Université de Montréal

Un système multi-agent pour l'enseignement et la simulation de tâches coopératives

Par

Tadié Guepfu Serge

Département d'informatique et de recherche opérationnelle

Faculté des arts et des sciences

Thèse présenté à la Faculté des études supérieures

En vue de l'obtention du grade de

Philosophiae Doctor (Ph.D.)

En Informatique

Juin 1998

Université de Montréal

Faculté des études supérieures

Cette thèse intitulée :

Un système multi-agent pour l'enseignement et la simulation de tâches coopératives

Présentée par :

Tadié Guepfu Serge

a été évaluée par un jury composé des personnes suivantes :

Michel Boyer	Président-rapporteur
Rudolf K. Keller	Membre du Jury
Alain Derycke	Examinateur externe
Claude Frasson	Directeur de recherche
Bernard Lefebvre	Codirecteur

Thèse accepté le : 12 novembre 1998

À mon Père Guepfu François et à ma mère Kembou Élise.

Pour tout l'amour que je vous porte.

Remerciements

Je remercie mon Directeur, le professeur Claude Frasson (Université de Montréal) pour avoir cru en moi et de m'avoir donné l'opportunité de faire une thèse au sein de son laboratoire. Il m'a offert un cadre académique et financier exceptionnel pour mener à bien mes recherches. Ses conseils permanents et pertinents ont été d'un grand support à la conduite de cette thèse.

Je remercie tout particulièrement mon codirecteur le professeur Bernard Lefebvre (Université du Québec à Montréal) pour avoir accepté de co-diriger mes travaux. Les nombreuses séances de travail que j'ai eues avec lui ont permis de bien fixer les idées et les résultats forts de cette thèse. De plus le travail coopératif assisté par ordinateur que nous avons effectué à distance durant son séjour en France a été un facteur important à la qualité de la rédaction de cette thèse.

Je remercie le professeur Alain Derycke (Université des Sciences et Technologies de Lille) d'avoir accepté d'être l'examinateur externe de ma thèse. Son expertise dans le domaine a permis d'éclaircir un peu plus quelques aspects novateurs de cette thèse.

Je remercie les professeurs Michel Boyer et Rudolph Keller (Université de Montréal) qui ont accepté respectivement d'être Président et membre de mon jury de thèse. Le temps qu'ils ont investi à la lecture de cette thèse a permis d'affiner la formalisation de certains aspects.

Je remercie J-Y Rossignol qui dans le cadre de sa maîtrise à implanté quelques aspects de cette thèse. Je remercie également Martine Gemme pour sa sollicitude. Elle a œuvré pour que mon séjour à l'université soit un délice d'un point de vue administratif.

Je remercie tous les membres des projets SAFARI et TELELEARNING et du laboratoire HERON piloté par les professeurs Claude Frasson, Jan Gecsei, Gilles Gauthier, Bernard Lefebvre, Marc Kaltenbach, Esma Aimeur avec les membres suivants J-Y Djamen, T Mengelle, R Nkambou, A Kengné, D, Phalp, C Alexe, R Nadjib, R Khalid, A Mghayar, D Lupascu, S Reina, T.H. Le, L Duperval, A Seffah, A Serroud, J-Y Rossignol, F

Demers, L Martin, C Dufour, H Dufort avec qui j'ai passé de merveilleux moment que ce soit du point de vue personnel ou académique.

Je remercie mes amis Roger Nkambou, J-Y Djamen, Celestine Nkambou, Brigitte Chatué, Karine Coudert, Sabina Reina, Daniéla Lupascu, Thiérry Mengelle, Virginie Nanhou, dont l'amitié a fait de ces dures années de recherche des moments agréables.

Je remercie mon frère Job Kenmoé, qui, géographiquement le plus proche de moi, a su m'insuffler tout l'amour familial nécessaire pour garder ma motivation intacte. Je remercie également mon cousin le docteur Ernest Simo, qui, m'a beaucoup motivé par son exemple et ses conseils. Je remercie mon ami Capitant qui malgré la distance qui nous sépare a su m'encourager sans cesse.

Je remercie tous mes frères et sœurs qui m'ont toujours tous porté dans leur cœur, et dont le soutien a été d'une aide précieuse.

Je remercie Mon père François Guepfu et ma mère Élise Kembou pour les sacrifices qu'ils ont consentis depuis ma plus tendre enfance jusqu'aujourd'hui. L'aboutissement de cette thèse est pour moi le fruit de leurs efforts.

Je remercie enfin Alice Nanhou pour l'amour qu'elle me porte. Sa présence a été un catalyseur important pour la bonne finition de cette thèse.

Je remercie aussi tous ceux qui de près ou de loin ont participé à la réalisation de cette thèse et que j'ai oublié de mentionner ici.

Sommaire

L'enseignement à l'aide des ordinateurs fait partie de la recherche informatique depuis plus d'une quarantaine d'années. Jusqu'aujourd'hui, les systèmes tutoriels intelligents (STI) se sont focalisés sur l'enseignement de connaissances individuelles comme des faits, des concepts ou des tâches individuelles. Pourtant dans le quotidien, l'homme est confronté à la résolution de tâches coopératives.

Cette thèse se focalise dans la construction d'un STI capable d'enseigner les tâches coopératives. Enseigner les tâches coopératives c'est d'une part veiller à la maîtrise des compétences individuelles et d'autre part s'assurer de la coordination de l'équipe chargée de réaliser la tâche. Le STI produit doit être capable fournir ces deux types d'enseignement même si tous les postes de travail nécessaires pour la réalisation de la tâche ne sont pas occupés. Pour cela, le système doit simuler intelligemment le rôle des postes non occupés.

Cette thèse propose donc un modèle de tâche coopérative exploitable dans un environnement d'enseignement et une architecture de STI permettant d'exploiter ce modèle. Le modèle de tâche est organisé sous forme d'arbre et de couches intégrant les règles de la logique des prédicats. La tâche ainsi construite est hiérarchisée, distribuée et dynamique. Le STI est construit sous forme d'architecture multi-agents, et fonctionne aussi bien en mode enseignement qu'en mode simulation de tâche.

Le système que nous avons mis en place, nous a permis de générer des explications complexes par coopération entre agents. De plus nous avons conçu un environnement graphique d'édition de tâche coopérative qui intègre des outils de test des tâches construites.

Cette thèse nous a permis :

- De proposer un modèle de tâche coopérative contenant toutes les informations utiles pour la génération automatique d'explications.
- De proposer une architecture multi-agents de STI permettant d'enseigner et de simuler des tâches coopératives.

Ces deux réalisations permettent au STI de pénétrer les domaines de travail en équipe comme le pilotage d'avions ou de navettes, la gestion d'organisations complexes, etc.

Mots clés : Système tutoriel intelligent, modélisation de tâches coopératives, enseignement de tâches coopérative, architecture de STI, agent.

Table des matières

Table des figures

Liste des tableaux

1 Introduction

> *Stop worrying. The computer revolution has not passed you by. In fact, quite the opposite is the case. The computer revolution hasn't caught up to where you are. If you can't use today's computers without pain, then just wait. It is the computers that will have to change, not you.*
>
> Roger C. Schank, 1984, pp 3.

L'avènement de l'informatique dans les années 1950 marque un tournant dans l'évolution industrielle de notre société. Dans ses débuts, l'homme est obligé de s'adapter au mode de fonctionnement des ordinateurs. Cette adaptation se situe à plusieurs niveaux.

Le langage de programmation. La programmation se fait principalement par l'intermédiaire de séries de bits et un peu plus tard de l'utilisation du langage assembleur.

Le langage d'interaction entre l'ordinateur et l'homme. L'interface est uniquement textuelle.

Le manque de sens de l'ordinateur. Les ordinateurs sont incapables de voir, d'entendre ou de reconnaître des formes.

Le manque de raisonnement. L'ordinateur est un gros calculateur, qui prend des données en entrée et produit un résultat.

Le caractère individualiste de l'ordinateur. Chaque ordinateur est fermé sur lui-même, et permet à un humain de se connecter pour y faire des opérations.

Jusque dans les années 1980, la plupart des applications informatiques ont souffert de ces limites technologiques. Il en est ainsi des systèmes tutoriels intelligents (S.T.I.).

Les STI sont des logiciels qui ont pour but de promouvoir un enseignement ou un apprentissage [Frasson 91]. Ce domaine a vu le jour dans les années 1960 et s'appelait Enseignement Assisté par Ordinateur (E.A.O.). Les limitations de ces systèmes étaient principalement la prise en compte d'un apprenant type, l'incapacité de mettre en place des stratégies d'enseignement et l'incapacité d'analyser le raisonnement de l'apprenant.

La recherche en informatique s'est donnée pour mission de faire évoluer l'ordinateur afin de le rendre plus proche de l'homme. C'est ainsi que les langages de programmation se sont rapprochés du langage de l'homme. On est passé du langage machine aux langages de quatrième génération, en passant par les langages procéduraux et les langages orientés objet. Au niveau de l'interaction entre l'homme et l'ordinateur, il y a également eu des avancées notables telles que l'introduction des interfaces graphiques, vocales et plus généralement la communication est devenue multimédiatique. En ce qui concerne le raisonnement, les recherches initiées par McCarthy, Minsky, Shannon, Newell et Simon ont permis de voir l'ordinateur comme un système de traitement de l'information plutôt qu'un calculateur. C'est ainsi que l'intelligence Artificielle (IA) est née. Elle permet, grâce à des structures de connaissances complexes, de raisonner. On a ainsi pu développer le raisonnement logique (logique classique et modale), le raisonnement à base de cas, le raisonnement à partir des réseaux de neurones, etc. Dans la dimension sociale, l'émergence des réseaux d'ordinateurs a permis la naissance de disciplines comme le CSCW(Computer Supported Cooperative Work) qui permet la construction d'outils de travail coopératif [Derycke & Hoogstoel 95]. Cette discipline fait évoluer la recherche sur trois axes principaux :

- l'axe social, dans ce cas le travail coopératif étudie les aspects socio-technique ;
- l'axe cognition, dans ce cas, la discipline se penche sur l'étude de la connaissance distribuée et sa production par un ensemble d'agent;
- l'axe de la communication, dans ce cas, les recherches s'orientent vers le développement d'outils de travail coopératif servant de milieu de médiation entre agents.

Au niveau social, deux notions émergent : l'interdépendance dans le travail ce qui est un corollaire du fait que le travail dans la société est essentiellement social, et l'organisation du travail en terme d'acteurs, de responsabilités, de tâches, d'activités et de ressources [Derycke & Hoogstoel 95]. Cette analyse des systèmes de CSCW fait ressortir la nécessité de réactivité et d'auto-organisation. Elle fait également ressortir la nécessité de la distribution du travail aux participants. Au niveau cognitif, cette émergence a également permis l'intégration des concepts de l'intelligence artificielle distribuée et des systèmes multi-agents. L'intelligence artificielle distribuée (IAD) [Bond & Gasser 88] a pour but de remédier aux insuffisances de l'intelligence artificielle en ce qui concerne l'intelligence et les connaissances distribuée. Elle propose la distribution des connaissances à un groupe d'agent et l'intégration des mécanismes de raisonnement

coopératif pour l'exploitation de ces connaissances. Au niveau communication, la robustesse des infrastructures est un élément important, car c'est d'elles que dépendent le succès des applications coopératives. Il faut donc concevoir des stratégies de communication qui tiennent compte des limites des infrastructures et du réseau de communication.

L'évolution de la technologie informatique couplée à l'évolution dans les technologies de l'éducation a entraîné une évolution dans les STI. Au niveau de l'éducation, des théories d'enseignement ont été développées. Il s'agit principalement des connaissances à acquérir et des stratégies d'enseignement de ces connaissances [Bloom 59/69], [Gagné 79]. La recherche en psychologie cognitive a permis de comprendre mieux l'apprenant et de pouvoir produire un modèle permettant de simuler ses connaissances et son comportement. L'informatique, à travers les concepts développés en intelligence artificielle, a permis d'adapter les résultats des recherches en éducation et en psychologie cognitive. Malgré tous ces développements les STI ne répondent pas encore à toutes les préoccupations de l'homme. Parmi ceux construits à ce jour, SCHOLAR [Carbonnel, 70], SOPHIE [Brown, Burton & Bell, 75], GUIDON [Clancey 87] et SAFARI [Frasson 97] les connaissances enseignées ont toujours eu un caractère individualiste. Ces connaissances ne nécessitent pas la présence de plusieurs humains dans le processus d'apprentissage ou dans sa réalisation effective. Or le travail ou les tâches quotidiennes dans la société font intervenir très souvent les contraintes d'interdépendance. Alors que la connaissance en général peut être abstraite et individuelle, le travail lui est essentiellement social c'est à dire qu'il fait intervenir une coopération entre plusieurs personnes.

L'homme est un être social. Il fonctionne principalement en coopération et est amené à interagir avec ses pairs pour résoudre les problèmes qui se posent dans la société. La problématique d'enseignement ou d'apprentissage de connaissances coopératives (problèmes ou tâches résolus en coopération par plusieurs personnes) est donc capitale dans le processus d'instruction de l'homme. Intégrer ce concept dans les STI revient à étudier l'enseignement de tâches coopératives par un système automatique d'enseignement. Ceci définit la problématique de nos travaux.

Les connaissances coopératives sont des tâches qui pour être réalisées ont besoin de la présence de plusieurs agents. Penser à l'enseignement de tâches coopératives c'est construire un environnement où plusieurs personnes peuvent se connecter sur un réseau d'ordinateur et apprendre comment réaliser une tâche de cette nature. Lorsqu'on veut enseigner une tâche coopérative, il faut enseigner les techniques individuelles au niveau

des différents postes de travail et ensuite la technique collective qui permet la coordination des membres de l'équipe chargée de réaliser la tâche.

Le système à construire doit être capable de fournir à tout apprenant un tuteur qui va le guider dans les activités du poste de travail qui l'intéresse au niveau de la tâche coopérative. Ce tuteur représente donc une partie du système d'enseignement de tâche coopérative. Le système doit également permettre la résolution de la tâche coopérative par une équipe d'agents informatiques. Cette résolution peut servir de démonstration à une équipe d'apprenants. L'équipe chargée de résoudre une tâche peut être composée d'apprenants humains et d'agents simulés ce qui permet de recevoir les enseignements à un poste de travail même si le nombre d'apprenants disponibles ne couvre pas l'ensemble des postes.

Ces contraintes nous permettent d'envisager trois types d'environnements devant cohabiter dans notre système. Chacun d'eux met en exergue des problèmes particuliers.

Un environnement permettant à un ensemble d'agents de participer à un travail coopératif à travers un réseau d'ordinateur. Les problèmes rencontrés à ce niveau sont ceux relatifs au domaine du CSCW. Il s'agit essentiellement des problèmes liés à l'espace de communication, à l'espace de coordination et à l'espace de production [Derycke & Hoogstoel 95]. Pour résoudre ces problèmes, nous adapterons les résultats des travaux du domaine. Ceci consistera à prendre en compte la nature de l'infrastructure de communication que nous utiliserons (Internet) et les contraintes liées au fait que nous sommes dans une situation d'enseignement.

Un environnement multi-agents composé aussi bien d'agents logiciels que d'agents humains et chargé de réaliser une tâche coopérative. Cet environnement fait intervenir aussi bien les problèmes de coopération entre agents logiciels (on parle d'interaction multi-agents) et ceux existant entre les agents humains et les agents logiciels (dans ce cas on parle d'interaction humain-machine). Les recherches en IAD permettent actuellement de distribuer l'expertise sur un groupe d'agents. Cette branche se définit comme la modélisation de comportements intelligents par la coopération entre un ensemble d'agents [Huhns 87]. Les principaux problèmes que nous avons à résoudre ici sont de faire coopérer un ensemble d'agents dotés d'un comportement intelligent, de coordonner leurs buts et leurs plans d'actions pour la résolution d'un problème et de diviser un problème particulier sur un ensemble d'entités coopérantes et distribuées.

Un environnement multi-agents composé aussi bien d'agents logiciels qu'humains et chargé d'enseigner à des apprenants humains comment réaliser une tâche coopérative.

Dans ce contexte, en plus des problèmes liés aux environnements précédents, nous devons également permettre la génération des explications pour un apprenant qui aurait des problèmes. Dans ce cas la modélisation des connaissances manipulées par les agents participants au système doit être structurée de telle sorte que les mécanismes de raisonnements sur les actions des agents puissent être implantés.

1.1 Objectif de la thèse

Notre objectif est de proposer un système qui permette l'enseignement de tâches coopératives. Enseigner une tâche coopérative revient d'une part enseigner les concepts à maîtriser au niveau de chaque poste de travail et d'autre part à enseigner la coordination des activités d'une équipe mixte d'apprenants (réels ou simulés) afin de lui permettre de réaliser la tâche coopérative. Dans la mesure ou le premier volet concerne les STI (Systèmes Tutoriel Intelligent) classiques, nous allons focaliser nos recherches sur l'enseignement ou l'apprentissage de la coordination des activités d'une tâche coopérative. Il nous faut donc trouver un modèle de représentation de tâches coopératives exploitable dans un environnement d'enseignement. Nous devons également construire l'environnement d'enseignement exploitant le modèle de tâche que nous avons proposé et permettant de promouvoir l'enseignement et l'apprentissage d'une tâche coopérative.

Les modèles de tâches coopératives existants sont construits pour permettre la résolution automatique des problèmes dans les systèmes multi-agents. Leur principal défaut est qu'ils ne contiennent pas les informations permettant de générer les explications à fournir à un apprenant qui voudrait comprendre le fonctionnement de cette tâche. Le modèle de tâche que nous allons proposer permet de conserver toute la structure cognitive de la tâche. Il permet également de modéliser aussi bien les tâches conceptuelles que les tâches qui ont pour objectif de manipuler un système.

Le système d'enseignement exploitant le modèle de tâche doit être un système distribué permettant une coopération synchrone entre d'une part les apprenants travaillant sur une tâche coopérative et d'autre part les tuteurs chargés de superviser leur travail. Lorsque le système est utilisé en situation d'apprentissage, c'est à dire sans contraintes de temps ni de supervision, les communications pourront se faire de manière asynchrone. Ce système implante les concepts développés dans le domaine du travail coopératif supporté par

l'ordinateur (CSCW). Ceux-ci permettent à plusieurs apprenants connectés à travers un réseau d'ordinateurs de suivre un enseignement. Il exploite aussi les stratégies développées dans les environnements multi-agents. Ainsi il devra résoudre les problèmes liés à la résolution de problème en coopération par un ensemble d'agents logiciel, et ceux liés à la division et la distribution des compétences aux différents agents. Il s'agit principalement de l'exploitation d'agents intelligents pour la réalisation d'une tâche coopérative. Enfin le système que nous devons mettre en place doit intégrer à un environnement coopératif les techniques utilisées dans le cas des STI (générations d'explication contextuelle à la demande) . Les explications pourront ainsi être générées par coopération entre agents.

1.2 Plan de la thèse

Notre objectif étant de proposer un système permettant d'enseigner les tâches coopératives, il fait référence à deux domaines majeurs de l'informatique : la coopération et les systèmes tutoriels intelligents. Après le premier chapitre dédié à l'introduction, nous analysons les STI dans le second chapitre. Cette analyse chronologique nous amène à considérer l'intégration de différentes formes de coopérations dans les STI. Cette analyse nous permet de constater que malgré la nature essentiellement sociale de l'homme il n'existe pas de travaux sur les STI permettant l'enseignement de tâches coopérative. Le système que nous nous proposons de mettre en place fait intervenir plusieurs dimension du concept de coopération tel que les outils de support au travail coopératif, la résolution coopérative de problème, la coopération humain-machine, etc. Dans le but de bien comprendre ces différents phénomènes, nous allons dans le troisième chapitre faire une analyse de la coopération et de tous les systèmes coopératifs. Cette analyse nous amène à proposer une taxonomie de la coopération basée sur le type des agents intervenant dans la coopération.

Le quatrième chapitre présente MONACO_T, le modèle de tâche coopérative que nous proposons. Ce modèle est adapté aussi bien à l'enseignement qu'à la résolution d'une tâche coopérative par une équipe d'agents informatiques. Le chapitre cinq décrit EDER_TC, l'environnement distribué qui permet d'exploiter une tâche à la norme

MONACO_T pour réaliser un enseignement de tâche coopérative ou pour permettre à des agents de réaliser une tâche coopérative. Cette architecture est multi-agents et utilise trois types d'agents : les agents informatiques ou humains pour la simulation et les agents tuteurs.

Pour valider l'exploitation du couple (MONACO_T, EDER_TC), nous proposons dans le chapitre six un exemple de tâche coopérative modélisée à l'aide de MONACO_T et exploitée dans EDER_TC aussi bien en mode enseignement qu'en mode réalisation.

2 Les systèmes d'enseignement par ordinateur

> *Over the years, I have become increasingly disenchanted with the value of CAI programs. The problem is that most of our current attempts to use computers for instruction are too simplistic to have significant effects on learning. We need much more sophisticated instructional software to really help people learn via computers. More specifically, we need to be able to incorporate the kind of teaching strategies and subject matter knowledge possessed by good teachers into our programs.*
>
> Greg P. Kearsley, 87

2.1 Introduction

Les systèmes tutoriels intelligents (STI) sont la continuité des systèmes d'enseignement assisté par ordinateur (EAO). Parler de l'évolution des STI, c'est montrer comment on est parti de l'EAO pour aboutir aux STI actuels. On ne peut parler d'EAO ou de STI sans définir ce que sont l'enseignement et l'apprentissage. En commençant la description de cette évolution nous définirons donc ces notions. Nous poursuivrons en décrivant l'EAO et ses limites. Nous analyserons ensuite les STI tels qu'ils sont vus aujourd'hui et les limites qu'ils ont par rapport à la coopération. Une conclusion critique viendra mettre un terme à cette section.

2.2 Apprentissage et enseignement

On a tendance très souvent à confondre les termes apprentissage et enseignement. Nous allons définir ces deux concepts dans les paragraphes suivants. Ces définitions nous permettrons plus tard de situer notre système par rapport à ces deux processus.

2.2.1 Apprentissage

Le Larousse définit l'apprentissage au sens de la psychologie comme "un ensemble de méthodes permettant d'établir chez les êtres vivants des connexions entre certains stimuli et certaines réponses, et dont le résultat est d'augmenter l'adaptation de l'être vivant à son milieu". Cette définition quoique assez claire reste très générale. La définition que nous adaptons est très précise et provient du domaine de l'éducation [Legendre, 93]. L'apprentissage est donc "le processus d'acquisition ou de changement, dynamique à une personne, laquelle mue par le désir et la volonté de développement, construit de nouvelles représentations explicatives cohérentes et durables de son réel à partir de la perception de matériaux, de stimulation de son environnement, de l'interaction entre les données internes et externes au sujet et d'une prise de conscience personnelle."

2.2.2 Enseignement

La définition de l'enseignement que nous donnons provient également du domaine de l'éducation et s'exprime en fonction de l'apprentissage. Dans son dictionnaire de l'éducation Legendre [Legendre, 93] définit l'enseignement comme "un processus de communication en vue de susciter l'apprentissage". Une autre définition qui met en exergue le rôle de l'enseignant ou du pédagogue considère l'enseignement comme "un ensemble d'actes de communications et de prises de décision mis en œuvre intentionnellement par une personne ou un groupe de personnes qui interagit en tant qu'agent dans une situation pédagogique".

2.3 Les différentes théories de l'apprentissage

Les théories de l'apprentissage ou théories pédagogiques sont des théories mises en place par les chercheurs de l'éducation et les psychologues dans le but de trouver une meilleure harmonisation de l'ensemble des relations inhérentes à toute situation pédagogique. Il s'agit plus précisément de voir comment organiser les relations entre pédagogue, apprenant, objet de l'apprentissage et l'environnement de l'apprentissage afin que l'apprenant puisse acquérir le maximum de connaissance. Les deux grandes théories sont le cognitivisme et le béhaviorisme.

2.3.1 Le courant cognitiviste

Ce courant dans l'apprentissage vient de la psychologie cognitive qui conçoit la pensée comme un centre de traitement des informations capable de se présenter la réalité et de prendre des décisions. Cette conception vient d'Aristote qui concevait l'esprit humain comme un esprit capable d'organiser et de synthétiser les données perceptibles. Ce mode de pensée a été adopté majoritairement en intelligence artificielle ou on considère les processus cognitifs comme des calculs et du traitement de l'information. Le comportement de l'humain est donc dicté par sa conscience. La pédagogie cognitiviste compte exclusivement sur la conscience de l'apprenant pour favoriser l'apprentissage.

2.3.2 Le courant béhavioriste

Le béhaviorisme pense que le développement de l'humain n'est lié qu'à l'apprentissage. Il postule que l'acquisition de comportement se fait de manière associative. Au niveau de l'éducation, le béhaviorisme pense que l'humain est totalement éducable [Skinner, 76]. À partir de là, Skinner propose une conception de l'éducation où tout apprentissage est défini en terme de comportement observable. Chaque étape menant à l'acquisition d'un comportement élémentaire doit faire l'objet d'un renforcement. L'organisation systématique d'un ensemble de contingence de renforcement permettant un apprentissage constitue un programme. Cette notion est à la base de l'enseignement programmé tel que le définit le béhaviorisme. De tels programmes peuvent être administrés par une machine, Skinner propose à ce propos l'introduction de machine à enseigner. L'importance est donnée ici à l'environnement d'apprentissage.

2.3.3 Les types d'apprentissage ou d'enseignement

Les types d'enseignement font ici référence aux approches pédagogiques utilisées pour transmettre la connaissance aux humains. Plusieurs approches ont été utilisées. Elles prennent leurs sources dans les théories de l'apprentissage existant. Ainsi le béhaviorisme à donné naissance à l'apprentissage collectif, le cognitivisme a inspiré l'enseignement individualisé. L'enseignement coopératif prend en compte aussi bien la théorie du cognitivisme que celle du béhaviorisme.

2.3.3.1 Enseignement collectif

Ce mode d'enseignement représente l'approche pédagogique usuelle dans laquelle l'ensemble des apprenants sans égard à leurs différences individuelles (styles, préalables,

rythmes, etc.), cheminent d'une façon similaire au sein d'un programme d'études [Legendre, 93]. Dans ce type d'enseignement on suppose, comme le préconise les béhavioristes, que les apprenants ont des attributs communs au départ. L'accent est donc mis sur le rôle de l'enseignant qui est perçu comme l'agent principal de la transmission du savoir et du développement d'habiletés chez les apprenants. C'est donc l'environnement de l'apprenant qui est le plus important dans le processus d'apprentissage.

2.3.3.2 Enseignement Individualisé

L'enseignement individualisé est un enseignement qui est conçu en fonction des besoins particuliers de chaque élève. Dans cette stratégie, on vise une grande harmonie entre les trois composantes de l'enseignement que sont l'enseignant, la matière et l'apprenant. Cette stratégie a été utilisée dans ce qu'on a appelé la pédagogie de la réussite. Elle a pour origine l'étude de Bloom [Legendre 93] qui définissant l'aptitude scolaire comme "le temps requis par l'élève pour apprendre une matière donnée à un niveau donné" a démontré que si les apprenants reçoivent un temps uniforme d'enseignement, on peut s'attendre a ce que peu d'entre eux atteignent un niveau adéquat de maîtrise alors que si ces apprenants reçoivent un temps différentiel d'enseignement la grande majorité devrait atteindre ce niveau.

2.3.3.3 Enseignement coopératif

Ce mode d'enseignement est celui où les apprenants cheminent en petits groupes autour d'un même objet d'étude. Dans ces groupes, les apprenants coopèrent à l'accomplissement des tâches d'enseignement. Cette coopération se fait sous la supervision de l'enseignant qui fournit des consignes pour favoriser la coopération et l'interdépendance mutuelle entre les coéquipiers d'une même équipe de travail. Certains chercheurs du domaine de l'éducation en l'occurrence Slavin [83] ont réalisé des recherches qui ont montré l'efficacité de ce mode sur la réussite des apprenants. Cette approche adhère aux thèses de la théorie cognitiviste, mais considère également que l'environnement social de l'apprenant joue un rôle important dans le processus d'acquisition des connaissances.

2.4 L'Enseignement Assisté par Ordinateur

2.4.1 Définition de l'EAO

L'EAO se définit comme le domaine qui étudie l'utilisation interactive de l'ordinateur comme outil pédagogique et moyen d'aide à la formation [Bestougeff, 82]. Cet outil est au centre d'une relation pédagogue (professionnel de l'enseignement), objet de l'enseignement et apprenants. Cette définition considère les systèmes d'EAO comme des systèmes qui ont la capacité de remplacer l'humain dans le processus d'enseignement. Une définition plus modeste [Lefevre, 84] considère l'EAO comme l'utilisation d'un matériel informatique et des logiciels pour assurer tout ou partie d'un processus de formation.

2.4.2 Caractéristique des systèmes d'EAO

Les systèmes d'EAO sont généralement appelés des didacticiels. Ils permettent dans le processus d'enseignement de remplacer l'enseignant humain par l'ordinateur. Ces systèmes mettent en place de véritables programmes d'enseignement présentant une matière découpée en leçons. Les leçons sont à leur tour découpées en scénarios, et chaque scénario contient des scènes pédagogiques présentées sous forme d'information-question-réponse [Courouble, 83]. Ces didacticiels sont principalement construits pour donner un enseignement individualisé dans la mesure ou ils ne prennent en compte qu'un apprenant à la fois.

2.4.3 Limites des systèmes d'EAO

L'utilisation des didacticiels a fait ressortir un ensemble de défauts qui limitait la portée de l'EAO par rapport à l'enseignement classique. Les limites recensées se retrouvent au niveau de la structure des systèmes d'EAO, de l'organisation de la matière de la prise en compte des caractéristiques de chaque apprenant.

2.4.3.1 Limite de la structure des systèmes d'EAO

Les systèmes d'EAO à l'origine n'avaient pas de structuration particulière. Ils étaient formés d'un bloc monolithique de programme. Ce manque de structure fait en sorte que la construction des systèmes d'EAO, leur mise à jour ou leur évolution était très difficile.

2.4.3.2 Limite de l'organisation de la matière

Les connaissances du domaine dans les systèmes d'EAO n'ont pas de structure particulière. Elles sont stockées sous forme textuelle et découpées en leçons, scénarios et scènes. Lors de la phase d'acquisition des connaissances par un apprenant, les textes sont présentés à l'apprenant pour qu'il en prenne connaissance. Dans la phase de performance, le système dispose de textes sous forme de questions à choix multiple. Ce type d'organisation ne permet pas de manipuler les connaissances que l'on doit transmettre aux apprenants.

2.4.3.3 Limite de l'expertise pédagogique

Malgré toutes les stratégies tutorielles développées dans le domaine de l'éducation, les systèmes d'EAO n'implantent aucune des stratégies disponibles. Cette incapacité à raisonner sur les stratégies pédagogiques est due aux techniques utilisées pour la construction de tels systèmes. Elles permettent uniquement de présenter une suite de scènes (texte) et une suite de questions à choix multiple.

2.4.3.4 Limite dans la prise en compte des caractéristiques de chaque apprenant

Malgré le fait que les systèmes d'EAO soient faits pour promouvoir un enseignement individualisé, ils ne sont pas capables de prendre en compte les caractéristiques de l'apprenant ni au niveau cognitif, ni au niveau affectif, ni au niveau conatif. L'enseignement donné est donc le même quelque soit le niveau de l'apprenant. Les systèmes d'EAO ne sont également pas capables d'analyser le raisonnement d'un apprenant afin de lui donner une aide juste et mesurée.

Afin d'améliorer toutes ces limites, la recherche a introduit les techniques d'intelligence artificielle et les théories de l'éducation dans la construction des systèmes d'EAO, ce qui a donné naissance aux STI.

2.5 Les Systèmes Tutoriels Intelligents

2.5.1 Définition des STI

Les STI en tant que discipline, sont au carrefour de trois sciences (Figure 1): l'Informatique, l'Éducation et la Psychologie.

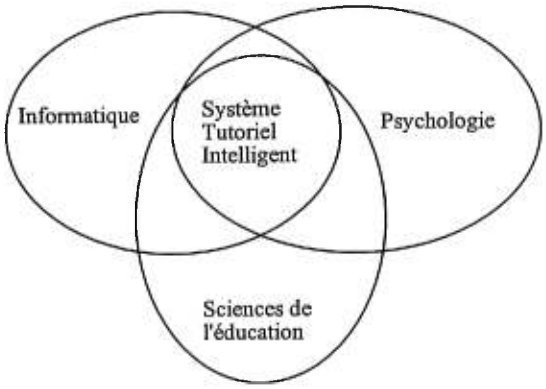


Figure 1 : STI au carrefour de trois sciences

Un STI est défini comme étant un logiciel capable de promouvoir un enseignement ou un apprentissage adapté aux caractéristiques de chaque apprenant [Frasson, 91]. Cette définition est assortie d'une architecture [Polson & Richardson, 88] faisant ressortir quatre grands modules (Figure 2) qui représentent l'expertise du domaine, la modélisation de l'apprenant, les stratégies tutorielles (module pédagogique) et enfin l'interface avec l'apprenant.

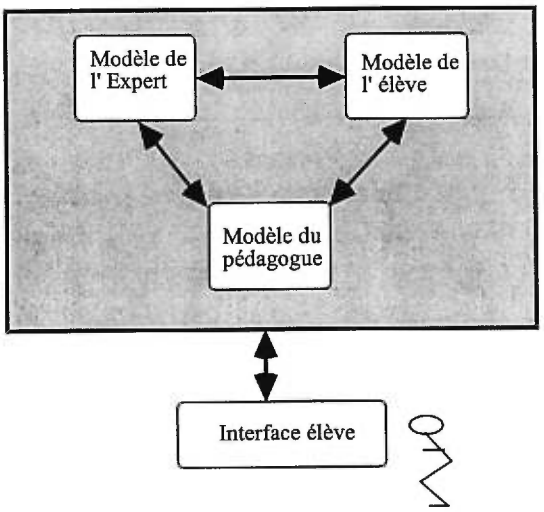


Figure 2 : Architecture de base des STI

Cette architecture de base ne permet pas de comprendre tous les mécanismes qui se mettent en œuvre dans le cadre d'un STI. Pour permettre un meilleur aperçu des STI tels qu'ils fonctionnent aujourd'hui, nous allons adopter une approche systémique d'analyse qui nous permettra de mettre en exergue les relations entre les composants du STI.

2.5.2 Analyse systémique des STI

Cette analyse [Tadié, 96]considère un STI comme un macrosystème formé de mésosystèmes interdépendants. Chaque mésosystème est composé de sous-systèmes reliés entre eux et formant un tout cohérent et orienté vers l'atteinte d'un même objectif qui est de fournir le meilleur apprentissage possible à tout apprenant qui viendrait à utiliser le STI. De par la définition du STI et de son rôle, on se rend compte qu'il met en œuvre deux phases d'apprentissage (Figure 3). Une phase où c'est le STI qui apprend des experts humains et une deuxième phase où il transmet ses connaissances aux apprenants.

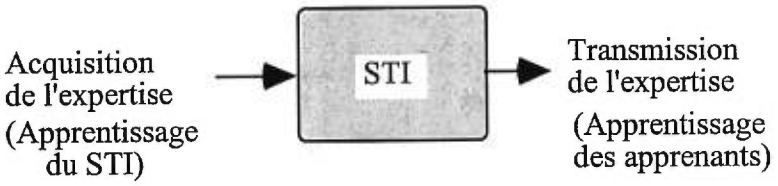


Figure 3 : Les phases d'apprentissage du STI

Dans les STI classiques, l'accent est surtout mis sur la phase de transmission des connaissances à l'apprenant, en négligeant quelque peu celle d'acquisition de connaissances. Nous savons pourtant que l'acquisition des connaissances est la composante principale dans un système de formation utilisant l'ordinateur. Cette acquisition consiste au repérage, à la collecte et à la formalisation des connaissances en vue de leur utilisation par la machine.

Les deux phases que nous avons répertoriées, représentent les deux mésosystèmes de notre STI. Une analyse plus poussée de chacun d'eux (figure 4) nous montre que dans le mésosystème d'acquisition de connaissances, deux sous-systèmes se dégagent:

- l'acquisition des connaissances du domaine,
- l'acquisition des connaissances du pédagogue.

Dans le mésosystème de transmission de connaissance, nous avons également deux sous-systèmes qui sont:

- le sous-système de transmission de connaissances à l'apprenant et
- le sous-système d'acquisition des connaissances de l'apprenant.

Nous allons étudier chaque sous-système dans le détail. Cette étude nous permettra de relever les limites actuelles des STI. Ces limites seront vues principalement du point de vue de la coopération.

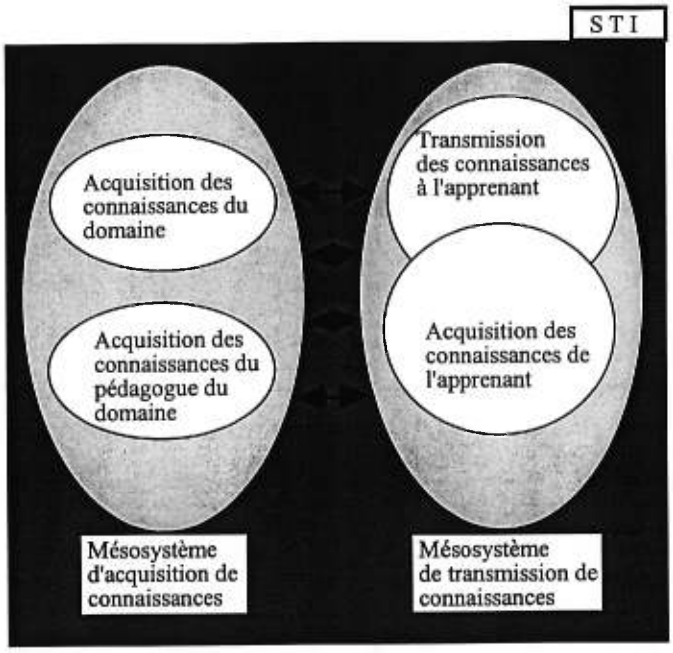


Figure 4 : Représentation du STI sous forme de systèmes

2.5.3 Acquisition des connaissances

2.5.3.1 Organisation des connaissances du domaine

Beaucoup de travaux ont été effectués pour essayer de trouver le formalisme de représentation des connaissances qui faciliterait la tâche aux concepteurs de STI. Ce formalisme devrait leur permettre en particulier d'analyser le raisonnement de l'apprenant, de générer des aides contextuelles et d'implanter des stratégies tutorielles adaptées à chaque apprenant. C'est ainsi qu'on a utilise les réseaux sémantiques dans SCHOLAR [Carbonell, 70], les réseaux de sujets [Woolf, 89], [Murray & Woolf, 93], les règles [Reiser & al., 85; Anderson, 83a], les graphes conceptuels [Kabbaj, 93], les graphes [Goldstein, 82; Clancey, 92]. Anderson dans [Polson & Richardson, 88] regroupe ces divers formalismes en six groupes qui sont les approches: boîtes noires, boîtes de verre, cognitives, connaissances procédurales, connaissances déclaratives et processus qualitatif. Les recherches les plus récentes utilisent le terme curriculum pour représenter les connaissances du domaine. Le domaine est maintenant considéré comme l'ensemble structuré des expériences d'enseignement et d'apprentissage (objectifs de contenu et d'habiletés spécifiques, cheminements ramifiés et règles de progression, matériel didactique, activité d'enseignement et d'apprentissage, relations d'aide, mesures, évaluations et critères de réussite, environnement éducatif, ressources humaines, horaires,

etc.) planifiées et offertes sous la direction d'une institution scolaire en vue d'atteindre des buts éducatifs prédéterminés [Legendre, 93]. L'un des travaux les plus marquants dans le domaine est le modèle CREAM [Nkambou & Ghauthier, 96].

2.5.3.2 Critiques de l'organisation des connaissances du domaine

Tous les STI construits jusqu'à ce jour sont principalement dédiés à l'enseignement dans les domaines ou la matière est individuelle [Carbonell, 70; Clancey, 87; Lesgold & al., 92]. Nous appelons matière individuelle, les matières dont l'apprentissage ne concerne qu'un individu à la fois. Pourtant l'homme est un être social. Il est donc amené à travailler très souvent en coopération avec d'autres humains. Il est donc important que les STI intègrent l'enseignement de matières coopératives.

2.5.3.3 Acquisition des connaissances du domaine

Ce processus permet de construire le curriculum du domaine d'enseignement. L'atelier de construction de curriculum construit dans [Nkambou., 96] en est un exemple. L'outil d'acquisition des connaissances du domaine à enseigner devient ici un atelier de construction de curriculum à cause de la diversité des types de connaissances que l'on a au niveau du curriculum. L'acquisition des connaissances du curriculum implique l'acquisition de plusieurs modèles différents tels que: le modèle des objectifs, le modèle des capacités, le modèle des tâches, le modèle des concepts, etc. L'architecture du sous-système d'acquisition des connaissances du domaine (Figure 5) fait ressortir cinq grands modules:

- le modèle de l'expert en ingénierie des connaissances,
- le modèle de l'expert du domaine,
- les stratégies d'aide à l'acquisition des connaissances,
- le curriculum,
- l'interface d'acquisition des connaissances du pédagogue.

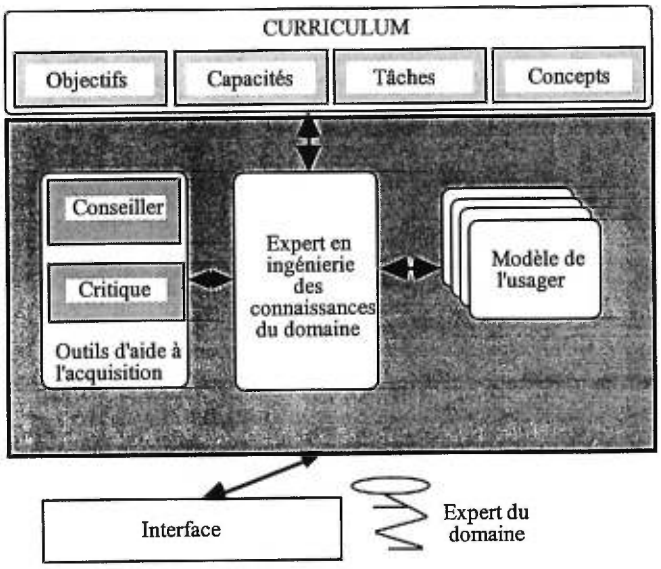


Figure 5 : Architecture du sous-système d'acquisition des connaissances du domaine

a) Le modèle de l'expert en ingénierie des connaissances

Ce module permet de représenter les stratégies d'acquisition de connaissance que l'on connaît dans le domaine de l'ingénierie des connaissances. L'implantation de ces stratégies permettra de transformer les interfaces personne-machine en coopération personne-machine

b) Le modèle de l'usager

Il représente toutes les informations concernant l'expert du domaine. Ces connaissances sont utilisées par le modèle de l'ingénieur des connaissances pour mieux structurer son discours. Cette capacité d'adaptation de l'outil d'acquisition permet de mettre à l'aise les experts humains.

c) Les outils d'aide à l'acquisition des connaissances

Ce sont les différents outils que le cogniticien utilise pour aider les experts humains dans leur tâche de construction du domaine. Les systèmes actuels permettent soit de donner des conseils aux experts humains, soit de critiquer les informations données par les experts. Ces conseils et ces critiques sont généralement basés sur les métaconnaissances du domaine que possède le cogniticien.

d) Curriculum

Il représente la base de connaissances que le système d'acquisition des connaissances du pédagogue du domaine va construire au cours de son entretien avec l'expert humain. Cette base de connaissances peut avoir plusieurs représentations, Nkambou dans [Nkambou & Gauthier, 96] utilise trois réseaux interconnectés. Ce sont les réseaux des objectifs, des capacité et des ressources alors qu'Imbeau dans [Imbeau, 90] le représente à l'aide d'un automate à états fini.

e) Interface d'acquisition des connaissances du domaine

Lorsque l'on veut acquérir les connaissances d'un expert à l'aide de l'ordinateur, deux problèmes majeurs se posent. Le premier est que l'expert n'a pas une structuration claire des connaissances qu'il possède. Le deuxième est la mise en place d'une interface plaisante, facile à utiliser, une interface qui encourage l'expert à dialoguer avec le système.

Pour résoudre le premier problème, le module d'expert en ingénierie des connaissances que nous proposons dans l'architecture du système d'acquisition des connaissances de l'expert doit être chargé de mettre en place toutes les stratégies nécessaires pour extraire et structurer le maximum de connaissances que possède l'expert.

Pour que les stratégies mises en place par le cogniticien puissent être efficaces, il faut que l'expert du domaine prenne le temps de dialoguer avec le système, ce qui entraîne la mise en place d'une interface conviviale. Il faut qu'elle se fasse la plus absente possible car moins l'interface est remarquée, mieux elle est utilisée. Elle se doit donc d'être la plus proche possible de l'interface naturelle de communication de l'homme. Si sa construction fait intervenir deux grandes composantes, le langage et le média, le langage doit alors être proche du langage naturel de l'homme, la gestuelle doit se faire de préférence en trois dimensions, le support de communication doit être tel que l'on ne remarque pas sa présence, et il doit permettre d'utiliser le maximum de sens parmi ceux que possède l'homme.

Les techniques mises en œuvre dans l'interface dépendent du domaine que l'on a choisi pour la formation. L'interface doit respecter les contraintes d'ergonomie.

2.5.3.4 Critiques du processus d'acquisition des connaissances du domaine

Le processus d'acquisition de connaissances est de plus en plus vu comme l'extraction des connaissances d'un expert par le système grâce à un agent jouant le rôle de cogniticien. Cette approche est une première intégration de la coopération dans le processus d'acquisition de connaissances. D'autres formes plus subtiles existent et permettent également d'extraire davantage de connaissances des experts humains. On s'est rendu compte par exemple que les experts en réunion pouvaient produire plus d'information que chaque expert interrogé individuellement. La simulation d'une réunion de travail entre experts artificiels ou réels permettrait donc de galvaniser la production des ces derniers.

2.5.3.5 Acquisition des connaissances pédagogique

Les connaissances pédagogiques font référence ici aux différentes stratégies d'enseignement que peuvent utiliser les STI. Dans les STI actuels, ces connaissances sont considérées comme des procédures et sont directement codifiées dans des programmes. Il n'y a donc pas de processus d'acquisition proprement dit.

2.5.3.6 Critiques de l'acquisition des connaissances pédagogique

Le fait de coder directement dans les STI les connaissances pédagogiques fait en sorte que leur manipulation soit du domaine des informaticiens. Si on considérait les connaissances pédagogiques comme des tâches, leur manipulation serait alors plus simple. Dans cette optique, certaines stratégies tutorielles comme l'enseignement coopératif pourraient être considérées comme des tâches coopératives. Un autre exemple pourrait être l'utilisation d'un pédagogue assistant qui aurait pour rôle de discuter des stratégies tutorielles avec le pédagogue humain. Le pédagogue assistant après avoir formé des apprenants, pourrait analyser leurs performances et présenter les résultats au pédagogue humain. Suivant les résultats de l'analyse, des décisions telles que la restructuration de la matière pourraient être prises ce qui impliquerait la mise à jour du curriculum.

2.5.4 Transmission des connaissances à l'apprenant

Ce mésosystème représente l'enseignement ou l'apprentissage du point de vue de l'apprenant. Il y a trois phases à distinguer dans le processus d'apprentissage: une phase de motivation, une phase d'acquisition et une phase de performance.

Au cours de la phase de motivation on présente à l'apprenant les objectifs de la leçon que l'on veut lui faire atteindre afin qu'il soit intéressé par le cours que l'on va lui présenter.

Lors de la phase d'acquisition, on définit le plan que la présentation du cours va suivre. Dans la littérature, trois types de planifications sont présentés: fixe, libre et hybride.

En phase de performance on doit vérifier si l'apprenant a bien assimilé le cours qu'il a suivi. Dans cette phase, on met également en place des outils de rétroaction qui permettront d'assister l'apprenant tout au long des tests de performance en lui apportant l'aide nécessaire pour rectifier ses erreurs.

Les trois phases de traitement de ce mésosystème font ressortir deux sous-systèmes fortement interdépendants: celui de transmission de connaissances à l'apprenant et celui d'acquisition de connaissances de l'apprenant. Cette interdépendance est due au fait que, pendant que l'on enseigne à l'apprenant ou tout simplement pendant que l'apprenant utilise le système pour apprendre, on en profite pour acquérir une foule d'informations le concernant. Elles permettent non seulement d'analyser le raisonnement de l'apprenant mais aussi de mettre à jour son modèle. Le mésosystème de transmission est celui qui est le plus en vue dans les STI, il est donc assez bien détaillé dans la littérature portant sur les STI mais aussi dans celle de l'éducation. L'interdépendance entre les deux sous-systèmes est tellement forte qu'ils ont la même architecture quoique leur dynamique soit différente. L'architecture que nous proposons prend donc en compte de manière explicite les deux sous-systèmes. Elle (Figure 6) fait ressortir cinq grands modules: un tuteur, un planificateur, des outils d'aides à l'enseignement, un modèle de l'apprenant et une interface utilisateur.

2.5.4.1 Le tuteur

Dans cette architecture, le tuteur est l'enseignant c'est-à-dire celui qui discute avec l'apprenant. Il lui présente le cours que le planificateur a produit. Il gère les outils d'aide disponibles pour l'apprenant. Ceux-ci peuvent être: un conseiller, un critique, un coach, un système coopératif, un système compétitif, etc. La gestion de ces outils est fortement reliée au modèle de l'apprenant avec qui le tuteur dialogue également.

Le tuteur est le module du STI qui est en contact direct avec l'apprenant Il comporte donc des outils qui permettent de recueillir les informations concernant celui-ci. L'analyse de ces informations permet de mettre à jour le modèle de l'apprenant. Certains auteurs comme Djamen [Djamen 95] ont développé des langages d'acquisition des connaissances de l'apprenant.

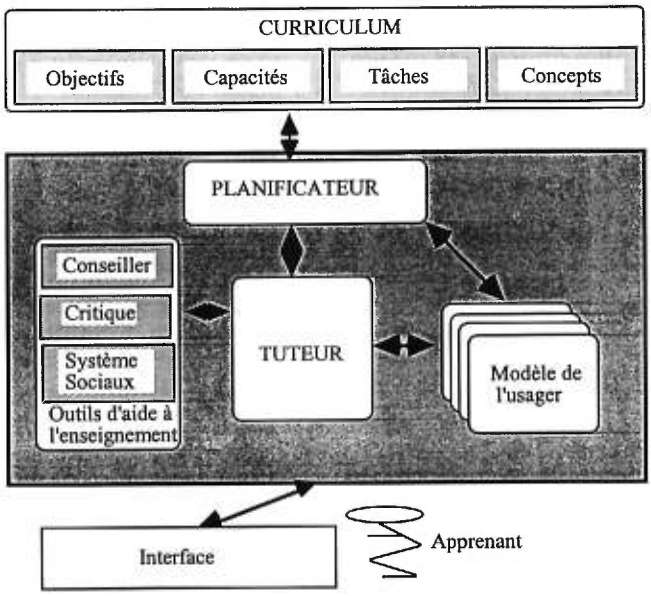


Figure 6 : Architecture du sous-système d'apprentissage

2.5.4.2 Le planificateur

Dans cette architecture, le planificateur intervient dans la phase d'acquisition de connaissances de l'apprenant, nous dirons plutôt dans la phase de présentation du cours à l'apprenant. Le planificateur développe donc un ensemble de stratégies de planification de cette présentation en fonction du modèle de l'apprenant qui utilise le système.

Le planificateur peut utiliser une stratégie fixe, c'est le cas dans SHERLOCK [Lesgold & al., 92]. Dans ce cas, l'apprenant n'a pas d'initiative par rapport au plan du cours. Il est donc contraint de suivre le parcours que le STI a tracé pour lui.

Le planificateur peut utiliser une stratégie libre, dans ce cas l'apprenant planifie lui-même son cours. Suivant ses sensations, il définit le parcours qu'il aura à suivre. Cette stratégie est surtout implantée dans les environnements de découverte.

Le planificateur peut enfin utiliser une stratégie hybride ou mixte. Dans ce cas, le STI impose des portions de parcours et laisse d'autres portions libres ou négocie ce parcours avec l'apprenant. Le parcours final résulte d'une diversité de types de planification.

2.5.4.3 Les outils d'aides à l'enseignement

Dans la littérature il existe deux grandes familles d'outils d'aide: ceux qui dans l'environnement technologique d'apprentissage ne tiennent compte que de

l'environnement cognitif de l'apprenant et ceux qui en plus tiennent compte de son environnement social.

Quelques outils ne prennent en compte que l'environnement cognitif de l'apprenant. Ils ne tiennent compte que des connaissances que l'apprenant manipule et des capacités intellectuelles que l'on veut lui faire acquérir.

- Le conseiller: il intervient à la demande de l'apprenant et répond aux questions comment faire, quoi faire, pourquoi.
- Le critique: il intervient sur la demande de l'apprenant et donne son avis sur le travail qui a déjà été effectué, il dira c'est bien ce que tu as fait mais tu aurais également pu le faire de telle manière ou n'exécute jamais l'action A avant l'action B comme tu l'as fait.
- Le coach: il se comporte comme un véritable entraîneur et suit l'apprenant pas à pas en phase de résolution d'exercice, et intervient quant il le juge nécessaire.

Quelques outils prenant en compte l'environnement cognitif et social de l'apprenant (les systèmes sociaux) . Dans ces systèmes, en plus des connaissances que l'apprenant manipule dans le cadre de son apprentissage, on essaye de prendre en compte tout un ensemble de phénomènes affectifs liés à l'influence des effets du travail en société sur l'apprenant tels que l'esprit de collaboration, l'esprit de compétition, le jeu de rôles, etc. Tous ces systèmes considèrent la composante sociale en tant que catalyseur dans le processus d'apprentissage.

Certains auteurs comme Clancey, Tak-Wai Chan ou John Self ont mis en place un ensemble de stratégies tutorielles basées sur l'aspect social de l'apprentissage, plus précisément sur la pédagogie du jeu de rôle et également sur des techniques comme l'inversion de rôle, le double jeu, la compétition, la collaboration.

- Le système co-apprenant [Dillembourg & Self, 91]. Le tuteur se comporte face à l'apprenant réel comme un autre apprenant et non comme un enseignant. L'échange entre les deux apprenants, le fictif et le réel devraient permettre à l'apprenant réel de mieux maîtriser l'objet de l'apprentissage.
- Le système compagnon[Chan & Baskin, 88]. Dans ce système l'apprenant est assisté d'un compagnon qui est simulé, et l'apprentissage est effectué sous la supervision d'un enseignant également simulé. En phase de performance, l'apprenant réel fait des échanges avec l'apprenant simulé et lorsqu'ils n'arrivent pas à résoudre un problème, l'enseignant simulé intervient en tant que conseiller ou critique pour les aider.

- Le système d'apprentissage par l'enseignement ou "Learning by teaching" [Chan & Baskin, 88]. Dans ce type de système, le tuteur joue le rôle de l'apprenant et l'apprenant réel joue le rôle d'enseignant. En phase de performance, le tuteur se met à exécuter des tâches et à un moment, il manifeste des difficultés et l'apprenant doit intervenir pour l'aider à les résoudre.

- D'autres systèmes faisant intervenir plusieurs apprenants réels ont été développés. Un exemple est celui décrit dans [Chan & al., 92] où on analyse les résultats lorsque les apprenants collaborent pour résoudre un problème, ou lorsqu'ils sont en concurrence.

2.5.4.4 Critique de la transmission des connaissances

Dans les STI actuels on observe de plus en plus une prise en compte de l'environnement social de l'apprenant. Malheureusement l'enseignement coopératif tel que le définit Slavin [Slavin, 88] n'est pas encore intégré. Dans la mesure où les STI conservent le modèle de chaque apprenant avec qui ils ont travaillé, il est possible de faire coopérer ces différents modèles. Cette coopération permettra de savoir à l'avance quels sont les points difficiles dans la matière, et quelles sont les stratégies les plus efficaces pour l'enseignement.

3 La coopération dans les systèmes informatique :Épistémologie de la coopération

3.1 Introduction

Le Larousse définit la coopération selon trois points de vue.

- Sur le plan politique: La coopération est définie comme la politique d'aide économique, technique et financière à certains pays en voie de développement.
- Sur le plan économique: la coopération est définie comme une méthode d'action par laquelle des personnes ayant des intérêts communs constituent une entreprise, ou les droits de chacun à la gestion sont égaux et où le profit est réparti entre les seuls associés au prorata de leurs activités.
- Sur le plan théorique: la coopération désigne l'action de coopérer c'est-à-dire agir conjointement avec quelqu'un.

Dans le domaine de l'informatique, la coopération est un concept jeune, vaste et pas encore clairement défini.

Jeunesse: les premiers articles parlant de la coopération dans le milieu informatique datent du début des années 1980 [Grundin, 94]. Cette coopération a commencé dans le domaine de l'interaction humain-machine.

Vaste: le mot coopération est actuellement présent dans presque toutes les disciplines liées à l'informatique. On parle ainsi de l'édition coopérative de documents, du dessin coopératif, de la coopération des agents, de l'enseignement coopératif etc.

Pas clairement défini: chaque domaine d'activité a sa propre définition de la coopération, [Tadié & Aimeur, 96] et vu la jeunesse du domaine, ces différentes définitions n'ont pas encore pu être unifiées.

Pour mieux appréhender le concept de coopération dans le milieu informatique, nous allons proposer un modèle de classification qui nous permettra de définir une taxonomie de la coopération dans l'informatique, et pour chaque classe dégagée, nous décrirons les

approches de mise en place. Mais avant cette description, nous allons analyser les différentes définitions de la coopération telles qu'elles existent aujourd'hui.

3.2 Définition de la coopération

Comme c'est souvent le cas pour des nouvelles technologies, il existe plusieurs définitions de la coopération dans l'informatique. Certaines sont très spécifiques à un domaine, d'autres essayent d'être le plus général possible. Certains auteurs dans leurs définitions font une différence entre la coopération et la collaboration, alors que d'autres considèrent ces deux termes comme équivalents. Nous allons dans la suite donner la définition de coopération du point de vue de certains auteurs, ensuite nous proposerons une définition que nous voudrons fédératrice, enfin nous mettrons en exergue la différence entre les termes coopération et collaboration.

3.2.1 Définitions actuelles

3.2.1.1 Définition selon BREZILLON

Patrick Brézillon dans [Brézillon, 92] définit la coopération par rapport à ce qu'elle est et ce qu'elle n'est pas. Ainsi donc pour Brézillon, la coopération implique une division des responsabilités entre les coopérants pour l'exécution des tâches et la volonté d'adhérer aux buts des autres.

La coopération n'est pas la collaboration, car cette dernière suppose une action conjointe des participants et l'entretient d'une compréhension mutuelle pour la tâche à accomplir, chaque participant ayant ses propres objectifs.

La coopération n'est pas l'aide informatique à l'activité coopérative entre plusieurs acteurs humains en faisant appel aux nouvelles technologies telles que le multimédia, les réseaux, etc.

Un élément essentiel de la coopération est le dialogue qui est utilisé pour négocier et résoudre les différences.

C'est aussi l'initiative qui doit être repartie entre les coopérants suivant leurs compétences respectives.

3.2.1.2 Définition de Schmidt et Bannon

Schmidt et Bannon qui sont spécialisés dans le domaine des travaux coopératifs communément appelés CSCW (Computer Supported Cooperative Work). donnent une définition de la coopération qui ne concerne pratiquement que leur domaine [Schmidt & Bannon, 92].

Définition: Les systèmes supportant le travail coopératif sont tous les systèmes qui servent de média de communication entre les différents membres d'un groupe. Ces membres sont relativement éloignés géographiquement, mais doivent travailler sur le même objet.

3.2.1.3 Définition d'Ellis, Gibbs et Rein

Ellis, Gibbs et Rein qui sont aussi du domaine des CSCW définissent les systèmes coopératifs comme étant [Ellis & al., 91] « des systèmes qui supportent des groupes de personnes engagés dans la réalisation de tâches communes ou d'objectifs communs, et qui donnent une interface commune à un environnement réparti ».

3.2.1.4 Définition de Loren G. Terveen

Terveen dans sa définition [Terveen, 95] ne fait pas de différence entre la coopération et la collaboration, en ce sens que la définition de la collaboration qu'il donne retrouve la définition de la coopération telle que décrite par Brézillon. Pour Terveen la collaboration est un processus par lequel deux ou plusieurs agents travaillent ensemble en vue d'atteindre un objectif commun.

3.2.2 Analyse des définitions existantes

Comme nous le voyons, les différentes définitions sont très ciblées, et chaque domaine exclut les autres dans sa définition de système coopératif. Brézillon dans sa définition fait une bonne distinction entre la coopération et la collaboration, il fait également bien ressortir la notion d'objectif commun, mais il exclut de sa définition tout ce qui ne fait pas partie du domaine de l'interaction Humain-Machine.

Schmidt et Bannon donnent une définition qui non seulement ne désigne comme coopérant que les humains, mais en plus il ne tient pas du tout compte de la notion d'objectif commun entre les différents coopérants, car le fait de travailler sur un même objet n'implique pas qu'on ait des objectifs communs.

Ellis, Gibbs, et Rein se soucient de la notion d'objectif commun entre les coopérants, mais ne prennent en compte que les coopérants humains.

Terveen dans sa définition ne fait pas de différence entre la coopération et la collaboration, ce qui étymologiquement n'est pas vrai dans la mesure où comme nous verrons plus loin dans la définition que nous proposons, la notion d'objectif commun n'existe pas dans la collaboration.

3.2.3 Notre définition

La définition que nous proposons doit prendre en compte les quatre points qui doivent caractériser un système informatique coopératif. Ces quatre points sont :

- Présence d'agents ou acteurs coopérants.
- Présence d'objectif commun pour tous les coopérants.
- La borne de communication entre les coopérants doit être un ordinateur.
- Un dialogue doit s'établir entre les coopérants.

Un agent ou acteur coopérant est un humain ou un module informatique doté de capacités de communication c'est-à-dire qui possède des organes capables de percevoir et de transmettre des informations sur le média de communication utilisé. Cette obligation de posséder des organes de communication fait en sorte que le dialogue soit possible entre les coopérants.

Un système informatique coopératif doit fournir des outils permettant aux différents coopérants de définir l'objectif qu'ils devront partager au cours d'une session de travail. Cette obligation d'objectif commun fait en sorte que l'on puisse faire une distinction nette entre les systèmes collaboratifs et les systèmes coopératifs. En effet, si l'on a deux partenaires qui résolvent chacun un problème qui lui est propre, si en plus les deux partenaires peuvent s'aider mutuellement à la demande, alors on est en présence d'un système qui permet la collaboration mais pas la coopération, car les objectifs ici sont différents malgré le fait qu'il y ait une entraide entre les partenaires.

La borne de communication où encore terminal de communication est l'objet qui sert d'interface entre un agent coopérant et le médium de communication. On doit parler de système informatique coopératif uniquement lorsque l'ordinateur est impliqué dans le processus de coopération, plus précisément lorsqu'il sert de borne de communication. Cette contrainte fait en sorte que deux personnes assises devant un ordinateur et

travaillant ensemble dans le but d'atteindre des objectifs communs ne doivent pas être considérées comme étant en train de travailler sur un système coopératif, malgré le fait que l'on soit bien devant un phénomène de coopération entre nos deux individus. Cette forme de coopération n'a rien à voir avec l'informatique.

Si aucun dialogue n'est établi entre les différents coopérants, il est impossible de parler de coopération même si toutes les autres conditions sont remplies. Cette exigence est nécessaire dans la mesure où si l'on dispose d'agents partageant le même objectif, utilisant l'ordinateur comme borne de communication, mais prenant soin de ne rien se communiquer, on ne pourra pas parler de système coopératif.

Après toutes ces contraintes, notre définition de la coopération dans le domaine de l'informatique serait donc:

Un système informatique est dit coopératif s'il permet à deux ou plusieurs agents coopérants de dialoguer dans le but d'atteindre un objectif commun en utilisant l'ordinateur comme borne de communication.

3.3 Taxinomie des systèmes coopératifs

3.3.1 Introduction

Il n'existe pas encore de classification pour les systèmes coopératifs dans leur ensemble, mais certaines classifications ont déjà été effectuées pour certains sous domaines comme celui des CSCW où Ellis, Gibbs et Rein [Ellis & al. 91] proposent une taxonomie des groupwares ou travaux coopératifs supportés par l'ordinateur.

Selon eux on peut classer les « groupwares » selon deux axes: un représentant le temps et l'autre l'espace (figure 7). Ainsi, on peut classer les systèmes de travail coopératifs assistés par ordinateur selon qu'ils exigent que les membres du groupe soient au même endroit, en même temps ou pas.

	Même temps	Temps différents
Même Endroit	Interaction Face-à-face	Interaction asynchrone
Endroit différent	Interaction synchrone distribuée	Interaction asynchrone distribuée

Figure 7 : Matrice temps/espace du groupware

3.3.2 Proposition

La taxinomie de la coopération que nous proposons est orientée usager, c'est-à-dire que les caractères de discrimination sont basés uniquement sur les acteurs d'une coopération. Les caractéristiques que nous avons recensées et qui nous permettent de créer une taxinomie de la coopération sont:

- le type de coopération, qui est une combinaison possible des types d'intervenant,
- la distance physique entre les intervenants,
- la distance temporelle entre les intervenants.

3.3.2.1 Types de coopération

Le type des intervenants dans un processus de coopération définit la nature de la coopération. Les deux types d'intervenants que nous considérons sont l'humain et le système. Ils nous permettent de définir trois types de coopération:

- coopération humain-machine,
- coopération machine-machine,
- coopération humain-humain.

3.3.2.2 Distance physique entre intervenants

La distance physique entre les membres d'un processus de coopération permet de savoir si la communication nécessitera l'utilisation d'un réseau de communication distant ou pas. On a deux modalités pour représenter les distances:

- face à face, c'est-à-dire que les intervenants sont au même endroit,
- distant, dans ce cas les intervenants ne sont pas sur le même site.

3.3.2.3 Distance temporelle entre intervenants

La distance temporelle entre les intervenants dans un processus de coopération permet de savoir à quel moment un interlocuteur perçoit le message qu'a transmis un autre intervenant. Comme dans le modèle d'Ellis Gibbs et Rein, la distance temporelle permet d'avoir une coopération synchrone ou asynchrone.

Ces différentes caractéristiques nous permettent de proposer une première classification (figure 8) de la coopération dans le monde informatique.

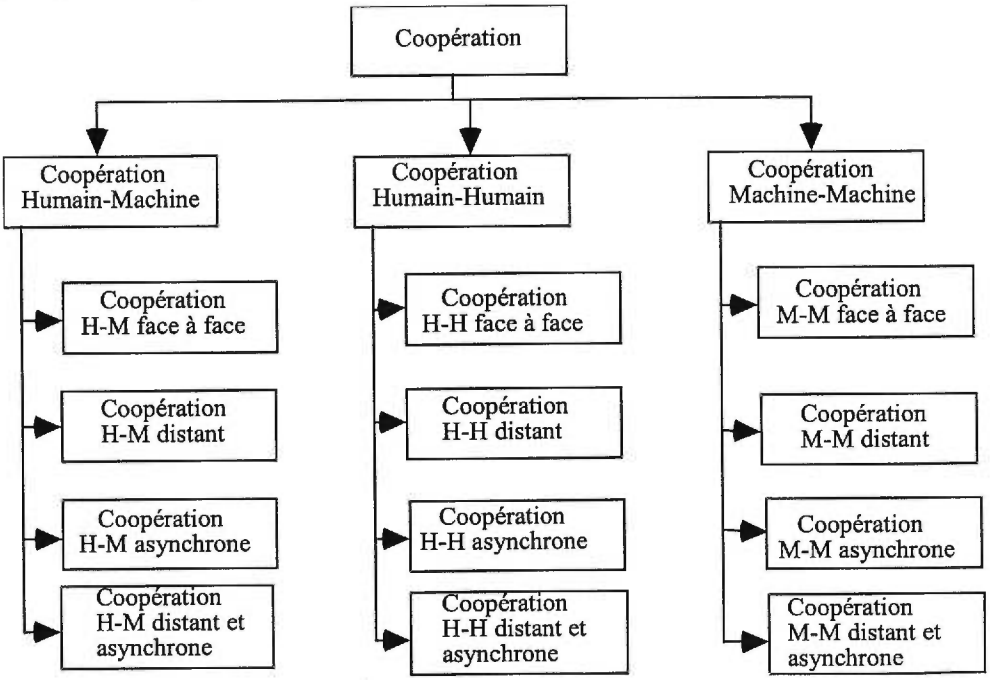


Figure 8 : Une classification de la coopération en informatique

Dans cette classification, seule la première caractéristique nous permet de décrire un ensemble cohérent de classes, car les autres caractéristiques ne sont pas applicables à toutes les classes obtenues après la première transformation. Par exemple la classe coopération humain-machine distant ou asynchrone est un peu irréelle. Nous allons de ce fait utiliser les trois premières classes générées par notre taxinomie pour analyser la coopération.

3.4 Coopération humain-machine

La coopération demande l'introduction des méthodes d'intelligence artificielle dans la construction des interactions entre l'humain et la machine. Avant d'arriver à cette introduction de l'IA, il est à noter que, jusque dans les années 1980, c'est l'humain qui devait s'adapter à l'ordinateur, et ceci n'était pas toujours facile. Il est arrivé plusieurs fois que des systèmes informatiques ne puissent pas être utilisés car les utilisateurs potentiels du système n'arrivaient pas à communiquer avec le système. Pour remédier à cela les chercheurs ont été amenés à construire des systèmes qui coopèrent avec l'humain plutôt que de résoudre des problèmes.

Cette nouvelle approche dans l'interaction entre l'humain et la machine a donné naissance à deux écoles de pensées.

La première approche considère que pour, que la machine puisse coopérer avec l'humain, il faut que la machine puisse simuler les habiletés de l'homme. Avec cette approche, la coopération humain machine devient une coopération humain-humain, car la machine doit être dotée d'outils d'expression comme le langage, les sens, le raisonnement. Cette approche est une approche d'émulation

La deuxième approche considère la coopération humain-machine comme étant le moyen d'exploiter les habiletés intrinsèques de la machine afin que cette dernière soit le complément de l'humain. Cette approche est une approche de complémentarité.

3.4.1 Approche d'émulation.

Cette approche nécessite la compréhension et la modélisation des mécanismes de communication chez l'humain. Cette modélisation se focalise principalement sur les intentions, les pensées, les connaissances, et tout ce qui peut affecter l'état mental d'un agent [Terveen, 95].

L'homme est vu comme un agent rationnel, c'est-à-dire quelqu'un capable de former des plans lui permettant d'atteindre les buts qu'il s'est fixé. La communication permet aux agents de dialoguer dans le but de résoudre un problème posé.

Cette école de pensée s'est donc penchée sur des formalismes de représentation de pensées, de buts, de plans et d'actions. Ensuite pour ces différents formalismes, il a fallu mettre en place les moyens de faire de la coopération, de développer des algorithmes pour la communication de planning et pour la reconnaissance de plan.

3.4.2 Approche de complémentarité

Le but de cette approche est de faire en sorte que l'ordinateur devienne un partenaire de discussion un peu plus intelligent. Plus intelligent dans ce cas ci ne veut pas dire ressembler à l'homme, mais plutôt d'être capable par exemple de pouvoir faire des anticipations sensées lorsqu'il est en communication avec l'homme [Terveen, 95], [Tadié & al. 96b].

Pour atteindre ce but, les recherches se sont focalisées sur la structuration des connaissances de l'humain, la manière dont les humains travaillent individuellement ou en groupe, la représentation des connaissances, le raisonnement, l'interaction et les techniques de présentation des informations.

Pour anticiper dans ses interventions, le système doit être capable de prédire les intentions de son partenaire à partir de ses échanges avec lui. Cette prédiction d'intention sur la base des actions à effectuer étant très difficile, la plupart des systèmes contournent cette difficulté en faisant en sorte que l'usager définisse explicitement ses intentions.

3.4.3 Architecture commune

Malgré ces différentes approches, l'architecture d'un système coopératif Humain-Machine est unique (Figure 9).

On a dans cette architecture un modèle de l'usager qui va permettre au système d'adapter ses interventions.

Les premières recherches sur le modèle de l'usager proviennent du domaine de la langue naturelle qui a été ensuite généralisé aux systèmes interactifs. Actuellement les recherches les plus poussées sont faites dans la communauté des Systèmes Tutoriels Intelligent(STI) [Nkambou & al., 96]. Trois grandes questions se posent lorsque l'on veut se pencher sur le modèle de l'usager:

Acquisition: comment obtient-on les informations nécessaires à propos d'un usager? faut-il explicitement les lui demander, ou faut-il les récupérer de manière implicite à partir de ses actions ?

Représentation: quelles sont les informations sur l'usager qui devraient être utiles? Comment ces informations doivent-elle être représentées. On parle de modèle cognitif, de modèle affectif ou encore de modèle conatif [Nkambou & al., 96].

Raisonnement: quels types de raisonnement sont appropriés pour le type d'information que nous manipulons au niveau d'un usager?

Le deuxième module de cette architecture est l'univers du discours dans lequel le système va puiser les connaissances nécessaires afin de coopérer de manière optimum avec l'usager. Cet univers est encore appelé domaine d'interaction.

Le troisième module est celui du raisonnement qui permet de construire les plans que devra suivre le système afin de coopérer avec l'usager. Ce module peut être adaptatif (s'adapte au modèle de l'usager) ou standard (raisonne à partir d'un modèle standard de l'usager).

Le quatrième module met en œuvre le dialogue avec l'usager en tenant compte du modèle de ce dernier et du plan généré par le planificateur.

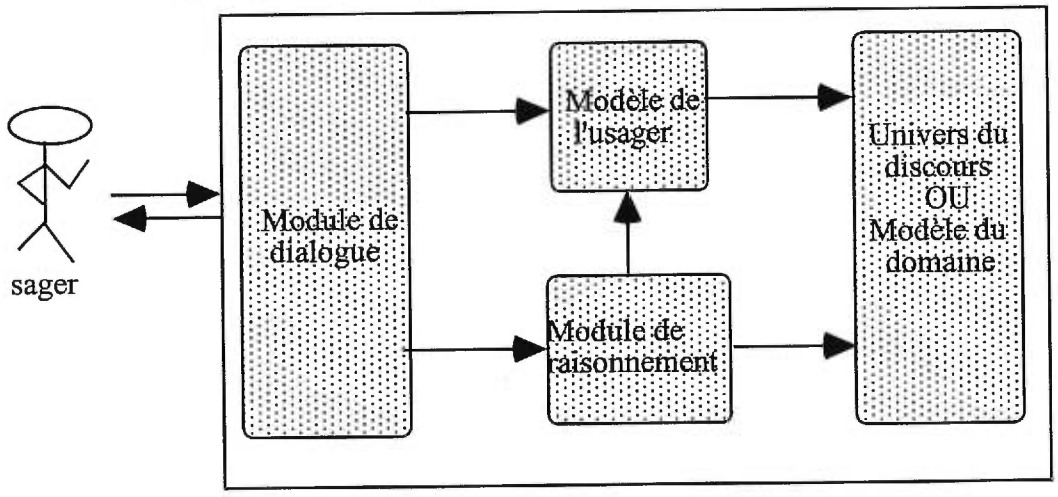


Figure 9: Structure d'un système coopératif entre l'humain et la machine

3.5 Coopération humain-humain

Cette classe fait référence au domaine du CSCW où le souci est de faire coopérer deux ou plusieurs humains à la réalisation d'une tâche coopérative en utilisant l'ordinateur comme

borne de communication. Ce domaine des systèmes coopératifs étant actuellement très développé, nous allons essayer de l'analyser un peu plus profondément que les autres classes.

Définition: les systèmes de travaux coopératifs sont des systèmes coopératifs où les agents coopérants sont tous des humains.

Pour mieux comprendre ce domaine de la coopération, nous allons utiliser un modèle qui nous permet de caractériser les travaux coopératifs, et du même coup proposer une taxinomie pour cette classe.

Pour définir les caractéristiques du travail coopératif, nous proposons une approche orientée où l'usager est au centre des préoccupations. Cette approche de modélisation se rapproche des travaux d'Ellis et de Wainer [Ellis & Wainer, 94], et le modèle que nous proposons englobe le modèle en trois aspects qu'ils proposent (modèle ontologique, modèle fonctionnel, modèle de la communication). Cette approche nous a permis de mettre au point un modèle basé sur quatre niveaux.

- Un niveau décrivant les objets et les opérations disponibles aux usagers. Ces opérations et ces objets sont ceux sur lesquels travaille le groupe. Cette couche représente le modèle de la statique du système.
- Un niveau décrivant l'évolution des objets dans le système. Cette couche représente le modèle de la dynamique du système.
- Un niveau décrivant les différentes formes de présentations et de communications entre les différents usagers et le système. Cette couche représente le modèle de la communication du système.
- Un niveau décrivant les règles de comportement des différents membres d'un groupe de travail. Cette couche représente le modèle de la stratégie.

Nous enrichirons également ce modèle par les spécifications faites dans le cadre de la construction de GroupKit [Roseman & Greenber, 92] qui est un ensemble d'outils servant au travail coopératif en temps réel. Roseman et Greenberg pensent que la construction d'un logiciel de groupware passe par trois stratégies.

- Une architecture orientée objet et extensible, supportant la gestion distribuée des processus et les communications entre ces processus.

- Un recouvrement transparent offrant une méthode pour ajouter des composants généraux sur diverses applications de groupware.
- Une intégration des façons conventionnelles de travailler de l'humain.

3.5.1 Le modèle de la statique

Ce modèle représente les spécifications statiques du système comme le modèle entité association ou une analyse orientée objet. Ce modèle décrit donc les classes d'objets et les opérations que le système met à la disposition des usagers. Au niveau du modèle de la statique, le groupware doit également permettre de gérer les notions telles que la confidentialité, la protection (intégrité) des objets manipulés et le contrôle [Dourish, 93].

Ces problèmes de sécurité sont particuliers au travail coopératif en ce sens qu'ils introduisent de nouveaux concepts de sécurité, tel que le droit d'avoir une vue particulière sur un objet partagé, qui n'existaient pas dans les modèles classiques de sécurité. Sachant que chaque membre du groupe peut avoir un rôle particulier, il faut définir les droits d'accès de manière à ce que chacun ne puisse manipuler que les objets sur lesquels il en a le droit [Shen & Dewan, 92]. Cette limitation est importante dans la mesure où si on laisse chaque usager accéder à tous les objets du système, en espérant contrôler les accès par le protocole social, on arrive très vite dans des problèmes inextricables comme ce fut le cas avec le prototype de l'éditeur coopératif GROVE [Ellis & al., 91]. Ceci s'explique par le fait que, lorsqu'un usager a la volonté de détruire, il est impossible de l'en empêcher avec le protocole social.

Suivant les applications du groupware, le modèle de la statique peut servir à représenter quatre réalités différentes.

- Un objet « soft », c'est-à-dire un objet qui est crée par un logiciel: c'est le cas d'un document électronique qui représente le modèle de la statique dans le cas de l'écriture coopérative.
- Un objet « hard », c'est-à-dire un objet réel qui est visualisé par voie de vidéo par tous les membres du groupe ou tout simplement par certains d'entre eux: c'est le cas d'un malade qui est examiné par plusieurs médecins à la fois dont certains sont géographiquement éloignés du patient.
- Un objet « hard-sSoft », c'est-à-dire un objet réel qui est visualisé à l'aide de caméras vidéo et sur lequel on construit un modèle simulé à partir d'un logiciel de simulation de telle sorte que les actions que l'on effectue sur ce modèle aient des impacts sur

l'objet réel par l'intermédiaire d'outils d'expression tels que des capteurs, des bras manipulateurs, ou des robots.

- Un objet « virtuel », c'est-à-dire un objet qui n'est pas visible ou concret comme une conférence ou une réunion.

3.5.2 Le modèle de la dynamique

Ce modèle représente les spécifications fonctionnelles, qui peuvent être des diagrammes de flux de données, des réseaux de Pétri, des ATN (Augmented Transition Network), ou toute autre forme de description de la dynamique des systèmes. Ce modèle décrit donc les activités que chaque participant doit accomplir et comment celles-ci doivent être synchronisées afin que le travail du groupe soit mené à bien.

Les principaux problèmes rencontrés à ce niveau sont:

- le problème de consistance des objet,
- le problème de l'accès concurrent aux objets,
- le problème de la synchronisation et
- le problème plus général de la communication.

3.5.2.1 Problème de consistance

Ce problème se pose lorsque les objets partagés par un groupe sont répliqués sur plusieurs sites. Il faut donc s'assurer que lorsqu'il y a des modifications, tous les exemplaires sont mis à jour au même moment. Ce problème pose celui plus général de l'architecture des logiciels de travaux coopératifs.

Trois architectures sont possibles de nos jours.

- L'architecture centralisée. Dans ce cas l'objet de travail est physiquement sur un seul site, et les autres sites y accèdent par l'intermédiaire de réseaux de communication. Cette architecture facilite la gestion des objets partageables, mais elle est très lourde pour les usagers car les temps d'accès peuvent être extrêmement longs selon que l'on soit proche ou éloigné géographiquement du site où se trouve l'objet, et selon le fait que le réseau utilisé soit un réseau à haut débit ou non.

- L'architecture distribuée. Dans ce cas, l'objet de travail est réparti physiquement sur plusieurs sites distincts. Avec cette architecture, le rendement est déjà meilleur mais

les temps d'accès demeurent longs lorsqu'un usager veut accéder à une partie de l'objet qui est sur un autre site que celui sur lequel il travaille.

- L'approche avec réplication. Dans ce cas, l'objet de travail est répliqué en autant de copies qu'il y a d'usagers dans le groupe de travail et chaque poste de travail héberge une copie de l'objet. Le problème au niveau de cette stratégie est de pouvoir maintenir une certaine cohérence de l'objet à travers ses différentes copies.

3.5.2.2 Problème d'accès concurrents

Lorsque plusieurs personnes peuvent accéder de manière concurrente à un même objet, il faut s'assurer que l'objet reste cohérent après des mises à jour. Habituellement pour résoudre ce problème que l'on retrouve également dans le domaine des bases de données, on utilise habituellement une technique de verrouillage, c'est-à-dire que si un usager accède à un objet dans le but de le modifier, il en prend le contrôle exclusif jusqu'à la fin de la mise à jour. Cette stratégie est adéquate dans le cadre des bases de données car la durée du blocage et de la mise à jour sont acceptables.

Dans le cadre du travail coopératif cette technique s'avère exaspérante pour les usagers dans la mesure ou la tâche de mise à jour est effectuée par une action humaine qui peut prendre une durée assez appréciable. Pour résoudre ce problème, certains systèmes ont mis en place des mécanismes de mise à jour d'objets partagés sans blocage [Ellis & al., 91].

Au niveau de la coordination, il y a des systèmes qui permettent au groupe de ne travailler que sur une seule tâche à la fois alors que d'autres permettent la cohabitation de plusieurs tâches dans le système à un même moment avec la possibilité qu'il puisse y avoir des échanges d'information entre elles.

3.5.2.3 Problème de synchronisation

Ce problème est relatif à la synchronisation dans l'exécution des tâches et non à la synchronisation des vues. Dans la mesure où une tâche est exécutée de manière coopérative, la synchronisation entre les différents postes de travail est très importante pour la réussite de cette tâche. Ce problème est d'autant plus grave que le média de communication, dans le cas où il n'est pas fiable, entraîne des dérèglements préjudiciables à la bonne synchronisation de la tâche coopérative.

Une des façons de régler ce problème, encore héritée du monde des bases de données, est l'utilisation de la technique d'estampillage. Il s'agit d'accoler à chaque transaction (mise à jour ou recherche) une étiquette indiquant l'heure à laquelle cette transaction a été émise, la station et le processus émetteur, le type de transaction et sa priorité. En se servant de cette étiquette on applique une stratégie de détermination de priorité ou de synchronisation appropriée. Étant donné que les transactions voyagent à travers le réseau il faut pouvoir tenir compte du délai de propagation dans l'ordre de priorité des opérations sur les objets.

3.5.2.4 Problème de communication

Un système de travail coopératif doit fournir des infrastructures robustes en matière de communication, car de ces infrastructures dépendent, en grande partie, le succès ou l'échec de ce type d'application. Ces infrastructures de communications doivent permettre à chaque participant de pouvoir envoyer des messages collectifs ou individuels. Pour cela il faut donc implanter les stratégies d'envoi de messages par diffusion ou point à point.

Plusieurs technologies de télécommunication pouvant faciliter la mise en place des systèmes de travail coopératif assisté par ordinateur ont fait leur apparition sur le marché. Les technologies les plus attrayantes semblent être les technologies de communication à large bande: ATM, Frame Relay, SMDS [Mark & Miller, 94]. Mais ces technologies demeurent relativement chères. Des solutions avec des vieilles technologies telles que Ethernet, moins performantes mais acceptables sont souvent utilisées.

À ce problème de robustesse des infrastructures de communication, il faut ajouter d'autres problèmes tels que la transparence de localisation et ceux qui se rattachent à la qualité de service. En effet, dans certaines applications, il est souhaitable que les utilisateurs n'aient pas à connaître la localisation des différents agents ou ressources constituant le réseau. Dans d'autres, il est important, pour des raisons de coût ou d'ergonomie que les utilisateurs puissent déterminer eux mêmes certains paramètres du système tels que la qualité de l'image ou du son et le nombre de sessions en parallèle.

3.5.3 Le modèle de la communication

Au vu des deux premiers modèles, on constate que le groupware constitue un changement radical dans l'utilisation de l'ordinateur. Ce dernier n'est plus utilisé pour résoudre des problèmes ponctuels mais plutôt pour promouvoir et faciliter les interactions entre les personnes. Pour profiter au maximum de cette nouvelle approche nous devons également

envisager une nouvelle approche dans la construction des interfaces usagers qui ne seront plus des interfaces personne-machine mais plutôt des interfaces groupe de personne-machine ou tout simplement interface groupe-machine.

Ellis et Wainer [Ellis & Wainer, 94] pensent que cette interface peut être vue sous la forme de trois composantes: les vues de l'objet du travail, des participants et du contexte. Nous complétons cette approche par une quatrième vue, qui est la vue des outils d'expression qui pour nous est capitale dans la construction des outils de travail en groupe. L'humain utilisant une variété de modes d'expression, les modèles de groupeware doivent prendre en compte cette dimension médiatique. Ainsi, nous avons les vues suivantes:

- une vue des informations de l'objet de travail,
- une vue des outils d'expression (langage de communication),
- une vue des participants et
- une vue du contexte.

3.5.3.1 Vue des objets

L'interface usager doit être capable de présenter les différents objets du système et les opérations de manipulation qui s'appliquent à ces objets. Cette représentation, pour être conviviale, doit être faite en tenant compte des droits d'accès et des préférences de manipulation de chaque membre du groupe de travail.

Les objets d'une interface usager se divisent en deux grands groupes:

- les objets propres à l'usager, ceux qui ne sont visibles que par l'usager et qui sont des objets de travail individuels et
- les objets accessibles à tous les membres du groupe de travail.

En ce qui concerne ces derniers, ils peuvent être caractérisés suivant deux facteurs : le mode de travail et la granularité des informations échangées.

Le mode de travail qui peut être synchrone ou asynchrone. Dans le mode synchrone, une modification sur l'un des postes de travail est immédiatement visible sur les autres postes de travail. Dans ce cas on parle de synchronisation en temps réel ou de système WYSIWIS (What You See Is What I See). Lorsque le travail est fait de façon asynchrone, une modification sur l'un des postes de travail met un certain temps avant d'être prise en compte par les autres postes.

La granularité des informations à transmettre qui dépend de l'objet sur lequel le travail coopératif est effectué. Par exemple, s'il s'agit de l'écriture d'un document, la granularité peut aller du simple caractère au chapitre en passant par la ligne, le paragraphe, ou la page.

3.5.3.2 Vue des outils d'expression

Les outils de groupware doivent être capables de supporter les actions des usagers à travers l'espace de travail visuel. Ces outils devraient permettre d'exploiter différents modes d'expression, car l'être humain utilise des variétés de modes d'expressions qui vont de la gestuelle au regard, en passant par la parole et les annotations. Pourtant, ces formes d'expression ne sont pas prises en compte par les interfaces usagers classiques. Un bon système permettant le travail coopératif devrait donc être capable d'introduire la gestuelle et le regard au niveau de l'interface de communication.

Les chercheurs de Xerox Parc [Tang, 91] ont montré que sur des surfaces de dessins traditionnelles, les participants avaient toujours tendance à utiliser des gestes pour présenter une idée, pour signaler une prise de parole, pour focaliser l'attention du groupe ou pour faire une référence à des objets sur la surface de travail. Cette constatation a amené les constructeurs de systèmes de dessins coopératifs à supporter la gestuelle avec l'apparition de plusieurs curseurs sur l'écran de chaque poste de travail [Greenberg & al, 92]. L'utilisation de ces systèmes a confirmé l'importance de la gestuelle dans le travail en groupe.

Un autre outil d'expression que devraient supporter les systèmes de travaux coopératifs est l'outil d'annotation graphique. Toujours chez Xerox Parc, la même équipe a montré qu'il est important, lorsqu'on désigne un objet, qu'il y ait une annotation particulière sur cet objet. Cette annotation peut-être soit un soulignement, un encadrement ou toute autre façon de focaliser sur cet objet. En définitive, les systèmes de travaux coopératifs devraient implanter des outils d'expression qui permettent à l'usager de s'exprimer le plus naturellement possible.

3.5.3.3 Vue des participants

Les systèmes de groupware étant principalement dédiés à la coopération entre usagers, il est important que chaque usager sache qui sont les autres membres du groupe de travail. Cette information est importante pour deux raisons principales:

Suivant la présence de telle ou telle autre personne, le travail du groupe (les interventions individuelles) peut évoluer différemment. Par exemple, dans un débat sur l'évolution des Systèmes Tutoriels intelligents (STI), si l'on sait que parmi les participants il y a une personne nommée Claude Frasson, notre discours sera différent de celui que nous aurions fait s'il n'y était pas.

Les recherches ont également montré que, lorsqu'on entend quelqu'un sans le voir, le taux de rétention de ce qu'il dit est plus faible que si on l'entendait et le voyait à la fois.

Une vue plus poussée de ces principes a fait naître la notion de télé présence, qui consiste à faire en sorte que lorsqu'un usager voit un autre usager à travers un écran, c'est comme s'il le voyait à travers une vitre [Okada & al., 94]. Ainsi chaque usager a l'impression d'être dans la même salle que les autres. Pour atteindre cet objectif, la recherche travaille sur des domaines tels que: le vidéo fenêtrage, le vidéo hall, les salles de classes virtuelles et la réalité virtuelle.

3.5.3.4 Vue du contexte

Le contexte est vu ici comme les informations de service ou encore les informations qui n'ont pas directement trait au travail lui-même. Ces informations peuvent être catégorisées en quatre groupes:

- les informations structurelles, qui sont les informations relatives à la session de travail en cours;
- les informations à caractère social, c'est-à-dire les informations sur le groupe, ses normes, son historique social, etc.;
- les informations sur l'organisation, il s'agit ici de l'organisation hiérarchique du groupe et
- les informations concernant les outils ad hoc de travail tels que, les outils de vote, les outils de « brainstorming », les outils d'organisation d'idées et autres.

3.5.4 Modèle de la stratégie

Ce modèle permet de mettre en place tous les outils liés à la construction et à la gestion des groupes de travail. Au niveau de la progression du groupe durant la session de travail plusieurs stratégies sont envisageables. Certains auteurs pensent que le système devrait imposer un modèle social pour l'interaction du groupe [Nunamaker & al, 91] ce qui serait une tentative basée sur la théorie du management selon laquelle il faut fournir des

méthodes précises pour concentrer le groupe sur la tâche. Ceci renforcerait les rôles et les engagements en rendant le groupe plus efficace et plus productif. D'autres pensent que le protocole social devrait être déterminé uniquement par les membres du groupe de travail [Dykstra & Carasik, 91]. Une dernière catégorie d'auteurs [Johnson-Lenz & Johnson-Lenz, 91; Greenberg, 91] pense qu'une méthode hybride est la meilleure dans la mesure où elle permet de conjuguer les avantages des deux autres.

3.5.5 Conclusion sur les modèles de « groupware »

Les caractéristiques que nous avons énoncées dans cette section ne sont pas toutes nécessaires dans tous les systèmes de « groupware ». Par exemple les contraintes sur la granularité qui sont très importantes pour des domaines comme le dessin coopératif ou l'écriture coopérative n'ont pas beaucoup de sens pour des domaines comme la téléconférence.

Au niveau des groupwares génériques, il y a de plus en plus de prototypes disponibles sur le marché. Ces prototypes n'ont pas pour but d'effectuer le travail coopératif, mais plutôt de fournir des outils pour la construction d'environnement de travail coopératif. Il s'agit principalement du logiciel Lotus Notes [Lotus, 93] et d'Info Lens. Ces logiciels, en plus de fournir des outils pour la construction de « groupwares », mettent en place des mécanismes pour leur analyse ainsi que des mécanismes de coordination entre divers groupes en phase d'utilisation [Ellis & Keddara, 93].

Dans cette section nous avons proposé un modèle générique permettant de caractériser les systèmes de travaux coopératifs assistés par ordinateur. Le tableau 1 récapitule les éléments de ce modèle.

Tableau 1 : Synthèse du modèle de classification des groupwares

Modèle de la statique	Modèle de la dynamique	Modèle de la communication	Modèle de la stratégie
Types d'objets du travail coopératif: soft hard hard-soft virtuel	Types de problèmes rencontrés: consistance accès concurrents synchronisation communication	Types de vues offertes par l'interface: objet de travail outils d'expression participants contextes	Types de stratégies de gestion du groupe: imposées par le système décidées par les membres hybrides

3.6 Coopération machine-machine

Cette classe de coopération fait référence aux systèmes multi-agents, ou aux systèmes acteurs. Ferbert [Ferbert, 94] définit les systèmes multi-agents comme étant des systèmes composés des éléments suivants:

- un espace environnemental E, c'est-à-dire un espace muni au moins d'une topologie et plus généralement d'une notion de distance;
- un ensemble situés d'objets O , c'est-à-dire que, pour tout objet, il est possible, à un moment donné, d'associer une position dans E;
- un sous-ensemble A de O qui représente les entités actives du système;
- un ensemble de relations R qui unit des objets (et donc des agents) entre eux;
- un ensemble d'opérations F permettant aux agents de A de percevoir, produire, consommer, transformer et manipuler des objets de O.

Cette définition de Ferbert est fortement liée au fait qu'il associe les systèmes multi-agents à des systèmes dédiés à la simulation alors qu'ils sont aussi bien adaptés à la résolution de problème. Au niveau de la coopération, c'est cette caractéristique des

systèmes multi-agents de pouvoir résoudre des problèmes qui nous vaut son utilisation dans la coopération Machine-Machine.

3.6.1 Différents types d'agents

On distingue dans les systèmes multi-agents plusieurs types d'agents [Frasson & al., 96]. Ce sont les agents réactifs, les agents cognitifs et les agents sociaux.

Agent réactif: c'est un agent qui réagit à des stimulus pour modifier son environnement ou pour envoyer un message à d'autres agents. Il n'est pas doté de mécanismes lui permettant de raisonner sur ses intentions. Les actions qu'il effectue découlent d'un plan prédéterminé.

Agent cognitif: c'est un agent qui est capable de raisonner sur ses intentions, ses pensées, afin de créer des plans d'actions qu'il exécute dans un environnement donné. Les agents cognitifs ne possèdent pas de modèle des autres agents, ils ne sont donc pas capables de raisonner sur les intentions et les plans des autres agents.

Agent social: c'est un agent cognitif possédant un modèle explicite des autres agents. Il est donc capable de raisonner sur les connaissances des autres agents. Cela lui permet de connaître leurs intentions, d'anticiper leurs réactions. Il se construit ainsi un plan d'actions qui respecte celui des autres.

Dans un système multi-agents coopératif, les agents peuvent être de tous les types. En effet dans un système coopératif, ce qui importe c'est d'être capable de communiquer avec les autres membres du groupe, c'est-à-dire d'être capable de synchroniser ses activités avec les autres membres du groupe de travail.

Nous proposons ici une définition de système multi-agents coopératif découlant de la définition de système coopératif que nous avons donnée et de la description des agents que nous venons de faire.

3.6.2 Définition de système multi-agent coopératif

Un système multi-agents coopératif est un système où plusieurs modules informatiques dialoguent et agissent sur un même objet de travail dans le but d'atteindre un objectif commun.

3.6.3 Interaction et communication entre agents.

Il existe deux principaux modes de communication dans les systèmes multi-agents.

Communication par modification de l'environnement des agents (Figure 10): dans ce mode encore appelé tableau noir, les informations manipulées dans le système coopératif sont partagées dans une mémoire commune à tous les agents. Ainsi si un agent veut prendre une décision par rapport à une valeur du système, il aura tout simplement à aller dans la mémoire partagée et récupérer l'information désirée. Si un agent veut plutôt mettre des informations à la disposition des autres agents, il ira déposer l'information dans la mémoire partagée par les agents.

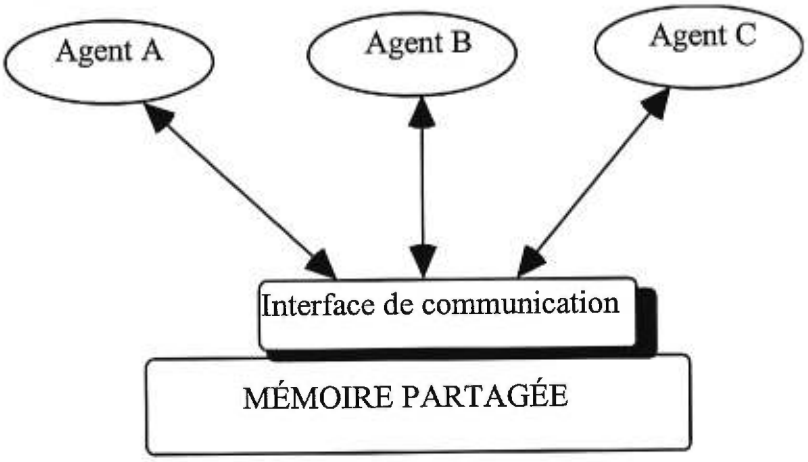


Figure 10 : Communication entre agents à partir d'une mémoire partagée

Communication par échanges de messages (Figure 11): Dans ce mode, les agents communiquent entre eux par envoi de message. Si un agent A1 a besoin d'une information détenue par un autre agent A2, alors il envoie une requête à A2, en retour l'agent A2 envoie la réponse à l'agent A1.

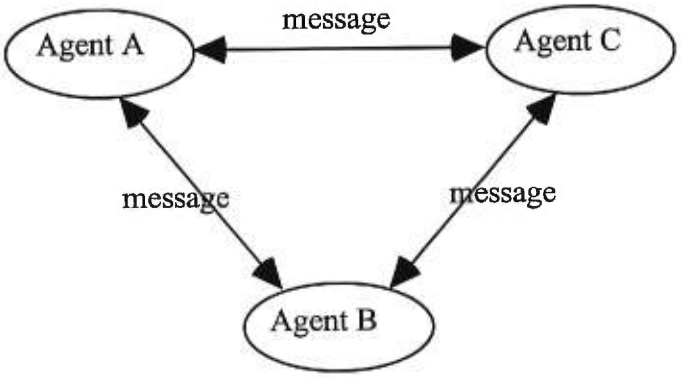


Figure 11: Communication entre agents par envoi de messages

3.7 Conclusion

Dans cette partie, nous avons essayé de définir clairement ce qu'est un système coopératif. Cette définition fait également ressortir la différence qu'il y a entre un système coopératif et un système collaboratif.

Cette mise au point étant effectuée, nous avons défini et analysé trois grandes classes de coopération dans le monde informatique. Ce sont:

- la coopération humain-machine,
- la coopération humain-humain,
- la coopération machine-machine.

Ces trois classes représentent exactement les styles de coopération auxquels nous devons faire face dans le système d'enseignement de tâches coopératives. La coopération humain-machine représente la relation devant exister entre le tuteur à un poste de travail et l'apprenant humain qu'il doit suivre. La coopération humain-humain modélise la situation où plusieurs apprenants humains se mettent à résoudre une tâche coopérative sous la supervision ou non d'un tuteur. La coopération machine-machine quant à elle renvoie à la résolution coopérative d'une tâche par un ensemble d'agents informatiques. Le système d'enseignement que nous devons mettre sur pied est donc un environnement de coopération où gravitent les différentes formes de coopération recensées. Pour le construire nous allons dans le chapitre suivant proposer un modèle permettant de représenter les tâches coopératives.

4 MONACO_T : Un modèle de tâche coopérative supportant l'enseignement et la simulation de tâches coopératives

4.1 Introduction

Lorsqu'on construit un modèle de tâche coopérative destiné à être enseigné par un Système Tutoriel Intelligent (STI), on doit s'organiser pour que ce modèle permette de garder la structure cognitive de la tâche coopérative, soit capable d'en présenter une vue globale et facilite la génération d'explications pouvant être utilisées par un conseiller, un critique, ou toutes autres stratégies d'apprentissage [Tadié & al., 96a].

Lorsque le modèle de tâche coopérative est destiné à être exécuté dans un environnement de réalisation, on doit s'assurer que les outils permettant de simuler le rôle de chaque poste de travail soient présents ainsi que les mécanismes permettant de suivre l'exécution de la tâche.

Les travaux coopératifs sur lesquels nous nous penchons sont des travaux d'équipe où chaque membre doit maîtriser d'une part les actions à faire au niveau du poste qu'il occupe et d'autre part l'intégration de ses actions dans le processus de la tâche collective. Un STI orienté dans l'enseignement du travail coopératif doit répondre à ces deux objectifs.

Dans les STI classiques qui sont consacrés à l'enseignement de tâches individuelles, plusieurs modèles de tâches ont été proposés. Il s'agit en l'occurrence de GUIDON [Clancey, 87], des tâches génériques [Chandrasekaran, 87], du modèle de Kieras [Bovair & al., 90], de la couche tâche dans le modèle KADS [Schreiber & al., 93] ou encore du graphe de tâche utilisé dans le projet SAFARI [Djamen, 95]. Ces modèles sont efficaces dans les STI, mais ne sont pas adaptés pour les tâches coopératives. Cette inadéquation est principalement due au fait qu'il n'y a pas de prise en compte de plusieurs intervenants à la fois.

L'univers des systèmes multi-agents, a donné naissance à des modèles de tâches coopératives [Péninou, 93], [Ferber, 94]. Ces modèles sont orientés agent, car ils donnent

la priorité de la modélisation à la structure de l'agent. Avec cette approche, la structure cognitive de toute la tâche coopérative est dispersée dans les différents agents chargés de la réaliser . Ces modèles sont donc orientés vers la simulation et la réalisation. Ils sont en général structurés sous la forme de boîte de verre ou de boîte noire. Dans le premier cas, les explications peuvent être données mais la structure cognitive de la tâche n'est pas conservée. Dans le second cas, les explications ne peuvent même pas être générées.

Au niveau des modèles de coordination des activités de groupe, le schéma de coordination est représenté par les réseaux de Pétri. Ils sont en effet un bon outil pour représenter les notions de synchronisation, de choix exclusif et de parallélisme d'exécution. Si les jetons dans les réseaux sont porteurs d'information et que les transitions intègrent des algorithmes, on parle de réseaux de Pétri colorés. Ce modèle est très efficace dans le processus de simulation d'une tâche coopérative. Lorsqu'il s'agit de générer des explications permettant de justifier les choix d'actions qui ont été effectués le modèle montre ses limites. De plus les algorithmes étant programmés directement dans le modèle, l'utilisation d'un tel système pour codifier le comportement d'une tâche coopérative ne peut se faire que par un expert informaticien.

Dans ce chapitre, nous proposons un modèle de représentation de tâches coopératives utilisable aussi bien dans un environnement d'enseignement que de simulation de tâche coopérative. Le modèle de tâche sur lequel nous nous penchons concerne aussi bien des tâches abstraites que des tâches de manipulations de système. Les tâches abstraites sont des tâches comme celle qui consiste à prendre une décision en fonction d'un certain nombre de paramètres alors que les tâches de manipulations de systèmes, comme celle qui consiste à piloter un avion, nécessitent la présence d'un système doté d'un comportement propre. Dans ce dernier cas le système peut être un logiciel, la simulation d'un appareil physique ou tout autre appareil dont les actions peuvent être déclenchées à partir d'un ordinateur.

Dans la suite nous allons d'abord décrire les orientations choisies, nous présenterons le modèle MONACO_T et enfin, nous verrons comment il prend en considération la dimension coopérative de la tâche.

4.2 Modélisation orientée tâche versus modélisation orientée agent

Dans le domaine des tâches coopératives, deux approches de modélisation sont possibles: modélisation orientée agent (utilisée dans les systèmes multi-agents) et modélisation orientée tâche (utilisée dans les STI classiques). Cette dernière donne la priorité à l'analyse cognitive de la tâche avant d'y greffer les différents agents.

La structure cognitive de la tâche est largement utilisée dans le cadre des STI pour contrôler l'activité, donner des conseils, répondre aux questions ou évaluer la performance. Dans la mesure où l'enseignement de tâches coopératives est un volet important dans nos objectifs et qu'une tâche modélisée pour l'enseignement contient une information plus importante que celle nécessaire à sa réalisation, nous allons opter pour une approche orientée tâche (Figure 12) et nous y intégrerons les éléments nécessaires pour obtenir un environnement de réalisation de tâches coopératives.

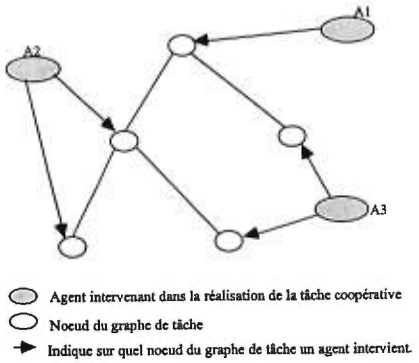


Figure 12: Modélisation de la tâche coopérative suivant l'orientation tâche

Dans MONACO_T, décrire une tâche coopérative consiste donc dans un premier temps à modéliser la tâche en se souciant uniquement de sa structure cognitive sans s'intéresser aux interactions entre les acteurs de la coopération et ensuite affecter chaque sous-tâche à un des postes de travail.

4.3 Tâche conceptuelle versus tâche de manipulation de système

Nous prenons en considération dans notre analyse les tâches conceptuelles et celles de manipulation de système. Une tâche conceptuelle est définie comme une tâche ne

nécessitant pas d'interaction avec un logiciel ou un appareil quelconque. Ce sont principalement des tâches de prise de décision en fonction de certains paramètres.

Les tâches de manipulation de système quant à elles permettent d'intégrer dans un même environnement le modèle de la tâche et le modèle du système à manipuler. Ce couplage permet de prendre en compte les effets que les actions d'une tâche ont sur le système qu'elles sont censées manipuler. Nous donnerons comme exemple la tâche de pilotage d'avion, dans ce cas la tâche coopérative est couplée avec un simulateur de vol de telle sorte que les activités sur la tâche ont un impact réel sur le comportement du vol.

4.4 Spécification de MONACO_T

Notre approche de modélisation considère trois couches qui distinguent les propriétés d'une tâche coopérative. Ce sont ses caractéristiques intrinsèques, le comportement de cette tâche lors de son exécution et les stratégies d'intervention. Ces trois couches sont respectivement la statique, la dynamique et le scénario (Figure 13):

- la statique représente la structure arborescente de la décomposition de la tâche en sous-tâches,
- la dynamique représente le comportement de la tâche en phase d'exécution avec la génération et l'utilisation d'événements se produisant au niveau de chaque sous-tâche,
- la couche scénario décrit un ensemble de stratégies de réalisation de la tâche coopérative. Si elles sont définies par poste de travail on parle alors de scénarios individuels. Elles peuvent également être décrite pour une équipe, dans ce cas on parle de scénarios coopératifs.

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

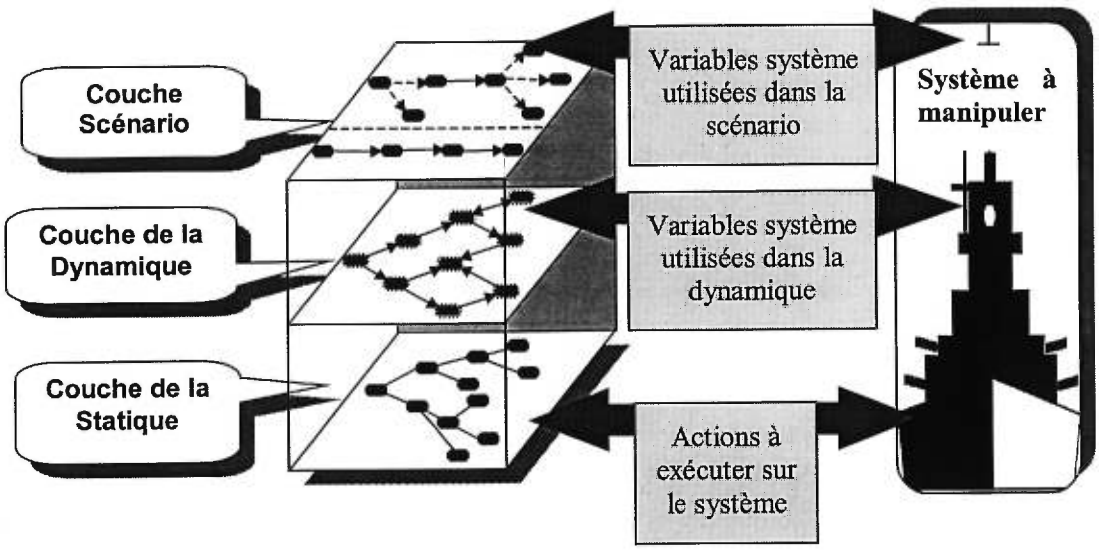


Figure 13 : Structure du modèle MONACO_T

4.4.1 Description de la statique

La plupart des modèles qui permettent de générer des explications sont des cas particuliers du modèle GOMS (Goals, Operators, Methods, and Selection Rules). Nous citerons par exemple la structuration des tâches dans SHERLOCK [Lesgold & al., 92], le modèle de tâche de Kieras ou encore le graphe de tâche dans le projet SAFARI. Au niveau de la statique, cette approche considère que:

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

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

- Une tâche n'est pas un but abstrait à atteindre, mais représente une action à réaliser. Lorsque cette tâche est décomposable, son action consiste en la composition des

résultats produits par ses sous-tâches. Lorsqu'il s'agit d'une tâche de manipulation de système, l'action consiste à exécuter une commande sur le système concerné.

- Il n'y a pas de notion de sous-tâches abstraites, toutes les sous-tâches sont opérationnelles car représentant une action.
- Il y a une prise en considération de la possibilité d'une exécution coopérative de la tâche par l'introduction de postes de travail au niveau de chaque sous-tâche.
- Chaque sous-tâche intègre également des activités complémentaires qui permettent de maintenir la cohérence de la tâche coopérative. Ces activités appelées activités de dépendances fonctionnelles mettent à jour certaines variables caractéristiques de la tâche coopérative dans des états particuliers.
- Pour permettre une meilleure manipulation de l'ensemble des sous-tâches de la tâche coopérative, nous créons des sous-ensembles de sous-tâches. Les éléments d'un sous-ensemble ont une relation d'appartenance particulière. Exemple : l'ensemble des sous-tâches du poste de travail X, l'ensemble de sous-tâches permettant de faire des entrées de données, etc.). Cet ajout permet principalement de pouvoir manipuler la logique des prédicats au niveau des couches dynamique et scénario.

La notion de composition telle que nous l'entendons est une notion vaste et complexe qui peut aller de la simple juxtaposition des résultats produits par les sous-tâches composantes à une combinaison complexe de ces résultats.

Si nous voulons comparer les modèles de type GOMS avec le modèle MONACO_T, nous dirons qu'un modèle GOMS est un modèle MONACO_T où toutes les sous-tâches abstraites ont comme action de composition la juxtaposition des résultats produits par leurs sous-tâches. Les tâches modélisées à l'aide du modèle GOMS permettent de faire la simulation mais ne sont pas assez adaptées à l'enseignement de la réalisation de cette tâche. Cette inadaptation est due au fait que ce modèle n'intègre pas des informations d'explication et il n'est pas toujours aisé de détecter les intentions d'un utilisateur. Le modèle GOMS ne s'occupe également pas de l'intégration de plusieurs agents dans le processus de réalisation d'une tâche donnée.

Un exemple de tâche coopérative décomposable dans le milieu médical est représenté ci-dessous (Figure 14).

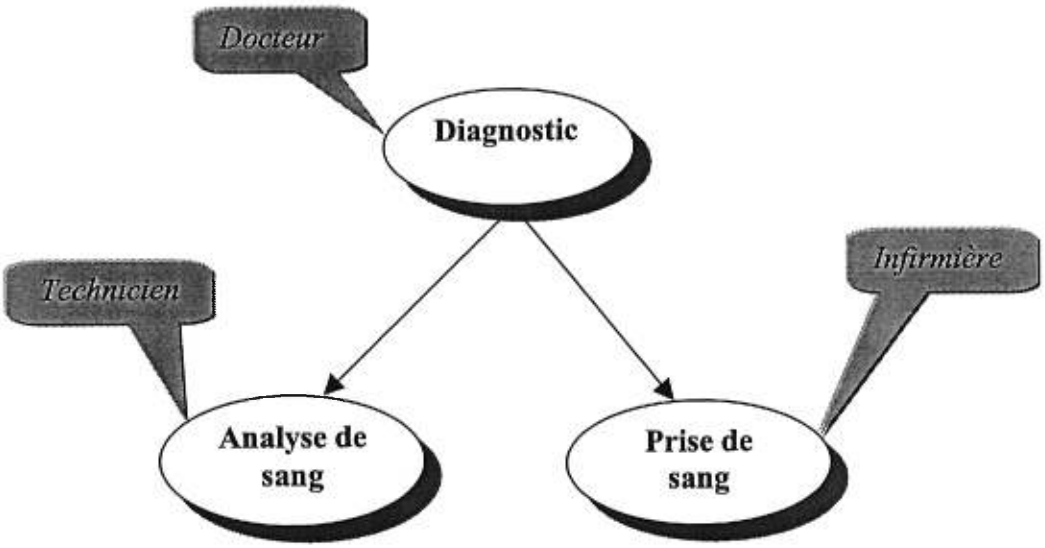


Figure 14: Exemple de décomposition d'un tâche coopérative

Cette décomposition hiérarchique se lit: pour faire un diagnostic, on a besoin des résultats de l'analyse du sang et de la prise de sang.

Chaque nœud au niveau de la statique possède un ensemble de propriétés: nom, description, méthode de réalisation (qui décrit comment réaliser l'action de la sous-tâche), liste de résultats possibles (utile pour spécifier la dynamique), méthode d'évaluation (permet d'évaluer les apprenants)) et poste de travail (ressource).

4.4.2 Définition formelle de la statique.

Une tâche à la norme MONACO_T est structurée sous forme de neuf ensembles: un ensemble de postes de travail (E_{pt}), un ensemble de sous-tâches (E_{st}), un ensemble d'étapes d'exécution (E_{ee}), un ensemble d'actions (E_a), un ensemble de systèmes manipulés (E_{sm}), un ensemble d'activités de dépendance fonctionnelle (E_{adf}), un ensemble de variables caractéristiques (E_{vc}), un ensemble de valeurs possibles relatives aux variables caractéristiques (E_{vp}), un ensemble d'opérateurs de comparaison (E_{oc}) et un ensemble contenant des ensembles de groupes homogènes de sous-tâches (E_{gst}).

L'ensemble (E_{ee}) contient exactement les éléments suivants : Déclenchement, Réalisation, Terminaison. L'ensemble (E_{vc}) contient au minimum les éléments suivants : « État », « Statut », « NombreRéalisation ». L'ensemble (E_{vp}) quant à lui contient au minimum les éléments suivants : *Initial*, *Déclenché*, *Réalisé*, *Terminé*, *Encours*, *Suspendu*. Nous verrons plus tard le rôle de ces variables et ces valeurs possibles dans le processus d'évolution de la tâche coopérative.

Dans l'exemple de la Figure 3, E_{pt} ={Médecin, Laborantin, Infirmière}, E_{st} ={Diagnostic, PriseDeSang, AnalyseDeSang}, E_a ={Diagnostiquer, PréleverLeSang, AnalyserLeSang}.

Cardinalité de E_a = cardinalité de E_{st} ·

4.4.2.1 Définition des ensembles

Définir les ensembles manipulés au niveau de la couche statique c'est expliciter la sémantique des éléments qui la compose.

Ensemble des postes de travail E_{pt}

Lorsque nous disons qu'une tâche est coopérative, c'est parce que son exécution demande la participation de plusieurs expertises représentées chacune par un poste de travail. Par exemple, pour s'occuper d'un patient dans une salle de soins intensifs on a besoin des postes de travail infirmière, laborantin et médecin. Pour faire décoller un avion d'un aéroport les postes de travail sont pilote, copilote et opérateur de tour de contrôle.

Ensemble des sous-tâches E_{st}

Une sous-tâche représente un ensemble homogène d'actions dont la réalisation concoure à atteindre un but donné. Cette homogénéité est forcément relative. Deux individus ne produiront pas forcément la même décomposition pour une tâche donnée. L'ensemble des sous-tâches représente donc une décomposition possible d'une tâche coopérative suivant des sous-buts.

Ensemble des systèmes manipulés E_{sm}

L'ensemble des systèmes manipulés contient des éléments uniquement lorsque la tâche coopérative modélisée est une tâche de manipulation de système. Un système représente

une entité qui reçoit des stimulus et transmet des réponses à travers l'univers médiatique dans lequel il est en activité.

Par exemple, si notre entité est une lampe, les stimulus actionnables sont « allumer », « éteindre », « brancher » et « débrancher ». Lorsqu'on envoie la requête « allumer » à la lampe, cette dernière vérifie si elle est branchée et éteinte. Dans le cas où ces conditions sont vérifiées, elle transmet dans l'univers médiatique le résultat de l'allumage d'une lampe. Si l'univers médiatique est une pièce, alors la pièce va s'illuminer. Si l'univers médiatique est une ligne de communication, alors le message lampe allumée va être envoyé sur la ligne. Si l'univers médiatique est composé de ces deux médias, la pièce va s'illuminer et un message va être envoyé sur la ligne.

Une tâche coopérative pouvant manipuler plusieurs systèmes à la fois, l'ensemble E_{sm} doit donc permettre de contenir tous ces systèmes.

Les systèmes participants à une tâche coopérative doivent mettre, à la disposition de la tâche coopérative les méthodes et les paramètres qui correspondent aux requêtes ou stimulus auxquels ils peuvent réagir. Ces méthodes et leurs paramètres sont semblables aux interfaces des composants utilisés dans les systèmes de programmation comme les JavaBeans de Sun.

Ensemble des actions E_a

Une action est un objet qui permet de décrire comment réaliser la sous-tâche qui l'héberge. Lorsqu'on est en présence d'une tâche de manipulation de système, l'action, en plus de décrire comment réaliser la sous-tâche, permet également de faire une requête à l'un des systèmes manipulés par la tâche coopérative. Cette requête consiste à demander au système de rendre effective la réalisation de la sous-tâche.

Si on modélise une tâche coopérative de pilotage d'avion, et que l'une des sous-tâches consiste à tourner le gouvernail de l'avion, alors l'exécution de la requête par le système va nous faire voir le gouvernail qui tourne et l'avion qui change de direction.

Ensemble des étapes d'exécution de sous-tâches E_{ee}

Dans la mesure où le modèle de tâche coopérative que nous mettons en place doit servir aussi pour l'enseignement, nous avons jugé utile qu'il faille annoncer ses intentions avant

de réaliser une sous-tâche. Ce choix permet au système tutoriel qui utilise notre modèle d'avoir toutes les informations lui permettant d'analyser le raisonnement de l'apprenant. Lorsque le modèle est utilisé dans le but de soutenir la réalisation d'une tâche coopérative, ce choix permet à tous les postes de travail de signifier où on se trouve dans l'accomplissement de la tâche coopérative. Après avoir réalisé la tâche, il est également souhaitable de préciser quand on a terminé. Ainsi l'ensemble des étapes d'exécution de sous-tâches contient les éléments suivants : « Déclenchement », « Réalisation », « Terminaison ».

Ensemble des variables caractéristiques E_{vc}

Ce sont les variables qui permettent de caractériser une tâche coopérative. Du pilotage d'avion à une de prise en main d'un patient dans une salle de soins intensif, ces variables sont différentes. Il en existe cependant toujours au moins trois dans le modèle de toute tâche coopérative qui permet d'en décrire l'évolution syntaxique au cours de sa réalisation. Ce sont : « État », « Statut » et « NombreRéalisation ».

Ensemble des valeurs possibles pour les variables caractéristiques E_{vp}

Le modèle MONACO_T doit permettre entre autres de simuler des rôles (postes de travail) lors de la réalisation d'une tâche coopérative. Définir les valeurs possibles pour les variables caractéristiques c'est préciser le domaine dans lequel l'agent simulé va puiser les résultats à leur affecter. Cette opération se fait à la construction de la tâche. Pour celles qui sont communes à toutes les tâches coopératives, les valeurs possibles sont prédéfinies et sont développées dans le tableau 2.

Tableau 2: Valeurs possibles de certaines variables caractéristiques de Monaco-T

Variable caractéristique	Valeurs possibles pour la variable caractéristique
État	Initial, Déclenché, Réalisé, Terminé
Statut	Initial, EnCours, Suspendu, Terminé
NombreDeRéalisation	$x / x \in 0,1,2,...,Sup(N)$ *ou* $Sup(N) = \max$ *des entiers*

Ensemble des opérateurs de comparaison E_{oc}

Les opérateurs de comparaisons permettent de construire des tests à partir des variables caractéristiques et de leurs valeurs possibles. Ainsi l'usager en fonction de la tâche coopérative qu'il construit peut introduire tous les opérateurs de comparaison binaires ou unaires qu'il juge utiles dans la manipulation de la tâche coopérative.

Ensemble des activités de dépendances fonctionnelles E_{adf}

On appelle activité de dépendance fonctionnelle (adf) le passage de la valeur d'une variable caractéristique d'une valeur initiale donnée à une valeur d'arrivée donnée. Cette transition permet d'assurer une évolution fonctionnelle de la tâche coopérative au fil de sa réalisation. Les variables caractéristiques utilisées ici sont celles de la tâche coopérative. Nous verrons au niveau de l'étude de la dynamique quelles valeurs sont affectées à ces variables et à quels moments, afin que la cohérence de la tâche coopérative puisse être maintenue.

Une activité de dépendance fonctionnelle exprime le fait que si la variable caractéristique y d'une sous-tâche x a la valeur v1 alors celle-ci doit être remplacée par v2. Lorsque aucune valeur de départ n'est précisée (v1 est absent) v2 doit être la nouvelle valeur quelque soit la valeur initiale.

Ensemble des groupes homogènes de sous-tâches E_{gst}

Définir l'ensemble des groupes homogènes de sous-tâches c'est permettre de manipuler les sous-tâches autrement qu'élément par élément. Pour chaque constructeur de tâche coopérative tout groupe de sous-tâches liées par un lien sémantique quelconque peut être identifié par un nom particulier. À partir du modèle que nous proposons, il y a des groupes qui émergent tels que le groupe représentant les sous-tâches d'un poste de travail donné, celui représentant les sous-tâches feuilles et celui représentant les sous-tâches internes.

4.4.2.2 La relation de composition

Cette relation permet de construire l'arborescence des sous-tâches par une méthode de décomposition basée sur le fait que, pour s'exécuter, une sous-tâche a besoin des résultats produits par d'autres sous-tâches.

$\forall x, y \in E_{st}$, $x \Omega y \Leftrightarrow$: y entre dans la composition de x .

Exemple: Diagnostic Ω AnalyseDeSang

La relation (Ω, E_{st}, E_{st}) est une relation d'ordre qui définit une arborescence de sous-tâches.

4.4.2.3 Définition des fonctions de manipulations de la statique

Pour manipuler efficacement la statique de la tâche coopérative, un ensemble de fonctions a été défini.

Fonction fils d'une sous-tâche

Utilise la relation de composition qui existe entre les sous-tâches de notre tâche coopérative.

$$F_{fst} : E_{st} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{fst}(x)$$

Elle retourne comme résultat l'ensemble des sous-tâches directement liées à x par la relation de composition

Fonction descendance d'une sous-tâche

Utilise la relation de composition jusqu'aux feuilles de l'arbre.

$$F_{fdst} : E_{st} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{fdst}(x)$$

Le résultat est l'ensemble des sous-tâches descendant de x par la relation de composition. Cet ensemble correspond également à l'ensemble des sous-tâches appartenant à l'arborescence dont la racine est x.

Fonction père d'une sous-tâche

Se définit comme suit :

$$F_{pst} : E_{st} \longrightarrow E_{st}$$

$$x \longmapsto f_{pst}(x)$$

Le résultat produit est la sous-tâche père de la sous-tâche x.

Fonction ascendance d'une sous-tâche

Utilise la relation de composition en chaînage arrière. Elle correspond à l'application récursive de la fonction père jusqu'à l'atteinte de la sous-tâche racine.

$$F_{fast} : E_{st} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{fast}(x)$$

Cette fonction donne comme résultat l'ensemble des sous-tâches ascendantes de x. C'est également l'ensemble des sous-tâches qui sont sur le chemin qui va de la racine à la sous-tâche x.

Fonction action d'une sous-tâche

$$F_{ast} : E_{st} \longrightarrow E_a$$

$$x \longmapsto f_{ast}(x)$$

Cette fonction retourne l'action à exécuter lors de la réalisation de la sous-tâche x. Cette fonction est bijective.

Fonction actions d'un sous arbre

$$F_{asa} : E_{st} \longrightarrow P(E_a)$$

$$x \longmapsto f_{asa}(x)$$

Ensemble des actions se trouvant dans la sous arborescence de racine x.

Fonction sous-tâche d'une action

Cette fonction permet d'obtenir la sous-tâche qui héberge une action donnée. Cette fonction représente la fonction inverse de la fonction action d'une sous-tâche.

$$F_{sta} : E_a \longrightarrow E_{st}$$

$$x \longmapsto f_{sta}(x)$$

Le résultat produit est la sous-tâche hébergeant l'action x.

Fonction poste de travail d'une sous-tâche

$$F_{ptst} : E_{stt} \longrightarrow E_{pt}$$

$$x \longmapsto f_{ptsy}(x)$$

Le résultat produit est le poste de travail chargé de réaliser l'action de la sous-tâche x.

Fonction postes de travail d'un sous arbre

Cette fonction permet d'obtenir les postes de travail chargés de réaliser les sous-tâches appartenant à un sous arbre de la tâche coopérative.

$$F_{ptsa} : E_{st} \longrightarrow P(E_{pt})$$

$$x \longmapsto f_{ptsa}(x)$$

Cette fonction donne comme résultat l'équipe de postes de travail chargé d'effectuer la tâche coopérative dont la racine est représentée par la sous-tâche x.

Fonction sous-tâches d'un poste de travail

$$F_{stpt} : E_{pt} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{stpt}(x)$$

Cette fonction donne comme résultat l'ensemble des sous-tâches dont est responsable le poste de travail x dans la tâche coopérative.

Fonction étape d'exécution d'une sous-tâche

Pour les sous-tâches internes, les étapes de réalisations sont : *déclenché*, *réalisé* et *terminé*. Au niveau des sous-tâches feuilles elles sont les suivantes : *réalisé* et *terminé*. Dans ce cas, les étapes *déclenché* et *terminé* peuvent être confondues sans perte

d'information. En d'autres termes, pour réaliser une sous-tâche feuille, je n'ai pas besoin de signaler mon intention en la déclenchant.

$$F_{eest} : E_{st} \longrightarrow P(E_{ee})$$

$$x \longmapsto f_{eest}(x)$$

Cette fonction donne comme résultat l'ensemble des étapes d'exécution disponibles au niveau de la sous-tâche x.

Fonction sous-tâches feuilles

$$F_{stf} : E_{st} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{stf}(x)$$

Le résultat produit est l'ensemble des sous-tâches feuilles du sous arbre dont la racine est représentée par la variable x.

Fonction sous-tâches internes

Cette fonction permet de ressortir les sous-tâches qui sont internes (pas des feuilles) dans un sous arbre de la tâche coopérative.

$$F_{sti} : E_{st} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{sti}(x)$$

Cette fonction fournie en résultat l'ensemble des sous-tâches internes du sous arbre dont la racine est x.

Fonction activités de dépendances fonctionnelles d'une étape

$$F_{adfe} : E_{st} \times E_{ee} \longrightarrow P(E_{adf})$$

$$(x,y) \longmapsto f_{adfe}(x)$$

Le résultat est l'ensemble des activités de dépendances fonctionnelles qui sont réalisées lorsque l'étape y de la sous-tâche x est exécutée.

Fonction étape d'une activité de dépendances fonctionnelles

Pour une activité de dépendance fonctionnelle particulière, c'est à dire la mise à jour d'un champ par une valeur donnée, la fonction retourne les différentes actions de la tâche coopérative (étapes d'exécution de sous-tâches) qui sont capables de l'activer. Par exemple pour faire passer le champ « Etat » de la sous-tâche prise de sang de *Suspendu* à *EnCours* les étapes d'exécutions sont: « déclenchement prise de sang » et « terminaison prise de sang ».

$$F_{eadf} : E_{eadf} \longrightarrow P(E_{st} \times E_{st})$$

$$x \longmapsto f_{eadf}(x)$$

Le résultat produit est l'ensemble des étapes d'exécution de sous-tâches qui ont l'activité de dépendance fonctionnelle x dans la liste de leurs activités de dépendance.

Fonction variables caractéristiques d'une sous-tâche

$$F_{vcst} : E_{st} \longrightarrow P(E_{vc})$$

$$x \longmapsto f_{vcst}(x)$$

En résultat nous avons l'ensemble des variables caractéristiques qui permettent de décrire la sous-tâche x.

Fonction valeurs possibles d'une variable caractéristique

$$F_{vpvc} : E_{vc} \longrightarrow P(E_{vp})$$

$$x \longmapsto f_{vpvc}(x)$$

Cette fonction retourne comme résultat l'ensemble des valeurs possibles que l'on peut affecter à la variable caractéristique x au cours du processus de résolution de la tâche coopérative.

Remarques :

La variable caractéristique « État » a comme valeurs possibles : *Initial*, *Déclenché*, *Réalisé*, *Terminé*.

La variable caractéristique « Statut » a comme valeurs possibles : *Initial*, *EnCours*, *Suspendu*, *Terminé*.

La variable caractéristique « NombreItération » a comme valeurs possibles l'ensemble des entiers naturels.

Fonction opérateurs de comparaison d'une variable caractéristique

$$F_{ocvc} : E_{vc} \longrightarrow P(E_{oc})$$

$$x \longmapsto f_{ocvc}(x)$$

Le résultat produit par cette fonction est l'ensemble des opérateurs de comparaison que l'on peut utiliser pour faire des tests sur la variable caractéristique x.

Fonction sous ensemble d'un groupe

Les sous-tâches dans MONACO_T étant regroupées en familles appelées groupes, cette fonction nous permet de retourner toutes les sous-tâches qui appartiennent à un groupe donné.

$$F_{seg} : E_{gst} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{seg}(x)$$

Cette fonction produit comme résultat l'ensemble des sous-tâches liées sémantiquement dans le groupe référencé par la variable x.

4.4.2.4 Implantation

Pour faciliter la construction de tâches suivant Monaco-T, nous avons mis en place un environnement graphique d'édition. Il permet de produire des objets Monaco-T intégrant les trois couches spécifiées. Au niveau de la statique, l'éditeur fournit des outils permettant de construire les différents ensembles qui la définisse. On peut parler de l'ensemble des sous-tâches, de l'ensemble des postes de travail, de l'ensemble des variables caractéristiques, etc. Graphiquement et d'une manière interactive (figure 15) les éléments de ces ensembles sont mis en relation. Il y a donc une instanciation graphique de la relation de composition entre sous-tâches et une instanciation interactive de la

relation de propriétaire de variables caractéristiques. Lorsque la tâche à construire est de type manipulation de système, l'environnement d'édition fournit un outil permettant de lier la tâche au système manipulé (figure 15).

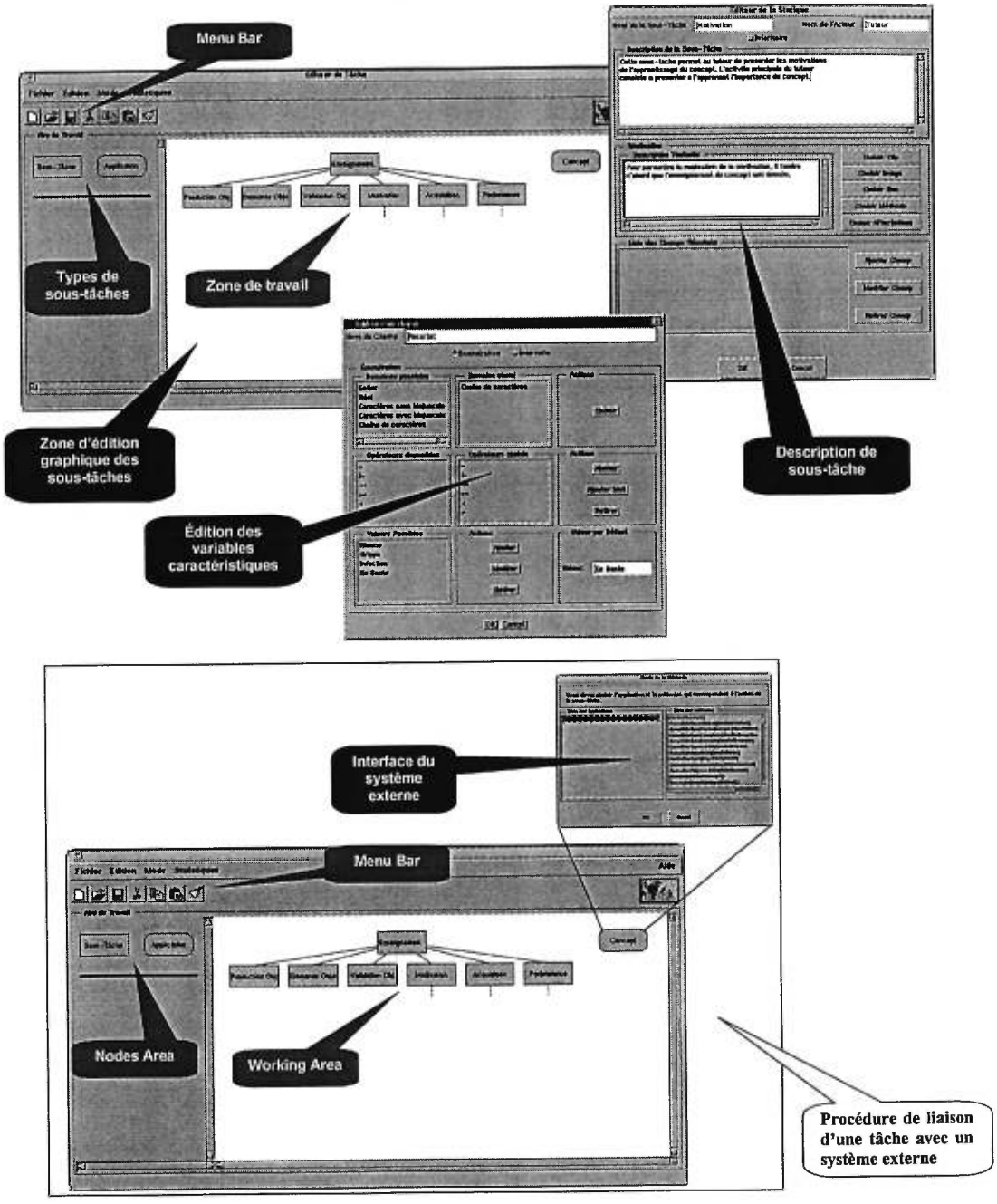


Figure 15: Interface d'édition des informations de la statique

4.4.3 Description de la dynamique d'une tâche.

La dynamique d'une tâche représente la manière dont celle-ci évolue en phase d'exécution. Dans les modèles proches de GOMS, cette dynamique est représentée par des règles de sélection. Dans MONACO_T elle est construite au-dessus de la statique de la tâche et est définie suivant trois axes :

- la description du comportement des variables caractéristiques,
- l'évolution de la tâche coopérative en phase de réalisation par un groupe d'agent et
- les règles ou les conditions qui définissent le moment d'activation des différentes étapes de réalisation de chaque sous-tâche.

Dans le modèle, quatre grands types de variables caractéristiques sont utilisées : « État », « Statut », « NombreDeRéalisation », « Résultat ».

La variable caractéristique « État » permet de savoir à tout moment à quelle étape de réalisation se trouve une sous-tâche donnée. « Statut » définit quant à elle, dans quelle branche de la tâche coopérative l'activité est en train de se dérouler. « NombreDeRéalisation » est un outil de comptage et donne à tout moment le nombre de fois qu'une sous-tâche donnée a déjà été réalisée. Les variables caractéristiques de type résultat permettent de suivre l'évolution de la réalisation de la tâche coopérative.

Comme nous l'avons vu au niveau de la statique, pour chaque sous-tâche, nous avons trois étapes de réalisation : « Déclenchement », « Réalisation » et « Terminaison ».

L'étape de DÉCLENCHEMENT annonce le commencement de la réalisation de l'action de la tâche. Au niveau des tâches internes elle permet à l'agent exécuteur d'annoncer ses intentions. On peut ainsi connaître à l'avance la sous-tâche qu'il veut exécuter.

L'étape de RÉALISATION (synthèse de résultats) permet l'exécution de l'action associée à la tâche en mettant en œuvre sa méthode de réalisation.

L'étape de TERMINAISON permet de clore l'exécution de l'action de la tâche.

Analyser l'évolution de la tâche coopérative en phase de réalisation par un groupe d'apprenants revient à étudier les affectations des activités de dépendances fonctionnelles

aux différentes étapes de réalisation de sous-tâches. Ces affectations sont faites sur les variables caractéristiques de la tâche coopérative.

À l'étape de déclenchement les activités de dépendances fonctionnelles suivantes sont affectées :

- « État » = *Déclenché*; pour la sous-tâche courante.
- « Statut » = *Suspendu*; pour toutes les sous-tâche où la variable « Statut » contient la valeur *EnCours*.
- « Statut » = *EnCours*; pour toutes les sous-tâches appartenant à l'ascendance de la sous-tâche courante.

À l'étape de réalisation les activités de dépendances fonctionnelles suivantes sont affectées :

- « Etat » = *Réalisé*; pour la sous-tâche courante.
- « NbItération » = « NbItération » + 1; pour la sous-tâche courante.
- « Statut » = *Suspendu*; pour toutes les sous-tâches où la variable « Statut » vaut *EnCours*.
- « Statut » = *EnCours*; pour toute les sous-tâches appartement à l'ascendance de la sous-tâche courante.
- « Résultat » = Valeur quelconque; pour la sous-tâche courante.

À l'étape de terminaison les activités de dépendances fonctionnelles suivantes sont affectées :

- « État » = *Terminé*; pour la sous-tâche courante.
- « Statut » = *Suspendu*; pour toutes les sous-tâches où la variable « Statut » = *EnCours*.
- « Statut » = *EnCours*; pour toutes les sous-tâches appartenant à l'ascendance de la sous-tâche courante.
- « Statut » = *Terminé*; pour la sous-tâche courante.

Pour contrôler l'exécution de la tâche coopérative, trois bases de règles ont été implantées au niveau de chaque sous-tâche. Une base spécifiant les conditions de déclenchement, une autre, les conditions de réalisation et une dernière, les conditions de terminaison. Ces trois bases ressemblent aux préconditions, invariants et postconditions que l'on trouve dans le domaine des preuves de programmes. Les bases de règles utilisées sont construites sous la forme de conjonctions de disjonctions de termes. Les termes sont des tests sur les états caractéristiques des sous-tâches de la tâche.

Les différentes bases de règles intégrées dans l'arbre de tâche permettent de modéliser le comportement général de la tâche coopérative. À cette étape de la construction on a une base de règle distribuée et hiérarchique. Elle est distribuée dans la mesure où chaque sous-tâche possède une partie de la base. La structure hiérarchique des sous-tâches entraîne la hiérarchisation de la base de règles. Nous allons définir plus formellement le comportement, l'évolution et les règles de la couche de la dynamique.

4.4.4 Définition formelle de la dynamique

Cette description formelle utilise les ensembles et les relations décrites dans la couche statique. Décrire formellement la dynamique revient à décrire :

- le comportement évolutif des différents types de variables caractéristiques,
- l'évolution de la tâche coopérative au cours de sa réalisation,
- la construction des règles déclenchant les événements qui gèrent le moment d'exécution des différentes étapes de réalisation de sous-tâches et
- les fonctions de manipulation de la couche dynamique.

4.4.4.1 Comportement évolutif des types de variables caractéristiques

Pour définir formellement ce comportement nous allons utiliser une version augmentée de la théorie des automates à états finis (ATN). Dans les ATN classiques, les transitions à partir d'un état quelconque sont représentées par des actions. À partir d'un état, toutes les transitions présentes sont activables. Dans notre cas l'automate définit le comportement de la variable alors que les actions (transitions) sont régies par le comportement de la

tâche coopérative. La disponibilité des actions est conditionnée à l'état courant de la tâche et non à sa présence comme transition dans un état de l'automate. Nous allons utiliser une version des ATN qui intègre cette contrainte. L'automate ainsi construit nous permet de spécifier sans ambiguïté les actions qui font évoluer une variable d'un état à un autre.

L'ATN que nous construisons se définit formellement par les éléments suivants :

Un ensemble d'états E_e

Cet ensemble représente les différentes valeurs possibles que peut prendre la variable caractéristique.

Un ensemble d'actions de transition E_{at}

Ces transitions correspondent à des étapes de réalisations de sous-tâche avec pour chacune sa condition d'activation. Il permet de représenter les étapes de réalisation qui lors de leur exécution entraîne un changement d'état sur la variable caractéristique concernée par cet automate.

Une fonction de transition F_t

Elle dépend de l'état de départ et d'une action de transition donnée et produit un ensemble d'états. Elle permet d'identifier le rôle de chaque étape de réalisation dans l'ATN. Cette fonction permet également de construire les transitions de notre automate.

Un état de départ e_d

Il représente l'état de départ de l'automate. Il est également caractérisé par le fait qu'aucune transition n'y aboutit.

Un état terminal e_t

C'est l'état terminal de l'automate. C'est un état qui n'est l'origine d'aucune transition.

Pour chaque variable utilisée dans la description d'une sous-tâche donnée, on doit construire l'automate correspondant. Dans la mesure où deux variables caractéristiques du même type ont le même comportement évolutif et donc le même automate, nous allons donner la spécification des automates pour les quatre types de variables caractéristiques manipulées dans MONACO_T.

Variable caractéristique de type « État » : Soit x la sous-tâche à laquelle appartient la variable caractéristique « Etat ». On a: $E_e = f_{vpvc}(Etat)$ = {Initial, Déclenché, Réalisé, Terminé}.

Soit $f_{dfvc}(x.Etat)$ l'ensemble des adf liées à la variable caractéristique « État » de la sous-tâche x, on définit E_{at} comme suit : $\cup f_{eadf}(y_i)$ où y_i *parcourt* $f_{dfvc}(x.Etat)$

Pour cette variable, la fonction F_t est définie en extension à partir du graphe de transition de l'automate (Figure 16) dont les fonctions de transitions sont détaillées dans le tableau 3.

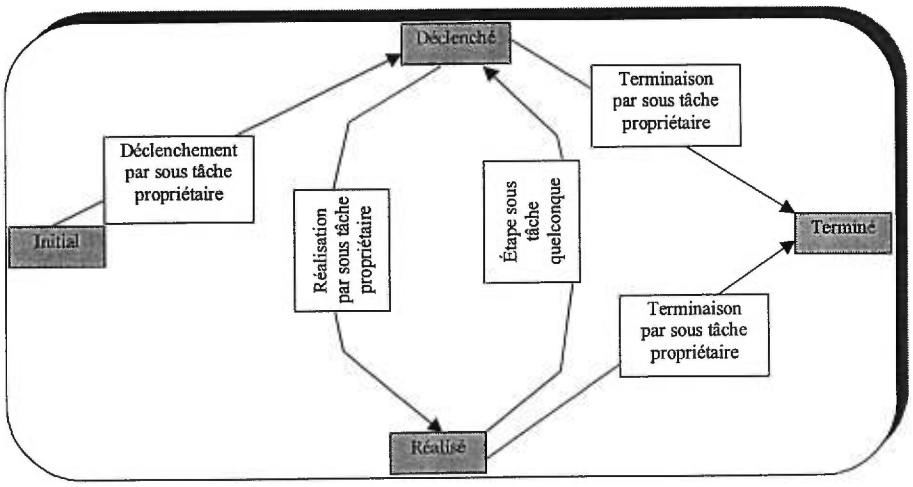


Figure 16 :Graphe de transition de la variable caractéristique « État »

Tableau 3 : Description des actions de transition de la variable « Etat »

État de départ	*État d'arrivée*	*Action de transition*
Initial	Déclenché	Déclenchement de la sous-tâche x
Déclenché	Réalisé	Réalisation de la sous-tâche x
Réalisé	Déclenché	N'importe quelle étape de réalisation de n'importe quelle sous-tâche à l'exception de x. Il peut ne rien exister, dans ce cas la transition sera supprimée

Déclenché	Terminé	Terminaison de la sous-tâche x
Réalisé	Terminé	Terminaison de la sous-tâche x

Variable caractéristique de type « Statut » : Soit x la sous-tâche à laquelle appartient la variable caractéristique « Statut ». On a :

$E_e = f_{vpvc}(Statut)$ = {Initial, EnCours, Suspendu, Terminé}.

Soit $f_{dfvc}(x.Statut)$ l'ensemble des adf liées à la variable caractéristique Statut de la sous-tâche x, alors E_{at} vaut : $\cup f_{eadf}(y_i)$ où y_i *parcourt* $f_{dfvc}(x.Statut)$. E_{at} définit donc l'ensemble des actions de transition de notre automate.

La fonction F_t est aussi définie en extension à partir du graphe de transition de l'automate de la Figure 17 et détaillé dans le tableau 4.

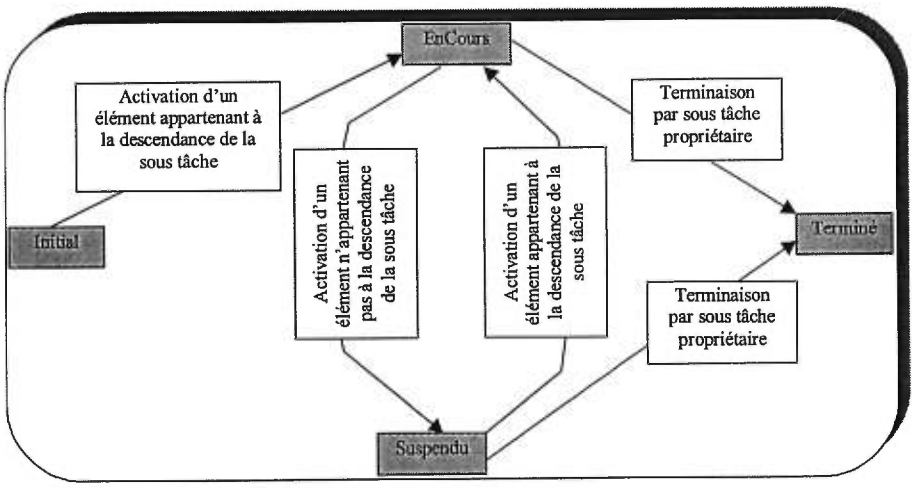


Figure 17 : Graphe de transition de la variable caractéristique Statut

Tableau 4: Description des actions de transition de la variable "Statut"

État de départ	*État d'arrivée*	*Action de transition*
Initial	EnCours	Activation d'une étape de réalisation d'une sous-tâche appartenant à l'ensemble $f_{fdst}(x)$ qui est

		l'ensemble des descendants de la sous-tâche x
EnCours	Suspendu	Activation d'une étape de réalisation d'une sous-tâche n'appartenant pas à l'ensemble $f_{fdst}(x)$ qui l'ensemble des descendants de la sous-tâche x. l'ensemble utilisé est encore appelé complémentaire de $f_{fdst}(x)$ dans E_{st}
Suspendu	EnCours	Activation d'une étape de réalisation d'une sous-tâche appartenant à l'ensemble $f_{fdst}(x)$ qui est l'ensemble des descendants de la sous-tâche x
EnCours	Terminé	Terminaison de la sous-tâche x
Suspendu	Terminé	Terminaison de la sous-tâche x

Variable caractéristique de type « NombreDeRéalisation » : Soit x la sous-tâche à laquelle appartient la variable caractéristique « NombreDeRéalisation ». On a:

$E_e = f_{vpvc}(NombreDeRéalisation)$. C'est donc l'ensemble des valeurs possibles de la variable caractéristique « NombreDeRéalisation ». Elle se définit en compréhension de la manière suivantes : $x / x \in 0,1,2,..., Sup(N)$ *où* $Sup(N) = \max$ *des entiers*.

Soit $f_{dfvc}(x.NombreDeRéalisation)$ l'ensemble des adf liées à la variable caractéristique « NombreDeRéalisation » de la sous-tâche x, alors l'ensemble E_{at} des actions de transitions est égal à : $\cup f_{eadf}(y_i)$ où y_i *parcourt* $f_{dfvc}(x.NombreDeRéalisation)$

Comme dans les cas précédents, la fonction F_t est définie en extension à partir du graphe de la Figure 18 et dans le tableau 5

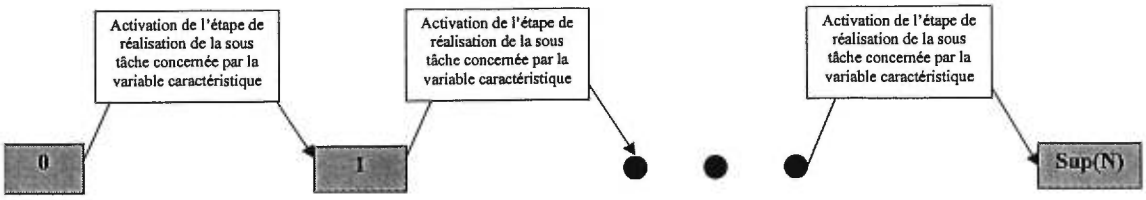


Figure 18 : Graphe de transition de la variable caractéristique « NombreDeRéalisation »

Tableau 5: Description des actions de transition de « NombreDeRéalisation »

État de départ	*État d'arrivée*	*Action de transition*
0	1	Activation de l'étape de réalisation de la sous-tâche x.
1	2	Activation de l'étape de réalisation de la sous-tâche x.
…	…	…
i	i+1	Activation de l'étape de réalisation de la sous-tâche x.
…	…	…
Sup(N)-1	Sup(N)	Activation de l'étape de réalisation de la sous-tâche x.

Variable caractéristique de type « Résultat » : Soit x la sous-tâche à laquelle appartient la variable caractéristique « Résultat1 » (on utilise « Résultat1 » comme identificateur car une sous-tâche peut posséder plusieurs exemplaires de variables de ce type). Comme le comportement de ces variables n'est pas connu d'avance, celui-ci est défini lors de la construction de la tâche coopérative.

Soit $E_e = f_{vpvc}(Résultat1)$ l'ensemble des valeurs possibles de la variable caractéristique « Résultat1 ».

Soit $f_{dfvc}(x.Résultat1)$ l'ensemble des adf liées à la variable caractéristique « Résultat1 » de la sous-tâche x.

On définit alors E_{at} comme $\cup f_{eadf}(y_i)$ où y_i *parcourt* $f_{dfvc}(x.Résultat1)$.

Dans le cadre des variables caractéristiques de type résultat, E_{at} est un singleton dont l'élément est l'étape de réalisation de la sous-tâche x. En d'autres termes, ceci veut dire que seule la sous-tâche propriétaire d'une variable caractéristique de type résultat peut modifier cette dernière.

La fonction F_t est définie en extension par le constructeur de la tâche coopérative. Par défaut, le modèle considère que tout état de l'automate peut être atteint à partir de tout autre état en exécutant l'étape de réalisation de la sous-tâche propriétaire de la variable caractéristique incriminée.

4.4.4.2 Évolution de la tâche coopérative

Définir l'évolution de la tâche coopérative, c'est décrire comment ses variables caractéristiques sont mises à jour en fonction des différentes étapes de réalisation (déclenchement, réalisation, terminaison). Pour chaque étape de réalisation, nous allons uniquement énoncer les affectations qui y sont effectuées.

Évolution à l'étape de déclenchement

Lorsque l'action de déclenchement est effectuée pour une sous-tâche quelconque les affectations suivantes sont réalisées :

- « Etat » = *Déclenché*, pour la sous-tâche concernée.
- « Statut » = *Suspendu*, pour l'ancienne hiérarchie en cours c'est à dire toutes les sous-tâches où le statut a pour valeur *EnCours*.
- « Statut » = *Encours*, pour toutes les sous-tâches appartenant à l'ascendance de la tâche.

Évolution à l'étape de réalisation

Lors de la réalisation d'une sous-tâche donnée les affectations suivantes sont effectuées :

- « Etat » = Réalisé, pour la sous-tâche concernée.

- « NbItération » = « NbItération » + 1, pour la sous-tâche concernée.
- « Statut » = *Suspendu*, pour l'ancienne hiérarchie en cours c'est à dire toutes les sous-tâches où le statut a pour valeur *EnCours*.
- « Statut » = *Encours*, pour toute les sous-tâche appartement à l'ascendance de la tâche courante.

Les variables caractéristiques de type résultat de cette sous-tâche sont mises à jour en fonction de la sémantique de la tâche coopérative traitée.

Évolution à l'étape de terminaison

Les affectations suivantes sont effectuées dans ce cas:

- « Etat » = *Terminé*, pour la sous-tâche courante.
- « Statut » = *Terminé*, pour la sous-tâche courante.

4.4.4.3 Règles de contrôle des activités

Les règles de contrôle des activités sont constituées de deux parties : une partie condition et une partie action. La partie condition permet de définir le moment de déclenchement des événements qui rendent possible la réalisation de la partie action. Elle prend la forme d'une disjonction de conjonctions de termes. Quant à la partie action, celle-ci correspond à une étape d'exécution d'une sous-tâche donnée.

La dynamique se définie donc formellement par la construction de trois grands ensembles et de la distribution des éléments de ces ensembles dans les différentes sous-tâches de la tâche coopérative. Ces trois ensembles sont: E_t, l'ensemble des termes, E_c, l'ensemble des conjonctions et E_d, l'ensemble des disjonctions.

L'ensemble des termes

Un terme est un triplet défini à partir des informations de la couche statique.

Soit $x \in E_{st}$, soit $vc \in E_{vc}$, on note $x.vc$ la variable caractéristique vc de la sous-tâche x. Soit $oc \in E_{oc}$ tel que $vc\ P_{oc}\ oc$, soit $vp \in E_{vp}$ tel que $vc\ P_{vp}\ vp$ et soit $y \in E_{st}$, dans ces conditions, les triplets $(x.vc, oc, vp)$ et $(x.vc, oc, y.vc)$ représentent les deux types de termes que l'on peut rencontrer au niveau de la couche de la dynamique. Le premier

représente la comparaison d'une variable caractéristique avec une valeur résultat possible. Le deuxième représente la comparaison d'une variable caractéristique d'une sous-tâche avec la même variable d'une autre sous-tâche.

L'ensemble des conjonctions

Littéralement une conjonction se définit comme un ensemble de termes reliés entre eux par des connecteurs logiques ET. Il faut que cette conjonction soit vérifiée pour que l'action associée soit activable.

$$c \in E_c \Leftrightarrow \quad c \text{ est de la forme } (t_1 \cap t_2 \cap ... \cap t_n) \text{ où } t_k \in E_t \text{ pour } k = 1..n$$

L'ensemble des disjonctions

Cet ensemble est composé de disjonctions de conjonctions. Ses éléments sont de la forme :

$$d \in E_d \Leftrightarrow \quad d \text{ est de la forme } (c_1 \cup c_2 \cup ... \cup c_n) \quad \text{avec } c_k \in E_c \text{ pour } k = 1..n$$

Une disjonction de conjonctions de termes affectée à une étape d'exécution d'une sous-tâche peut être ainsi considérée comme une base de règles ayant en commun la même partie action.

Chaque élément de E_{st} (l'ensemble des sous-tâches) contient trois bases de règles qui correspondent au déclenchement, à la composition et à la terminaison de l'action intégrée dans la sous-tâche.

4.4.4.4 Définition des fonctions de manipulation de la dynamique

Les fonctions de manipulation au niveau de la dynamique permettent au cours de la réalisation d'une tâche coopérative par un groupe d'agents humains ou simulés de répondre aux questions telles que:

- Quelles sont les sous-tâches déclenchables pour un poste de travail donné?
- Quelles sont les sous-tâches réalisables pour un poste de travail donné?
- Quelles sont les sous-tâches terminables pour un poste de travail donné?
- Pourquoi une sous-tâche X n'est elle pas activable (déclenchable, réalisable ou terminable) à un moment donné?

- Que faut-il faire pour qu'une sous-tâche X soit activable à un moment donné?

Répondre à ces différentes questions revient à manipuler la base de règles qui a été construite. Pour cela, il faut construire un moteur d'inférence pour une base de règles hiérarchique, distribuée et coopérative. Ce moteur est intégré dans la construction des différentes fonctions de la couche dynamique. Les structures de données exploitées seront détaillées lors de la description des différentes fonctions.

Certaines fonctions de la dynamique font intervenir un nouvel ensemble de données appelé ensemble de degrés de raisonnement ou de degrés de finesse qui permettent d'affiner le résultat de ces fonctions. L'ensemble des degrés de raisonnement E_{dr} manipulés dans notre système est composé des éléments suivants : « réactif », « coopératif simple », « coopératif étendu », « social simple » et « social étendu ».

Le niveau réactif exploite uniquement les règles de déclenchement de la sous-tâche concernée.

Le niveau coopératif simple propage la requête aux sous-tâches dont l'exécution permet de déclencher la sous-tâche courante. La propagation s'arrête lorsque la sous-tâche courante devient exécutable. Le résultat retourné est composé uniquement des sous-tâches que la fonction a trouvé exécutables.

Dans le cas du niveau coopératif étendu, la propagation se fait comme pour le niveau coopératif simple, par contre le résultat est structuré sous forme de plan allant des actions immédiatement disponibles à l'action concernant la sous-tâche initiale.

Le niveau social simple est comparable au niveau coopératif simple à la seule différence qu'il fait la propagation de la requête jusqu'à l'agent chargé d'exécuter une sous-tâche dont les conditions d'activation sont valides.

De même, le niveau social étendu est comparable au niveau coopératif étendu, à la seule différence qu'il fait la propagation de la requête jusqu'à l'agent chargé d'exécuter une sous-tâche dont les conditions d'activation sont valides.

Fonction d'évaluation d'un terme F_{et}

L'évaluation est faite en fonction de l'état courant de la tâche coopérative.

$$F_{et} : E_t \longrightarrow \{vrai, faux\}$$

$$x \longmapsto f_{et}(x)$$

Elle retourne le résultat vrai ou faux selon l'état de la tâche coopérative.

Fonction d'évaluation d'une conjonction F_{ec}

Une conjonction étant constituée de plusieurs termes, cette fonction exploite la fonction d'évaluation d'un terme. Elle évalue les différents termes, et fait une conjonction de l'ensemble des résultats obtenus.

$$F_{ec} : E_c \longrightarrow \{vrai, faux\}$$

$$x \longmapsto f_{ec}(x)$$

Elle retourne le résultat vrai ou faux selon l'état de la tâche coopérative.

Fonction d'évaluation d'une disjonction F_{ed}

Une disjonction étant composé de conjonctions, l'évaluation d'une disjonction combine le résultat de l'évaluation de ses différentes conjonctions.

$$F_{ed} : E_d \longrightarrow \{vrai, faux\}$$

$$x \longmapsto f_{ed}(x)$$

Cette fonction retourne le résultat vrai ou faux selon l'état de la tâche coopérative.

Fonction valeur de validité d'un terme F_{vvt}

Cette fonction permet de trouver la valeur minimale à donner à la variable caractéristique de gauche du triplet décrivant le terme pour que celui-ci devienne vrai. Pour déterminer cette valeur nous exploitons principalement l'opérateur de comparaison et l'automate décrivant le comportement évolutif de la variable. La valeur minimum est ainsi établie en fonction du nombre de transitions qu'il faut effectuer dans le graphe de l'automate pour l'obtenir à partir de l'état actuel. L'automate induit ainsi un ordre partiel sur les valeurs possibles. Plusieurs valeurs minimum peuvent être trouvées. Il est également possible qu'aucune valeur ne résolve le problème. Cette possibilité arrive dans le cas où il n'existe

pas dans le graphe de transition de chemin allant de l'état actuel de la variable caractéristique à un état tel que le terme soit vrai.

Exemple1 :

Soit le terme « Diagnostic.Etat = *Terminé* ».

Soit un état de la tâche coopérative tel que la variable caractéristique « Diagnostic.Etat » contienne la valeur *Déclenché*.

Dans ce cas, en fonction du graphe de transition de la variable caractéristique « Etat », la seule valeur qui permette de faire passer le terme de la valeur de vérité faux à vrai est la valeur *Terminé*.

Exemple2 :

Soit le terme « Diagnostic.Etat ≠ *Déclenché* ».

Soit un état de la tâche coopérative tel que la variable caractéristique « Diagnostic.Etat » contienne la valeur *Déclenché*. Cet état de notre tâche donne la valeur de vérité faux à notre terme.

Dans ce cas le graphe de transition de la variable caractéristique « Etat » nous montre que les valeurs *Réalisé* et *Terminé* permettent de faire passer le terme de la valeur de vérité faux à vrai en un minimum de transitions. Dans notre cas ce nombre de transition est 1.

$$F_{vvt} : E_t \longrightarrow P(E_{vp})$$

$$x \longmapsto f_{vvt}(x)$$

Le résultat produit par cette fonction est donc l'ensemble des valeurs qui peuvent être atteintes en un minimum d'activités.

Fonction distance de validité d'un terme F_{dvt}

Cette fonction permet de définir les suites des activités de dépendances fonctionnelles qu'il faut exécuter pour permettre à la variable caractéristique de gauche d'un terme d'obtenir une valeur qui rende vrai le terme. Cette fonction prend donc en entrée un terme dont l'évaluation actuelle donne la valeur faux et retourne l'ensemble des suites d'activités de dépendances fonctionnelles qui rendent vrai ce terme.

Exemple :

Soit le terme « Diagnostic.Etat » = *Terminé*.

Soit un état de la tâche coopérative tel que la variable caractéristique « Diagnostic.Etat » contienne la valeur *Déclenché*. Cet état de la tâche fait en sorte que notre terme prenne la valeur de vérité faux.

Dans ce cas en fonction du graphe de transition de la variable caractéristique « Etat » nous devons trouver toutes les suite d'adfs qui nous permettent de passer de l'état *Déclenché* à l'état *Terminé*. Les suites d'actions que nous pouvons déduire sont les suivantes :

- (*Déclenché* vers *Terminé*)
- (*Déclenché* vers *Réalisé*; *Réalisé* vers *Terminé*)
- (*Déclenché* vers *Réalisé*; *Réalisé* vers *Déclenché*; *Déclenché* vers *Terminé*)

Ces trois suites représentes les différentes distances à parcourir afin de faire passer le terme de la valeur de vérité faux à vrai.

$$F_{dvt} : E_t \longrightarrow P(E^*_{adf})$$

$$x \longmapsto f_{dvt}(x)$$

Le résultat produit par cette fonction est donc l'ensemble de suites d'activités de dépendances fonctionnelles permettant de faire passer la validité du terme x de faux à vrai.

Fonction sous-tâche déclenchable F_{std}

En fonction de l'évaluation de règles de déclenchement des sous-tâches de la tâche coopérative, elle fait ressortir les sous-tâches dont l'évaluation retourne la valeur de vérité vrai.

$$F_{std} : E_{pt} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{std}(x)$$

Cette fonction produit comme résultat l'ensemble des sous-tâches déclenchables par le poste de travail x.

Fonction sous-tâche réalisable F_{str}

Identique à la fonction sous-tâche déclenchable, à la seule différence que les règles à évaluer sont celles de réalisation.

$$F_{str} : E_{pt} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{str}(x)$$

Le résultat produit par la fonction est l'ensemble des sous-tâches réalisables par le poste de travail x.

Fonction sous-tâche terminable F_{stt}

Cette fonction permet d'obtenir les sous-tâches dont les conditions de terminaison sont respectées (valeur de vérité est vraie).

$$F_{stt} : E_{pt} \longrightarrow P(E_{st})$$

$$x \longmapsto f_{stt}(x)$$

Cette fonction retourne comme résultat l'ensemble des sous-tâches terminables par le poste de travail x

Fonction raison de non déclenchabilité d'une sous-tâche F_{rnd}

Cette fonction permet d'obtenir les raisons pour lesquelles une sous-tâche donnée n'est pas déclenchable.

Le moteur d'inférence que nous avons intégré dans notre modèle pour générer ce type d'explications exploite deux types d'arbres ET/OU. Un arbre problème(figure 19) et un arbre solution (figure 20). Le premier permet de trouver une étape de la réponse à fournir, alors que le second permet de générer toutes les réponses possibles à la question posée.

Pour répondre à une question du style : « Que faut-il faire pour rendre déclenchable une sous-tâche? », l'arbre problème est construit de la manière suivante :

Détecter la disjonction de conjonctions de termes fausse qui fait en sorte que les conditions de déclenchement de la sous-tâche concernée ne soient pas vérifiées. La racine est donc un nœud OU qui relie les conjonctions et les fils sont des nœuds ET qui relient les termes.

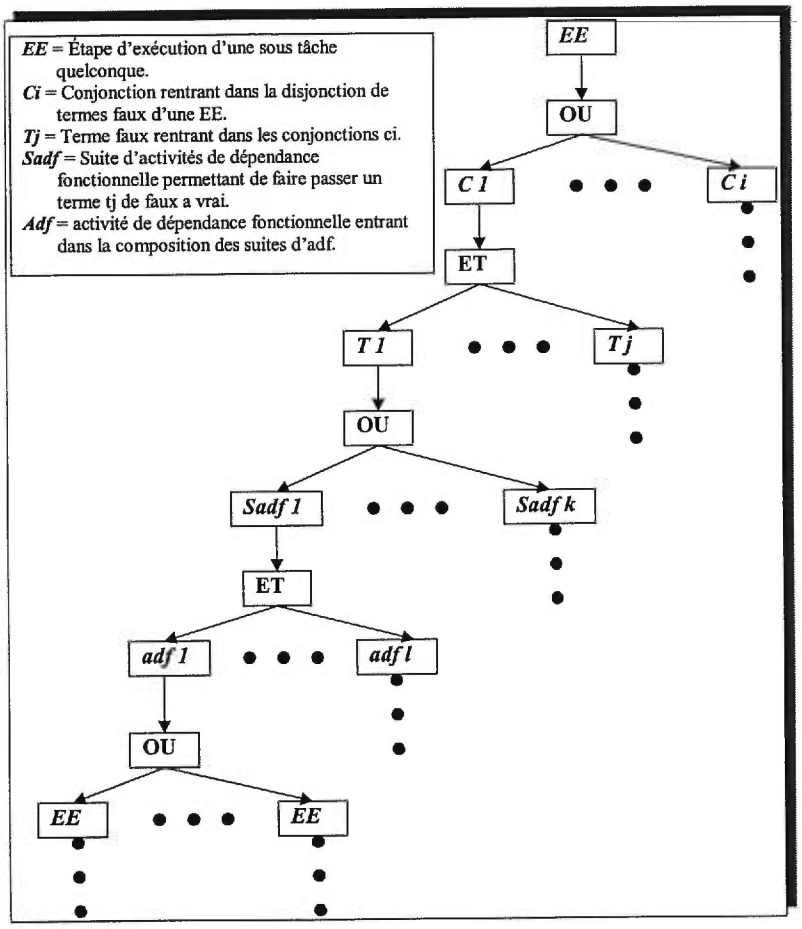


Figure 19: Construction d'une étape de l'arbre ET/OU problème

Pour chaque terme faux, retrouver les différentes suites d'activités de dépendance fonctionnelle (adf) qu'il faut réaliser pour le rendre vrai. Pour cela il faut calculer la distance qui sépare le terme faux de ses valeurs vraies les plus proches. Ceci nous permet de construire un nœud ET pour chaque suite d'adfs. Lorsque l'on est en présence d'un ensemble de suites d'adfs, les différentes suites sont reliées entre elles par un nœud OU.

Pour chaque adf trouvée, rechercher dans la tâche coopérative les étapes d'exécution de sous-tâches qui permettent d'activer ces adfs. Chaque adf donne lieu à un nœud OU reliant les différents étapes d'exécution des sous-tâches qui permettent d'activer l'adf concernée.

Pour chaque étape d'exécution de sous-tâche trouvée, si les conditions d'activation ne sont pas vérifiées alors générer la disjonction de conjonctions de termes fausse qui est en cause. Si le raisonnement est de type social, propager la question à l'agent chargé de l'exécution de cette sous-tâche sinon arrêter la propagation sur ce nœud et passer au nœud suivant.

La construction d'une branche de cet arbre s'arrête lorsqu'on trouve une action qui est activable. Le fait d'activer cette action peut modifier toutes la structure de la tâche coopérative. Ce qui implique que le reste de l'arbre problème ne soit plus valide. Ainsi l'action trouvé est une étape de l'explication à produire. Pour avoir toutes les étapes, il faut à chaque fois construire un nouvel arbre problème à partir de la question initiale. On arrête le processus lorsque l'arbre problème généré est une racine sans feuilles. Ce processus permet de construire l'arbre résultat.

À chaque réalisation d'une étape d'exécution de sous-tâche, nous obtenons un nouvel état de la tâche coopérative. À chaque nouvel état de la tâche coopérative, nous avons un nouvel arbre problème ET/OU.

L'arbre de simulation (arbre résultat) à utiliser est un arbre où les nœuds sont constitués par les états successifs de la tâche coopérative après la réalisation des étapes de réalisation disponibles aux feuilles de l'arbre problème ET/OU.

Les fonctions ainsi construites peuvent être exécutées de manière centralisée où distribuée. Lorsque l'exécution est distribuée, les agents chargés des différents rôles de la tâche coopérative s'occupent des explications à partir des sous-tâches qui leur appartiennent [Tadie & al., 96c].

Certaines fonctions de la dynamique font intervenir un nouvel ensemble de données appelé ensemble de degrés de raisonnement ou de degrés de finesse qui permettent d'affiner le résultat des fonctions qui les utilisent. L'ensemble des degrés de

raisonnement E_{dr} manipulés dans notre système est composé des éléments suivants : Réactif, coopératif simple, coopératif étendu, social simple et social étendu.

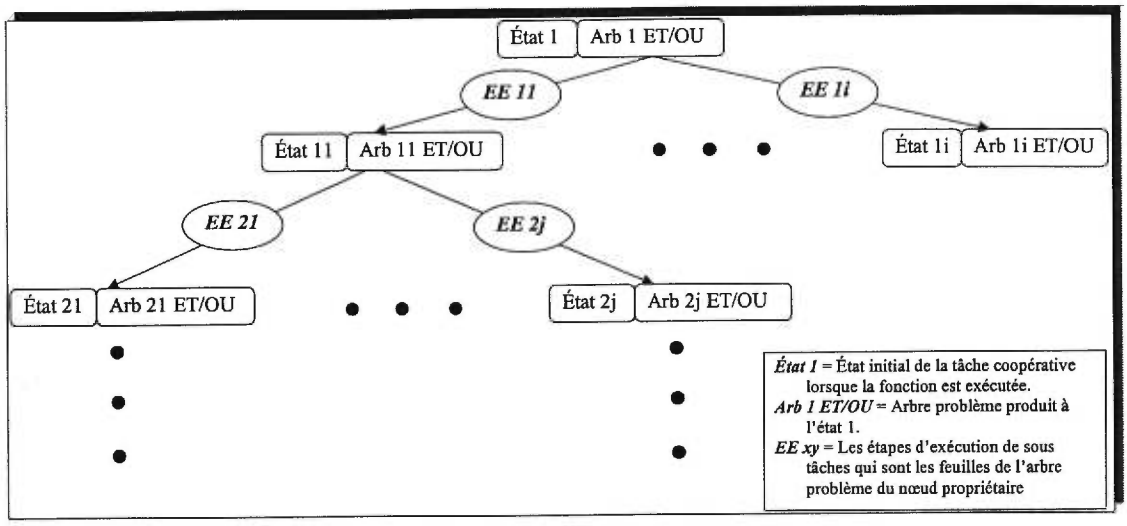


Figure 20 : Structure de l'arbre ET/OU résultat

Le niveau réactif exploite uniquement les règles de la sous-tâche concernée. Sans propager la requête aux autres sous-tâches. L'arbre problème généré ne contient alors qu'un seul niveau.

Le niveau coopératif simple utilise l'arbre problème au complet, mais la solution retournée n'est constituée que de la disjonction des actions exécutables. Il n'y a donc pas utilisation de l'arbre solution.

Dans le cas du niveau coopératif étendu, la solution est composée de l'ensemble des chemins de la racine aux feuilles sur l'arbre solution.

Le niveau social simple est comparable au niveau coopératif simple à la seule différence qu'il fait la propagation de la requête jusqu'à l'agent chargé d'exécuter une sous-tâche dont les conditions d'activation sont valides.

De même, le niveau social étendu est comparable au niveau coopératif étendu, à la seule différence qu'il fait la propagation de la requête jusqu'à l'agent chargé d'exécuter une sous-tâche dont les conditions d'activation sont valides.

Fonction raison de non réalisation d'une sous-tâche F_{rnr}

Cette fonction est identique à la fonction raison de non déclenchabilité, à la seule différence que la base de règles utilisée est celle qui est incluse dans l'étape d'exécution appelée réalisation.

Fonction raison de non terminaison d'une sous-tâche F_{rnt}

Cette fonction est identique à la précédente à la seule différence que la base de règles utilisée est celle de l'étape terminaison.

Fonction solution pour déclenchabilité d'une sous-tâche F_{spd}

Cette fonction permet de déterminer la suite d'actions à effectuer afin qu'une sous-tâche s puisse être déclenchable. Plus concrètement il s'agit de trouver la suite d'étapes d'exécution de sous-tâches à réaliser afin que l'évaluation des conditions de déclenchement de s puisse retourner comme résultat la valeur vrai.

Le principe est le même que dans le cas des fonctions de non déclenchabilité, à la seule différence qu'ici le résultat à produire est une disjonction de conjonctions d'étapes d'exécution de sous-tâches. On exploite donc toujours le même arbre problème multi-nœuds "ET/OU" mais lors de la compilation du résultat, seuls les nœuds de type étape d'exécution de sous-tâche sont récupérés.

Fonction solution pour réaliser une sous-tâche F_{spr}

Cette fonction est identique à la fonction F_{spd} à la seule différence que les conditions utilisées proviennent de l'étape d'exécution « Réalisation ».

Fonction solution pour terminer une sous-tâche F_{spt}

Dans ce cas les conditions utilisées proviennent de l'étape d'exécution « Terminaison ».

4.4.4.5 Remarques

Toutes ces fonctions utilisent le graphe de la statique (qui fournit la base des faits nécessaires à l'inférence) augmenté des règles de la dynamique pour produire les résultats.

Les résultats produits lors des appels effectués dépendent du moment où la fonction est appelée. Cette volatilité est due au fait que la base des faits exploités par le moteur

d'inférence de MONACO_T évolue en fonction des actions des agents chargés de réaliser la tâche coopérative. La même fonction appelée à deux instants i et j peut produire deux résultats différents.

Les règles de la dynamique utilisées sont des règles évolutives dans le temps en fonction de la couche scénario que nous verrons plus loin.

Le graphe de la dynamique ainsi construit est isomorphe au graphe de la statique. En pratique les deux graphes théoriques sont réunis en un seul graphe où les sous-tâches contiennent les informations de la statique et les règles qui définissent la dynamique. Cette modélisation de la dynamique fait ressortir plusieurs avantages .

- La flexibilité dans la définition du comportement de chaque sous-tâche, qui est due au fait que chaque sous-tâche ne définit que son comportement propre et non celui de ses fils ou de ses descendants.
- On peut spécifier de façon assez complète par les conditions de déclenchement de composition et de terminaison, l'interface qu'une sous-tâche a avec les autres. Cette spécification joue un rôle important dans le cadre du travail coopératif.
- L'intervention possible de toutes les sous-tâches de l'arbre de tâche dans la définition des conditions d'une sous-tâche quelconque, permet de représenter la plupart des contraintes se produisant dans le monde réel.
- L'utilisation de toute la puissance expressive de la logique des prédicats pour spécifier le comportement de la tâche, permet à MONACO_T d'être particulièrement adapté à la représentation des dépendances entre les sous-tâches.

L'un des problèmes que pose la modélisation de tâche avec des règles est que la saisie est relativement compliquée mais surtout difficile à contrôler. Pour pouvoir résoudre ce problème, nous avons construit le module de saisie interactive de règles (figure 21), qui permet à tout utilisateur d'entrer les règles qui régissent le comportement d'une tâche en phase d'exécution avec une interface conviviale produisant des règles sûres dans la mesure où le choix des nœuds, des états et des opérateurs est contrôlé par un module de saisie.

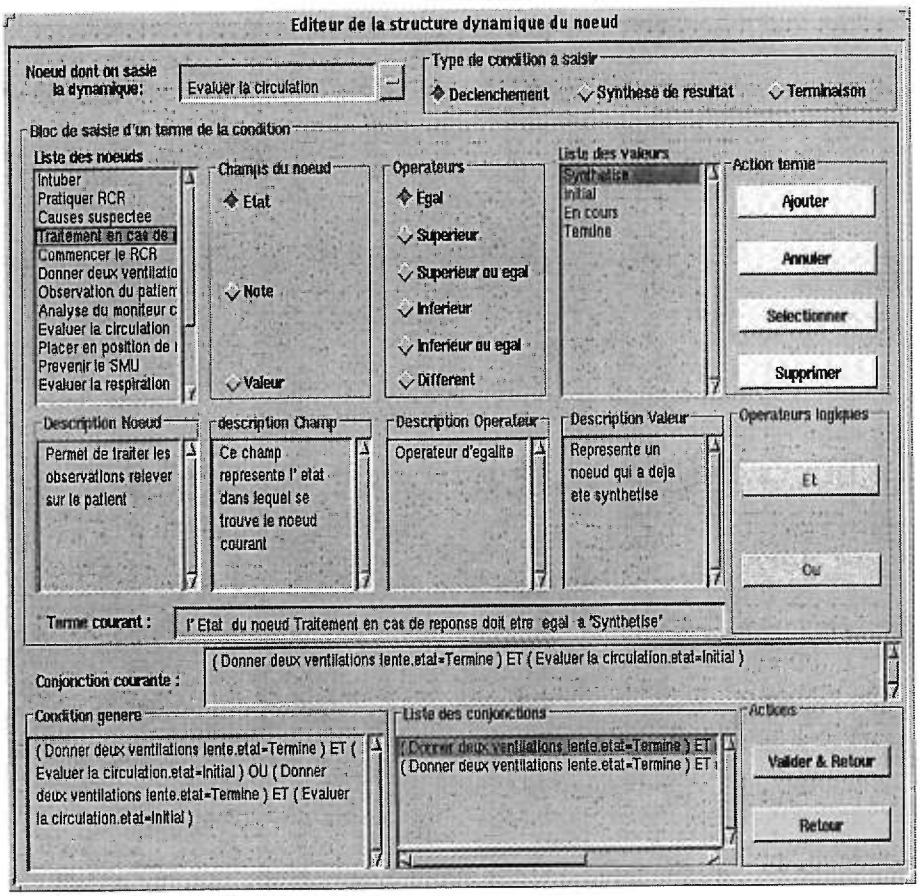


Figure 21 : Écran de création des règles définissant la dynamique d'une sous-tâche

4.4.5 Description de la couche scénario

Elle permet de définir des stratégies de réalisation de la tâche coopérative. Il existe deux familles de stratégies possibles : les scénarios coopératifs et ceux par poste de travail.

Les recherches dans le domaines des scénarios permettent de trouver des modèles de descriptions partielles du comportement d'un système informatique [Some 97]. Cette définition permet de d'implanter les différentes fonctions d'un système. La plupart des langages de représentation de scénario sont basés sur le formalisme MSC(Message Sequence Charts)[ITU-T 93]. Ce langage est adapté pour les applications interactifs ne faisant intervenir qu'un utilisateur et un système informatique. Dans le cas des systèmes coopératifs tel que nous l'envisageons, où l'interaction est le résultat des actions de plusieurs agents, il devient difficile de représenter touts les scénarios d'une stratégie d'exécution de la tâche coopérative et ceci pour chaque poste de travail. Pour prendre en

compte les scénarios par poste de travail nous avons défini un modèle qui n'exige pas la représentation des actions de tous les autres postes. Ce modèle nous permet également d'éviter toutes les représentations explicite d'une stratégie de réalisation de la tâche coopérative.

Les scénarios coopératifs permettent de spécifier des démonstrations intégrant tous les postes de travail impliqués dans la réalisation de la tâche coopérative. La structure est assez simple. Il s'agit d'un planning figé des sous-tâches à réaliser par l'ensemble des postes de travail. Formellement un scénario coopératif est un ensemble d'actions ordonné suivant le moment d'exécution. La structure d'une action est composée au minimum des informations suivantes :

- Poste de travail.
- Sous-tâche.
- Étape d'exécution.
- Moment d'exécution.
- Liste d'activités de dépendances fonctionnelles à propager.

Les actions sont triées suivant le moment d'exécution, et lorsque deux actions ont le même moment d'exécution, elles peuvent être exécutées en parallèle ou l'une à la suite de l'autre.

Les scénarios par postes de travail ou scénarios individuels permettent quant à eux de donner un comportement précis à un agent informatique chargé de simuler les activités d'un poste de travail donné. Ils permettent également à la tâche coopérative de contrôler le travail d'un agent humain chargé de ces mêmes activités suivant une stratégie précise. Plus généralement, ce type de scénario permet de spécifier le comportement d'un poste de travail par rapport à la tâche coopérative toute entière.

Ce comportement est défini en spécifiant les attitudes que l'agent chargé du poste de travail doit avoir en fonction de l'état général de la tâche coopérative. Modéliser le scénario individuel revient à trouver une représentation aux deux concepts suivants :

- les actions correspondant aux attitudes que doit avoir l'agent chargé du poste de travail,
- les conditions sous lesquelles ces actions doivent être exécutées.

Le couplage des conditions et des actions ci-dessus mentionnées forme des règles. Les scénarios individuels sont donc définis comme un ensemble de règles qui viennent modifier en temps réel les informations de la statique et les règles qui définissent la dynamique.

Les conditions de ces règles sont construites exactement de la même façon que pour celles de la couche de la dynamique.

Ce modèle de scénario permet de construire une planification dynamique et distribuée de la réalisation d'une tâche coopérative. Le but de ces scénarios est de permettre à l'agent concerné de sélectionner une seule action à faire parmi toutes celles qui auraient été possibles en fonction de la dynamique originelle de la tâche coopérative.

Plus formellement, un scénario par poste de travail est composé de plusieurs ensembles de règles. On peut par exemple avoir les trois règles suivantes :

- les règles d'ordonnancement,
- les règles de coupure de sous-tâches et
- les règles de coupure de termes.

Plus généralement, une règle de la couche scénario peut être définie comme une contrainte que l'on voudrait ajouter au niveau de la dynamique ou de la statique en fonction d'un état particulier de la tâche coopérative. Les trois types de règles énoncées ne sont pas exhaustives.

4.4.5.1 Règle d'ordonnancement

Elles permettent en fonction d'un contexte défini par la prémisse de la règle de proposer un ordonnancement des actions de la tâche coopérative. L'ordonnancement de l'arbre de tâche est réalisé par un parcours infixe (Racine, Fils de gauche à droite). Ainsi, si un

agent simule les actions de ce poste en fonction du contexte de réalisation de la tâche coopérative, il n'y aura pas d'ambiguïté dans le choix de l'action à réaliser.

4.4.5.2 Règle de coupure de sous-tâches

Elles permettent en fonction d'un contexte défini par la prémisse de la règle de désactiver des sous-tâches que l'on ne voudrait pas voir s'exécuter dans le cadre d'un scénario donné. L'application de la règle permet donc de suspendre temporairement la disponibilité de la sous-tâche à laquelle elle fait référence.

4.4.5.3 Règle de coupure de termes

Elles permettent en fonction d'un contexte défini par la prémisse de la règle de désactiver des termes (des parties de règles) dont on ne voudrait pas qu'ils aient une influence sur la dynamique de la tâche coopérative. Exemple : si pour réaliser une action A la condition d'activation est C1 ou C2, on décide que dans le contexte X on ne veut pas que C1 ait une influence sur l'activation de A, il est alors nécessaire de construire une règle de coupure de terme et l'intégrer dans le scénario concerné.

4.5 Caractéristiques coopératives de MONACO_T

Lorsqu'on parle d'un modèle, ou d'un système coopératif, il est bon de faire ressortir les points suivants: présence de plusieurs agents, objectif commun, mode de coopération, système de communication, gestion du dialogue, synchronisation des activités. Parler des caractéristiques coopératives de MONACO_T, c'est expliquer comment il intègre ces aspects majeurs de la coopération.

MONACO_T est un modèle qui représente une tâche répartie sur plusieurs postes de travail, chacun d'eux étant occupé par un agent.

La définition d'un objectif commun est donc déduite du fait que tous les agents sont chargés de réaliser la même tâche.

Le mode de coopération désigne la manière dont les acteurs se répartissent la tâche afin de la résoudre. Trois modes de répartition sont possibles: figé (fixé à l'avance),

dynamique (précisé lors de l'exécution), hybride (fixé au début et modifié lors de l'exécution). Dans MONACO_T la répartition est définie à la création de la statique, par l'attribution de sous-tâches aux acteurs. Ce choix correspond bien à la réalité des tâches coopératives où les différents postes de travail sont prédéfinis. Lorsqu'on est en présence d'une tâche coopérative où les postes de travail sont distribués de manière dynamique au cours de son exécution, il suffit d'ajouter une nouvelle famille de règles dans la couche scénario appelées règles de distribution de sous-tâche. Ainsi donc en fonction de certaines conditions on pourra modifier la distribution des sous-tâches aux postes de travail. Les fonctions implantées dans les trois couches du modèle s'adapterons automatiquement à la nouvelle distribution des sous-tâches dans la mesure où elles sont toujours évaluées en temps réel. En définitif, le mode de répartition dans notre modèle est hybride.

MONACO_T implante le modèle de communication par tableau noir. Celui-ci est constitué de l'ensemble des valeurs contenues dans les nœuds de la tâche. Chaque cellule du tableau noir a un propriétaire (un des acteurs de la tâche) et seul ce propriétaire peut mettre à jour l'information dans la cellule, les autres n'ont que la possibilité de lecture. Avec ce choix, les problèmes d'accès concurrents sont éliminés. Lorsqu'on opte pour une distribution de la tâche à des agents, la communication peut se faire par échanges de messages entre agents.

Dans MONACO_T, le dialogue entre les acteurs se fait par l'intermédiaire des modifications que les actions entraînent dans l'environnement de travail et par la prise en considération de ces modifications par les bases de règles.

La synchronisation dans MONACO_T est régie par le modèle de la dynamique. La puissance expressive qu'offre ce modèle par l'utilisation de la logique des prédicats, fait en sorte que toutes les contraintes modélisables à l'aide de ce type de logique peuvent être incorporées.

4.6 Conclusion.

La modélisation que nous proposons est structurée en couches (statique, dynamique et scénario). Chaque couche a une structure arborescente. Les nœuds de la dynamique possèdent des règles qui permettent de définir le comportement global de la tâche.

Contrairement aux autres modèles de tâche, qui exploitent uniquement les avantages d'une seule technique de représentation de connaissances ou au maximum deux comme le modèle PIF [Djamen, 95], Monaco-T tire les avantages de ces trois techniques: puissance expressive de la logique, mise en valeur des niveaux d'abstractions par le découpage en couches, bonne représentation cognitive de la tâche par les graphes. La couche scénario permet de modifier dynamiquement les règles de la dynamique et la répartition des informations de la statique.

Les sous-tâches internes ou abstraites sont le siège d'actions et peuvent être exécutées contrairement à certains modèles qui font la différence entre les sous-tâches abstraites et les sous-tâches opérationnelles. Ainsi dans ce modèle, lorsqu'un apprenant veut réaliser une tâche, il doit en maîtriser la structure abstraite alors que dans une structure où les actions se situent dans les nœuds opérationnels, l'apprenant peut exécuter une suite d'actions permettant de réaliser la tâche sans pour autant en connaître la structure abstraite.

Ce modèle a d'autres avantages tels que:

- La flexibilité dans la définition du comportement de chaque sous-tâche, dans la mesure où chaque sous-tâche ne définit que son comportement propre et non celui de ses fils ou de ses descendants.
- On peut spécifier de façon assez complète par les conditions de déclenchement de composition et de terminaison, l'interface qu'une sous-tâche a avec les autres. Cette spécification joue un rôle important dans le cadre du travail coopératif.
- L'intervention possible de toutes les sous-tâches de l'arbre de tâche dans la définition du comportement d'une sous-tâche quelconque, permet de représenter la plupart des contraintes se produisant dans le monde réel.
- L'utilisation de toute la puissance expressive de la logique des prédicats pour spécifier le comportement de la tâche, permet à MONACO_T d'être particulièrement adapté à la représentation des dépendances entre les sous-tâches.
- La structure coopérative de la tâche est représentée d'une manière assez simple par l'attribution de sous-tâches à des postes de travail. Lorsqu'on a un seul poste de

travail dans le modèle d'une tâche, on est en présence d'une tâche individuelle. MONACO_T est donc également capable de modéliser les tâches individuelles.

MONACO_T s'est avéré assez robuste pour la construction d'un conseiller coopératif pour tâche coopérative [Tadié & al., 97a, Tadié & al., 98b]. Il nous a également permis de générer des agents conseillers coopératifs et des agents simulateurs de tâche [Tadié & al., 97b]. Pour permettre une utilisation simple de ce modèle, nous avons construit un environnement d'édition [Tadié & al., 98a] où certains processus d'édition sont automatisés. Le futur de nos travaux s'oriente vers la coopération à travers le WEB, plus précisément il s'agit de la construction d'un STI distribué pour l'enseignement de tâche coopératif fonctionnant sur le WEB.

5 EDER_TC : Environnement Distribué pour l'Enseignement et la Simulation de Tâches Coopératives

L'architecture que nous voulons construire peut être considérée comme étant un CMCW (Computer Managed Cooperative Work) permettant de faire l'enseignement et la réalisation de tâches coopératives. Elle permettra d'aller au-delà des outils d'assistance à la réalisation de tâche coopérative en intervenant directement dans la sémantique des tâches à exécuter. Nous allons commencer par présenter les caractéristiques que le système EDER_TC doit comprendre. Celles-ci nous permettront de mettre en exergue les problèmes que nous devons adresser dans la construction de notre architecture. Nous allons ensuite proposer l'architecture qui nous permet de répondre à toutes ces attentes. Dans la mesure ou cette architecture utilise une approche multi-agents, nous allons dans la troisième partie de ce chapitre décrire la structure des agents qui seront exploitées dans EDER_TC. Enfin, comme l'un des objectifs fondamentaux de notre architecture est d'enseigner la réalisation de tâches coopérative, nous allons utiliser un exemple de ce qu'est une tâche coopérative pour montrer comment notre système peut générer automatiquement des explications contextuelles en faisant intervenir la coopération entre plusieurs agents.

5.1 Introduction

Les objets MONACO_T étant construits, nous devons mettre sur pied une architecture (EDER_TC) permettant à des agents humains ou artificiels de simuler ou d'apprendre la réalisation d'une tâche coopérative. Les caractéristiques que cette architecture doit avoir sont les suivantes :

- Permettre à des agents physiquement distants de travailler sur une tâche coopérative en temps réel.

- Comprendre la sémantique des actions effectuées par un groupe d'agents. Exploiter cette compréhension pour intervenir non seulement au niveau du protocole des échanges, mais également au niveau du contenu.
- Permettre à des agents de travailler sur des tâches coopératives ayant un nombre de postes de travail inconnu d'avance.
- Être capable d'intégrer dans l'environnement multi-agent des agents humains et des agents informatiques dans le processus de réalisation ou d'apprentissage d'une tâche coopérative.
- Permettre à une équipe d'agents de types différents de réaliser une tâche coopérative.
- Faire en sorte qu'une équipe de différents types d'agents participe à l'enseignement d'une tâche coopérative.
- Autoriser la réalisation où l'enseignement de divers types tâches coopératives allant des tâches conceptuelles aux tâches de manipulation de systèmes.

Les contraintes de notre architecture nous renvoient à plusieurs domaines de recherche. Il s'agit de l'intelligence artificielle distribuée (DAI), des systèmes supportant le travail coopératif (CSCW), des systèmes multi-agents, et des systèmes tutoriel intelligent (ITS).

Alors que l'intelligence artificielle (IA) se focalise sur la représentation et l'exploitation de l'intelligence d'un humain, les recherche en DAI se penchent quant à elles sur la représentation et l'exploitation de l'intelligence d'un groupe. Cette intelligence touche au raisonnement, à la vision, au langage etc. Selon Gasser [Gasser 92] les systèmes de DAI concernent l'étude et la construction d'automates semi-autonomes qui interagissent entre eux et sur leur environnement. Ces systèmes vont au-delà de l'étude d'individus intelligents résolvant des problèmes individuels, ils possèdent en plus une composante sociale. Ces principes ont été utilisés pour construire :

- les programmes de recherche [Gasser 89, Corkill 79, Ephrati 92],
- les programmes de traitement de la parole et du langage [Erman 80, Cullingford 84, Cohen 79],
- les programmes de contrôle de trafic aérien [Cammarata 83, Findler 86].

Les recherches en DAI nous aident à construire des environnements permettant à des agents automatiques de résoudre un problème donné. Celles-ci permettent d'adresser des problèmes tels que : la communication (langage et protocole d'interaction), la cohérence du comportement d'une équipe chargé de résoudre un problème et la modélisation des agents automatiques impliqués dans la résolution.

La résolution de tâches coopératives pouvant être effectuée par une équipe incluant des agents humains, les recherches dans le domaine du CSCW (Computer Supported Cooperative Work) doivent être intégrées dans notre architecture. Ceci consiste à concevoir des outils permettant à des humains de travailler en coopération sur un même objet en utilisant l'ordinateur comme borne de communication. Plusieurs chercheurs se sont penchés sur le sujet notamment Ellis et Wainer [94], qui ont proposé une taxinomie de ce type de systèmes et Roseman et Greenber [92] qui ont construit un système de groupeware appelé GroupKit. Le CSCW nous fournit les outils (voir chapitre 3) permettant à des agents humains de communiquer entre eux.

L'architecture que nous mettons en place doit également permettre de produire un enseignement de tâches coopératives. Il n'existe pas encore de systèmes pour l'enseignement de telles connaissances. Les seules recherches pouvant nous être utiles sont celles sur les STI (Systèmes Tutoriel Intelligent). Dans notre cas, le tuteur devra être capable de produire des conseils pour un poste de travail en tenant compte des activités des autres postes [Tadie & al 96c].

Dans un système de travail ou d'enseignement coopératif, les agents doivent avoir conscience de la présence de leurs partenaires d'équipe. Cette conscience permet entre autre d'initier des communications entre agents lors du processus de réalisation d'une tâche coopérative. Cette communication étant à la base du partage de responsabilité, de la résolution de conflit ou tout simplement d'une demande d'assistance. À partir de cette contrainte et des caractéristiques des systèmes de CSCW, DAI et Multi-agents, nous avons opté pour une approche multi-agents pour la représentation d'EDER_TC.

Pour être capable de réaliser toutes les fonctions que nous lui assignons, nous proposons une architecture multi-agents comportant quatre différents types d'agents :

- un agent informatique pour la simulation,

- un agent humain pour la simulation,
- un agent enseignant d'un poste de travail et
- un agent gestionnaire de la tâche coopérative.

Ce dernier est l'élément central de notre architecture. Il permet de gérer toutes les connaissances qui ont été emmagasinées dans l'objet MONACO_T.

5.2 Architecture d'EDER_TC

EDER_TC est une architecture multi-agents permettant l'exploitation des objets MONACO_T et produisant l'enseignement ou la simulation d'une tâche coopérative. Pour expliquer la construction d'EDER_TC, nous allons analyser sa structure et les différents agents qui y participent. Mais avant cela, nous définissons les orientations sociales prises dans l'univers EDER_TC.

5.2.1 Orientation sociale d'EDER_TC

EDER_TC représente une société d'agents. Définir l'orientation sociale de cette société c'est définir son organisation et sa structure de communication.

5.2.1.1 Organisation sociale dans EDER_TC

L'organisation sociale représente la manière dont le groupe est organisé. Deux modes d'organisations sont possibles dans EDER_TC.

Un mode centralisé où les agents reçoivent les directives de comportement d'un agent maître (agent gestionnaire de la tâche coopérative).

Le second mode est celui où les décisions sont décentralisées et où chaque agent est maître de son propre destin.

Dans le premier mode, le contrôle et la prise de décision sont détenus par l'agent central du système : le gestionnaire de tâche coopérative (TC). Il peut quelques fois déléguer des choix mineurs au niveau des autres agents du système. En mode décentralisé, la prise de décision est uniquement de la responsabilité des agents participants à la réalisation d'une

TC. Le gestionnaire de TC dans ce cas se contente de fournir le support de communication permettant la coopération entre agents.

5.2.1.2 Communication

Étudier la communication dans un système multi-agents c'est définir le langage de communication, le protocole de communication, l'architecture de communication, le mode de communication et la structure des messages utilisés.

Dans un univers multi-agents, les différents agents doivent être capables de se comprendre. Pour cela ils ont besoin de s'entendre sur un langage de communication unique accessible et compréhensible par tout le monde. Dans EDER_TC, le langage est constitué des actions spécifiées dans l'objet MONACO_T exploité par le système multi-agents. Les primitives de ce langage sont représentées par les activités de déclenchement, de réalisation et de terminaison des sous-tâches de la tâche coopérative à un moment donné.

Le protocole de communication décrit quant à lui le moment de prise de parole de chaque agent participant à EDER_TC. Lorsqu'on est en présence d'un agent humain, ce protocole est intuitif et se déduit de la sémantique des échanges. Lorsqu'on est présence d'un agent informatique, il faut trouver des protocoles plus robustes. Ceci permet d'éviter des situations de famine où un agent peut être empêché d'intervenir alors que son apport pourrait être capital dans la résolution du problème collectif. Dans EDER_TC, nous avons choisi de régir le protocole de communication par les règles de la couche dynamique et celles de la couche scénario du modèle de tâche exploité (MONACO_T). Ainsi si deux actions appartenant à deux agents différents sont disponibles pour exécution, alors ces deux actions peuvent être activées en parallèle ou en séquence. De ce fait les agents dans EDER_TC n'ont plus à se préoccuper du protocole de communication. Lorsqu'un agent veut intervenir dans la résolution de la tâche coopérative, il le fait sans autres formes de procès.

L'architecture de communication fait ici référence à structure logique du réseau de communication entre les agents d'EDER_TC. En général on peut avoir quatre types de structures.

Une structure en anneau : Dans ce cas, les connections entre tous les agents forment un anneau. Ainsi lorsqu'un message veut partir d'un agent A1 vers un agent An, il passe par tous les agents intermédiaires.

Une structure en étoile : Dans ces cas il y a un agent central qui se charge de recevoir les messages et les transmet aux destinataires.

Une structure de semi-maillage : Ici chaque agent est directement lié à un certain nombre d'autres agents de telle sorte qu'on puisse toujours trouver un chemin partant de n'importe quel agent à n'importe quel autre agent.

Une structure en maillage complet : Chaque agent est directement lié à tous les autres agents de l'univers multi-agents.

Choix d'une architecture de communication : Dans EDER_TC, nous avons choisit la structure en étoile. Cette structure est très adaptée à l'exploitation d'un système multi-agents comportant un agent gestionnaire de la tâche coopérative. Cet agent gère les interactions avec les systèmes que la tâche est censée manipuler, il est donc de ce fait directement lié à tous les autres agents d'EDER_TC. Nous lui avons simplement attribué les tâches de superviseur de la communication.

Le mode de communication représente la technique de notification entre les agents dans un système multi-agents. Deux modes de communication sont usuellement utilisés : la communication par envoi de message et celle par partage d'information (tableau noir). Dans la mesure où les agents dans EDER_TC peuvent être physiquement éloignés, nous avons opté pour une communication par envoi de message.

La structure des message contenant les informations est la suivante :

L'émetteur, C'est un objet complexe décrivant l'agent émetteur du message. Il comprend des informations telles que le type d'agent (humain, automatique ou enseignement), le poste de travail occupé dans la tâche coopérative, le port par où tout autre agent peut communiquer avec lui, etc.

Le destinataire, Il a la même constitution que l'émetteur.

La date d'émission, Représente l'heure à laquelle le message a été envoyé.

Le type de message, Il permet de savoir si c'est une action sur la tâche coopérative ou un message de service.

L'information, Elle contient le message de service ou l'action effectuée par l'agent sur la tâche coopérative et les paramètres correspondants.

5.2.2 Schéma architectural

Cette architecture (figure 22) fait ressortir trois composantes principales :

- l'agent gestionnaire de la tâche coopérative,
- le média de communication utilisé (Internet dans notre cas) et
- un ensemble d'agents chacun gérant un poste de travail de la tâche coopérative.

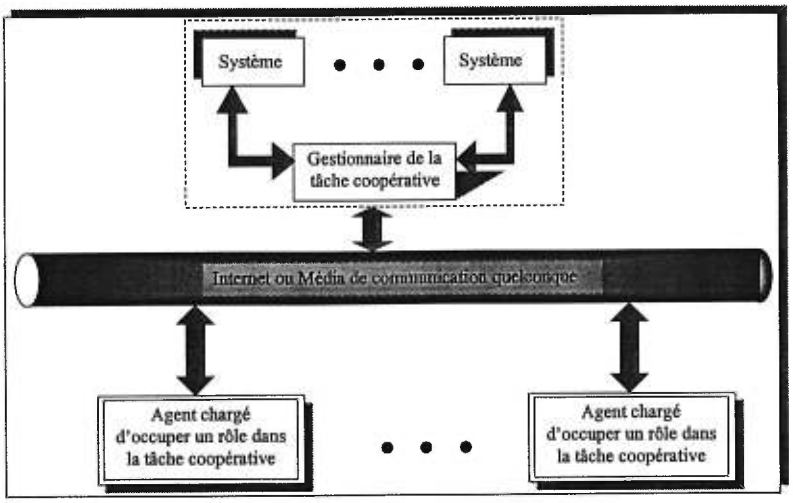


Figure 22: Architecture générale d'EDER_TC

L'agent gestionnaire de tâche coopérative prend en charge les interactions entre le monde extérieur et les objets MONACO_T. Dans la mesure où MONACO_T est un modèle pour l'enseignement et la simulation de tâches coopératives, les interactions font référence aux fonctions suivantes de MONACO_T :

- maintenir l'évolution de la tâche coopérative au cours d'un processus de simulation ou d'enseignement,

- générer des explications en fonction de la hiérarchie de la tâche coopérative, des règles de la dynamique, des règles de la couche scénario et du contexte d'exécution de la tâche,
- propager des actions effectuées par les postes de travail sur les systèmes manipulés et retourner des résultats.

Un agent chargé de la gestion d'un poste de travail est vu dans EDER_TC comme un module informatique chargé d'exécuter les actions d'un poste de travail défini dans la tâche coopérative.

Ce module informatique peut être complètement automatique, dans ce cas il intègre un module de raisonnement lui permettant de planifier et d'exécuter des actions rentrant dans la réalisation de la tâche coopérative. Ce type d'agent est appelé agent automatique pour la simulation.

Ce module peut être totalement manuel, dans ce cas il fonctionne comme une interface et son module de raisonnement est l'usager humain qui est chargé des activités au niveau de ce poste de travail. Ce type d'agent est appelé agent humain pour la simulation.

Ce module peut également représenter l'enseignant d'un poste de travail donné. Dans ce cas, il contient un module de raisonnement permettant de guider un humain qui voudrait apprendre les activités de ce poste de travail. Il joue également le rôle d'une interface en permettant à l'humain d'agir dans l'environnement multi-agents.

De plus les modules de raisonnement utilisés dans les agents d'EDER_TC sont capables d'intervenir suivant plusieurs niveaux (réactif, coopératif, social).

Le niveau réactif permet à l'agent d'utiliser uniquement les connaissances qu'il renferme pour répondre à une requête.

Le niveau coopératif permet d'initier une coopération avec les autres agents participant à EDER_TC pour répondre à une requête.

Le niveau social permet non seulement de faire participer les agents d'EDER_TC, mais aussi les humains qui utilisent EDER_TC pour apprendre à réaliser des tâches coopératives ou ceux qui simulent des tâches coopératives.

En fonction des règles de la dynamique et celle de la couche scénario, les différents agents présents savent quoi dire et quand le dire.

5.2.2.1 Organisation sociale

L'organisation sociale nous permet de décrire comment les agents de notre architecture se regroupent pour pouvoir réaliser les tâches qui leur incombent. Cette organisation sociale dépend du type des agents que nous utilisons. Dans le cas des agents réactifs, il n'y a pas d'organisation particulière. Le comportement du système émerge de celui des agents et d'une organisation particulière [Steels 90]. Dans notre cas, les agents sont du type cognitif et communiquent donc selon un protocole de haut niveau. Ceci nécessite la mise en place d'une organisation spécifique afin d'optimiser la coopération entre agents. Dans notre architecture tous les agents sont au même niveau, car chaque agent possède des connaissances particulières dont le groupe a besoin pour résoudre la tâche coopérative.

5.2.2.2 Contrôle et prise de décision

Dans les architectures multi-agents, le contrôle et la prise de décision peuvent être localisés dans les différents agents ou alors être localisés au niveau d'un agent dit superviseur. Ce pouvoir permet de gérer l'allocation des ressources, le choix des actions et la résolution de conflits éventuels. Dans EDER_TC, les deux modes sont fonctionnels. Lorsqu'on est en mode de fonctionnement décentralisé, le contrôle et la prise de décision sont distribués entre les agents participant à la réalisation de la tâche coopérative. En mode centralisé, ce pouvoir est aux mains de l'agent gestionnaire de tâche coopérative.

5.2.2.3 Coopération entre agents

La coopération entre agents désigne la façon dont les agents interagissent entre eux. Ce choix permet de définir par la suite les informations que doit posséder chaque agent participant au processus de réalisation d'une tâche coopérative. Dans notre architecture, les agents tirent les connaissances relatives à leurs croyances, leurs buts et leurs intentions du modèle de tâche Monaco-T utilisé dans l'architecture. Pour participer aux activités de coopération telles que, la mise à jour de l'objet manipulé par la tâche coopérative, la prise en compte des informations provenant des autres agents, EDER_TC

utilise l'agent gestionnaire de tâche coopérative. Cet agent a donc à lui seul la responsabilité de mettre à jour les objets manipulés, d'intégrer les informations provenant des différents agents et de les distribuer. Il permet également d'assurer les demandes d'aide.

5.2.2.4 Résolution de conflit

Pour mener à bien une résolution de tâche coopérative, les agents doivent éviter autant que possible les situations conflictuelles. Ils doivent donc coordonner leurs actions. Deux formes de coordination sont possibles : planifier de façon globale les actions de tous les agents ou donner une autonomie totale aux agents qui mettront en place des mécanismes de résolution de conflit chaque fois qu'il y en aura un. Dans notre système, une planification globale est construite à travers le modèle Monaco-T. Elle est ensuite distribuée aux agents. En résultat, on a donc un environnement où les agents sont autonomes, mais leurs actions résultent d'une planification globale. Les conflits sont donc automatiquement exclus de notre système.

5.2.2.5 Communication

Dans un système multi-agents, la communication est la base de tout travail coopératif. Elle permet de coordonner les actions et de résoudre les conflits. Pour communiquer, les agents doivent disposer d'un langage, d'un protocole et d'un ou plusieurs médias supportant les échanges. Les primitives du langage de communication découlent de la tâche Monaco-T exploitée par EDER-TC. Le protocole qui représente les règles nécessaires à la communication est géré par les règles des couches dynamique et scénario de Monaco-T. Dans la mesure où notre système se veut assez ouvert, le média utilisé est Internet. Nous détaillerons lors de la description des différents agents comment ces outils de communications sont exploités.

5.2.3 Comportement d'EDER_TC

L'architecture EDER_TC a deux modes de fonctionnement :

- centralisé et

- décentralisé.

Dans le mode centralisé, les agents gestionnaires de postes de travail jouent le rôle d'interface distribuée fonctionnant à distance. Ils agissent ainsi pour l'agent gestionnaire de la tâche coopérative qui est chargé de toutes les actions dites intelligentes.

Dans le mode décentralisé, chaque agent gestionnaire d'un poste de travail joue un plus grand rôle. Il gère une vue de MONACO_T contenant au moins les informations liées au poste de travail qu'il représente. Au moment de la description de chaque type d'agent, nous verrons quelles sont les informations pertinentes de MONACO_T qu'il faut transférer au niveau de ces agents.

Le fonctionnement d'EDER_TC se fait en deux phases,

- initialisation et
- exploitation.

La phase d'initialisation permet de mettre en place tous les agents devant participer à la simulation ou à l'enseignement d'une tâche coopérative. En fonction du mode de fonctionnement, les informations adéquates sont transférées de l'agent gestionnaire de la tâche coopérative aux agents gestionnaires de postes.

Dans la phase d'exploitation, chaque agent réagit en fonction des activités du poste de travail qu'il contrôle, de ses niveaux d'intelligence (réactif, coopératif, social) et de son type (automatique, humain, enseignant).

Le comportement d'EDER_TC est défini lors de la mise en place des agents devant participer à l'exploitation de la tâche coopérative et mise à jour pendant le processus d'exploitation.

5.2.3.1 Processus de mise en place des agents dans EDER_TC

Dans la phase de mise en place des agents, l'agent gestionnaire négocie avec l'initiateur (un humain ou un agent automatique) les points suivants :

- Le mode de fonctionnement (centralisé ou décentralisé) et

- la nature des agents gérant chaque poste de travail (automatique, humain ou enseignant).

Chaque fois qu'un nouvel agent se connecte pour prendre en charge un des postes de travail, le gestionnaire lui envoie les informations lui permettant de jouer son rôle en tenant compte du mode de fonctionnement choisi et de la nature de l'agent.

Lorsque tous les postes de travail sont occupés le processus d'exploitation peut enfin démarrer.

5.2.3.2 Processus d'exploitation d'un objet MONACO_T dans EDER_TC

Au niveau du processus d'exploitation, chaque agent participant a un comportement qui lui est propre et qui dépend des paramètres suivant :

- l'occupation du poste de travail (défini par chaque tâche),
- le mode de fonctionnement d'EDER_TC (centralisé, décentralisé), et
- la nature de l'agent (automatique, humain, enseignant).

Le comportement d'EDER_TC pendant le processus d'exploitation émerge de celui de chaque agent participant à l'exploitation de la tâche coopérative. L'analyse des différents agents génériques va donc nous permettre de comprendre ce comportement global.

5.3 Description des agents génériques utilisés

Les agents utilisés sont qualifiés de génériques car ils sont capables de participer à la réalisation de toute tâche coopérative en autant que cette dernière soit modélisée suivant Monaco-T. Dans la littérature il existe plusieurs catégories d'agents.

- Les agents réactifs qui réagissent à des stimuli pour modifier leur environnement.
- Les agents cognitifs qui sont capables de raisonner et d'établir des plans d'action. Ils sont dotés d'une intentionnalité (ils sont capables de déclarer explicitement leurs buts et leurs moyens d'y parvenir), d'une rationalité (ils sont conscients des effets des actions qu'ils accomplissent et sont capables de les expliquer) et d'un système de planification (ils planifient leurs actions en fonction du but à atteindre).

- Les agents sociaux, qui sont capables, dans le processus de planification de leurs activités, de prendre en compte l'état cognitif et mental des autres agents participant à la tâche coopérative. Ils peuvent ainsi anticiper les actions des autres et avoir un comportement plus coopératif.

Dans notre système, nous allons nous contenter d'implanter des agents cognitifs. Ces derniers nous permettant d'expérimenter valablement tout le processus d'enseignement de tâches coopératives. EDER_TC utilise plusieurs types d'agents. Chaque type a un comportement particulier qui entraîne une différence au point de vue architectural. Nous allons définir chaque type en détaillant le rôle qu'il a dans l'architecture, les différents modules qu'il intègre et le contenu des bases de connaissances qui lui servent de mémoire de travail et de mémoire à long terme. Les types d'agents que nous allons décrire sont les suivants : agent automatique pour la simulation, agent humain pour la simulation et agent automatique pour l'enseignement.

5.3.1 Agent automatique pour la simulation

Les agents automatiques pour la simulation sont utilisés en mode réalisation de tâche coopérative. Par exemple si la tâche à réaliser est celle de piloter un avion avec les postes de travail pilote et copilote, faire jouer les deux rôles par des agents automatiques revient à enclencher le pilotage automatique. Dans ce cas les décisions sont prises par les agents en fonction des paramètres du vol et les actions adéquates sont déclenchées.

5.3.1.1 Schéma architectural

L'architecture d'un agent automatique de simulation (figure 23) est structurée autour de trois modules et cinq bases de connaissances. Chaque module implante un processus particulier de l'agent. Les modules disponibles dans l'agent automatique de simulation sont : le module d'écoute, le module d'envoi et le module de raisonnement. Les bases de connaissances sont : les banques de messages reçus et de messages à envoyer, la base contenant les scénarios du poste de travail et la base de connaissance contenant une vue partielle des couches statique et dynamique de la tâche à réaliser.

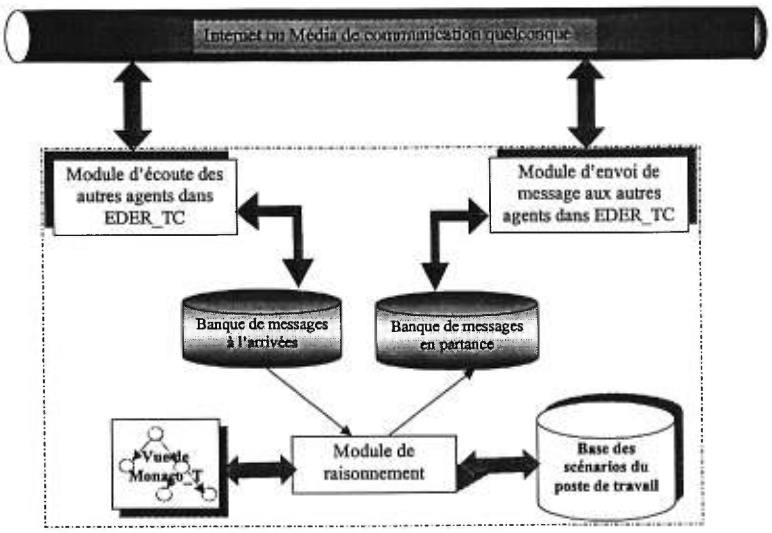


Figure 23: Architecture de l'agent automatique de simulation

5.3.1.2 Module d'écoute des autres agents

Ce module est en permanence en train d'écouter le réseau sur lequel opère l'architecture EDER_TC. Dès qu'un message arrive et concerne l'agent automatique de simulation, le message est récupéré et stocké dans la banque des messages à l'arrivée.

5.3.1.3 Module d'envoi de messages aux autres agents

Le module d'envoi de messages aux autres agents a pour but de récupérer les messages disponibles dans la banque de messages à envoyer et de les transmettre aux destinataires. Ce module est également construit sous forme d'un processus indépendant qui n'est actif que lorsque la banque de messages à transmettre lui communique un nouveau message.

5.3.1.4 Module de raisonnement

Ce module a deux phases de fonctionnement : la phase d'initialisation et la phase d'exploitation.

Lorsque EDER_TC fonctionne en mode centralisé, la première phase consiste uniquement à initialiser les adresses de correspondance. En phase d'exploitation, le module de raisonnement reçoit du gestionnaire de tâche coopérative les actions à faire après filtrage avec les règles de la dynamique et de la couche scénario. Le module de

raisonnement n'a plus qu'à choisir aléatoirement l'action à exécuter. En général le filtrage effectué par le gestionnaire de tâche coopérative ne donne qu'une seule action.

Lorsque EDER_TC fonctionne en mode décentralisé, l'agent gestionnaire du poste de travail est plus actif. Dans la suite nous allons détailler le comportement de l'agent uniquement lors d'un fonctionnement selon ce mode.

Phase d'initialisation.

La phase d'initialisation consiste à recevoir du gestionnaire de tâche coopérative toutes les informations de MONACO_T permettant de gérer les activités du poste de travail. La liste des informations à recevoir est la suivante :

- l'ensemble des variables caractéristiques,
- les règles de la dynamique pour les sous-tâches du poste de travail,
- l'ensemble des scénarios du poste de travail,
- l'état de départ de la tâche coopérative et
- le message de démarrage de la session de travail.

Phase d'exploitation

Dans cette phase, le module de raisonnement active une série d'actions (figure 24) qui est activée lorsque la banque de messages à l'arrivée reçoit un nouveau message. Ainsi après chaque action, l'agent automatique doit attendre que l'agent gestionnaire lui renvoie les modifications de son action avant de pouvoir chercher à exécuter une nouvelle action. Cette contrainte fait en sorte qu'il est impossible de planifier d'avance ses interventions. Les agents automatiques dans EDER_TC sont considérés comme des agents réactifs.

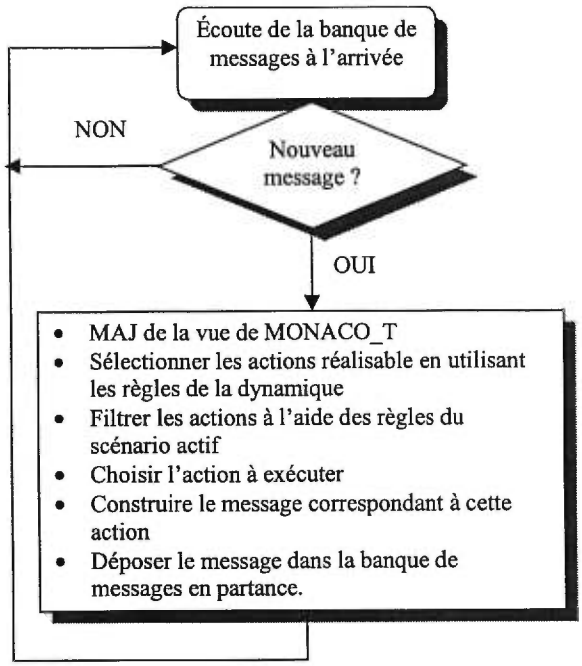


Figure 24 : Algorithme de raisonnement de l'agent automatique de simulation

5.3.1.5 Banque de messages à l'arrivée

Cette banque est construite sous forme d'une base de données active. Lorsqu'arrive un nouveau message de la banque elle avertit le module de raisonnement .

5.3.1.6 Banque de messages en partance

La banque de message en partance est identique à celle des messages arrivés, à la seule différence que le module à prévenir en cas d'arrivée d'un nouvel événement est le module d'envoi de messages aux autres agents d'EDER_TC.

5.3.1.7 Base de scénarios du poste de travail

Dans cette base sont transférés les éléments de la couche scénario de la tâche relatifs au poste de travail pris en charge par un agent automatique de simulation. Ainsi pour chaque scénario du poste de travail occupé cette base contient : les règles d'ordonnancement, les règles coupure de sous tâches et les règles coupure de termes.

5.3.1.8 Vue partielle de l'objet MONACO_T.

La vue partielle envoyée à chaque poste de travail doit permettre d'évaluer les règles régissant la dynamique et la couche scénario pour ce poste. Il s'agit donc de transférer au niveau de l'agent :

- toutes les sous-tâches disponible dans MONACO_T,
- pour chacune des sous-tâches, ses variables caractéristiques,
- les règles régissant la dynamique des sous-tâches appartenant au poste de travail de l'agent,
- les actions et les paramètres associés aux sous-tâches appartenant au poste de travail de l'agent, ce qui représente l'interface du système à manipuler.
- les variables caractéristiques des systèmes à manipuler lorsqu'il s'agit d'une tâche de manipulation de système.

5.3.2 Agent humain pour la simulation

Il représente un humain qui à travers son ordinateur participe à la réalisation d'une tâche coopérative avec d'autres agents. Dans la mesure où le principal agent de notre architecture (l'agent gestionnaire de tâche coopérative) est un agent automatique, l'agent humain doit être capable de communiquer dans un langage compréhensible par ce dernier.

5.3.2.1 Schéma architectural

Pour être capable de dialoguer avec les agents automatiques dans EDER_TC, tout agent humain doit être capable de comprendre et de s'exprimer dans leur langage. Pour cette raison, le schéma architectural de l'agent humain (figure 25) est similaire à celui de l'agent automatique, à la seule différence que le module de raisonnement est représenté par l'humain. De plus cette architecture doit traduire le message des autres agents en des termes compréhensibles par l'humain et doit fournir à ce dernier les instruments lui permettant de communiquer avec les autres agents.

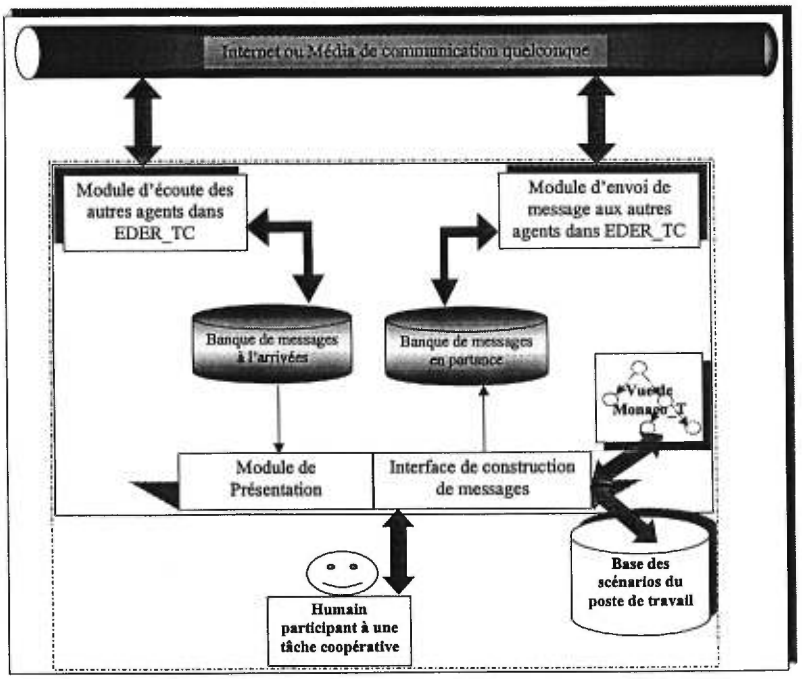


Figure 25 : Architecture agent humain pour la simulation

L'architecture ainsi définie comporte quatre bases de connaissances (banque de message à l'arrivée, banque de messages en partance, vue de l'objet MONACO_T et de la base des scénarios du poste de travail) et quatre modules (module d'écoute des autres agents, module d'envoi de messages aux autres agents, module de présentation et l'interface de construction de messages). Par rapport à l'architecture de l'agent automatique de simulation, nous avons deux ajouts, le module de présentation et l'interface de construction des messages. Le comportement et le contenu des autres entités de l'architecture étant identiques à ceux rencontrés dans l'architecture des agents automatiques, nous ne décrirons que le comportement de ces deux nouveaux modules.

5.3.2.2 Module de présentation

Le rôle de ce module est de récupérer tous les messages destinés à l'agent humain et de les convertir en un format que ce dernier est capable de comprendre. Lorsque le message consiste à mettre à jour la vue de MONACO_T, le module de présentation le fait. Ce module s'apparente à un navigateur sur le WEB. En fonction du type de message il active une fonction de présentation. Lorsqu'un type de message lui est inconnu, il faut le mettre à jour en lui intégrant le dispositif capable d'en réaliser la présentation. Par exemple si le

message envoyé est un fichier Quicktime, il faut être capable de déclencher le logiciel Quicktime.

5.3.2.3 Interface de construction de messages

L'interface de construction de messages permet à l'humain participant à la réalisation d'une tâche coopérative d'exécuter les actions relatives à cette tâche. Elle permet à l'humain de sélectionner ou définir le destinataire, l'action à transmettre, ses paramètres éventuels et le type du message qui lui est associé. Elle place le message ainsi conçu dans la banque de messages en partance. Si l'humain veut consulter l'état de la tâche coopérative avant d'agir, le module exploite la vue de MONACO_T qu'il gère et la base des scénarios du poste de travail pour le satisfaire. Pour permettre la conception d'un message valide, la liste des actions présentées à l'humain est filtrée. Elle ne contient que les actions exécutables en fonction des règles de la dynamique. Si l'humain le désire, l'interface de construction de message peut même utiliser un scénario pour raffiner ce filtrage. En plus, le module interface utilise la variable « Statuts » disponible dans MONACO_T pour présenter les actions à exécuter suivant leur proximité du contexte d'exécution.

5.3.3 Agent automatique pour l'enseignement d'un poste de travail

Cet agent est le plus complexe des agents évoluant dans l'architecture EDER_TC. En fonctionnement centralisé il joue le rôle de relais pour le gestionnaire de tâche coopérative qui réalise toutes les activités.

En mode décentralisé, lors de l'initialisation, il reçoit exactement les mêmes informations que l'agent automatique de simulation. Lors de l'exploitation, il est capable de jouer trois rôles différents :

- simuler automatiquement les activités de son poste de travail,
- permettre à un humain de réaliser ces activités et
- enseigner à un humain comment les réaliser.

Cet agent permet d'implanter un STI (Système Tutoriel Intelligent) qui s'inscrit dans la grande famille des STI procéduraux. Il permet d'enseigner les habiletés et les procédures nécessaires pour maîtriser les activités d'un poste de travail dans une tâche coopérative. Ce tutorat se fait principalement sous la forme d'un coaching. Plusieurs stratégies d'intervention sont possibles :

- *« Model tracing »* [Anderson 83b, Anderson 88, VanLehn 88,] : dans cette stratégie, le système suit l'apprenant dans le processus de réalisation d'une tâche et intervient lorsque ce dernier commet une erreur. Notre agent se base principalement sur les règles de la dynamique et celle de la couche scénario pour réaliser cette performance.
- *« Issue-based »* [Burton 82, Wenger 87, Halff 88] : dans cette stratégie, le système intervient lorsqu'il se rend compte que l'apprenant n'a pas exploité une opportunité accessible. L'agent automatique d'enseignement se base principalement sur les règles définissant la couche scénario pour implanter cette stratégie.
- *Conseiller* [Lajoie & Lesgolg 89, Brown 75] : dans cette stratégie, lorsque l'apprenant a des difficultés, il demande à l'agent de l'aider. Dans ce mode, l'agent attend d'être solliciter avant de répondre.
- *Démonstrateur* [Allessi 85; Kerr 94; Elsom-Cook & al. 88] : dans ce cas l'agent automatique est capable de simuler entièrement ou partiellement les activités du poste de travail dans la tâche coopérative. L'apprenant suit le travail du coach et essaye de l'imiter.

Dans la mesure ou cet agent s'occupe d'enseigner les connaissances relatives à un poste de travail d'une tâche coopérative, il permet d'implanter un STI distribué, dédié à l'enseignement de tâches coopératives.

5.3.3.1 Schéma architectural

L'architecture de l'agent automatique pour l'enseignement d'un poste de travail (figure 26) fait la fusion des entités appartenant à l'agent automatique de simulation et à l'agent humain de simulation. Les entités implantées dans cette architecture sont les suivantes : module d'écoute des autres agents d'EDER_TC, module d'envoi de messages aux autres

agents, la banque de messages à l'arrivée, la banque de messages en partance, le module de présentation, l'interface de construction de messages, le module de raisonnement, une vue de l'objet MONACO_T et une base de scénarios du poste de travail.

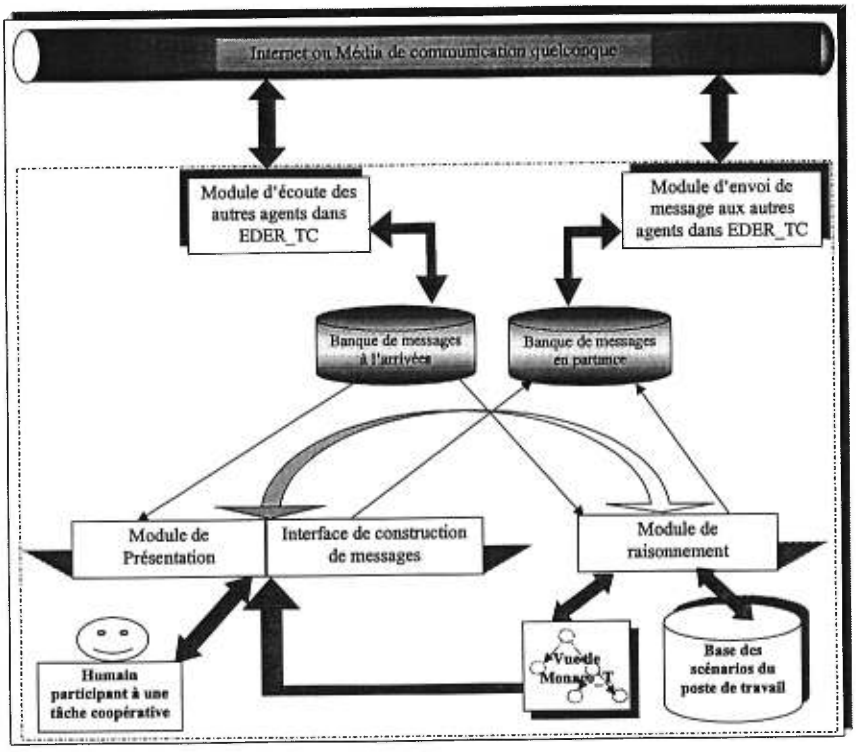


Figure 26 : Architecture agent d'enseignement

Les entités servant à la communication telles que le module d'écoute des autres agents d'EDER_TC, le module d'envoi de messages aux autres agents, la banque de messages à l'arrivée, la banque de messages en partance, la vue de l'objet MONACO_T et la base de scénarios du poste de travail ont le même comportement que dans les deux types d'agents précédents. Définir le comportement de l'agent automatique d'enseignement, c'est se focaliser sur le module de présentation, l'interface de construction des messages et le module de raisonnement.

5.3.3.2 Module de présentation

Pour cet agent, le module présentation doit être capable de présenter à l'humain aussi bien les informations venant des autres agents de l'architecture que les informations provenant du module de raisonnement.

5.3.3.3 Interface de construction de messages

En plus de la construction de messages à envoyer aux autres agents d'EDER_TC, l'interface de construction de messages doit également permettre de poser des questions ou de répondre aux questions posées par le module de raisonnement.

5.3.3.4 Module de raisonnement

Chez cet agent, le module de raisonnement fonctionne comme un coach dans un STI. Il est capable de participer à la réalisation d'une tâche coopérative en simulant les activités liées à son poste de travail. Dans ce cas il se comporte comme un agent automatique de simulation. Dans les STI ce comportement permet de mettre en œuvre et de superviser une démonstration. Cette démonstration peut être interactive, ce qui implique que, l'humain est autorisé à intervenir quand il le désire. Pour réaliser cette performance, le module de raisonnement exploite les règles de la dynamique dans MONACO_T, ensuite filtre les sous tâches et les trie en fonction des règles de la couche scénario, récupère l'action à exécuter dans la sous tâche sélectionnée et enfin construit le message à envoyer à l'agent gestionnaire de la tâche coopérative.

Le module de raisonnement peut également fonctionner en mode critique. Dans ce cas, il laisse l'humain participer à la réalisation de la tâche coopérative tout en contrôlant ses actions. Lorsque l'humain commet une erreur, le module de raisonnement intervient en lui expliquant pourquoi l'action effectuée n'est pas bonne. Ce contrôle se base principalement sur les règles de la dynamique et celle de la couche scénario.

Le module de raisonnement peut enfin fonctionner comme un conseiller. Dans ce cas il répond à toutes questions que pourrait lui poser l'humain chargé de réaliser les activités du poste de travail. Dans la mesure où les questions posées peuvent demander la prise en compte des informations appartenant à d'autres postes de travail, ce mode de fonctionnement peut impliquer une coopération entre les différents agents participant à la réalisation d'une tâche coopérative. Trois grandes familles d'aide existent dans ce mode. Il s'agit d'une aide aux niveaux réactif, coopératif et social. Ces familles correspondent aux fonctions d'aide qui ont été implantées au niveau du modèle MONACO_T.

5.3.4 Agent gestionnaire de la tâche coopérative

Cet agent est central dans notre architecture. Non seulement il joue le rôle d'orienteur de communication entre les agents d'EDER_TC, mais encore, il gère l'objet MONACO_T.

Rôle d'orienteur : ce rôle lui est attribué dans la mesure où il est obligé de dialoguer avec tous les autres agents de notre architecture pour leur transmettre les informations provenant de l'objet MONACO_T en cours de traitement.

Gestion de l'objet MONACO_T : comme nous l'avons vu au chapitre précédent, MONACO_T permet de modéliser les actions et les contraintes de manipulation d'une tâche coopérative. De plus ce modèle est aussi bien adapté pour l'enseignement de tâche coopérative que pour la simulation. Gérer donc l'objet MONACO_T revient à fournir aux différents agents participant à la tâche coopérative les actions à exécuter, les informations expliquant pourquoi et comment la tâche doit être réalisée et finalement à activer les composants MONACO_T (systèmes que la tâche doit manipuler).

5.3.4.1 Schéma architectural

L'architecture du gestionnaire de la tâche coopérative (figure 27) fait ressortir les mêmes organes de communication que ceux des agents précédents. Il comprend en plus un module de raisonnement, la structure complète de la tâche en MONACO_T et l'ensemble des systèmes (composants MONACO_T) que la tâche doit manipuler.

La structure des organes de communication ne change pas pour cet agent sauf le module d'envoi de messages qui doit intégrer des protocoles de diffusion. Pour décrire le comportement particulier de cet agent nous devons nous focaliser sur le module de raisonnement et le module d'envoi de message aux autres agents. Par ailleurs, le comportement de MONACO_T et de ses composants a déjà été défini dans le chapitre précédent.

5.3.4.2 Module d'envoi de messages aux autres agents d'EDER_TC

Le module d'envoi de messages aux autres agents, en plus d'expédier les messages construit par l'agent, permet également d'implanter des algorithmes de diffusion de

messages. Lorsque ce module reçoit un message à transmettre, l'adresse du destinataire peut être d'une des quatre formes suivantes :

- Adresse d'un destinataire unique. Dans ce cas ce module retransmet le message au destinataire sans traitement particulier.
- Adresse d'un sous-groupe prédéfini. Le module conserve les adresses physiques de chaque membre du sous-groupe. En exploitant cette liste, il construit un message pour chaque membre du sous-groupe et le lui envoie.
- Adresse de l'ensemble des agents. Dans ce cas, un message est construit pour chaque membre participant à EDER_TC à l'exception de l'émetteur du message.
- Adresse d'une liste exhaustive de destinataires. Dans ce cas le module d'envoi de messages construit les messages à envoyer en utilisant la liste d'adresses contenue dans le message initial.

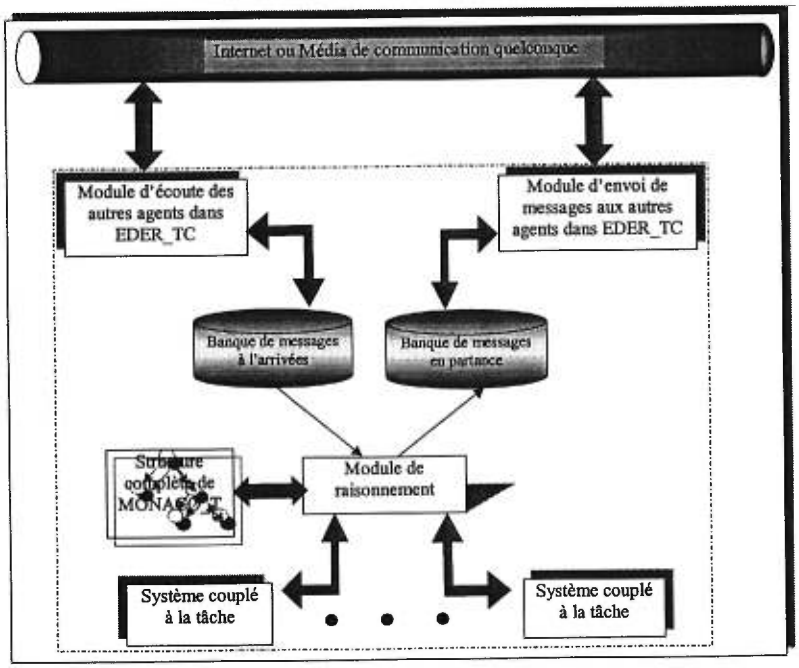


Figure 27 : Architecture agent gestionnaire de la tâche coopérative

5.3.4.3 Description du module de raisonnement du gestionnaire de tâche coopérative

Ce module implante les différents rôles que doit jouer l'agent gestionnaire de tâche coopérative. Pour faire le routage, la simulation et l'enseignement, le module est construit

en couches (figure 28). Chaque couche permet d'implanter une fonction particulière. Les couches recensées sont les suivantes : routage, exécution de sous-tâche, exécution action d'un composant externe, module d'explication et génération des activités pour postes de travail.

Ce module de raisonnement définit aussi bien le comportement en mode centralisé qu'en mode décentralisé.

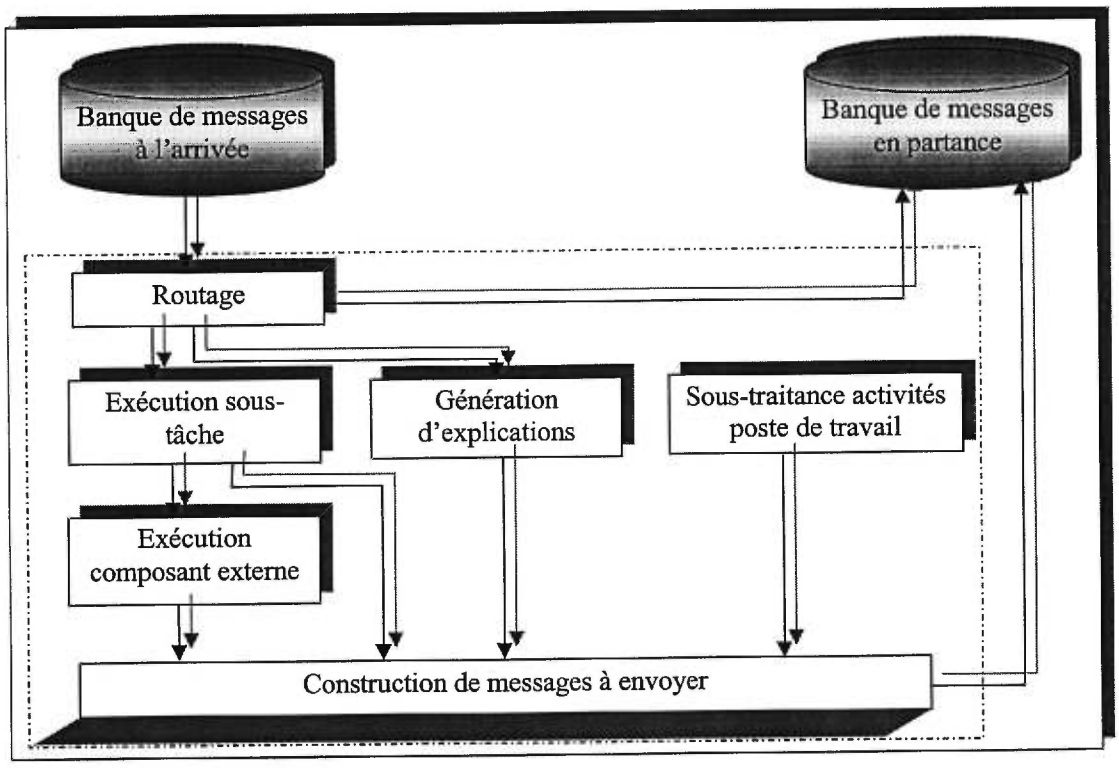


Figure 28 : Module de raisonnement du gestionnaire de tâche coopérative

Routage

L'activité de routage permet au module de raisonnement de l'agent gestionnaire de la tâche coopérative de jouer le rôle de nœud central dans le réseau en étoile formé par les agents actifs dans EDER_TC. Pour cela lorsque le module de routage détecte que le destinataire final du message n'est pas le gestionnaire de tâche coopérative, il dépose le dit message dans la banque des messages à envoyer. Il est alors soit destiné à un agent particulier, soit à un sous groupe des agents, soit à tous les agents d'EDER_TC.

Ce module a également pour but d'acheminer le message au bon sous module du module de raisonnement dans le cas où le message est destiné au gestionnaire de tâche

coopérative. Les messages à router aux modules sont de deux types : messages d'activation d'une sous-tâche et les messages de demande d'explications. Le message d'activation est envoyé au module d'exécution de sous-tâches, alors que le message de demande d'explication est envoyé au module de génération d'explications.

Exécution de sous-tâches

Le module d'exécution de sous-tâches peut être considéré comme le bras armé des autres agents évoluant dans l'architecture EDER_TC. Il permet d'activer les étapes d'exécution de sous-tâches en utilisant les fonctions adéquates fournies par le modèle MONACO_T. Lorsque l'architecture EDER_TC fonctionne en mode centralisé, le module d'exécution de sous tâches effectue les opérations suivantes :

- vérifier que les conditions d'activation de la sous-tâche sont réalisées,
- exécuter l'étape de la sous-tâche cible,
- récupérer de MONACO_T les activités de dépendances fonctionnelles (adf) qui permettent de mettre à jour les variables caractéristiques de MONACO_T.

Lorsque l'architecture fonctionne en mode décentralisé, toutes les étapes sont reprises sauf la vérification des conditions d'activation, car elle est faite au niveau des agents gestionnaires de postes de travail.

Ces adfs sont ensuite transférées aux autres agents afin qu'ils puissent mettre à jour la vue de MONACO_T qu'ils possèdent.

Exécution action de composant

Le module d'exécution des actions de composant permet de manipuler les systèmes qui sont couplés à la tâche coopérative. Il vient à la suite de l'exécution de la sous-tâche. Lorsqu'une sous-tâche exécutée pointe sur une action d'un composant MONACO_T, les paramètres de l'action sont récupérés dans le message d'activation envoyé d'un des postes de travail et l'action est exécutée. Le résultat est récupéré et envoyé à tous les agents participant à la réalisation de la tâche coopérative.

Explication

Le module explication dans le raisonnement du gestionnaire de tâche coopérative est utilisé uniquement dans le cas où EDER_TC fonctionne en mode centralisé. Il permet de répondre à un ensemble de questions relatives à la tâche coopérative telles que :

- Qu'est ce qui contrôle l'activation d'une action donnée ?
- Pourquoi une sous tâche ne peut-elle pas être déclenchée ?
- Pourquoi une sous tâche ne peut-elle pas être réalisée ?
- Pourquoi une sous tâche ne peut-elle pas être terminée ?
- Que faut-il faire pour rendre valide les conditions de réalisation d'une sous-tâche ?

Pour générer ces conseils, le module d'explication utilise les fonctions de génération de conseil développées dans MONACO_T. Ces fonctions permettent au module d'explication de donner des aides suivant trois niveaux : réactif, coopératif et social. Ces niveaux correspondent à ceux utilisés par les fonctions de MONACO_T.

Lorsque EDER_TC fonctionne en mode décentralisé, les explications sont générées par coopération entre les agents gestionnaires de postes de travail comme nous le verrons plus tard.

Génération activités pour postes de travail

Ce module est uniquement fonctionnel lorsque l'architecture EDER_TC est utilisée en mode centralisé. Il permet après chaque activité au niveau de la tâche coopérative de générer les actions disponibles (condition d'activation valide) pour chaque poste de travail dont l'agent gestionnaire est en mode simulation.

Comportement général du module de raisonnement

Le comportement général du module de raisonnement de l'agent gestionnaire de la tâche coopérative (figure 29) est fonction de celui des sous-modules qu'il contient.

Ce comportement général tient compte du fait que l'architecture fonctionne en mode centralisé ou non. Le mode de fonctionnement est donc un paramètre essentiel dans le comportement du gestionnaire de la tâche coopérative.

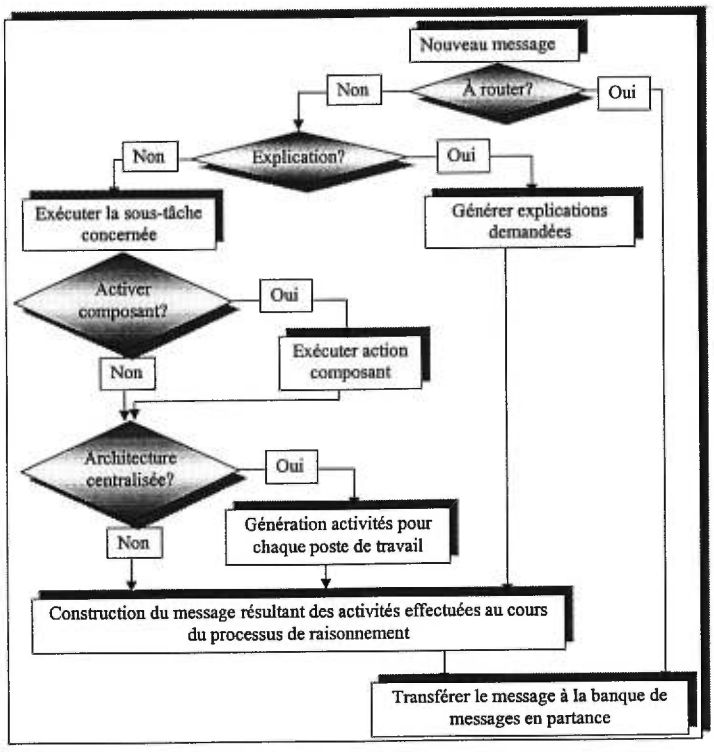


Figure 29 : Algorithme utilisé dans le raisonnement du gestionnaire de tâche coopérative

5.4 Génération d'explications en mode décentralisé

En mode décentralisé, la génération d'explications est plus complexe qu'en mode centralisé car ceci demande la coopération de l'ensemble des postes de travail. Une explication à un poste de travail faisant référence aux informations appartenant à d'autres postes de travail.

Pour expliquer cette génération coopérative des explications, nous allons partir d'un exemple de représentation de tâche coopérative à l'aide de MONACO_T, et nous montrerons comment les agents gestionnaires des postes de travail coopèrent pour répondre à une question qui leur est posée.

5.4.1 Exemple de tâche coopérative

Dans un hôpital, un médecin est chargé de faire le diagnostic médical d'un patient. Dès que le médecin déclenche le processus de diagnostic, il demande à l'infirmière de

prélever et de transmettre au laborantin le sang du patient. Le laborantin analyse les échantillons de sang et transmet le résultat au médecin. Ce n'est qu'après avoir examiné les résultats du laboratoire que le médecin donne son diagnostic.

5.4.1.1 Statique de la tâche de diagnostique

Si nous modélisons cette tâche suivant l'approche MONACO_T, nous faisons ressortir trois sous-tâches (diagnostique, prise de sang, analyse de sang) et trois postes de travail (médecin, infirmière, laborantin). Ces sous-tâches forment une hiérarchie (figure 30).

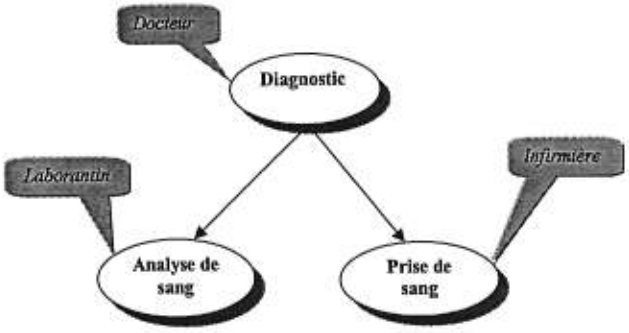


Figure 30: Exemple de tâche coopérative

Les trois postes de travail sont les suivants : le médecin, l'infirmière et le laborantin. Au niveau de la statique l'une des informations les plus importantes dans le cadre de l'explication est la méthode de réalisation de chaque sous-tâche. Dans ce cas, nous allons considérer que cette méthode de réalisation est un texte décrivant comment l'action de la sous-tâche est réalisée.

5.4.1.2 Dynamique de la tâche de diagnostique

Les règles simples permettant de définir la dynamique de cette tâche sont distribuées dans les trois sous-tâches recensées.

La dynamique de la sous-tâche Diagnostic (figure 31) permet d'en contrôler le comportement au cours du processus d'enseignement ou de simulation de la tâche coopérative.

La dynamique de la sous-tâche Prise de sang (figure 32) permet de contrôler le comportement de cette sous-tâche.

La dynamique de la sous-tâche Analyse de sang (figure 33) permet également, comme décrit dans le modèle MONACO_T, de contrôler le comportement de cette sous-tâche.

Condition de déclenchement:
État diagnostic = `initial`.

Condition de réalisation:
(État diagnostic = `déclenché`) ET
(État analyse de sang = `terminé`).

Condition de terminaison:
État diagnostic = `réalisé`.

Figure 31 : Dynamique de la sous-tâche Diagnostique

Condition de déclenchement:
(État Prise de sang = `initial`) ET
(État diagnostic = `déclenché`).

Condition de réalisation:
État prise de sang = `déclenché`.

Condition de terminaison:
État prise de sang = `réalisé`.

Figure 32: Dynamique de la sous-tâche prise de sang

Condition de déclenchement:
(État Analyse de sang = `initial`) ET
(État Prise de sang = `terminé`).

Condition de réalisation:
État Analyse de sang = `déclenché`.

Condition de terminaison:
État Analyse de sang = `réalisé`.

Figure 33: Dynamique de la sous-tâche Analyse de sang

5.4.2 Génération des explications

La génération des explications par les agents d'enseignement peut se faire suivant trois niveaux de raisonnement, le niveau réactif, le niveau coopératif et le niveau social.

5.4.2.1 Niveau réactif

Il permet à l'agent conseiller de réagir de manière instantanée aux demandes de l'apprenant. Cette réaction s'appuie principalement sur les règles de déclenchement, les règles de composition, les règles de terminaison et la base de méthodes de résolution. Ce niveau d'intervention définit l'aspect individuel de l'agent conseiller car il se base uniquement sur les informations disponibles dans sa mémoire à long terme pour répondre aux questions de l'apprenant dont il a la charge.

Niveau réactif face à « *Comment réaliser une action?* »: l'agent retourne simplement les conditions qu'il faut vérifier pour que l'action puisse être réalisée.

Niveau réactif face à « *Pourquoi une action ne peut-elle pas être réalisée?* »: l'agent retourne les termes qui rendent les conditions de réalisation fausse.

Niveau réactif face à « *Pourquoi doit-on réaliser une action? »: l'agent* retourne une phrase type du genre 'la résolution de la tâche coopérative à besoin que cette action soit réalisée'

Exemple: *En prenant comme exemple la tâche de la figure 34, on suppose que le poste ou est affecté l'agent conseiller est le poste du médecin. En supposant que la tâche soit dans l'état initial (aucune action n'a encore été effectuée), si un apprenant demande:*

Que faut-il pour que je puisse synthétiser (réaliser) l'action de diagnostic ?

Au niveau réactif, la réponse de l'agent conseiller est:

Tu dois déclencher l'action diagnostic, et le laborantin doit terminer l'analyse de sang.

La figure 34 présente graphiquement un exemple de génération de conseils avec un agent conseiller raisonnant au niveau réactif. Le niveau réactif est surtout adapté aux questions

de la forme « *comment faire une action?* » , car dans ces cas, l'agent conseiller doit tout simplement récupérer la méthode de réalisation pour la présenter à l'apprenant.

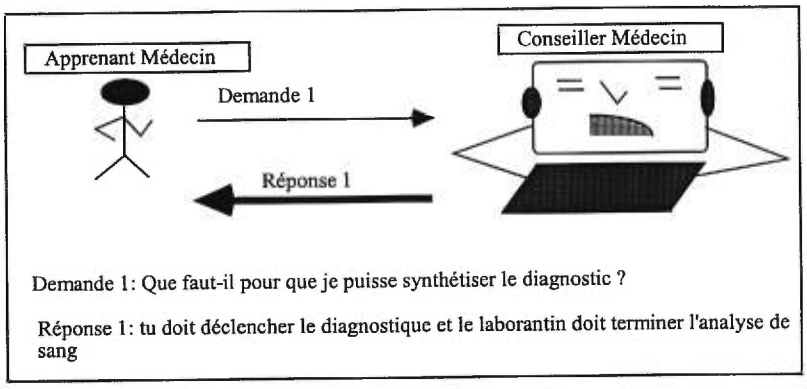


Figure 34 : Processus de conseil avec raisonnement au niveau réactif

5.4.2.2 Niveau coopératif

Il permet de trouver les réponses les plus complètes aux questions que pose l'apprenant dont il a la charge. Pour cela il sollicite la coopération de ses collègues conseillers qui l'aident à trouver les conseils qui ne sont pas de ses compétences. Ce niveau de raisonnement rend notre conseiller encore plus intelligent.

Niveau coopératif face à « *Comment réaliser une action*? »: la réponse est la même que celle du niveau réactif.

Niveau coopératif face à « *Pourquoi une action A ne peut-elle pas être réalisée*? »: l'agent cherche les termes qui rendent "faux" les conditions de réalisation de l'action A, ensuite il récupère l'adresse des agents concernés par ces termes, il envoie à chacun d'eux un message leur demandant pourquoi les termes qui sont à la charge de leur apprenant sont dans cet état. Ce processus va d'agent en agent jusqu'à obtenir qu'une action B doive être effectuée afin de débloquer tous les autres apprenants. La réponse retournée est soit la dernière réponse soit la succession des réponses produites durant les échanges de messages entre agents.

Niveau coopératif face à « *Pourquoi doit-on réaliser une action A?* »: l'agent envoie un message à tous les autres agents du système conseiller pour demander ceux qui dépendent de lui en fonction de l'action A. Ensuite il récupère les réponses pour les présenter à

l'apprenant comme étant les raisons pour lesquelles l'action A doit être réalisée (Figure 35).

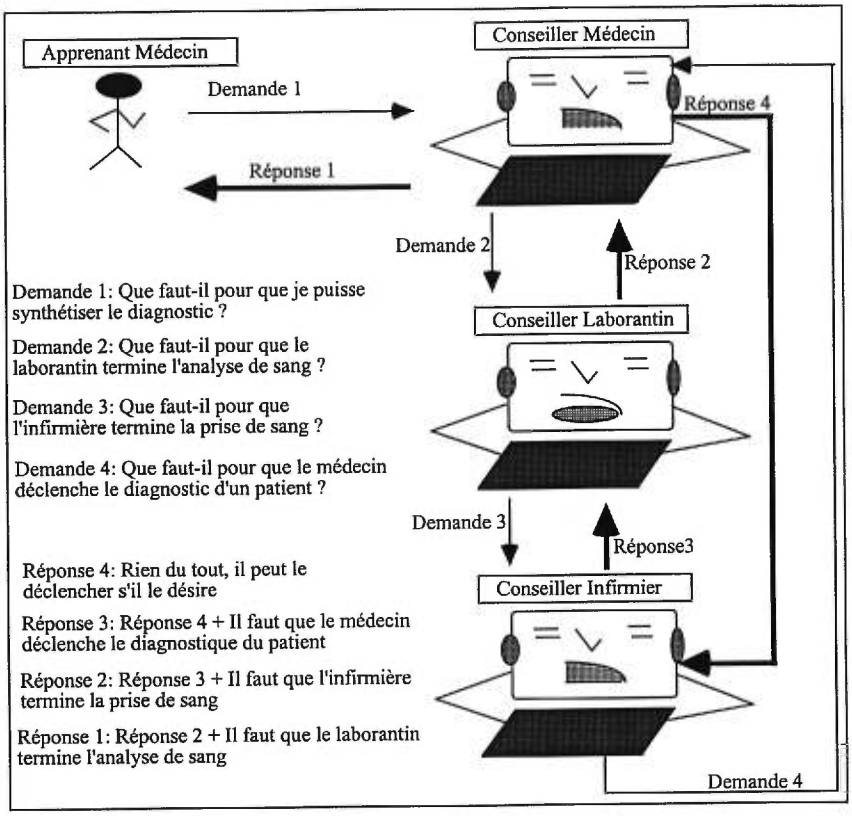


Figure 35 : Processus de conseil avec raisonnement au niveau coopératif

Exemple: *en reprenant l'exemple précédent, pour une question telle que:*

Que faut-il pour que je puisse synthétiser(réaliser) l'action de diagnostic?

l'agent conseiller fera les actions suivantes:

- *détecter que le médecin doit déclencher la tâche de diagnostic,*
- *détecter que le laborantin doit terminer l'analyse de sang,*
- *envoyer un message à l'agent conseiller chargé de suivre le poste de laborantin pour lui demander ce qu'il faut pour que ce dernier puisse terminer l'action d'analyse de sang.*

Si l'agent conseiller du laborantin juge (en fonction des bases définissant son comportement) qu'il a besoin du conseil d'un autre agent, il en fait la demande avant de retourner la réponse finale.

La réponse finale dans ce cas est : « tu dois déclencher le diagnostic, l'infirmière doit déclencher, synthétiser et terminer la prise de sang et le laborantin doit déclencher, synthétiser et terminer l'analyse de sang ». La réponse finale contient donc tous les préalables à travers la tâche coopérative qui donnent la possibilité de synthétiser l'action de diagnostic.

5.4.2.3 Niveau social

Ce niveau considère comme agent conseiller tous les agents informatiques et tous les apprenants participant à la résolution d'une tâche coopérative. Ainsi les apprenants humains sont sollicités dans la construction coopérative des conseils.

Cette caractéristique est innovatrice car elle permet au système conseiller d'interagir avec les apprenants sans être sollicité ni directement (par une demande de l'apprenant) ni indirectement (par une mauvaise action de l'apprenant). Ce niveau nous permet de mettre en œuvre le jeu de rôle qui permet de recevoir des conseils en jouant au conseiller ("Learning by advising"). Pour mieux comprendre ce principe, nous allons étudier le comportement du raisonnement social de l'apprenant à travers quelques questions types.

Niveau social face à « *Comment réaliser une action* ? »: la réaction du conseiller ici est la même que celle du niveau coopératif.

Niveau social face à « *Pourquoi une action ne peut-elle pas être réalisée* ? »: ici le processus est également le même que celui du niveau coopératif, sauf que lorsqu'un agent constate que l'apprenant dont il a la charge a les moyens de faire une action (qui débloquerait un autre agent) et qu'il ne le fait pas, alors l'agent lui envoie un message lui demandant pourquoi il n'exécute pas la dite action. Cette question posée à l'apprenant alors qu'il ne s'y attend pas l'entraîne dans un dialogue avec son agent conseiller dans le but de fixer les idées. Dans ce cas, l'apprenant initiateur du processus de construction du conseil et celui terminant le processus recevront un conseil du système conseiller. Le

niveau social donne au conseiller une dimension préventive car il est capable de donner des conseils à un apprenant sans demande implicite ou explicite de ce dernier.

Niveau social face à « *Pourquoi doit-on réaliser une action* ? »: la réaction est la même que celle produite par le niveau coopératif.

5.5 Implantation

Pour implanter le système que nous avons proposé, nous avons commencé par utiliser Smaltalk un langage orienté objet. Ce choix nous permettait d'étendre le concept d'objet vers le concept d'agent. Les agents de notre système devant être connectés à partir du réseau Internet, ce langage c'est avéré limité au niveau de la communication inter-machine. Avec la venue du langage Java qui était dès sa conception dédié au réseau Internet, nous avons recommencé nos développements sur cette plate-forme. Nous y avons défini la structure des messages de communication, la gestion des messages reçus, la gestion des message à envoyer, les primitives du langage de communication et le module de raisonnement qui permet de décider des actions à entreprendre au cours du processus de réalisation d'une tâche coopérative.

La communication dans notre système, au plus bas niveau, utilise les sockets qui sont un moyen de communication point à point utilisé sur les ordinateurs dotés du protocole de communication TCP/IP. L'utilisation d'un agent central permettant de recevoir et de redistribuer les messages nous permet d'avoir une structure où chaque agent est capable de communiquer avec qui il veut. De plus cette architecture en étoile nous permet de simuler les autres techniques de communication au besoin. Ainsi si on veut que les messages soient distribués à tous les agents, on peut simuler une communication par diffusion. Pour cela l'agent Gestionnaire de tâche coopérative reçoit le message d'un agent et l'envoie à tout les autres.

Les messages envoyés sont structurés sous forme d'objet. Chaque message contient en plus de l'information principale des éléments tels que l'émetteur, le récepteur, l'heure d'envoi et le type de message. Cette dernière information fait partie des primitives de base du langage entre agents. Elle permet de signaler au récepteur la sémantique du

message envoyé. L'envoi de l'objet message à travers le réseau se fait grâce à l'utilisation des fonctions de sérialisation intégrée dans le langage Java.

Les primitives de communication sont de deux types. Il y a celles qui permettent de décrire le type de message envoyé et celles qui permettent de décrire l'action à exécuter au cours de la réalisation d'une tâche coopérative. Les premières sont directement intégrées dans la structure des agents lors de leur construction et permettent de reconnaître les messages de connexion, de déconnexion, de suspension, de re-connexion, de commentaires et d'actes sur la tâche coopérative. Les secondes proviennent de la tâche coopérative modélisée à l'aide de Monaco-T. Elles sont distribuées aux agents au début de chaque session d'utilisation d'EDER-TC.

Le protocole de communication et la planification des activités est régie par les règles de la dynamique et celles de la couche scénario. Pour planifier ses activités, un agent utilise donc les règles qu'il a reçues en début de session. Le protocole étant déjà inclus dans la définition des règles, il n'y a pas d'effort supplémentaire à faire par les agents pour éviter les conflits.

5.6 Conclusion

Le modèle architectural que nous avons proposé est un modèle générique. Il est capable d'assurer la simulation et l'enseignement de toutes tâches coopératives modélisées en MONACO_T. Le caractère générique d'EDER_TC est supporté par l'implantation de quatre types d'agents : agent automatique pour la simulation des activités à un poste de travail, agent humain pour la simulation à un poste de travail, agent pour le tutorat d'un poste de travail et un agent gestionnaire de l'objet MONACO_T. EDER_TC est capable de gérer un nombre indéterminé de postes de travail, en autant que ceux-ci soient définis dans la tâche coopérative exploitée. L'agent pour le tutorat est un agent complexe, il est capable de simuler les activités du poste de travail. Cette possibilité lui permet de montrer à l'apprenant qu'il est chargé de suivre, comment les activités de ce poste peuvent être réalisées. Il peut également permettre à un intervenant humain de participer à la simulation d'une tâche coopérative. Pour cela il n'intervient pas dans le processus pour donner des conseils ou pour critiquer. Ces deux autres rôles s'apparentent à ceux de

l'agent automatique et de l'agent humain pour la simulation. En pratique, nous avons construit uniquement l'agent pour le tutorat d'un poste de travail. Cet agent fonctionne sous trois modes :enseigner, simuler et permettre la simulation. Chacun d'eux correspond au comportement d'un des agents recensés dans le cadre de l'occupation d'un poste de travail.

Avec cette structure, notre système est aussi bien adapté pour l'enseignement que pour l'apprentissage. Lorsqu'un apprenant ou un groupe d'apprenants veut se pratiquer à la réalisation d'une tâche coopérative sans la présence de module de tutorat, il peut le faire. Si le groupe veut suivre un cours de réalisation d'une tâche coopérative, le système est également capable de mettre à sa disposition un tuteur pour promouvoir l'enseignement.

Au niveau de la communication, l'infrastructure utilisée n'est pas très fiable. Il n'existe pas d'assurance au niveau de la qualité de service dans le réseau Internet. Lorsqu'on envoie un message, on n'est ni sure avec exactitude qu'il arrivera au destinataire ni du temps qu'il mettra à arriver. Malgré cette incertitude, le système que nous avons mis en place permet de simuler le principe d'une communication synchrone. Lorsqu'il y a des goulots d'étranglement au niveau d'un agent, le système est en mesure de lancer au niveau du serveur un agent automatique capable de remplacer celui dont les lignes de communications sont défaillantes. La tâche coopérative peut ainsi continuer son exécution sans interruption.

L'architecture que nous avons proposée permet également d'éliminer les problèmes de consistance d'objet. Dans la mesure où les objets partageables sont manipulés uniquement à travers l'agent gestionnaire de tâche coopérative il n'est donc pas nécessaire de dupliquer ces objets à travers le réseau. Les problèmes d'accès concurrents sont également résolus grâce au même agent qui peut se comporter à l'occasion comme un système de gestion de bases de données. Il peut ainsi bloquer la consultation d'une donnée jusqu'à ce qu'elle soit mise à jours par un autre agent.

Au niveau de la synchronisation des actions de réalisation d'une tâche coopérative, l'efficacité de notre système découle de la robustesse du modèle de représentation de tâche coopérative que nous avons utilisé.

6 Utilisation d'EDER_TC : cas de l'enseignement individualisé.

Ce chapitre nous permet d'expérimenter le modèle de tâche coopérative proposé et le système permettant de l'exploiter. Pour cela nous considèrerons une tâche coopérative que nous allons représenter suivant la description de Monaco-T. Le résultat obtenu sera ensuite utilisé dans le système de simulation et d'enseignement de tâche coopérative (EDER-TC).

L'exemple que nous avons choisi est un peu complexe, car il fait intervenir des meta-connaissances. Cependant, dans la mesure où nous possédons l'expertise au niveau de cette tâche, et que celle-ci nous permet de démontrer toute la puissance du couple Monaco-T, EDER-TC, cet exemple mérite d'être expérimenté.

La tâche coopérative que nous utiliserons est le processus d'enseignement lui-même. Il s'agit de l'enseignement individualisé qui fait intervenir deux postes de travail : l'enseignant et l'apprenant. Nous allons commencer par démontrer pourquoi le processus d'enseignement peut être considéré comme un processus coopératif. Nous allons ensuite choisir et expliquer un modèle d'organisation de l'enseignement qui sert à implanter la tâche d'enseignement de concept à l'aide d'EDER-TC. Cette implantation nécessite la modélisation du concept à enseigner et celle de la tâche d'enseignement de concept suivant le protocole établi par Monaco-T. Nous allons terminer ce paragraphe par les exploitations possibles de cette modélisation. Il s'agit principalement de l'utilisation du système en mode simulation pour enseigner un concept à un apprenant, de l'utilisation en mode enseignement pour enseigner comment il faut enseigner les concepts et de l'extension à apporter aux données modélisées pour permettre de faire de l'enseignement coopératif tel que le conçoit Slavin.

6.1 Enseignement vu comme tâche coopérative

Pour démontrer que l'enseignement est un processus coopératif, nous allons présenter ce que nous entendons par système coopératif, et point par point nous montrerons comment le processus d'enseignement respecte cette définition.

6.1.1 Définition de la coopération

Dans le chapitre sur l'étude de la coopération, nous avons défini un système informatique coopératif de la manière suivante :

Un système est dit coopératif s'il permet à deux ou plusieurs agents coopérants de dialoguer dans le but d'atteindre un objectif commun en utilisant l'ordinateur comme terminal de communication.

Partant de cette définition, nous allons montrer que l'enseignement classique est une tâche coopérative et que l'objet de l'enseignement peut être vu comme le système que la tâche permet de manipuler.

6.1.2 Définition de l'enseignement

Au niveau de l'éducation, l'enseignement est vu comme l'ensemble des actes de communication et de prises de décision mis en œuvre intentionnellement par une personne ou un groupe de personnes qui interagit en tant qu'agent en vue de susciter l'apprentissage. Dans le cadre de l'enseignement à l'aide d'un tutoriel, une situation pédagogique (figure 36) fait ressortir un agent (enseignant), un sujet (apprenant) et un objet (capacité). Dans cette situation, les échanges entre l'apprenant et l'enseignant se font à travers les connaissances à enseigner. Il existe une relation d'apprentissage entre l'apprenant et la capacité. Cette relation amène donc l'apprenant à manipuler la capacité. Il existe également une relation didactique entre l'enseignant et la capacité. Les manipulations faites par l'enseignant sur la capacité lui permettent d'enseigner l'apprenant pour qu'il maîtrise cette dernière.

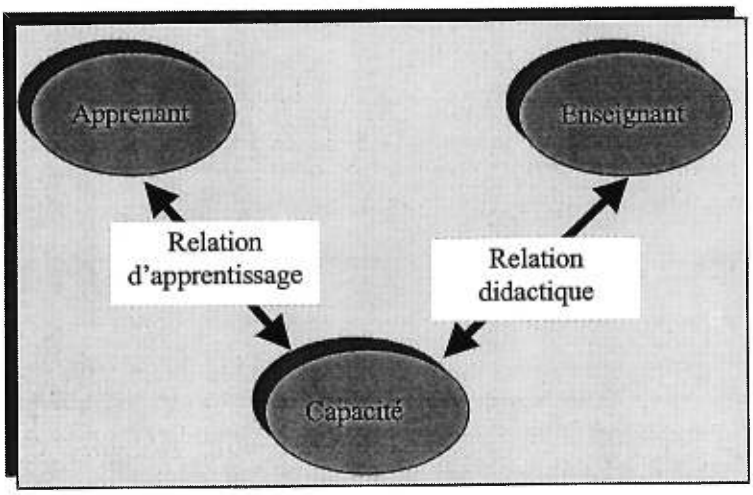


Figure 36 : relations pédagogiques dans un système tutoriel

6.1.3 Enseignement égal tâche coopérative

L'enseignement est un processus coopératif, s'il vérifie les quatre caractéristiques que nous avons retenues de la définition de la coopération. Ces caractéristiques sont : présence d'agents ou d'acteurs coopérants, présence d'un objectif commun pour tous les agents coopérants, utilisation de l'ordinateur comme terminal de communication et établissement d'un dialogue entre agents coopérants.

6.1.3.1 Présence d'agent coopérant dans l'enseignement

D'après la définition de l'enseignement, il apparaît que l'apprenant et l'enseignant exécutent chacun des opérations dans le processus d'enseignement. L'apprenant dans ce processus n'est plus considéré comme une personne qui se contente de recevoir et d'emmagasiner des renseignements, mais plutôt comme une personne apte à résoudre des problèmes et qui a une grande responsabilité dans le processus d'apprentissage. Les activités de l'apprenant sont qualifiées de processus d'apprentissage, alors que celles de l'enseignant sont qualifiées de processus didactique. On ne peut donc parler d'enseignement que si les deux acteurs majeurs du processus sont présents et sont actifs (enseignant, apprenant).

6.1.3.2 Présence d'un objectif commun

Dans un processus d'enseignement, le but est de susciter l'acquisition des connaissances, le développement d'habiletés, d'attitudes et de valeurs de l'apprenant. Ce but est poursuivi aussi bien par l'enseignant que par l'apprenant. Les agents participant au processus d'enseignement ont donc le même objectif.

6.1.3.3 Utilisation de l'ordinateur comme terminal de communication

La contrainte d'utiliser l'ordinateur comme terminal de communication est valable lorsqu'on veut définir la coopération dans l'informatique. Dans les STI le tuteur étant un module informatique, ses interactions avec les apprenants se font par l'intermédiaire de l'ordinateur.

6.1.3.4 Dialogue entre agents coopérants.

L'enseignement est vu comme un ensemble d'actes de communication. À ce titre, il existe un protocole d'échange de messages entre le tuteur et l'apprenant. Cet échange constitue le dialogue entre ces agents coopérants. Il s'effectue au moyen d'une manipulation coopérative de l'objet de l'enseignement (capacité).

6.1.3.5 Conclusion sur la nature coopérative de l'enseignement.

L'enseignement étant donc un processus coopératif, il est mis en place par un certain nombre d'agents. Construire un système pour l'enseignement c'est construire une société d'agents. Cette approche s'apparente à une modélisation qualitative de l'enseignement et permet donc une meilleure étude des phénomènes se produisant au niveau du STI. Pour exécuter le processus d'enseignement dans EDER_TC, nous devons le convertir sous forme d'objet MONACO_T. Nous nous sommes inspirés de l'analyse de l'enseignement faite par Gagné [Gagné, 79] pour en réaliser la décomposition. Nous commencerons par étudier l'organisation de l'enseignement avant de voir comment celui-ci est représenté dans MONACO_T.

6.2 Organisation de l'enseignement

Les recherches conduites depuis quelques décennies en psychologie de l'apprentissage par des chercheurs tels que Gagné, Bloom, Ausubel, Lindsay, Norman et Rumelhart [Gagné 72; Bloom 59/69; Lindsay & Norman 72; Norman & Rumelhart 75; Ausubel 68] ont permis de construire des modèles théoriques de l'apprentissage permettant d'améliorer l'acquisition des connaissances. Leur construction est principalement liée aux différents types d'informations ou capacités que l'humain emmagasine. Dans le cadre de notre travail, nous allons utiliser le modèle mis en place par Gagné. Ce choix s'impose dans la mesure où il est celui qui, en plus de la classification des capacités, propose des méthodes pour l'enseignement de celles-ci.

6.2.1 Classification des capacités selon Gagné

Selon Gagné, la plupart des informations manipulées par les humains peuvent se classifier en cinq classes : les informations verbales, les habiletés intellectuelles, les habiletés motrices, les attitudes et les stratégies cognitives. Le résumé de cette classification (tableau 6) est tiré du dictionnaire actuel de l'éducation [Legendre 93].

Tableau 6: Classification des capacités selon Gagné

Types de capacités	Sous-types	Définitions	Exemples
Informations verbales (représentation et communication de la réalité)	Idées	Représentation intellectuelle d'objets ou d'actions qu'un individu peut communiquer à d'autres individus	Définir le terme « Pollinisation ». Nommer les parties d'un micro-ordinateur
	Propositions	Groupe d'idées qui permettent à un individu de se représenter et de communiquer des événements	Décrire le cycle de l'eau dans la nature. Décrire le processus de la pollinisation des plantes
	Ensemble de propositions interreliées	Connaissances d'un individu dans un domaine précis	Résumer l'histoire du Cameroun Tracer le schéma de la création d'un logiciel d'enseignement
Habiletés intellectuelles (opérations mentales à exécuter)	Concepts concrets	L'individu peut se représenter des choses à l'aide de ses sens et les classifier	Identifier, parmi un ensemble de véhicules, ceux qui sont des autobus.

	Concepts définis	Niveau d'abstraction où les sens ne permettent pas de percevoir les caractéristiques du concept.	Sur un triangle rectangle, identifier l'hypoténuse.
	Règles	Conventions faites de concepts et d'opérations qui permettent d'effectuer des opérations	Accorder les participes passés avec l'auxiliaire avoir. Calculer la surface d'un cercle.
	Règles d'ordre supérieur	Ensemble de règles qui permettent d'effectuer des opérations ou d'associer une situation à une autre	Rédiger un rapport annuel. Effectuer des prévisions météorologiques.
Habiletés motrices		Ce type d'apprentissage génère des activités physiques et permet de développer des habiletés physiques	Dessiner une figure géométrique. Conduire une automobile. Sauter à la perche. Empiler des boîtes selon une méthode donnée.
Attitudes		Processus mental et affectif qui détermine le choix d'une action personnelle face aux choses, aux personnes ou aux événements	Choisir une saine alimentation. Choisir de protéger son environnement. Choisir d'arrêter de fumer.
Stratégies cognitives	Stratégies d'apprentissage	L'individu recherche, parmi divers stimuli, ceux qui favorisent le mieux un apprentissage	Utiliser la stratégie « X » pour apprendre des formules géométriques
	Stratégies de résolution de problèmes	L'individu utilise des informations, des concepts ou des règles qu'il connaît pour résoudre un problème	Développer une stratégie pour gagner au jeu du Monopoli.

6.2.2 Stratégies d'enseignement décrites par Gagné

À partir de la classification des capacités, Gagné a proposé des stratégies d'enseignement pour chaque type de capacité rencontré. Il existe donc une tâche d'enseignement particulière pour chacun de ces types. Le tableau décrivant les méthodes d'enseignement correspondantes (tableau 7) est tiré du dictionnaire de l'éducation [Legendre, 93].

Tableau 7: Stratégies d'enseignement en fonction des capacités

Type de capacités	Phase de motivation	Phase d'acquisition	Phase de performance
Informations verbales Idées et propositions	Donner à l'étudiant un aperçu de ce qu'il pourra faire à la fin de la leçon ou une notion générale de l'idée ou de la proposition à acquérir	Faire le rappel d'un contexte familier à l'étudiant auquel l'idée ou la proposition pourrait se rattacher. Utiliser des images ou des diagrammes, de l'animation ou des simulations.	Présenter des illustrations à compléter. Devinettes. Répétitions. Compléter des phrases ou des diagrammes. Si la liste de mots à apprendre est longue, faire un apprentissage fractionné du vocabulaire. Rétroaction appropriée.
Ensemble d'idées et de propositions interreliées	Délimiter le contenu à apprendre.	Fournir un schéma organisateur. Présenter l'information de façon logique et structurée. Utiliser des graphiques, des diagrammes, des animations ou des simulations pour faire le lien entre les idées ou propositions	Phrases à compléter. Rétroaction appropriée.
Habiletés intellectuelles Concept concret	Informer l'étudiant qu'il pourra, à la fin de la leçon, identifier dans un ensemble de cas, les exemples du concept.	Rappeler les discriminations nécessaires et les attributs-critères du concept. Présenter des exemples du concept en faisant ressortir les attributs-critères et comparer ces exemples avec ceux qui n'en sont pas. Présenter les attributs variables du concept	Identifier le concept dans des exemples et contre-exemples significatifs. Fournir une rétroaction appropriée.
Concept défini	Idem	Idem Rappeler les sous-concepts du concept à former. Aider l'étudiant au moyen de directives à associer les sous-concepts en un nouveau concept.	Idem
Règle	Fournir à l'étudiant un aperçu de ce qu'il pourra faire lorsqu'il maîtrisera la règle.	Rappeler les concepts et les opérations préalables à l'acquisition de la règle à former. Aider l'étudiant à associer les concepts et les opérations en fournissant l'énoncé de la règle et des exemples d'application	Fournir à l'étudiant des situations variées et nouvelles et lui demander d'appliquer la règle. Fournir une rétroaction appropriée.
Règle d'ordre	Informer l'étudiant du	Rappeler les règles et les	Fournir une variété de

supérieur	genre de problème qu'il pourra résoudre à la fin de l'unité	informations nécessaires à la solution de ce genre de problème.	problème à résoudre de même que les rétroactions appropriées.
Habiletés motrices	Fournir des illustrations du comportement à adopter en valorisant ce comportement	Rappeler les sous-routines. Enseigner l'habileté par parties, en utilisant des communications verbales, des illustrations, des démonstrations pour communiquer la routine maîtresse. Faire pratiquer les différentes parties de l'habileté et fournir la rétroaction appropriée.	Idem à l'acquisition. (Les phases acquisition et performance se réalisent toujours conjointement dans le cas des habiletés motrices).
Attitudes	Fournir des illustrations du comportement à adopter en valorisant ce comportement.	Inciter l'étudiant à se rappeler les informations et les attitudes nécessaires à l'atteinte de l'attitude. Inciter l'étudiant à s'identifier à un modèle. Faire adopter le comportement par le modèle en montrant la satisfaction qu'il en retire.	Fournir des occasions, réelles ou simulées où l'attitude peut s'appliquer et renforcer le comportement.
Stratégies cognitives	Piquer la curiosité en présentant une situation hypothétique et en posant une question. Donner un aperçu général de la stratégie, souligner les avantages que l'utilisation de la stratégie apportera dans l'exécution de tâches (économie de temps, facilité d'exécution, etc).	Faire le rappel des informations verbales et des habiletés intellectuelles susceptibles d'aider à la formation de la nouvelle stratégie. Décrire la stratégie et donner des exemples d'application.	Présenter de nombreux cas où la stratégie peut s'appliquer et demander de l'utiliser dans ces cas. Fournir une rétroaction appropriée (simulation, animation verbale, etc).

6.3 Implantation de l'enseignement à l'aide d'EDER_TC

Utiliser l'architecture EDER_TC pour faire de l'enseignement c'est construire une tâche coopérative d'enseignement pour chaque type de connaissances à enseigner. Cette contrainte est due aux conditions particulières requises par chaque connaissance pour son enseignement.

Pour montrer l'exploitation possible d'EDER_TC pour l'enseignement automatisé nous avons choisi le concept. D'après la classification de Gagné le concept fait partie de la classe des habiletés intellectuelles. L'objet MONACO_T à utiliser dans l'environnement EDER_TC doit intégrer la tâche d'enseignement et le système manipulé.

6.3.1 Modélisation du système concept exploitable par une tâche MONACO_T

Modéliser ce système concept revient à spécifier l'interface de communication entre la tâche et le système et ensuite définir les connaissances que doit posséder ce dernier pour être capable de répondre aux requêtes de la tâche coopérative qui le manipule.

6.3.1.1 Interface de communication du système concept

Elle permet de produire les ressources nécessaires pour l'enseignement d'un concept donné. Elle se base principalement sur la description de la tâche d'enseignement faite par Gagné. Les fonctions de cette interface se divisent en trois catégories : les fonctions utilisées en phase de motivation, celles utilisées en phase d'acquisition et celle utilisées en phase de performance.

Interface phase de motivation

Le système doit être capable de produire une ressource dont la présentation à l'apprenant permettra de le motiver à participer activement au processus d'enseignement. Cette phase implante une seule fonction : la fonction ressource de motivation.

Cette fonction utilise des paramètres tels que :

- le média de présentation (image fixe, image animée, écrit, parlé, combinaison quelconque de ces médias),
- le public cible (théoricien, praticien, novice, expert, etc.).

Interface phase d'acquisition

Au niveau de la phase d'acquisition de connaissances, le système concept doit produire plusieurs ressources répondant aux conditions décrites par Gagné pour la phase d'acquisition de l'enseignement de concept. Ces ressources permettent d'atteindre les objectifs suivant : rappeler les discriminations, présenter les attributs caractéristiques,

présenter les exemples du concept, présenter les contre-exemples, présenter les attributs variables. Ces différentes ressources peuvent utiliser différents types de médias. Les fonctions de communication disponibles pour cette phase sont :

La définition du concept : cette fonction prend en paramètre le public cible, le média et produit une définition du concept à enseigner compatible avec le public cible et utilisant le média sélectionné.

Les variables caractéristiques du concept : produit une ressource qui présente les variables caractéristiques du concept et les valeurs possibles pour chacune. Une variable caractéristique est une variable dont la valeur permet de déterminer si un objet est ou n'est pas représentatif du concept. Si on parle du concept de voiture, le nombre de roue est une variable caractéristique. Les paramètres sont le public cible et le média à utiliser.

Les variables non caractéristiques du concept : à l'image des variables caractéristiques, elle permet de générer une ressource qui présente les variables non caractéristiques du concept et les valeurs qu'elles peuvent prendre. Les variables non caractéristiques d'un concept sont les variables dont la valeur ne détermine pas l'appartenance ou la non-appartenance d'un objet au concept. Si on parle du concept de voiture, la couleur est une variable non caractéristique. Les paramètres utilisés sont les mêmes que ceux de la fonction variables caractéristiques.

Exemple du concept : construit une ressource présentant un exemple du concept. Les paramètres de cette fonction sont le média et degré de complexité.

Valeurs caractéristiques d'un exemple : cette fonction permet de présenter pour un exemple donné ses valeurs pour les variables caractéristiques du concept. Les paramètres pour cette fonctions sont le média et l'exemple à analyser.

Valeurs non caractéristiques d'un exemple : cette fonction permet de présenter les valeurs des variables non caractéristique pour un exemple donné. Elle prend en paramètre le média et l'exemple à scruter.

Contre-exemple : semblable à exemple à la différence que c'est un contre-exemple qui est passé en paramètre à la fonction. Les paramètres utilisés pour cette fonction sont le média et le degré de voisinage par rapport à la définition du concept.

Valeurs caractéristiques contre-exemple : idem .

Valeurs non caractéristiques d'un contre-exemple : idem.

Caractéristiques non vérifiées : cette fonction permet de donner les caractéristiques qu'un contre-exemple viole par rapport à la définition du concept. En paramètre on a le média à utiliser et le contre exemple incriminé.

Exception exemple : une exception est en fait un contre-exemple qui vérifie toutes les caractéristiques que peut avoir un exemple. La fonction est identique à celle du contre-exemple.

Valeur caractéristique exception exemple : identique à valeur caractéristique exemple.

Description d'une variable caractéristique : permet de produire une ressource présentant une variable caractéristique particulière. Les paramètres sont le média et la variable caractéristique.

Interface phase de performance

Dans cette phase Gagné recommande de présenter des ressources qui permettent d'identifier le concept dans des exemples et contre-exemples significatifs et ensuite de fournir une rétroaction. La mise en œuvre peut être faite au moyen de questions à choix multiple (QCM).

Une première fonction de cette phase est donc de générer des exercices sous formes de QCM pour vérifier l'assimilation du concept. Les paramètres pour cette fonction sont le média, le nombre d'exemples, le nombre de contre exemples et le degré de difficulté.

Il faut définir ce que doit être la rétroaction dans ces cas. Un exemple de scénario peut être le suivant.

L'apprenant désigne un contre-exemple comme étant un exemple du concept. Le tuteur demande à l'apprenant les valeurs des variables caractéristiques pour ce choix. L'apprenant répond à la question. Si la réponse est juste, le tuteur conclut qu'il ne connaît pas la règle de discrimination du concept et la lui présente. Si la réponse est fausse, le tuteur lui rappelle les valeurs des variables caractéristiques pour ce choix.

Les autres fonctions sont :

Évaluation d'un QCM : cette fonction permet de retourner l'évaluation de la solution à un QCM proposé par un apprenant. Ce résultat est principalement composé de la liste des mauvaise réponses

Question à trou : cette fonction renvoie une ressource qui permet à l'apprenant de répondre à une question de détermination des valeurs des variables caractéristiques pour un exemple, contre-exemple ou exception.

Évaluation d'une question à trou : renvoie une ressource permettant à l'apprenant de savoir si ses croyances sont justes ou pas.

Remarques

Ces fonctions ne sont pas exhaustives.

On pourrait également ajouter une phase d'évaluation dans le processus d'enseignement tel que le décrit Gagné. Cet ajout n'entraîne pas de modification dans l'interface de communication du système concept car l'évaluation peut être vue comme des exercices de performance sans rétroaction autre que l'évaluation du résultat.

6.3.1.2 Base de connaissance exploitée par le système concept

Ce système est amené à répondre à une multitude de fonctions sur un concept donné. Pour ce faire, il doit gérer une base de connaissances très détaillée (figure 37). Un concept est composé des éléments suivants :

Nom : permet d'identifier un concept parmi un ensemble. On a, par exemple, le concept « figure géométrique » ou « véhicule ».

Définition : la définition est elle-même un objet complexe renfermant les différentes définitions du concept. Ces définitions varient suivant le média à utiliser ou suivant le public cible. Ainsi on peut avoir une définition qui utilise des images fixes, des images animées, du texte écrit, du texte oral ou d'une combinaison quelconque de ces médias. La définition peut également être théorique, pratique ou une combinaison des deux.

Motivation : ce champ contient les ressources permettant de justifier l'importance qu'il y a à maîtriser ce concept. Ce composant du concept est également complexe. Il permet de représenter les ressources de motivation suivant le média et suivant le public cible.

Propriétés caractéristiques : ce champ contient la liste des propriétés caractéristiques du concept. Une propriété caractéristique est une variable du concept dont certaines valeurs permettent de discriminer le concept parmi un ensemble de concepts. Par exemple pour le carré, une propriété caractéristique est le nombre de côtés. Si la valeur est trois la figure géométrique n'est pas un carré alors qu'avec une valeur de quatre la figure est candidate. Chaque propriété caractéristique comporte son domaine de définition et les opérateurs de comparaison qui peuvent être utilisés sur elle. Pour l'exemple du concept de carré, le domaine de définition de la variable nombre de côté est l'ensemble des entiers naturels et l'ensemble des opérateurs contient tous les opérateurs utilisables avec les entiers naturels.

Propriétés non caractéristiques : Ce sont des propriétés qui entrent dans la description d'un concept, mais ne permettent pas de discriminer. Pour l'exemple du concept de carré, nous dirons que la propriété taille du côté est une propriété non caractéristique de ce concept.

Exemples : ce champ contient l'ensemble des exemples du concept. Chaque exemple est une instanciation du concept. Il permet ainsi de mieux fixer les idées par rapport à un concept. Par exemple pour le concept du carré, un exemple serait l'image d'un carré sur un tableau.

Contre-exemples : ce champ contient tous les contre-exemples du concept modélisé. Un contre-exemple est un objet voisin des objets représentant le concept. Il permet de mieux appréhender les objets qui appartiennent au concept. Par exemple pour le concept de carré, un contre-exemple serait un losange.

Exception : contient toutes les exceptions du concept. Une exception représente un objet qui respecte toutes les caractéristiques du concept mais qui ne désigne pas un objet de ce concept. Par exemple, pour le concept des mots dont le pluriel se termine par « aux », on aura comme exemple d'exception le mot « carnaval ».

Règle de discrimination : règle permettant d'évaluer si un objet donné fait partie des exemples ou des contre-exemples d'un concept. Cette règle est construite sous une forme où le système concept peut l'exploiter pour évaluer des objets qui lui sont présentés. C'est une disjonction de conjonctions de termes. Un terme étant représenté par le trio <variable caractéristique> <opérateur de comparaison> <valeur caractéristique>. Pour

l'exemple du concept du carré, la règle de discrimination est : (nombre de cotés = 4) et (longueur des cotés = toutes égales) et (angles entre cotés = 90 degrés).

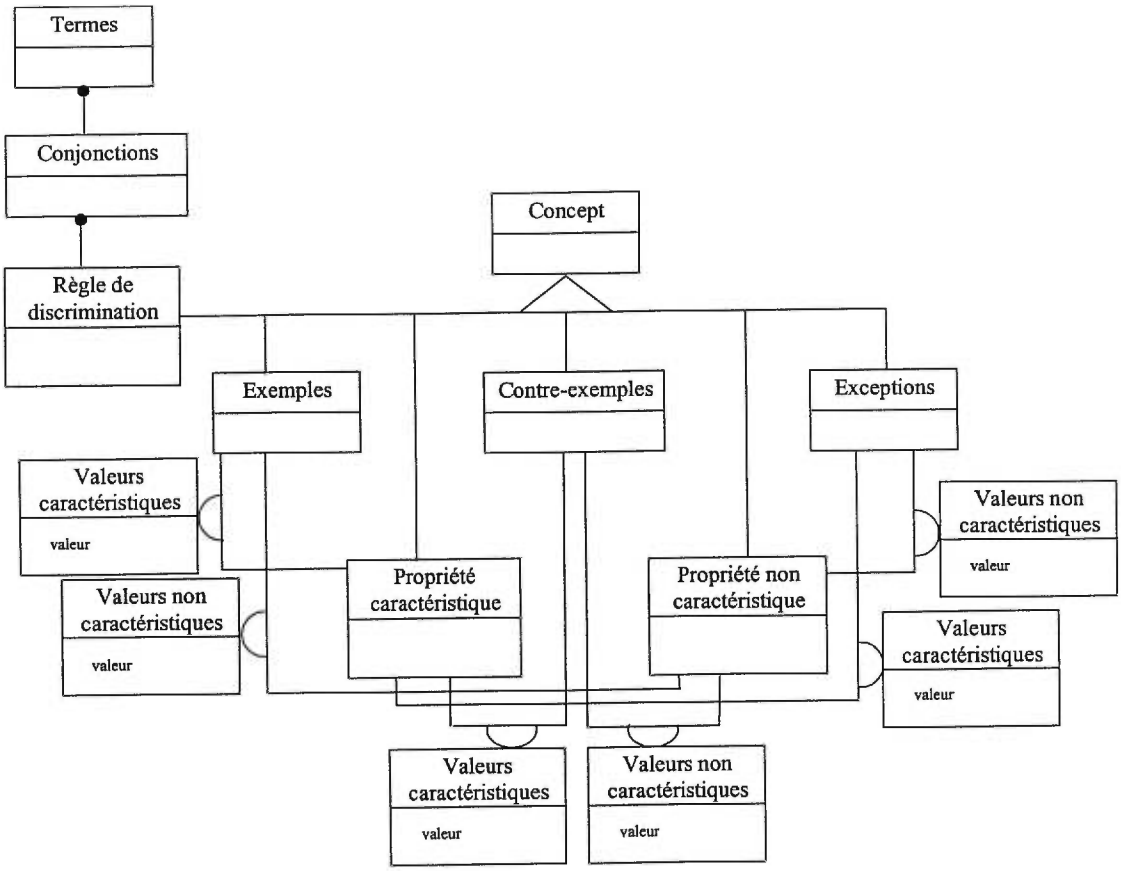


Figure 37 : Modèle objet simplifié du concept

Structure de l'exemple d'un concept

Pour que le système concept qui exploite l'objet concept puisse générer toutes les ressources nécessaires pour répondre aux problèmes de rétroaction, l'objet concept doit avoir des exemples dont le degré de détail permet une analyse fine des actions de l'apprenant. Les champs que nous proposons pour structurer un exemple de concept sont les suivants :

Désignation : ce champ contient les différentes présentations de l'exemple du concept selon le média et le public cible.

Degré de complexité : ce champ permet de savoir la complexité de l'exemple. Il peut être simple, moyen ou complexe.

Valeurs caractéristiques : ce champ comporte pour chaque valeur caractéristique du concept la valeur que prend l'exemple. Ces informations se présentent sous la forme d'un ensemble de couples (Propriété caractéristique, Valeur).

Valeurs non caractéristiques : identique à « valeurs caractéristiques » à la seule différence que l'on fait référence aux propriétés non caractéristiques du concept. Elles se présentent sous la forme d'un ensemble de couples (Propriété non caractéristique, Valeur).

Structure du contre-exemple d'un concept

La structure est la même que dans le cas de l'exemple à l'exception du degré de complexité qui est remplacé par un degré de voisinage. Ce champ permet de savoir jusqu'à quel point le contre-exemple ne correspond pas aux caractéristiques du concept. Le domaine de définition est : {TrèsVoisin, Voisin, Différent, ...}

Structure de l'exception d'un concept

L'exception étant un objet qui respecte les contraintes d'un exemple sans pour autant en être un, la structure adoptée pour le représenter est exactement la même que celle utilisée pour l'exemple.

6.3.2 Modélisation de la tâche d'enseignement de concept à l'aide de MONACO_T

Les recherches en psychologie de l'apprentissage ont permis de construire des modèles pour l'enseignement des différentes connaissances telles que : les informations verbales, les habiletés intellectuelles, les stratégies cognitives, les habiletés motrices et les attitudes. Gagné a synthétisé les résultats obtenus et a fourni un cadre conceptuel pour la planification et le développement d'activités d'enseignement. Gagné [Gagné 74/76] cité dans [Brien 92] considère que la tâche d'enseignement consiste en une suite d'exécutions de processus mentaux. Ces processus mentaux ont été regroupés en trois phases : motivation, acquisition et performance (figure 38). Gagné a décrit pour chaque phase, les événements susceptibles de provoquer les processus mentaux chez l'apprenant.

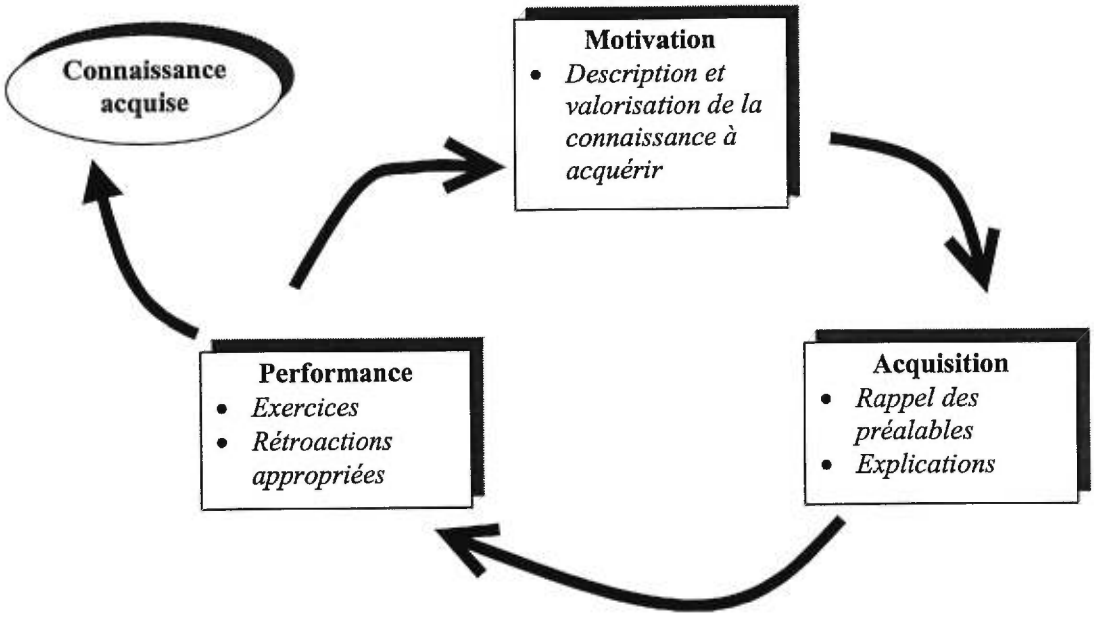


Figure 38 : Phases du processus d'enseignement

Nous allons nous inspirer des propositions de Gagné pour construire une tâche suivant la norme MONACO_T qui permette à un enseignant d'assurer l'apprentissage d'un concept.

Décrire une tâche suivant MONACO_T revient à construire la hiérarchie des sous-tâches et la distribution de ces sous-tâches au différents postes de travail impliqués (statique de la tâche), les règles régissant le comportement des sous-tâches (dynamique de la tâche) et les règles décrivants des stratégies particulières d'exécution de la tâche coopérative. (couche scénario de la tâche).

6.3.2.1 Couche statique de la tâche d'enseignement de concept

Au niveau de la couche de la statique, nous allons décrire l'ensemble des postes de travail, l'ensemble des sous-tâches, les liens de composition et l'ensemble des actions.

L'ensemble des postes de travail est composé des éléments suivants : l'enseignant et l'apprenant.

L'ensemble des actions est composé des fonctions de communications du système concept définies précédemment.

Dans la mesure ou l'ensemble des sous-tâches forme une hiérarchie, nous allons tout d'abord donner le principe qui nous a guidé dans la décomposition, ensuite nous décrirons chacune de ces sous-tâches quel est le poste de travail qui est chargé de son exécution et quelle action du système concept elle déclenche.

Principe de décomposition de sous-tâches

La tâche d'enseignement est divisée en trois sous-tâches, chacune correspondant à une des phases de l'enseignement décrit par Gagné.

Chaque type de ressource intervenant dans une phase est considéré comme une sous-tâche qui est subdivisée en fonction des activités que l'apprenant ou l'enseignant peut avoir à effectuer.

Chaque sous-tâche interne pouvant être considérée comme un sous-objectif à par entière, il lui est associé un ensemble de sous-tâches permettant de justifier son existence à un apprenant.

Les sous-tâches avec les préfixes « Demande » et « Validation » sont attribuées au poste de travail apprenant alors que celles préfixées par le terme « Production » sont attribuées à l'enseignant.

Seules les sous-tâches feuilles peuvent être le siège d'une fonction de communication du système concept. Les activités dans les sous-tâches internes permettent de mettre à jour les variables caractérisant la tâche coopérative.

Description des différentes sous-tâches

Dans cette section, nous ne décrirons que quelques sous-tâches. La version que nous avons implanté [Tadié & al, 97b] en comporte environ 150. Le protocole pour décrire les différentes sous-tâches met en exergue les informations suivantes :

Nom : contient le nom de la sous-tâche.

Objectif : décrit l'objectif de cette sous-tâche dans la tâche coopérative d'enseignement.

Action du système : désigne la fonction de communication du système concept qu'il faut déclencher lorsque l'on désire réaliser cette sous-tâche.

Poste de travail : désigne le poste de travail qui est chargé de la réalisation de la sous-tâche.

Liste des sous-tâche : contient la liste des sous-tâches de la tâche courante.

Tableau 8: Protocole de la sous-tâche « enseignement de concept »

Nom :	Enseignement de concept (EC)
Objectif :	Décrit le processus global de l'enseignement d'un concept quelconque
Action du système	Néant
Poste de travail :	Enseignant
Liste des sous-tâches :	Demande objectif EC, Production objectif EC, Validation objectif EC, Motivation, Acquisition, Performance

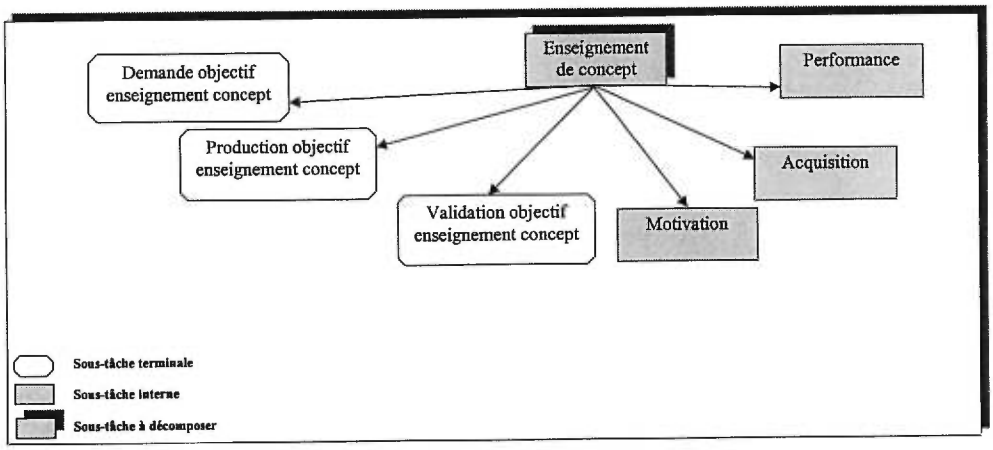


Figure 39 : Décomposition de la sous-tâche Enseignement de concept

Tableau 9: Protocole de la sous-tâche « Motivation »

Nom :	Motivation (M)
Objectif :	Décrit les activités permettant de valoriser le concept à acquérir
Action du système	Néant
Poste de travail :	Enseignant
Liste des sous-tâches :	Demande objectif Motivation, Production objectif Motivation, Validation objectif Motivation, Demande ressource Motivation, Production ressource de Motivation, Validation ressource de Motivation

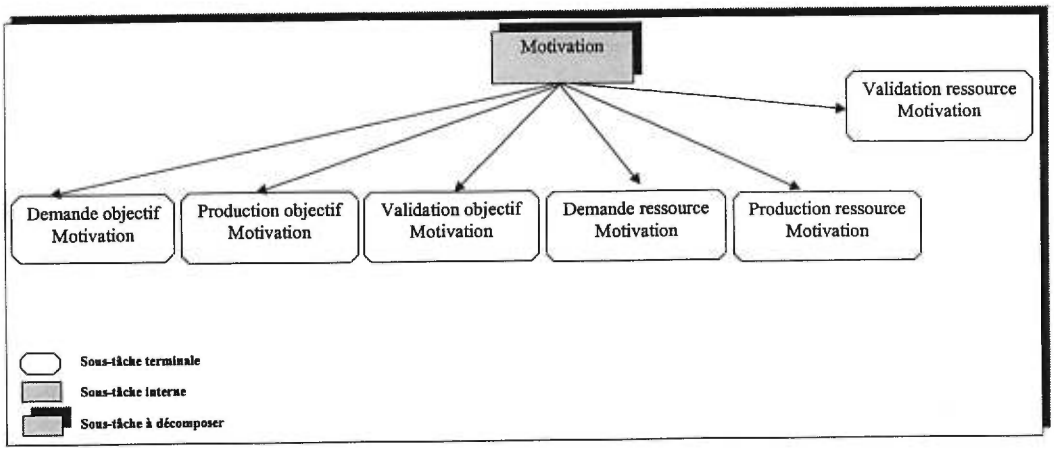


Figure 40 : Décomposition de la sous-tâche Motivation

Tableau 10: Protocole de la sous-tâche « Acquisition »

Nom :	Acquisition (A)
Objectif :	Décrit les activités permettant à l'apprenant de codifier et d'emmagasiner en mémoire le concept enseigné
Action du système	Néant
Poste de travail :	Enseignant
Liste des sous-tâches :	Demande objectif Acquisition, Production objectif Acquisition, Validation objectif Acquisition, Préalable, Description, Règle de discrimination, Attribut caractéristiques, Attribut non caractéristiques, Exemple, Contre-exemple, Exception

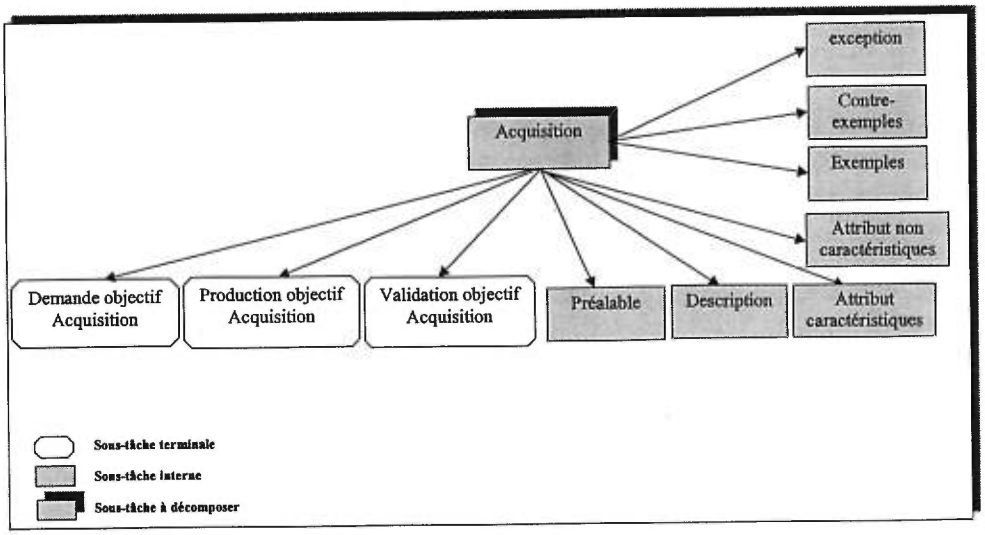


Figure 41 : Décomposition de la sous-tâche « Acquisition »

Tableau 11: Protocole de la sous-tâche « Exemple »

Nom :	Exemple (E)
Objectif :	Décrit les activités exécutable lorsque l'enseignant a décidé de traiter un exemple du concept
Action du système	Néant
Poste de travail :	Enseignant
Liste des sous-tâches :	Demande objectif Exemple, Production objectif Exemple, Validation objectif Exemple, Demande Exemple, Production Exemple, Validation Exemple, Demande valeurs caractéristiques Exemple, Production valeurs caractéristiques Exemple, Validation valeurs caractéristiques Exemple, Demande valeurs non caractéristiques Exemple, Production valeurs non caractéristiques Exemple, Validation valeurs non caractéristiques Exemple

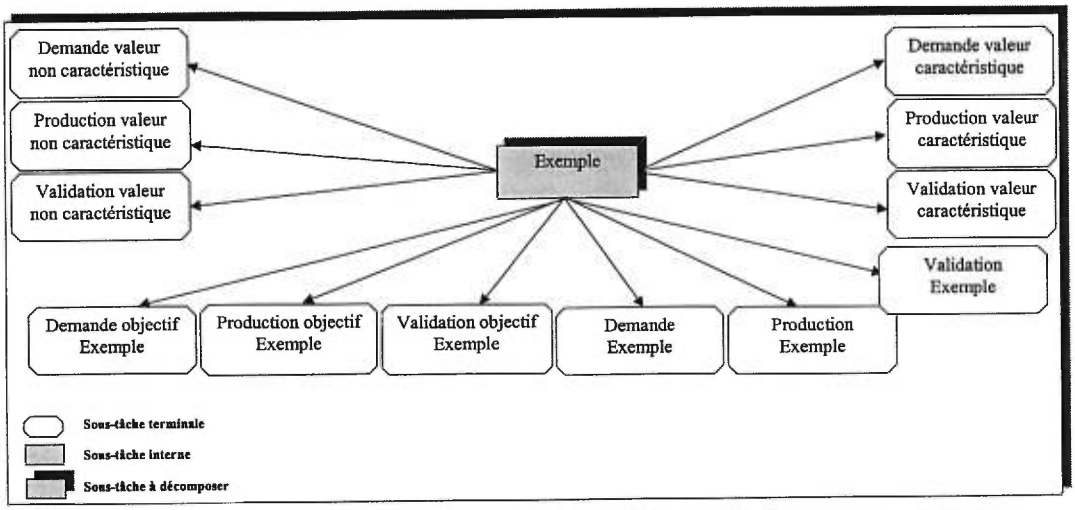


Figure 42 : Décomposition de la sous-tâche « Exemple »

Tableau 12: Protocole de la sous-tâche « Production Exemple »

Nom :	Production Exemple (PE)
Objectif :	Décrit l'activité consistant à présenter un exemple du concept à l'agent apprenant
Action du système	Exemple concept (média, degré de complexité)
Poste de travail :	Enseignant
Liste des sous-tâches :	Néant

Tableau 13: Protocole de la sous-tâche « Demande Exemple »

Nom :	Demande Exemple (DE)

Objectif :	Décrit l'activité consistant à demander à l'agent enseignant un exemple du concept
Action du système	Néant
Poste de travail :	Apprenant
Liste des sous-tâches :	Néant

6.3.2.2 Couche dynamique de la tâche d'enseignement de concept

Définir la couche de la dynamique revient à construire les règles qui permettent de gérer le moment d'exécution de chaque sous-tâche. Dans la mesure où l'exécution d'une sous-tâche est subdivisée en trois étapes, déclenchement, réalisation et terminaison, nous devons construire les règles.

Principe de construction des règles de la dynamique

Nous avons classé les principes qui nous ont guidé dans cette construction en trois blocs : les principes pour les règles de déclenchement, ceux pour les règles de réalisation et ceux pour les règles de terminaison

Règles de déclenchement. En fonction des types de sous-tâches, nous définissons des comportements particuliers. Les différents principes pour les règles de déclenchements sont les suivants :

- Pour les sous-tâches internes, il faut que la sous-tâche père soit déjà déclenchée et ne soit pas encore terminée.
- Pour les sous-tâches de type « Production Objectif », il faut que la sous-tâche père soit déjà déclenchée et ne soit pas encore terminée ou alors que « Demande Objectif » ait été effectué.
- Pour les sous-tâches de type « Demande Objectif », il faut que la sous-tâche père soit en cours de traitement (une sous tâche est en cours de traitement si le chemin qui va de l'action courante à la racine de la tâche passe par elle).
- Pour les sous-tâches de type « Validation Objectif », il faut que la sous tâche « Production objectif » correspondante ait été déclenchée et que la sous-tâche « Validation Objectif » n'aie pas encore été déclenchée .

- Pour les sous-tâches de type « Production quelconque », il faut que la sous-tâche demande correspondante ait été effectuée ou que la tâche d'enseignement ait été déclenchée ou alors que la production d'une autre sous-tâche puisse être complétée par la réalisation de cette sous-tâche.
- Pour les sous-tâches de type « Demande quelconque », il faut que la sous-tâche racine soit déjà déclenchée et ne soit pas encore terminée.
- Pour les sous-tâches de type validation, il faut que la production correspondante ait été effectuée et que la validation n'ait pas été déclenchée

Règles de réalisation. Au niveau de l'étape de réalisation d'une sous-tâche, le principe est suivant :

Pour les sous-tâches terminales, il faut que la sous-tâche soit déclenchée pour que la réalisation puisse s'effectuer.

Pour les sous-tâches internes, la condition de réalisation de l'état d'exécution doit être vérifiée.

Règles de terminaison. Au niveau de l'étape de terminaison, le principe utilisé est le suivant :

- Pour les sous-tâches terminales, la sous-tâche père doit être terminée.
- Pour les sous-tâches interne, la règle produite fait intervenir les variables caractéristiques de ces sous-tâches.

Description de la dynamique de quelques sous-tâches

Pour les sous-tâches dont nous allons définir la dynamique, le protocole sera le suivant : nom de la sous-tâche, condition de déclenchement, sémantique de la règle de déclenchement, condition de réalisation, sémantique de la règle de réalisation, condition de terminaison, sémantique de la règle de terminaison.

Tableau 14: Dynamique de la sous-tâche « Enseignement de concept »

Nom sous-tâche	Enseignement de concept (EC)
Condition de déclenchement	EC.Etat = Initial
Sémantique condition	Pour déclencher l'enseignement de concept (EC), il faut que cette

déclenchement	sous-tâche soit à l'état initial
Condition de réalisation	Acquisition.Etat = Réalisé OU Performance.Etat = réalisé
Sémantique condition réalisation	Pour réaliser la sous-tâche EC, il faut que la sous-tâche Acquisition soit déjà réalisée ou que la sous-tâche Performance soit réalisée
Condition de terminaison	EC.Etat = Réalisé
Sémantique condition déclenchement	Pour terminer la sous-tâche enseignement de concept (EC), il faut que cette sous-tâche soit déjà réalisée

Tableau 15: Dynamique de la sous-tâche « Motivation »

Nom sous-tâche	Motivation (M)
Condition de déclenchement	M.Etat = Initial AND EC.Etat = Déclenché AND SubTaskProduction.Etat <> Réalisé
Sémantique condition déclenchement	Pour déclencher la sous-tâche Motivation (M), il faut que cette sous-tâche soit à l'état initial et que la sous-tâche Enseignement de concept soit déjà déclenchée et qu'aucune sous-tâche de type production ne soit à l'état réalisé. Cette dernière contrainte signifie que l'enseignant ne peut déclencher cette sous-tâche si une sous-tâche de type production est dans l'état réalisé. Les sous-tâches de type production sont les sous-tâches de l'enseignant qui demande une réaction de l'apprenant avant qu'il ne puisse exécuter une autre sous-tâche quelconque. Par exemple si l'enseignant présente un questionnaire à choix multiples (QCM) ou un exemple à un apprenant, il faut absolument que l'apprenant réagisse à la ressource qui a été présentée avant qu'il ne puisse continuer à agir sur la tâche. Si on veut contrôler le temps de réaction de l'apprenant, il faut créer une sous-tâche chronomètre qui se déclenche à un moment donné et fait passer la sous-tâche Production bloquante de l'état réalisé à l'état suspendu.
Condition de réalisation	ValidationMotivation.Etat = Réalisé
Sémantique condition réalisation	Pour réaliser la sous-tâche Motivation, il faut que la sous-tâche Validation ressource Motivation exécutée par l'apprenant soit dans l'état réalisé.
Condition de terminaison	M.Etat = Réalisé
Sémantique condition déclenchement	Pour terminer la sous-tâche Motivation (M), il faut que cette sous-tâche soit déjà réalisée

Les sous tâches internes manipulées par l'enseignant ont à peu près le même comportement que la sous-tâche « Motivation » à l'exception des conditions de réalisations qui doivent dépendre des sous-tâches de chaque tâche interne.

Tableau 16: Dynamique de la sous-tâche « Production Exemple »

Nom sous-tâche	Production Exemple (PE)
Condition de déclenchement	(PE.Etat = Initial AND Exemple.Etat = Déclenché AND SubTaskProduction.Etat <> Réalisé) OR

	(DemandeExemple.Etat = Réalisé)
Sémantique condition déclenchement	Pour déclencher la sous-tâche Production Exemple (PE), il faut que cette sous-tâche soit à l'état initial et que la sous-tâche Exemple soit déjà déclenchée et qu'aucune sous-tâche de type production ne soit à l'état réalisé. La deuxième possibilité de déclenchement de la sous-tâche Production Exemple est que la sous-tâche demande exemple soit à l'état réalisé. Cette dernière contrainte veut dire que lorsque l'apprenant fait la demande d'un exemple, quelque soit l'état de la tâche coopérative, la sous-tâche Production Exemple doit être disponible à un déclenchement
Condition de réalisation	ProductionExemple.Etat = Déclenché
Sémantique condition réalisation	Pour réaliser la sous-tâche Production Exemple, il faut cette sous-tâche ait été déclenchée.
Condition de terminaison	Production Exemple.NombreDeRéalisation > valeur
Sémantique condition déclenchement	Pour terminer la sous-tâche Production Exemple, il faut cette sous-tâche ait été réalisée au moins un certain nombre de fois.

Tableau 17: Dynamique de la sous-tâche « Demande Exemple »

Nom sous-tâche	Demande Exemple (DE)
Condition de déclenchement	(DE.Etat = Initial AND EC.Etat = Déclenché AND SubTaskDemande.Etat <> Réalisé)
Sémantique condition déclenchement	Pour déclencher la sous-tâche Demande Exemple (DE), il faut que cette sous-tâche soit à l'état initial et que la sous-tâche Enseignement Concept soit déjà déclenchée et qu'aucune sous-tâche de type demande ne soit à l'état réalisé. Ainsi donc si aucune demande faite par l'apprenant n'est en cours (SubTaskDemande.Etat <> Réalisé) et que le processus d'enseignement a déjà démarré (EC.Etat = Déclenché) alors l'apprenant peut déclencher la demande d'un exemple.
Condition de réalisation	DemandeExemple.Etat = Déclenché
Sémantique condition réalisation	Pour réaliser la sous-tâche demande Exemple, il faut cette sous-tâche ait été déclenchée.
Condition de terminaison	Demande Exemple.NombreDeRéalisation > valeur
Sémantique condition déclenchement	Pour terminer la sous-tâche Demande Exemple, il faut cette sous-tâche ait été réalisée au moins un certain nombre de fois.

6.3.2.3 Couche scénario de la tâche d'enseignement de concept

La couche scénario permet de définir des stratégies d'exécution de la tâche coopérative. Les stratégies qui nous importent le plus sont les stratégies par poste de travail. Définir une telle stratégie revient à définir les règles d'ordonnancement, les règles de coupures de sous-tâches et les règles de coupure de termes.

Dans le cas de l'enseignement de concept, nous allons voir quelles sont les stratégies que peut implanter le poste de travail enseignant et celles que peut implanter le poste de travail apprenant.

Bref rappel de quelques styles d'enseignement

Dans le domaine de l'éducation, plusieurs styles d'enseignement ont été définis [Legendre 93]. Nous avons retenu ceux qui peuvent le mieux s'appliquer à l'enseignement de concept et qui ne font intervenir que deux agents (apprenant, enseignant). Il s'agit des styles directif, entraîneur, interrogateur et accompagnateur.

Style directif : Ce style permet de favoriser l'apprentissage d'une connaissance spécifique avec exactitude et en peu de temps.

Rôle de l'enseignant : présenter des ressources, expliquer ces ressources, faire des démonstrations, demander à l'apprenant de faire des démonstrations, évaluer l'apprenant et donner la rétroaction nécessaire.

Rôle de l'apprenant : suivre les indications de l'enseignant, observer les démonstrations, imiter les démonstrations, répondre aux questions posées.

Style entraîneur : Ce style permet de favoriser l'apprentissage individuel d'une connaissance Il permet à l'apprenant de transformer ses échecs et réussites en expériences d'apprentissage.

Rôle de L'enseignant : planifier les activités, se rendre disponible pour répondre aux questions de l'apprenant, évaluer le travail de celui-ci et fournir une rétroaction, le surveiller et intervenir lorsque c'est nécessaire.

Rôle de l'apprenant : réaliser les activités de l'enseignement avec l'aide de l'enseignant.

Style interrogateur : Ce style permet à l'apprenant de découvrir un concept en répondant à une série de questions posée par l'enseignant.

Rôle de l'enseignant : Présenter une séquence de questions permettant la découverte de connaissances, fournir les rétroactions nécessaires.

Rôle de l'apprenant : lire les questions et les indices de l'enseignant, tenter de découvrir la réponse à chaque question.

Style accompagnateur : Ce style permet à l'apprenant d'être le maître d'œuvre du processus d'enseignement. Dans ce cas, l'apprenant planifie et réalise les activités liées à l'enseignement d'une connaissance.

Rôle de l'enseignant : déclencher le processus d'enseignement, observer la démarche de l'apprenant, activer uniquement les activités demandées par l'apprenant.

Rôle de l'apprenant : Choisir une ressource en fonction de l'objectif général déterminé par l'enseignant, exploiter la ressource et demander la prochaine ressource jusqu'à l'atteinte de l'objectif initial.

Implantation du style directif

Implanter le style directif revient à décrire pour chaque poste de travail les règles d'ordonnancement, les règles de coupures de sous-tâches et de termes.

Tableau 18: Description du poste de travail « Enseignant »

Règles d'ordonnancement	Dans ce style, l'enseignement suit l'ordre établi par Gagné. Les activités s'exécutent donc dans l'ordre suivant : motivation, acquisition et performance. Dans la phase d'acquisition l'ordre est le suivant : objectif, définition, description, exemple, exception et contre-exemple.
Règles de coupures de sous-tâches	Aucune sous-tâche n'est supprimée dans cette stratégie.
Règles de coupures de termes	Dans la mesure où l'enseignant dirige toutes les interventions, les termes permettant à un apprenant de demander la présentation d'une ressource sont supprimés. Par exemple pour la sous-tâche ProductionExemple la condition de déclenchement est : (PE.Etat = Initial AND Exemple.Etat = Déclenché AND SubTaskProduction.Etat <> Réalisé) OR (DemandeExemple.Etat = Réalisé) Le terme à supprimer est le suivant : DemandeExemple.Etat = Réalisé. supprimer ce terme fait en sorte que le déclenchement de cette sous-tâche ne dépend pas de la volonté de l'apprenant mais uniquement de celle de l'enseignant.

Tableau 19: Description du poste de travail « Apprenant »

Règles d'ordonnancement	Dans ce style, l'apprenant suit méticuleusement les ressources envoyées par l'enseignant. Il n'y a donc pas de règles d'ordonnancement car c'est l'enseignant qui dirige le processus d'enseignement.
Règles de coupures de	Toutes les sous-tâches permettant à l'apprenant de demander une ressource

sous-tâches	d'enseignement à l'enseignant sont supprimées. Exemple : les sous-tâches Demande exemple, Demande définition, Demande variables caractéristiques, etc.
Règles de coupures de termes	Pour cette stratégie, il n'y a pas de termes particulier à supprimer pour les sous-tâches appartenant à l'apprenant.

Nous implantons de la même manière les styles entraîneur, interrogateur et accompagnateur.

6.4 Utilisation d'EDER_TC pour l'enseignement de concept

Montrer comment EDER_TC permet de faire l'enseignement de concept, revient à décrire l'exploitation d'EDER_TC avec la tâche d'enseignement de concept modélisée à l'aide de MONACO_T. Dans cette exploitation, le poste de travail enseignant est géré par un agent informatique de simulation et celui de l'apprenant par un agent humain de simulation.

6.4.1 Comportement de l'agent automatique de simulation au poste de l'enseignant

Dans la phase d'exploitation, comme le prescrivent les agents automatique de simulation, le simulateur de l'enseignant reçoit les activités de dépendances fonctionnelles du gestionnaire de tâche coopérative, ensuite, il met à jour sa vue de la tâche coopérative, sélectionne l'action à exécuter en fonction des règles de la dynamique et de celles du scénario choisi et envoie son choix au gestionnaire de la tâche coopérative.

6.4.2 Comportement de l'agent humain de simulation au poste de l'apprenant

Il fournit une interface à un humain afin que celui-ci participe à la réalisation de la tâche coopérative d'enseignement de concept au poste de l'apprenant. En conséquence cet humain reçoit les ressources qui lui permettront d'apprendre le concept enseigné. Pour répondre aux contraintes de la tâche d'enseignement de concept, le module de présentation de cet agent doit pouvoir prendre en compte tous les types de ressources que produit le système concept (QCM, question à troue, texte, image fixe, image animée, son, etc.).

6.4.3 Chronologie des actions au cours du processus de simulation de la tâche coopérative d'enseignement de concept

Nous allons présenter ici quelques étapes du protocole de communication entre les agents d'EDER_TC participant à la simulation de la tâche coopérative d'enseignement de concept. Dans cet exemple, le scénario utilisé est de type directif. Les sous-tâches permettant à l'apprenant de manifester ses volontés ne sont pas accessibles.

Top Horloge	Agent gestionnaire de la tâche coopérative	Agent Enseignant simulé	Agent Apprenant humain
	Signal de démarrage		
		Déclenchement sous-tâche EC	Aucune action disponible
	Envoi activité de dépendance fonctionnelle (adf) pour mettre à jour les vues de MONACO_T		
		MAJ vue MONACO_T	MAJ vue MONACO_T
		Déclenchement sous-tâche Production Objectif Enseignement Concept(POEC)	Aucune action disponible
	Transfert ressource relatif à POEC + adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T + Présentation ressource POEC
		Aucune action disponible	Déclenchement et Réalisation Validation Objectif Enseignement Concept
	Envoi adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T
		Déclenchement sous-tâche Motivation	Aucune action disponible
	Envoi adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T
		Déclenchement sous-tâche Production Objectif Motivation (POM)	Aucune action disponible
	Transfert ressource relatif à POM + adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T + Présentation ressource POM
		Aucune action disponible	Déclenchement et Réalisation Validation Objectif Motivation

	Envoi adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T
		Déclenchement sous-tâche Acquisition	Aucune action disponible
	Envoi adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T
		Déclenchement sous-tâche Production Objectif Acquisition (POA)	Aucune action disponible
	Transfert ressource relatif à POA + adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T + Présentation ressource POA
		Aucune action disponible	Déclenchement et Réalisation Validation Objectif Acquisition
	Envoi adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T
		Déclenchement sous-tâche Préalable	Aucune action disponible
	Envoi adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T
		Déclenchement sous-tâche Production Rappel Préalable (PRP)	Aucune action disponible
	Transfert ressource relatif à PRP + adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T + Présentation ressource PRP
		Aucune action disponible	Déclenchement et Réalisation Validation Objectif Acquisition
	Envoi adf de MAJ		
		MAJ vue MONACO_T	MAJ vue MONACO_T
		Etc.	Etc.
		.	.

6.5 Utilisation d'EDER_TC pour enseigner comment enseigner des concepts

Utiliser EDER_TC pour enseigner comment enseigner des concepts revient à faire de l'enseignement de tâches coopératives à l'aide d'EDER_TC dans la mesure où l'enseignement est considérée comme une tâche coopérative.

Pour étudier cet aspect d'EDER_TC nous allons commencer par définir ce qu'est l'enseignement de tâches coopérative, ensuite nous verrons le comportement des différents agents d'EDER_TC lors du processus d'instruction d'un enseignant.

6.5.1 Présentation de l'enseignement de tâches coopératives

Dans un travail coopératif, seule une bonne performance collective compte car si on a de très bonnes individualités qui se distinguent à leur poste et que le résultat de l'équipe est mauvais alors toute l'équipe aura échoué. Dans le cas de la tâche d'enseignement, si l'enseignant connaît très bien les théories de l'enseignement sans pour autant être capable de les appliquer convenablement, alors on dira que l'enseignement a échoué.

Dans la formation d'une équipe, il y a deux grandes parties: une formation aux techniques individuelles (poste de travail), et une formation collective aux stratégies du travail en équipe.

6.5.1.1 La technique individuelle

Dans un travail d'équipe, chaque membre doit maîtriser les tâches qui lui incombent dans le groupe. Dans cette partie de la formation, on va donc définir et enseigner les tâches spécifiques à chaque poste de travail en simulant le travail aux autres postes de travail. Cette phase utilise beaucoup les spécifications faites dans le cadre des STI classiques. En plus de ces spécifications, l'enseignement de tâche coopérative doit utiliser des agents capables de s'adapter dynamiquement à l'apprenant humain.

Lorsque l'apprenant évolue dans l'exécution de sa tâche, les autres agents doivent faire évoluer leur poste de travail comme il se doit afin que l'objectif collectif soit poursuivi. Ainsi, ces agents participent au processus d'enseignement.

6.5.1.2 La technique collective

Ayant vu plus haut que les stratégies individuelles ne suffisaient pas pour faire une bonne équipe, on s'attelle dans cette partie à définir et à enseigner les tâches d'un poste non pas par rapport à lui-même, mais en relation avec les autres postes de travail qui participent à l'accomplissement du but final. En plus on définit et enseigne les méta-stratégies qui indiquent comment mener à bien les stratégies au niveau des différents postes de travail pour une performance optimale de l'équipe.

6.5.2 Comportement du gestionnaire du poste de travail enseignant

L'agent gérant le poste de travail enseignant dans cette situation est l'agent automatique pour l'enseignement. Il sert de tuteur à tout enseignant humain qui voudrait se former à l'enseignement de concept.

Le comportement de cet agent est caractérisé par les échanges entre l'humain et le tuteur du poste de travail. Ces échanges sont fait aussi bien au niveau du processus d'acquisition des connaissances individuelles qu'à celui de l'acquisition des connaissances collectives.

6.5.2.1 Comportement acquisition de connaissances individuelles

Dans cette étape de l'enseignement l'humain demande des informations telles que : l'objectif de l'enseignement de concept, la structure de la tâche d'enseignement de concept, les règles définissant le comportement des sous-tâches, la description des différentes stratégies du poste de travail, etc.

Le tuteur qui possède toutes ces informations y répond sans communiquer avec le gestionnaire de la tâche coopérative.

6.5.2.2 Comportement acquisition de connaissances coopératives

Dans cette phase, le tuteur met l'humain en situation d'enseignement avec un apprenant réel ou simulé au poste de travail apprenant. Le but ici est de permettre à l'élève enseignant de pratiquer les connaissances obtenues lors de la phase d'acquisition de connaissances individuelles. En fonction de l'état de l'enseignement il choisit les activités à réaliser et réagit aux interactions venant du poste de travail apprenant.

Lorsqu'une action non permise est activée, en fonction des règles de la dynamiques et de celles de la couche scénario, le tuteur intervient et signale l'erreur. Ensuite il peut répondre à une suite de questions venant de l'humain telles que : « pourquoi l'action ne peut-elle pas être réalisée? », « quelle action doit-on réaliser? », etc.

Par exemple devant une question du type : « pourquoi ne puis-je pas envoyer un QCM à l'apprenant? », le tuteur peut générer la réponse suivante : « il faut que l'apprenant valide la ressource qui lui a été transférée avant que tu ne puisses lui en envoyer une autre ».

Certaines réponses sont générées par coopération entre les différents agents participant à la tâche coopérative [Tadié & al., 98b].

6.5.3 Comportement du gestionnaire du poste de travail apprenant

Ce poste de travail est actif uniquement lorsque l'architecture est dédiée à l'enseignement de connaissances coopératives. Ce poste peut être géré par un agent de simulation automatique ou humain.

Il permet de réagir aux actions effectuées par le poste de travail enseignant et participe à la génération de conseils coopératifs délivrés à l'humain qui manipule ce poste.

6.6 Utilisation d'EDER_TC pour faire de l'enseignement coopératif

L'enseignement coopératif ou encore apprentissage coopératif est défini comme le mode d'apprentissage où les apprenants cheminent en petit groupe autour d'un même objet d'étude ou d'un projet. Dans cette section, nous allons commencer par voir les fondements théoriques de l'enseignement coopératif. Nous verrons ensuite l'état de l'art de l'introduction de cette stratégie dans les STI. Nous terminerons par le cas particulier du système compagnon implanté à l'aide d'EDER_TC.

6.6.1 Fondement théorique de l'enseignement coopératif

L'apprentissage coopératif est une des idées qui ont bouleversé l'organisation du processus d'apprentissage dans nos écoles durant la dernière décennie.

Dans le domaine de l'éducation, les recherches sur les stratégies d'enseignement coopératif n'ont vu le jour que dans les années 1980. Le principe de cette méthode est basé sur l'organisation des apprenants en petit groupe. Ainsi la majeure partie de l'apprentissage se fait à travers les discussions que les apprenants ont entre eux. Ils apprennent donc dans ce système à se prendre en charge eux même.

L'un des objectifs principaux de l'enseignement coopératif est de favoriser l'esprit d'équipe au lieu de la rivalité individuelle à laquelle le système classique nous a habitué. Les apprenants dans un travail en équipe sont là pour s'entraider et le succès de chaque membre rejaillit sur les autres.

Vygotsky dans ses travaux [Vygotsky, 78] émet les hypothèses selon lesquelles les interactions sociales dans l'apprentissage ont un rôle fondamental dans la structuration cognitive des apprenants.

Les recherches de Slavin dans [Slavin, 89] ont prouvé que les étudiants travaillant en coopération donnent de meilleurs résultats que si chacun d'eux travaillait individuellement.

L'approche coopérative de l'enseignement permet de voir l'enseignement sous un autre angle que celui plus classique où l'apprenant est comme un receveur passif en ce sens qu'il est là uniquement pour enregistrer ce que l'enseignant lui propose. Dans le système coopératif, les apprenants sont très actifs en s'aidant mutuellement..

Dans les systèmes coopératifs, l'enseignant se comporte comme un entraîneur, il gère les activités des apprenants qui travaillent [Adams & al., 90].

Dans [Slavin, 89] une analyse des résultats de recherche sur l'apprentissage coopératif a montré que cette méthode était plus performante que la méthode classique sur le plan de l'atteinte des objectifs académiques, des relations sociales entre étudiants, de l'estime que chaque étudiant avait de lui.

On a également constaté que l'anxiété, le stress et la recherche de productivité sont les problèmes générés par l'apprentissage classique qui inhibent le développement de la personne [Adams & al., 90].

Par essence, apprendre est un processus plus coopératif que compétitif. Créer un environnement coopératif d'apprentissage, c'est créer un environnement où il est possible de faire des erreurs et où on apprend de ses erreurs.

L'un des problèmes qui se pose dans les systèmes coopératifs est la résolution de conflits qui peuvent intervenir entre les membres d'un groupe de travail. Ainsi ses membres peuvent ne pas s'entendre sur la solution à adopter lorsqu'il y a plusieurs alternatives possibles pour un problème donné. Pour éviter ces conflits, nous proposons cinq étapes à suivre dans le processus de résolution d'un problème par un groupe d'apprenants:

- définir le problème à résoudre,
- générer les différentes alternatives de résolution,
- examiner les avantages et inconvénient de chaque alternative,
- implanter la meilleure alternative,
- évaluer le résultat obtenu.

6.6.2 Intégration de l'apprentissage coopératif dans les STI: état de l'art.

L'influence des phénomènes sociaux dans les STI a été remarquée pratiquement depuis les années 1980. Ainsi dans SOPHIE [Brown & al., 82] les études ont montré que les équipes se comportaient mieux dans l'utilisation du logiciel que les individus travaillant individuellement avec le STI. Schoenfeld dans le cadre de la résolution de problème[Schoenfeld, 85] se rend compte également que le fait d'analyser les problèmes sous diverses perspectives, de planifier et d'évaluer de nouvelles idées est mieux appréhendé par un groupe de personnes.

Gilmore et Self [88] introduisent l'idée d'avoir un partenaire dans le STI à la place du tuteur. Dans ce cas le système se comporte comme un apprenant et travaille avec l'apprenant réel pour résoudre les problèmes qui se posent.

Dans le cas où on simule un apprenant, le système doit être vraisemblable en tant que partenaire en ce sens qu'il doit se comporter comme un partenaire réel l'aurait fait. Un des systèmes développés qui respecte cette norme est PEOPLE POWER [Dillenbourg & Self, 92] qui permet d'analyser l'apport des structures coopératives dans le dialogue réflexif.

Dans ce système, le raisonnement est considéré comme un dialogue interne et la collaboration comme un dialogue entre apprenants.

Chan et Baskin [88] ont introduit le système d'apprentissage avec compagnon. Dans ce cas, il existe toujours un tuteur dans le système mais en plus le système met à la disposition de l'apprenant un compagnon simulé qui travaillera avec lui. Ce principe a été mis en œuvre dans Integration-Kid [Chan, 91], qui permet de former des apprenants aux techniques d'intégration de fonction. Dans ce cas le mode de coopération consiste à partager les tâches entre le compagnon et l'apprenant réel. Pendant que l'un est responsable du choix de la prochaine opération à résoudre, l'autre est responsable de l'exécution.

Ces deux systèmes sont les seuls qui peuvent être considérés comme des systèmes tutoriels intelligents supportant l'apprentissage coopératif.
D'autres systèmes comme le Turtle Graph qui permet de faire travailler des étudiants réels en coopération, peuvent être considérés comme des systèmes supportant le travail coopératif. Turtle Graph permet de faire travailler deux étudiants réels sur un problème, chaque étudiant étant sur son poste de travail. Ce système ne comporte pas de tuteur ni de modèle d'élève, il permet simplement de mettre en œuvre les mécanismes de communication pour qu'il y ait dialogue entre les deux apprenants. Ce type de système est plus proche des CSCW (Computer Supported Cooperative Work) que des STI coopératifs.

6.6.3 Implantation du compagnon à l'aide d'EDER_TC

Implanter la stratégie du compagnon à l'aide d'EDER_TC, revient à construire une tâche coopérative d'enseignement de concept utilisant trois agents au lieu de deux. Ces agents: l'enseignant, l'apprenant et le compagnon.

Les modifications à apporter à la tâche coopérative d'enseignement de concept avec enseignant et apprenant pour qu'il puisse permettre l'intégration d'un agent compagnon sont les suivantes :

- Toutes les demandes effectuées par l'apprenant sont dupliquées et dirigées vers le tuteur et le compagnon.

- Les productions correspondant à ces demandes sont également dupliquées et exécutées par le tuteur et par le compagnon.
- Les nouvelles productions doivent avoir des validations correspondantes.

Après cette première transformation, toutes les sous tâches appartenant à l'apprenant doivent être dupliquées et mises à la disposition du compagnon. Celles appartenant au compagnon sont également dupliquées et mises à la disposition de l'apprenant. Parmi ces nouvelles sous tâches, celles qui étaient dirigées vers le compagnon sont dirigées vers l'apprenant et vice versa.

Les règles de la dynamique doivent permettre d'implanter plusieurs protocoles d'interaction entre les agents enseignant, apprenant et compagnon. En plus des scénarios de l'enseignant (stratégie d'enseignement) et de celles de l'apprenant (stratégie d'apprentissage), nous devons construire des scénarios pour compagnon (stratégies de compagnonnage) qui permettent de simuler un comportement constructif à ce niveau.

7 Conclusion et Perspectives

7.1 Conclusion

L'objectif que nous avions dans cette thèse était de construire un système permettant de produire un enseignement et une simulation de tâches coopératives. Cette tâche nous a amené à travailler dans trois domaines distincts de l'informatique, les systèmes tutoriels intelligents (STI), les systèmes supportant le travail coopératif et les systèmes d'intelligence artificielle distribuée.

Les techniques d'intelligence artificielle distribuée nous ont permis, à partir d'une analyse cognitive de la structure d'une tâche coopérative, de construire un modèle de cette tâche. Plus précisément nous avons utilisé les modes de représentation des connaissances de l'intelligence artificielle. Les techniques des STI nous ont permis de trouver les informations à intégrer dans le modèle afin que le système d'enseignement puisse produire des explications ou des conseils adaptés aux différentes situations d'enseignement. Les systèmes supportant le travail coopératif fournissent les éléments qui permettent l'enseignement ou la simulation de tâches par une équipe d'apprenants humains.

Le modèle issu de toutes ces recherches est appelé MONACO_T. Il permet de représenter aussi bien les tâches conceptuelles que les tâches de manipulation de systèmes. Dans le contexte de ces dernières, les actions effectuées modifient l'environnement dans lequel se trouvent les agents participant à la tâche. Ce modèle utilise une approche orientée tâche plutôt qu'orientée agent comme la plupart des systèmes multi-agents. Ce choix nous permet de toujours avoir accès à la structure cognitive de la tâche même si sa réalisation est effectuée par plusieurs agents physiquement éloignés. Ce modèle contient également des informations permettant de donner des explications spécifiques en situation d'enseignement, comme la description de la méthode d'exécution d'une action dans une situation donnée.

La modélisation de MONACO_T exploite les représentations de connaissances de types couche, graphe, règle et méta-règle. Elle contient trois niveaux : statique, dynamique et scénario. La statique permet de modéliser les postes de travail, les objets, les actions et les sous-tâches que fait ressortir l'analyse cognitive de la tâche coopérative. Chaque sous-tâche contient les actions de déclenchement, de réalisation et de terminaison. Les sous-tâches sont organisées suivant une hiérarchie de composition qui permet de conserver les différents degrés d'abstraction que l'on retrouve dans une tâche. La coopération est assurée par la distribution des sous-tâches aux différents postes de travail impliqués dans la réalisation de la tâche coopérative.

La couche dynamique contient toutes les règles qui régissent le comportement de chaque action présente dans la tâche coopérative. Ainsi chaque sous-tâche contient trois bases de règles : les règles de déclenchement, de réalisation et de terminaison. Chaque base détermine les conditions qu'il faut respecter pour que l'action correspondante soit activable. Dans l'état actuel, les règles sont exprimées en logique des prédicats du premier ordre.

La couche scénario permet de décrire des stratégies particulières d'exécution de la tâche coopérative. Elle est implantée sous forme de règles de production. Chaque règle permet de modifier dynamiquement (en cours d'exécution) la structure de la tâche coopérative. Cette modification peut aller de la modification des conditions d'activation d'une règle de la dynamique à la redistribution des sous-tâches aux postes de travail en passant par un nouvel ordonnancement des actions à exécuter.

Le modèle MONACO_T ainsi construit représente une base de règles distribuée (distribution dans les sous-tâches et à travers les sous-tâches aux postes de travail), hiérarchique (les sous-tâches formant une hiérarchie, la distribution des règles dans les sous-tâches entraîne une hiérarchisation des règles) et dynamique (grâce aux règles de la couche scénario qui permettent de modifier en cours d'exécution la structure des règles).

La tâche étant construite, il faut l'exploiter pour faire de l'enseignement ou de la simulation. Pour cela nous avons mis sur pied une architecture multi-agents (EDER_TC) qui exploite les tâches à la norme MONACO_T et permet d'enseigner ou de simuler une tâche coopérative. Cette architecture est basée sur des agents génériques capables de

s'adapter à toute tâche modélisée avec MONACO_T. Pour remplir les différents rôles que nous assignons à notre système, quatre types d'agents ont été nécessaires dans EDER_TC. Il s'agit d'un agent gestionnaire de la tâche coopérative, d'un agent informatique pour la simulation des activités à un poste de travail, d'un agent humain pour la simulation et d'un agent pour le tutorat d'un poste de travail.

Les tests d'EDER_TC ont été effectués en utilisant comme exemple de tâche coopérative le processus d'enseignement individualisé de concept. Nous avons montré que notre architecture est aussi bien utile pour la simulation que pour l'enseignement. Dans le mode simulation, avec le poste de l'enseignant occupé par un agent informatique de simulation et le poste de l'apprenant utilisé par un agent humain de simulation, EDER_TC devient un STI comme tous les autres, enseignant à l'humain les caractéristiques d'un concept donné. Dans le mode enseignement, avec le poste de l'enseignant occupé par un agent tuteur et l'humain qu'il est chargé de suivre et le poste apprenant occupé par un agent humain de simulation, EDER_TC simule la formation d'un enseignant mis en situation d'enseignement où le superviseur est l'agent tuteur et l'élève est l'agent humain de simulation.

Quelle que soit la tâche coopérative modélisée avec MONACO_T, EDER_TC est capable de gérer son exécution. La gestion d'EDER_TC est interventionniste par la mise en place d'agents informatiques de simulation pour aider les humains à résoudre une tâche coopérative. Cette façon d'intervenir dans la réalisation de la tâche coopérative fait passer EDER_TC du monde des systèmes supportant le travail coopératif à celui des systèmes gérant le travail coopératif.

7.2 Perspectives

À l'issue de cette thèse, plusieurs perspectives peuvent être dégagées. Nous parlerons de l'utilisation d'un système de gestion d'objets distribués pour gérer les objets MONACO_T. Cette perspective s'inscrit dans l'amélioration du système mis en place. Nous allons également présenter les perspectives du point de vue de l'exploitation du système.

Le système EDER_TC est un système d'agents coopératifs basé sur la distribution des connaissances représentant une tâche coopérative. Pour assurer cette coopération, le système est implanté en utilisant des sockets pour la communication et en confiant la gestion de la tâche coopérative à un seul agent. Ainsi on évite les problèmes de synchronisation, de réplication et d'interblocage que l'on peut retrouver dans des systèmes réellement distribués. Or un standard de système de gestion d'objets distribués existe sur le marché (CORBA). Il pourrait être utilisé pour gérer la distribution des objets MONACO_T à travers plusieurs agents.

Le second volet des perspectives concerne les différentes utilisations possibles du système EDER_TC. En plus de l'enseignement de tâches coopératives, EDER_TC peut servir à comprendre le fonctionnement d'organisations complexes. Il peut aussi permettre la construction d'un STI complet fonctionnant sous forme de client-serveur. Ce STI peut lui-même être utilisé pour mettre sur un pied un kiosque d'enseignement sur INTERNET. EDER_TC peut enfin être utilisé pour construire des logiciels à tutoriel intégré.

Les organisations humaines sont par définitions des systèmes coopératifs, où les hommes interagissent à travers des ressources dispersées dans l'espace et dans le temps pour résoudre des problèmes. Les grandes organisations comme les ministères, les banques, les assurances sont des systèmes où plusieurs rôles coopèrent. Dans celles-ci il est difficile de trouver quelqu'un qui puisse expliquer le comportement global de l'organisation ainsi que les interactions qui existent entre chacun des rôles. La modélisation des activités de telles organisations avec MONACO_T permettra de définir les postes de travail qui figureront les différents rôles et l'arborescence des sous-tâches qui représentera les activités effectuées dans l'organisme. L'utilisation d'EDER_TC avec l'objet MONACO_T résultant permettra de suivre l'évolution des activités coopératives dans l'organisation et d'expliquer le cas échéant comment cette dernière fonctionne. Le modèle que nous proposons peut ainsi être un outil pour la compréhension et l'explication du comportement d'organisations complexes.

L'enseignement consiste en l'acquisition d'un ensemble de capacités. Pour chaque type de capacités, il existe une procédure permettant son acquisition [Gagné, 79] en fonction des objectifs recherchés. Si on construit chacune de ces procédures sous forme d'objet MONACO_T, le curriculum peut ainsi se présenter par une combinaison de ressources et de ces objets. L'exploitation de ce curriculum par EDER_TC peut se faire en utilisant le WEB comme support de communication. En faisant exécuter le poste apprenant comme une extension aux navigateurs et les autres agents comme serveurs d'enseignement, on construit ainsi un kiosque d'enseignement qui peut être utilisé comme serveur d'enseignement sur INTERNET.

Les logiciels sont conçus pour permettre l'exécution de tâches précises. Ainsi un logiciel de traitement de texte permet de créer et de formater des textes, un simulateur de vol permet d'apprendre à piloter un avion. Ces logiciels implantent les fonctions élémentaires utilisées pour exécuter les tâches, mais laissent leur description dans les guides utilisateur. Un livre sur le traitement de texte Word vous décrira les principales tâches que l'on peut effectuer avec le logiciel. Il en va de même des tutoriels de tous les logiciels. Notre proposition consiste à intégrer ces tâches dans le logiciel sous forme d'objets MONACO_T. Ainsi quand un usager ignore comment réaliser une tâche, au lieu de lire le manuel ou utiliser le tutoriel, il peut provoquer l'utilisation de la tâche MONACO_T correspondante par un agent de simulation (dans ce cas il observe comment le système résout la tâche) ou par un agent tuteur (il se fait guider pas à pas dans la réalisation de la tâche). On obtient ainsi un logiciel à tutoriel intégré.

8 Bibliographie

[Adams & al., 90] Adams Dennis, Carlson Helen, Hamm Mary. (1990). *Cooperative Learning and Educational Media: Collaborative with technology and each other.* Educational Technology Publication Englewood Cliffs, New Jersey 07632.

[Allessi 85] S.M. Allessi & S.R. Trollip. Computer-based Instruction : methods and development. Prentice-Hall, Englewood Cliffs, NJ, 1985.

[Anderson 83a] Anderson, J.R. (1983). *The Architecture of Cognition*, Cambridge: Harvard University Press.

[Anderson 83b] J.R. Anderson. Acquisition of proof skills in geometry. In J.G. Carbonell, E. Mickalski & T. Mitchell(dir.), Machine Learning : An Artificial Intelligence Approach. Tioga, Palo Altos, CA, 1983

[Anderson 88] J.R. Anderson. The expert module. In Polson & Richardson pages 21-53

[Ausubel 68] Ausubel, D.P. Educational psychology : A cognitive view. New York : Holt, Rinehart and Winston, 1968.

[Bestougeff 82] Bestougeff, H., Fargette, J-P. (1982). Enseignement et ordinateur. Presse universitaires de France - Paris.

[Bloom 59/69] Bloom, B.S. Taxonomie des objectifs pédagogiques (M. Lavallée, trad.). Montréal : Education Nouvelle, 1969 (Publ. Orig., 1956).

[Bond & Gasser 88] Bond A.H. and Gasser L. Reading in Distributrd Artificial Intelligence. Morgan Kaufmann, Palo Alto, CA, 1988

[Bovair & al., 90] Bovair, S., Kieras, D.E., & Polson, P.G. The acquisition and performance of text-editing skill: A cognitive complexity analysis. Human Computer Interaction, (1990) 5, 1-48.

[Brézillon 92] Brézillon, P.(1992). *Explication et coopération:Une contribution bibliographique au travail du groupe COOP*. Rapport No 92/31

[Brien, 92] Robert Brien (1992). Design Pédagogique :Introduction à l'approche de Gagné et de Briggs, Les Editions Saint-Yves, Inc.

[Brown & al., 75] J.S. Brown, R.R. Burton & A.G. Bell. SOPHIE : a step toward a reactive learning environment. Man-machine studies, vol. 7, pages 675-696, 1975

[Brown & al., 82] Brown, J.S., Burton, R.R., & DeKleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, and III. In D. Sleeman & J.S. Brown (Eds.), *Intelligent Tutoring Systems* (pp. 227-282). New York: Academic Press

[Burton 82] Burton, R. R. (1982) *Diagnosing bugs in a simple procedural skill.* Dans G Sleeman et J. S. Brown(Eds), Intelligent Tutoring System. pp. 157-183. NY:Academical Press

[Cammarata 83] Cammarata S., McArthur D. and Steeb R., Strategies of Cooperation in Distributed Problem Solving, In Proceedings of the 1983 International Joint Conference on Artificial Intelligence, pages 767-770, 1983

[Carbonell 70] Carbonell, J.R. (1970) AI in CAI: an artificial intelligence approach to computer-assisted instruction. IEEE *Transactions on Man-Machine Systems*, vol. 11, no.4, pp. 190-202.

[Chan & al., 92] Chan, T.W., Chung, I-L., Ho, R-G., Hou, W-J., Lin, G-L. (1992) Distributed Learning Companion System: West Revisited. In *The Proceeding of Second international Conference of Intelligent Tutoring Systems*, , June, Montreal, Canada, pp. 643-650.

[Chan & Baskin, 88] Chan, T.W. et Baskin, A.B. (1988). Learning Companion Systems. *In C. Frasson et G. Gauthier (Eds.) Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*, Chapter 1, New Jersey: Ablex Publishing Corporation.

[Chan 91] Chan, T.W. (1991). Integration-Kid: A Learning Companion System. *The Proceeding of the 12th International Joint Conference on Artificial Intelligence*, Sydney, Australia, Morgan Kaufmann Publishers, Inc., pp. 1094-1099.

[Chandrasekaran, 87] Chandrasekaran, B. Towards a functional architecture for intelligence based on generic information processing tasks. Proceedings of the Tenth International

Joint Conference on Artificial Intelligence, (1987) pages 1183-1192, Milan, Italy .

[Clancey 87] Clancey, W.J. (1987) Knowledge-based Tutoring: The GUIDON Program. MIT Press, Cambridge, Massachusetts.

[Clancey 92] Clancey, W.J. (1992). Guidon-Manage Revised: A socio-Technical System Approach. *Invited Talk, The 2nd International Conference of Intelligent Tutoring Systems, Lecture Notes in computer science*, 608, Springer-Verlag, pp. 21-36.

[Corkill 79] Corkill D.D. Hierarchical Planning in Distributed Environment, In Proceeding of the 1979 International Joint Conference on Artificial Intelligence, pages 168-175, 1979

[Courouble 83] Courouble, T. (1983). *Le rôle des pédagogues dans le développement de l'EAO en formation professionnelle.* Education permanente numéro 70/71. Ouvrage édité sous la direction de Jobert, G. et Perriault, J.

[Cullingford 84] Cullingford R.E. and Pazzani M.J. Word-Meaning Selection in Multiprocess Language Understanding Programs, IEEE Transaction on Pattern Analysis and Machine Intelligence, Pami-6(4) :493-509, july 1984

[Derycke & Hoogstoel 95]: "Le travail coopératif assisté par ordinateur: quels enjeux pour les concepteurs ?", Alain DERYCKE, Frédéric HOOGSTOEL, Actes des journées INFORSID'95.

[Dillenbourg & Self, 91] Dillenbourg, P., Self, J. (1991) Designing Human-Computer Collaboration Learning. In O'Malley, C.(Ed), Human-Computer Collaborative Learning, Springer-Verlag.

[Dillenbourg & Self, 92] Dillenbourg, P & Self, J. (1992). People Power: A Human-Computer Collaborative Learning System. *The 2nd International Conference of Intelligent Tutoring Systems, Lecture Notes in Computer Science*, 608, Springer-Verlag, pp. 651-660.

[Djamen, 95] Djamen J.Y. (1995). Une architecture de STI pour l'analyse du raisonnement d'un apprenant. Thèse de Ph.D., DIRO, Université de Montréal, Montréal, CANADA, Juin 1995

[Dourish 93] Dourish, P. (1993). Culture and Control in a Media Space. *In Proceeding of the third European Computer Supported Cooperative Work,* pp. 125-138.

[Dykstra & Carasik, 91] Dykstra, E.A. et Carasik, R.P.(1991). Structure and support in cooperative environments: The Amsterdam Conversation Environment. IJMMS 34, pp. 419-434.

[Ellis & al., 91] Ellis, C., Gibbs, S.J., et Rein, G.L. (1991). Groupware: Some issues and experiences. *CACM*, 34(1), pp. 38-58.

[Ellis & Keddara, 93] Ellis, C.A. et Keddara, K. (1993). Dynamic Change Withing Workflow Systems. *Technical report CU-CS-667-93 Department of Computer Science*, University of Colorado.

[Ellis & Wainer, 94] Ellis C. et Wainer J. (1994) A Conceptual Model of Groupware. *In Proceeding of ACM 1994 Computer Supported Cooperative Work.* pp. 79-88.

[Elsom-Cook & al., 88] M. Elsom-Cook, F. Spensley, P. Byerley, P. Brooks, M. Mhende, M. Federici & C. Scaroni. Using multiple teaching strategie in an ITS. In C. Frasson & G. Gauthier (dir.), Proceeding of the first international conference on intelligent tutoring systems, pages 286-290, Montreal, QC, 1988

[Ephrati 92] Ephrati E. and Rosenschein J.J. Constrained Intelligent Action : Planning Under the Influence of a Master Agent, Proc. Tenth National Conference on AI, July, 1992

[Erman 80] Erman L.D., Hayes-Roth F.A., Lesser V.R. and Raj Reddy D. The Hearsay-II Speech-Understanding System : Integrating Knowledge to resolve Uncertainty, Computing Surveys, 12(«2) :213-253, June 1980

[Ferbert 94] Ferbert, J. (1994). Des systèmes multi-agent pour simuler le vivant. *In Actes du 2e colloques africain sur la recherche en informatique 12-18.* Edition ORSTOM.

[Findler 86] Findler N.V. and Ron Lo. An Examination of Distributed Planning in the World of Traffic Control, Journal ofb Parallel and Distributed Computing, 3 :411-431, 1986.

[Frasson & al.,96] Frasson, C., Mengelle, T., Aïmeur, E., Gouaders, G. (1996). An Actor-

based Architecture for Intelligent Tutoring Systems. 3ème conférence internationale sur les systèmes tutoriels intelligents Montréal, Canada

[Frasson 91] Frasson, C. "Systèmes tutoriels intelligents : états et perspectives en Amérique du Nord", Revue Génie Éducatif, vol 1, no 1, pp 7-15, march 1991.

[Frasson 97] Frasson, C., Aïmeur, E. "Lessons Learned from a University-Industry Cooperative Project in Tutoring Systems", Int. Journal of Failures & Lessons Learned in Information Technology Management, J. Liebowitz Editor-in-Chief, 1(2), June 1997.

[Gagné 72] Gagné, R.M. Domains of learning. Interchange, 1972, 3, 1-8.

[Gagné 79] Gagné, R.M. et Briggs, L.J. Principles of instructional design. New York : Holt, Rinehart and Winston, 1979

[Gasser 89] Gasser Les, Rouquette N.F., Hill R.W. and Lieb Jon. Representing and Using Organizational Knowledge in DAI Systems, In L. Gasser and M.N. Huhms, editors Distributed Artificial Intelligence, Volume II, page 55-78. PitmanéMorgan Kaufmann, London 1989.

[Gasser 92] Gasser L., An Overview of DAI, In N.M. Avouris and L. Gasser (eds.) Distributed Artificial Intelligence : Theory and Praxix, 1-7. 1992 ECSC, EEC, EAEC, Brussels and Luxembourg. Printed in the Netherlands

[Gilmore & Self, 88] Gilmore, D., Self, J. (1988) The application of machine learning to intelligent tutoring systems. In J. Self, (Ed) Artificial Intelligence and Human Learning, Intelligent computer-assisted instruction, pp. 179-196, New York: Chapman and Hall.

[Goldstein 82] Goldstein, I. P. (1982) *The Genetic Graph : A Representation for the Evolution of Procedural Knowledge.* Dans G Sleeman et J. S. Brown(Eds), *Intelligent Tutoring System* . pp. 51-77. NY:Academical Press .

[Greenberg & al., 92] Greenberg, S., Roseman, N., Webster, D., et Bohnet, R.(1992). Issues and Experiences Designing and Implementing Two Group Drawing Tools. *Proceeding of HICSS '92 IEEE Computer Society, Los Alamitos, CA, pp. 139-150.*

[Greenberg 91] Greenberg,S. Personalizable groupware: Accomodating individual roles and group differences. *In Proceeding of the 2nd European Computer Supported Cooperative Work,* (EC-CSCW '91).

[Grudin 94] Grudin, J. (1994). Computer-Supported coopérative Work: History and Focus. *In Computer.*

[Halff 88] H. Halff. Curriculum and instruction in ITS. In Foundation of intelligent tutoring systems, pages 19-108. Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[Huhns 87] Huhns M.N. Distributed Artificial Intelligence. Pitman Publishing Morgan Kaufman, 1987

[Imbeau 90] Imbeau, G. (1990) Modélisation et réalisation d'un système tutoriel intelligent. Thèse de Ph.D., Département d'Informatique et de Recherche Opérationnelle, Université de Montréal.

[ITU-T 93] ITU-T. Message Sequence Chart(MSC). Recommendation Z.120, 1993

[Johnson-Lenz & Johnson-Lenz, 91] Johnson-Lenz, P.et Johnson-Lenz, T. (1991). Post-mechanistic groupware primitives: rhythms, boundaries and containers. *IJMMS 34, 3(Mar. 1991),* pp. 385-418.

[Kabbaj 93] Kabbaj, A., Frasson, C. (1993) Représentation et acquisition des connaissances dans les systèmes tutoriels intelligents. Dans actes du congrès Acquisition de Connaissances dans les Tuteurs Intelligents ACTI93. Patronage: AFCET, AFIA.

[Kearsley 87] Kearsley Greg P., Artificial Intelligence & Instruction : Application and Methods. Addison-Wesley Publishing Copany, 1987

[Kerr 94] M.P. Kerr & S.J. Payne. Learning to use a spreadsheet by doing and watching. Interacting with computers, vol. 6, no. 1, 1994

[Lajoie & al., 89] S. Lajoie, A. Lesgold & al. A procedural guide to the avionics troubleshooting tutor development process. Rapport recnique, Learning research and development center, University of Pittsburgh, Pittsburgh, 1989.

[Lefevre 84] Lefevre, J-M. (1984). *Guide pratique de l'EAO.* Edition Cedic Nathan.

[Legendre 93] Legendre R. Dictionnaire actuel de l'éducation, 2nd Édition, Guérin éditeur ltée, 1993

[Lesgold & al., 92] Lesgold, A., Lajoie, S., Bunzo, M., et Eggan, G. A coached practice environment for an electronics troubleshooting job. In Larkin, J. et Chabay, R., Editeur, *Computer-Assisted Instruction and Intelligent Tutoring System: Shared Goals and Complementary Approaches*, (1992) pages 201-238. Lawrence Erbaum Associates.

[Lindsay & Norman 72] Lindsay, P.H. et Norman, D.A. Human information processing : An introduction to psychology. New York : Academic Press, 1972.

[Lotus 93] Lotus Corporation (1993). *Lotus Notes: Application Developer's Reference(Release 3)*.

[Mark & Miller, 94] Mark A., Miller P.(1994). *Analysing Broadband Networks*, M&T Books, N.Y.

[Murray & Woolf, 93] Murray, T. et Woolf, B.(1993) Design and implementation of an intelligent multimedia tutor AAAI-93 tutorial.

[Nkambou & al., 96] Nkambou, R., Lefebvre, B., Gauthier, G. (1996). A Curriculum-Based Student Modelling for ITS. *In Proc. of the Fifth International Conf. on User Modelling*. PP. 89-97. Kailua-Kona, Hawaii.

[Nkambou & Gauthier, 96] Nkambou, R., Gauthier, G., (1996) Un modèle de représentation de connaissances dans les STI. In Proceeding of 3th International Conference on Intelligent Tutoring System ITS96. Springer-Verlag.

[Norman & Rumelhart 75] Norman, D.A. et Rumelhart, D.E. Explorations in cognition. San francisco : W.H. Freeman and Company, 1975.

[Nunamaker & al., 91] Nunamaker, J.F., Dennis, A.R., Valacich, J.S., Vogel, D.R., and George, J.F.(1991). Electronic meeting systems to support group work. *CACM* 34, pp. 40-61.

[Okada & al., 94] Okada K., Maeda F., Ichikawaa Y., Matsushita Y.(1994). Multiparty Videoconferencing at Virtual Social Distance: *MAJIC Design. Proc. of the ACM*

1994 Conference on CSCW, pp. 385-393.

[Péninou, 93] Péninou, A., (1993). MACT: un modèle d'agents centrés tâches pour la production de systèmes tuteurs intelligents par l'atelier de génie didacticiel intégré. Thèse de doctorat de l'université paul Sabatier Toulouse. France.

[Polson & Richardson, 88] Polson Martha C., Richardson J. Jeffrey. (1988). *Foundation of Intelligent Tutoring Systems*. Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey.

[Reiser & al., 85] Reiser, B.J., Anderson, J.R., and Farrell, G.G. (1985) Dynamic student modeling in an intelligent tutor for LISP programming. Proceeding of the Ninth International Joint Conference on Artificial Intelligence Conference, Los Angeles, pp. 8-14.

[Roseman & Greenberg, 94] Roseman, M., et Greenberg S. (1994). GroupKit: A Groupware Toolkit for Building Real-Time Conferencing Applications. *In Proceeding of ACM 1992 Computer Supported Cooperative Work,* pp. 43-50.

[Schank 84] Schank Roger C. with Childers Peter. The Cognitive Computer on Language, Learning, and Artificial Intelligence. Addison-Wesley Publishing Company, Inc. 1984

[Schmidt & Bannon, 92] Schmidt, K. et Bannon, L. (1992). Taking CSCW Seriously. *In Computer Supported Cooperative Work Journal, 1,1,* pp. 7-40.

[Schoenfeld 85] Schoenfeld, A.H. (1985). *Mathematical problem solving*. New York: Academic Press.

[Schreiber & al., 93] Schreiber, A.T., Wielinga, B.J., Breuker, J.A., - KADS : A Principled approach to knowledge-Based System Development, Acamic Press Harcourt Brace Jovanovich, Publishers (1993)

[Shen & Dewan, 92] Shen, H.,et Dewan P. (1992). Access Control for Collaborative Environments. *In Proceeding of ACM 1992 Computer Supported Cooperative Work,* pp. 43-50.

[Skinner 76] Skinner, B. F. (1976). *About behaviorism*, Vintage books, New York.

[Slavin 83] Slavin, R. (1983). *Cooperative Learning*. NEW YORK: Longman.

[Slavin 88] Slavin, R. (1988). Cooperative Learning and Student Achievement. *Educational Leadership*, 54, pp. 31-33.

[Slavin 89] Slavin, R. (1989). *School and classroom organisation.* Hillsdale, NJ: Erlbaum

[Somé 97] Somé S.S., Dérivation de spécification à partir de scénarios d'interaction, Thèse de Ph.D, DIRO, FAS, Université de Montréal, Montréal, 1997

[Tadié & Aimeur, 96] Tadié, G.S., Aïmeur, E., (1996) Épistémolagie de la coopération. Publication no. 1023 Département d'Informatique et de Recherche Opérationnelle. Université de Montréal.

[Tadie & al., 96b] Tadie, S., Frasson , C. et Lefebvre, B. (1996b). Vers une acquisition coopérative des connaissances acquises par l'expérience. Dans proceedings de la 3e conférence africaine sur la recherche en informatique CARI'96. Libreville - GABON.

[Tadie & al., 96b] Tadie, S., Lefebvre, B. et Frasson, C. (1996). MONACO_Tâche: un modèle à nature coopérative pour la modélisation de tâche dans un STI. In *The Proceeding of Thirth international Conference of Intelligent Tutoring Systems*, June, Montreal, Canada.

[Tadie & al., 96c] Tadie, S., Frasson , C. et Lefebvre, B. (1996a). Coopération entre agents conseillers pour la génération d'explications dans un environnement coopératif. *actes des Journées Explication96 Sophia-Antipolis, France* .

[Tadié & al., 97a]Tadié G.S., Lefebvre B., Frasson C., Teaching Cooperative task Using the Web. Dans les proceeding de la conférence Webnet 97. Toronto du 31-Oct. Au 5 Nov. 1997.

[Tadie & al., 97b] Tadie, G.S., Rossignol, J-Y, Frasson, C. et Lefebvre, B. (1997). Collaborative Learning System in Industry. In *Demonstration at the Second Annual Conference on the TeleLearning NCE*, November 4-6, 1997, Toronto, Canada.

[Tadie & al., 98a] Tadie, S., Rossignol, J.Y., Frasson, C. et Lefebvre, B. (1998). An Interactive Graphical Tool for Efficient Cooperative Task Acquisition Based on Monaco-T Model. Will appear In *The Proceeding of fourth international Conference*

of Intelligent Tutoring Systems, August, San-Antonio, USA.

[Tadie & al., 98b] Tadie, S., Frasson, C. et Lefebvre, B. (1998). Un modèle de représentation de tâche coopérative permettant la génération d'explications dans les environnements distribués d'enseignement :Monaco-T. Soumis à la conférence NTICF'98, Rouen, France.

[Tadié, 96] Tadié, G.S., (1996) Coopération et Système Tutoriel Intelligent. Rapport de l'examen prédoctoral oral. Département d'Informatique et de Recherche Opérationnelle. Université de Montréal.

[Terveen 95] Terveen, L.G.(1995). Overview of human-computer collaboration. *In Knowledge-Based Systems volume 8 Numbers 2-3*.

[VanLehn 88] K. VanLeehn. Student modeling. In Polson & Richarson, pages 55-77

[Vygotsky 78] Vygotsky, L. (1978). *Mind in society*. Translated by M. Cole, V. John-Steiner, S. Scriber, & E. Soubermam, Cambridge, MA: Harvard University Press.

[Wenger 87] E. Wenger. Artificial intelligence and tutoring systems. Morgan Kaufmann Publishers, Inc, 1987

[Woolf 89] Woolf, B.(1989) *Representing, Acquiring, and Reasoning about Tutoring Knowledge*. Dans Proceedings of 2nd Intelligent Tutoring Systems Research Forum, San Antonio, Texas.

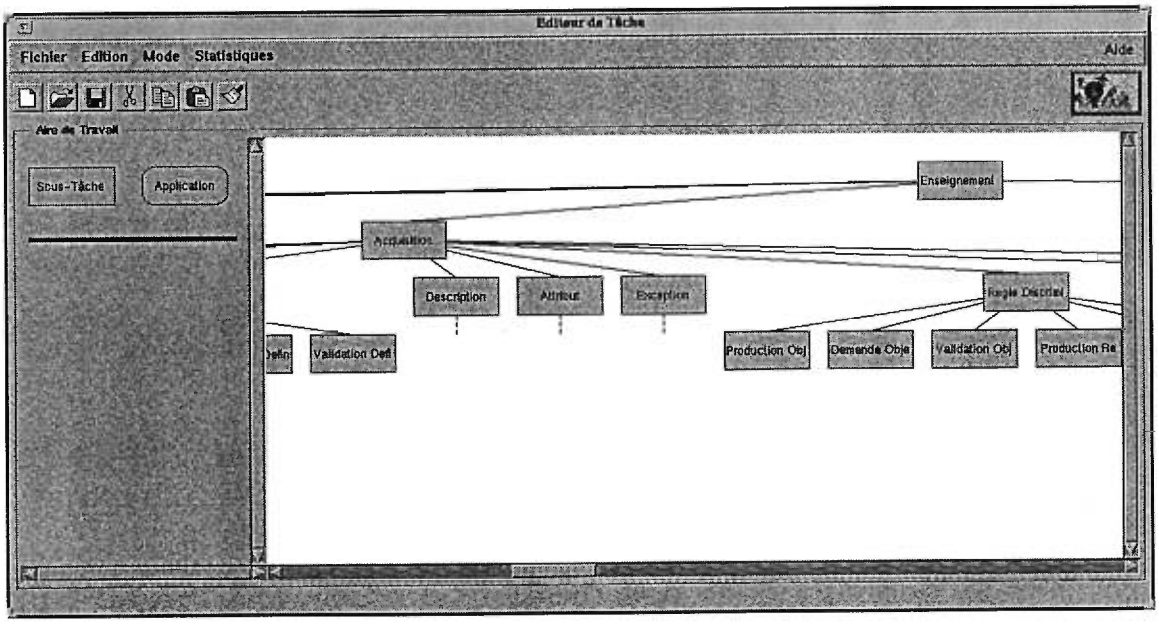
9 Annexe : Quelques interfaces

Les interfaces que nous présentons se répartissent en deux blocs : un pour l'édition de la tâche coopérative suivant la norme Monaco-T et l'autre pour la phase d'exploitation avec l'exemple de l'enseignement de concept où on a un agent apprenant et un agent enseignant.

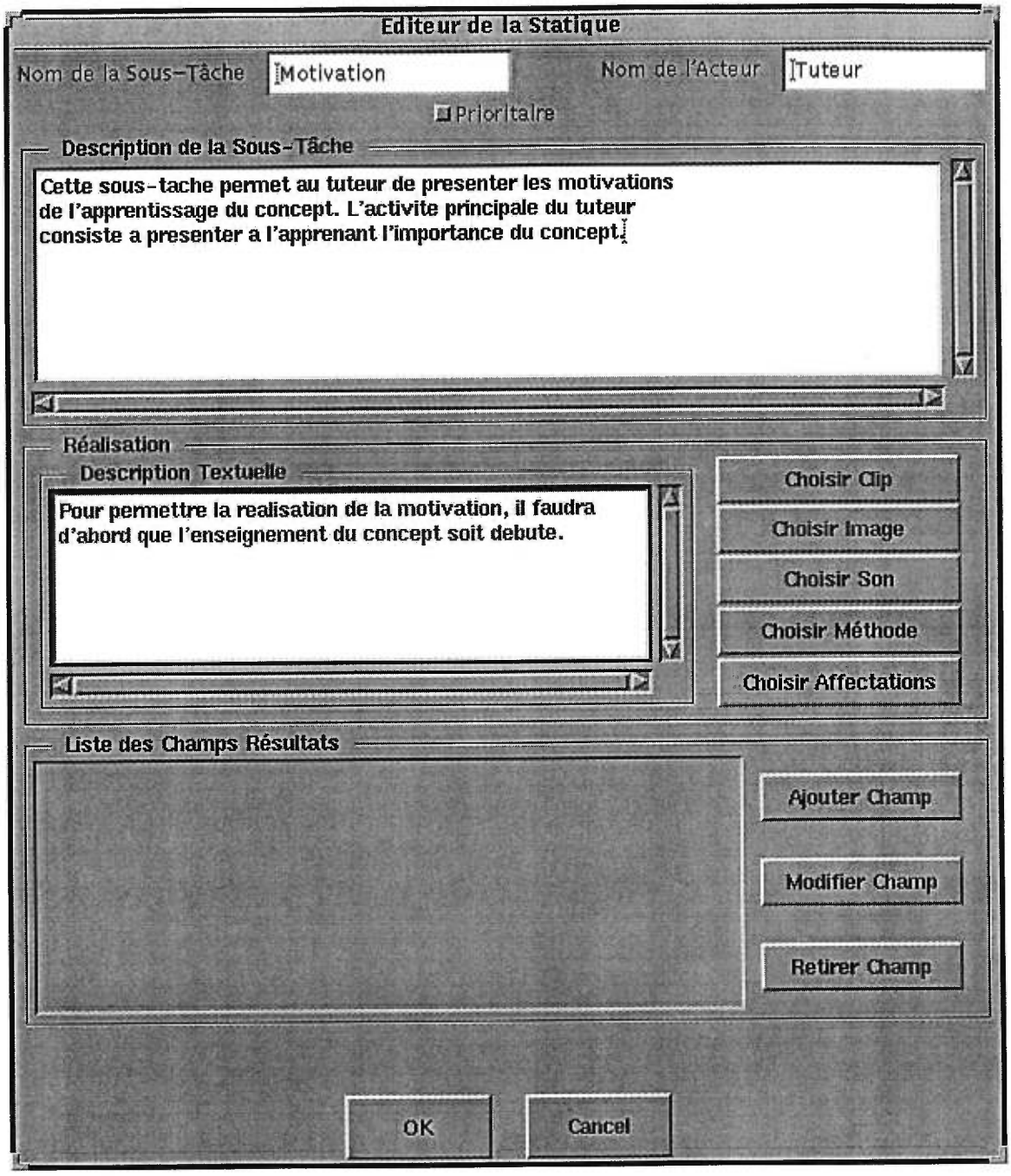
9.1 Interface pour l'édition de la tâche coopérative

Dans cette section on dispose d'écrans relatifs à la construction de chaque couche du modèle Monaco-T.

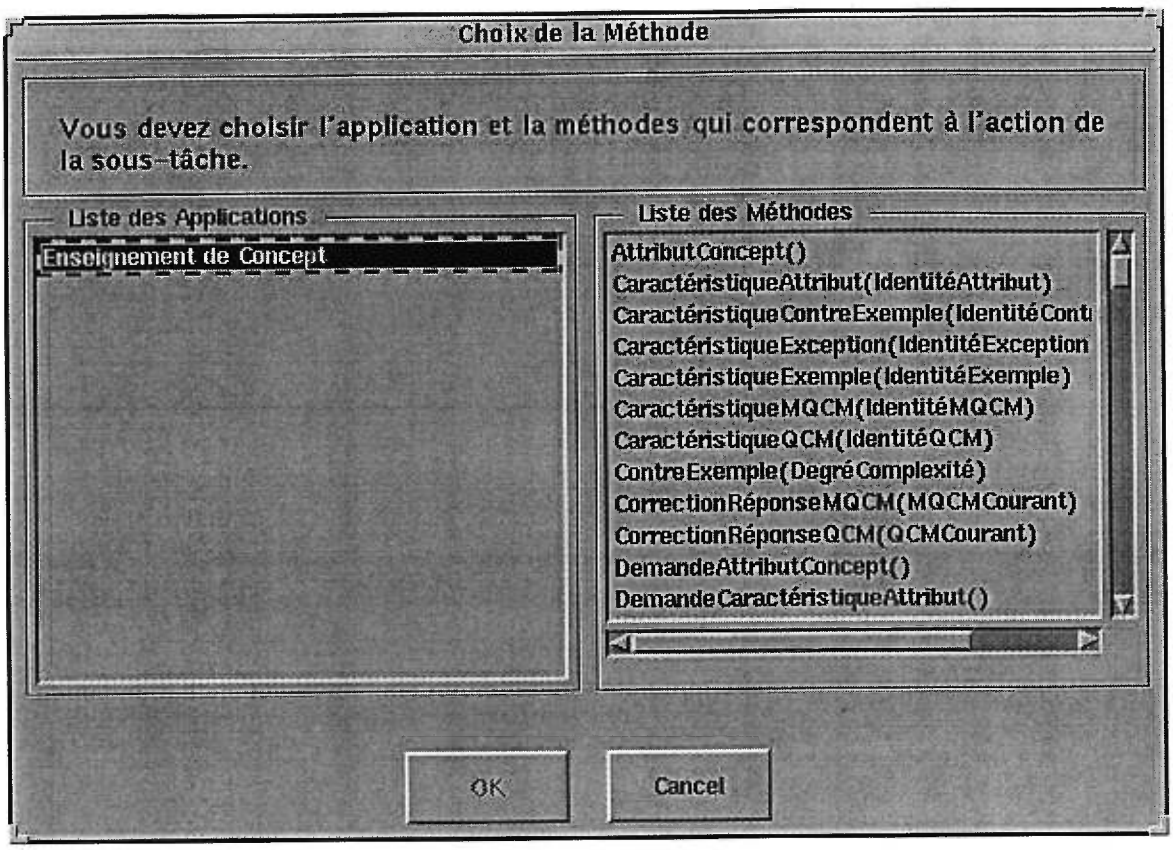
9.1.1 Interface pour l'édition de la couche statique



Interface principale d'édition de la tâche suivant le modèle Monaco-T. Elle permet de construire l'arborescence des sous-tâches, et donne accès aux outils de construction des autres informations des couches statiques, dynamique et scénario.

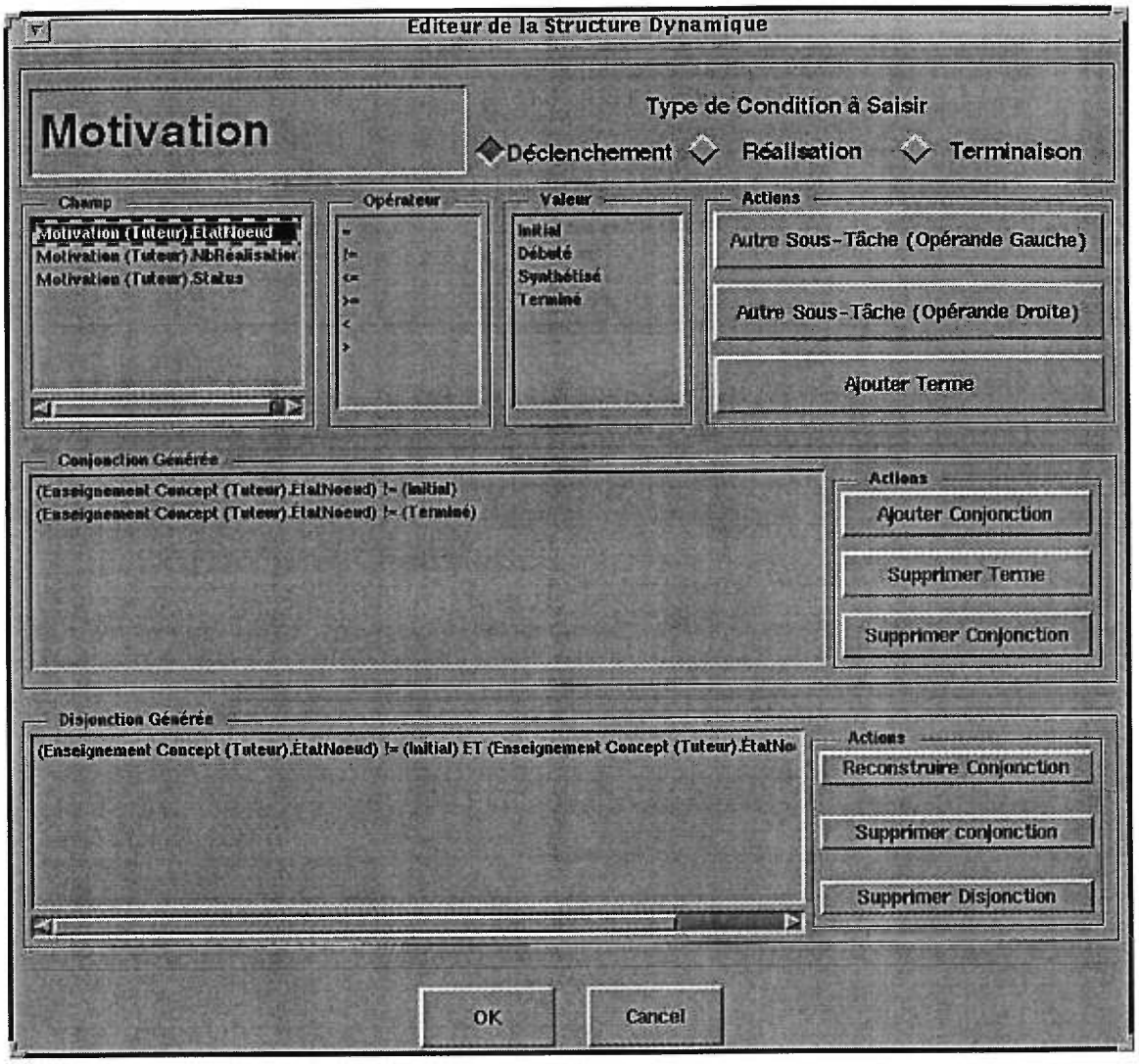


Cette interface permet de définir les champs caractéristiques d'une sous-tâche.



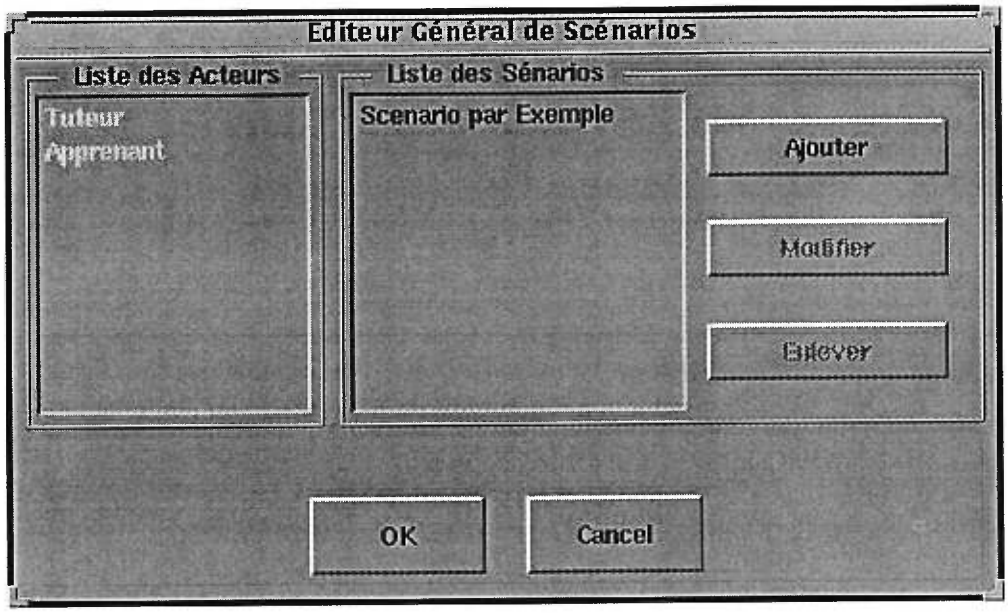
Cette interface permet de rattacher une sous-tâche à une fonction effective provenant de l'environnement de la tâche coopérative. Par exemple pour la tâche d'enseignement de concept, chaque concept possède la fonction AttributConcept, qui permet de générer les attributs caractéristiques du concept que l'on est en train d'enseigner. La Sous-tâche "présentation attribut caractéristique du concept" doit donc être effectivement liée à la fonction AttributConcept pour qu'en phase d'exploitation un agent puisse effectivement déclencher la présentation des attributs caractéristiques du concept.

9.1.2 Interface pour l'édition de la couche dynamique

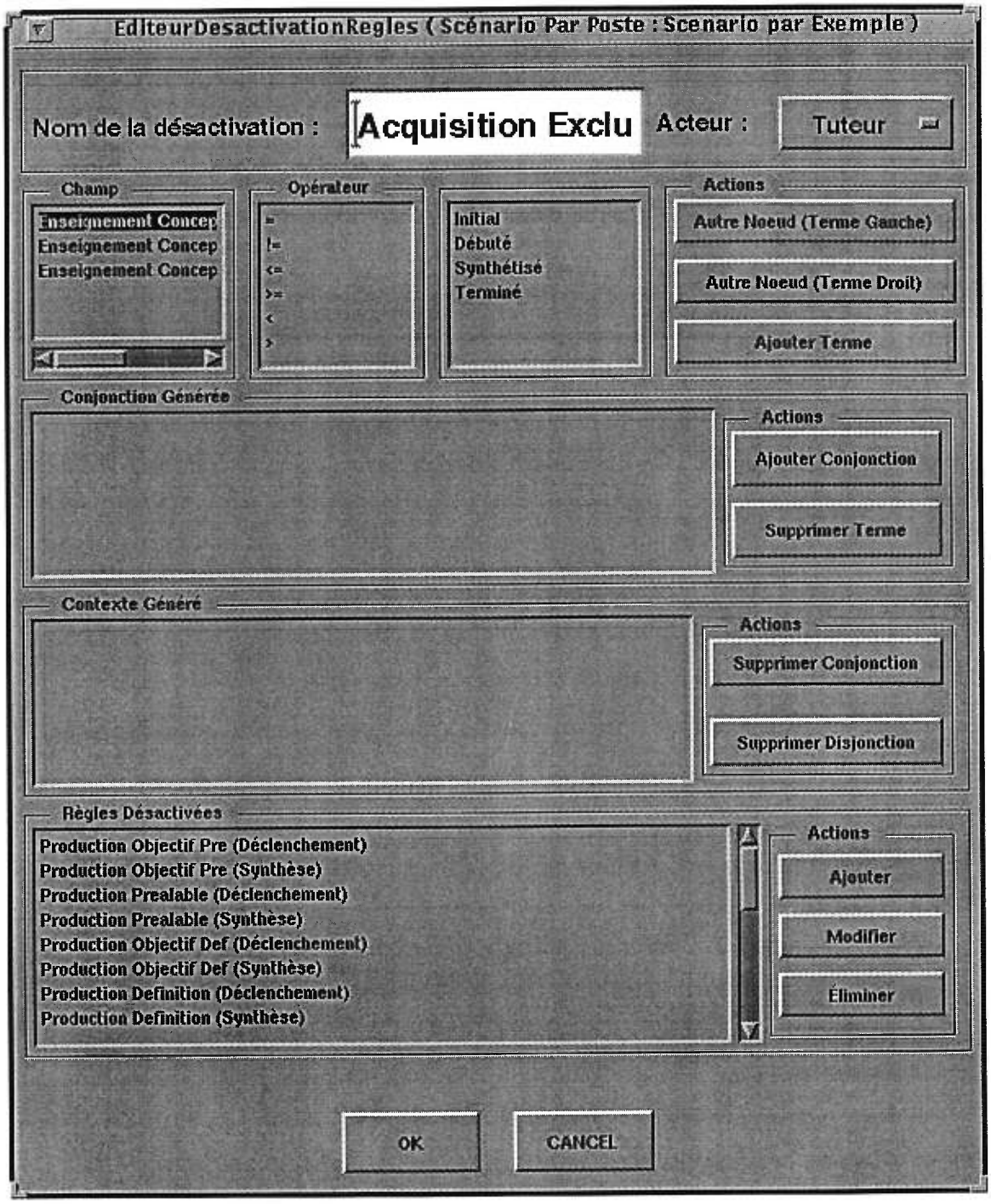


Cette interface permet de construire de manière interactive les règles qui définissent la dynamique d'une sous-tâche.

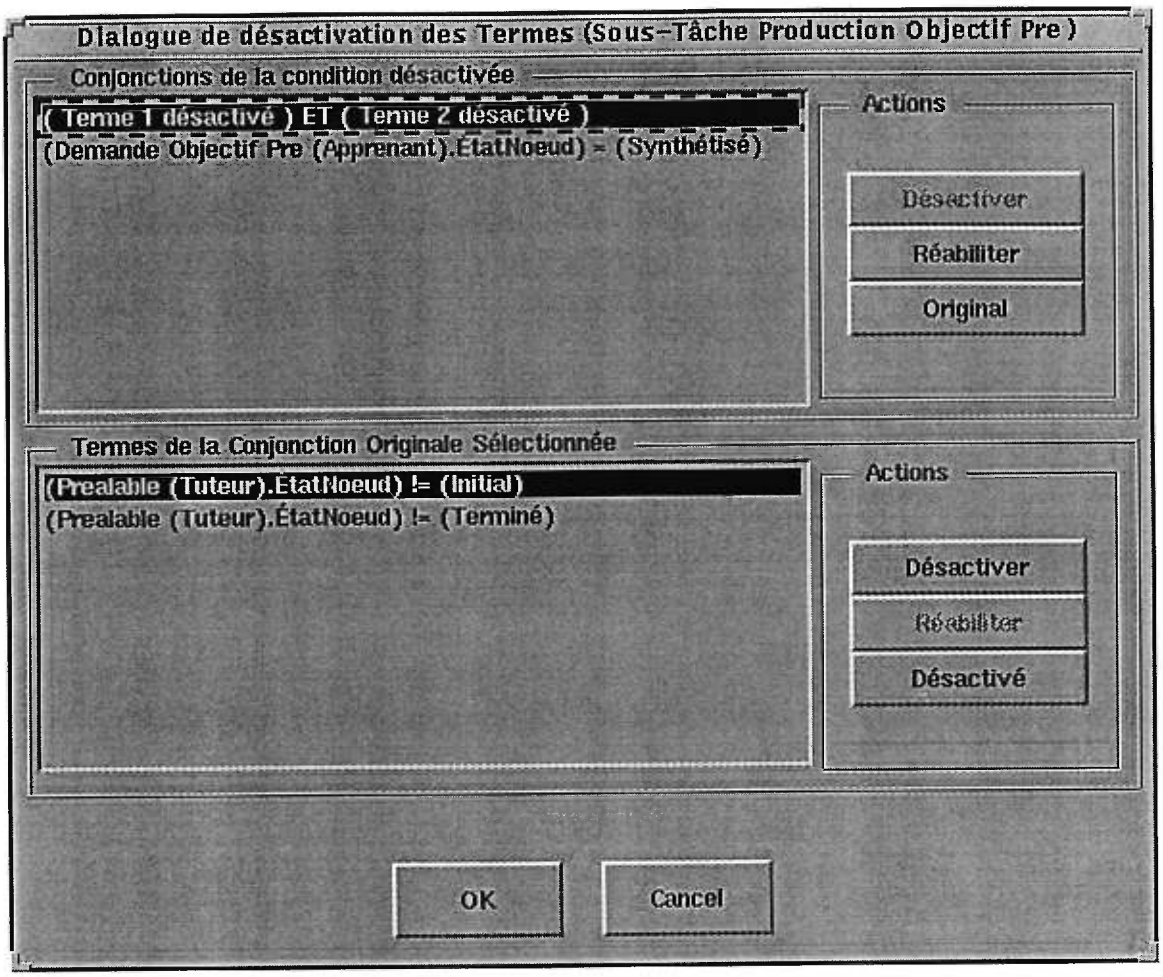
9.1.3 Interface pour l'édition de la couche scénario



Cette interface permet de créer pour chaque poste de travail de la tâche coopérative, un ensemble de scénarios d'exécution.



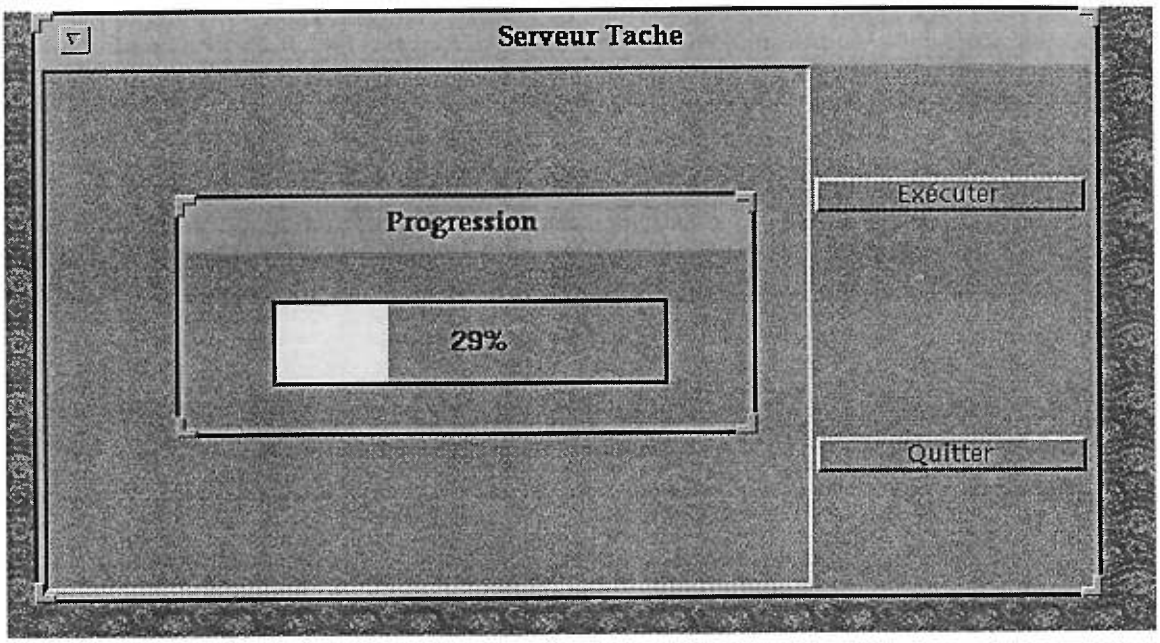
Pour un scénario donné, cette interface permet de rendre inactives certaines règles qui régissaient la dynamique des sous-tâches appartenant au poste de travail concerné.



Pour un scénario donné, cette interface permet de rendre inactifs certains termes qui participaient à définition des règles de la dynamique d'une sous-tâche donnée.

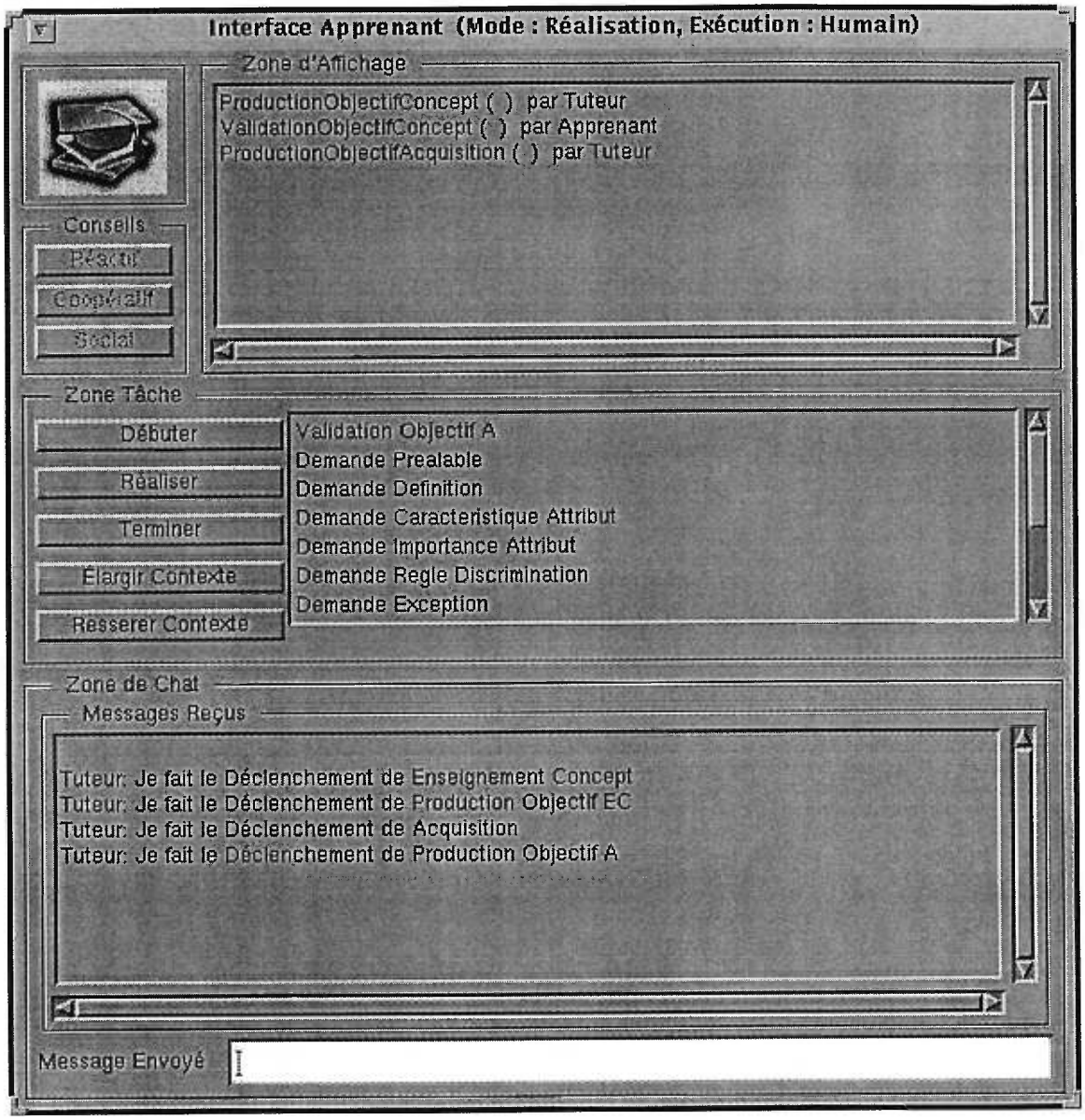
9.2 Interface pour la phase d'exploitation : Cas de l'enseignement de concept avec les agents enseignant et apprenant

9.2.1 Chargement de la tâche coopérative par l'agent gestionnaire de tâche coopérative



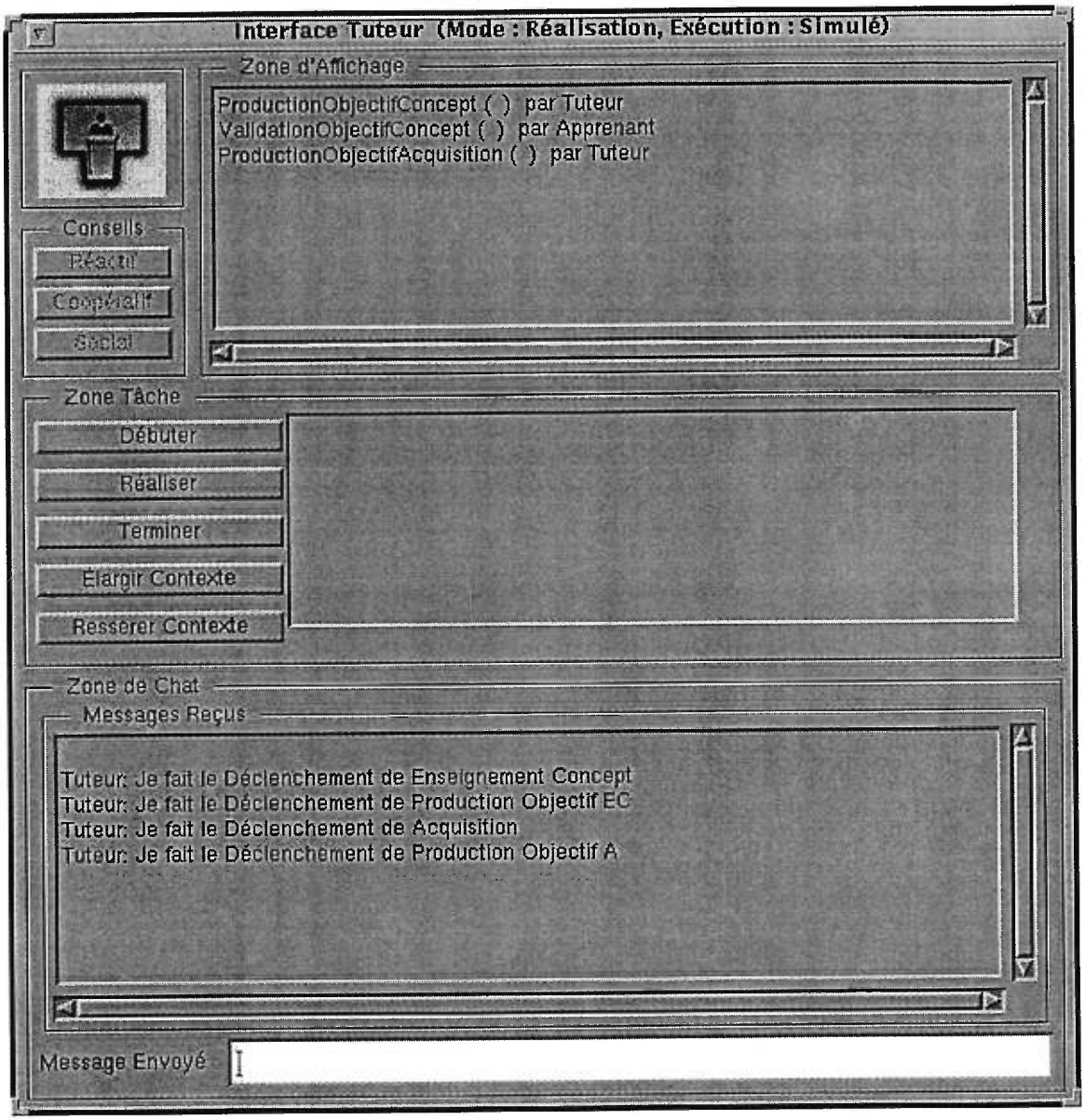
Cette interface nous montre l'agent gestionnaire de tâche coopérative en train de charger du disque une tâche coopérative à la norme Monaco-T.

9.2.2 Interface de l'agent apprenant



Cette interface est celle qui est vue et manipuler par un humain qui participe, à la tâche coopérative d'enseignement de concept au poste de l'apprenant.

9.2.3 Interface de l'agent enseignant



Cette interface est celle qui est vue et manipuler par un humain qui participe, à la tâche coopérative d'enseignement de concept au poste d'enseignant.

10 Annexe : Exemple de décomposition d'une tâche

La tâche que nous allons décrire est celle qui représente le processus d'enseignement d'un concept par un enseignant à un apprenant. Dans ce cas les deux agents présents sont l'enseignant et l'apprenant. Dans la mesure où enseigner un concept demande la manipulation d'un objet contenant les informations spécifiques à un concept, nous allons donner la liste des primitives que cet objet est capable de comprendre afin que notre système puisse être utilisé pour n'importe quel concept. Après cela nous présenterons les sous-tâches résultants de la décomposition de la tâche d'enseignement de concept.

10.1 Protocole du système manipulé par la tâche

Nom	Description	Paramètres
1. NomConcept	Identité du concept en cours	
2. DemandeNomConcept		
3. ValidationNomConcept		
4. ImportanceConcept	Description textuelle ou multi-médiatique de l'utilité de ce concept	
5. DemandeImportanceConcept		
6. ValidationImportanceConcept		
7. PréalableConcept	Permet de préciser les concepts qu'il faut maîtriser avant d'aborder le concept courant	
8. DemandePréalableConcept		
9. ValidationPréalableConcept		
10. DescriptionConcept	Donne une description du concept	
11. DemandeDescriptionConcept		
12. ValidationDescriptionConcept		
13. DéfinitionConcept	Donne la définition du concept tel que le pédagogue aimerait le présenter aux apprenants	
14. DemandeDéfinitionConcept		
15. ValidationDéfinitionConcept		
16. AttributConcept	Donne les différents caractères qui rentrent dans la spécification du concept	
17. DemandeAttributConcept		

18. ValidationAttributConcept		
19. DescriptionAttributConcept	Donne la description d'une des caractéristiques du concept	Identité Caractéristique
20. DemandeDescriptionAttributConcept		
21. ValidationDescriptionAttributConcept		
22. ImportanceAttributConcept	Décrit l'importance de la propriété caractéristique dans la discrimination du concept	Identité Caractéristique
23. DemandeImportanceAttributConcept		
24. ValidationImportanceAttributConcept		
25. RègleDiscrimination	Présente la règle de discrimination du concept à présenter aux apprenants	
26. DemandeRègleDiscrimination		
27. ValidationRègleDiscrimination		
28. Exemple	Présente un exemple du concept	Degré de complexité
29. DemandeExemple		
30. ValidationExemple		
31. CaractéristiqueExemple	Présente les caractéristique d'un exemple	Identité exemple
32. DemandeCaractéristiqueExemple		
33. ValidationCaractéristiqueExemple		
34. ContreExemple	Présente un contre exemple du concept	Degré de complexité
35. DemandeContreExemple		
36. ValidationContreExemple		
37. CaracteristiqueContreExemple	Présente les caractéristiques d'un contre exemple	Identité contreExemple
38. DemandeCaracteristiqueContreExemple		
39. ValidationCaracteristiqueContreExemple		
40. Exception	Présente une exception du concept	
41. DemandeException		
42. ValidationException		
43. CaractéristiqueException	Présente les caractéristiques de l'exception	Identité exception
44. DemandeCaractéristiqueException		
45. ValidationCaractéristiqueException		
46. QCM	Présente un QCM sur le concept	NombreChoix, DegréComplexité
47. DemandeQCM		
48. ReponseQCM	Présente la réponse au QCM courant	QCM courant
49. CorrectionReponseQCM	Produit le texte validant la réponse	QCM courant

	de l'apprenant au QCM courant	
50. DemandeCorrectionReponseQCM		
51. ValidationCorrectionReponseQCM		
52. SolutionQCM	Produit la solution au QCM courant	QCM courant
53. DemandeSolutionQCM		
54. ValidationSolutionQCM		
55. JustificationSolutionQCM	Produit un texte expliquant pourquoi la solution proposée est une solution	
56. DemandeJustificationSolutionQCM		
57. ValidationJustificationSolutionQCM		
58. MQCM	Présente un MQCM sur le concept	NombreChoix, NombreBonChoix, DegréComplexité
59. DemandeMQCM		
60. ReponseMQCM	Présente la réponse au MQCM courant	MQCM courant
61. CorrectionReponseMQCM	Produit le texte validant la réponse de l'apprenant au MQCM courant	MQCM courant
62. DemandeCorrectionReponseMQCM		
63. ValidationCorrectionReponseMQCM		
64. SolutionMQCM	Produit la solution au MQCM courant	MQCM courant
65. DemandeSolutionMQCM		
66. ValidationSolutionMQCM		
67. JustificationSolutionMQCM	Produit un texte expliquant la solution proposée	
68. DemandeJustificationSolutionMQCM		
69. ValidationJustificationSolutionMQCM		
70.		

10.2 Décomposition de la tâche d'enseignement de concept

Dans cette représentation, pour un souci de clarté nous avons uniquement représenté les règles de déclenchement. Le terme capsule à utiliser fait référence à la primitive de l'objet concept qu'il faut déclencher pour obtenir la ressource nécessaire à l'atteinte de l'objectif de la sous-tâche.

Sous-tâche père	**Sous-tâche**	**Acteur**	**Capsule à utiliser**	**Conditions de déclenchement**
1. Racine	EnseignementConcept(EC)	Tuteur		EC.Etat = Initial
2. EnseignementConcept	***ProductionObjectifEC(POEC)***	Tuteur		(EC.Etat # Initial ET EC.Etat # Terminé) OU (DOEC.Etat = Réalisé)
3. EnseignementConcept	***DemandeObjectifEC(DOEC)***	Apprenant		EC.Statut = EnCours
4. EnseignementConcept	***ValidationObjectifEC(VOEC)***	Apprenant		VOEC.NbItération < POEC.NbItération
5. EnseignementConcept	***Motivation(M)***	Tuteur		EC.Etat # Initial ET EC.Etat # Terminé
6. EnseignementConcept	***Acquisition(A)***	Tuteur		EC.Etat # Initial ET EC.Etat # Terminé
7. EnseignementConcept	***Performance(P)***	Tuteur		EC.Etat # Initial ET EC.Etat # Terminé
8. Motivation	***ProductionObjectifM(POM)***	Tuteur	*	(M.Etat # Initial ET M.Etat # Terminé) OU (DOM.Etat = Réalisé)
9. Motivation	***DemandeObjectifM(DOM)***	Apprenant	*	M.Statut = EnCours
10. Motivation	***ValidationObjectifM(VOM)***	Apprenant	*	VOM.NbItération < POM.NbItération
11. Motivation	***ProductionImportanceConcept(PIC)***	Tuteur	*	(M.Etat # Initial ET M.Etat # Terminé) OU DIC.Etat = Réalisé
12. Motivation	***DemandeImportanceConcept(DIC)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
13. Motivation	***ValidationImportanceConcept(VIC)***	Apprenant	*	VIC.NbItération < PIC.NbItération
14. Acquisition	***ProductionObjectifA(POA)***	Tuteur		(A.Etat # Initial ET A.Etat # Terminé) OU

				(DOA.Etat = Réalisé)
15. Acquisition	***DemandeObjectifA(DOA)***	Apprenant		A.Statut = EnCours
16. Acquisition	***ValidationObjectifA(VOA)***	Apprenant		VOA.NbItération < POA.NbItération
17. Acquisition	***Prealable(Pre)***	Tuteur		A.Etat # Initial ET A.Etat # Terminé
18. Acquisition	***Définition(Def)***	Tuteur		A.Etat # Initial ET A.Etat # Terminé
19. Acquisition	***Description(Des)***	Tuteur		A.Etat # Initial ET A.Etat # Terminé
20. Acquisition	***Attribut(Att)***	Tuteur		A.Etat # Initial ET A.Etat # Terminé
21. Acquisition	***Exception(Exc)***	Tuteur		A.Etat # Initial ET A.Etat # Terminé
22. Acquisition	***RègleDiscrimination(RD)***	Tuteur		A.Etat # Initial ET A.Etat # Terminé
23. Acquisition	***Exemple(Exe)***	Tuteur		A.Etat # Initial ET A.Etat # Terminé
24. Acquisition	***ContreExemple(CE)***	Tuteur		A.Etat # Initial ET A.Etat # Terminé
25. Préalable	***ProductionObjectifPre(POPre)***	Tuteur		(Pre.Etat # Initial ET Pre.Etat # Terminé) OU (DOPre.Etat = Réalisé)
26. Préalable	***DemandeObjectifPre(DOPre)***	Apprenant		Pre.Statut = EnCours
27. Préalable	***ValidationObjectifPre(VOPre)***	Apprenant		VOPre.NbItération < POPre.NbItération
28. Préalable	***ProductionPrealable(PPre)***	Tuteur	*	(Pre.Etat # Initial ET Pre.Etat # Terminé) OU (DPre.Etat = Réalisé)
29. Préalable	***DemandePrealable(DPre)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
30. Préalable	***ValidationPrealable(VPre)***	Apprenant	*	VPre.NbItération < PPre.NbItération
31. Définition	***ProductionObjectifD(PODef)***	Tuteur		(Def.Etat # Initial ET Def.Etat # Terminé) OU (DODef.Etat = Réalisé)
32. Définition	***DemandeObjectifD(DODef)***	Apprenant		Def.Statut = EnCours
33. Définition	***ValidationObjectifD(VODef)***	Apprenant		VODef.NbItération < PODef.NbItération
34. Définition	***ProductionDéfinition(Pdef)***	Tuteur	*	(Def.Etat # Initial ET

				Def.Etat # Terminé) OU (DDef.Etat = Réalisé)
35. Définition	***DemandeDéfinition(Ddef)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
36. Définition	***ValidationDéfinition(Vdef)***	Apprenant	*	VDef.NbItération < PDef.NbItération
37. Description	***ProductionObjectifD(PODes)***	Tuteur		(Des.Etat # Initial ET Des.Etat # Terminé) OU (DODes.Etat = Réalisé)
38. Description	***DemandeObjectifD(DODes)***	Apprenant		Des.Statut = EnCours
39. Description	***ValidationObjectifD(VODes)***	Apprenant		VODes.NbItération < PODes.NbItération
40. Description	***ProductionDescription(Pdes)***	Tuteur	*	(Des.Etat # Initial ET Des.Etat # Terminé) OU (DDes.Etat = Réalisé)
41. Description	***DemandeDescription(Ddes)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
42. Description	***ValidationDescription(Vdes)***	Apprenant	*	VDes.NbItération < PDes.NbItération
43. Attribut	***ProductionObjectifAtt(POAtt)***	Tuteur		(Att.Etat # Initial ET Att.Etat # Terminé) OU (DOAtt.Etat = Réalisé)
44. Attribut	***DemandeObjectifAtt(DOAtt)***	Apprenant		Att.Statut = EnCours
45. Attribut	***ValidationObjectifAtt(VOAtt)***	Apprenant		VOAtt.NbItération < POAtt.NbItération
46. Attribut	***ProductionAttribut(PAtt)***	Tuteur	*	(Att.Etat # Initial ET Att.Etat # Terminé) OU (DAtt.Etat = Réalisé)
47. Attribut	***DemandeAttribut(DAtt)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
48. Attribut	***ValidationAttribut(VAtt)***	Apprenant	*	VAtt.NbItération < PAtt.NbItération
49. Attribut	***ProductionCaracteristiqueAttribut(PCAtt)***	Tuteur	*	(Att.Statut = EnCours ET VAtt.NbIteration < PAtt.NbIteration) OU (DCAtt.Etat = Realise)
50. Attribut	***DemandeCaracteristiqueAttribut(DCAtt)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
51. Attribut	***ValidationCaracteristiqueAttribut(VCAtt)***	Apprenant	*	VCAtt.NbItération < PCAtt.NbItération
52. Attribut	***ProductionImportanceAttribut(PIAtt)***	Tuteur	*	Att.Statut = EnCours ET VAtt.NbIteration < PAtt.NbIteration) OU

				(DIAtt.Etat = Realise)
53. Attribut	***DemandeImportanceAttribut(DIAtt)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
54. Attribut	***ValidationImportanceAttribut(VIAtt)***	Apprenant	*	VIAtt.NbItération < PIAtt.NbItération
55. Exception	***ProductionObjectifExc(POExc)***	Tuteur		(Exc.Etat # Initial ET Exc.Etat # Terminé) OU (DOExc.Etat = Réalisé)
56. Exception	***DemandeObjectifExc(DOExc)***	Apprenant		Exc.Statut = EnCours
57. Exception	***ValidationObjectifExc(VOExc)***	Apprenant		VOExc.NbItération < POExc.NbItération
58. Exception	***ProductionException(PExc)***	Tuteur	*	(Exc.Etat # Initial ET Exc.Etat # Terminé) OU (DExc.Etat = Réalisé)
59. Exception	***DemandeException(DExc)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
60. Exception	***ValidationException(VExc)***	Apprenant	*	VExc.NbItération < PExc.NbItération
61. Exception	***ProductionCaracteristiqueException(PC Exc)***	Tuteur	*	(Exc.Statut = EnCours ET VExc.NbIteration < PExc.NbIteration) OU (DCExc.Etat = Realise)
62. Exception	***DemandeCaracteristiqueException(DCE xc)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
63. Exception	***ValidationCaracteristiqueException(VCE xc)***	Apprenant	*	VCExc.NbItération < PCExc.NbItération
64. RègleDiscriminat ion	***ProductionObjectifRD(PORD)***	Tuteur		(RD.Etat # Initial ET RD.Etat # Terminé) OU (DORD.Etat = Réalisé)
65. RègleDiscriminat ion	***DemandeObjectifRD(DORD)***	Apprenant		RD.Statut = EnCours
66. RègleDiscriminat ion	***ValidationObjectifRD(VORD)***	Apprenant		VORD.NbItération < PORD.NbItération
67. RègleDiscriminat ion	***ProductionRègleDiscrimination(PRD)***	Tuteur	*	(RD.Etat # Initial ET RD.Etat # Terminé) OU (DRD.Etat = Réalisé)
68. RègleDiscriminat ion	***DemandeRègleDiscrimination(DRD)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
69. RègleDiscriminat ion	***ValidationRègleDiscrimination(VRD)***	Apprenant	*	VRD.NbItération < PRD.NbItération
70. Exemple	***ProductionObjectifExe(POExe)***	Tuteur		(Exe.Etat # Initial ET Exe.Etat # Terminé) OU

				(DOExe.Etat = Réalisé)
71. Exemple	***DemandeObjectifExc(DOExe)***	Apprenant		Exe.Statut = EnCours
72. Exemple	***ValidationObjectifExc(VOExe)***	Apprenant		VOExe.NbItération < POExe.NbItération
73. Exemple	***ProductionExemple(PExe)***	Tuteur	*	(Exe.Etat # Initial ET Exe.Etat # Terminé) OU (DExe.Etat = Réalisé)
74. Exemple	***DemandeExemple(DExe)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
75. Exemple	***ValidationExemple(VExe)***	Apprenant	*	VExe.NbItération < PExe.NbItération
76. Exemple	***ProductionCaracteristiqueExemple(PCExe)***	Tuteur	*	(Exe.Statut = EnCours ET VExe.NbIteration < PExe.NbIteration) OU (DCExe.Etat = Realise)
77. Exemple	***DemandeCaracteristiqueExemple(DCExe)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
78. Exemple	***ValidationCaracteristiqueExemple(VCExe)***	Apprenant	*	VCExe.NbItération < PCExe.NbItération
79. ContreExemple	***ProductionObjectifCE(POCE)***	Tuteur		(CE.Etat # Initial ET CE.Etat # Terminé) OU (DOCE.Etat = Réalisé)
80. ContreExemple	***DemandeObjectifCE(DOCE)***	Apprenant		CE.Statut = EnCours
81. ContreExemple	***ValidationObjectifCE(VOCE)***	Apprenant		VOCE.NbItération < POCE.NbItération
82. ContreExemple	***ProductionContreExemple(PCE)***	Tuteur	*	(CE.Etat # Initial ET CE.Etat # Terminé) OU (DCE.Etat = Réalisé)
83. ContreExemple	***DemandeContreExemple(DCE)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
84. ContreExemple	***ValidationContreExemple(VCE)***	Apprenant	*	VCE.NbItération < PCE.NbItération
85. ContreExemple	***ProductionCaracteristiqueContreExemple(PCCE)***	Tuteur	*	(CE.Statut = EnCours ET VCE.NbIteration < PCE.NbIteration) OU (DCCE.Etat = Realise)
86. ContreExemple	***DemandeCaracteristiqueContreExemple(DCCE)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
87. ContreExemple	***ValidationCaracteristiqueContreExemple(VCCE)***	Apprenant	*	VCCE.NbItération < PCCE.NbItération
88. Performance	***ProductionObjectifP(POP)***	Tuteur		(P.Etat # Initial ET P.Etat # Terminé) OU(DOP.Etat = Réalisé)

89. Performance	***DemandeObjectifP(DOP)***	Apprenant		P.Statut = EnCours
90. Performance	***ValidationObjectifP(VOP)***	Apprenant		VOP.NbItération < POP.NbItération
91. Performance	***QCM***	Tuteur		P.Etat # Initial ET P.Etat # Terminé
92. Performance	***MQCM***	Tuteur		P.Etat # Initial ET P.Etat # Terminé
93. QCM	***ProductionObjectifQCM(POQCM)***	Tuteur		(QCM.Etat # Initial ET QCM.Etat # Terminé) OU (DOQCM.Etat = Réalisé)
94. QCM	***DemandeObjectifQCM(DOQCM)***	Apprenant		QCM.Statut = EnCours
95. QCM	***ValidationObjectifQCM(VOQCM)***	Apprenant		VOQCM.NbItération < POQCM.NbItération
96. QCM	***ProductionQCM(PQCM)***	Tuteur	*	(QCM.Etat # Initial ET QCM.Etat # Terminé) OU (DQCM.Etat = Réalisé)
97. QCM	***DemandeQCM(DQCM)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
98. QCM	***ReponseQCM(RQCM)***	Apprenant	*	RQCM.NbItération < PQCM.NbItération
99. QCM	***ProductionCorrectionReponseQCM(PCR QCM)***	Tuteur	*	RQCM.Etat = Réalisé OU DCRQCM.Etat = Réalisé
100. QCM	***DemandeCorrectionReponseQCM(DCR QCM)***	Apprenant	*	RQCM.Etat = Réalisé
101. QCM	***ValidationCorrectionReponseQCM(VCR QCM)***	Apprenant	*	VCRQCM.NbItération < PCRQCM.NbItération
102. QCM	***ProductionSolutionQCM(PSQCM)***	Tuteur	*%	RQCM.Etat = Réalisé OU DSQCM.Etat = Réalisé
103. QCM	***DemandeSolutionQCM(DSQCM)***	Apprenant	*%	PQCM.Etat = Réalisé
104. QCM	***ValidationSolutionQCM(VSQCM)***	Apprenant	*	VSQCM.NbItération < PSQCM.NbItération
105. QCM	***ProductionJustificationSolutionQCM(PJ SQCM)***	Tuteur	*%	DJSQCM.Etat = Réalisé
106. QCM	***DemandeJustificationSolutionQCM(DJS QCM)***	Apprenant	*%	PSQCM.Etat = Réalisé
107. QCM	***ValidationJustificationSolutionQCM(VJ SQCM)***	Apprenant	*	VJSQCM.NbItération < PJSQCM.NbItération
108. QCM	***ProductionCaracteristiqueChoixQCM(P CCQCM)***	Tuteur	*%	DCCQCM.Etat = Réalisé
109. QCM	***DemandeCaracteristiqueChoixQCM(DC CQCM)***	Apprenant	*%	PQCM.Etat = Réalisé
110. QCM	***ValidationCaracteristiqueChoixQCM(VC***	Apprenant	*	VCCQCM.NbItération <

	CQCM)			PCCQCM.NbItération
111. MQCM	***ProductionObjectifMQCM(POMQCM)***	Tuteur		(MQCM.Etat # Initial ET MQCM.Etat # Terminé) OU (DOMQCM.Etat = Réalisé)
112. MQCM	***DemandeObjectifMQCM(DOMQCM)***	Apprenant		MQCM.Statut = EnCours
113. MQCM	***ValidationObjectifMQCM(VOMQCM)***	Apprenant		VOMQCM.NbItération < POMQCM.NbItération
114. MQCM	***Production MQCM(PMQCM)***	Tuteur	*	(MQCM.Etat # Initial ET MQCM.Etat # Terminé) OU (DMQCM.Etat = Réalisé)
115. MQCM	***DemandeMQCM(DMQCM)***	Apprenant	*	EC.Etat # Initial ET EC.Etat # Terminé
116. MQCM	***ReponseMQCM(RMQCM)***	Apprenant	*	RMQCM.NbItération < PMQCM.NbItération
117. MQCM	***ProductionCorrectionReponseMQCM(PCRMQCM)***	Tuteur	*%	RMQCM.Etat = Réalisé OU DCRMQCM.Etat = Réalisé
118. MQCM	***DemandeCorrectionReponseMQCM(DCRMQCM)***	Apprenant	*%	RMQCM.Etat = Réalisé
119. MQCM	***ValidationCorrectionReponseMQCM(VCRMQCM)***	Apprenant	*	VCRMQCM.NbItération < PCRMQCM.NbItération
120. MQCM	***ProductionSolutionMQCM(PSMQCM)***	Tuteur	*%	RMQCM.Etat = Réalisé OU DSMQCM.Etat = Réalisé
121. MQCM	***DemandeSolutionMQCM(DSMQCM)***	Apprenant	*%	PMQCM.Etat = Réalisé
122. MQCM	***ValidationSolutionMQCM(VSMQCM)***	Apprenant	*	VSMQCM.NbItération < PSMQCM.NbItération
123. MQCM	***ProductionJustificationSolutionMQCM(PJSMQCM)***	Tuteur	*%	DJSMQCM.Etat = Réalisé
124. MQCM	***DemandeJustificationSolutionMQCM(DJSMQCM)***	Apprenant	*%	PSMQCM.Etat = Réalisé
125. MQCM	***ValidationJustificationSolutionMQCM(VJSMQCM)***	Apprenant	*	VJSMQCM.NbItération < PJSMQCM.NbItération
126. MQCM	***ProductionCaracteristiqueChoixMQCM(PCCMQCM)***	Tuteur	*%	DCCMQCM.Etat = Réalisé
127. MQCM	***DemandeCaracteristiqueChoixMQCM(DCCMQCM)***	Apprenant	*%	PMQCM.Etat = Réalisé
128. MQCM	***ValidationCaracteristiqueChoixMQCM(VCCMQCM)***	Apprenant	*	VCCMQCM.NbItération < PCCMQCM.NbItération
129.				