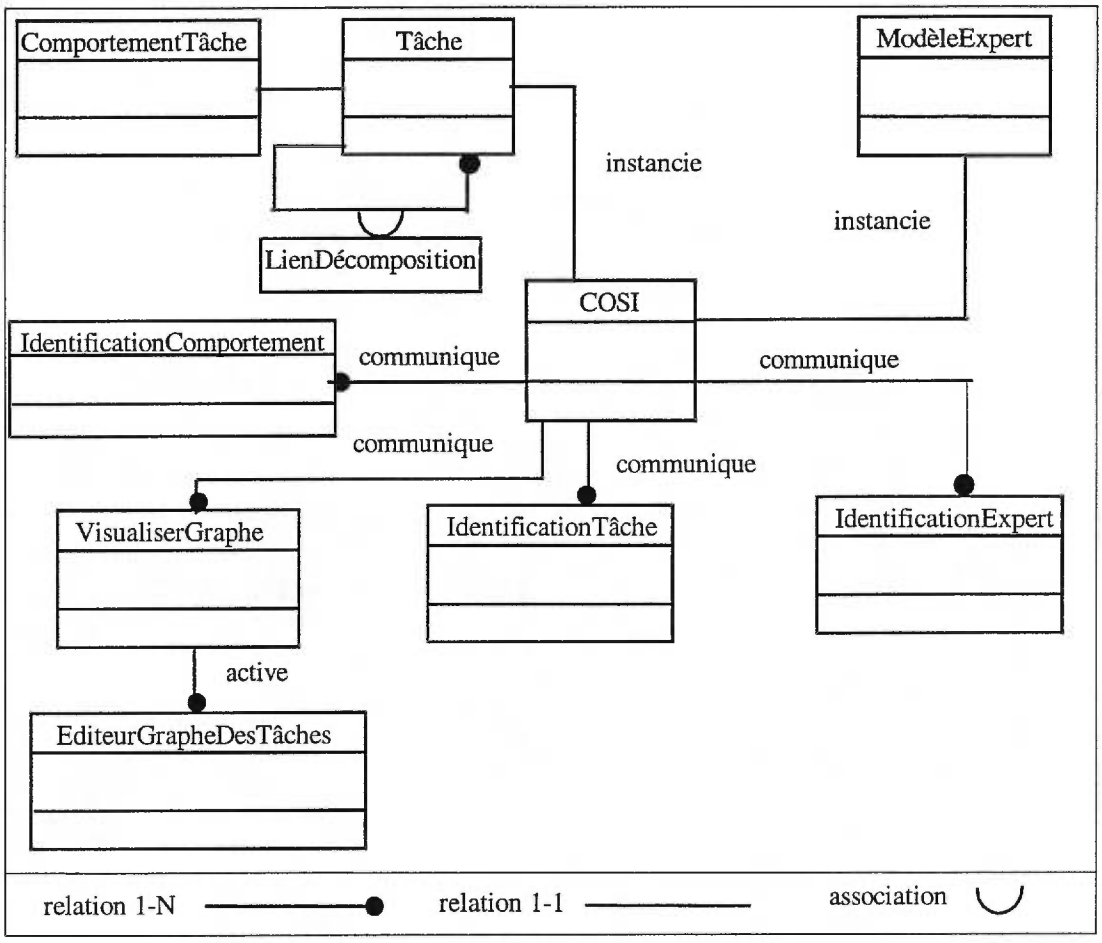


figure 7.1: Modèle objet (une partie) de l'application.

7.1.3 Les écrans d'aide à l'utilisation du système

Nous avons créé plusieurs écrans d'aide pour permettre à l'expert d'être précis en ce qui concerne les informations saisies et de faciliter l'utilisation de COSI. Plutôt que d'offrir une "aide globale" comme cela est le cas dans la plupart des logiciels (les traitements de textes en sont un bon exemple), chaque interface dispose de son propre "conseiller" concernant aussi bien les termes manipulés par COSI que les informations à demander à l'expert. Ainsi, chaque interface dispose d'un système d'aide structuré de manière hiérarchique. Au premier niveau nous trouvons un index contenant les éléments de l'interface et les "termes" utilisés par COSI. De cette manière COSI peut expliquer à l'expert des

notions comme stratégie, tâche, etc., et lui donner des précisions sur les informations à saisir. La figure 7.2 présente deux écrans d'aide attachés à l'interface concernant les informations de l'expert (figure 7.4).

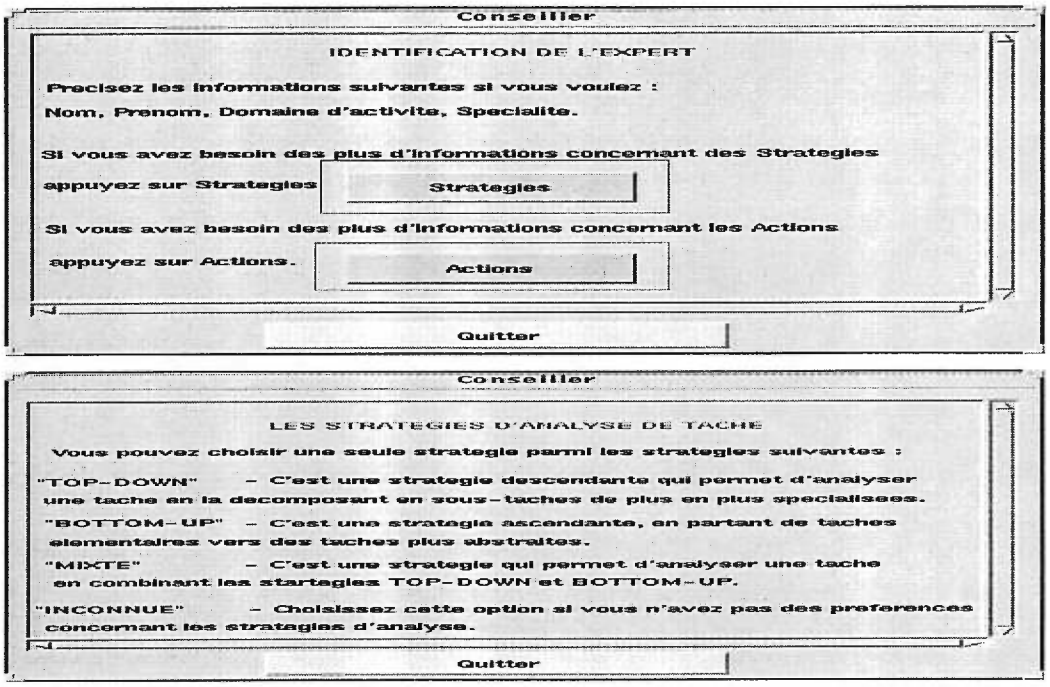


figure 7.2 Les écrans d'aide attachés à l'interface concernant les informations de l'expert.

7.2 Une session d'explicitation de connaissances avec COSI

Nous allons présenter ici une expérimentation de l'outil COSI pour une application médicale. La première section est un bref résumé du raisonnement clinique d'un médecin. Nous présentons ensuite, à l'aide de copies d'écran, le déroulement d'une session de travail avec COSI, pour analyser la tâche "examen cardio-vasculaire". Nous soulignons ici le rôle important joué par le médecin Nâila Zalila, en qualité d'expert en médecine, concrétisé en deux tâches essentielles:

- la mise en œuvre de différents scénarios,
- les recommandations sur la présentation des interfaces.

7.2.1 Les principales étapes du raisonnement clinique

D'une manière générale on peut scinder le raisonnement clinique d'un médecin en plusieurs phases [Lupascu & Zalila, 95]:

- La première est l'obtention d'une *histoire détaillée du patient*: données générales (âge, poids, taille, sexe) et antécédents pathologiques (personnels et familiaux).
- La deuxième phase - la réalisation de l'*examen clinique* du malade - est l'étape la plus importante. L'examen doit être pratiqué sur tout le corps à la recherche des anomalies en vue d'un diagnostic. Il comporte des examens spécifiques à chaque système du corps humain: examen des signes généraux (température, état de la peau), examen cardio-vasculaire, examen abdominal, examen neurologique, etc. À la fin de l'examen clinique, on dispose d'un ensemble d'*hypothèses de diagnostic* fondées sur l'histoire du patient et sur les signes ayant amené le malade à consulter.
- Dans certains cas, lors d'une troisième étape, le médecin doit prescrire des examens complémentaires afin de confirmer ou d'infirmer tel ou tel autre diagnostic.
- Dans une quatrième phase, il faut réaliser la synthèse entre les données cliniques et les résultats des examens prescrits afin d'apprécier la probabilité de la maladie par rapport aux autres diagnostics différentiels. L'objectif de cette étape est d'aboutir à un diagnostic précis et fiable.
- La dernière étape est relative à la mise en place d'une attitude thérapeutique adaptée au diagnostic évoqué.

7.2.2 Comment COSI aide-t-il l'expert à analyser le problème "Examen clinique d'un patient"

Le problème "Examen clinique d'un patient" est particulièrement intéressant car l'expertise qui s'y rattache contient beaucoup de connaissances de type tâche décrivant la pratique clinique d'un médecin à la recherche d'anomalies. De plus, la résolution d'un tel problème nécessite tout d'abord une analyse rigoureuse par décompositions successives car l'examen clinique est composé d'examens spécifiques pour chaque système du corps humain

(par exemple, examen abdominal). De plus, pour chacun de ces systèmes il faut examiner systématiquement les organes qui lui appartient en vue de localiser de manière précise l'organe atteint.

7.2.2.1 Déroulement d'une session de travail

Nous présentons par la suite le dialogue expert-COSI pour expliciter les connaissances nécessaires à définir la statique et la dynamique de la tâche "Examen du système cardio-vasculaire".

La figure 7.3 présente le menu principal de l'application qui donne l'accès aux trois modules du système. Ainsi, à partir de cette interface l'expert peut choisir une des options suivantes:

- lancer le dialogue avec COSI pour expliciter une tâche (l'option "Analyse de tâche").
- construire un graphe de tâche en utilisant l'éditeur graphique de tâche. Ceci demande de la part de l'expert d'avoir une certaine habileté dans la manipulation de différents objets graphiques de l'éditeur de tâches.
- expliciter les principaux concepts du domaine et leurs relations. Étant donné notre focalisation sur les connaissances de type tâche, le module "Définition de concepts" est implanté de façon minimale. Plus précisément, nous nous sommes limité à identifier les éléments de base pour un problème de diagnostic médical tels : les maladies, les examens, les bilans et les symptômes. Nous envisageons une amélioration de ce module afin de le rendre utilisable pour des applications liées à d'autres domaines.

Par la suite nous nous concentrons sur la manière dont un expert peut expliciter une tâche en dialoguant avec COSI. Supposons donc que l'expert commence la session de dialogue en sélectionnant l'option "Analyse de tâche" (figure 7.3).

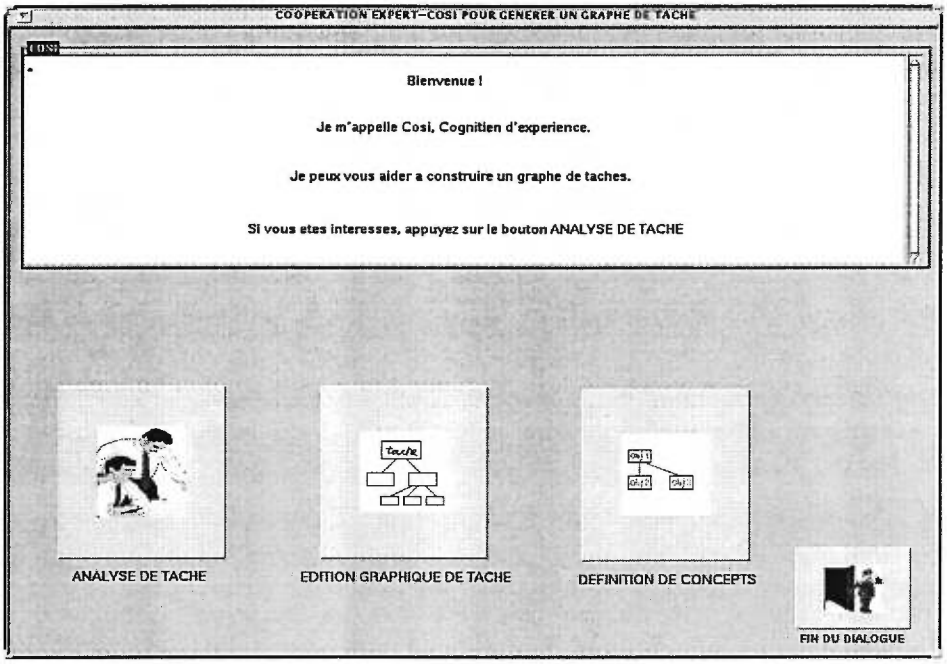


figure 7.3: Menu principal de l'application.

Le modèle de l'expert sera présenté partiellement dans les interfaces. Le but est d'avoir plusieurs vues pour une interface en fonction des caractéristiques de l'expert. Par exemple, la vue suivante permet le recueil des informations personnelles de l'expert (figure 7.4). Dans ce cas, étant donné que l'expert a une bonne expérience en explicitation des connaissances, COSI lui demande de choisir une stratégie d'explicitation (pour un expert sans expérience en explicitation, COSI va proposer directement une stratégie d'explicitation en fonction du type de problème qui sera identifié ultérieurement figure 7.5)

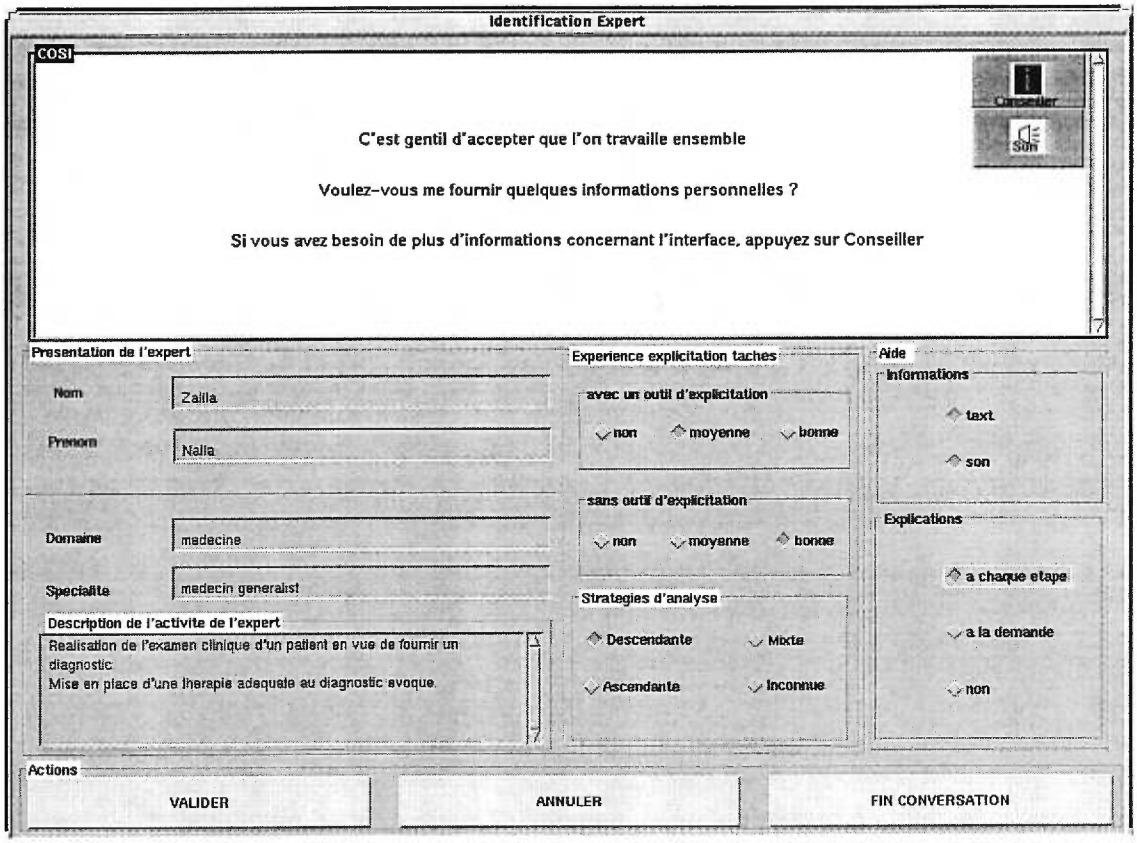


figure 7.4: Questionnaire préliminaire d'identification de l'expert.

Ainsi, un expert expérimenté peut expliciter la tâche en fonction de ses préférences; il a donc la possibilité de procéder de manière descendante, ascendante ou mixte. COSI demande des informations à l'expert et en même temps il lui laisse le contrôle du dialogue. Ainsi, comme montré dans les figures 7.5-7.9, l'expert peut, lors d'une consultation, changer la stratégie d'explicitation, revenir sur des étapes antérieures, ou encore suspendre une consultation à tout moment

Après que l'expert se soit présenté à COSI, ce dernier déclenche le processus d'explicitation de la tâche. Supposons que l'expert ait choisi la stratégie de résolution descendante.

COSI déclenche le processus d'explicitation de la tâche en demandant à l'expert de spécifier la nature de la tâche à construire (figure 7.5).

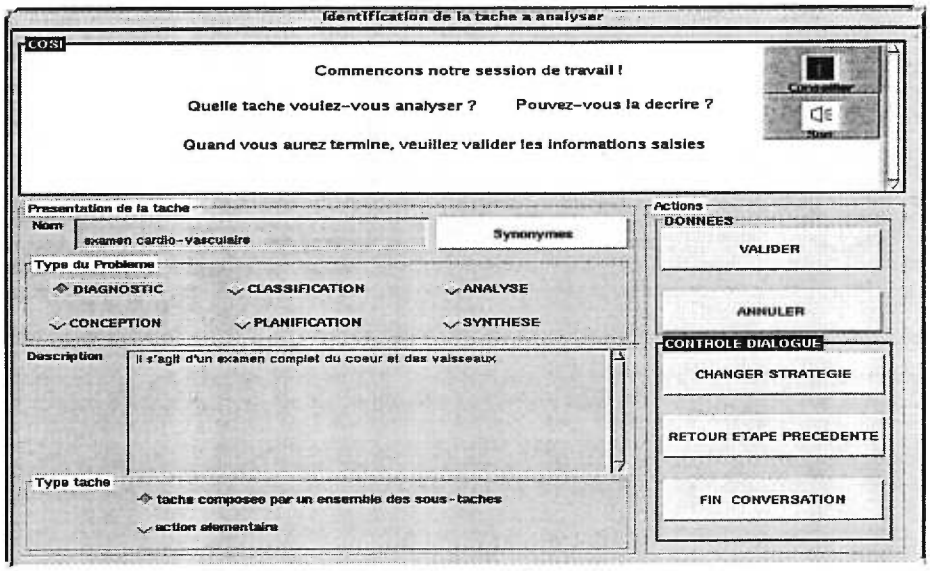


figure 7.5: Identification de la tâche

Le "type du problème" est une information importante pour le bon déroulement du processus d'explicitation d'une tâche. Dans ce cas, puisqu'il s'agit d'un problème de type diagnostic, COSI vérifie, grâce à ses heuristiques (voir section 6.3.3), si la stratégie choisie par l'expert est la stratégie appropriée pour ce type de problème. Étant donné qu'il existe une différence entre la stratégie proposée par l'expert (qui a choisi la stratégie ascendante) est celle connue par COSI comme appropriée pour un problème de diagnostic (la stratégie descendante), COSI demande à l'expert s'il souhaite réviser son choix (figure 7.6).

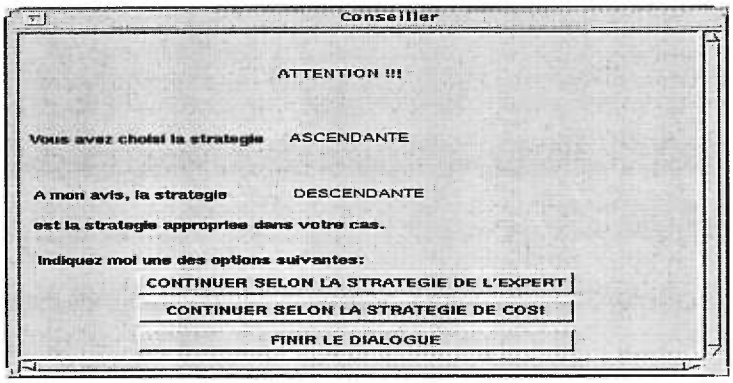


figure 7.6: Conseil de COSI concernant la stratégie d'explicitation.

Puisque l'expert est d'accord avec la proposition de COSI (que la stratégie d'explicitation appropriée est la stratégie descendante), le dialogue expert-COSI sera guidé par les méthodes associées à la stratégie descendante.

Sur l'interface présentée dans la figure 7.5, il est également à noter qu'un bouton permet de lancer l'éditeur de synonymes de tâches. L'expert à donc la possibilité d'ajouter, d'effacer, d'afficher ou encore de sélectionner un synonyme de tâche.

A partir de là, COSI cherche à savoir comment se déroule le processus d'explicitation de la tâche "examen cardio-vasculaire". Puisque la tâche "examen cardio-vasculaire" est une tâche décomposable, COSI demande à l'expert s'il préfère identifier toutes les sous-tâches de la tâche "examen cardio-vasculaire" avant de passer à la décomposition de chacune de ses sous-tâches (ce qui correspond à la politique en "largeur d'abord") ou s'il veut les analyser une à une (la politique "largeur ou profondeur")(figure 7.7).

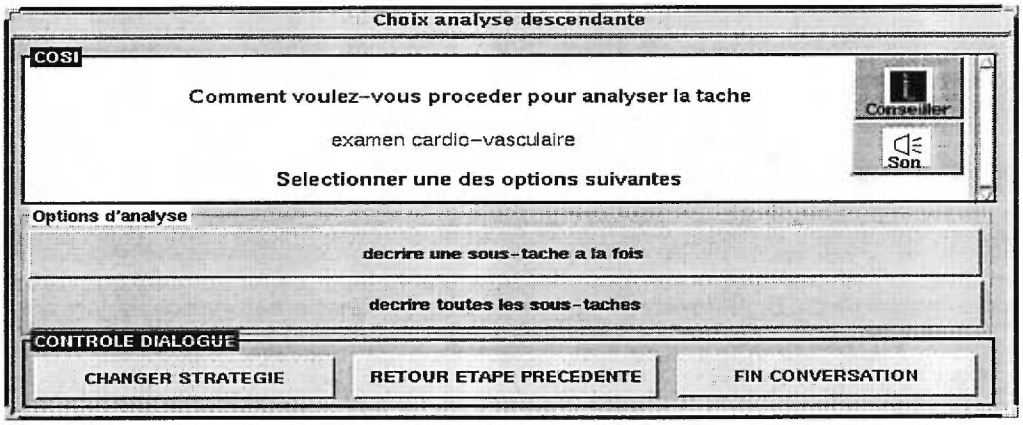


figure 7.7: Choix de politique pour la stratégie descendante.

Supposons que l'expert ait choisi la deuxième option. L'interface suivante (figure 7.8) a pour objectif d'identifier les sous-tâches à expliciter. Ainsi l'expert saisit les deux sous-tâches: "examen cœur" et "examen vaisseaux". Le bouton "AJOUTER une sous-tâche" permet l'ajout d'une sous-tâche dans la liste de sous-tâches de la tâche "examen

cardio-vasculaire". Une fois toutes les sous-tâches saisies, le bouton "VALIDER les sous-tâches" fermera l'interface.

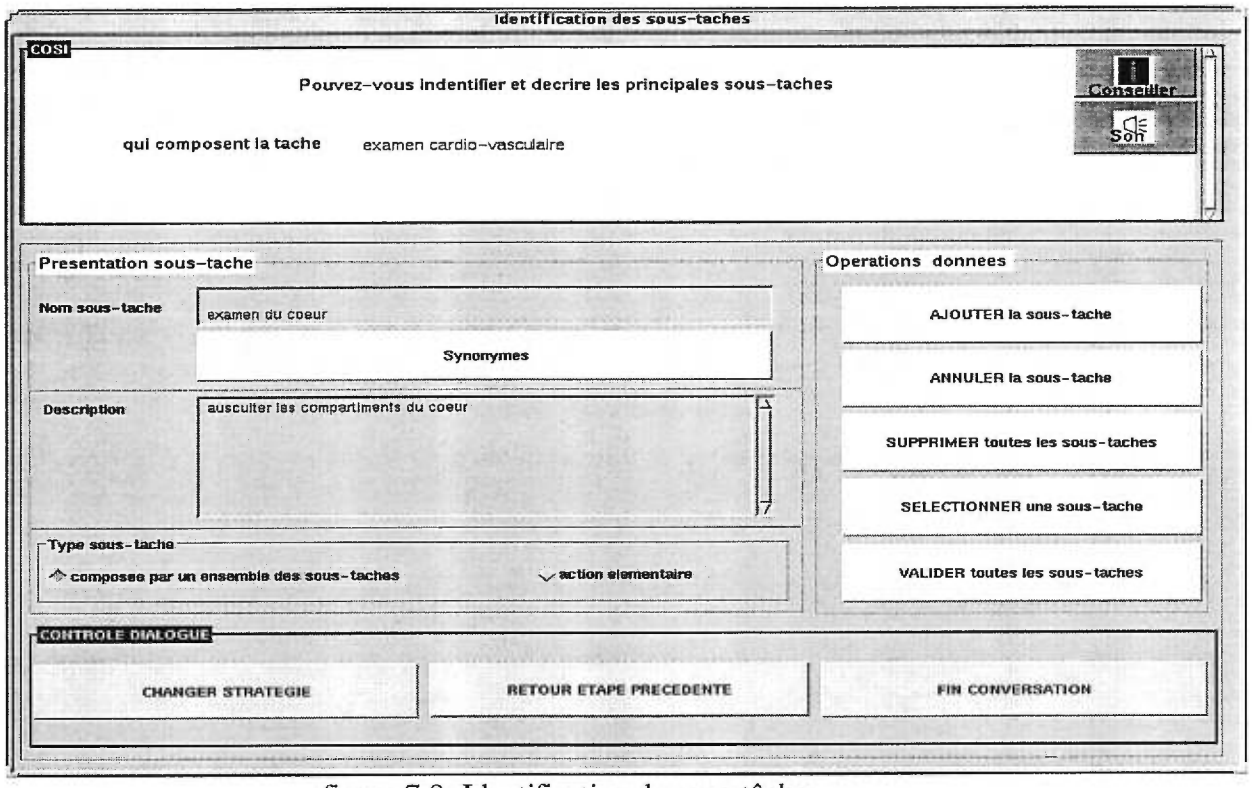


figure 7.8: Identification de sous-tâche.

Une fois que toutes les sous-tâches ont été identifiées pour une tâche, COSI demande à l'expert de préciser le comportement (l'ordonnancement) de l'ensemble des sous-tâches lors de l'exécution. L'expert peut choisir parmi plusieurs options comme, par exemple:

- l'ordre d'exécution entre plusieurs sous-tâches n'est pas important,
- les sous-tâches doivent être réalisées dans un ordre précis,
- il faut réaliser une seule sous-tâche parmi celles ayant été définies.

Pour cet exemple (figure 7.9), l'option choisie par l'expert signifie que les deux sous-tâches (examen cœur et examen vaisseaux) peuvent être effectuées dans n'importe quel ordre.

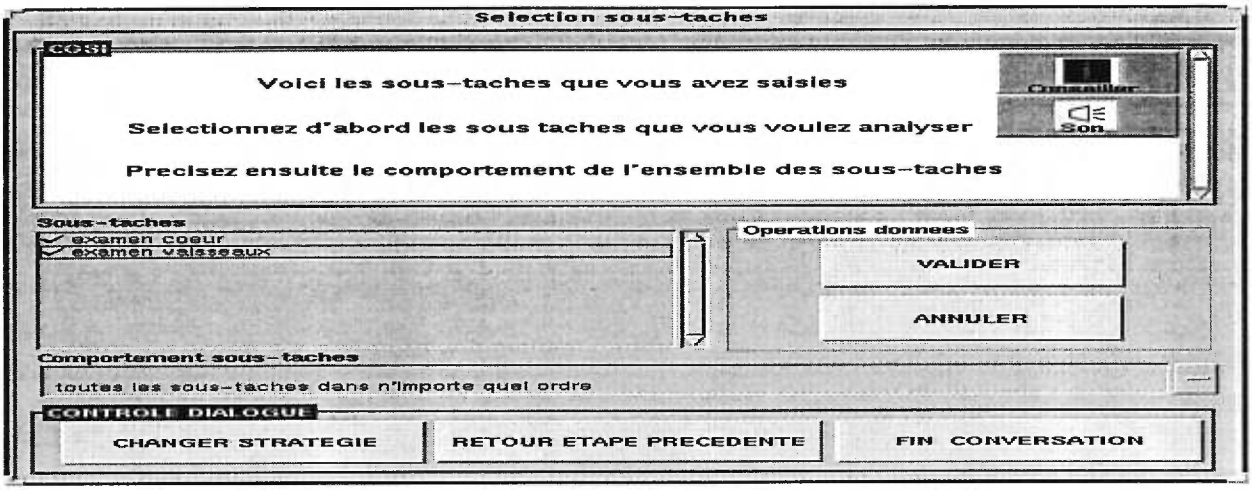


figure 7.9: Comportement de la tâche "examen cardio-vasculaire".

Le même processus va permettre d'expliciter de manière récursive chaque sous-tâche. L'explicitation de la tâche est terminée lorsque l'expert considère qu'il ne reste plus de tâches à décrire.

Une fois l'explicitation de la tâche achevée, l'option "validation de la tâche" du menu principal de l'application permet à l'expert de visualiser et de valider la tâche construite (figure 7.10).

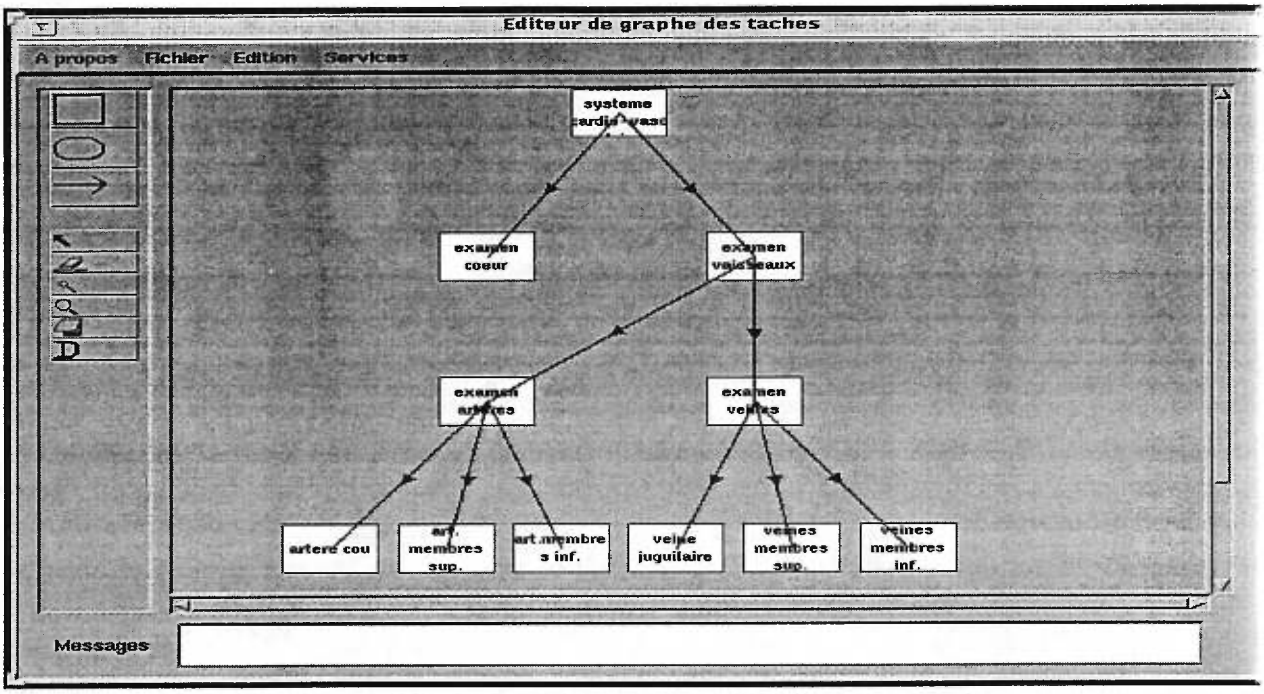


figure7.10: Le graphe pour la tâche "examen cardio-vasculaire"

Le scénario présenté dans cette section correspond à un des chemins possibles dans l'arbre illustré dans le schéma suivant (flèches en gras, figure 7.11).

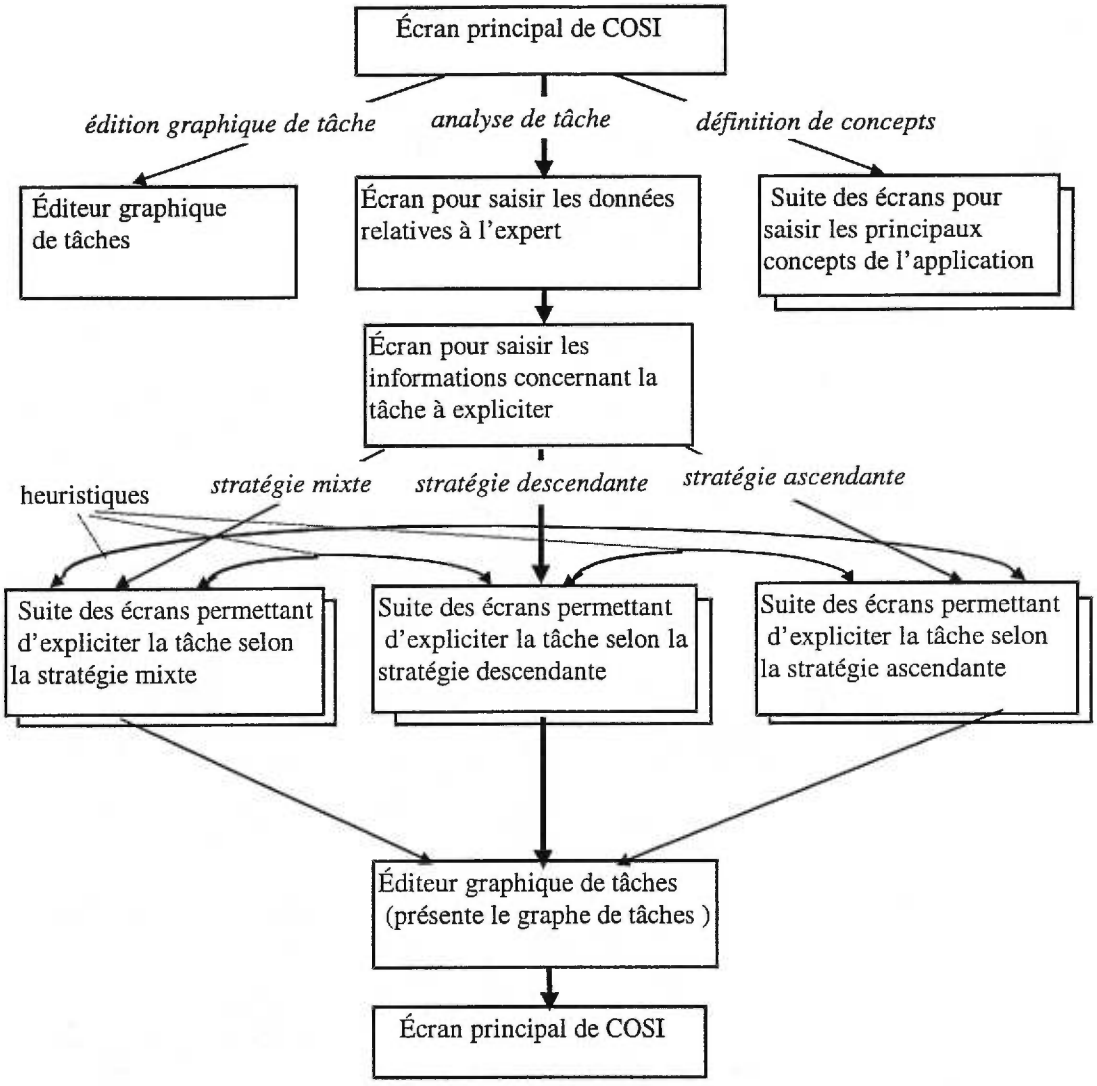


figure 7.11 : L'arbre de scénarios possibles dans l'utilisation de COSI

L'implantation de l'outil que nous proposons supporte le dialogue expert-COSI permettant la construction d'un graphe de tâches. Nous avons terminé la session d'utilisation de COSI sur un exemple, le problème de l'examen cardio-vasculaire d'un patient en présentant le graphe de tâches résultant.

Chapitre 8

Conclusion

Bien que le but principal d'un système tutoriel soit l'enseignement intelligemment assisté par ordinateur, il n'en demeure pas moins que la qualité de cette activité dépend en grande partie de la modélisation du domaine à enseigner, d'où notre intérêt pour l'acquisition des connaissances. Toutefois, l'explicitation de l'expertise du domaine reste une question peu abordée dans la communauté des STI. En effet, cette dernière s'est plus intéressée à la manière d'enseigner (stratégies pédagogiques, objectifs d'enseignement, etc.) plutôt qu'au contenu de la matière.

L'objectif de cet étude est double: tout d'abord étudier l'intérêt de l'application des notions et des techniques d'acquisition des connaissances pour la conception de tutoriels intelligents et, deuxièmement, concevoir un outil d'aide à l'acquisition des connaissances capable de réaliser en partie le travail d'un cogniticien humain. En effet, COSI coopère avec l'expert en vue de modéliser une tâche selon un modèle générique appelé structure opérationnelle. Nous avons synthétisé par la suite les caractéristiques principales de COSI, ses limites, ainsi que les perspectives ouvertes par ce travail.

Caractéristiques de COSI

La conception de COSI permet de résoudre certains problèmes relatifs à l'acquisition des connaissances:

- Contrairement à la plupart des systèmes d'acquisition de connaissances, l'outil COSI est impliqué dans le processus de construction d'un STI. Il a été intégré dans l'architecture CREAM en tant que ressource d'acquisition qui offre à un expert, même peu familier de l'outil informatique, un moyen de communication avec le système. Ce moyen repose sur un dialogue interactif expert-COSI.
- Une des caractéristiques importantes de COSI est la généricité. En effet, COSI n'est dédié ni à un type particulier de problème, ni à un domaine spécifique. De ce point de vue, il est comparable à des systèmes comme SCARP [Willamowski, 94], un système générique dédié essentiellement aux connaissances de type tâche.
- Dans la version actuelle, COSI permet l'explicitation des connaissances selon trois stratégies de dialogue eu égard aux aspects statiques et dynamiques de la tâche. Plus précisément, il prend en compte les préférences de l'expert quant à un type de dialogue donné.
- COSI peut utiliser des heuristiques afin de conseiller l'expert sur la manière de modéliser une tâche compte tenu de la nature de celle-ci (diagnostic, conception, planification etc.).
- Pour réaliser un dialogue "flexible" avec l'expert, COSI offre à ce dernier plusieurs alternatives pour "court-circuiter" le système. Ainsi, l'expert peut quitter le programme à tout moment ou interrompre la session de dialogue pour changer sa stratégie par une autre.
- En tant qu'outil d'acquisition basé sur un modèle, COSI guide l'explicitation des connaissances, en tenant compte d'une part du modèle de tâche et, d'autre part, des caractéristiques de l'expert qui le manipule.
- Étant donné que la construction d'un modèle de l'utilisateur est un problème difficile, nous nous sommes limité à la définition d'un modèle simplifié de l'expert contenant les préférences de ce dernier par rapport à une méthode d'explicitation de tâche.

Limitations

Dans l'état actuel de nos travaux plusieurs aspects ne sont pas encore résolus:

- L'utilisation de la méthode de planification hiérarchique limite l'applicabilité de COSI à des tâches pour lesquelles l'expert sera capable de préciser l'ensemble de sous-tâches à faire ainsi que leur ordonnancement à l'exécution. En revanche, cette approche est inadaptée à certains problèmes (comme par exemple ceux de type conception ou création) où les sous-tâches peuvent se succéder sans ordre a priori.
- Le modèle de tâches qui guide le travail de COSI est un modèle très général, indépendant de toute application et de tout domaine d'étude. De nouvelles heuristiques doivent être intégrées pour pouvoir conseiller l'expert en fonction de domaine d'application.
- Le modèle de l'expert ne tient pas compte des aspects liés aux facteurs humains. Ce modèle doit être étendu en tenant compte d'autres caractéristiques de l'expert (au delà de ses préférences par rapport à la méthode utilisée).

Travaux futurs

Les idées ainsi que les résultats pratiques présentés dans ce document offrent des nouvelles perspectives. Nous envisageons plusieurs travaux futurs.

- Nous voulons enrichir la dimension dynamique de la tâche en rajoutant par exemple des pré-conditions et des post-conditions associées à l'exécution de la tâche.
- Nous considérons important d'identifier tous les concepts du domaine manipulés par un expert pour réaliser une tâche. En effet, si les connaissances de type tâche permettent à un élève d'apprendre comment résoudre un problème, les concepts attachés à chaque tâche (ou sous-tâche) nous permettent de lui expliquer pourquoi on doit faire cette tâche. Comme premier pas dans cette direction, nous avons construit un modèle de domaine adapté à une application médicale.

- Pour faciliter le travail de l'expert, COSI devra lui permettre de construire d'une tâche à partir d'une tâche existante dans la base de tâches. De plus, la réutilisation des connaissances permet de réduire le temps d'acquisition.
- À long terme, notre objectif est d'offrir à chaque intervenant du STI (tuteur, expert du domaine ou élève) un outil d'explicitation de connaissances qui permette: au tuteur d'expliciter ses stratégies pédagogiques, à l'expert du domaine de modéliser la matière à enseigner et à l'élève de décrire sa manière de résoudre un problème.
- Intégrer COSI dans un environnement coopératif pour donner la possibilité à plusieurs experts de collaborer en vue d'expliciter les connaissances d'un système tutoriel intelligent. Des modèles de coopération, comme par exemple MONACO [Tadié, 96] pourraient être utilisés à cet effet.

BIBLIOGRAPHIE

[Aïmeur, 96] Aïmeur, E. - Acquisition de connaissances structurées par classification et discrimination, dans *Acquisition et Ingénierie des connaissances,* N. Aussenac-Gilles, P. Laublet, C. Reynaud (Editeurs), Ed. Cépaduès, ch. 12, pp.225-246, Paris, 1996.

[Aïmeur, 94] Aïmeur, E. - METIS: un système et une Méthode d'Explicitation de Taxinomies destinés à l'Identification de Structures conceptuelles, Thèse de l'Université Paris VI, 1994.

[Aïmeur & Frasson, 95] Aïmeur, E. & Frasson, C. - Modélisation de la tâche par explicitation dans un système tutoriel intelligent, Montreal, 1995.

[Alexe, 96] Alexe, C. - Analyse des modules de formation, Rapport interne NOVASYS, 1996.

[Aussenac et al., 96] Aussenac, N. - Aussenac, N., Laublet, P., Reynaud, C., - L'acquisition des connaissances: une composante à part entière de l'informatique du futur, dans *Acquisition et Ingénierie des connaissances*, N. Aussenac-Gilles, P. Laublet, C. Reynaud (Editeurs), Ed. Cépaduès, ch. 1, pp.3-25, Paris, 1996.

[Aussenac, 89] Aussenac N. - Conception d'une méthodologie et d'un outil d'acquisition de connaissances experts, Thèse de l'Université Paul Sabatier de Toulouse, 1989.

[Bazin, 93] Bazin, J.M. - GEOMUS: un système informatique de résolution de problèmes qui s'inspire du comportement humain, *Journées ACTI'93,* France, 1993.

[Boose & Bradshaw, 93] Boose, J.H., Bradshaw, J.M. - Expertise Transfer and Complex Problems: using AQUINAS as a Knowledge-Acquisition Workbench for Knowledge-Based Systems, *Readings in Knowledge Acquisition and Learning*, pp. 240-252, Edited by B.G. Buchnan & D.C Wilkins, Morgan Kaufmann Publishers, 1993.

[Breuker, 94] Breuker, J. - Components of Problems Solving and types of problems, *Proceedings of 8th European Workshop, pp.118-137, EKAW '94*, Belgium, 1994.

[Brown & Chandrasekaran, 86] Brown, D. and Chandrasekaran, B. - Knowledge and control for a mechanical design expert system, *IEEE-Computer*, 19(7), 1986.

[Bylander & Chandrasekaran, 87] Bylander, T. and Chandrasekaran B.- Generic tasks for knowledge-based reasoning: the "right" level of abstraction fir knowledge acquisition, *Int. J. Man-machine Studies*, pp.231-243, 1987.

[Bylander, 86] Bylander T. "CSRL: A language for Classificatory Problem Solving and Uncertainty Handiling", *AI Magazine, vol. 7*, no. 3, pp. 66-77, 1986.

[Canãmero & Geldof, 93] Canãmero, D., Geldof, - Modélisation et validation dans COMMET: une intégration des deux perspectives, *Actes de Deuxièmes Journées d'Acquisition des Connaissances (JAC'93),* France, 1993.

[Causse, 94] Causse-Gobinet, K - MCC (Modèle Conceptuel du Cas) vers l'acquisition de connaissances de contrôle, Thèse de l'Université Paris-Sud, 1994.

[Causse et al., 92] Causse-Gobinet, K; Canãmero, D.; Gobinet, P. - La généricité en acquisition des connaissances, *Actes de Troisièmes Journées d'Acquisition des Connaissances (JAC'92),* France, 1992.

[Chan, 92] Chan, T.W. - Curriculum tree: A knowledge-based architecture for intelligent tutoring systems, Frasson, C., Gautier G., McCalla, G.I. (Eds.), *Intelligents TutoringSsystems (ITS'92),* pp. 140-147, Springer-Verlag, Montreal, 1992.

[Chandrasekaran & Johnson, 93] Chandrasekaran B.& Johnson T. - Generic Task and Task Structures: History, Critique and New directions, *Second Generation Expert Systems*, Springer-Verlag, pp. 232-272, 1993.

[Chandrasekaran, 87] Chandrasekaran B. - Towards a Fonctional Architecture for Intelligence Based on Generic Information Processing Tasks, *Proceedings of the Tenth International Joint Confwerence on Artificial Intelligence*, pp. 1183-1192, Milan, 1987.

[Chandrasekaran, 86] Chandrasekaran B. -Generic Tasks in Knowledge-Based reasoning : High-Level Building Blocks for Expert system Design, *Revue IEEE Expert*, 1986.

[Clancey, 89] Clancey, W. - The Knowledge Level Reinterpreted: Modeling how Systems Interact, *Machine Learning*, special Issue on Knowledge Acquisition, 4(3,4), pp. 285-292, 1989.

[Clancey, 85] Clancey, W. - Heuristic Classification, *Artificial Intelligence*, vol. 27, pp. 289-350, 1985.

[Courouble, 83] Courouble, T. - Le rôle des pédagogues dans le développement de l'EAO en formation Professionnelle, Education Permanente no. 70/71, 1983.

[Davis & Lenat, 82] Davis, R. & Lenat, B. - Knowledge Based System in Artificial intelligence, Mcgraw Hill, 1982.

[Diaper, 89] Diaper, D. - Knowledge Elicitation: principles, techniques and applications, England, 1989.

[Dieng, 93] Dieng, R. - Méthodes et outils d'acquisition des connaissances, *L'ergonomie dans la conception des projets informatiques*, Ed. OCTARES, 1993.

[Djamen, 95] Djamen, J.Y. - Une architecture de STI pour l'analyse du raisonnement d'un apprenant, Thèse de l'Université de Montréal, 1995.

[Dugerdil, 89] Dugerdil, P. Smalltalk-80, Programmation par objets, 1989.

[Drummond & Tate, 90] Drummond, M. & Tate, A. - AI Planning, *Knowledge Engineering, vol. 1, Fundamentals*, H. Addeli Ed., McGraw-Hill Publishing Company, 1990.

[Duribreux & Houriez, 90] Duribreux, M. & Houriez, B. - Application industrielle d'une approche mixte de modélisation des connaissances, *Journées acquisition, apprentissage, JAC'96, Sète 8, 9 et 10*, 1996.

[Ermine, 93] Ermine, J.-L. - Génie Logiciel & Génie Cognitif pour les systèmes à base de connaissances, volume 1 & volume 2, Paris, 1993.

[Eshelman et al., 93] Eshelman, L. ; Ehret, D. ; McDermott, J. ; and Tan, M - MOLE : A tenacious knowledge-acquisition tool, *Readings in Knowledge Acquisition and Learning*, pp. 253-261, Edited by B.G. Buchnan & D.C Wilkins, Morgan Kaufmann Publishers, 1993.

[Ferbert, 94] Ferbert, J. - Des systèmes multi-agents pour simuler le vivant, Actes du 2e colloques african sur la recherche en informatique, Eds. ORSTOM, 1994.

[Firlej & Hellens, 91] Firlej, M. & Hellens, D. - Knowledge Elicitation: a Practical Book, Prentice Hall International (UK) Ltd, 1991.

[Frasson & Gauthier, 90] Frasson, C., Gauthier, G. (Eds.) Intelligent Tutoring Systems, Ablex Publishing Corporation, New Jersey, 1990.

[Fuentes et al., 94] Fuentes, R., Gaudillere, P., Regnier, M., Pierret-Golbreich, C. - Modélisation d'une application juridique selon KADS à l'aide de l'outil OPEN KADS, *Actes de Troisièmes Journées d'Acquisition des Connaissances (JAC'94),* pp. J1-J13, France, 1994.

[Gagné, 85] Gagné, E. D. The cognitive psychology of school learning, Edited by Little, Brown and Company, Boston, 1985.

[Gaines, 93] Gaines, B.R. Modelling and Extending Expertise, *Proceedings of 7th European Workshop, pp.1-23,EKAW '93*, Toulouse and Caylus, 1993.

[Galloüin, 88] Gallouïn, J.F. - Transfert de Connaissances, Systèmes Experts : Téchniques et Mèthodes, Edition Eyrolles, 1988.

[Gecsei & Frasson, 94] Gecsei, J., Frasson, C., - SAFARI: an Environment for Creating Tutoring Systems in Industrial Training, EdMedia, *World Conference on Educational Multimedia and Hypermedia*, Vancouver, 1994.

[Goldberg & Robson, 89] Goldberg, A., Robson, D., Smalltalk-80, The Language, Addison-Wesly, 1989

[Gruber, 89] Gruber, Th.R. - The Acquisition of Strategic Knowledge, *Perspectives in Artificial Intelligence - volume 4,* Stanford University, California, 1989.

[Halff, 88] Halff, H. - Curriculum and instruction in ITSs, *Fondations of intelligent tutoring systems*, pp. 19-108, Hillsdale, 1988.

[Hayes-Roth et al., 83] Hayes-Roth, F., Waterman, D.A., Lenat, D.E. - An Overview of Expert Systems, *Building Expert systems*, Eds. Hayes-Roth, F., Waterman, D.A., Lenat, D.E., 1983.

[Hickman et al., 89] Hickman, F.R., Killing, J.L., Land, L., Mulhall, T., porter, D., Taylor, R.M. - Analysis for Knowledge-Based Systems: a practical guide to the KADS methodology, Ellis Horwood, 1989.

[Hoc, 87] Hoc, J. M - Psychologie cognitive de la planification, Ed. PUF Paris, 1987.

[Jacob & Jehl, 96] Jacob, I. &Jehl, O. - LISA-runtime: vers une utilisation industrielle de langage LISA, *Journées acquisition, apprentissage (JAC'96), Sète 8, 9 et 10*, 1996.

[Kahn, 88] Kahn, G. - MORE: From observing knowledge engineer to automating knowledge acquisition, *Automating Knowledge Acquisition for Experts Systems*, Kluwer Academic Publishers, 1988.

[Karbach et al., 90] Karbach, W., Linster, M. & Voss, A. - Models, methods, roles and tasks: many labels - one idea?, *Knowledge Acquisition no. 2*, pp. 279-299, 1990.

[Klinker, 88] Klinker, G. - Sample-driven knowledge acquisition for reporting systems, *Automating Knowledge Acquisition for Expert Systems*, Kluwer Academic Publishers, 1988.

[Krivine & David, 91] Krivine, J-P. & David, J-M. - L'acquisition des connaissances vue comme un processus de modélisation : méthodes et outils, *Intellectica*, 1991/2 12, pp. 101-137, 1991.

[Laâsri & Maître, 89] Laâsri, H.,Maître, B. Organisation du contrôle dans les architectures de blackboard, *Revue d'Intelligenge Artificielle*, vol. 3, pp. 105-146, 1989.

[Lenat & Feigenbaum, 87] Lenat, D. B. & Feigenbaum, E.A. - On the thresolds of knowledge. *Proceedings of the Tenth International Joint Confwerence on Artificial Intelligence*, pp. 1173-1182, Milan, 1987.

[LeRoux, 94] Le Roux, B. - Éléments d'une approche constructive de la modélisation et de la réutilisation en acquisition des connaissances, Thèse de l'Université Paris VI, 1994.

[LeRoux, 93] Le Roux, B. - Tuteur intelligent pour la modélisation et l'explicitation des connaissances, *Journées ACTI'93*, France, 1993.

[Leroux & Vivet, 96] Leroux, P. & Vivet, M. - De la modélisation d'un processus de coopération à la conception de système coopératifs d'apprentissage, Frasson, C., Gautier G., Lesgold, A. (Eds.), *Intelligents Tutoring Systems (ITS'96)*, pp. 197-205, Springer-Verlag, 1996

[Lesgold & Lajoie, 94] Lesgold, A., Lajoie,S - Appling Cognitive Task Analysis and Reserch Methods to Assessment, University of Pittsburg, 1994.

[Luger & Stubblefield, 93] Luger, G.F., Stubblefield, W.W. - Artificial intelligence and the design of Experts Systems, Benjamin/Cummings, 2e édition, 1993..

[Lupascu & Zalila, 95] Lupascu, D., Zalila, N. - Acquisition des connaissances pour la tâche de diagnostic médical, Rapport technique, Projet SAFARI, Université de Montréal, 1995.

[Major & Reichgelt, 90] Major, N. & Reichgelt, H. - ALTO : An Automated Laddering Tool, *Current Trends in Knowledge Acquisition*, pp.222-237, Amsterdam, 1990

[Marcus & McDermott, 93] Marcus, S & McDermott - SALT : A knowledge Acquisition language for "propose-and-revise" systems, *Readings in Knowledge Acquisition and Learning*, pp. 263-282, Edited by B.G. Buchnan & D.C Wilkins, Morgan Kaufmann Publishers, 1993.

[McDermott, 88] McDermott, J. - Preliminary Steps Toward a Taxinomy Problem-Solving Methods, *Automating Knowledge Acquisition for Expert Systems*, Kluwer Academic Publishers, 1988.

[Mengelle, 95] Mengelle, T. - Étude d'une architecture d'environnement d'apprentissage bases sur le concept de preceptorat avisé, Thèse de l'Université Paul Sabatier de Toulouse, 1995.

[Moinard & Joab, 94] Moinard, C., Joab, M. - Étude de cas: acquisition de connaissances pour le système DIAPASON, *Actes de Troisièmes Journées d'Acquisition des Connaissances (JAC'94)*, pp. M1-M14, France, 1994.

[Murray & Woolf, 93] Tools for teacher participation in ITS design, In Frasson, C., Gautier G., McCDalla, *Intelligents tutoring systems*, pp593-600, Springer-Verlag, 1993.

[Musen, 89a] Musen, M. A. -Automated Generation of Model-Based Knowledge-Acquisition Tools, *Research Notes in A.I.*, London, 1989.

[Musen, 89b] Musen, M. A. - Conceptual models of interactive knoweledge acquisition tools,*Knowldge Acquisition* vol 1 no 1, pp. 59-89, 1989.

[Newell, 82] Newell, S. - The Knowledge Level, *Artificial Intelligence* 18 , pp. 87-127, North-Holland, 1982.

[Nicaud & Vivet, 88] Nicaud, J-F., & Vivet, M., - Les tuteurs Intelligents: réalisations et tendances de recherches, *Revue Tecnique et Sciences de l'Informatique*, no. 1, vol. 7, Dunod/Afcet, 1988.

[Nkambou, 96] Nkambou, R. - Modélisation des connaissances de la matière dans un système tutoriel intelligent: modèles, outils et applications, Thèse de l'Université de Montréal, 1996.

[Nkambou & Gautier, 96] Nkambou, R. Gautier, G.- Un modèle de représentation du curriculum dans un STI, In Frasson, C., Gautier G., Lesgold, A., *Intelligents tutoring systems*, pp 420-429, Springer-Verlag, 1996.

[Offut, 88] Offut, D. - A knowledge acquisition tool specialised for the sizing task, *Automating Knowledge Acquisition for Expert Systems*, Kluwer Academic Publishers, 1988.

[Pachet et al, 95] Pachet F., Djamen J.-Y. et Frasson C., - Production de conseils pertinents exploitant les relations de composition et de précédence dans un graphe de tâches,Raport technique, SAFARI, Université de Montréal, 1995.

[Péninou, 93] Péninou, A., - MACT: un modèle d'agents centrés tâches pour la production de systèmes tuteurs intelligents par l'atelier de génie intégré, Thèse de l'Université Paul Sabatier de Toulouse, 1993.

[Polson & Richardson, 88]. Polson, &Richardson – Fondations of Inteligents Tutoring Systems, LEA Ed., 1988.

[Puerta et al., 91] Puerta, A. ; Egar, J. Tu, S, Musen, M. - A Multiple-Method Knowledge Acquisition Shell for the Automatic Generation of Knowledge-Acquisition Tools, *Proceedings of BANFF'91*, 1991.

[Rademaekers & VanWelkenhuysen, 90] Rademaekers P., & VanWelkenhuysen, J. - Mapping the Knowledge Level into te Computational Level, *Proceedings of ECAI-90*, London, 1990.

[Shaldbolt & Wielinga, 90] Shaldbolt, N. & Wielinga, B.J. - Knowledge-Based Knowledge Acquisition : The next generation of support tools, *Current Trends in Knowledge Acquisition*, pp.313-339, Amsterdam, 1990

[Shadbolt & Burton, 90] Shadbolt, N.&Burton, A.M. - Knowledge Acquisition, In Wilson and Corlett (Eds) Evaluation of human work: Practical ergonomics methodology, 1990.

[Shlaer & Mellor, 92] Shlaer, S. & Mellor, S.J.- Object life cycles: Modeling the World in States, Yourdon Press, Prentice Hall, 1992.

[Schreiber, 93] Schreiber, G. - KADS, a principled approach to knowledge based system developement, *Knowledge Based Systems Academic Press*, 1993.

[Steels, 90] Steels, L. - Components of Expertise, The *AI Magasine* 11(2), 1990.

[Steels, 93] Steels, L. - The componential framework and its role identification reusability,
Seconds Generation Expert Systems, David, J.-M., Krivine, J.-P., & Simmons, R. Eds., Springer-Verlag, Berlin, 1993.

[Stefik, 81] Stefik, M. - Planning with Constraints (MOLGEN: Part1), *Artificial Intelligence*, vol 16, pp.111-140, 1981.

[Tadié, 96] Tadié, S.G. - Coopération et système tutoriel intelligent, Examen prédoctoral (2ème partie), 1996.

[Tardif, 92] Tardif, J. - Pour un enseignement stratégiqueLes Éditions LOGIQUES, Montréal, 1992.

[Tecuci, 96] Tecuci, G., - Is it an Agent, or just a Program ? : A Taxonomy for Autonomous Agents, *Proceedings of the Third International Workshop on Agent Theories*, Architectures and Languages, Springer-Verlag, 1996.

[Thomas & Leroux, 96] Thomas, J., Leroux, B. Cycle de raffinements du biais d'apprentissage: un exemple bassé sur les modèles, *Journées acquisition, apprentissage (JAC'96), Sète 8, 9 et 10*, 1996.

[Van de Velde, 93] Van de Velde, W. - Issues in Knowledge Level Modellling, *Second Generation Expert Systems*, pp. 211-231, David, J.-M., Krivine, J.-P., & Simmons, R. Eds., Springer-Verlag , 1993.

[van Melle et al., 84] van Melle, W. ; Shortliffe, E.H. ; Buchanan, B.G. - EMYCIN: A knowledge engineer's tool for constructing rule-based expert systems, *Rule-Based Expert Systems*, 1984.

[Van Heijst & Schreiber, 94] Van Heijst, G. & Schreiber, G. - CUE : Ontology Based Knowlwdge Acquisition, *Proceedings of 8th European Workshop*, pp.178-200, *EKAW '94*, Belgium, 1994

[VisualWorks., 92] - Visual Works - user's guide, release 2.0, Parc Place Systems, 1992.

[Wielinga et al., 92] Wielinga, B.J, ; Van de Velde, W. ; Schreiber, A.T.;.Akkermans, H. - The common KADS framework for knowledge modelling, Esprit project, 1992.

[Wielinga et al., 93] Wielinga, B.J, ; Schreiber, A.T.; Breuker, J.A. - KADS : A modelling approach to knowledge engineering, *Readings in Knowledge Acquisition and Learning*, pp. 92-117, Edited by B.G. Buchnan & D.C Wilkins, Morgan Kaufmann Publishers, 1993.

[Wielinga & Breuker, 86] Wielinga, B.J., & Breuker, J.A. - Models of Expertise, *European Conference of Artificial Intelligence, ECAI-86,* 1986.

[Willamowski, 94] Willamowski, J. - Modélisation de tâches pour rèsolution de problèmes en coopération système-utilisateur, Thése Université Joseph Fourier - Grenoble 1, 1994.

[Winograd & Flores, 89] Winograd, T., Flores, F. - L'intelligence artificielle en question, Presses Universitaires de France, 1989.

[Wooldridge & Jeannings, 95] Wooldridge, M.,. Jennings, N.R. - Agent Theories, Architectures and Languages: a Survey, *Intelligent Agents*, Wooldridge and. Jennings Eds., Springer-Verlag, p 1-22, Berlin, 1995.

[Zacklad, 93a] Zacklad, M. - Principes de modélisation qualitative pour l'aide à la décision dans les organisations, Méthode d'utilisation du logiciel d'acquisition des connaissances C-KAT., Thèse de l'Université de Thechnologie de Compiègne, 1993.

[Zacklad, 93b] Zacklad, M. - Les systèmes à base de connaissances vus comme des systèmes interactifs d'aide à la résolution de problèmes, *quatrièmes Journées Acquisition des Connaissances (JAC 93)*, Saint-Raphaël, France, 1993.