Conception de services et de protocoles pour la gestion de groupes coopératifs

Thierry Villemur

Année 1995

THÈSE

Présentée

au **LABORATOIRE D'ANALYSE ET D'ARCHITECTURE DES SYSTÈMES DU CNRS**

en vue de l'obtention du titre

de **DOCTEUR DE L'UNIVERSITÉ PAUL SABATIER DE TOULOUSE**

Spécialité : Informatique

par

Thierry VILLEMUR

CONCEPTION DE SERVICES ET DE PROTOCOLES POUR LA GESTION DE GROUPES COOPÉRATIFS

Soutenue le 3 Janvier 1995, devant le jury :

MM.	G.	JUANOLE	Président
	E.	HORLAIT	Rapporteurs
	M.	TRÉHEL	
	M.	DIAZ	Directeur de thèse
	P.	AZÉMA	Examinateurs
	B.	BOUCHARÉ	
	J. J.	MERCIER	
	J. J.	PANSIOT	
	M.	RAYNAL	

Rapport LAAS N° 95002
Cette thèse a été préparée au Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
7, avenue du Colonel Roche31077 TOULOUSE Cedex

Labor omnia vincit improbus.
(Un travail opiniâtre vient à bout de tout.)

Virgile

Remerciements

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS) dirigé au cours de mon séjour par Monsieur A. Costes que je tiens à remercier cordialement pour son accueil.

J'adresse mes plus vifs remerciements à Monsieur M. Diaz, Directeur de Recherche au CNRS et responsable du groupe Outils Logiciels pour la Communication (OLC) qui m'a accepté au sein de son équipe de recherche et qui a encadré ma thèse durant ces trois dernières années. Que sa constante aide dans mon travail, ses conseils pertinents et sa grande disponibilité puissent trouver ici la marque de ma sympathie la plus sincère.

Je remercie également Monsieur J. L. Durieux, professeur à l'Université Paul Sabatier, de m'avoir accepté dans la formation doctorale Informatique Fondamentale et Parallélisme qu'il dirige.

Je suis très reconnaissant à :

Monsieur P. Azéma, Directeur de Recherches au CNRS,

Monsieur B. Boucharé, Ingénieur au CNET de Lannion,

Monsieur M. Diaz, Directeur de Recherches au CNRS,

Monsieur E. Horlait, Professeur à l'Université Pierre et Marie Curie de Paris,

Monsieur G. Juanole, Professeur à l'Université Paul Sabatier de Toulouse III,

Monsieur J. J. Mercier, Professeur à l'IUT d'Informatique de Blagnac,

Monsieur J. J. Pansiot, Professeur à l'Université Louis Pasteur de Strasbourg,

Monsieur M. Raynal, Professeur à l'Université de Rennes I,

Monsieur M. Tréhel, Professeur à l'Université de Besançon,

pour l'honneur qu'ils me font en participant au jury de thèse, et plus particulièrement à Messieurs E. Horlait et M. Tréhel qui ont accepté la charge de Rapporteur.

J'adresse un grand merci à tous mes collègues de bureau et à tous ceux du bureau voisin que j'ai côtoyés durant ces trois années. Je leur sais particulièrement gré de leur amical et réconfortant soutien durant tout ce temps, mais surtout au cours de la rédaction de ce mémoire : C. Chassot, A. Lozes, B. Meunier, M. Fournier, P. Gradit, E. Guillochin, M. Koens, P. Owezarski.

A travers eux, mes remerciements s'étendent à l'ensemble du groupe OLC. Les discussions fructueuses et chaleureuses menées avec les différents membres de l'équipe contribuent à fournir un cadre de travail convivial, enrichissant et performant.

Je remercie le service Informatique et Instrumentation, plus particulièrement M. Aguera, V. Baudin et J. M. Pons, qui assure avec une grande efficacité le support technique indispensable à la réalisation de nos travaux.

Le travail de ce mémoire a été réalisé grâce au support financier du Centre National d'Etudes des Télécommunications (CNET) et du Centre Commun d'Etudes des Télédiffusions et Télécommunications (CCETT) dans le cadre du projet CESAME. Je leur sais gré de leur soutien et je remercie également tous les participants du projet CESAME que j'ai pu rencontrer durant ces trois dernières années.

Dans un cadre plus général à ce mémoire, je tiens à remercier l'ensemble des membres du Conseil de Laboratoire dirigé par A. Costes et ceux du Comité de Rédaction de la Lettre du LAAS dirigé par M. T. Ippolito. Le premier m'a permis de découvrir le côté administratif d'un laboratoire et m'a apporté une vision plus globale très enrichissante de la vie d'un laboratoire alors que le deuxième comité contribue à la promotion du LAAS.

Finalement, j'adresse mes remerciements à l'ensemble du personnel du service de «Documentation-Edition» qui a permis la réalisation de ce mémoire et, à travers eux, je remercie l'ensemble du personnel administratif.

Table des matières

Introduction

Le travail coopératif, dénommé aussi «Computer Supported Cooperative Work» (CSCW), est un nouveau domaine multi-disciplinaire qui cherche à mêler la notion de travail d'utilisateurs en groupe à celle d'outils ou de supports pour assister ce travail de groupe. Plus précisément, le travail coopératif englobe des études faisant partie des sciences sociales (sociologie, théorie des organisations) qui s'occupent des relations homme-homme à travers les systèmes coopératifs, des performances et des problèmes posés par le travail de groupe ou par l'utilisation d'outils de travail de groupe. Cette partie représente la part la moins «technique» de la coopération par rapport aux autres domaines plus technologiques et plus matériels. Une autre branche de la coopération s'inspire des recherches menées en sciences cognitives, en intelligence artificielle distribuée et en systèmes multi-agents. Cette deuxième partie s'attache davantage à la représentation de la connaissance, à la sémantique des données, aux raisonnements sur ces données ainsi qu'à la planification et à l'aide à la réalisation de tâches de groupe. Le troisième domaine contient toutes les études d'architecture informatique, les solutions logicielles, la réalisation d'outils pour l'exécution du travail de groupe. Cette partie provient des domaines des systèmes distribués multi-utilisateurs et des réseaux pour la conception de services et de protocoles de coopération qui assurent les transmissions et les échanges de données entre les membres des groupes distribués. Ce dernier domaine fournit surtout les plate-formes ainsi que les supports logiciels indispensables aux interactions entre les divers participants à une tâche coopérative. Le mélange de ces trois grandes catégories de recherche, sociologique, cognitif, technologique, forme le CSCW. Cependant, sa principale finalité reste le collecticiel, ou logiciel de groupe, qui comprend les applications informatiques destinées à faire travailler ensemble des groupes d'utilisateurs et de leur fournir un support pour ce travail.

Face à une telle diversité d'approches, le travail présenté dans ce mémoire se focalise davantage sur le domaine de la conception formelle des protocoles dans les systèmes distribués, du fait de nombreux modèles, formalismes et langages disponibles, tout en cherchant cependant à garder un lien avec les autres domaines. En effet, le cadre général de cette étude est de fournir un ensemble de services et de protocoles de communication génériques qui pourraient être de ce fait utilisés par de multiples applications coopératives, et qui pourraient faire partie d'une plate-forme logicielle de support des applications de groupe. Nous avons en particulier développé l'architecture des logiciels étudiés en proposant et en concevant de façon formelle des services de groupe et des protocoles qui les mettent en oeuvre.

La conception formelle des services et des protocoles, s'appuyant le plus souvent sur des modèles de machines à états ou de réseaux de Petri, permet de spécifier de manière non ambiguë les mécanismes et les règles de communication définis entre des groupes d'entités distribuées. Elle facilite leur mise au point et fournit également des techniques d'analyse pour la vérification exhaustive des propriétés que le système décrit doit respecter. Actuellement, de plus en plus, les modèles formels ne sont pas utilisés tels quels, mais ils interviennent dans la définition de langages tel Estelle qui offrent de plus grandes facilités pour la représentation des entités distribuées, pour la simulation de leurs comportements et pour l'étude de l'évolution du système. Cette adaptation et cette représentation des modèles sous une forme langage adéquate présente également l'avantage non négligeable de pouvoir réutiliser directement des parties de la spécification initiale pour une implantation sans procéder à des réécritures du code, toujours source d'erreur. Plusieurs techniques de description formelle ont donc servi à supporter la conception des services et des protocoles coopératifs proposés.

L'organisation de ce mémoire respecte la dualité entre les parties générales provenant du CSCW et la conception formelle des services et des protocoles correspondants. Le lien est cependant fait par les services qui utilisent les modèles et les représentations des groupes coopératifs provenant du CSCW, tandis que leur réalisation par les protocoles s'appuie sur les techniques de description formelles des systèmes distribués.

D'une façon plus générale, le travail présenté dans ce mémoire s'inscrit dans le cadre du projet CESAME qui a débuté en 1992 dans le cadre d'une collaboration entre le Centre National d'Etudes des Télécommunications (CNET), le Centre Commun d'Etudes des Télédiffusions et Télécommunications (CCETT) et le Centre National de la Recherche Scientifique (CNRS) et qui vise à étudier la conception formelle de systèmes haut débit multimédias coopératifs. A l'intérieur de ce projet, quatre grands axes de recherche ont été dégagés :

- Le premier comprend l'étude et la proposition de nouveaux services et protocoles de communication qui utilisent les réseaux à haut débit, adaptés aux besoins des données multimédias.
- Une deuxième partie de ce projet s'occupe des besoins en synchronisation que les objets multimédias nécessitent. On rencontre ici la proposition de modèles pour représenter et traduire la synchronisation à travers des systèmes distribués.
- Le troisième axe de recherche, appelé coopération, comprend l'extension des échanges, limités auparavant à des communications point à point entre deux entités, à des groupes coopératifs d'utilisateurs, ainsi que l'étude de leurs performances.
- Finalement, le dernier axe de recherche vise à la proposition d'une application illustrative dans laquelle seront intégrés les travaux précédents.

Cette thèse s'inscrit principalement dans le troisième axe de recherche, c'est-à-dire celui de la coopération tout en restant également liée au domaine de l'application. Dans ce cadre, un modèle et des outils logiciels doivent être proposés, afin de caractériser et de manipuler les groupes coopératifs pouvant être de grande taille pour des organisations complexes.

Le *chapitre I* effectue une synthèse de la coopération à partir d'une importante recherche bibliographique. Il présente de façon générale les termes et les notions du travail coopératif, puis classe les groupes coopératifs suivant les modes d'interaction entre leurs membres. Par la suite, un découpage fonctionnel des applications coopératives est proposé. Les modèles et paradigmes rencontrés permettent de représenter soit le comportement individuel d'agents au sein de coopérations, soit des groupes coopératifs contenant diverses entités. Les applications coopératives sont ensuite classées en six grandes catégories : les travaux tirés des systèmes distribués, les systèmes support de décisions de groupe, les messageries sensibles au contexte, les systèmes d'intelligence artificielle distribuée, les conférences, et les applications diverses. Chacune de ces parties contient un ensemble d'applications illustratives afin de mettre en évidence leurs principales caractéristiques. Quelques remarques et quelques critiques sur les modèles et sur les applications sont données à la fin.

Le *chapitre II* décrit le modèle défini pour représenter les groupes coopératifs et leur structuration. Ce modèle est basé sur la logique modale. Les graphes qui proviennent de cette logique donnent la structure des groupes coopératifs. Les noeuds du graphe représentent les entités en coopération tandis que les flèches donnent les liens que l'on a entre les membres des groupes. Ces liens définissent les possibilités de communication de chaque membre à l'intérieur d'un groupe. Des domaines assurent la répartition de la connaissance, ainsi que sa consistance entre divers sous-groupes d'agents. Trois extensions du modèle initial sont proposées : la première prend en compte l'évolution dynamique de la coopération dans le temps. Elle définit des configurations valides qui indiquent quels sont les agents qui peuvent ou doivent coopérer ensemble pour effectuer le travail coopératif. La deuxième ajoute la notion d'apartés dans le modèle. Un aparté est en fait un sous-groupe temporaire et très dynamique créé à l'intérieur d'un groupe coopératif. Sa structure respecte celle donnée par le groupe coopératif. La troisième extension montre l'influence des dépendances logiques entre les données de la coopération sur les communications effectivement réalisées.

Le *chapitre III* présente d'abord le langage de spécification formelle Estelle ainsi que son utilisation dans le cadre de la spécification de services et de protocoles de communication. Puis, il décrit le service de communication qui gère l'évolution dynamique d'un groupe coopératif dans le temps. Ce service prend en compte les demandes des entités pour participer ou abandonner la coopération. Il fait ainsi évoluer dynamiquement dans le temps la structure du groupe qui doit cependant rester cohérente et doit correspondre à une configuration adaptée au travail réalisé par le groupe. De plus, chaque modification potentielle est soumise de façon coopérative à la décision des entités présentes en coopération. Un protocole associé au service est également défini. Le système composé du service et du protocole est spécifié en Estelle en suivant le modèle OSI de l'ISO. Le service et le protocole sont ensuite étendus pour prendre en compte les déconnexions brutales possibles des différents coopérants et pour reconfigurer la coopération suite à ces anomalies. Une nouvelle spécification Estelle traitant les erreurs est alors présentée. En dernier lieu, l'adaptation possible des services et des protocoles proposés à des coopérations multi-domaines est montrée.

Le *chapitre IV* est la suite directe du chapitre précédent. Il présente la vérification des services et protocoles d'appartenance. Tout d'abord, il donne le principe de vérification basé sur l'obtention d'un graphe d'accessibilité qui représente l'ensemble des états du système à étudier, puis sur son analyse par des techniques de bissimulation. La vérification a été effectuée à partir de l'environnement multi-agents VAL. Une spécification VAL est composée d'un ensemble d'agents communiquants dont le comportement est défini par un formalisme qui décrit des réseaux de Petri dynamiques à prédicat/transition. La partie suivante montre le passage de la spécification Estelle initiale à la spécification VAL adaptée à des fins de vérification et présente l'incidence des transformations sur la taille des graphes d'accessibilité. L'analyse du protocole

est alors effectuée au moyen de projections. La deuxième partie de ce chapitre présente l'implantation semi-automatique du service et du protocole à partir de la spécification Estelle. Elle décrit la démarche utilisée, les transformations effectuées et les adaptations réalisées. Ce système a été compilé sur un réseau de machines SUN qui fonctionnent sous le système d'exploitation SOLARIS2. Les utilisateurs coopérants sont interfacés avec le système de multi-fenêtrage X Window pour faciliter les interactions avec le service d'appartenance dynamique.

Le *chapitre V* étend les fonctionnalités déjà présentées. Il décrit tout d'abord le service qui assure la formation dynamique des apartés au cours de la coopération. Ce service permet à un ensemble d'entités de former un aparté, d'échanger des données et de communiquer à l'intérieur de cet aparté, puis de détruire cet aparté lorsque l'ensemble des membres de l'aparté veulent le quitter. Il gère également les interférences entre la formation des apartés et l'évolution dynamique de la structure du groupe coopératif. Un protocole qui fournit le service présenté est également proposé. Ce service et ce protocole sont ensuite spécifiés en Estelle en suivant le modèle OSI. La seconde partie présente un service de gestion des dépendances de données manipulées en coopération. Ce service ne gère que les cas particuliers de dépendances simples dans lesquelles la valeur d'une donnée ne peut dépendre qu'au plus de celle d'une autre valeur. Il permet de créer, détruire, modifier de telles dépendances. Le protocole proposé cherche le plus possible à favoriser les changements rapides des valeurs des données dépendantes au détriment des changements de dépendances. Ce protocole est totalement distribué entre les coopérants et son fonctionnement s'appuie sur le principe du protocole de verrouillage à deux phases. Son fonctionnement a été mis au point en Estelle. En dernier lieu, son intégration possible avec les deux services précédents est montrée.

A terme, l'ensemble de ces études devrait permettre de définir et de concevoir un noyau logiciel pour la gestion de groupes coopératifs qui serait utilisable pour supporter de multiples applications.

Chapitre I

Synthèse sur la coopération et sur le travail coopératif

Introduction

Ce chapitre vise à présenter les principales notions relatives à la coopération et au travail coopératif qui apparaissent dans la littérature et à proposer quelques définitions liées à ces notions. Une synthèse préliminaire des principales approches développées dans le domaine de la coopération a été effectuée dans [DIAZ93b], [DIAZ93c].

Tout d'abord, ce chapitre cherche à caractériser et à expliquer de façon générale ce qu'est la coopération, puis analyse par la suite les modes d'interaction entre les membres de groupes coopératifs, propose un découpage fonctionnel des applications coopératives, étudie les divers modèles proposés, et classifie en grands groupes les applications coopératives rencontrées. Dans chaque groupe, un ensemble d'applications significatives sont présentées et leurs principales caractéristiques détaillées.

Cette étude vise à extraire une définition globale de la coopération et des travaux coopératifs, à analyser les approches retenues, ceci afin de servir de base à la suite de nos travaux. En effet, les modèles utilisés pour représenter des groupes coopératifs ont servi de support à la proposition du modèle de coopération, présenté par la suite, sur lequel s'appuie l'ensemble de nos travaux. De plus, l'étude des applications et de leurs caractéristiques vise également à dégager un ensemble de services de communication génériques à tous ces travaux. Ces services pourraient faire partie d'une plate-forme coopérative, et seraient appelés et utilisés par des applications multi-utilisateurs s'appuyant sur cette plate-forme.

L'organisation de ce premier chapitre est la suivante :

La section 1 présente de façon générale la coopération et en donne quelques définitions. Elle propose une classification des interactions en coopération, puis décrit les principales fonctionnalités des systèmes coopératifs. La section 2 présente l'ensemble des modèles et paradigmes rencontrés pour formaliser les fonctions des agents coopératifs ainsi que leurs diverses interactions, et pour décrire des groupes coopératifs. La section 3 donne les grands types d'applications coopératives. La section 4 effectue quelques remarques critiques sur les modèles, sur les applications, et sur la coopération en général.

1. Historique et définitions

La notion de travail coopératif est ancienne. Cependant, son application au travail sur ordinateur est assez récente et a commencé à s'étendre vers le milieu des années 80.

1.1. Généralités

La première définition de la coopération et du travail coopératif a été donnée par K. Marx en 1867 [BANN89]. Le dictionnaire Larousse [LARO77] donne les définitions suivantes :

- *Coopérer* : Agir conjointement avec quelqu'un.
- *Coopération* :
  a) Méthode d'action économique dans laquelle des personnes ayant des intérêts communs constituent une entreprise où les droits de chacun à la gestion sont égaux et où le profit est réparti entre les seuls associés au prorata de leur activité.
  b) Forme d'aide à certains pays en voie de développement.

Bien que les explications précédentes soient peu rattachées à l'informatique, elles donnent une première idée de ce que recoupe la notion de coopération. Deux autres explications provenant du monde informatique sont :

- *Collecticiel* [ELLI91] : un système à base d'ordinateurs qui supporte des groupes de personnes réalisant en commun une tâche ou un but et qui fournit une interface pour accéder à un environnement commun.
- *Travail coopératif avec prise de décision en commun* [KRAE88] : un système informatique qui facilite la résolution de problèmes par un ensemble de décideurs travaillant en groupe.

1.2. CSCW. Collecticiel

Le terme *CSCW*, sigle de «*Computer Supported Cooperative Work*», a été introduit en 1988 pour désigner tous les travaux et toutes les applications développés pour des groupes d'utilisateurs [GREI88], [BANN89], [DESP93], [BLAI94]. Il ne s'agit pas seulement, dans ces applications, de prendre en compte uniquement les interactions qu'une personne peut avoir avec un ordinateur, mais il s'agit de s'occuper des interactions entre un groupe de personnes à travers un réseau d'ordinateurs. Ici, on n'a pas uniquement un dialogue homme/machine mais également des échanges entre des personnes (ou des agents) via un système informatique. Le CSCW est tout d'abord un domaine multi-disciplinaire dans lequel interviennent des recherches provenant des sciences sociales (sociologie, théorie des organisations, psychologie). Ces études prennent en compte l'organisation des personnes travaillant ensemble, leurs rapports entre elles, l'ef-

ficacité d'un groupe pour la réalisation d'un travail, etc... Les sciences cognitives, notamment l'intelligence artificielle distribuée, interviennent aussi dans le CSCW. Leur apport se situe dans l'interprétation de la sémantique des informations échangées, l'attribution des tâches, la planification, l'aide à la réalisation de tâches en commun... En dernier lieu, le CSCW utilise les travaux réalisés en informatique dans les domaines des systèmes distribués et des télécommunications pour les stockages, transferts et échanges d'informations.

Le CSCW peut être découpé en quatre grandes catégories de complexité conceptuelle différentes. Les deux premières catégories se rattachent plus aux systèmes distribués et réseaux, les deux suivantes s'inspirent davantage de toutes les recherches menées en intelligence artificielle distribuée et systèmes multi-agents :

Le Travail Collectif : on a là un ensemble d'individus, ou groupe de personnes, qui partagent de manière identique les responsabilités et les résultats lors de la réalisation d'une tâche. Il s'agit d'une assemblée démocratique dans le sens où chaque membre a les mêmes pouvoirs, les mêmes responsabilités, voit les mêmes résultats que les autres membres du groupe. Ce groupe met en commun et partage les responsabilités et les résultats d'une tâche précise. Les conférences sont un cas classique de travail collectif.

La Coopération : trois principaux niveaux composent un groupe coopératif. Tout d'abord, il s'agit de caractériser les objectifs et les buts à atteindre. Il faut donc décrire les buts, les tâches, les objectifs de façon à ce qu'ils soient compréhensibles et interprétables par les membres du groupe. On décrit dans ce niveau ce que doit réaliser le groupe. Un deuxième niveau est un méta-niveau de décomposition et d'attribution des tâches à un groupe. On met ici l'accent sur la façon de réaliser les tâches. Il faut notamment pouvoir partager entre les membres du groupe les objectifs à atteindre, décomposer les tâches en sous-tâches en évitant le plus possible les conflits, désaccords, redondances... Le troisième niveau comprend l'exécution des tâches par le groupe avec la nécessité que les membres se synchronisent et aient des résultats qui convergent. Bien entendu, pendant la réalisation des tâches, des désaccords ou des conflits peuvent survenir. Il faut donc des mécanismes pour résoudre ces problèmes afin que les membres du groupe n'aboutissent pas à des vues divergentes, voire contradictoires [DURF87]. Le point le plus délicat reste de minimiser et de résoudre les éventuels désaccords ou conflits. On peut voir un groupe coopératif comme un groupe qui partage les responsabilités pour réaliser des tâches en commun. Dans le cas de systèmes assistant les procédures administratives, la réalisation d'une commande ou la planification d'un voyage sont des tâches de coopération.

La Collaboration : il s'agit maintenant d'un ensemble d'individus (ou d'agents) qui possèdent chacun une certaine vue d'un problème. Ces individus cherchent à converger vers un même but global, chacun participant à la tâche globale et effectuant une partie de la résolution du but. Les chemins empruntés par chacune des personnes convergent vers un même but, mais peuvent être différents si les capacités et les rôles de chaque personne sont différents. Cette catégorie est caractéristique des résolveurs distribués de problèmes, provenant des recherches en intelligence artificielle distribuée, dans lesquels un ensemble d'agents cherche à résoudre de façon coopérative un problème global [DURF89]. Ce groupe a en commun une tâche à réaliser et il partage la réalisation de cette tâche.

Les Mondes multi-agents : cette extension des catégories précédentes considère un ensemble de groupes d'utilisateurs qui évoluent dans un même monde [HENN89], [BIGN89], [MARU90], [HEWI91]. Cela complique les problèmes rencontrés car on ne considère plus seulement un groupe d'utilisateurs mais un ensemble de groupes en interaction. Un utilisateur (ou agent) peut faire partie de plusieurs groupes. Comme il est décrit dans les catégories précédentes, on rencontre des problèmes de cohérence, de gestion de conflits à l'intérieur des groupes, mais en plus ces problèmes se posent dans les échanges inter-groupes.

Le terme CSCW englobe l'ensemble des études concernant le travail en groupe assisté par ordinateur, ceci comprenant les études humaines et sociologiques, alors que le collecticiel, qui est la traduction de «groupware», contraction de «group» et de «software», désigne tous les logiciels conçus pour des groupes d'utilisateurs. Le collecticiel comprend ainsi tous les supports ou produits logiciels qui supportent les applications du CSCW [MITT91], [ELLI91], [KARS94].

1.3. Modes d'interaction de la coopération

Les systèmes coopératifs peuvent supporter divers types de groupes, et offrir plusieurs possibilités pour les échanges de connaissance entre les membres des groupes.

1.3.1. Coopération synchrone, asynchrone

La *coopération synchrone* comprend les applications qui nécessitent que les membres du groupe soient présents en même temps pour effectuer le travail coopératif. Elle s'appuie sur des supports de communication synchrones. Quelques exemples de ce type d'application sont les conférences [OHMO92a], les fenêtres vidéo, les murs vidéo [FISH90], la téléprésence [GUIL92].

La *coopération asynchrone*, au contraire, correspond aux applications coopératives dont le support de communication est asynchrone, de type courrier électronique. Pour se dérouler, la coopération asynchrone ne nécessite pas la co-présence des différents membres du groupe. On rencontre par exemple dans cette catégorie des systèmes pour classifier les messages de manière semi-automatique [RODD89], des systèmes pour assister des conversations par courrier électronique [BIGN89].

Parfois, il est plus difficile de classifier les applications coopératives en synchrones ou asynchrones [RODD92]. Par exemple, l'édition de textes en commun peut se faire de manière synchrone ou asynchrone [MINO93]. Il est également possible que des applications coopératives utilisent les deux modes, comme dans certaines conférences.

La granularité des messages échangés est assez variée. Pour la coopération synchrone, elle peut aller de la mémoire partagée aux tableaux noirs [LESS80]. La coopération asynchrone va également de la simple émission de bits à l'échange de formules logiques [INTO91] en passant par l'expédition de messages plus ou moins structurés [RODD89], [BIGN89].

1.3.2. Coopération formelle, informelle

La *coopération informelle* regroupe toutes les applications coopératives dont le rôle ne consiste qu'à véhiculer des données sans les traiter ou les interpréter. Le support formel de ce type d'application est très faible ou inexistant. La sémantique de tous les échanges de données est laissée à la charge des membres du groupe coopératif, à priori les utilisateurs. A titre d'exemple, on trouve dans ce type d'application les conférences, les supports logiciel pour des réunions face à face, des plate-formes qui servent à rendre multi-utilisateurs une application mono-utilisateur [BONF89]...

La *coopération formelle* vise à prendre en compte la sémantique des données échangées. Le système, en analysant les données échangées, peut ainsi juger de l'avancée du travail effectué. Il peut aussi détecter des conflits, des incohérences. Le domaine de la coopération formelle comprend l'intelligence artificielle distribuée dans laquelle des groupes d'agents recherchent de façon collaborative la solution de problèmes globaux. Les résolveurs distribués de problèmes constituent un exemple de ce type d'application [DURF89]. L'intelligence artificielle décentralisée [DEMA90] fait aussi partie de la coopération formelle. Elle repose sur la définition d'agents et porte sur l'étude des activités d'agents autonomes dans un monde multi-agents. Chaque

agent réalise sa tâche et gère les conflits éventuels avec les tâches des autres agents. Le support formel utilisé est souvent la logique. Cependant, comme il sera vu par la suite, d'autres types de formalismes sont aussi employés.

1.3.3. Coopération en un même lieu, distante

Une troisième classification très fréquemment rencontrée dans la littérature [ELLI91], [RODD92], [NAVA92], [NAVA93], [HSU93], [KARS94] consiste à séparer les systèmes coopératifs suivant la distance entre les coopérants. Les premiers s'appliquent à des groupes dont les membres sont en un même lieu, les autres supportent des utilisateurs distants. Les supports logiciels des réunions face à face [KRAE88] entrent dans la première catégorie, tandis que la coopération distante utilise les ressources de communication fournies par les systèmes distribués et les réseaux.

1.4. Caractéristiques des systèmes coopératifs

Cette section présente un découpage modulaire des besoins des systèmes coopératifs (figure I.1). Ce découpage peut servir de base à la réalisation d'un modèle plus conséquent pour décrire des systèmes coopératifs. Il s'inspire essentiellement des modèles de [SMIT89], [PRIN89], [ROBI94] et des idées de [BANN89], [KRAS91], [WILS91].

1.4.1. Schéma de découpage

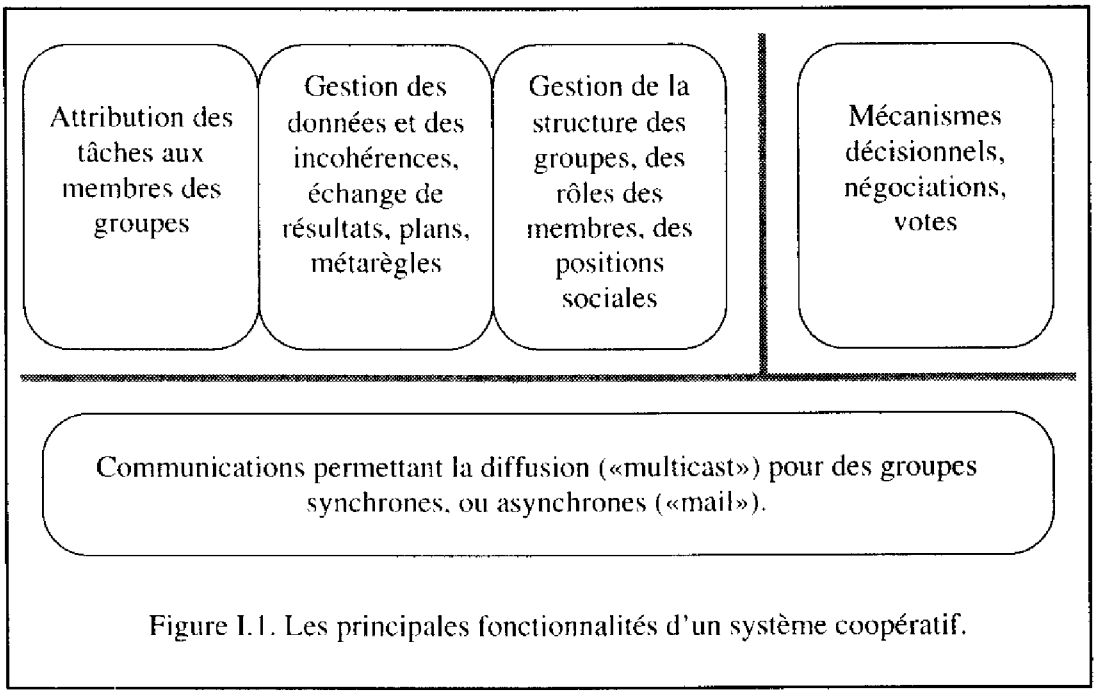


Figure I.1. Les principales fonctionnalités d'un système coopératif.

Les groupes fonctionnels de gestion des tâches, des données, de la structuration des groupes et des mécanismes décisionnels doivent s'appuyer sur des services de communication permettant aux entités en coopération d'échanger des informations. Les modules de gestion des tâches, des données, et des groupes sont finalement fortement reliés entre eux, et sont même interdépendants. L'exécution d'une tâche donnée introduit des manipulations d'informations et dépend de la configuration et des positions des agents dans le groupe coopératif. Les mécanismes décisionnels peuvent intervenir et être utilisés par les trois modules précédents soit pour des prises de décision, soit pour gérer des conflits.

1.4.2. Communications

Les notions de communication de groupe et de gestion de groupes de communication proviennent directement des domaines des systèmes distribués et des réseaux.

Pour des communications entre les membres de groupes asynchrones, les messageries de type X.400 [ISO10021] ont été utilisées avec la possibilité de se servir du répertoire X.500 [ISO9594] pour nommer et stocker les membres des groupes ou les ressources rattachées aux groupes [PRIN89], [BENF93].

Comme exemples de réalisations pour des groupes synchrones, les systèmes d'exploitation V [CHER85], [CHER88], AMOEBA [KAAS91] et ISIS [BIRM87], [BIRM91], [SCHI93] fournissent des facilités pour les communications de groupes, et pour la gestion d'adresses de groupe. Dans ces trois systèmes, les agents peuvent créer dynamiquement des groupes qu'ils peuvent joindre et quitter, et peuvent échanger des informations dans ces groupes. Le projet DELTA4 [DELT91], [VERI89], [BAPT90], [RODR93] a également vu la création de services pour la diffusion dans des groupes synchrones. Les communications de groupe proposées pour lesquelles les agents peuvent être émetteurs, receveurs de données ou les deux, ont ainsi conduit à la production d'un document ISO pour les transmissions de données «multipeer» [ISON2031]. Les services proposés pour les communications de groupes synchrones [HUGH89], [LIAN90a], [LIAN90b], [NGOH91], [OWEZ93] diffèrent suivant la fiabilité acceptée, l'ordonnancement entre les messages, l'ordonnancement entre plusieurs diffusions provenant de sources différentes, le nombre d'acquittements attendus...

1.4.3. Structuration des groupes

La définition de la structure des groupes, des rôles des membres et des positions sociales est tirée des travaux de la sociologie, des études du travail en entreprise et de l'automatisation des tâches de bureautique. Par exemple, un groupe de personnes peut être organisé en une hiérarchie ou bien en une organisation démocratique.

[HEWI91] propose un modèle de groupe basé sur le modèle d'acteurs dans lequel les acteurs sont regroupés en organisations. On peut ainsi avoir des communications entre acteurs appartenant à la même organisation ou entre organisations différentes.

[HENN89], [BENF93] effectuent un découpage en domaines pour la gestion et le partage des informations. Ces domaines permettent la gestion des ressources et des informations rattachées à chaque domaine, des rôles et des personnes de chaque domaine.

La structuration du groupe coopératif permet aux agents de posséder des connaissances sur eux-mêmes et sur les autres agents avec lesquels il coopèrent [KRAS91], [DURF89], [ROUS92]. Ces connaissances peuvent n'être que de simples croyances, qui sont des hypothèses sur l'évolution de ces agents, ou bien des hypothèses plus complexes sur le devenir des autres agents. Cette structuration introduit des bornes ou des limites d'action de chaque agent à l'intérieur des groupes coopératifs [DURF87]. Ainsi, suivant la structure d'un groupe, un membre peut ne pas connaître l'ensemble des membres du groupe et peut n'être en relation qu'avec un sous-ensemble des membres. La structure du groupe définit ainsi les possibilités ou limites d'action de chaque agent.

Le ou les rôles que chaque utilisateur possède servent à spécifier les responsabilités et les devoirs de chaque personne. Un rôle offre ainsi un ensemble de possibilités, de connaissances, de qualifications [SMIT89], [DOLL89]. Les rôles définissent des capacités de raisonnement qui s'appliquent sur ces données et connaissances. Ces capacités permettent de filtrer, d'interpréter, d'analyser les données et connaissances échangées [DURF87]. La structuration du groupe, en mettant en place les liaisons entre les divers membres du groupe, permet aussi de choisir les ty-

pes de coordination entre agents et d'attribuer les rôles et fonctions des agents dans la structure logicielle. Suivant leur position dans le groupe, certains agents peuvent avoir un rôle coordinateur : des autorités et pouvoirs peuvent leur être donnés afin de surveiller l'avancée du travail, de contrôler et d'arrêter les déviations des agents, de supprimer d'éventuels blocages ou ralentissements inacceptables. Ces capacités de raisonnement permettent de planifier le travail, d'éviter ou de résoudre d'éventuels conflits.

Les groupes servent aussi à maintenir une connaissance commune à un ensemble de personnes [BIGN89], [HEWI91]. Un groupe coopératif, lorsqu'il évolue, crée un contexte qui lui est propre et, suivant l'apport de nouvelles informations et la mise à jour d'autres informations, ce contexte est modifié dans le temps. A chaque instant, ce contexte définit la connaissance actuelle du groupe, connaissance dépendante de l'historique du groupe et des informations échangées. Ce contexte doit être maintenu, notamment lorsque le groupe coopératif est dynamique et qu'il peut recevoir de nouveaux arrivants. Il faut donc définir des points dans lequel le contexte courant est cohérent, afin de pouvoir le stocker ou le communiquer.

Les groupes coopératifs peuvent donc être définis par une structure statique, figée tout le long de la réalisation du travail coopératif, ou bien par une structure plus dynamique qui cherche à s'adapter à chaque instant aux besoins instantanés du travail à réaliser. Cette structure dynamique peut être guidée par les buts ou sous-buts associés au groupe, qui représentent les travaux que ce groupe doit effectuer.

1.4.4. Gestion des données

La gestion de la signification des données et des incohérences est un problème souvent rencontré en intelligence artificielle distribuée. Ces incohérences proviennent du fait qu'un utilisateur ne possède qu'une vision partielle de l'ensemble du système, n'accède qu'à une partie des données et doit donc se contenter pour raisonner des données qu'il peut connaître.

Les agents possèdent et se composent de données et de connaissances, qui sont reçues, stockées, échangées. Ils filtrent sémantiquement l'information reçue. Ce filtrage peut être sémantique, c'est-à-dire que le contenu du message est analysé, mais aussi cognitif, c'est-à-dire que le récepteur prend en compte l'identité de l'agent qui a envoyé le message, le contexte qu'il possède sur cet agent, l'importance ou la position sociale dans le groupe du détenteur de l'information. Lors d'échanges d'informations, les agents peuvent adopter plusieurs degrés ou tons dans leurs communications. Cela leur permet de traduire l'importance ou l'urgence de certains messages par rapport à d'autres. On peut ainsi avoir des énoncés assertifs, qui expriment des propriétés, des énoncés directifs, qui correspondent plus à des ordres, des énoncés explicatifs...

Lorsque un système se trouve découpé en plusieurs groupes, un utilisateur ne peut pas avoir accès aux informations appartenant aux groupes dont il ne fait pas partie. Par conséquent, le découpage en plusieurs groupes masque de l'information aux agents ne faisant pas partie des groupes et est donc un facteur supplémentaire d'incohérence [BIGN89]. Un autre facteur d'incohérence est l'arrivée de nouveaux membres dans un groupe. En effet, les nouveaux arrivants doivent être mis au courant de l'évolution du groupe pour qu'ils aient la même information que les autres membres [BIGN89]. Le système doit donc garder l'état courant du groupe afin de l'indiquer aux nouveaux arrivants.

La première technique utilisée pour résoudre le problème des incohérences ou conflits est la structuration des groupes [DURF87]. Par exemple, si les responsabilités sont partagées entre les membres du groupe, chaque membre s'occupe de la partie qui lui a été confiée. On peut avoir un agent coordinateur qui supervise les autres agents. Dans une hiérarchie, les agents des niveaux supérieurs ont une vue plus complète du problème à résoudre et sont donc plus aptes à gérer les conflits que les agents des niveaux inférieurs qui n'ont que des vues très partielles des

problèmes. Une deuxième technique est l'ajout de «plans» à chaque agent pour le guider dans sa démarche [DURF87]. Cette idée se rapproche assez de celle des rôles attribués à chaque personne. On peut aussi ajouter des métarègles [DURF87] pour tester l'avancée des autres membres du groupe et pour se synchroniser avec eux. Il est également possible pour lever les conflits que les agents échangent entre eux des résultats partiels [LESS81], [DURF87], [DURF89], ou bien qu'ils échangent entre eux les données manquantes [INTO91]. Une autre approche [MALL91] consiste à accorder un certain degré de confiance aux données et résultats obtenus. Cela permet de quantifier la fiabilité des données et de raisonner avec des données non absolument fiables.

1.4.5. Attribution des tâches

L'attribution de tâches aux membres des groupes est une activité qui provient de l'automatisation des tâches de bureautique et de l'intelligence artificielle.

L'allocation des tâches à exécuter doit se faire en caractérisant leurs incertitudes vis-à-vis des autres tâches et des conséquences sur ces autres tâches. Il faut décomposer les tâches en sous-tâches. L'attribution de coûts ou de fonctions caractérisant les possibilités de réalisation, peuvent servir à gérer les degrés de conflits entre sous-tâches [ZLOT91]. Un modèle basé sur des réseaux de Petri est proposé pour spécifier, découper et vérifier des activités bureautiques dans le système DOMINO [VICT89]. Un autre formalisme, ressemblant également aux réseaux de Petri, permet de spécifier et de raffiner des tâches bureautiques [TUEN89].

Des comportements définissent l'évolution de l'agent dans le temps. Ces comportements, souvent définis au moyen de plans, permettent aussi de prendre en compte les buts que doivent atteindre les agents, les stratégies qu'ils doivent utiliser.

L'attribution de tâches peut se faire au moyen d'enchères. Le protocole «contract-net» [SMIT81] est basé sur ce principe. Un agent désirant faire exécuter une tâche lance une enchère vers les autres agents du système. Les agents aptes à réaliser la tâche répondent à l'enchère. La tâche est attribuée à celui qui offre les meilleures caractéristiques. L'agent qui reçoit une tâche à exécuter se voit responsabilisé de cette exécution en des termes de qualité de service, ou autre. Pour cela, il est nécessaire de définir les fonctions à remplir, le degré d'urgence de la tâche, le niveau d'interaction entre les agents. De plus, il faut connaître les critères et les expressions de satisfaction des buts et des tâches.

1.4.6. Mécanismes décisionnels

Les mécanismes décisionnels interviennent lorsque les utilisateurs d'un système coopératif doivent prendre une décision entre eux. Le champ des décisions à prendre peut bien entendu comprendre la gestion de la structure du groupe, l'attribution des rôles aux utilisateurs, l'attribution des tâches à réaliser ou bien la gestion des incohérences. Les agents utilisateurs détectent des incompatibilités qui peuvent se produire entre les données, les plans de chaque agent, les buts à atteindre. Ils sont également capables de négocier lors de conflits à résoudre, lors de prises de décisions [KLEI91], [MURA90], [MURA91].

Les mécanismes décisionnels sont un terme générique qui regroupe plusieurs possibilités. Ils peuvent être informels sous forme de discussions [HAHN89], ou bien peuvent faire appel à des mécanismes plus formels tels que les votes [GARC85], [HAHN89], [CHEU90], [AHAM91], [KUMA91], [NEIL92]. Une décision peut être laissée à la charge d'un arbitre [HEWI91]. Les différentes stratégies et les diverses heuristiques doivent être sélectionnées afin d'obtenir une solution au problème posé.

Lorsque deux tâches, par exemple, sont en conflit [ZLOT91], il faut négocier pour choisir d'exécuter une des tâches.

Pour résoudre des conflits ou pour faire face à des situations imprévues, les agents regroupés en organisations forment des forums [HEWI91]. Un forum correspond à une assemblée où les agents négocient et délibèrent entre eux. Un forum possède une politique propre pour prendre une décision, politique qui peut être un vote ou bien un arbitrage par un agent...

2. Paradigmes et modèles pour la coopération

Dans cette section vont être présentées les grandes classes de modèles et de paradigmes utilisés pour formaliser des groupes coopératifs ou des activités coopératives. Bien que la distinction paradigme-modèle ne soit pas précise, on rencontre surtout deux grandes écoles : l'école logique, tirée de l'intelligence artificielle, et l'école comportementale, d'où proviennent les langages.

2.1. Paradigmes

Cinq paradigmes de conception ont été rencontrés afin de définir à un haut niveau les systèmes coopératifs. Ce sont : les contrats, la cognition distribuée, la répartition et l'échange d'information, les actes de conversation, la théorie des organisations. A ce jour, aucun ne s'impose réellement.

1. Les *contrats* sont une forme de réalisation de tâches par un groupe coopératif. Le protocole «contract-net» [SMIT81] permet une négociation entre un site et un contractant. Lorsqu'un site doit réaliser une tâche qu'il ne peut accomplir, il lance une offre à l'ensemble des autres sites du groupe. Il collecte les réponses (ou enchères) obtenues. Il attribue alors la tâche au site qu'il juge le plus apte à la remplir. Ce protocole crée donc un lien entre un site et un contractant pour réaliser une tâche.

2. Les paradigmes de *cognition distribuée*, réalisée entre des individus ou des agents, comprennent les travaux tirés de l'intelligence artificielle distribuée. Ils montrent les effets d'un ensemble de mécanismes pour coordonner les actions d'un groupe d'agents. La répartition des informations entre les différents agents, la structuration des agents à qui sont confiées des zones d'intérêt, l'ajout de «planners», l'utilisation d'un méta-niveau de communication qui permet d'échanger des parties de plan en plus des informations [DURF87] aident à la résolution de problèmes en coopération. D'autres travaux [RODD89] sont basés sur le filtrage sémantique et cognitif de données. Les décisions sont prises d'après le contenu des messages reçus et d'après l'identité de l'agent qui a fourni l'information. Le système utilisant ce principe vise à faciliter les échanges de messages entre groupes de personnes. Les messages reçus sont également classés et archivés de manière automatique.

3. La *répartition et l'échange d'information* est aussi un élément important de la coopération. On peut rencontrer des échanges de messages entre agents, messages plus ou moins structurés, plus ou moins formels, des espaces partagés accessibles par un groupe ou un sous-groupe d'agents [HENN89], [SMIT89], des tableaux noirs [LESS80], [DURF87]...

4. Les systèmes basés sur les *conversations* (*«speech act»*) appliquent une approche linguistique pour les échanges entre agents [TUEN89], [KAPL92], [RODD92]. Les messages échangés sont semi-structurés. Ils sont également liés entre eux et forment ainsi une conversation. Cette conversation est reliée à un contexte qui permet l'interprétation des messages. Les éléments contenus dans chaque message peuvent servir à enrichir le contexte. Le système peut aussi analyser si un message ne possède pas des éléments non définis dans le contexte de la conversation, peut rechercher ces éléments ou bien le signaler aux agents en conversation [BIGN89].

5. La *théorie des organisations* [MALO90] tirée des travaux de la sociologie et de l'intelligence artificielle distribuée peut servir à organiser et à structurer les membres de groupes coopératifs [FOX81], [DURF89]. On peut par exemple avoir des organisations hiérarchiques dans lesquelles des agents surveillent et contrôlent le travail d'autre agents [LESS80], [DURF87], des organisations plus équitables...

2.2. Modèles

Les modèles sont utilisés pour préciser et éventuellement donner une signification exécutable à un paradigme. Les principales approches sont présentées ci-après.

2.2.1. Logique

La *logique modale* peut être utilisée pour modéliser des systèmes distribués ou des systèmes d'intelligence artificielle distribuée [MAZE91]. Les formules logiques permettent de représenter les propriétés globales du système et les propriétés locales de chaque agent. L'ensemble des propriétés que peut posséder un système se trouve ainsi défini par des formules logiques. La logique modale appliquée à la modélisation de protocoles de validation négociée (comme le protocole «contract net») et à des systèmes basés sur les tableaux noirs a permis de montrer que ces protocoles sont parfois irréalisables sous certaines hypothèses telles que les fautes ou les pertes de messages [MAZE91].

L'utilisation de la *logique du premier ordre* dans le système DARES [INTO91] permet de représenter les messages échangés. Ce système est un résolveur distribué de problèmes, ou «prouveur» distribué de théorèmes [INTO91]. Un problème est en fait un but à prouver représenté au moyen de formules logiques du premier ordre. Le problème est résolu lorsque les formules logiques qui le représentent forment un théorème. Les clauses à prouver sont réparties entre les agents qui travaillent de façon coopérative. Lorsqu'un agent se trouve bloqué et ne peut plus résoudre de clause, deux heuristiques peuvent s'appliquer. Elles visent toutes deux à fournir de la connaissance supplémentaire à l'agent qui se trouve bloqué.

La logique du premier ordre peut aussi servir à la vérification de propriétés [KREI89]. Un groupe de personnes communique via le courrier électronique. La gestion et la coordination de ces groupes est confiée à un agent, appelé médiateur, qui joue en quelque sorte le rôle d'un centralisateur. Le rôle et le comportement de ce médiateur est défini au moyen de règles semblables à celles de PROLOG. Pour vérifier l'exactitude et l'absence d'erreurs (statiques ou dynamiques), ces règles peuvent être transformées en un réseau de Petri prédicat/transition au moyen d'un simulateur [KREI89].

2.2.2. Réseaux de Petri

Les *réseaux de Petri* sont utilisés pour la planification de tâches bureautiques coopératives [VICT89]. Le réseau est simulé et vérifié. Les conflits sont détectés et signalés par le système. Les réseaux de Petri étendus servent très souvent à modéliser et analyser la circulation de documents («workflow») pour des tâches bureautiques au sein de groupes coopératifs [TOUZ94], [AGOS94a], [AALS94], [KELL94].

Les réseaux de Petri peuvent aussi être employés pour représenter les plans individuels de chacun des agents d'un groupe d'agents [MART90]. Ces agents évoluent dans un environnement commun, mais ne sont pas forcément au courant de tout ce que font les autres agents. Pour éviter des incompatibilités, des redondances entre les différents plans établis, il est nécessaire de les associer et de les coordonner. Le système fusionne les réseaux représentant chaque agent, analyse les différents plans, lève les contraintes, les incompatibilités et résout les problèmes afin d'obtenir un plan global synchronisé.

2.2.3. Acteurs

Le modèle *acteur* est aussi utilisé pour représenter des agents coopératifs [HEWI91]. Cependant, ce formalisme n'est pas suffisant pour représenter des groupes coopératifs. Les acteurs sont donc rassemblés dans des groupes d'acteurs appelées «Organisations of Restricted Generality» (ORG). Chaque acteur est placé dans un seul ORG. Les communications entre deux groupes se font via des acteurs spéciaux qui servent d'interface entre les groupes. Ainsi, un acteur d'un groupe peut communiquer avec des acteurs d'autres groupes. Lorsque des conflits surviennent, les acteurs forment des forums qui possèdent plusieurs politiques de négociation (votes, arbitrage par un chef...).

Un modèle proche du modèle d'acteurs permet également un autre représentation des agents de groupes coopératifs [MARU90]. Un agent est composé d'une file d'attente qui permet de stocker les messages reçus et d'un interpréteur de messages qui sélectionne les messages de la file suivant certains critères sémantiques et non suivant leur ordre d'apparition dans la file. Cet interpréteur retire les messages importants de la file et invoque les méthodes adéquates. De plus, les agents possèdent un comportement propre représenté sous la forme d'une machine à états finis. De ce fait, les agents sont appelés semi-autonomes car leurs actions ne sont pas seulement conditionnées par les messages reçus comme pour des acteurs classiques, mais aussi par leurs états internes. Afin de manipuler des ensembles d'agents, un modèle de groupe a été défini. Une organisation de grande taille est définie comme des ensembles de groupes d'agents. Un agent peut ainsi envoyer un même message vers un groupe d'agents.

2.2.4. Orienté objet

Le projet MacAll II [SMIT89] utilise un modèle *orienté objet* pour représenter un système coopératif. Sept composants sont identifiés : *les activités*, qui représentent les travaux à réaliser, *les membres*, qui représentent les individus travaillant en coopération ainsi que leurs capacités, *les rôles*, qui décrivent un ensemble de devoirs et de responsabilités, *les espaces de travail*, qui représentent les ressources communes associées à un rôle, *les messages échangés*, *les règles* pour contraindre le comportement des autres composants, et *les fonctions* manipulées par les rôles. Les relations entre ces différents éléments sont les suivantes : une activité regroupe un ensemble de personnes et de rôles. Chaque instance de rôle est reliée à un espace de travail associé au rôle, espace dans lequel la personne possédant le rôle échange des messages. Les règles et les fonctions sont associées à un rôle pour la réalisation de l'activité.

Un autre modèle orienté objet est proposé pour représenter des groupes manipulant des informations de façon asynchrone [BENF93]. Les groupes forment des domaines d'application qui contiennent des informations et des personnes. Toutes les entités manipulées dans un groupe (items, personnes, domaines...) sont représentées sous la forme de classes qui dérivent de la classe «group communication», pour un groupe particulier. Un mécanisme global qui s'inspire de la norme X.500 [ISO9594] permet de nommer les divers objets. Un ensemble de primitives de service sert à la manipulation d'objets instanciés et à la création de liens entre ces objets.

2.2.5. Modèle procédural

Un modèle est dit *procédural* lorsque la communication est décrite en terme de passage de messages entre des rôles, suivant un ensemble de procédures représentant un cadre spécifique de communication [HENN92].

Le modèle utilisé dans le projet COSMOS [DOLL89] comprend des activités, simples ou évoluées, qui doivent être exécutées de façon coopérative. Ces activités possèdent des rôles attribués aux différents membres du groupe. Les membres échangent entre eux des messages interprétés au moyen de scripts événementiels.

Le modèle d'activité AMIGO [PRIN89] comprend quatre grandes composantes : les rôles, les messages échangés, les fonctions et les règles. Les messages sont les informations échangées entre les différents membres du groupe coopératif. Les rôles donnent une description abstraite des caractéristiques de chaque membre. Les fonctions décrivent les opérations que les rôles réalisent. Les règles fournissent les synchronisations des échanges de messages entre les rôles et les conditions d'utilisation des fonctions.

2.2.6. Partage d'information

Certains systèmes s'appuient sur un modèle de *partage d'information* [BENF92] ou de *partage de documents* [FUKS94] plutôt que sur un modèle procédural d'échange d'informations par envoi de messages. Dans le projet Grace [BENF92], la communication est représentée par l'accès à des informations structurées à l'intérieur de domaines qui définissent des espaces spécifiques d'activités. On n'a plus d'envoi de messages vers un groupe de gens car on se contente de rendre un document accessible et visible au groupe. Les objets manipulés peuvent être simples ou complexes. Dans ce cas, ils contiennent d'autres objets regroupés en «clusters», suivant un modèle orienté objet.

3. Classification des applications coopératives

Cette section présente les grands groupes d'applications coopératives rencontrées dans la littérature. Dans chaque partie, quelques applications significatives et illustratives sont développées. Plusieurs bibliographies répertoriant toutes les applications sont aussi parues sur le CSCW [MALM93] et sur l'intelligence artificielle distribuée [CHAI92], [BOND92] et quelques classifications incluant également les applications multimédias [WILL94].

3.1. Systèmes distribués multi-utilisateurs

Les systèmes distribués offrent un ensemble d'outils généraux utilisables au sein d'un groupe.

3.1.1. Editeurs multi-utilisateurs

Un premier type d'application tiré des systèmes distribués regroupe les éditeurs multi-utilisateurs. Dans ce type d'application, un groupe d'utilisateurs manipule et modifie un même texte.

GROVE

L'outil de collecticiel GROVE («GRoup Outline Viewing Editor») [ELLI91] est un éditeur multi-utilisateur pour un groupe synchrone. Ses principales caractéristiques font que les ensembles d'items manipulés dans un texte peuvent être privés, partagés ou publics. Les entrées et sorties du groupe d'édition se font n'importe quand. On n'a pas de verrouillage sur les données : les conflits sont alors gérés par les utilisateurs. Lors d'une modification, le nouveau texte apparaît coloré puis, au fur et à mesure de son intégration chez les autres membres du groupe, il devient de nouveau de couleur habituelle.

DistEdit

[KNIS90] présente l'éditeur synchrone DistEdit qui permet de partager un fichier entre plusieurs utilisateurs et de maintenir la vue de ce fichier consistante chez tous les utilisateurs. Les utilisateurs peuvent joindre ou quitter une session d'édition à tout moment. Un seul utilisateur peut modifier le fichier à la fois. Les autres ne sont que de simples observateurs des modifications.

MESSIE

L'édition de texte en groupe peut aussi se faire de manière asynchrone, comme dans l'environnement MESSIE [SASS93]. Le problème est alors de gérer la divergence des copies et de fusionner les contributions de chacun des utilisateurs. L'édition peut être aussi envisagée de manière semi-synchrone [MINO93], avec un graphe des versions et des parties fusionnées.

3.1.2. Espaces partagés

MBLINK

L'outil MBLINK [SARI85] est un espace «bitmap» partagé. Il sert à reproduire le contenu d'une fenêtre «bitmap» chez un ensemble d'utilisateurs. Chaque membre du groupe possède une fenêtre «bitmap» et un pointeur personnel. Les pointeurs de tous les membres sont affichés sur la fenêtre «bitmap» de chacun des utilisateurs. Le pointeur d'un présentateur permet d'attirer l'attention des autres membres du groupe sur une partie de la fenêtre.

3.1.3. Systèmes de partage d'applications

Cette catégorie comprend un ensemble de plate-formes qui permettent de rendre multi-utilisateur une application initiale mono-utilisateur.

Conference Toolkit

Le système «Conference Toolkit» [BONF89] est un système décomposé en couches logicielles pour bâtir des applications multimédias pour des groupes d'utilisateurs. Chaque utilisateur se sert de X Window pour l'affichage sur son écran (présentation). Entre l'application multi-utilisateur (élément de traitement) et X Window (élément de présentation) a été ajoutée la couche «composant de conférence» pour le contrôle de la distribution. Cette couche peut être contrôlée soit par l'application (mode esclave), soit par les utilisateurs (mode maître). Les principaux composants de la couche «composant de conférence» sont les commutateurs qui servent à multiplexer ou à démultiplexer des flux de données entre les personnes connectées. Par la suite, les éléments de cette couche sont détaillés :

- les commutateurs sont spécialisés dans la gestion de flux de données spécifiques (voix par exemple). Dans un commutateur, un flux de donnée est associé à chaque utilisateur. Un autre flux est associé à l'application. Chaque utilisateur et l'application peuvent recevoir des données des autres utilisateurs ou de l'application (notion de matrice de commutation). Des méthodes sont déclenchées lors de chaque multiplexage de donnée.
- Les commutateurs sont regroupés dans des ponts.
- Les ponts sont à leur tour regroupés dans des gestionnaires de conférence qui contrôlent l'ensemble de la couche «composant de conférence».

Cette plate-forme a permis la réalisation d'un prototype, le «bureau de conférence», qui permet de partager toutes les applications mono-utilisateurs X et NeWs. A chaque instant, la liste des gens connectés est affichée. Le contrôle de l'application partagée est fait à la demande.

MERMAID

Le système MERMAID («Multimedia Environment for Remote Multiple Attendee Interactive Decision-making») est aussi une plate-forme de collecticiel pour partager des applications prévues à l'origine pour un seul utilisateur [OHMO92a], [OHMO92b]. Un module spécifique capte les événements provenant ou arrivant à une application, et les rediffuse vers d'autres destinataires chez lesquels s'exécutent des copies de l'application. L'architecture est distribuée, et les événements sont ordonnés globalement. On a aussi une politique de gestion des événe-

ments. A un instant, seul un utilisateur peut envoyer des commandes à l'application. Ce droit de contrôler l'application à tour de rôle est négocié entre les utilisateurs.

Rendez-Vous

D'autres systèmes comme «Rendez-Vous» [PATT90], visent à faciliter la réalisation d'applications multi-utilisateurs, et permettent aux utilisateurs d'avoir des vues différentes des objets mis en commun. Trois dimensions de partage des objets sont définies : le partage proprement dit des objets, le partage des vues des objets, le partage des accès aux objets. L'architecture des applications est cependant centralisée. Ce système propose aussi une aide pour le déroulement des sessions multi-utilisateurs.

3.1.4. Autres applications

D'autres travaux peuvent être notés tels que PREP [NEUW90], un autre éditeur multi-utilisateur permettant d'argumenter et d'annoter, SEPIA [HAAK92] un éditeur hypertexte multi-utilisateur, CoMEdiA [SANT93] [SANT94] une plate-forme offrant un ensemble d'éditeurs graphiques et textuels multi-utilisateurs, GroupKit [ROSE92] une autre plate-forme pour la réalisation d'applications multi-utilisateurs, GroupDesign [KARS93] un éditeur graphique multi-utilisateur...

3.2. Systèmes support de décision de groupe

Ces systèmes permettent d'assister un groupe d'utilisateurs pour des discussions, et facilitent l'obtention d'un consensus, soit sur des sujets généraux, soit dans un domaine spécifique (création de document, génie logiciel [JOHN93]) supporté par le système.

Souvent, les systèmes support de décision de groupe offrent deux fonctionnalités. En effet, un système coopératif pour assister une décision de groupe sur un objet, un document est le plus souvent précédé d'une phase de création de cet objet ou document intégrée dans la même application. Cette phase de création en groupe est souvent réalisée avec l'aide des outils et applications provenant de la classification des systèmes distribués multi-utilisateurs.

3.2.1. Création commune de documents

COLAB

La plate-forme expérimentale de conférences COLAB, développée au Xerox PARC, permet à des groupes synchones de créer des documents en commun, et d'évaluer le résultat produit [STEF87], ou bien d'annoter des propositions.

Le premier outil, appelé COGNOTER, sert à écrire de façon coopérative des articles, des présentations... Cette création se déroule en trois phases qui ne sont pas séparées et qui permettent de faire des retours en arrière.

- La phase 1 est celle du «brainstorming». Les idées sont générées, puis introduites dans le système.
- La phase 2 est celle de l'organisation. Les idées sont classées et reliées entre elles. Les idées qui font partie des mêmes concepts sont regroupées.
- La phase 3 est celle de l'évaluation. Le résultat obtenu est analysé. Si nécessaire, ce résultat est réorganisé en ajoutant, en enlevant certaines parties ou en déplaçant d'autres éléments.

Le deuxième outil s'appelle ARGNOTER. Il sert à présenter et à évaluer toujours de façon coopérative des propositions. Le déroulement se fait également en trois phases :

- La phase 1 est celle où l'on donne les propositions au système.
- La phase 2 comporte l'argumentation des propositions. Les membres du groupe donnent leurs opinions sur ces propositions et indiquent notamment s'ils désirent accepter ou rejeter ces propositions.
- La phase 3 est celle de l'évaluation du résultat. Les propositions qui ont obtenu une forte approbation sont conservées.

CoAUTHOR

Le système CoAUTHOR est une autre plate-forme qui permet de produire de façon coopérative des documents hypermédias [HAHN89].

La création des documents hypermédias se déroule en trois étapes :

- La première phase est celle du traitement des idées. Il s'agit du «brainstorming» des membres du groupe. Chaque membre génère les idées qu'il veut insérer dans le document hypermédia. Ces idées sont raffinées de manière hiérarchique.
- La phase suivante est celle de la génération de la structure formelle du document (conception du document). Les relations conceptuelles entre les idées sont prises en compte. Les parties du document sont regroupées de façon logique, d'autres sont séquencées en suivant les dépendances thématiques du document. Cette phase aboutit à la définition des liens hypertextes et du schéma de navigation dans le document. Finalement, les idées de la première phase sont reliées à la structure du document.
- La dernière phase effectue la genèse du document en reliant ce dernier aux médias appropriés. Le document est rempli à l'aide des textes, formules, graphiques, images... nécessaires. Les responsabilités de création des parties de documents sont affectées au moyen d'enchères aux différents membres du groupe.

Pour chaque phase de création du document, le groupe intervient en suivant la procédure qui suit. Cette procédure comporte également trois étapes :

- Tout d'abord, chaque membre du groupe entre les objets ou les parties qu'il veut insérer dans le document.
- Chaque membre annote le résultat produit au moyen de commentaires, de critiques, de jugements... Ces annotations sont typées «PRO» ou «CONTRA» suivant l'acceptation ou le refus d'un certain découpage, suivant la pertinence de certaines idées... L'examen des contributions individuelles permet de sélectionner les passages préférés, de juger la structuration proposée, de résoudre les conflits, les oublis, les hors-sujet. Le système analyse ensuite l'ensemble des annotations. Toutes les parties marquées à la fois «PRO» et «CONTRA» sont inconsistantes et doivent donc être jugées par le groupe.
- La troisième étape est celle des prises de décision sur les parties en conflit. Les utilisateurs peuvent choisir une négociation informelle menée au travers d'une conférence. Mais, s'ils le désirent, ils peuvent aussi se servir d'un type de négociation plus formel, à savoir un vote. Ce vote donne lieu à un score qui indique si l'objet organisé tel quel est accepté ou refusé dans le document.

Le système CoAUTHOR est organisé suivant trois couches logicielles. Le noyau est formé d'une base de données, appelée «ConceptBase» pour stocker les idées, les structures des documents, les relations entre les idées et les structures de documents et d'une base de données multimédias, appelée «MULTOS» pour stocker tous les objets multimédias. La deuxième couche se compose de deux parties : la première, appelée «Interaction Toolbox», comporte tous les éditeurs pour manipuler les objets multimédias, la deuxième, appelée «Group Toolbox» sert plutôt à gérer le groupe et contient les outils pour argumenter, voter... La dernière couche est une conférence qui gère les interactions entre les différents utilisateurs.

3.2.2. Génie logiciel

ICICLE

Le système ICICLE («Intelligent Code Inspection Environment in a C Language Environment») [BROTH90] est un outil support de décision de groupe pour l'inspection de code. L'inspection de code est une tâche très codifiée. Elle consiste à vérifier le code avant de commencer les tests. Elle met en oeuvre un lecteur du code, un ou plusieurs inspecteurs et un rapporteur qui note les remarques et changements à apporter au code. La phase d'annotation, de discussion pour garder et rejeter les remarques a été entièrement assistée par ordinateur.

CoNeX

CoNeX («Coordination and Negociation support for eXperts in design applications») [HAHN91] est un environnement expérimental qui offre un support coopératif dans le cadre de l'ingénierie du logiciel. Une base de données sert à stocker les éléments nécessaires au développement du système. Dans cette base, le logiciel est représenté suivant trois couches : la première définit le besoin des applications, la deuxième donne la conception des systèmes, la troisième donne l'implantation des sous-systèmes. Le premier outil proposé est un éditeur d'arguments pour assister les négociations entre le concepteur et les analystes. Le deuxième outil est un éditeur de contrat qui permet aux analystes d'attribuer les tâches à programmer et de responsabiliser les programmeurs. Le dernier outil est un système de conférence pour l'échange de messages informels.

3.2.3. Calendriers

Les systèmes de calendrier multi-utilisateurs permettent à un ensemble d'agents coopérants de négocier une date de rendez-vous commune.

Distributed Shedule-Board Agents

[MARU90] propose un modèle d'agents pour représenter les entités d'un groupe coopératif. Un agent est composé d'une file d'attente qui permet de stocker les messages reçus et d'un interpréteur de messages qui sélectionne les messages reçus suivant leur contenu et non suivant leur classement dans la file. Les agents possèdent également un comportement propre décrit par une machine à états finis. Ces agents sont aussi organisés en groupes structurés. Comme exemple d'application du modèle, un système pour trouver une date pour une réunion est proposé. Les personnes voulant se réunir sont représentées par un ensemble d'agents possédant chacun un calendrier personnel. Cet ensemble forme un groupe. Un autre ensemble d'agents représente les salles disponibles pour cette réunion. Les agents dialoguent entre eux de façon coopérative pour arriver à un consensus sur la date de réunion et dialoguent avec le groupe représentant les salles de réunion pour savoir dans quelle salle se réunir.

Visual Scheduler

[BEAR90] présente un calendrier visuel, bâti au-dessus du langage «smalltalk» qui permet à chacun des utilisateurs de décrire son emploi du temps. Chaque personne annote les intervalles de temps durant lesquels elle est occupée et choisit des couleurs pour définir la priorité des rendez-vous de son calendrier personnel. Cette priorité indique la difficulté de l'utilisateur pour déplacer son rendez-vous. Pour réaliser une réunion, le système fusionne les calendriers pour voir s'il reste des créneaux horaires libres. Sinon, suivant la couleur obtenue par fusion, les personnes peuvent modifier leur calendrier propre pour l'adapter. Il est aussi possible d'attacher des informations publiques ou privées que chaque utilisateur peut mettre sur ses rendez-vous.

3.2.4. Système d'aide général

SYBIL

Le système SYBIL est un système pour supporter des décisions générales de groupe [LEE90]. Le modèle utilisé est composé de buts qui peuvent être raffinés en sous-buts par les décideurs. Les décideurs proposent ensuite des alternatives qui sont similaires à des hypothèses. On obtient ensuite une matrice de décision entre les buts et les alternatives. A l'aide de demandes, les utilisateurs évaluent l'aptitude que les alternatives ont à résoudre les buts. Le système évalue alors les résultats. On peut aussi attribuer des critères d'importance aux buts.

3.2.5. Autres applications

[STEE81] décrit un système d'aide à la décision pour construire des arbres décisionnels lors de situations complexes, [TWID93] détaille DNP («Design Notepad») un environnement pour la conception coopérative de logiciel, [MELO94] présente Tempo, un environnement de génie logiciel coopératif...

3.3. Messageries «context-sensitive»

Les messageries «context-sensitive» proviennent des travaux visant à fournir des outils d'assistance des procédures de bureautique et d'automatisation des tâches bureautiques. Ces systèmes ont alors évolué en offrant des fonctionnalités supplémentaires au courrier électronique.

CHAOS

Le système CHAOS («Commitments Handling Active Office System») [BIGN89] classifie, répertorie, et archive des connaissances pour supporter des communications humaines. Les messages manipulés sont semi-structurés de manière à ce que le système puisse les interpréter. Dans un groupe, la connaissance commune est constituée par l'évolution du groupe et par son historique. Cette connaissance commune permet par exemple d'informer les nouveaux arrivants. Par contre, la consistance des informations est perturbée par l'existence de sous-groupes ou par des sites appartenant à plusieurs groupes. En effet, les informations manipulées dans un groupe peuvent diverger par rapport à celles d'un autre groupe et laisser apparaître des incohérences, des ambiguïtés. En fait, les communications à l'intérieur d'un groupe augmentent la cohérence des données, les communications dans des sous-groupes diminuent la cohérence.

Le système est formé de trois éléments :

- Un réseau de communication.
- Un agenda des activités et des devoirs réalisés au niveau utilisateur et au niveau des groupes.

- Des liens organisationnels entre les membres du groupe et des liens organisationnels entre les membres du groupe et les activités en terme de responsabilités et d'expérience.

Le système est pleinement adaptatif dans le sens où les messages échangés influencent la structure du groupe, la connaissance du groupe, les règles linguistiques, et que ces dernières influent sur les messages échangés.

Le système est capable de détecter les inconsistances ou les références inconnues des messages, de stocker et d'associer un message à une conversation.

La base de connaissances est découpée en quatre entités :

- Le «Group Langage Expert» qui comporte la connaissance du groupe, la connaissance de chaque membre du groupe, et la connaissance que chaque membre a des autres membres du groupe.
- Le «Group Agenda Expert» qui contient les acquittements courants pour une conversation.
- Le «Group Structure Expert» qui possède la connaissance de la structure du groupe en terme de rôle et d'expérience.
- Le «Conversation Handler» qui garde les traces des conversations.

Les messages d'une conversation sont analysés. On associe à chaque message un centre d'intérêt et on regroupe les centres d'intérêt en sujets. Pour vérifier le contenu des messages, le système regarde si les références des messages sont connues par le récepteur. Les références inconnues sont cherchées dans les centres d'intérêt du destinataire, puis en cas d'absence chez l'émetteur.

La nouvelle connaissance peut rester locale à une conversation ou bien être ajoutée à la base de connaissances pour des utilisations futures. Cet ajout risque de modifier l'agenda et l'expérience de l'utilisateur.

Mailtrays

Le système présenté par [RODD89] permet de classifier de manière automatique les messages qui arrivent chez un utilisateur. Les messages échangés sont représentés comme des objets contenant des attributs, les champs des messages, et des méthodes, les boutons des messages. Cette représentation par objets permet d'utiliser des possibilités d'héritage et facilite le regroupement et le classement des messages. Les messages sont donc analysés suivant leurs attributs et suivant les valeurs des attributs.

Cette analyse permet de classer les messages dans des casiers («mailtrays»). Chaque casier possède donc une liste de gardes ou liste de prédicats. Ces listes permettent de tester les valeurs des attributs des messages et, suivant ces valeurs, de sélectionner ou de rejeter les messages. Lorsqu'un nouveau message arrive dans un casier, une série d'actions est exécutée. Ces actions sélectionnent des listes de messages contenus dans le casier. Suivant le résultat de ces sélections, d'autres actions sont réalisées.

Pour manipuler les informations de sortie et pour envoyer des messages, on a deux possibilités :

- La première est d'envoyer un message en adressant les destinataires par rôle et par qualification. Ces destinations abstraites sont gérées par un registre qui fournit les destinations réelles.

- La deuxième possibilité est de définir des chemins explicites entre les casiers. En effet, pour certains types de conversation comme un vote par exemple, les types de messages échangés sont connus à priori, et les liens entre les casiers aussi. Un éditeur permet ainsi de définir les liens entre les casiers. Ces liens donnent le type des messages échangés ainsi que le casier émetteur et le casier destinataire du message. De la même façon que pour l'arrivée d'un message dans un casier, des sélections et des actions peuvent être exécutées lors de la sortie d'un message d'un casier.

GRACE

Le projet GRACE («GRoup Activity Environment») [BENF91], [BENF92] est un projet de recherche sur deux ans pour développer des communications asynchrones de groupe dans un environnement OSI («Open System Interconnection»). Ce projet a permis de développer un modèle conceptuel de groupes asynchrones dont les membres partagent des informations, et de définir un service de communications de groupe pouvant être utilisés par un vaste ensemble d'applications coopératives. Ce service s'appuie sur la messagerie électronique X.400 [ISO10021] pour les communications entre membres des groupes. Les informations manipulées sont structurées de façon hiérarchique, et sont regroupées en domaines qui permettent également de relier les membres des groupes aux différentes données manipulées. Les objets du système sont nommés au moyen du répertoire X.500 [ISO9594]. La gestion et la sécurité ont aussi été abordées dans le cadre des modèles et architectures proposées par l'OSI.

Ces travaux ont permis la réalisation de deux applications prototype. La première est un service d'aide («Help Desk») pour un département informatisé. Les personnes connectées au système émettent des requêtes lorsqu'elles rencontrent des problèmes. Le système trie et aiguille les questions suivant les compétences des gens. Les personnes ayant un rôle de spécialiste reçoivent alors les requêtes, et peuvent formuler des réponses. La deuxième application proposée est une conférence asynchrone, équivalente à un tableau de bulletins, entre groupes d'utilisateurs. Chaque domaine forme un forum portant sur un sujet précis, le contrôle des informations échangées étant fait par un membre modérateur. Les personnes connectées au système peuvent ainsi joindre ou quitter des domaines suivant leurs centres d'intérêt.

Autres applications

D'autres projets comme COSMOS [DOLL89], [HENN89], AMIGO [HENN89], SuperKOM [PALM92], et UTUCS [AGOS94a], [AGOS94b] offrent également un support à des groupes d'utilisateurs au dessus d'un système de messagerie. ATOMICMAIL [BORE92] est une variante qui permet le partage de programmes exécutables transmis au moyen d'une messagerie...

3.4. Intelligence artificielle distribuée

Cette section vise à définir les applications coopératives qui proviennent des travaux de l'intelligence artificielle distribuée et des systèmes multi-agents, et à montrer les domaines traités, les problèmes abordés et les solutions proposées.

3.4.1. Présentation

L'article de [DURF89] fait l'état de l'art en matière de résolveurs de problèmes distribués coopératifs (CDPS). Il présente en particulier les principaux groupes d'applications développées et les grandes directions de recherche de l'intelligence artificielle distribuée.

Les grands groupes d'applications des CDPS sont :

- *L'interprétation distribuée* : dans cette catégorie sont classés les réseaux de capteurs, les diagnostics de fautes pour des réseaux de communication...
- *Le planning et le contrôle distribué* : notamment, les applications sont le contrôle aérien distribué, les robots coopérants, les véhicules pilotés à distance, le contrôle distribué de processus dans la production...
- *Les systèmes experts coopératifs.*
- *La coopération humaine supportée par ordinateur* : ces applications servent à focaliser l'attention, à filtrer le trop-plein d'information. Parmi elles sont classées la commande et le contrôle de systèmes, la coordination de projets multi-utilisateurs...
- *Les plate-formes pour appliquer et tester les modèles cognitifs.*

Le plus gros problème rencontré pour les CDPS est que les agents doivent prendre des décisions sur leur problème, réaliser des actions de communication avec des vues locales qui peuvent être incomplètes, inconsistantes, ou qui ne sont plus à jour. Les communications doivent servir à améliorer leurs connaissances mais aussi leurs actions et leurs interactions [YOKO92].

Les approches pour pallier au problème précédent sont :

- *Les négociations* : on dialogue pour résoudre les vues inconsistantes. On peut avoir des contrats entre le propriétaire d'une tâche et l'exécuteur de cette dernière, une planification, des modèles cognitifs de négociation comme l'obtention de compromis en relâchant des hypothèses...
- *La «Functionally-Accurate cooperation»* : il s'agit de l'échange de résultats partiels pour résoudre les inconsistances.
- *La structure organisationnelle* : on spécifie les rôles, les pouvoirs, les capacités de chaque agent.
- *Le planning multi-agents* : les agents spécifient leurs actions et leurs interactions futures. Pour lever les conflits, les plannings peuvent être collectés de façon centralisée, analysés pour détecter les conflits, et redistribués aux agents. On peut aussi lever les conflits de manière distribuée en demandant aux autres agents les ressources nécessaires pour affiner les buts.
- *Les contrôles locaux sophistiqués* : le raisonnement au sujet des actions et croyances des autres agents sur le problème est intégré au niveau de chaque agent. Chaque agent raisonne sur l'influence des communications qu'il effectue. Les décisions de coordination avec les autres agents font aussi partie des décisions locales et non d'une couche concernant le problème à résoudre. Chaque agent analyse la pertinence des données fournies et reçues.

3.4.2. Applications

Par la suite, plusieurs applications caractéristiques de l'intelligence artificielle distribuée sont présentées.

DVMT

Le système DVMT («Distributed Vehicle Monitoring Testbed») [DURF87] est composé d'un réseau d'agents distribués qui doivent recomposer la trace d'un véhicule. Chaque agent est associé à un capteur qui reçoit le signal de trace. Il s'agit d'un réseau de capteurs distribués. Chaque capteur ne voit qu'une partie de la zone de déplacement du véhicule. Les agents sont organisés suivant une architecture tableau noir avec des sources de connaissance (SC). Les résultats

obtenus sont comparés avec ceux d'une architecture centralisée dans laquelle un seul agent posséderait les deux capteurs.

- Première stratégie : chaque agent est responsable d'un capteur. Il élabore une solution partielle et échange le résultat partiel obtenu avec les autres agents.
- Deuxième stratégie : en plus de la première stratégie, les agents sont structurés de façon organisationnelle. L'espace de recherche de chaque noeud est spécifié, ainsi que ses responsabilités et sa compétence.
- Troisième stratégie : à chaque agent est rajouté, en plus des deux stratégies précédentes, un planificateur qui spécifie le rôle d'un agent suivant un certain intervalle de temps, et qui indique la stratégie à suivre dans le futur en analysant les buts de plus haut niveau à atteindre, en formulant des hypothèses de plus haut niveau.
- Quatrième stratégie : en plus des stratégies précédentes est rajouté à chaque agent un méta-niveau de communication. Cela permet d'échanger des plans pour que chaque agent se synchronise avec l'évolution des autres agents pour ne pas effectuer de tâches redondantes ou divergentes.

Les résultats de ces expériences montrent une amélioration des performances pour recomposer la trace initiale lors de l'application d'une nouvelle stratégie en partant de la première jusqu'à la quatrième, surtout lorsque les capteurs sont perturbés par des échos parasites. D'autres essais ont été faits en augmentant le nombre d'agents, en augmentant ou en diminuant le recouvrement des zones gérées par chaque agent. Ces essais montrent que le système recompose d'autant mieux la trace du véhicule et plus rapidement lorsqu'il est composé d'un nombre limité d'agents (une dizaine) hautement sophistiqués et faiblement couplés entre eux plutôt que d'un grand nombre d'agents relativement simples et fortement couplés entre eux.

DARES

Le système DARES («Distributed Automated REasoning System») [INTO91] est un résolveur distribué de problèmes, ou «prouveur» distribué de théorèmes. Un problème, qui est en fait un but à prouver, est représenté par des formules logiques du premier ordre. Résoudre le problème est donc, dans ce cas, équivalent à prouver que les formules logiques qui le représentent sont un théorème. Les clauses à prouver sont réparties entre les agents qui travaillent de façon coopérative à résoudre et à valider le but. Lorsqu'un agent se trouve bloqué et ne peut plus résoudre de clause, deux heuristiques s'appliquent. La première regarde s'il reste des nouveaux résolvants. Si la réponse est non, la clause la plus probable à résoudre (suivant la première heuristique) est diffusée vers tous les autres agents. L'agent émetteur attend la réponse d'un autre agent qui envoie des connaissances pour faire avancer la résolution. Si aucune connaissance ne simplifie cette clause, l'agent sélectionne une autre clause et la transmet comme précédemment aux autres agents, et ainsi de suite. Une deuxième heuristique chez chaque agent vérifie si la résolution avance. Dans la négative, cet agent demande de la connaissance aux autres agents.

Les mesures effectuées montrent que la distribution augmente fortement les performances du système. Les meilleurs résultats sont obtenus lorsque le nombre d'agents est assez important et que ces agents ont très peu ou pas d'information redondante entre eux. Ils se comportent alors comme des spécialistes.

Plans multi-agents

Le système décrit [MART90] permet de coordonner un ensemble de plans établis par un groupe d'agents. Ces agents évoluent dans un environnement commun. Chaque agent établit un plan individuel, représenté au moyen de réseaux de Petri. Mais, il faut coordonner ces plans pour éviter des incompatibilités, des redondances... Un classement des diverses relations entre deux plans est présenté :

a) Relations négatives : il s'agit d'actions conflictuelles avec celles d'autres plans.

- Les conflits peuvent provenir de l'accès limité à certaines ressources.
- Ils peuvent venir d'incompatibilités entre des états qui s'excluent mutuellement.

b) Relations positives : il s'agit d'actions qui peuvent être bénéfiques à la réalisation d'actions d'autres plans.

- Requêtes : un agent demande le concours d'un autre agent, soit pour réaliser un travail en commun («multiagent planning»), soit pour lui donner un travail à réaliser.
- Pas de requête vers un autre agent.

Egalité : des actions sont identiques et remplissent le même but. Dans ce cas, un seul agent réalise ces actions.

Conséquence : une action dans un plan nécessite l'accomplissement d'un but d'un autre plan.

Faveur : une action réalisée dans un plan serait potentiellement profitable à la réalisation d'un but d'un autre plan. Un agent modifie quelque peu son plan pour favoriser la réalisation d'un but d'un autre agent.

Le système analyse ces contraintes, cherche à profiter des relations positives, élimine les relations négatives et fournit des solutions. Les agents négocient entre eux ces solutions. A la fin, on obtient un plan global, dans lequel la majorité des conflits ont été levés.

Planification pour des véhicules

Ce travail [FRAI90] présente une méthode pour planifier le déplacement de véhicules dans un monde dynamique. Un ensemble de véhicules se déplace dans un monde composé de routes et d'intersections. Le système ne connaît pas les buts de tous les véhicules. Il va donc coordonner les mouvements des véhicules dont il connaît les buts et qui sont sous sa responsabilité, tout en évitant les collisions avec les véhicules qu'il ne contrôle pas. Les véhicules sous contrôle doivent également atteindre leur but le plus rapidement possible tout en évitant les collisions.

Ce système applique une décomposition chemin/vitesse. L'environnement est d'abord modélisé suivant un graphe. Un premier planificateur, appelé planificateur géométrique, détermine les chemins que les véhicules sous contrôle doivent emprunter. Un deuxième planificateur, appelé planificateur temporel, calcule les vitesses des véhicules. Comme le système ne maîtrise pas les comportements des véhicules dont il ne connaît pas le but, le planificateur temporel effectue des hypothèses et est obligé de recalculer périodiquement les vitesses des véhicules en prenant en compte les nouvelles positions des véhicules non contrôlés. Le principe du fonctionnement du planificateur temporel est le suivant :

- Le planificateur effectue une estimation à partir de leurs positions et de leurs vitesses des buts des véhicules non contrôlés.
- Ensuite, comme tous les véhicules sont sensés avoir les même capacités, il applique à tous le même algorithme.

- Le planificateur affecte des priorités à tous les véhicules. Cela permet d'obtenir un ordonnancement total de ces derniers.
- Il planifie le mouvement de chacun des véhicules en les prenant un par un suivant l'ordre établi.

Autres applications

[CONR91] présente un résolveur distribué traitant des problèmes de routage dans un réseau, [PAN91] décrit le système MKS composé d'agents logiciels «intelligents» et de personnes pour la réalisation de tâches d'entreprise...

3.5. Conférences

Ce paragraphe décrit un ensemble d'applications de conférences ou de visioconférences. Une classification des besoins des conférences a été effectuée par [ROBI91], [GALE92].

Rapport

Le système Rapport [AHUJ88] permet de réaliser des conférences multimédias. Le concept utilisé dans ce système est celui de «salle de conférence virtuelle». A un instant, un membre du système ouvre une conférence virtuelle et invite d'autres membres, au moyen d'un signal sonore, à le rejoindre dans cette conférence. Une personne peut donc appartenir à plusieurs conférences, mais, à chaque instant, elle ne peut se trouver que dans une et une seule conférence virtuelle et elle n'accède qu'aux données de cette conférence virtuelle.

Les caractéristiques d'une conférence sont les suivantes :

- L'affichage se fait de façon identique chez tous les participants. Il s'agit du concept d'écran partagé.
- Tout membre d'une conférence peut lancer des applications. Il possède alors le contrôle de ces dernières et l'affichage des sorties des applications se fait de façon identique chez tous les membres de la conférence. La personne qui possède le contrôle d'une application détermine aussi les caractéristiques et la position de l'affichage de cette application dans l'espace partagé.
- Chaque conférencier possède un pointeur unique qui lui sert à attirer l'attention des autres conférenciers sur des parties de l'espace commun affiché.
- Chaque conférencier a aussi la possibilité de lever des signaux pour attirer l'attention des autres conférenciers. Par exemple, ceci correspondrait à lever la main dans une conférence face à face.
- Il est aussi possible de communiquer grâce à un canal audio avec un autre membre de la conférence.
- Chaque conférencier est représenté sur l'écran.

Ce système a été développé en utilisant un ensemble de stations SUN reliées entre elles par un réseau Ethernet. Le service de communication est du type «sockets» et l'affichage se fait avec X Window.

MICA

Le système MICA («Multimedia Interactive Conferencing Application») [ROBI91] est une application pour des téléconférences multipoints. Il consiste en la création d'un espace partagé pour visualiser des documents photographiques, du texte, des images statiques, des dessins...

Les principaux besoins des applications de téléconférence identifiés sont :

- Pas de limite au nombre de participants.
- Nécessité de protocoles sociaux pour pouvoir gérer un grand nombre de personnes.
- L'ensemble des médias visuels utilisés pour les conférences doit pouvoir être partagé.
- Il est nécessaire de pouvoir modifier, annoter ou imprimer des documents visualisés.
- Il doit être possible de délivrer des documents formels, classés et organisés, comme il doit être possible d'ajouter des documents hors séquence à la demande.

Ce système offre un espace de visualisation partagé qui est identique chez chaque utilisateur. Tout document graphique, textuel, ou photographique peut être partagé. Chaque utilisateur possède également un stylet pour pouvoir annoter l'espace partagé. Le contrôle de l'application se fait de manière simultanée par l'ensemble des utilisateurs.

Les commandes accessibles par chaque utilisateur sont représentées sous la forme d'icônes. De la même façon, les objets ou documents manipulés sont représentables sous la forme d'icônes, miniatures exactes des documents qu'ils représentent. Chaque utilisateur est représenté par sa photo, qui peut aussi être mise sous forme d'icône. Les documents sont stockés dans des dossiers pour les sélectionner, les rechercher, et les afficher plus facilement.

Le support de l'application est formé de PCs possédant une souris et des écrans de moyenne ou de haute résolution, d'une tablette et d'un stylet pour les annotations. Les liaisons entre machines se font via un réseau local, un réseau ISDN, ou un réseau X.25. Des «scanners» peuvent aussi être ajoutés pour numériser des documents, et des imprimantes pour sortir ces derniers sur papier.

Pour chaque utilisateur, le logiciel est découpé en cinq parties :

- Le «scheduler» des processus nécessaires à l'application.
- L'interface pour l'affichage graphique.
- Un protocole multipoints appelé MCL.
- L'interface utilisateur.
- Le moniteur de conférence, qui possède l'état de la conférence.
- Le contrôleur de l'application qui gère à la fois les événements provenant des espaces partagés et leur consistance.

Télé-Amphi

Le système de conférence Télé-Amphi [GUIL92] permet de relier deux amphithéâtres par des médias de communication. L'orateur se trouve dans une salle alors que l'auditoire se trouve réparti dans la salle de l'orateur et dans la salle distante. Le conférencier dispose d'un équipement sonore qui diffuse ses propos dans les deux salles, de supports visuels projetés sur grand écran qu'il fait défiler avec des retours en arrière possibles, et d'un système de désignation pour pointer telle ou telle information. Les documents visualisés par le conférencier sont bien entendu vus par les deux salles, ainsi que le pointeur qu'il utilise. A tout moment, les questions émanant des deux salles ainsi que les réponses du conférencier sont entendues par l'ensemble des participants. Par contre, ce système ne propose pas de support vidéo «temps-réel», car il s'appuie sur des équipements courants et donc bon marché.

Le support matériel utilisé est composé de deux micro-ordinateurs de type PC reliés entre eux par une liaison NUMERIS, de microphones pour le conférencier et pour les auditoires, et de deux systèmes de vidéoprojection, chacun étant relié à un micro-ordinateur et à des enceintes audio.

Une évaluation du service proposé a fait apparaître l'importance de la qualité du son transmis, l'utilité du pointeur, ainsi que l'importance de la qualité des supports visuels couleur. Comme extensions, il est possible de rajouter la transmission de la photo du conférencier. L'ajout d'une fonction tableau permettrait aussi au conférencier d'annoter ses documents, d'écrire ou de réaliser des croquis de façon interactive. En dernier lieu, il serait utile d'étendre cette conférence à plusieurs salles reliées entre elles par des liaisons multipoints.

VideoWindow

Le mur vidéo [FISH90] est une visioconférence composée de deux écrans géants placés contre les murs de deux salles séparées physiquement. Le but de cette visioconférence est de rendre le plus proche possible de la réalité la confrontation à travers ce média, comme si tous les participants, qui se trouvent en fait dans deux endroits séparés, se trouvaient réunis dans une seule et même salle.

Par la suite, des études sociologiques ont été menées pour voir les différences de comportement entre des confrontations réelles et des confrontations via la visioconférence. On peut surtout noter que les gens avaient tendance à parler un peu plus fort au travers de la visioconférence, que peu de conversations entamées étaient abandonnées malgré quelques défauts technologiques (angles morts, problèmes de parallaxe...). Par contre, pour le contexte privé, il a été signalé l'impossibilité d'entreprendre des conversations privées. Ce système n'améliore pas non plus les relations sociales entre les gens. En effet, les gens qui ne se connaissent pas n'entament pas plus de conversation au travers de la visioconférence que s'ils se trouvaient réunis dans une même salle.

TeamWorkStation

Le système TeamWorkStation [ISHI90], [ISHI91], [ISHI93] est tout d'abord une conférence vidéo. Les conférenciers partagent un même espace à l'intérieur de la conférence. Cet espace est formé du mixage de deux (ou plus) images vidéo provenant de deux sources différentes, comme s'il s'agissait d'images transparentes.

L'application de ce concept de superposition d'écran est faite dans un contexte enseignant/enseigné. Un élève réalise un exercice de dessin et le professeur effectue par transparence les corrections et les annotations nécessaires.

Autres applications

D'autres projets de visioconférence peuvent être notés comme MultiG [PEHR91] qui intègre la visioconférence dans tout un environnement multimédia coopératif, PMTC [MASA92], et «The Broadband ISDN Group Tele-Working System» [HOSH92] pour réaliser des plate-formes au dessus du réseau à large bande passante ATM, MMConf [CROW90] une visioconférence incluant le partage d'applications, Comet [ANDE92] une plate-forme de conférence écrite sous forme de classes C++, CTUC [WANG92] et XTV [CHUN94] deux conférences au dessus de X Window, JVTOS [DERM93] une visionconférence avec partage d'applications, télépointage, TOSCA [PRIN93] une base de données multimédia couplée à un «browser» et à une visioconférence, LookingGlass [SCRI94] une autre visioconférence avec partage d'écran. Des recherches plus futuristes [TAKE92] cherchent à faire le lien avec la réalité virtuelle en créant des salles de conférences virtuelles...

3.6. Divers

Cette dernière section aborde une série d'applications coopératives qui ne rentrent pas dans les cadres précédemment définis, mais qui donnent une idée de la diversité des travaux réalisés sous l'appellation «coopératif». Les travaux réalisés cherchent plutôt soit à évaluer l'impact au niveau sociologique de produits de groupe, soit à réaliser des études sociologiques pour certaines formes de travail en groupe. Elles s'intéressent essentiellement à la partie «humaine» de la coopération.

Prototypage coopératif

Une démarche nouvelle [BODK89] pour la conception de logiciels est proposée sous la forme d'un prototypage coopératif. Il s'agit d'une tentative pour la création d'un logiciel de façon coopérative entre un ensemble de concepteurs et les futurs utilisateurs. Le logiciel à produire est un aide destiné aux secrétaires de dentistes pour la gestion des dossiers des patients. Les spécifieurs présentent une maquette du futur produit, maquette que les futurs utilisateurs (les secrétaires) essayent, testent, jugent, critiquent. Dans la mesure du possible, les spécifieurs modifient leurs maquettes en «temps réel» pour répondre aux modifications jugées nécessaires par les futurs utilisateurs.

Contrôle aérien

La coopération intervient aussi beaucoup dans le domaine du contrôle du trafic aérien. L'étude et l'évaluation effectuée à partir du logiciel de contrôle du trafic d'avions RD3 [HARP89] montre qu'il est nécessaire de prendre en compte la notion de groupe et les besoins du groupe pour obtenir un bon produit coopératif. De plus, les besoins de chacun des membres du groupe et leur hiérarchie sociale doivent également être bien intégrés au système coopératif. En effet, ce logiciel de contrôle de trafic aérien ne respectait pas tous ces critères et s'est très vite révélé lourd et peu performant.

Etude sociologique

[ROBI89] décrit une étude sociologique menée au sein d'une organisation. Il a été mis à la disposition de tous les membres de l'organisation un logiciel qui leur permet de négocier leurs salaires de façon coopérative avec ceux des autres. Finalement, les résultats obtenus ont été adoptés par le conseil d'administration de l'entreprise. En conclusion, l'expérience apparaît comme originale, mais ne présente pas un grand intérêt, sauf peut être au niveau sociologique.

Autres applications

[FILI93] présente l'analyse du comportement coopératif d'un ensemble de contrôleurs humains dans une salle de contrôle du Réseau Express Régional de Paris, [BECK93] étudie les performances d'équipes écrivant en groupe, [PAGA93] teste l'efficacité d'une visioconférence, [SHU92] analyse les actions d'un groupe de personnes concevant en commun des objets en trois dimensions...

4. Remarques. Commentaires

Cette section donne un certain nombre de remarques et de réflexions sur les modèles rencontrés, leur utilité, et sur les applications coopératives en général.

4.1. Liens entre les applications et les modes d'interaction coopératifs

Les grands groupes d'applications proposés reflètent assez fidèlement la classification asynchrone/synchrone et informelle/formelle proposée.

Applications distribuées multi-utilisateurs

Les applications distribuées multi-utilisateurs sont dans leur grande majorité synchrones (l'édition de texte asynchrone [SASS93] étant peu fréquente et représentant presque une exception) et par contre informelles.

Les données manipulées par ces applications ne sont jamais des données multimédias «temps-réel» telles que la vidéo par exemple.

Les groupes ne présentent non plus aucune structuration. Le rôle de chaque membre est identique à celui des autres membres du groupe. De ce fait, comme tous les membres ont les mêmes pouvoirs, les vues des données sont identiques. Du fait de la simplicité de la structuration des groupes, peu ou pas de protocoles sociaux sont utilisés. Ces protocoles pourraient servir à attribuer les responsabilités, à déterminer les rôles.

La gestion des conflits entre les données n'est parfois traitée que sommairement ; elle est alors laissée à la charge des utilisateurs [ELLI91].

Systèmes support de décisions de groupe

Ces applications manipulent des groupes synchrones et font partie des systèmes formels. En effet, l'évaluation des décisions ne peut se faire que si le système analyse les données qui lui sont fournies. Par contre, la partie de création peut se faire de manière plus informelle, mais le produit résultant devra être interprété par le système.

Les données manipulées par ces applications peuvent être multimédias.

Ces systèmes manipulent des groupes structurés dans lesquels les membres se voient attribuer des rôles et des fonctions.

Finalement, la plupart des systèmes permettent d'aider un ensemble d'utilisateurs pour une tâche bien précise comme la production de documents ou l'inspection de code. En effet, un système d'aide plus général bute très vite sur la façon de représenter n'importe quel but ou décision, de représenter et d'attribuer des rôles généraux, pour que ceux-ci soit manipulables automatiquement.

Messagerie «context-sensitive»

Les systèmes utilisant des messageries sont typiquement des systèmes asynchrones et formels. De ce fait, les communications entre les utilisateurs ne se font que de manière asynchrone. On n'a pas de connexion possible entre les utilisateurs, ni même de communications «temps-réel». Ces systèmes ne peuvent donc pas manipuler de données multimédias «temps-réel» comme la vidéo.

Par contre, les groupes possèdent une certaine structuration et il est possible d'attribuer des rôles et des pouvoirs aux membres des groupes.

Les messages échangés sont analysés et traités semi-automatiquement par le système. En quelque sorte, les caractéristiques des applications de messageries «context-sensitive» sont les opposées de celles des systèmes distribués et des conférences.

Intelligence artificielle distribuée

Les applications provenant de l'intelligence artificielle distribuée sont toutes formelles, mais peuvent être asynchrones ou synchrones. La sémantique des données, les traitements des erreurs ou des inconsistances, la gestion de l'attribution des rôles et des tâches sont bien pris en compte par ces systèmes.

Mais, ces systèmes restent abstraits dans le sens où ils ne fournissent pas d'application réelle, et qu'ils se contentent le plus souvent de simulations. Le but du domaine de l'intelligence artificielle distribuée est plutôt de tester et de valider des idées ou des stratégies et non, dans un premier temps, d'être efficace ou de réaliser des applications utilisables directement. L'intérêt de ce domaine est donc les idées, les principes, les concepts proposés dans ces travaux. L'inconvénient est de ne pas pouvoir les utiliser de façon réaliste et efficace. Les données multimédias ne sont pas non plus utilisées dans ce type d'application.

Conférences

Les conférences et visioconférences sont l'exemple type de systèmes synchrones et informels. Les données manipulées sont très peu ou pas utilisées par le système qui se charge uniquement de les véhiculer sans les analyser ni les traiter. La sémantique des données est laissée à la charge des utilisateurs ou à la charge des applications qui utilisent les conférences. On n'a donc qu'une acquisition et une visualisation.

Comme pour les applications provenant des systèmes distribués, les applications de conférence ne prennent pas en compte la structuration des groupes. Chaque membre a le même pouvoir que les autres membres du groupe, et, de ce fait, l'affichage des données est identique pour chaque membre. Cela pose parfois des problèmes car en particulier il est impossible de créer des apartés entre certains membres du groupe, ou de réaliser des conversations privées [FISH90].

De la même façon, le manque de protocoles sociaux pour gérer les responsabilités et les rôles provient d'un manque de structuration des groupes, bien que [ROBI91] pressente leur nécessité pour gérer un grand nombre de participants.

4.2. Remarques générales

Les problèmes généraux liés à la coopération sont très difficiles. Les intervenants peuvent avoir différentes données, différentes vues («scopes»), différentes heuristiques, différents objectifs, différentes habitudes et différentes sensibilités. En principe, le minimum d'incompatibilité est recherché : la sélection des agents, le partage de l'information, l'attribution des tâches et le contrôle du déroulement doivent se faire de la façon la plus adéquate possible. Le problème logiciel devient bien plus complexe que les problèmes classiques de contrôle de la concurrence et de la distribution (partage des ressources et maintien de la cohérence) dans les systèmes distribués. En effet, on ne cherche pas à masquer l'activité des autres, indépendante de sa propre activité dans un système distribué, mais on cherche à profiter et à coordonner son travail avec celui des autres [RODD92].

Les agents adoptent des stratégies, définissent des buts et effectuent des actions qui d'une certaine manière conduisent au succès de l'ensemble du groupe, et donc d'eux-mêmes. En fait, c'est la tâche globale qui constitue le succès. Le but que le groupe coopératif doit atteindre est souvent connu à priori et permet déjà de guider le déroulement de la coopération. Les agents ne sont pas des concurrents dans le sens des systèmes distribués où chaque agent doit accomplir

une tâche, cette tâche pouvant entrer en conflit avec la réalisation d'autres tâches, le système devant résoudre les conflits et de ce fait masquer l'exécution des autres tâches, mais des coopérants [LESS81]. Cela signifie que les agents ne sont pas en compétition les uns avec les autres, mais qu'ils forment un groupe coopératif dont le but est la résolution en commun de la tâche globale.

Quelques critiques sont cependant à faire concernant les modèles rencontrés et leurs applications. Il est souvent intéressant de définir des agents dits «intelligents», possédant de grandes capacités de raisonnement, s'appuyant sur des connaissances, des croyances, des hypothèses, des désirs, mais il paraît très difficile de les formaliser et de les utiliser de façon réaliste et efficace. Souvent, les agents ne sont même pas distribués mais s'exécutent sur une même machine. Les domaines d'application ne sont pas généraux et ne s'appuient que sur des exemples bien précis et bien délimités.

En dernier lieu, la coopération peut signifier :

- Un partage d'information.
- Un échange de messages entre agents.
- Le maintien de la cohérence en levant et en résolvant les conflits.
- L'utilisation de stratégies locales pour réaliser des travaux.
- Le choix de compromis acceptables par rapport à une solution virtuelle parfaite.
- etc...

Conclusion

Cette partie a essayé de caractériser la coopération et le CSCW, mais ces notions restent cependant imprécises du fait de multiples définitions existant actuellement et du fait que le CSCW provient de plusieurs domaines encore difficilement reliés entre eux : les systèmes informatiques distribués, l'intelligence artificielle, les sciences humaines et notamment la sociologie.

La provenance multi-disciplinaire de la coopération et l'étendue des recherches touchant aux travaux coopératifs se reflète aussi dans la diversité des applications coopératives. Chaque classification proposée (systèmes distribués multi-utilisateurs, systèmes support de décision de groupe, messageries, systèmes d'intelligence artificielle distribuée, conférences) se rapproche davantage d'un des domaines originels du CSCW et cherche à s'enrichir des apports des autres domaines.

L'intelligence artificielle distribuée est intéressante du point de vue des raisonnements sur les échanges d'informations, des apports d'heuristiques, des contrôles de la cohérence... et ces travaux s'appuient sur des modèles formels. Malheureusement, les applications sont souvent éloignées de la réalité et peu utilisables en tant que telles. Les travaux provenant des systèmes distribués sont plus réalistes, mais ils ne se contentent la plupart du temps que de transports d'informations entre des groupes d'utilisateurs. Ils sont souvent peu formels et peu modélisés. En dernier lieu, les sciences sociales ouvrent des perspectives notables dans la compréhension du comportement de groupes d'utilisateurs. Cependant, leur arrivée est nouvelle dans le monde de l'informatique et, dans un premier temps, l'association de leurs résultats avec les domaines précédents s'avère difficile à faire.

A partir de ces études générales, le chapitre suivant va présenter le modèle formel utilisé pour la représentation des groupes coopératifs.

Chapitre II

Modèle pour la coopération

Introduction

Ce chapitre présente une définition formelle du concept de coopération dans les systèmes distribués. L'approche retenue est basée sur la logique modale, ceci afin de caractériser non seulement la communication, mais aussi la structure coopérative dans laquelle le comportement distribué d'un ensemble d'agents se produit [DIAZ92], [DIAZ93a], [DIAZ94a]. Le modèle proposé permet d'exprimer la coopération en se situant au dessus de la couche de communication. L'aspect considéré dans la coopération est celui des relations entre données, relations déduites de leurs partages entre agents coopérants. En effet, la définition la plus faible retenue pour caractériser une coopération entre deux agents est celle de la mise à disposition d'information privée : un agent coopère avec un autre s'il lui communique une partie de sa connaissance. La définition prise pour représenter la coopération est donc celle du partage d'information. La logique modale permet de structurer la coopération à partir des liens qui existent entre activités coopérantes au sujet des données, et fournit une base sémantique pour l'interprétation des connaissances. Les groupes coopératifs sont découpés en domaines de coopération en attribuant judicieusement les données manipulées par la coopération.

Par rapport au modèle général, trois extensions sont ensuite proposées : la première permet de définir des groupes coopératifs dynamiques, dont la structure peut évoluer dans le temps. La deuxième extension propose la création d'apartés ou sous-groupes temporaires intervenant dans la coopération. La troisième effectue la distinction entre coopération et communication, et montre comment ces deux niveaux peuvent être reliés.

Le plan de ce chapitre est donc le suivant :

La première partie présente le modèle de coopération basé sur la logique modale. La deuxième partie montre comment la logique modale fournit une base sémantique pour l'évaluation des données et des prédicats. La troisième partie décrit les extensions du modèle, c'est-à-dire la dynamicité des groupes coopératifs, la formation dynamique d'apartés et leur cadre d'utilisation, puis les relations entre les dépendances de données et les communications.

1. Modèle logique pour la coopération

Les relations et les propriétés entre des agents communicants sont souvent représentées par des logiques non classiques [MANN89]. De nombreuses approches s'appuient sur des logiques modales [HUGH68], et permettent de vérifier les comportements distribués d'ensembles d'agents [EMER89]. La logique modale sert à interpréter les graphes des comportements des systèmes, et permet d'abord d'exprimer, puis de prouver les propriétés de sûreté et de vivacité de ces comportements.

Le modèle proposé pour la coopération repose sur cette dernière logique, afin d'être compatible avec les travaux précédents, et afin de fournir un support intuitivement adéquat pour exprimer certains aspects importants liés à la coopération. Il sert en particulier à caractériser la répartition et le partage d'information entre agents coopérants. De plus, il autorise l'expression de valeurs non spécifiées ou inconnues, et permet également la levée de contradictions entre prédicats lorsque, par exemple, plusieurs évaluations d'un même prédicat dans diverses parties d'un système coopératif complexe fournissent des résultats différents.

D'après ce modèle, une structure de coopération est composée d'un ensemble d'agents communicants qui possèdent et partagent des données privées de sémantiques différentes, et qui font partie de divers systèmes logiques modaux.

1.1. Logique modale. Présentation

La *logique modale* est la logique du possible et du nécessaire. La validité des prédicats de ce type de logique est basée sur les modèles ou structures de Kripke, composés de triplets <W, **R**, V> où :

- W est un ensemble d'états ou de mondes. W contient les mondes considérés. $W = \{w_i\}$.
- **R** est une relation binaire définie sur l'ensemble W.
- V est une fonction d'interprétation des variables propositionelles p_j et des formules $\alpha, \beta...$
- {0, 1} est l'ensemble d'interprétation des formules et des prédicats, 0 pour Faux, 1 pour Vrai.

La relation **R** permet de définir différents systèmes logiques dont dépend la validité des formules modales. En effet, certaines formules, valides dans un système, peuvent ne plus l'être dans un autre. Les systèmes modaux les plus fréquemment rencontrés sont T, B, S4, S5. Une relation **R** réflexive définit un système T. Un système B provient d'une relation **R** réflexive et symétrique. Si **R** est réflexive et transitive, elle définit un système S4. Un système S5, quant à lui, est caractérisé par une relation **R** à la fois réflexive, symétrique et transitive.

La fonction d'interprétation des prédicats V s'applique sur l'ensemble des prédicats et sur l'ensemble des noeuds. Elle est définie par les règles suivantes :

a. $\forall p_j$ et $\forall w_i$, $V(p_j, w_i)=0$ ou $V(p_j, w_i)=1$;

b. $\forall \alpha, \forall w_i$, $V(\neg\alpha, w_i)=0$ si $V(\alpha, w_i)=1$; $V(\neg\alpha, w_i)=1$ si $V(\alpha, w_i)=0$;

c. $\forall \alpha, \beta, \forall w_i$, $V(\alpha \vee \beta, w_i)=1$ si $V(\alpha, w_i)$ ou $V(\beta, w_i)=1$; $V(\alpha \vee \beta, w_i)=0$ sinon ;

d. $\forall \alpha, \forall w_i$, $V(L(\alpha), w_i)=1$ si $\forall w_j$ tel que $w_i \mathbf{R} w_j$ alors $V(\alpha, w_j)=1$;

sinon, $V(L(\alpha), w_i)=0$.

L'opérateur L exprime une forme de cohérence entre le monde w_i et les mondes w_{jk} qui sont en relation par $w_i \mathbf{R} w_{jk}$, c'est-à-dire un monde w_i et ses successeurs w_{jk}. Elle caractérise ainsi la propagation de l'information au sein d'un groupe [TREH93a], [TREH93b].

La figure II.1 donne trois exemples simples illustrant la signification de L. Sur la figure II.1, la relation **R** a été représentée graphiquement. Un lien entre deux mondes w_i et w_{jk} est représenté par un arc orienté allant du noeud w_i vers le noeud w_{jk}. Ainsi, sur la figure II.1(a) où tous les mondes w_i, w_{j1}, w_{j2}, et w_{j3} voient la même valeur du prédicat α (α=1), $V(L(\alpha), w_i)=1$. Sur la figure II.1(b), par contre, w_i voit α=0, donc $V(L(\alpha), w_i)=0$. De même, sur la figure II.1(c), w_{j1} voit α=0 donc $V(L(\alpha), w_i)=0$.

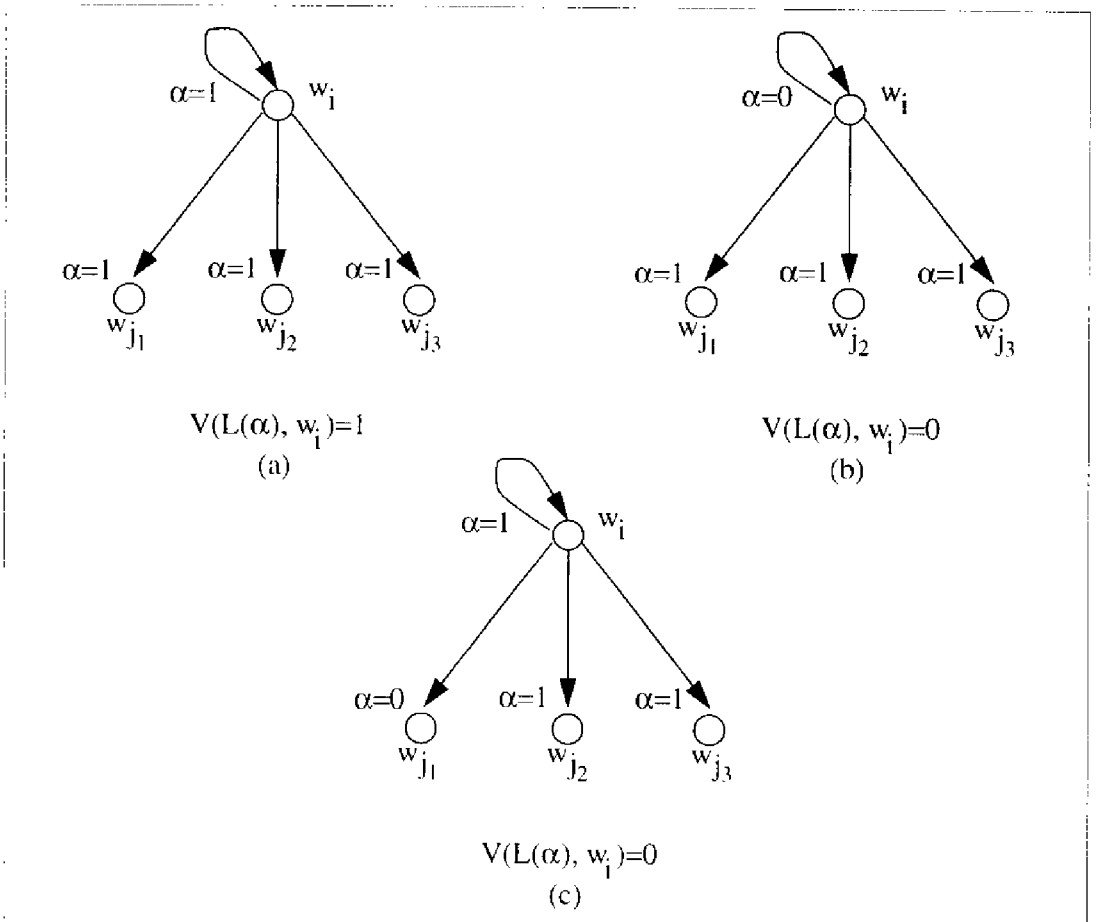


Figure II.1. Illustration du rôle de l'opérateur L.

Les choix de W, l'ensemble des mondes, et de **R**, la relation entre ces mondes, impliquent la définition de la validité. W et **R** vont être utilisés pour proposer un modèle formel d'une notion de coopération basée sur le partage d'information.

1.2. Modèle de systèmes coopératifs

L'approche retenue pour caractériser la coopération est celle du partage d'information entre agents [BENF92]. La coopération pour un agent signifie donc la mise à disposition d'une partie de ses connaissances, de ses informations personnelles et de ses données propres vers d'autres agents. En conséquence, un agent coopère avec un autre agent s'il rend une partie de ses données visibles et disponibles à l'autre agent [DIAZ92], [DIAZ93a], [DIAZ94a].

Soit A un ensemble d'agents. $A = \{A_i\}$. Chaque agent A_i possède un ensemble d'informations et de connaissances dont il est propriétaire. Soit $P = \{p_i\}$ l'ensemble des prédicats représentant la connaissance et les informations propres à un agent donné A_i. Une partie de ces informations reste uniquement connue de cet agent. Elle forme les informations locales à cet agent. Par contre, une autre partie des données va comprendre les données exportées qui vont être mises à disposition d'autres agents. Dans chaque agent, les prédicats sont soit locaux, soit exportés. Lorsqu'un agent va faire connaître à d'autres agents ses données exportées, alors il sera en relation avec ces autres agents. Le partage d'information crée une relation **R** entre les divers agents coopérants (figure II.2). Cette relation de partage d'information introduit donc une structuration de l'ensemble des agents du groupe coopératif. Les liens définis par les prédicats exportés par un agent vers d'autres agents caractérisent la relation de coopération. Ceci nous conduit aux définitions suivantes :

- *Définition 1* : A_i est en coopération avec A_j lorsque A_i permet à A_j d'accéder aux valeurs de certains de ses prédicats, les prédicats exportés. Ces prédicats deviennent connus de A_j.

- *Définition 2* : Les prédicats exportés par un agent deviennent connus des agents qui précèdent cet agent suivant la relation **R**, et non connus directement par leurs antécédents. Les prédicats locaux ne sont pas connus des agents qui sont en coopération avec l'agent considéré.

Figure II.2. L'agent A_j peut lire les valeurs «x» et «y» que A_i possède et exporte. A_i ne peut pas lire «z».

La relation **R** de partage d'information peut être représentée de façon graphique par un graphe. Les sommets représentent les agents en coopération, les flèches décrivent la relation **R**. Les sommets contiennent les informations et les connaissances de chacun des agents, les flèches du graphe représentent l'accès aux valeurs des données exportées par les agents. Le graphe formé de l'ensemble des agents coopérants et de la relation **R** est appelé *«graphe de coopération»* (figure II.3).

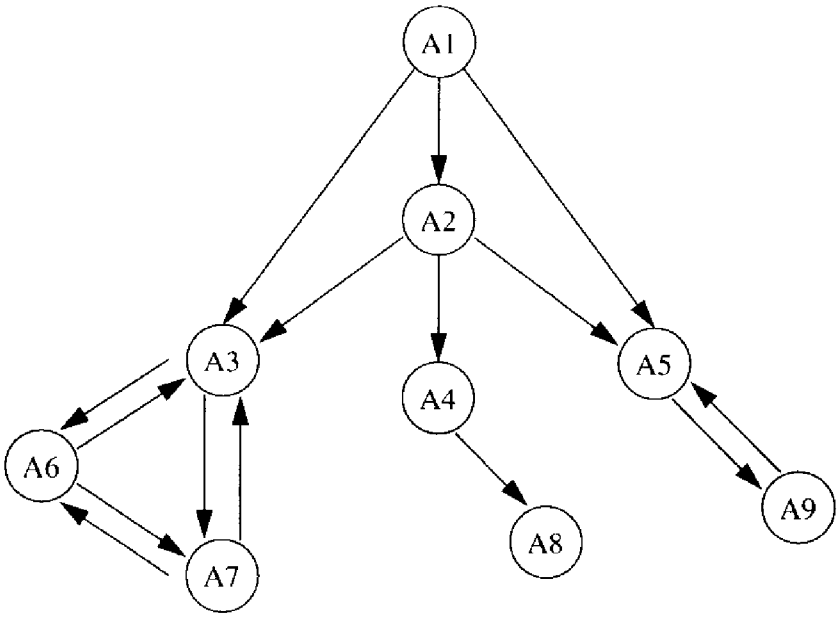


Figure II.3. Un graphe de coopération.

La structuration du groupe coopératif, les relations entre les agents coopérants et les partages d'information s'apparentent fort au modèle de logique modale proposé précédemment. Chacun des mondes w_i de la logique modale devient un agent coopérant A_i. La relation **R** entre les mondes de la logique modale correspond à la relation **R** de partage d'information pour la coopération. En conséquence, la structure de Kripke est en fait le graphe de coopération. La logique modale permettra aussi de fournir une base sémantique pour l'évaluation V des prédicats de la coopération.

Ce modèle donne une vision de la connaissance et de sa répartition entre agents coopérants [HALP86], [TREH93a], [TREH93b]. La difficulté, quand on modélise la connaissance, est de choisir l'ensemble des mondes accessibles à un agent donné. Si l'on considère des comportements distribués, l'ensemble des mondes accessibles peut être défini par le graphe de tous les états possibles, graphe déduit des descriptions formelles [JUAN88], en association avec l'utilisation des logiques modales et temporelles pour vérifier les propriétés dans les systèmes distribués. Cependant, le point de vue donné par la logique modale ne considère que les états d'information et non les états intentionnels [WERN88], ces derniers faisant partie d'un niveau conceptuel différent, la décision.

La relation **R** peut posséder diverses propriétés :

R est réflexive, donc $A_i \mathbf{R} A_i$, car un agent connaît ses propres prédicats.

Dans le cas général, **R** n'est ni symétrique ni transitive (figure II.3). Si **R** est symétrique, la coopération est dite bilatérale (figure II.4 (a)). Si **R** est transitive, la coopération est hiérarchique (figure II.4(b)). Si **R** est à la fois symétrique et transitive, on dit qu'elle est parfaite ou totale (figure II.4(c)). Les sémantiques de la coopération qui définissent le comportement d'un agent donné vont ainsi dépendre de la relation **R**, i. e. de la structure de la coopération.

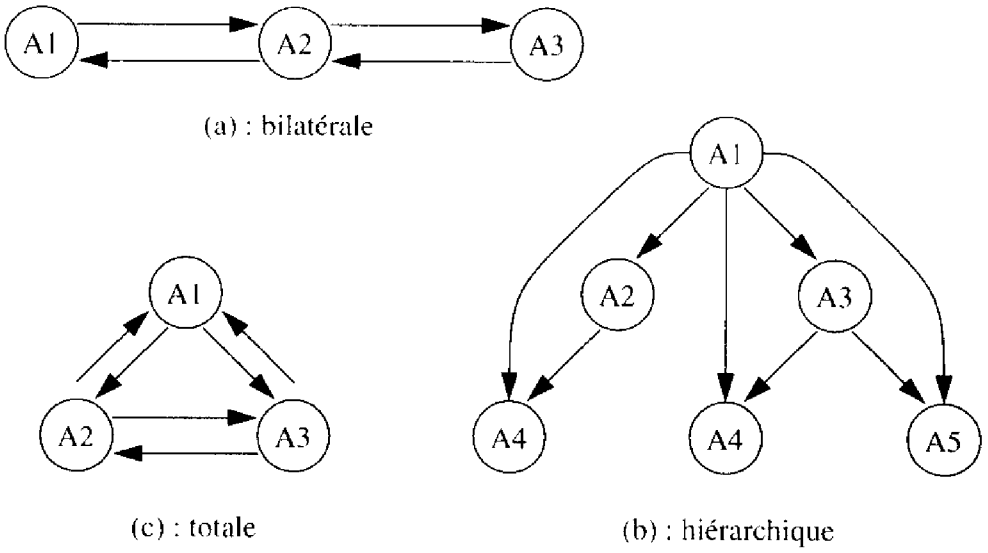


Figure II.4. Quelques structures de coopération particulières.

1.3. Dépendances des données et prédicats

La relation **R** introduit une structuration dans le groupe coopératif. Cette structuration entraîne également des dépendances entre les prédicats de chacun des agents Ai, un agent pouvant être propriétaire de prédicats dont la valeur dépend d'autres prédicats exportés par d'autres agents.

Soit un graphe de coopération suivant la logique S4 acyclique, où **R** est transitive. Chaque agent voit toutes les valeurs exportées par les agents situés au dessous de lui dans le graphe. Par contre, les prédicats évalués par les antécédents de A_j ne peuvent pas être connus par A_j. La connaissance des prédicats est donc directement liée à la structure du groupe. Dans le cas général, **R** peut ne pas être triviale en terme de prédicats et peut entraîner des relations de dépendances entre des valeurs de prédicats. A titre d'exemple, la figure II.5 donne un cas de dépendances de prédicats. Les données encerclées représentent les prédicats locaux ou importés connus de chaque agent. Ils donnent la connaissance de chaque agent dans la coopération. Les prédicats sur les flèches sont les prédicats exportés par un agent. Dans le cas de la figure II.5, le prédicat V appartenant à A2 dépend à la fois des prédicats exportés X et Y, mais aussi du prédicat U local à l'agent A2. De plus, comme A3 n'a pas accès aux prédicats exportés par A2, il ne peut connaître ni U, ni V. Par contre, les prédicats U et V sont exportés tels quels vers A1, sans porter à sa connaissance directe les prédicats dont U dépend. Ainsi, A3 connaît X, Y et Z. A2 connaît U, X, Y et V, et A1 connaît O, P, U et V.

L'allocation des prédicats aux agents est donc une étape majeure de la conception. Dans le cas particulier où l'on désire exporter une valeur importée, il suffit d'affecter la valeur reçue à un nouveau prédicat dont la valeur sera liée à celle de l'ancien prédicat par la relation d'égalité.

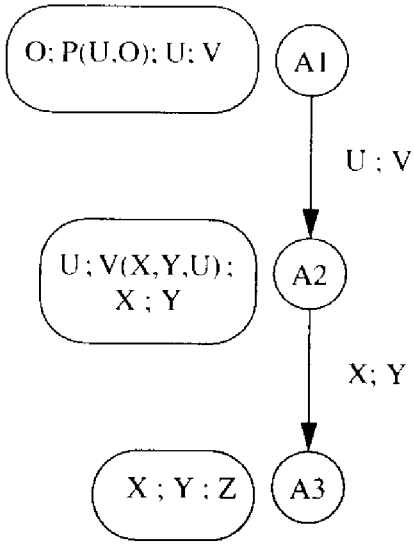


Figure II.5. Répartition des prédicats suivant la structure de la coopération.

1.4. Domaines de coopération

Dans le cas général, la relation **R** est quelconque. Il s'ensuit que, pris dans leur ensemble, tous les agents appartiennent à un système T, ce qui revient à considérer que l'ensemble de la coopération suit une seule et même règle donnée par le système T. En fait, intuitivement, les systèmes complexes ne sont pas conçus ainsi, c'est-à-dire ne sont pas conçus globalement, et ne forment pas un tout insécable. Les systèmes complexes sont le plus souvent organisés de façon modulaire, sont structurés et sont découpés en sous-systèmes relativement indépendants entre eux. Cette approche est également utilisée pour regrouper certains agents suivant leurs relations, suivant leurs rôles et suivant les travaux qu'ils doivent accomplir en commun au sein du groupe coopératif. Certains agents sont alors reliés entre eux et regroupés en sous-ensembles, en fonction des sous-tâches définies et identifiées. Le découpage de la tâche principale en sous-tâches revient à considérer la relation globale **R** comme un ensemble de sous-relations s'appliquant sur des sous-ensembles d'agents. La coopération est donc structurée en un ensemble de sous-coopérations, en décomposant **R** en un ensemble de sous-relations. Ces sous-relations et leurs ensembles d'agents associés sont appelés des *domaines de coopération*.

A titre d'exemple, le groupe coopératif de la figure II.6 peut être vu de différentes façons. Lorsque l'on regarde A1, A2, A3 et A5 ainsi que les arcs les connectant, ces agents sont en coopération hiérarchique : la coopération **R1** réduite aux sommets A1, A2, A3 et A5 et aux cinq arcs qui les relient, est une restriction transitive de **R** qui décrit le domaine DOM1. La relation **R2**, restriction de **R** aux sommets {A3, A6, A7} et cet ensemble de sommets forment le domaine de coopération totale DOM2. De même, la relation **R3**, restriction de **R** aux sommets {A5, A9} et cet ensemble de sommets forment un autre domaine de coopération totale, DOM3. Finalement, **R** réduite au sommets {A2, A4, A8} forme la relation **R4** qui décrit le comportement du domaine DOM4. L'agent A3 est donc impliqué dans deux sous-coopérations différentes, une hiérarchique, **R1**, l'autre totale, **R2**. Cependant, A3 fait aussi partie de la relation de coopération globale **R**, incluant tous les agents, et définie par l'ensemble du graphe de la figure II.6.

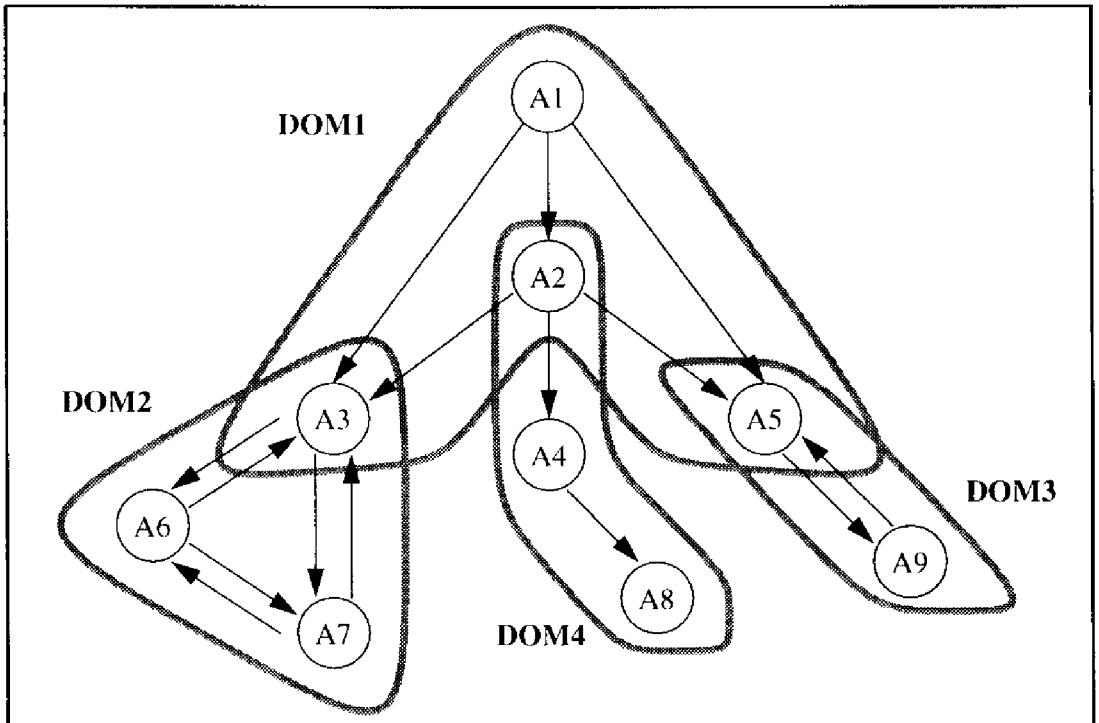


Figure II.6. Le groupe coopératif est découpé en plusieurs domaines DOMi.

Le comportement d'un agent dépend par conséquent de la logique guidant chacun des domaines de coopération :

- Dans une coopération totale, cet agent suit le système S5 comme tous les agents de cette coopération totale.
- Dans une coopération hiérarchique, il suit le système S4 comme tous les agents de la coopération hiérarchique.
- Dans une coopération bilatérale, il suit le système B comme tous les agents de la coopération bilatérale.
- Dans tous les autres cas, il respecte la logique du système T.

Sur la figure II.6, lorsque A3 est considéré dans **R**, il suit les règles données par le système logique T. Pour **R1**, il suit S4. Pour **R2**, il suit S5.

Domaines cohérents et prédicats

Le découpage en domaines permet de découper la tâche globale associée à la coopération en un ensemble de sous-tâches pouvant s'exécuter de manière autonome ou semi-autonome par rapport aux autres sous-tâches coopératives. **R** peut être divisée en différents domaines de coopération dédiés à des sous-problèmes, ou sous-applications, cohérents. Ces domaines définissent dans un certain sens des ensembles gérables d'agents.

C'est aussi un deuxième moyen de structurer la coopération, de grouper et de répartir l'information entre les différents domaines [HEWI91]. En effet, certaines informations pourront être regroupées à l'intérieur d'un domaine, et de ce fait n'être accessibles qu'aux membres de ce domaine. Il s'agit de la notion d'information privée.

De plus, comme un domaine définit un groupe devant réaliser un travail de manière coopérative, il est nécessaire que les données demeurent cohérentes à l'intérieur de ce même domaine. Cette condition est garantie si chaque prédicat du domaine est affecté de manière unique à un et un seul agent du domaine, qui devient propriétaire de ce prédicat. Seul l'agent propriétaire peut modifier sa valeur, les autres agents du domaine en relation avec le propriétaire se contentant d'accéder à cette valeur. Ceci conduit à la définition suivante :

- *Définition 3* : Soit une coopération divisée en domaines qui forment des sous-graphes de **R**. Ces domaines sont tels que tous les prédicats dans un domaine peuvent prendre soit la valeur Vrai, soit la valeur Faux, mais pas les deux en même temps.

Remarque : l'hypothèse d'attribuer la modification d'un prédicat à un seul agent dans un domaine peut être relâchée si les valeurs d'un prédicat, affecté à plusieurs agents dans un même domaine, restent cohérentes et non contradictoires. Ceci peut être réalisé au moyen d'algorithmes de gestion de copies multiples ou de gestion d'accès concurrents à une même donnée. Le fait de n'avoir qu'un seul et même rédacteur et plusieurs lecteurs est finalement un cas particulier mais qui se révèle plus conforme à la sémantique de partage d'information donnée pour la coopération.

Les tâches et les responsabilités de chacun des agents doivent être soigneusement allouées pour éviter que les prédicats n'aient de valeurs contradictoires dans un domaine et pour qu'il soient évalués de manière cohérente. Cette notion de cohérence est finalement la définition d'une sous-coopération. Par conséquent, si un prédicat donné possède des valeurs logiques différentes, alors les évaluations correspondantes ne doivent pas se trouver dans le même domaine. En effet, un agent peut appartenir à différents domaines de coopération et donc éventuellement voir des logiques différentes.

Cohérence inter-domaine

Un même prédicat peut apparaître plusieurs fois dans des domaines de coopération différents. En reprenant le modèle de la figure II.6, il est possible qu'un prédicat donné p soit évalué à Vrai dans A3 et à Faux dans A1. Bien que A1 puisse voir A3, ils ne sont pas d'accord sur la valeur de p. Le problème est de décider si une telle situation est logiquement consistante. Une telle situation peut être acceptée comme consistante car les valeurs de A1 ne dépendent que du domaine de coopération défini par **R1**, alors que les valeurs de A3 peuvent dépendre en plus des valeurs du domaine associé à **R2**.

Lorsqu'une coopération se trouve découpée en plusieurs domaines, un agent A_j peut faire partie de plusieurs domaines, chacun ayant sa propre logique de coopération. Un prédicat p_k connu par A_j peut donc apparaître à la fois dans deux domaines auxquels A_j appartient. Alors, par définition des domaines, ce prédicat p_k est consistant vis-à-vis du premier domaine et aussi consistant vis-à-vis du second. Néanmoins, rien n'empêche p_k d'être incohérent si l'on considère ensemble, de façon non indépendante, les deux domaines (figure II.7). Ceci peut survenir en particulier dans A_j, car A_j accède aux deux domaines. L'évaluation de p_k se faisant dans chacun des domaines peut conduire à ce qu'il soit évalué à Faux dans un domaine et à Vrai dans l'autre. Cette situation conduit alors à une contradiction qui ne doit cependant pas empêcher pas le raisonnement [MURA90], [MURA91] si des solutions sémantiques sont données.

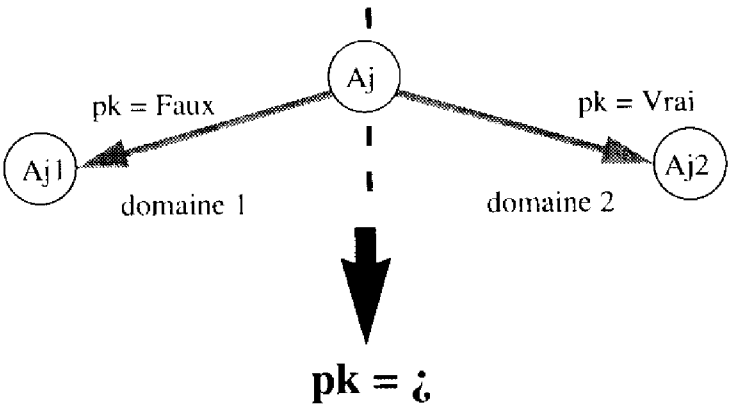


Figure II.7. Contradiction.

2. Sémantiques d'évaluation des prédicats

Cette section montre que la logique modale peut fournir plusieurs sémantiques pour l'évaluation et l'interprétation des données et prédicats échangés par la coopération. Deux principales sémantiques d'évaluation sont proposées, la première permettant d'évaluer les prédicats dans un seul domaine, la deuxième permettant de caractériser les évaluations sur plusieurs domaines.

2.1. Sémantique à trois valeurs

Afin de définir des sémantiques pour l'évaluation de ces prédicats dans un seul domaine de coopération, l'ensemble d'interprétation est étendu à trois valeurs. L'évaluation d'un prédicat peut donc donner l'une des trois valeurs suivantes :

1, 0, ø ou «don't care» ou «non-spécifié».

ø représente la valeur du prédicat p dans tous les noeuds où p n'est pas défini.

La fonction d'interprétation V doit également être étendue pour s'adapter au nouvel ensemble d'interprétation. La nouvelle fonction VD est définie comme suit :

a. $\forall p_j$ et $\forall w_i$, $VD(p_j, w_i)$ = soit 0 soit 1 soit ø.

b. $\forall \alpha, \forall w_j$, VD=V si $\alpha=0$ ou $\alpha=1$; $VD(\neg\alpha, w_i)=ø$ si $VD(\alpha, w_i)=ø$

c. $\forall \alpha$ et β, $\forall w_i$, $VD(\alpha \vee \beta)=\emptyset$ si $VD(\alpha, wi)$ et $VD(\beta, w_i)=\emptyset$. Sinon (l'un n'est pas $\emptyset$), $VD(\alpha \vee \beta)=1$ si $VD(\alpha, w_i)$ ou $VD(\beta, w_i)=1$; $VD(\alpha \vee \beta)$ égale 0 dans les autres cas.

d. $\forall \alpha$ et $\forall w_i$, $VD(L(\alpha), w_i)=\emptyset$ si $\forall w_j$ tel que $w_i \mathbf{R} w_j$, $VD(\alpha, w_j)=\emptyset$. Si $VD(L(\alpha), w_i)$ n'est pas $\emptyset$, alors $VD(L(\alpha), w_i)=1$ si $\forall w_j$ tel que $w_i \mathbf{R} w_j$, soit $VD(\alpha, w_j)=1$ soit $VD(\alpha, w_j)=\emptyset$. Dans les autres cas, $VD(L(\alpha), w_i)=0$.

Cette première sémantique est une extension de la valeur booléenne «don't care», ce qui correspond à supposer que $\alpha \vee \emptyset=\alpha$. Les coopérants ignorent les valeurs non spécifiées ou celles sur lesquelles ils n'ont aucune information.

Variantes sémantiques

En gardant $\emptyset \vee \emptyset=\emptyset$ et $0 \vee 1=1$, il apparaît que le choix de sémantique précédent n'est pas le seul possible et intéressant.

a) Par exemple, on peut aussi décider que $\alpha \vee \emptyset=\emptyset$, ce qui signifie que $\emptyset$ est plus prioritaire que 0 et 1. Cette nouvelle sémantique correspond au cas où une valeur non spécifiée empêche l'évaluation de toute une expression. Le symbole $\emptyset$ représente alors une sémantique de non-calculabilité.

b) Une autre possibilité de sémantique existe dans laquelle $0 \vee \emptyset=\emptyset$ et $1 \vee \emptyset=1$. Cette sémantique correspond à une vue optimiste des valeurs non connues. L'information associée aux prédicats évalués à Vrai prime sur celle qui est inconnue.

c) Dans la dernière sémantique possible, $0 \vee \emptyset=0$, $1 \vee \emptyset=\emptyset$. Cette sémantique correspond à une vue pessimiste des valeurs non connues. Toute information inconnue empêche l'accès à l'information associée aux prédicats évalués à Vrai.

Les quatre possibilités évoquées ont probablement chacune un intérêt dans des situations réelles différentes. Finalement, la définition d'une sémantique de coopération peut s'avérer très délicate et sophistiquée.

2.2. Sémantique à quatre valeurs

L'évaluation des prédicats sur plusieurs domaines doit pouvoir caractériser les cas d'incohérences. Par conséquent, la logique précédente est étendue avec une quatrième valeur qui sert à définir les contradictions.

1,0, $\emptyset$ ou «don't care», ¿ ou «contradiction».

La fonction d'interprétation doit encore être modifiée pour pouvoir prendre en compte ces quatre valeurs logiques. La différence est que, dans ce cas, les prédicats sont apparentés à plus d'un domaine, le problème étant de savoir comment manipuler les cas où se produisent les contradictions.

VC est la nouvelle fonction d'interprétation définie comme suit :

a. $\forall$ p et $\forall w_i$, on a $VC(p, w_i)= 0$ ou 1 ou $\emptyset$ ou ¿

b. Soit w_i appartenant à deux domaines distincts :

$\forall p_j$, avec la sémantique du «don't care» pour $\emptyset$,

$VC(p_j, w_i)=\emptyset$ si $VD(p_j, w_i)=\emptyset$ dans les deux domaines auxquels w_i appartient ;

$VC(p_j, w_i)=$¿ si $VD(p_j, w_i)=0$ dans un domaine et $VD(p_j, w_i)=1$ dans l'autre ;

sinon : a) $VC(p_j, w_i)=0$ si $VD(p_j, w_i)=0$ ou $VD(p_j, w_i)=\emptyset$ dans les deux domaines ;

b) $VC(p_j, w_i)=1$ si $VD(p_j, w_i)=1$ ou $VD(p_j, w_i)=\emptyset$ dans les deux domaines.

Cette définition s'étend facilement récursivement à N domaines en posant :

Si $VC(p_j, w_i)=¿$ pour les domaines k et k+1, alors $VC(p_j, w_i)=¿$ pour tout $j > k+1$.

c. $\forall \alpha, \forall w_i$, $VC=VD$ si $\alpha=0$ ou 1 ou $\emptyset$; ou $VC(\neg\alpha, w_i)=¿$ si $VC(\alpha, w_i)=¿$;

d. $\forall \alpha$ et $\beta, \forall w_i$, $VC(\alpha\vee\beta)=¿$ si $VC(\alpha, w_i)$ ou $VC(\beta, w_i)=¿$;

sinon (les deux différents de ¿) : $VC(\alpha\vee\beta)=1$ si $VC(\alpha, w_i)$ ou $VC(\beta, w_i)=1$;

sinon (différents de ¿ et de 1) : $VC(\alpha\vee\beta)=0$ si $VC(\alpha, w_i)=0$ ou $VC(\beta, wi)=0$;

sinon (différents de ¿, de 1 et de 0) : $VC(\alpha\vee\beta)=\emptyset$ (si $VC(\alpha, wi)$ et $VC(\beta, w_i)=\emptyset$) ;

e. $\forall \alpha$ et $\forall w_i$,

$VC(L(\alpha), w_i)=¿$ s'il existe w_j tel que $w_i\mathbf{R}w_j$ et $VC(\alpha, w_j)=¿$;

sinon $VC(L(\alpha), w_i)=\emptyset$ si $\forall w_j$ tel que $w_i\mathbf{R}w_j$, $VC(\alpha, w_j)=\emptyset$;

sinon $VC(L(\alpha), w_i)=1$ (si $\forall w_j$ tel que $w_i\mathbf{R}w_j$, soit $VC(\alpha, w_j)=1$ ou $\emptyset$) ;

sinon $VC(L(\alpha), w_i)=0$ (si $\forall w_j$ tel que $w_i\mathbf{R}w_j$, soit $VC(\alpha, w_j)=0$ ou $\emptyset$).

Traitement des contradictions

La sémantique proposée permet donc de détecter les incohérences, mais en aucun cas ne renseigne sur la manière de les traiter. Le problème essentiel pour les agents coopérants va donc être de poursuivre la coopération en prenant des décisions au sujet des prédicats incohérents. Les travaux provenant de l'intelligence artificielle distribuée [DURF89], [DURF87], basés sur les raisonnements sur des données incomplètes, erronées ou non fiables peuvent se révéler utiles.

Une possibilité de traitement des incohérences est de réaliser un choix pragmatique au dessus de l'interprétation sémantique. Ces choix sont définis comme des décisions qui réalisent pour ¿ la sélection d'une des deux valeurs Vrai ou Faux, i. e. d'une sémantique ou de l'autre. Ce point de vue, différent de [WERN88], se situe dans l'esprit de [VARD86]. En effet, donner une valeur logique Vrai ou Faux à ¿, c'est-à-dire choisir, par exemple dans un but de décision, signifie privilégier un des deux domaines, donc une des deux valeurs sémantiques possibles pour ¿, et donc, dans un certain sens, une sémantique par rapport à l'autre. Bien sûr un tel choix est difficile car les critères sont nombreux. Par exemple, le nombre d'agents participants ou de façon plus forte le système logique associé à un domaine peuvent fournir le critère prépondérant. On peut ainsi décider que les valeurs fournies par les coopérations totales seront toujours préférées aux valeurs fournies par les autres structures de coopération.

Une autre possibilité serait d'ignorer les incohérences. Dans ce cas, l'évaluation ¿ d'un prédicat serait transformée en $\emptyset$. Cependant, le fait d'ignorer les contradictions peut s'avérer simpliste et parfois irréaliste, si par exemple un prédicat représente un capteur surveillé par un agent. Certains prédicats optionnels ou peu critiques pourraient cependant se voir appliquer cette stratégie.

3. Extensions du modèle de coopération

Cette partie va présenter trois extensions du modèle de coopération précédent. Les deux premières extensions permettent de rendre dynamique la structure courante du groupe coopératif, la troisième fait le lien entre les dépendances des données et les échanges effectifs de ces données.

3.1. Evolution dans le temps

Le modèle précédent est restrictif dans le sens où il ne prend pas en compte l'évolution possible de la structure du groupe coopératif dans le temps. A un instant donné, tous les agents du groupe coopératif n'ont pas besoin d'être présents pour commencer ou pour accomplir le travail coopératif. Pour offrir une telle possibilité, le modèle a été étendu pour définir quels agents peuvent ou doivent être présents ensemble pour effectuer le travail coopératif [DIAZ93d], [DIAZ93e].

Graphe de domaine

Le graphe de domaine de coopération, appelé plus simplement *graphe de domaine*, est défini comme le sous-graphe du graphe de coopération ne comprenant que les agents qui font partie du domaine considéré. Si l'on se réfère à l'exemple de la figure II.6, le graphe de domaine DOM1 est formé des agents A1, A2, A3 et A5, le graphe de domaine DOM2 est formé des agents A3, A6 et A7, celui de DOM3 contient A5 et A9, celui de DOM4 est formé de A2, A4 et A8.

Graphe instantané de domaine

Le graphe de domaine de coopération représente les relations entre tous les agents d'un domaine sans tenir compte de l'évolution de la coopération dans le temps. Or, à chaque instant, dans un domaine de coopération, il n'est pas nécessaire que les agents participent tous en même temps au travail associé au domaine.

Pour chacun des domaines de coopération, un graphe instantané de domaine de coopération, appelé plus simplement *graphe instantané de domaine*, est défini comme un sous-graphe du graphe de domaine : il ne représente que les agents qui coopèrent effectivement à un instant donné dans le domaine considéré.

Graphe instantané de domaine valide

Parmi tous les sous-graphes des graphes de domaine, certains d'entre eux ne représentent rien ou n'ont aucun sens pour l'application coopérative réalisée. Les sous-ensembles d'agents qui ont une signification pour la réalisation du travail coopératif associé au groupe doivent donc être définis par la sémantique de l'application. Parmi tous les sous-graphes des graphes de domaines, l'application choisit donc ceux qui sont sémantiquement possibles. Les sous-graphes retenus par l'application sont appelés *graphes instantanés valides*. La figure II.8 montre deux exemples de sous-graphes valides du domaine DOM1 qui pourraient provenir de la structure coopérative de la figure II.6.

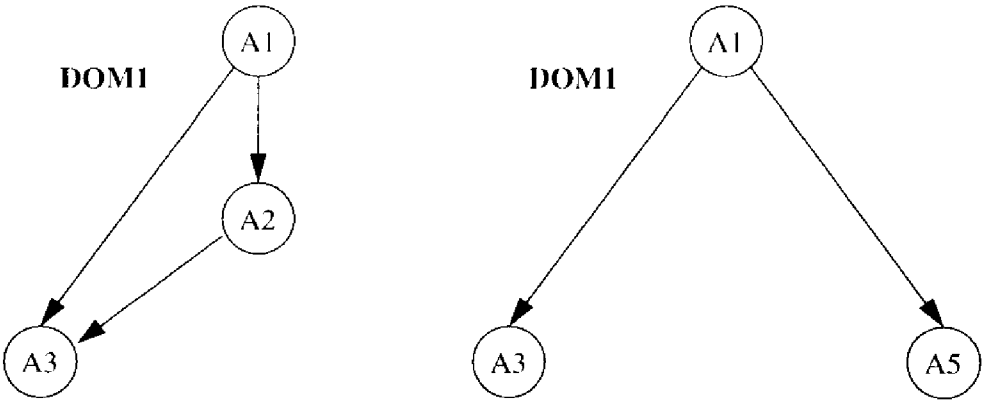


Figure II.8. Exemple de deux graphes instantanés valides provenant du domaine DOM1 du groupe coopératif de la figure II.6.

Pour définir quels sous-graphes sont valides, l'application doit donner un ensemble de propriétés exprimées par des règles ou des prédicats que ces graphes doivent respecter. Ces règles dépendent bien entendu du travail coopératif à accomplir. A titre d'exemple, quelques critères peuvent être donnés :

Pour un domaine, il faut :

- que tout graphe instantané valide soit connexe.
- au moins deux agents dans un graphe instantané valide.
- au moins N agents dans le graphe.
- au plus N agents dans le graphe.
- exactement N agents dans le graphe.
- que les antécédents d'un noeud soient dans le graphe.

etc...

Dans un cas plus général, s'il est impossible de trouver de simples critères pour la sélection des graphes instantanés valides, ces graphes peuvent être énumérés explicitement.

Remarque

L'ajout des configurations valides dans le modèle ne peut s'appliquer que dans le cas de coopérations synchrones. En effet, cette extension cherche à définir les configurations possibles pour lesquelles les coopérants peuvent travailler en même temps. Ceci n'est possible que si les agents peuvent être co-présents en coopération, ce qui correspond à la définition de la coopération synchrone. La coopération asynchrone se déroule sans la co-présence des différents membres : on n'a pas de notion de travail en même temps, et donc on ne peut avoir de configuration instantanée valide.

A priori, l'introduction des graphes instantanés valides semble restreindre la généralité du modèle de coopération. Ce choix a cependant été fait pour plusieurs raisons : tout d'abord, le but recherché est de formaliser et de modéliser la coopération synchrone. En effet, la plupart des modèles proposés s'appliquent davantage aux cas asynchrones, et beaucoup moins aux coopérations de type synchrone. La deuxième raison est de pouvoir par la suite utiliser ce modèle dans le cadre de manipulations de données vidéo «temps-réel» et multimédias. Or, ces données n'interviennent que dans le cadre de coopérations synchrones.

3.2. Apartés

La section précédente présentait un modèle pour lequel la structure de coopération peut évoluer et changer dans le temps. En fait, à des moments différents, plusieurs structures de coopération avec différents agents peuvent être utilisées pour réaliser le travail coopératif.

3.2.1. Présentation

On considère une période de temps durant laquelle un ensemble d'agents est en coopération et est organisé suivant une configuration valide dans un certain domaine. A l'intérieur de ce domaine de coopération, il est possible qu'un sous-ensemble d'agents veuille former dynamiquement un *aparté* et échanger des informations privées à l'intérieur de cet aparté [DIAZ94b], [DIAZ94e].

Dans l'approche proposée ici, la structure de cet aparté est contrainte par la coopération en cours, et elle doit suivre cette coopération. Cette structure est donc un sous-ensemble de celle du graphe instantané valide. La coopération donne les relations entre les agents, et les possibilités de communication qu'ils ont entre eux. La structuration de la coopération n'est pas remise en cause par les apartés. Ce choix est explicité dans la section 3.2.2, qui détaille le cadre d'utilisation des apartés. En fait, si l'on considère la représentation sous forme de graphes, la structure de ce nouvel aparté est le sous-graphe du graphe instantané valide en cours, ne comprenant que les agents de l'aparté. La figure II.9 donne un exemple, dérivé de la figure II.6, d'un aparté entre les agents «A6» et «A7», lorsque «A3», «A6» et «A7» sont en coopération et forment une configuration valide.

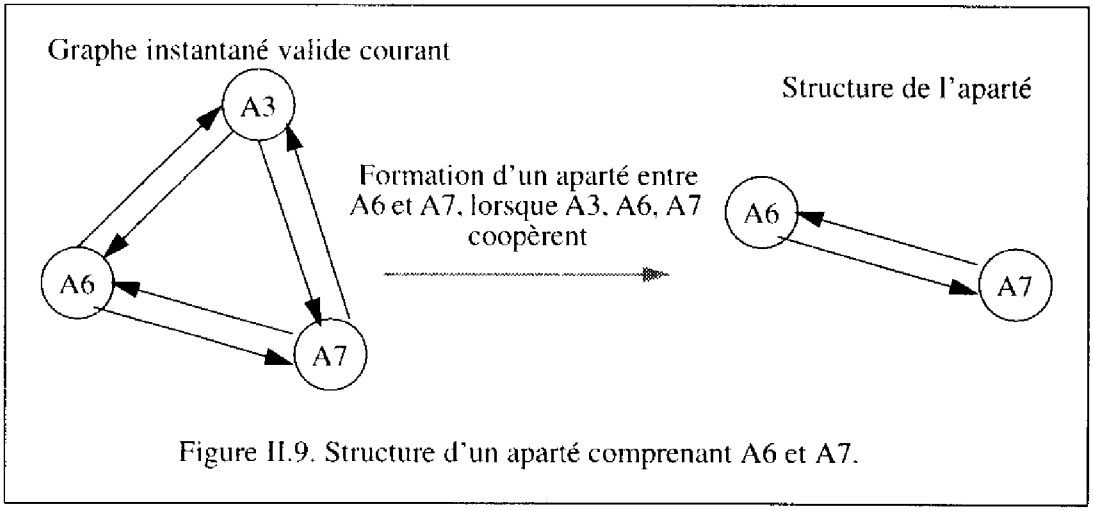


Figure II.9. Structure d'un aparté comprenant A6 et A7.

Le principe retenu pour manipuler les données à l'intérieur des apartés est le même que celui proposé pour la coopération. Chaque agent faisant partie d'un aparté possède un ensemble de données associées à cet aparté dont il est propriétaire et que lui seul peut modifier. Une flèche entre deux agents «Ai» et «Aj» appartenant au même aparté signifie que «Ai» peut lire des données de l'aparté que possède «Aj». De plus, les données partagées dans un aparté sont privées, et ne peuvent pas être connues par des agents n'en faisant pas partie. En fait, si l'on considère les données manipulées à l'intérieur de la coopération et à l'intérieur des apartés, un agent possède des données pour la coopération et des données pour chaque aparté dont il fait partie. Les données de la coopération sont connues de tous les agents avec lesquels il est en relation. De la même façon, les données de chaque agent pour un aparté sont connues par tous les agents avec lesquels il est en relation et qui font partie de l'aparté considéré.

3.2.2. Cadre d'utilisation des apartés

Les apartés ont été définis comme des groupes temporaires si l'on compare leur durée d'existence avec celle du travail coopératif. Par conséquent, des choix ont été faits pour ces apartés. A travers deux exemples d'applications coopératives, les caractéristiques des apartés sont présentées et justifiées.

Une téléconférence

Le premier exemple choisi est celui d'une téléconférence. Un conférencier fait une présentation ou un discours à un ensemble d'auditeurs. En utilisant notre modèle pour représenter des groupes coopératifs, les auditeurs doivent avoir accès aux informations provenant du conférencier. Les auditeurs peuvent librement échanger des remarques ou des commentaires à propos du discours. Par conséquent, chaque auditeur peut échanger des informations avec n'importe quel autre. L'ensemble des auditeurs forme une graphe complet. Pour éviter d'être interrompu ou perturbé dans sa présentation, le conférencier ne peut pas avoir accès aux informations des auditeurs. La structure retenue pour modéliser cette conférence coopérative est donnée par la figure II.10.

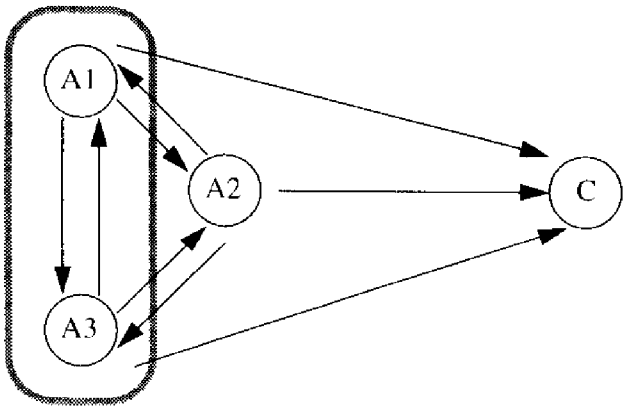


Figure II.10. Un conférencier «C» et trois auditeurs «A1», «A2» et «A3». «A1» et «A3» sont dans un aparté.

Les auditeurs peuvent échanger des informations à l'intérieur de la coopération. Dans ce cas, chaque information montrée par un auditeur est connue par l'ensemble des auditeurs. Avec une telle structure coopérative deux auditeurs ne peuvent pas échanger entre eux une information privée dans la coopération. Mais, dans des assemblées réelles ou dans des conférences, quelques membres peuvent s'échanger en privé de courts messages la plupart du temps écrits sur papier. Ceci peut être réalisé si deux auditeurs utilisent un aparté pour former un forum temporaire et privé. Par exemple, sur la figure II.10, «A1» et «A3» ont créé un aparté pour échanger une courte information non connue de «A2». Les apartés ont été introduits pour réaliser des interactions brèves et dynamiques entre sous-ensembles d'agents coopérants.

Une situation de télé-enseignement

Le second exemple est un groupe coopératif de télé-enseignement. Un enseignant surveille une classe d'étudiants qui font un exercice (ou passent un examen). Les étudiants ne peuvent pas communiquer entre eux pour éviter de copier, mais ils peuvent communiquer avec l'enseignant. A l'opposé, le professeur voit tous les travaux des étudiants. Le schéma coopératif de cette situation est décrit par la figure II.11.

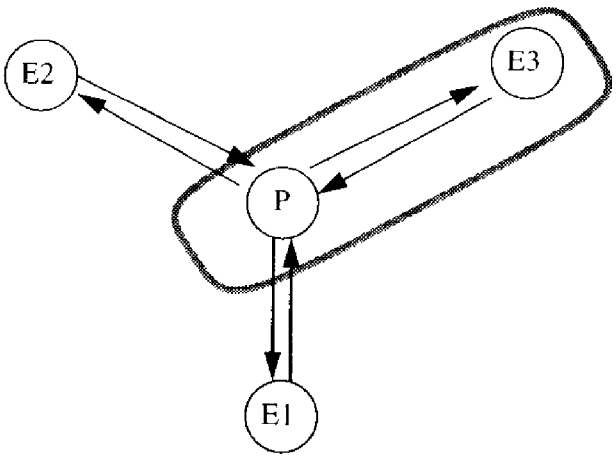


Figure II.11. Un groupe de télé-enseignement composé de trois étudiants «E1», «E2», «E3» et d'un enseignant «P». L'enseignant et «E3» sont dans un aparté.

Si l'étudiant «E3» pose une question au professeur, ce dernier peut répondre à l'intérieur du groupe coopératif. Or, dans ce cas, tous les étudiants vont connaître la réponse du professeur. Cela peut être justifié si la question posée est générale et peut intéresser tous les étudiants. Mais, lorsque le professeur ne désire dialoguer qu'avec l'étudiant ayant posé la question (ou avec un sous-ensemble d'étudiants), il doit créer un aparté avec le sous-ensemble d'étudiants concernés (figure II.11). Ainsi, il évite de distraire et de perturber les étudiants non concernés par son discours. Un aparté a été utilisé dans ce cas pour créer une courte interaction de type question/réponse entre un étudiant et son professeur.

Dans un autre cas, le professeur peut utiliser un aparté pour adresser une remarque ou un commentaire à un ensemble d'étudiants. La formation d'un aparté doit être dynamique et courte pour être adaptée à un bref discours.

3.2.3. Caractéristiques des apartés

Dans les deux précédents exemples illustratifs, les apartés ont été utilisés pour de brèves et courtes interactions entre sous-ensembles d'agents coopérants. L'objectif suivi a été de permettre aux agents coopérants de créer facilement et rapidement des apartés. Les solutions et simplifications retenues pour former ces sous-groupes «allégés» sont maintenant présentées.

Un agent ne peut pas refuser une invitation dans un aparté. Cette simplification a été retenue pour éviter de lourdes négociations, qui nous paraissent injustifiées dans un contexte déjà coopératif, lors de la formation du nouvel aparté. Cependant, un membre invité non intéressé par le fait de participer à cet aparté peut demander à le quitter tout de suite après son entrée : il n'est effectivement pas obligé de partager des informations dans cet aparté.

A l'intérieur du groupe coopératif général, chaque agent possède un ensemble de données qu'il peut partager avec d'autres agents en suivant la structure du graphe coopératif. Lorsqu'un aparté est créé aucune information initiale n'y est attachée. Un agent qui appartient à un nouvel aparté n'a pas de donnée initiale dans cet aparté. Cette simplification évite un échange de contexte initial juste après la création de l'aparté. Cet échange initial de contexte est obligatoire lorsque la structure du groupe coopératif change, i.e. lorsqu'un nouveau graphe instantané valide est utilisé pour poursuivre le travail coopératif. Mais lorsque le nouvel aparté a été créé, ses membres peuvent générer et partager de nouvelles informations à l'intérieur.

Un aparté est détruit lorsque tous ses membres ont demandé à le quitter. Les agents d'un aparté doivent former un consensus pour le quitter effectivement. Cette hypothèse a été retenue car nous avons supposé que la création d'un aparté était faite pour réaliser une tâche commune spécifique entre tous ses membres. Dès que les membres ont fini leur tâche, ils décident tous de quitter l'aparté. Ce choix évite de gérer de lourds problèmes de mise en cohérence des données [DIAZ93d], [DIAZ93e]. En effet, la sortie d'un agent d'un aparté n'est possible que si et seulement si toutes les données des agents de l'aparté sont mises dans un état cohérent. Les échanges de données doivent être synchronisés avec les sorties des agents. Si les sorties étaient faites après chaque requête d'agent, ceci nécessiterait autant de synchronisations que l'aparté contient d'agents.

3.2.4. Domaines et apartés

Les hypothèses choisies peuvent être considérées comme trop restrictives pour des sous-groupes plus généraux. Elles ne peuvent pas s'appliquer ni être justifiées pour des sous-groupes ayant une durée de vie comparable à celle de la coopération, créés à l'intérieur du groupe coopératif.

Les sous-groupes permanents de la coopération sont formés par les domaines de coopération [DIAZ93d], [DIAZ93e]. En effet, ces domaines découpent et structurent la coopération suivant l'ensemble des sous-tâches identifiables dans le travail associé au groupe coopératif. De plus, chacun des domaines possède sa propre logique de comportement. Les prédicats et leurs évaluations se font de façon cohérente dans chaque domaine. Par conséquent, un domaine possède un contexte de données qui lui est associé et qui lui est propre, contrairement aux apartés. Lorsque l'on considère les graphes instantanés de domaine valides, les domaines possèdent une structuration dynamique définie à priori qui peut évoluer au cours du temps. Les apartés suivent uniquement la structuration en cours et, pour éviter tous les lourds changements de contexte liés aux changements de structure, ils ne peuvent pas renégocier leur structuration. Un aparté adopte la configuration de coopération en cours et il ne peut pas jusqu'à sa terminaison changer sa configuration.

Finalement, les sous-groupes détenant la connaissance, ayant leur propre logique de comportement, et ayant une durée d'existence du même ordre que la coopération sont formés par les domaines de coopération.

Les sous-groupes «allégés», n'ayant pas de contexte initial, ni de contexte propre, ni de structuration définie à priori, créés très dynamiquement et de ce fait ayant une durée de vie très courte vis-à-vis de celle de la coopération, sont constitués par les apartés.

Les domaines sont les sous-groupes permanents de la coopération, les apartés ne sont que des sous-groupes temporaires.

3.3. Dépendances de données et communications

La structuration du groupe coopératif due à la relation de partage des informations entre les agents du groupe introduit également des dépendances entre les informations et les connaissances des agents. En effet, un agent coopératif peut posséder des données dont la valeur dépend d'autres données exportées par des agents avec lesquels il est en relation, et ainsi de suite récursivement.

Lorsque l'on cherche à caractériser les données manipulées par la coopération à un instant donné, deux grandes catégories apparaissent :

- Les *données libres* sont les données ne dépendant d'aucune autre donnée de la coopération. Leurs valeurs sont données directement par les agents coopérant.
- Les *données liées* sont les données dont la valeur dépend d'autres données, ces autres données pouvant être locales à un agent ou provenant des données exportées par d'autres agents.

La relation de dépendance des données dépend de la relation **R** de partage d'information, de la structuration du groupe en domaines et aussi de la dynamicité du groupe pour l'appartenance aux domaines et aux apartés. Représentée de manière graphique, cette relation forme un graphe acyclique dont les noeuds racines sont les données libres, les autres sommets étant formés des variables liées. A titre d'exemple, les trois formules suivantes donnent un cas de dépendances de données :

$$y1 = fy1\ (x1, x2)$$
$$y2 = fy2\ (x3, x4)$$
$$z = fz\ (y1, y2)$$

Le graphe des dépendances de cette situation est donné par la figure II.12.

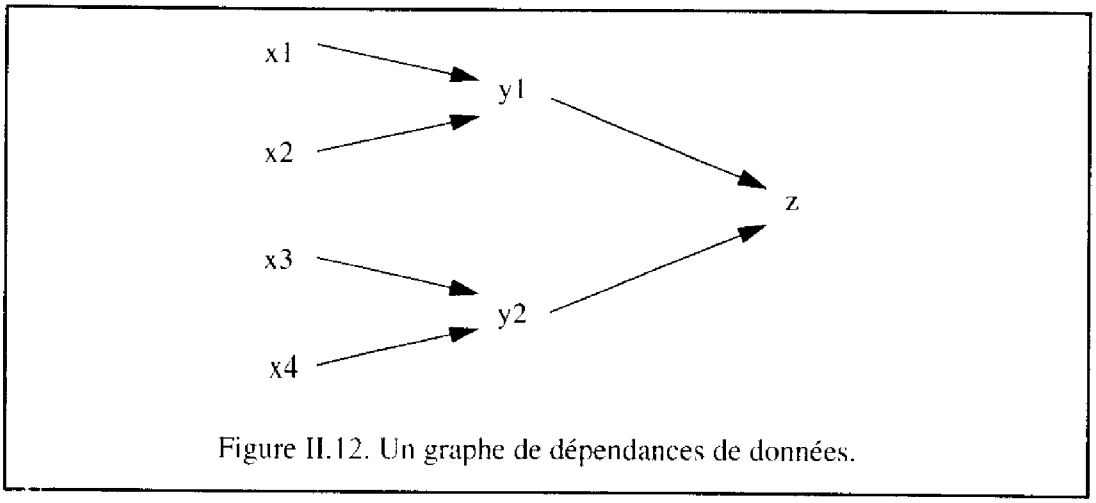


Figure II.12. Un graphe de dépendances de données.

Cette relation va servir de base pour caractériser les communications possibles ou nécessaires lorsque les valeurs des données vont changer et lorsqu'elles doivent être communiquées à d'autres agents. Pour simplifier, on considère un instant pour lequel la relation de dépendance des données est figée, et n'évolue pas dans le temps. Les données modifiables directement par les agents sont les données libres. Lorsqu'un agent modifie la valeur d'une donnée libre, cette modification doit être répercutée vers les agents possédant des données dépendant de la donnée libre initiale. La communication de la nouvelle valeur peut se faire simplement en suivant les flèches du graphe de dépendance, jusqu'aux noeuds feuilles. Dans ce cas, l'arborescence de communication est le sous-graphe du graphe de communication dont la racine est la donnée libre initiale, les autres noeuds de l'arborescence étant les noeuds extrémités des chemins partant de la racine. Chacun des noeuds intermédiaires reçoit la nouvelle valeur du noeud précédent, calcule les nouvelles valeurs de ses données, et rediffuse vers les autres noeuds les valeurs de ses données exportées. Le système de transmission est donc du type «store and forward» pour les noeuds intermédiaires. Sur l'exemple de la figure II.12, lorsque x1 est modifié, l'arborescence de communication est $\{x1 \rightarrow y1 \rightarrow z\}$.

Cependant, cette stratégie de communication n'est pas la seule applicable. En effet, il peut s'avérer coûteux et très pénalisant de passer par les noeuds intermédiaires lors d'une modification. Par exemple, si le graphe de dépendance a un diamètre important, si les coûts de stockage et de retransmission sont élevés, il peut être plus astucieux de diffuser directement la valeur modifiée vers tous les sites qui possèdent des données dépendant directement ou indirectement de la donnée libre modifiée. Dans ce cas, une modification de la variable x1 serait directement diffusée vers les sites possédant y1 et z. L'arborescence de communication obtenue aurait x1 pour racine, les données y1 et z étant les feuilles.

Entre ces deux stratégies extrêmes peuvent s'intercaler un ensemble de possibilités intermédiaires : certains sites peuvent préférer recevoir directement les valeurs des données libres et effectuer tous les calculs intermédiaires pour obtenir les valeurs de leurs données liées propres, d'autres préfèrent recevoir les valeurs des données liées intermédiaires. L'ensemble de ces possibilités intermédiaires peut être déduit du graphe de dépendance des données.

Graphe de communication

Cette partie va montrer comment obtenir à partir d'un graphe de dépendance de données l'ensemble des graphes de communication possibles pour réaliser ces dépendances. Le graphe de communication est un graphe dont les noeuds sont formés des données, les flèches représentant l'envoi d'une valeur lorsque la valeur d'une donnée libre est modifiée. Le premier graphe de communication possible est le graphe de dépendance de données. Les autres graphes s'en déduisent de la façon suivante : lorsque l'on veut réaliser une transmission directe sans passer par un noeud précédent, on détache le noeud considéré du noeud précédent et on le relie aux ancêtres du noeud précédent. Cette transformation peut être appliquée de manière récursive sur les graphes intermédiaires obtenus. Le graphe le plus réduit consiste en un graphe biparti dont les racines sont les données libres, les feuilles étant toutes les données liées. La figure II.13 donne l'ensemble des graphes de communication obtenus à partir du graphe de dépendances de données de la figure II.12.

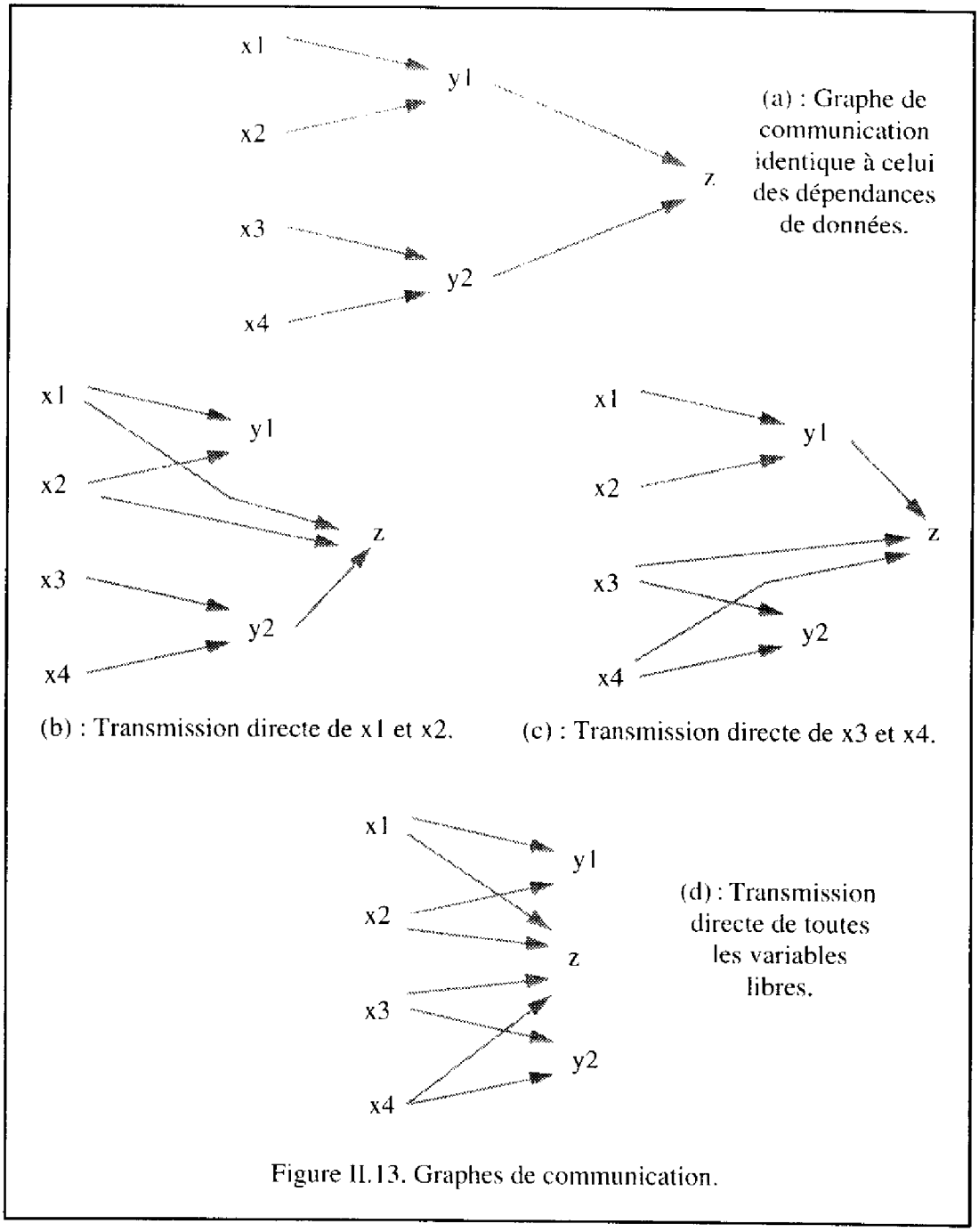


Figure II.13. Graphes de communication.

D'un point de vue plus formel, chacun des cas revient à considérer pour les communications :

$z = fz(y1, y2)$ pour la figure II.13(a)

$z = fz(y1, y2) = fz(fy1(x1, x2), y2) = Fb(x1, x2, y2)$ pour la figure II.13(b)

$z = fz(y1, y2) = fz(y1, fy2(x3, x4)) = Fc(y1, x3, x4)$ pour la figure II.13(c)

$z = fz(y1, y2) = fz(fy1(x1, x2), fy2(x3, x4)) = Fd(x1, x2, x3, x4)$
pour la figure II.13(d)

Tous ces schémas logiques sont équivalents au niveau des valeurs calculées. Cependant, au niveau des performances attendues, ces schémas de communication peuvent se révéler plus ou moins intéressants. Le principal problème va donc être de choisir le schéma qui correspond le mieux aux attentes des coopérants et aux contraintes qui leur sont associés. Dans le cas général, le choix d'un graphe de communication peut dépendre de la granularité et de la sémantique des données échangées, de la fréquence des changements des valeurs, du coût de calcul des dépendances, des services de diffusion plus ou moins efficaces disponibles... Ce choix peut être laissé à la charge des coopérants qui peuvent utiliser des mécanismes décisionnels de type vote ou arbitrage, ou choix pragmatique pour aboutir à un consensus.

Le problème peut encore être complexifié si, pendant l'existence d'un schéma logique des dépendances, les agents coopérants peuvent renégocier et changer dynamiquement de graphe de communication. Un deuxième niveau de complexité peut aussi apparaître lorsque le graphe de dépendances des données évolue dynamiquement dans le temps, soit en ajoutant ou en supprimant des variables libres ou liées, en transformant des variables libres en liées ou vice-versa, en modifiant les relations de dépendances, en suivant l'évolution dynamique de la structure de la coopération... Le graphe de communication subordonné au graphe de dépendances doit alors s'adapter et suivre les transformations.

Conclusion

L'approche formelle basée sur la logique modale fournit une base solide pour la représentation de groupes coopératifs formés d'un ensemble d'agents partageant entre eux des connaissances. Les structures provenant de la logique modale sont en adéquation avec l'organisation logique des groupes coopératifs et fournissent un cadre conceptuel permettant une réflexion à un niveau situé au dessus des couches de communication. De plus, cette organisation fournit un support d'interprétation et d'évaluation des données et prédicats manipulés par la coopération, support pouvant détecter les contradictions ou les ambiguïtés. La levée de ces conflits est cependant laissée à la charge des agents coopérants, le modèle se contentant de signaler les conflits.

Le modèle et ses extensions vont donc servir de base formelle pour la suite des travaux présentés dans ce mémoire. En particulier, il vont servir de cadre à la conception d'un ensemble de services et de protocoles coopératifs. Ces services vont être mis à la disposition de groupes de coopérants et vont permettre de réaliser les coopérations décrites par le modèle formel étendu. Le travail coopératif associé aux groupes d'agents coopérants va ainsi se dérouler sous le contrôle des services et protocoles proposés.

Chapitre III

Services et protocoles de gestion de groupe

Introduction

Le cadre de base pour la conception de services et de protocoles généraux de communication est le modèle de référence en couches OSI («Open System Interconnection») de l'ISO. Ce modèle, dans lequel la conception s'appuie sur des services de communication et est décrite au moyen d'un ensemble d'entités de protocole qui définissent un service de niveau supérieur, est donc composé d'un empilement de couches de services et de protocoles. Le modèle OSI peut être également étendu pour la description de protocoles de coopération car ces couches fournissent conceptuellement une distinction possible entre la coopération et la communication. En effet, les services coopératifs s'appuient sur un niveau de communication sous-jacent.

La conception de systèmes distribués et de protocoles est essentiellement basée sur deux approches : les propriétés et les comportements. La première approche s'appuie le plus souvent sur l'école logique, qui permet de décrire des propriétés telles que la sûreté ou la vivacité et de vérifier que ces propriétés se conservent au cours de l'évolution du système étudié. L'approche comportementale utilise le plus souvent des modèles basés sur les machines à états ou les réseaux de Petri [DIAZ82] qui servent à la définition des langages formels tels que Estelle, SDL. Les descriptions comportementales fournissent cependant des descriptions moins abstraites et de plus bas niveau sémantique que celles basées sur la logique. De ce fait, elles sont plus proches de réalisations finales et peuvent être utilisables plus facilement pour l'implantation.

Le langage Estelle a été choisi pour décrire un système basé sur le modèle OSI, car, par sa conception, ses fonctionnalités s'adaptent bien à ce modèle. De plus, plusieurs études très importantes ont été menées pour son utilisation dans le cadre de la conception de services et de protocoles [COUR87b], [DIAZ89].

A partir du modèle de coopération proposé dans le paragraphe précédent, plusieurs services et protocoles coopératifs ont été spécifiés formellement en Estelle. Le premier service gère l'appartenance dynamique des agents aux groupes coopératifs. Deux améliorations sont ensuite réalisées, avec la prise en compte de déconnexions brutales et leur traitement, puis avec l'adaptation du premier service à des groupes formés de plusieurs domaines.

Le plan de ce chapitre est donc le suivant :

La section 1 présente brièvement le modèle OSI, le langage Estelle et son utilisation pour la conception de services et de protocoles. La section 2 décrit le service d'appartenance dynamique à la coopération et le protocole associé réalisé en Estelle. La section 3 expose le traitement des erreurs, puis donne le nouveau service utilisé ainsi que son protocole spécifié en Estelle. La section 4 montre comment les services de gestion d'appartenance précédents peuvent être utilisés dans le cadre de domaines multiples.

1. Principe de conception de services et de protocoles

Cette partie va présenter succintement le modèle utilisé pour la conception de services et de protocoles de communication, puis décrit le langage Estelle en montrant comment les spécificités du langage s'adaptent à la représentation du comportement du modèle.

1.1. Modèle de référence OSI de l'ISO

Le modèle qui a été retenu et utilisé pour la conception des services et des protocoles proposés est le *modèle de référence OSI* («Open System Interconnection») de l'ISO [ISO7498], [ZIMM80]. Ce modèle, illustré par la figure III.1, décrit le comportement d'un ensemble de sites hétérogènes organisés en réseau, appelés «systèmes ouverts», sous la forme d'un empilement de modules ou couches de protocole, chaque couche définissant un niveau d'abstraction. Sept couches de communication ont été identifiées [ZIMM80], [TANN90] : les couches basses, c'est-à-dire physique, liaison de données, réseau et transport traitent de la fiabilité et de l'acheminement des données tandis que les couches hautes, c'est-à-dire session, présentation et application, s'occupent davantage de la sémantique des données et de leur représentation.

Un *service* de niveau N représente l'abstraction de la couche N et de l'ensemble des couches inférieures sur lesquelles la couche N s'appuie. Un service N est défini par une ensemble de *primitives de service*. Ces primitives servent pour le dialogue entre les utilisateurs du service N et ce service N. Chaque utilisateur est relié au service N au moyen d'un *Point d'Accès au Service* (SAP) qui lui permet d'envoyer et de recevoir des primitives de service N. Un service N n'est pas un bloc monolithique, mais est composé d'un ensemble d'entités de niveau N. Ces entités communiquent et coopèrent entre elles, et leur association fournit le service N. L'ensemble des règles de dialogue et des messages échangés entre entités N forment le *protocole* N. De ce fait, les messages échangés entre les entités N sont appelés *Unités de Données de Protocole* (PDUs). L'ensemble des entités N et le protocole N forment donc une réalisation du service N. Cette réalisation peut bien entendu ne pas être unique. Plusieurs possibilités composées de divers protocoles N peuvent être proposées. Cependant, le service N attendu doit rester le même. Pour communiquer entre elles, les entités N doivent utiliser un service de communication sous-jacent de niveau N-1 accessible au moyen d'un ensemble de primitives de service de niveau N-1.

Le modèle de référence est en fait un modèle «récursif». En effet, les utilisateurs du service N peuvent former un ensemble d'entités de niveau N+1, et ainsi fournir un service de niveau N+1. De même, le service N-1 peut être formé par des entités de niveau N-1, qui communiquent en utilisant un service de niveau N-2.

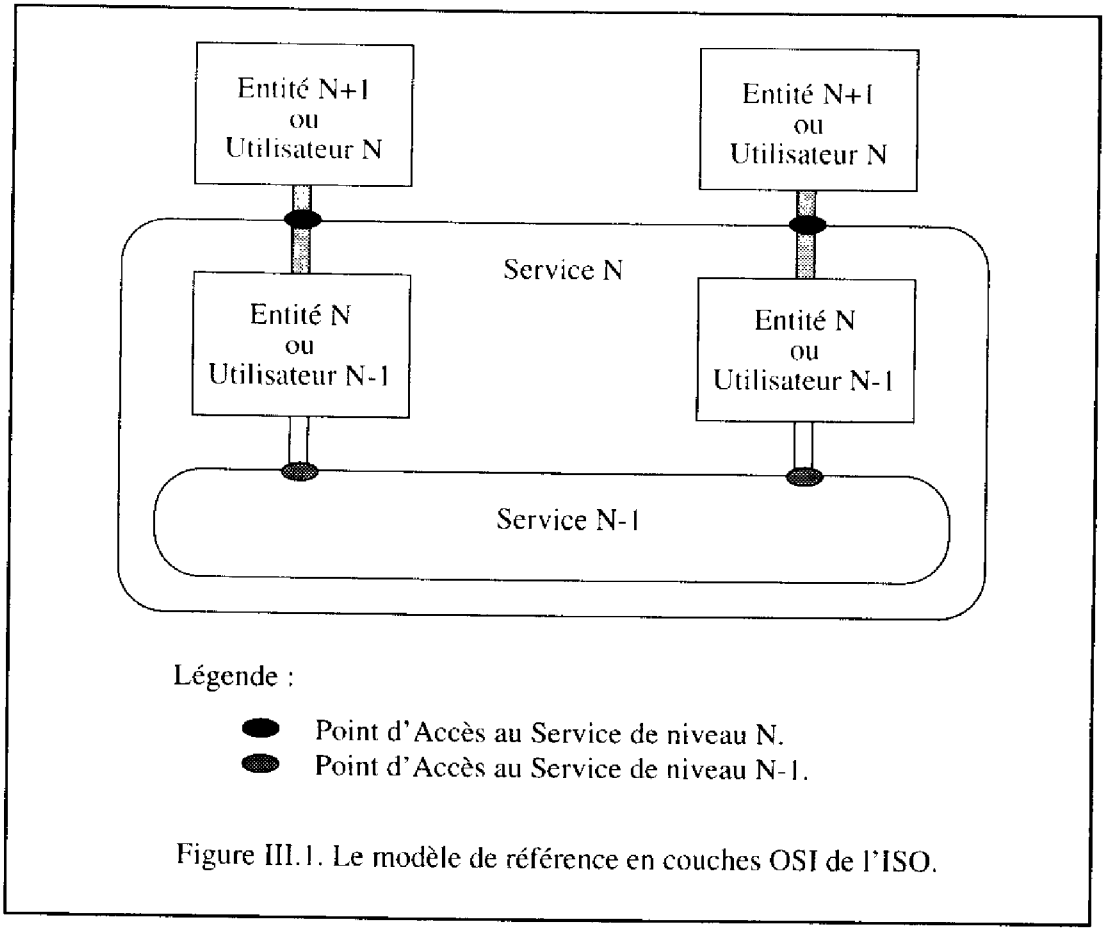


Figure III.1. Le modèle de référence en couches OSI de l'ISO.

1.2. Le langage Estelle

Le langage *Estelle* («Extended State Transition modEL») [ISO9074], [DIAZ89], [COUR87a], [BUDK87] est un langage de description formelle normalisé par l'ISO. Comme nous allons le voir par la suite, ce langage formel se prête bien aux spécifications de services et de protocoles suivant le modèle de référence OSI.

Estelle est basé sur le concept d'une hiérarchie de machines à états qui sont étendues par l'utilisation de files d'attente «First-In-First-Out» et par le langage PASCAL. Brièvement, une spécification Estelle peut être décrite comme un ensemble de modules. Ces modules sont en fait des instances d'un type interface de module associé à un type de comportement. Une spécification Estelle peut donc être dynamique car le nombre d'instances ou le comportement des instances peut changer dans le temps.

Les modules peuvent être emboîtés entre eux. En effet, un module peut contenir un ensemble de sous-modules. Une spécification Estelle forme donc une hiérarchie de modules.

L'interface de module décrit les possibilités de communication externes d'un module. Ces possibilités sont au nombre de deux : la première définit les points d'interaction d'une interface qui permettent aux différentes instances de communiquer entre elles au moyen de canaux. Un canal est une file d'attente bidirectionnelle qui relie deux instances de modules. La structure des messages en transit dans un canal est définie en utilisant le langage PASCAL. Le deuxième mécanisme permet de déclarer des données exportées, visibles par le père de l'instance considérée.

Chaque instance de module contient un ensemble de données et de fonctions ou de procédures PASCAL qui lui est propre et qui n'est visible et accessible qu'à l'intérieur de cette instance. Une instance de module Estelle peut contenir d'autres sous-modules, mais la visibilité des données et fonctions n'est pas étendue vers ces sous-modules.

Le comportement de chacun de ces modules est défini par une machine à états étendue. Cette machine à états est décrite au moyen d'un ensemble de transitions de la forme :

<Condition/Action>

où le premier terme donne la condition de sensibilisation de la transition, parfois aussi appelée sa garde. En Estelle, cette condition peut être l'attente d'arrivée d'un message particulier et/ou un test booléen sur des variables internes ou exportées du module. La partie «Action» donne l'ensemble des actions réalisées de façon atomique lorsque la transition est tirée. Ces actions peuvent être essentiellement des envois de messages dans les files d'attente associées aux points d'interaction du module, des modifications des valeurs de variables internes ou exportées, des créations ou des destructions dynamiques de sous-modules.

1.3. Conception Estelle de services et de protocoles

Le modèle de référence OSI va servir de base pour définir l'architecture de spécifications Estelle destinées à la conception de services et de protocoles (figure III.2). Typiquement, le service de communication sous-jacent est représenté au moyen d'un module Estelle. Chacune des entités définissant le protocole devient également un module Estelle, ainsi que chacun des utilisateurs du service spécifié. Les entités sont reliées au service de communication au moyen des canaux Estelle. Les utilisateurs sont également connectés aux entités grâce aux canaux Estelle. Les primitives de service sont définies comme les messages circulant dans les canaux. Il faut donc définir deux types de primitives : celles du service de communication et celles du service de niveau supérieur proposé. Les unités de protocole échangées entre entités sont spécifiées comme des unités de données de service du service de communication.

Remarque 1 : Pour être tout à fait conforme au modèle de référence OSI, le service à réaliser, représenté par un cadre arrondi sur la figure III.2, pourrait être constitué d'un module Estelle. Cependant, ce module serait quelque peu superflu car son seul rôle serait de contenir les modules entité et le service de communication sous-jacent. Ce module n'aurait aucun comportement propre et il ne servirait que de contenant aux autres modules. Ceci est lié au fait que le service est défini à lui seul par l'ensemble des entités, et qu'il représente une vue abstraite du comportement de l'ensemble des entités.

Remarque 2 : D'autres avantages prédisposent à l'utilisation d'Estelle pour la conception de services et de protocoles de communication. Tout d'abord, le formalisme utilisé, les machines à états, est très connu. Le langage PASCAL est aussi relativement utilisé et fait partie des langages algorithmiques courants. Il existe également des environnements de simulation, de mise au point, voire de vérification performants : citons à ce titre ESTIM [COUR92], EDT [EDB92], et VEDA [VEDA91]. Enfin, en dernier lieu, certains environnements permettent après la simulation ou la vérification de réutiliser une partie du code Estelle, ceci afin de l'incorporer dans une implantation réelle.

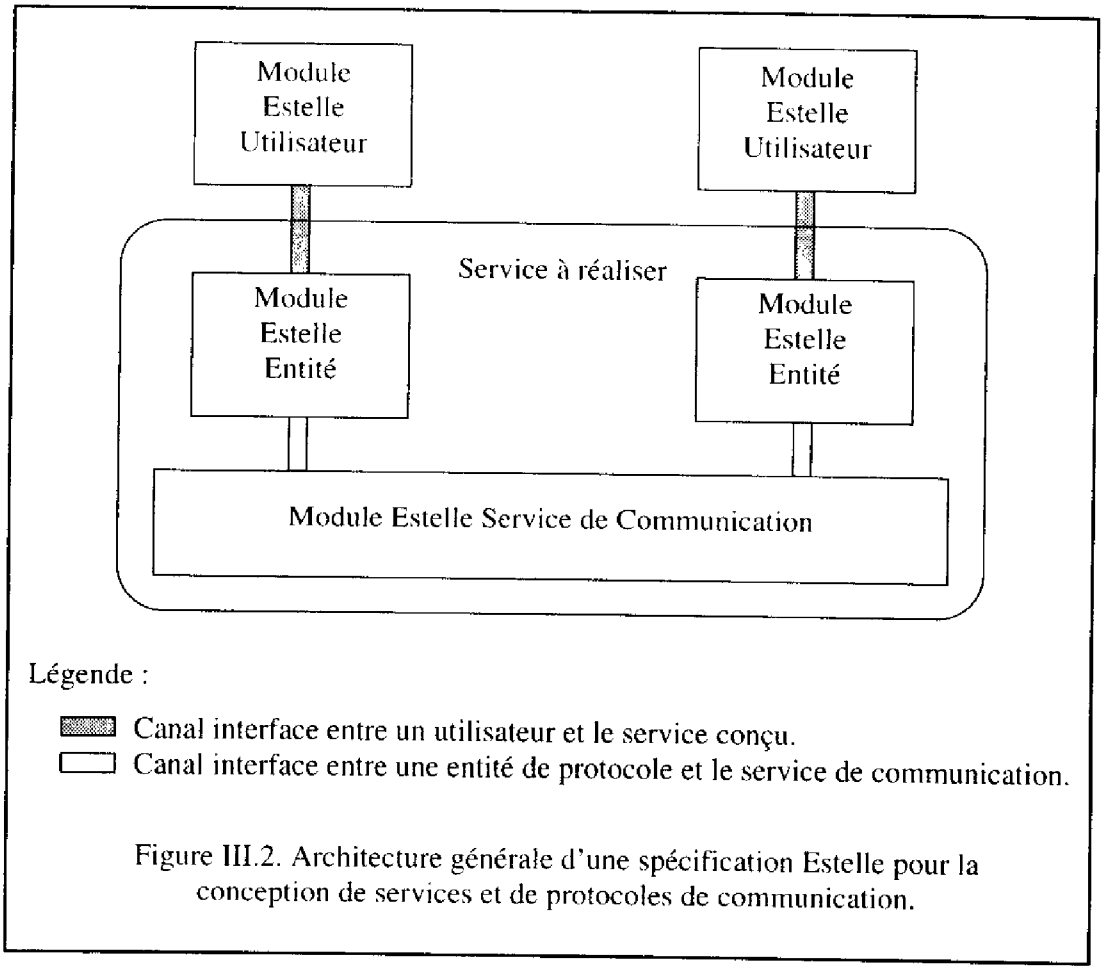


Figure III.2. Architecture générale d'une spécification Estelle pour la conception de services et de protocoles de communication.

2. Services et protocoles d'appartenance dynamique

Les services et protocoles proposés dans cette section permettent de gérer des groupes coopératifs dynamiques dont la structure évolue dans le temps [DIAZ93d], [DIAZ93e], [BAUD93], [TREH93a], [TREH93b], [DIAZ94c]. Suivant l'entrée et la sortie de la coopération des agents, la configuration courante de la coopération est modifiée.

Le service présenté ne couvre pas l'ensemble des cas définis par le modèle de coopération décrit précédemment, mais ne s'applique que si le groupe coopératif n'est composé que d'un seul domaine de coopération. Il sera ensuite étendu au cas de plusieurs domaines.

2.1. Service pour gérer des coopérations dynamiques

Le groupe ne comportant qu'un seul domaine n'est donc formé que d'un seul graphe de domaine. Par contre, un ensemble de graphes valides définis par l'application sont associés à la coopération.

2.1.1. Description du service

Le service défini contrôle l'évolution de la coopération dans le temps. Il permet de passer d'une configuration valide dans laquelle les agents sont en coopération vers une autre configuration en prenant en compte les demandes d'entrée et de sortie de la coopération provenant des agents. A un instant donné, nous supposons qu'un ensemble d'agents forme un graphe instantané valide et réalise un travail coopératif. Il est possible que certains agents veuillent quitter le travail coopératif et que d'autres agents souhaitent se joindre à la coopération. Le service analyse les demandes d'entrée et de sortie de la coopération et forme une nouvelle configuration en prenant en compte ces demandes.

Plusieurs possibilités sont envisageables pour la décision de changement de configuration. La première est de changer de configuration chaque fois que cela est possible. Le système gérant la coopération réalise la modification dès que possible. La deuxième possibilité qui a été retenue ici est de considérer la décision de changement de coopération à la fois comme une contrainte de l'application et comme une décision coopérative. Ainsi, le nouveau graphe n'est considéré comme acceptable que s'il constitue une configuration valide : le service propose alors cette nouvelle configuration à l'ensemble des agents en coopération. Les agents coopérants votent [GARC85], [HAHN89], [CHEU90], [AHAM91], [KUMA91], [NEIL92] pour accepter ou refuser de changer de configuration. Chaque agent en coopération vote «oui» s'il accepte le changement ou alors vote «non» s'il refuse. Il faut un quorum de voix «oui» pour changer de coopération. Le principe du vote est défini par l'application. Par exemple, ce principe peut être le vote à la majorité, le vote à l'unanimité, etc... Finalement, suivant le résultat du vote, le service change ou non de configuration valide (figure III.3).

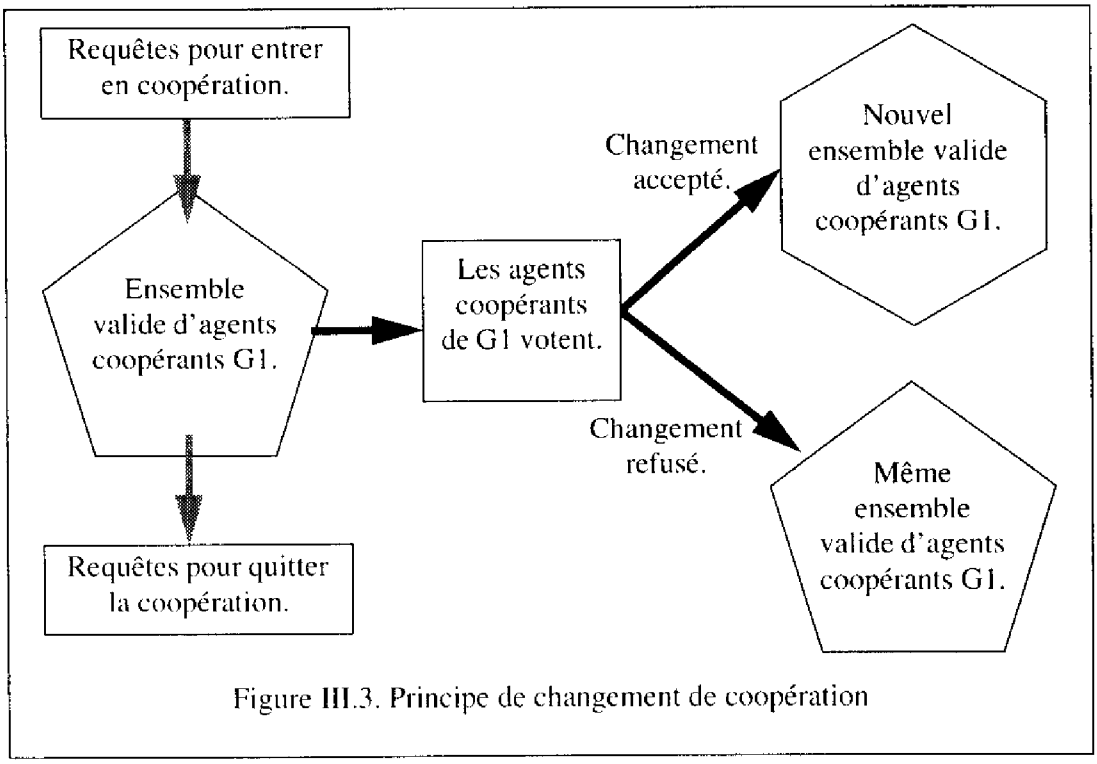


Figure III.3. Principe de changement de coopération

2.1.2. Fonctionnement du service

Pour décrire le comportement des entités coopérantes, quatre états peuvent être définis pour chacun des agents du groupe coopératif :

- IDLE : L'agent considéré ne coopère pas.
- WAIT_FOR_COOP : L'agent désire participer au travail coopératif.
- COOP : L'agent participe au travail coopératif et ne veut pas abandonner son travail.
- WAIT_FOR_REL : L'agent participe actuellement au travail coopératif, mais il souhaite abandonner le travail dès que possible.

A chaque instant, le graphe instantané de domaine valide est formé des agents dans l'état COOP et des agents dans l'état WAIT_FOR_REL ; les agents ne faisant pas partie de la configuration courante sont soit dans l'état IDLE soit dans l'état WAIT_FOR_COOP (figure III.4).

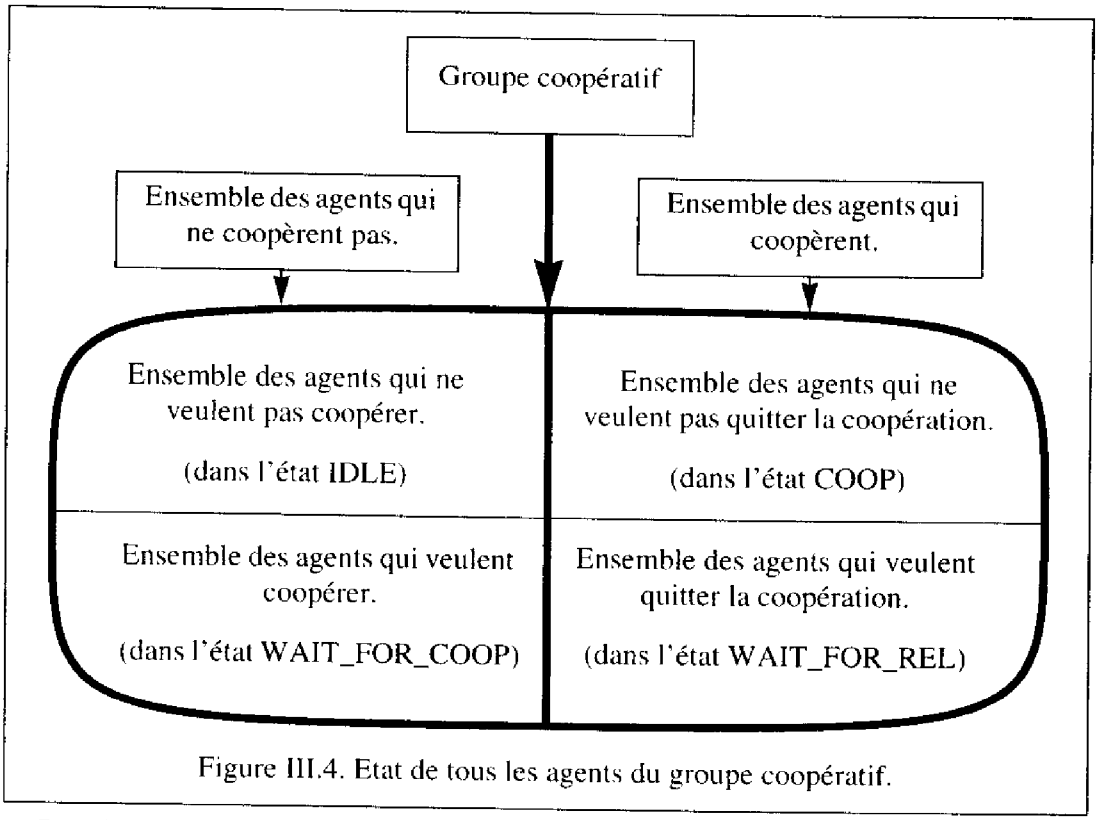


Figure III.4. Etat de tous les agents du groupe coopératif.

Parmi le groupe d'agents coopératifs, le service regarde s'il est possible de former un nouveau graphe instantané de domaine valide : pour cela, il ajoute à la coopération des agents dans l'état WAIT_FOR_COOP, garde tous les agents dans l'état COOP, enlève des agents dans l'état WAIT_FOR_REL et vérifie les règles d'obtention d'un nouveau graphe.

Si aucun nouveau graphe instantané de domaine valide ne peut être formé en ajoutant des agents dans l'état WAIT_FOR_COOP ou en enlevant des agents dans l'état WAIT_FOR_REL, alors le service attend que d'autres agents modifient leur état pour essayer d'obtenir une nouvelle configuration valide. Les sites ayant demandé d'entrer en coopération sont alors bloqués en attente d'autres modifications. Les sites coopérants ayant demandé à sortir sont également en attente d'autres modifications, mais ils continuent le travail coopératif sans être bloqués.

Supposons cependant qu'un nouveau graphe instantané soit possible. Dans ce cas l'ensemble des agents coopérants est appelé à voter pour savoir s'ils acceptent le changement ou non. Si ce changement est accepté, alors les agents du nouveau graphe en attente de coopération entrent dans cette dernière (de l'état WAIT_FOR_COOP, ils passent dans l'état COOP) et les agents qui voulaient arrêter de coopérer et qui sont exclus du nouveau graphe quittent la coopération (ils passent de l'état WAIT_FOR_REL dans l'état IDLE). Après le vote, si le changement est refusé, la structure courante de coopération reste la même. Dans ce cas, les agents qui désiraient quitter la coopération et qui n'appartenaient pas au nouveau graphe sont obligés de rester en coopération (de l'état WAIT_FOR_REL, il repassent à l'état COOP) et les agents du nouveau graphe qui désiraient entrer en coopération se voient refuser leur entrée (de l'état WAIT_FOR_COOP ils repassent à l'état IDLE).

Exemple

Afin de clarifier la description, l'exemple suivant a été choisi :

Le groupe coopératif est formé de trois agents : 1, 2 et 3. La structure de la coopération, donnée par la figure III.5, est un graphe complet pour lequel les agents sont tous connectés entre eux. Un seul domaine compose ce groupe coopératif. Le critère retenu pour définir les graphes instantanés valides est que le travail coopératif ne peut avoir lieu que si au moins deux agents sont présents en coopération. Les votes pour les changements de coopération s'effectuent à la majorité.

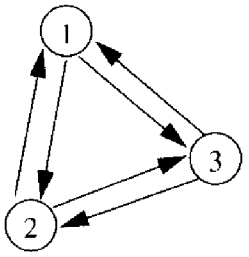


Figure III.5. Exemple de graphe coopératif.

A un instant donné, l'agent 1 ne coopère pas et se trouve donc dans l'état IDLE tandis que les agents 2 et 3 coopèrent et sont donc dans l'état COOP. Cette situation est décrite par la figure III.6(a).

Quelque temps après, l'agent 2 demande à quitter la coopération (figure III.6(b)). On ne peut pas changer de configuration car la configuration où l'agent 3 resterait seul en coopération n'est pas valide. Par la suite, se produit la situation de la figure III.6(c) où l'agent 1 demande à entrer en coopération. Dans ce cas, un changement de configuration est possible. L'agent 2 pourrait quitter la coopération tandis que l'agent 1 y entrerait. Les agents 2 et 3 votent donc pour savoir s'ils acceptent le changement. Implicitement, l'agent 2 vote «oui» car il désire quitter la coopération et car il ne fait pas partie de la nouvelle configuration proposée. Si le vote est positif, le changement est accepté et le nouveau schéma valide de coopération obtenu est celui de la figure III.6(d). Dans la négative, le changement est refusé, l'entrée de l'agent 1 est refusée, de même que la sortie de l'agent 2, et la configuration donnée par la figure III.6(a) est obtenue de nouveau.

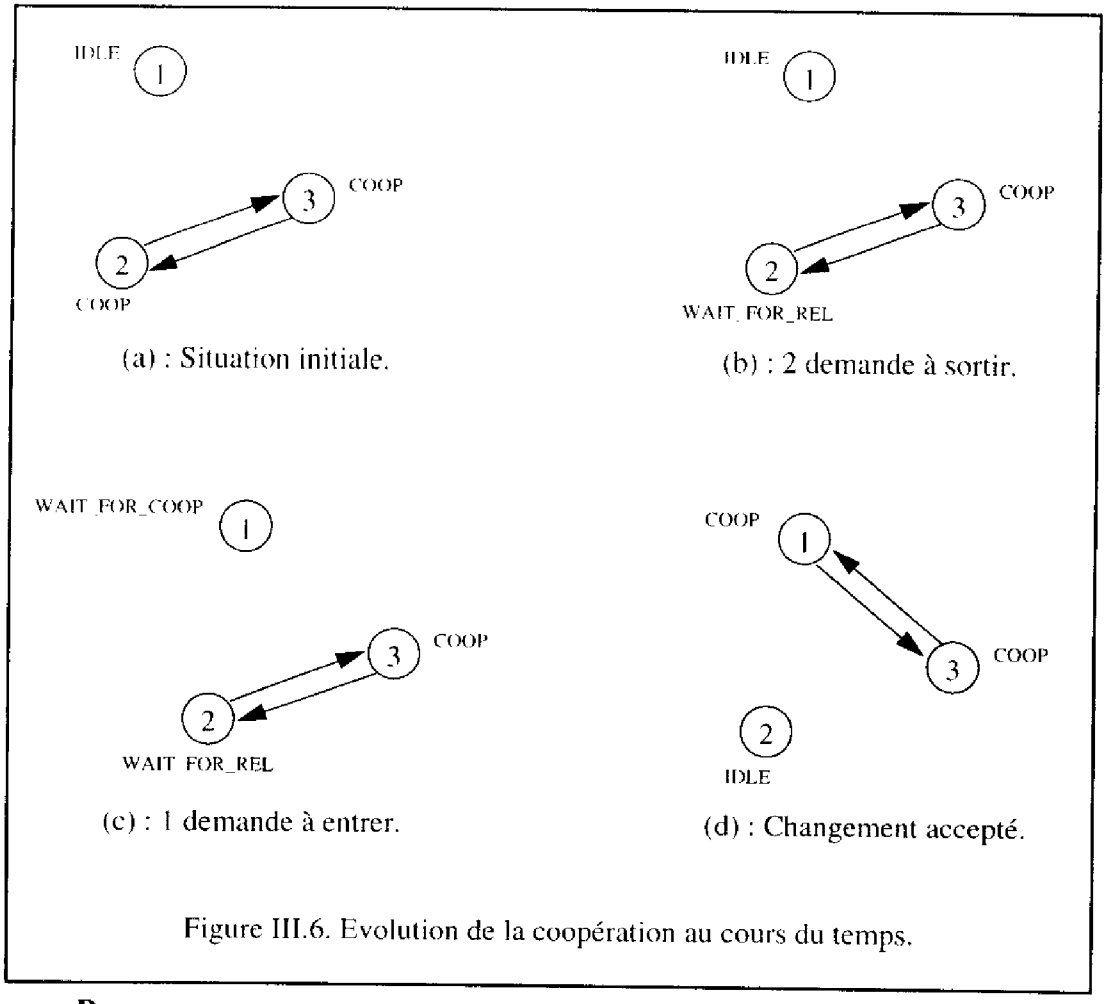


Figure III.6. Evolution de la coopération au cours du temps.

Remarque :

Le service décrit propose à chaque étape un seul nouveau graphe instantané valide pour changer de configuration. Une autre possibilité plus générale serait de proposer tous les graphes instantanés valides possibles lors d'un changement, le cas où l'on ne changerait pas de configuration faisant partie de l'ensemble des graphes proposés. Cette hypothèse n'a pas été retenue car le nombre de possibilités peut facilement devenir très élevé, lorsque le nombre d'agents coopérants est important. De plus, le service cherche le plus possible à correspondre aux desiderata de chacun des agents. Le graphe qu'il propose lors d'une modification est celui qui laisse entrer le plus possible d'agents ayant demandé à coopérer, et qui fait sortir le plus possible d'agents qui le désirent, tout en gardant une configuration valide. La décision finale de changement appartient cependant à l'ensemble des coopérants.

2.1.3. Cohérence des données

Jusqu'à présent, l'échange des données entre les agents réalisant le travail coopératif n'a pas été pris en compte. Or, chaque agent est propriétaire d'un ensemble de données. Lorsqu'un agent coopérant modifie ses données, il doit communiquer les nouvelles valeurs aux agents qui ont connaissance de ces données, c'est-à-dire les agents coopérants qui le précèdent dans le graphe. Cet envoi nécessite des mécanismes de diffusion («multicast») pour transmettre les nouvelles valeurs des données entre les agents du graphe de coopération instantané. Cependant, il apparaît clairement que les modifications de la structure de la coopération peuvent interférer avec les échanges de données et peuvent créer des incohérences. En effet, lorsque des valeurs des données sont en transit dans le réseau, il existe un état incohérent dans un même domaine, car les agents n'ont pas la même valeur pour des données identiques. Par conséquent, le service doit empêcher tout changement de structure coopérative lorsque les données ne sont pas dans un état cohérent. De plus, le service doit obliger les agents coopérants à mettre leurs données dans un état cohérent lorsqu'un changement de coopération est en attente.

Cohérence avant de modifier la coopération

Avant de terminer une coopération, i.e. avant de changer de structure coopérative, les données manipulées par les agents doivent se trouver dans un état cohérent. L'hypothèse retenue par ce service est qu'un état cohérent est obtenu lorsque tous les échanges de données entre les agents coopérants sont terminés. Pour cela, le service ordonne aux agents en coopération de suspendre les transmissions de données. Il attend que toutes les données en transit sur le réseau soient bien parvenues aux agents destinataires. De ce fait, les agents connaissent les mêmes valeurs des diverses données. Cette hypothèse peut être considérée comme minimale. En effet, d'autres contraintes pourraient être ajoutées comme par exemple des contraintes d'intégrité similaires à celles des bases de données. Dans ce cas non considéré dans notre domaine d'application, les données devraient respecter les règles d'intégrité avant de pouvoir être considérées comme cohérentes.

Cohérence lors d'une nouvelle coopération

Lorsqu'un changement de structure de coopération a été décidé, les nouveaux agents coopérants doivent recevoir des informations initiales formant le contexte de chaque agent avant de commencer cette nouvelle coopération. Les nouveaux coopérants doivent mettre à jour mutuellement leurs contextes ou données propres en suivant la structure de la coopération. En effet, au départ, les nouveaux arrivants ne connaissent pas le contexte des autres coopérants avec lesquels ils sont en relation, c'est-à-dire les agents qui les suivent dans le graphe de coopération. De même, les agents qui restent en coopération doivent connaître les contextes des nouveaux arrivants. Le service doit donc attendre la fin de ces échanges de valeurs initiales avant d'autoriser la reprise du travail coopératif.

Phases d'évolution de la coopération

Lorsque l'on combine les échanges de données avec les changements de structure coopérative, quatre phases principales apparaissent. Ces phases sont présentées sur la figure III.7.

- Lorsqu'une nouvelle coopération démarre, i.e. lorsqu'une nouvelle structure coopérative est acceptée, les nouveaux coopérants échangent entre eux leurs contextes propres. Dès que cet échange est terminé, le service indique aux agents coopérants de commencer la coopération.
- La seconde phase est celle de la réalisation du travail coopératif, où les agents communiquent et échangent entre eux des informations.

- Lorsqu'un nouveau schéma de coopération a été accepté après un vote positif, le service ordonne aux agents coopérants de mettre leurs données dans un état cohérent et de suspendre les transmissions de nouvelles valeurs. Le service attend jusqu'à ce que les données en transit soient arrivées. Ces derniers envois sont signalés au moyen de paquets spéciaux.
- La dernière phase est celle du changement de structure de coopération.

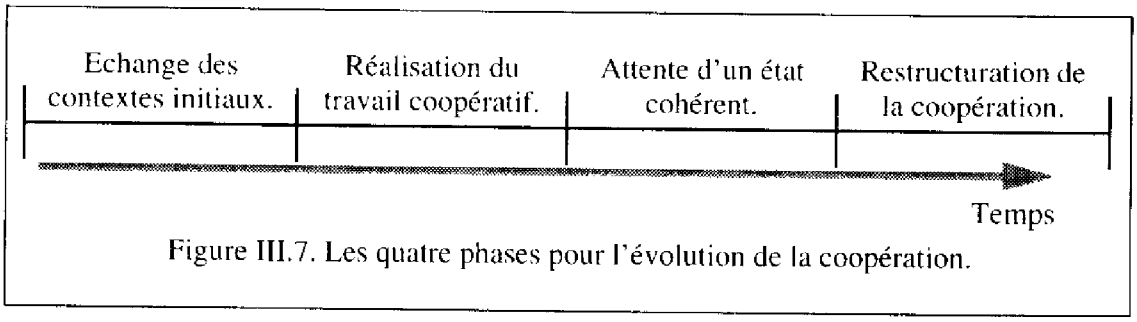


Figure III.7. Les quatre phases pour l'évolution de la coopération.

2.1.4. Primitives de service

Ce paragraphe donne l'ensemble des primitives de service qui nous ont parues nécessaires pour définir un service d'appartenance dynamique à un groupe coopératif. Ces primitives sont regroupées par unités fonctionnelles sur la figure III.8.

Les primitives COOP_REQ et COOP_IND de la première unité fonctionnelle sont utilisées par un agent non coopérant pour entrer en coopération, COOP_REQ servant à effectuer la demande, COOP_IND donnant la réponse.

De la même façon, les primitives RELEASE_REQ et RELEASE_IND de la deuxième unité fonctionnelle permettent aux agents en coopération de quitter cette dernière, RELEASE_REQ étant la demande de sortie, RELEASE_IND indiquant la réponse.

Les primitives VOTE_IND et VOTE_RESP sont utilisées par les agents en coopération lors d'un vote. VOTE_IND indique aux coopérants le vote à effectuer, VOTE_RESP donne le vote de chaque coopérant.

La primitive NEW_GROUP_IND donne le résultat du vote aux agents en coopération.

Les primitives DATA_REQ et DATA_IND servent au transport des données entre les coopérants soit pour l'échange des contextes initiaux, soit pour la réalisation du travail coopératif en phase de coopération.

Les primitives NO_NEW_DATA_IND et DT_COHERENTES_IND de la dernière unité fonctionnelle interviennent dans les phases de mise en cohérence des données et d'échange des contextes initiaux, la première primitive indiquant aux coopérants le début de la mise en cohérence, la seconde leur signalant que les données sont de nouveau cohérentes après les échanges des contextes initiaux.

Figure III.8. Primitives de service.

Nom et paramètres des primitives.	↘ Utilisateur vers Service. ↗ Service vers Utilisateur.	Rôle de la primitive.
COOP_REQ (domain: domain_type)	↘	Un agent demande à joindre le domaine de coopération.
COOP_IND (domain: domain_type; result: boolean; pure_receiver: boolean; list: list_site_type)	↗	Cette indication signale si l'entrée en coopération est acceptée ou pas. Si elle est acceptée, «list» représente la liste des agents coopérants dans le graphe instantané valide courant. Le booléen «pure_receiver» est mis à faux lorsque l'agent considéré envoie des données vers d'autres agents du domaine.
RELEASE_REQ (domain: domain_type)	↘	Un agent demande à quitter la coopération.
RELEASE_IND (domain: domain_type; result: boolean)	↗	Cette primitive indique si la demande de sortie de la coopération a été acceptée ou pas.
VOTE_IND (domain: domain_type; new_sites: list_site_type)	↗	Un nouveau schéma instantané comprenant les agents «new_sites» est possible et les agents présentement en coopération doivent voter pour savoir s'ils acceptent ou non le nouveau schéma.
VOTE_RESP (domain: domain_type; result: boolean)	↘	Un agent répond s'il accepte ou non la modification du schéma de coopération courant.
NEW_GROUP_IND (domain: domain_type; result: boolean; pure_receiver: boolean; new_sites: list_site_type)	↗	Cette primitive signale aux agents en coopération si la modification du schéma instantané a été acceptée ou pas. Si elle est acceptée, «new_sites» représente la liste des agents coopérants dans le graphe instantané valide courant. Le booléen «pure_receiver» est mis à faux lorsque l'agent considéré envoie des données vers d'autres agents du domaine.
DATA_REQ (domain: domain_type; data: data_type)	↘	Un agent demande à envoyer une unité de donnée.
DATA_IND (domain: domain_type; data: data_type)	↗	Un agent reçoit une nouvelle unité de donnée.
NO_NEW_DATA_IND (domain: domain_type)	↗	Les agents coopérant actuellement ne doivent plus envoyer de nouvelle donnée.
DT_COHERENTES_IND (domain: domain_type)	↗	Cette primitive indique que toutes les données des agents coopérants sont dans un état cohérent et que le travail coopératif peut reprendre et continuer.

2.2. Description formelle du protocole

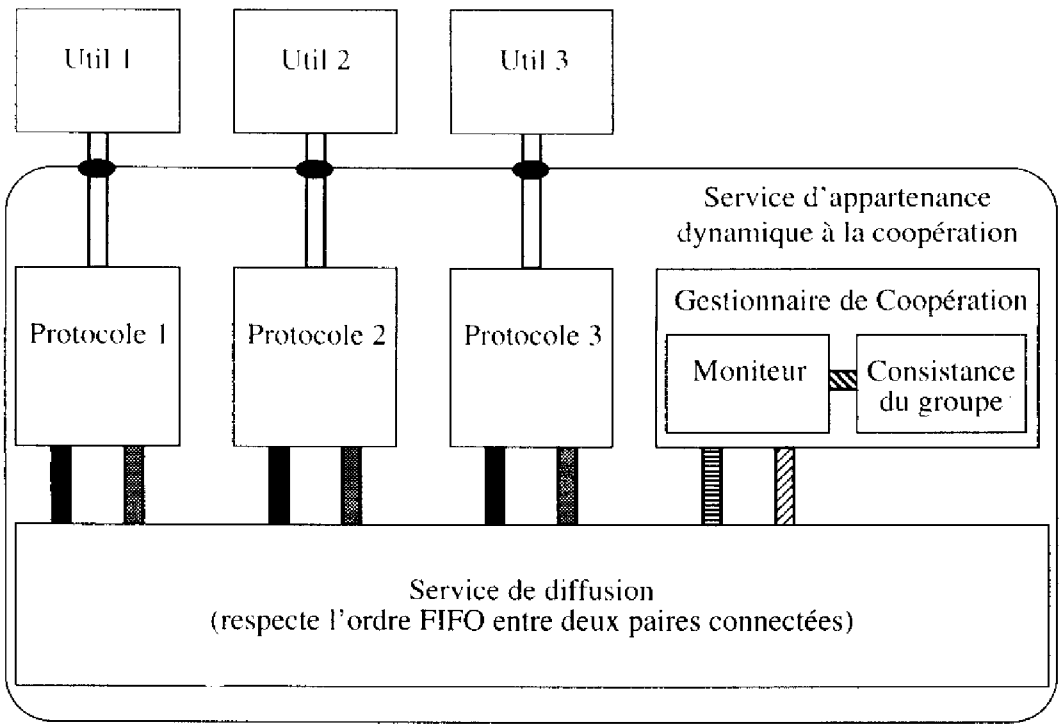
Le service proposé a été décrit formellement en utilisant la technique de description formelle Estelle. Cette spécification a été utilisée pour simuler et tester notre service.

Le simulateur et le «debugger» Estelle [EDB92] de l'environnement de programmation «Estelle Development Toolset» (EDT) ont été utilisés pour développer et simuler la spécification que cette section présente.

2.2.1. Description de la spécification

L'architecture de la spécification, présentée sur la figure III.9, est basée sur le modèle de référence en couches OSI. Chaque utilisateur est connecté au service d'appartenance dynamique à la coopération au moyen des files Estelle. Ces connexions sont des Points d'Accès au Service (SAPs) de la terminologie OSI. Les communications entre le service et les utilisateurs se font en utilisant les primitives de service définies dans la section 2.1.4. Le service d'appartenance dynamique à la coopération est composé de plusieurs modules Estelle. Il contient un ensemble de modules appelés «Protocole i», chacun connecté à un utilisateur, et un module gestionnaire de coopération. De plus, ces modules sont connectés à un service de niveau inférieur qui permet la diffusion de messages. Les communications entre modules, i.e. les échanges d'Unités de Données du Protocole (PDUs) sont faites en utilisant le service de diffusion sous-jacent.

Dans ce protocole, le contrôle global de l'évolution de la coopération se fait de manière centralisée et les contrôles des échanges de données sont répartis entre les agents. Toutes les requêtes pour l'entrée et la sortie de la coopération sont centralisées et contrôlées par un gestionnaire de coopération. Par contre, les échanges de données sont distribués en respectant la structure de la coopération. Cette conception permet de séparer les rôles. Chacun des agents coopérants est ainsi responsable des données qu'il véhicule et se charge de leur acheminement. Le gestionnaire a pour rôle de contrôler l'évolution dans le temps de la structure de la coopération et lui seul sera à modifier si la politique de contrôle de cette dernière change.



Légende :

- Canal pour les informations de contrôle de la coopération.
- Canal pour la transmission des données du travail coopératif.
- Canal interface entre un utilisateur et le service.
- Canal pour recevoir les requêtes pour entrer ou sortir de la coopération.
- Canal pour gérer l'évolution de la coopération.
- Canal interface entre le moniteur et le module «consistance du groupe».
- Point d'accès au service d'appartenance dynamique à la coopération.

Figure III.9. Architecture OSI de la coopération.

Les modules «Util i» sont les utilisateurs du service et ils représentent les agents du groupe coopératif. En utilisant les primitives du service d'appartenance, chaque utilisateur envoie à son module protocole respectif des requêtes pour joindre ou pour quitter le travail coopératif, envoie ou reçoit des données des autres agents en coopération et participe au processus de vote.

Les modules «Protocole i» fournissent le service défini. Chaque module transmet les requêtes pour rejoindre ou pour quitter le travail coopératif au gestionnaire de coopération. Ils se transmettent entre eux les données utilisées par le travail coopératif et gèrent les conflits de cohérence entre les données et les changements de structure coopérative. Finalement, chaque module envoie au gestionnaire de la coopération la réponse de l'utilisateur correspondant lors d'un vote.

Le module «Gestionnaire de coopération» gère l'arrivée dynamique des agents du groupe. Sa définition est basée sur le fait que la sélection des graphes instantanés valides, définie par l'application, ne peut pas être modifiée par les utilisateurs pendant le déroulement de la coopération. Par conséquent, les critères de sélection ont été placés dans ce module. Ce module reçoit

aussi les requêtes provenant des modules protocole pour joindre ou pour quitter le travail coopératif. Il envoie les ordres vers les modules protocole pour arrêter leurs échanges de données et pour mettre leurs données dans un état cohérent. Il gère le processus de vote en indiquant aux modules protocole si un changement de coopération a été accepté ou non. Dans l'affirmative, il donne l'ordre aux nouveaux agents coopérants d'échanger leur données initiales avant de reprendre le travail coopératif. Le gestionnaire de coopération est composé de deux modules : le moniteur et le module de consistance du groupe. Ces deux modules communiquent entre eux au moyen d'une file Estelle. Le moniteur contient l'état courant des agents du groupe coopératif. Le module de consistance du groupe contient les conditions de validité des graphes instantanés. Ces critères ont été séparés du moniteur pour renforcer la modularité de la spécification. Ainsi, pour changer de graphes instantanés valides, il suffit de modifier le module de consistance du groupe. De plus, si ce module est considéré comme un objet, plusieurs méthodes pourraient fournir différentes règles de validité.

Le service de diffusion, défini dans la spécification par un module Estelle, possède plusieurs propriétés. Tout d'abord, il doit respecter un ordre FIFO entre les paires connectées, i.e. un message transmis par une entité ne doit pas dépasser les messages précédents émis par cette même entité. De plus, les PDUs gérant l'évolution de la coopération ne doivent pas être perdus par le réseau. Par conséquent, un service de diffusion fiable est requis pour ces messages. Pour les échanges de données, le protocole peut supporter des pertes de PDUs mais, pour obtenir la cohérence des données lors d'un changement de coopération, les échanges de données finals doivent être fiables. De même, les échanges de données initiaux lors de l'échange des contextes doivent se faire sans perte. Les entités sont informées des derniers échanges de données par des paquets spéciaux transmis de façon fiable, provenant des entités les précédant dans le graphe de coopération, après que l'ordre de mise en cohérence des données ait été pris en compte par ces entités. Le protocole ne requiert donc pas un service fiable de façon continue, en ce qui concerne les échanges de données.

2.2.2. Exemple de coopération

Plusieurs cas et plusieurs configurations ont été considérés pour implanter les simulations et tester le protocole. A titre d'exemple, la figure III.10 donne un groupe coopératif utilisé dans un des tests effectués. Le graphe de coopération ne comporte qu'un seul domaine et contient trois agents. Le critère retenu pour définir les graphes instantanés valides est que le travail coopératif ne peut avoir lieu qui si au moins deux agents sont présents en coopération. Le système de vote choisi est le vote majoritaire. La description de cet exemple nécessite environ 1800 lignes de code Estelle.

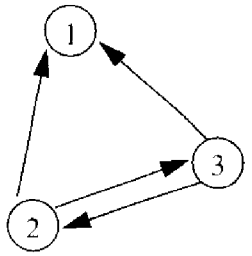


Figure III.10. Graphe coopératif utilisé par une spécification.

3. Service d'appartenance traitant les erreurs

Cette partie présente une amélioration du service d'appartenance dynamique à une coopération lorsque certaines erreurs surviennent lors du déroulement de la coopération, à l'intérieur du groupe coopératif. Les erreurs et défaillances prises en compte seront présentées, avec les modifications qu'elles induisent sur le service d'appartenance à la coopération.

3.1. Principe suivi

Jusqu'à présent, nous avons supposé que l'entrée ou la sortie en coopération se faisait sans anomalie, de façon négociée. Les cas d'erreur ajoutés correspondent à une déconnexion brutale d'un agent coopérant ou d'un agent ayant demandé à coopérer. Cette déconnexion peut venir soit d'un problème d'un agent qui quitte brutalement l'accès au service de coopération, soit d'un problème des couches inférieures de communication qui n'arrivent plus à transmettre les messages d'un agent.

L'idée principale lorsqu'une erreur survient suite à la déconnexion brutale de la coopération d'un ou de plusieurs agents, est d'arrêter toute coopération qui ne soit pas dans un état valide le plus tôt possible, puis de rechercher une configuration valide parmi les agents restants en coopération et de relancer la coopération à partir de cette configuration valide choisie en urgence. Le service de coopération doit donc exclure des agents coopérants jusqu'à obtenir une configuration valide, dans laquelle le travail coopératif pourra être poursuivi. Si aucune configuration valide n'est possible, la coopération est terminée et tous les agents sont exclus. Le but du service est d'assurer un fonctionnement dégradé de la coopération, et d'obtenir le plus rapidement possible une configuration valide, ces configurations étant les seules pouvant assurer la continuité du travail coopératif. Si le service ne peut fournir de configuration valide, alors la coopération est impossible : tous les coopérants sont alors exclus. La solution choisie laisse l'entière initiative de la reprise sur erreur au service d'appartenance dynamique à la coopération.

Un cas plus général serait de lier le traitement d'erreur à l'application. Le service de coopération signalerait l'anomalie survenue à l'application et cette dernière, en fonction de sa sémantique, proposerait ou choisirait une solution. Le service de coopération peut toutefois fournir une plus grande aide en proposant au niveau application l'ensemble des solutions possibles. Dans ce cas, il ne se contente pas uniquement de signaler l'anomalie, mais il donne à l'application l'ensemble des configurations valides pour une reprise. Cette dernière choisit parmi ce panel de solutions celle qui lui paraît correspondre le mieux, le niveau coopération se chargeant de réaliser le choix fait au niveau application. Par exemple, les agents restants pourraient décider, de façon coopérative, la configuration valide en urgence à retenir.

La possibilité de lier le traitement d'erreur à l'application est à étudier dans le futur. Dans le traitement des anomalies, ce choix n'a pas été retenu car la prise de décision au niveau application correspond davantage au déroulement normal du service de coopération, et non à un traitement exceptionnel qui lui est laissé sous la responsabilité du service.

Pour illustrer le traitement d'erreur proposé, nous supposons que le graphe de coopération est celui de la figure III.11.

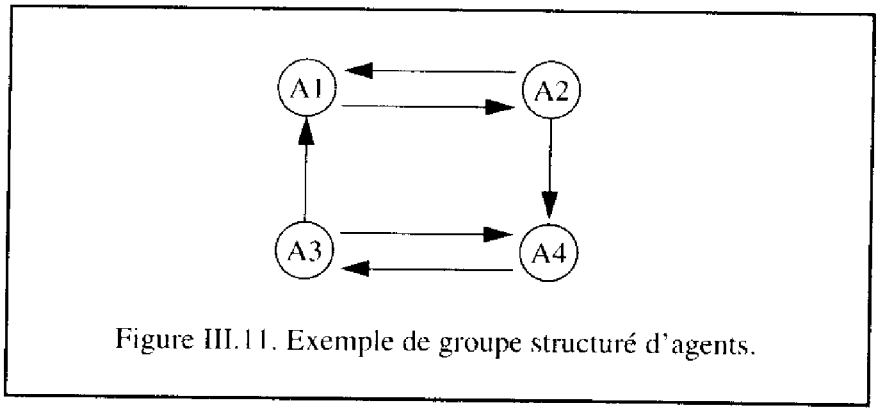
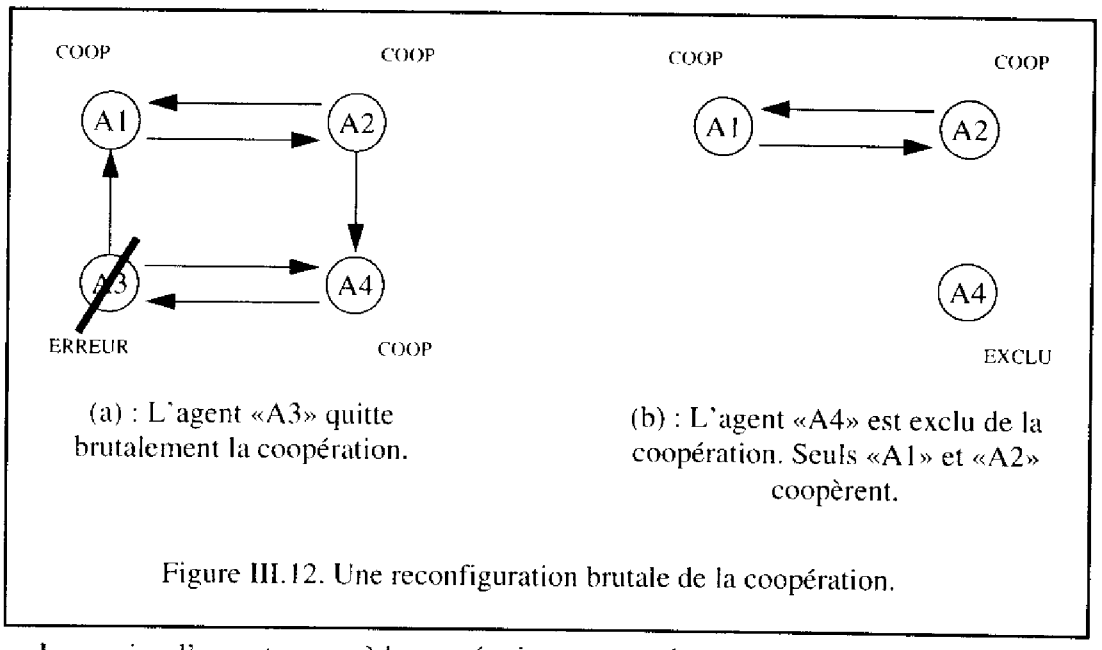


Figure III.11. Exemple de groupe structuré d'agents.

Les configurations valides sont les suivantes : {A1, A2}, {A1, A3}, {A2, A4}, {A3, A4}, et {A1, A2, A3, A4}.

A un instant, les quatre agents sont en coopération. La configuration valide instantanée comprend donc les quatre agents. Suite à une erreur, l'agent «A3» est brutalement déconnecté de la coopération. Seuls les agents «A1», «A2» et «A3» restent actifs (figure III.12(a)).



(a) : L'agent «A3» quitte brutalement la coopération.

(b) : L'agent «A4» est exclu de la coopération. Seuls «A1» et «A2» coopèrent.

Figure III.12. Une reconfiguration brutale de la coopération.

Le service d'appartenance à la coopération entame alors une procédure de reconfiguration en urgence. Il choisit de garder la configuration valide {A1, A2} pour poursuivre le travail coopératif. Pour cela, l'agent «A4» est exclu de la coopération. Le graphe instantané valide courant est alors celui de la figure III.12(b).

3.2. Fonctionnement détaillé

Les principales étapes du traitement en urgence lorsqu'un ou plusieurs agents sont brutalement déconnectés de la coopération valide courante sont décrites par la suite. A chaque étape, les primitives de service liées aux anomalies et utilisées par les agents sont présentées.

3.2.1. Reconfiguration en urgence

Déconnexions brutales

Suite à une anomalie au niveau utilisateur, un ou plusieurs agents se déconnectent brutalement du service de coopération, en utilisant la primitive de service :

USER_ABORT_REQ (domain: domain_type);

Les agents utilisant cette primitive peuvent se trouver soit bloqués en attente d'entrée en coopération, soit en coopération. Dans le premier cas, leur requête d'entrée est annulée par le service, et, si elle n'a pas encore été prise en compte dans un nouveau changement de coopération, cette annulation ne perturbe pas le déroulement de la coopération. Dans le deuxième cas, des agents coopérants sortent brutalement de la coopération, sans demander l'autorisation et sans attendre de négociation pour reconfigurer la coopération. Il s'agit là d'une anomalie qui va avoir des conséquences sur les autres agents coopérants.

Un deuxième type d'erreur survient dans le cas où les couches de communication inférieures qui permettent les échanges d'information entre les entités fournissant le service d'appartenance à la coopération présentent des anomalies. Il peut arriver qu'un agent ne puisse plus communiquer à cause d'un problème du médium de communication. Dans ce cas, l'utilisateur déconnecté du service d'appartenance dynamique à la coopération suite à un problème de transmission reçoit la primitive de service suivante :

SERVICE_ABORT_IND (domain: domain_type; ERREUR_COMMUNICATION);

Comme précédemment, cette primitive peut déconnecter soit un agent en attente d'entrée en coopération, soit un agent coopérant. Dans le premier cas, cette déconnexion annule la requête d'entrée de l'agent, mais ne perturbe pas le déroulement de la coopération si cette requête n'a pas encore été prise en compte pour un changement de coopération. Dans le deuxième cas, les agents coopérants brutalement déconnectés vont influer sur la structure instantanée valide en cours du groupe coopératif.

Cohérence en urgence

Le service donne alors l'ordre à l'ensemble des agents restant en coopération d'interrompre tous les échanges de données et d'obtenir en urgence un état cohérent. Cette phase est initialisée et terminée au moyen des primitives de service de mise en cohérence définies pour le service sans traitement d'erreur, c'est-à-dire :

NO_NEW_DATA_IND (domain: domain_type);

DT_COHERENTES_IND (domain: domain_type);

Reconfiguration

Lorsque tous les échanges de données sont suspendus, le service exclut certains agents de façon à obtenir de nouveau une configuration valide. Les agents exclus reçoivent la primitive :

SERVICE_ABORT_IND (domain, ERREUR_COOPERATION);

Les agents restant en coopération continuent le travail coopératif, et sont informés de la nouvelle configuration avec la primitive :

ERR_RECONF_IND (NV_GROUPE);

3.2.2. Cohérence d'un état

Comme pour le service d'appartenance sans erreur, la mise en cohérence des données lors d'une anomalie a été traitée de façon sommaire. Elle ne se traduit que par un arrêt des échanges de données pour que les agents aient tous connaissance des mêmes valeurs des données. Dans un cas réel, cette approche est trop simpliste. Une étude menée en parallèle [HELA94], [RAYN94] cherche à caractériser la notion de cohérence et de capture d'état cohérent, qui peut être comparée à la validation («commitment») des bases de données. En plus de cette étude, des algorithmes distribués de recherche d'état cohérent sont proposés [HELA94]. Le système enregistre des suites d'états cohérents, par exemple lors de tout changement de configuration valide. Lors d'une erreur, la mise en cohérence peut impliquer un retour en arrière, de façon à effectuer une reprise sur le dernier état cohérent atteint. Cette notion s'apparente à celle de l'annulation («rollback») des bases de données.

3.2.3. Anomalies lors de votes

Le changement de coopération négocié se fait au moyen d'un vote. Cependant, lorsque l'on ajoute la possibilité d'avoir des erreurs, les agents qui ont demandé à participer à la coopération, qui sont bloqués en attente d'entrée et qui font partie du graphe valide potentiel peuvent aussi être brutalement déconnectés. Ces agents ont été déconnectés avant d'entrer en coopération. Dans ce cas, la nouvelle configuration potentielle risque de ne plus être valide car l'ensemble des membres qui en font partie n'est plus connecté au service de coopération. La solution retenue est que le service annule le vote, car les agents coopérants sont en train de voter pour une configuration différente de celle qu'il est possible de réaliser. Le vote n'a alors plus de sens. Pour annuler le vote, le service envoie aux agents coopérants la primitive :

ERR_NO_VOTE_IND (domain: domain_type);

3.3. Primitives de service

Les primitives supplémentaires ajoutées pour le traitement et la prise en compte des erreurs ont été regroupées par unités fonctionnelles sur la figure III.13. La première unité fonctionnelle donne les deux primitives traduisant le fait qu'un agent est déconnecté du service de coopération. La primitive ERR_RECONF_IND informe les agents restant en coopération après une reconfiguration de la nouvelle configuration. La primitive ERR_NO_VOTE_IND est transmise aux agents coopérants lorsqu'un vote est annulé.

Figure III.13. Primitives de service supplémentaires pour le traitement d'erreur.

Nom et paramètres des primitives.	↘ Utilisateur vers Service. ↗ Service vers Utilisateur.	Rôle de la primitive.
USER_ABORT_REQ (domain: domain_type)	↘	Un agent participant à la coopération ou souhaitant y participer se déconnecte brutalement.
SERVICE_ABORT_IND (domain: domain_type; status: boolean)	↗	Déconnexion brutale d'un agent par le service de coopération. «status» indique l'erreur survenue : cela peut être une erreur du service de communication sous-jacent. Dans ce cas, «status» = ERREUR_COMMUNICATION. S'il s'agit d'une erreur du service de coopération, «status» = ERREUR_COOPERATION.
ERR_RECONF_IND (domain: domain_type; new_sites: list_site_type)	↗	Les agents restant en coopération sont informés d'une reconfiguration brutale de la coopération. «new_sites» donne la liste des sites restant en coopération.
ERR_NO_VOTE_IND (domain: domain_type)	↗	Annulation d'un vote lorsque des sites de la nouvelle configuration se déconnectent brutalement.

3.4. Spécification du protocole modifié

Le protocole modifié mis en oeuvre pour assurer le service a été décrit au moyen du langage de spécification Estelle. L'accent est mis sur les mécanismes spécifiques utilisés pour le traitement des erreurs.

3.4.1. Description de la spécification

La spécification Estelle du service et du protocole d'appartenance dynamique a été modifiée pour prendre en compte les erreurs pouvant survenir. La nouvelle architecture de cette spécification est présentée sur la figure III.14. La différence essentielle au niveau de l'architecture est l'ajout d'un canal supplémentaire entre le médium de communication et le gestionnaire de coopération. Ce canal sert à recevoir les indications d'erreurs survenues au niveau des agents coopérants connectés au service.

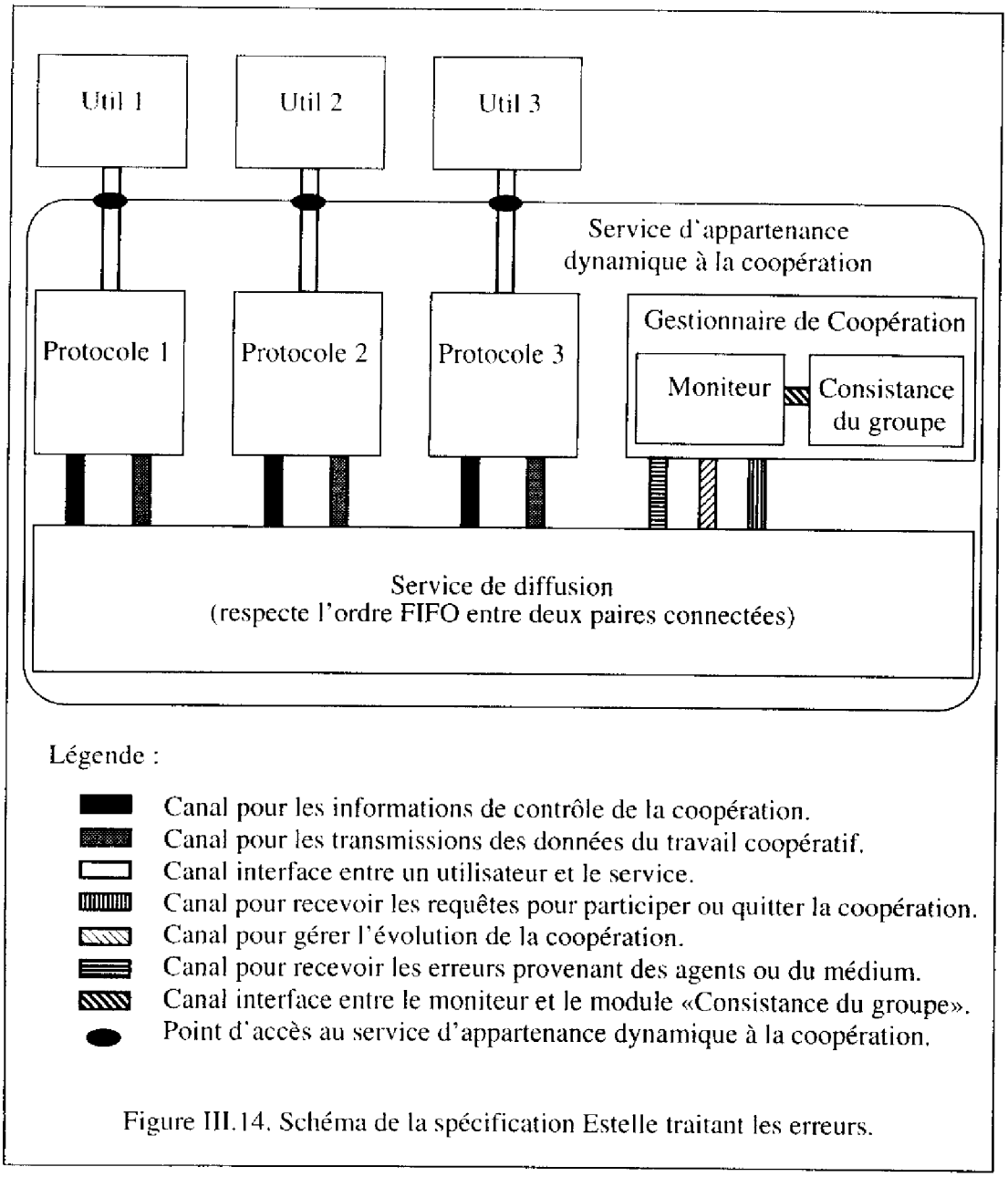


Figure III.14. Schéma de la spécification Estelle traitant les erreurs.

Le comportement des utilisateurs a été étendu pour qu'ils puissent utiliser et prendre en compte les primitives de service supplémentaires de la figure III.13 associées au traitement d'erreur.

Le code des modules protocole a aussi été modifié pour gérer les déconnexions brutales venant soit de leurs utilisateurs respectifs, soit du médium, soit du gestionnaire de coopération lors de l'exclusion d'un site. Lorsqu'un utilisateur se déconnecte brutalement de la coopération, le module protocole associé envoie un PDU d'erreur vers le canal de traitement d'erreur du gestionnaire de coopération. Ces modules assurent aussi l'obtention d'un état cohérent en urgence lors d'un problème dans la coopération et supportent les reconfigurations sur erreur. De plus, ils informent les utilisateurs lorsqu'un vote est annulé.

Le module «Gestionnaire de Coopération» gère l'évolution de la coopération dans le temps. Son rôle est étendu pour qu'il puisse également gérer la structure coopérative lorsque des déconnexions anormales surviennent au niveau des agents en coopération. Par un canal spécialement dédié à recevoir toutes les erreurs survenues (déconnexion d'un agent à la suite d'un problème interne ou d'un problème réseau), le gestionnaire de coopération traite les anomalies causées. Il forme une sous-coopération valide lorsque des agents coopérants sont brutalement déconnectés, ou interrompt le processus de vote si des agents pouvant participer au travail coopératif sont déconnectés.

Le comportement du module gestionnaire devient rapidement très complexe, lors du traitement des erreurs. Ce comportement peut être découpé en quatre phases qui correspondent aux phases d'évolution de la coopération de la figure III.7. Lors de la phase d'échange des contextes initiaux, les agents transmettent leur propres contextes suivant la structure de la nouvelle coopération. Supposons qu'une erreur survienne chez un agent coopérant. Le gestionnaire reçoit l'indication de cette erreur. Certains agents sont bloqués car ils ne reçoivent pas le contexte de l'agent en erreur. Le gestionnaire forme alors une sous-coopération valide à partir des agents restants. Il informe les modules protocole restant en coopération et ceux qui quittent la coopération. Les agents restants finissent leur phase d'échange de contexte. Pendant la phase de réalisation du travail coopératif, les agents échangent des données entre eux. Supposons qu'un agent coopérant tombe en panne. Le gestionnaire reçoit l'indication de l'erreur survenue. Il donne alors l'ordre aux modules protocole des agents coopérants non en panne, de mettre en urgence leurs données dans un état cohérent, en leur signalant l'agent en panne pour qu'ils ne s'attendent pas à recevoir d'information de la part de cet agent. Une fois cette mise en cohérence urgente effectuée, le gestionnaire forme une sous-coopération valide. Il prévient les modules protocole restant en coopération et ceux qui quittent la coopération. Pendant les phases de mise en cohérence, de vote et de restructuration de la coopération, des coopérants peuvent aussi tomber en panne. Le gestionnaire entame alors une mise en cohérence d'urgence similaire à celle de la phase de réalisation du travail coopératif. De plus, pendant un vote, si des agents faisant partie du nouveau graphe valide possible sont déconnectés, le vote est annulé. Le gestionnaire informe les modules protocole des agents coopérants de cette annulation.

Le service de diffusion possède également quelques caractéristiques supplémentaires. Les PDUs qui servent à acheminer une déconnexion brutale d'un utilisateur ne doivent pas être perdus par le réseau. Ces PDUs de contrôle de la coopération doivent être transportés de façon fiable. Ce médium a aussi comme particularité que lorsqu'une erreur due au médium survient (un agent est déconnecté par le médium suite à un problème de communication), le gestionnaire de coopération reçoit dans sa file de traitement des erreurs l'indication du problème survenu. Le service informe ainsi l'entité déconnectée et le gestionnaire de coopération. Ce service pourrait également signaler une déconnexion sur problème à l'ensemble des entités connectées, mais, dans la configuration proposée, les entités de protocole n'ont pas besoin de cette information.

Remarque :

Le traitement des erreurs effectué implique que le gestionnaire de coopération ne tombe jamais en panne ou n'est jamais arbitrairement déconnecté par le médium de communication. Cette hypothèse a été retenue car le gestionnaire de coopération sert à contrôler toute l'évolution de la coopération. S'il tombe en panne, aucun des sites ne peut entrer ou sortir de la coopération. On peut alors supposer que, si le gestionnaire tombe en panne, l'ensemble de la coopération est arrêtée et tous les coopérants sont exclus. Ces coopérants peuvent soit être informés par le service de communication au moyen d'une primitive spéciale, soit quitter par eux-mêmes la coopération.

3.4.2. Problèmes rencontrés

Le premier problème rencontré dans la spécification Estelle, est que suite à des erreurs, telles que les déconnexions des utilisateurs, du service de communication ou du service de coopération, des messages anciens peuvent traîner dans le système. Ces messages doivent être éliminés pour ne pas introduire un comportement aberrant du système. La technique retenue est que toute entité se connectant au médium de communication utilise une estampille ou numéro unique. Ce principe est analogue à celui du numéro de port unique et du «gel de port» pour éviter de recevoir de «vieux» messages. Une entité qui envoie un message le marque avec son propre numéro de port courant et indique tous les numéros de port des entités destinataires qu'il connaît (ces numéros de port ont été fournis par le moniteur lors de l'entrée en coopération). Toute entité recevant un message vérifie ainsi si le numéro de port correspondant à son identité est bien le même que son numéro de port courant. Dans la négative, le message est rejeté. Cette technique similaire à celle du gel de port est un artifice pour éliminer les «vieux» messages. Si l'on dispose d'un service de communication qui attribue des numéros de port uniques à chaque connexion, comme sous UNIX avec Internet Protocol, les «vieux» messages sont donc éliminés par le service et il n'est pas nécessaire de vérifier les numéros au niveau des transitions. Les gardes de la spécification Estelle peuvent alors être fortement simplifiées.

Le deuxième problème est le fait que des messages provenant d'une phase, non traités dans cette phase car superflus, traînent dans le système. Ce cas peut se produire après que des erreurs sont survenues. Ces messages en retard viennent encombrer et perturber les phases suivantes. Un exemple classique de ce problème a lieu lors de la mise en cohérence des données. Les agents coopérants mettent leurs données propres dans un état cohérent. Ils envoient un message pour le signaler au gestionnaire. Pendant ce temps, une erreur survient, un agent est déconnecté. Le gestionnaire reconfigure le groupe coopératif, exclut des agents qui avaient justement émis les messages de données cohérentes. Les agents restants arrêtent les échanges de données en urgence, et signalent au gestionnaire les fins de transmission. Le gestionnaire reçoit ces messages et change de phase. Suite à d'autres requêtes, une deuxième mise en cohérence des données est effectuée. Mais les messages non traités de la première mise en cohérence arrivent alors et viennent perturber le déroulement de la deuxième phase. Les numéros de port de ces messages sont corrects car les agents les ont envoyés avant de se déconnecter. La solution retenue dans la spécification Estelle est d'ajouter des gardes supplémentaires pour vérifier si les messages reçus font bien partie d'agents en coopération ou non. Dans la négative, ces messages sont rejetés.

Remarque :

Ces deux problèmes montrent bien la difficulté de concevoir un protocole prenant en compte les traitements des erreurs. De plus, la taille du logiciel croît fortement par rapport au système sans traitement d'erreur. Ainsi la spécification de la figure III.10, qui nécessitait 1800 lignes de code Estelle sans le traitement d'erreur, requiert près de 4000 lignes de code avec le traitement d'erreur. De plus, la mise au point est beaucoup plus délicate du fait du mélange des mécanismes propres au protocole et de ceux du traitement des erreurs.

4. Service pour plusieurs domaines

Cette partie présente une possibilité d'utilisation des services et des protocoles précédents pour des coopérations composées de plusieurs domaines. Deux cas vont se présenter : soit les domaines sont des entités faiblement couplées entre elles, soit au contraire, les domaines sont fortement reliés entre eux. Ces deux possibilités vont être abordées par la suite.

4.1. Domaines indépendants

Les domaines de coopération sont une autre façon de structurer la coopération. Les domaines sont définis à priori de façon statique par la coopération en même temps que le graphe de coopération qui donne la structuration de cette dernière. Ils ne peuvent pas être créés ou détruits dynamiquement au cours de la coopération. Tout d'abord, et c'est ce que nous allons développer dans ce paragraphe, il est possible de considérer que chaque domaine est une entité autonome de la coopération, ayant sa propre logique de coopération, possédant ses propres règles internes, et étant indépendante des autres domaines. Cette vision des domaines, conforme au modèle proposé, considère la coopération comme un ensemble de domaines indépendants et faiblement couplés entre eux. Par conséquent, si l'on considère que l'appartenance aux domaines est dynamique, chacune des règles définissant son évolution est interne à un domaine, et non liée aux règles des autres domaines. Les participations ou abandons de la coopération à l'intérieur d'un domaine se font donc indépendamment de celles d'autres domaines.

Dans ce cas, un agent faisant partie de deux domaines indépendants peut directement utiliser le service précédent pour chacun des deux domaines. On considère alors que le service ne gère pas une coopération formée d'un seul domaine, mais qu'il contrôle un seul domaine d'une coopération. Par conséquent, deux instances indépendantes du service permettent de gérer l'appartenance dynamique à deux domaines de coopération indépendants. Pour chaque participation à un domaine, un agent instancie ainsi le service d'appartenance dynamique correspondant au domaine (figure III.15).

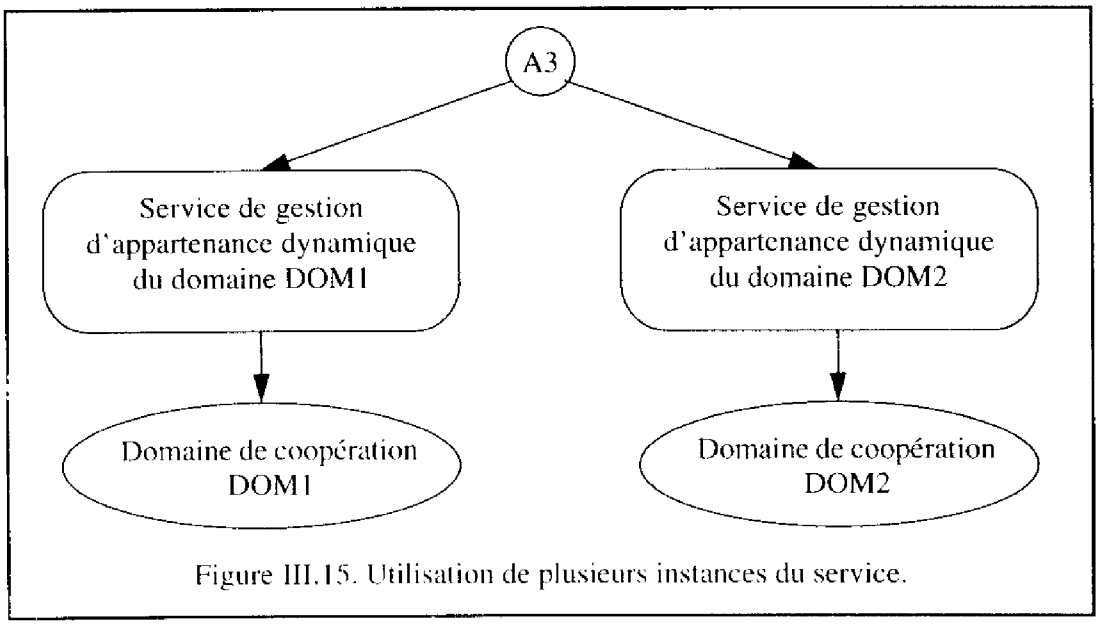


Figure III.15. Utilisation de plusieurs instances du service.

Le protocole précédent peut aussi être utilisé pour réaliser le service d'appartenance dynamique pour plusieurs domaines coopératifs. La figure III.16 donne l'architecture d'une spécification Estelle pour un groupe comprenant plusieurs domaines.

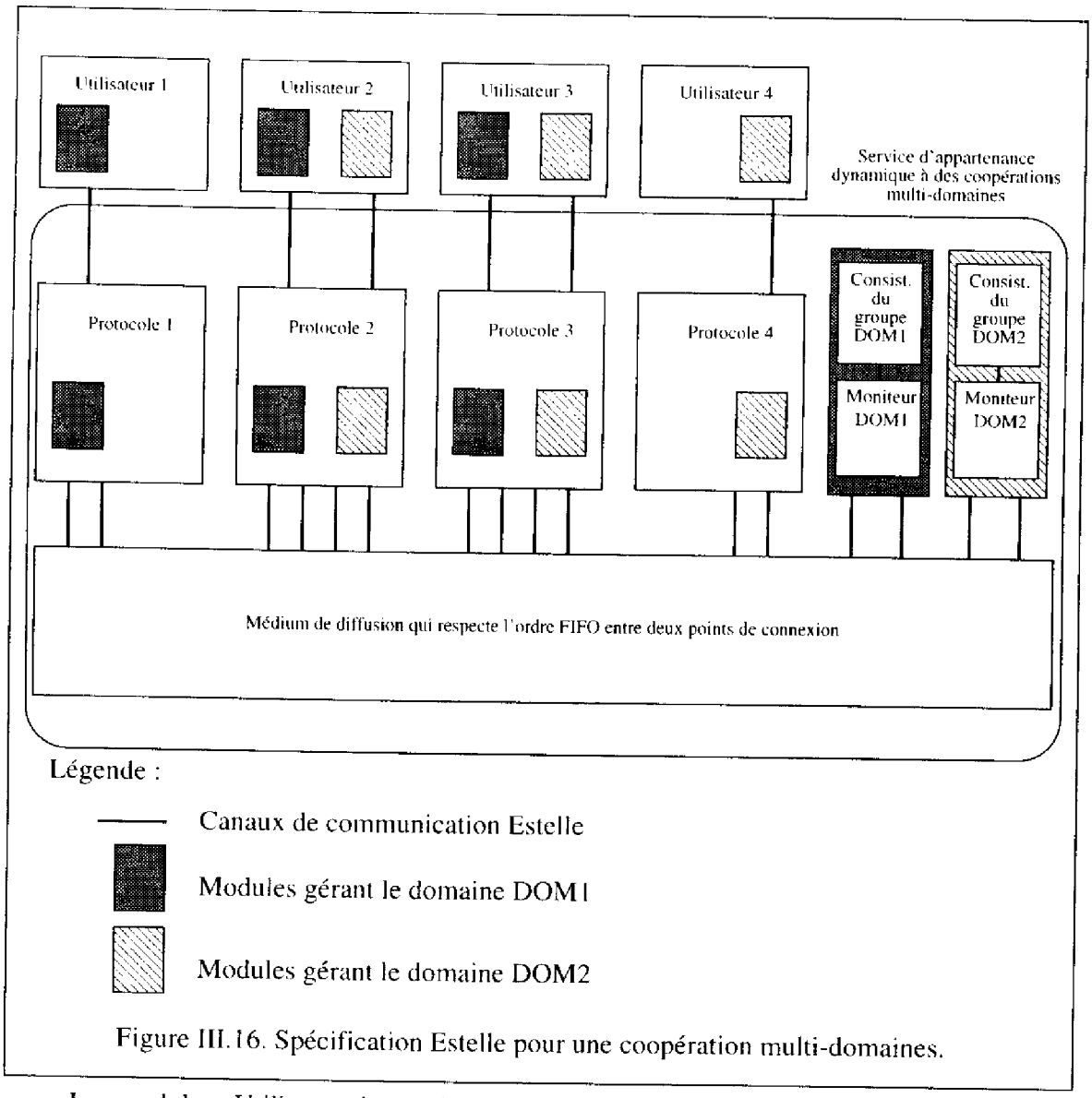


Figure III.16. Spécification Estelle pour une coopération multi-domaines.

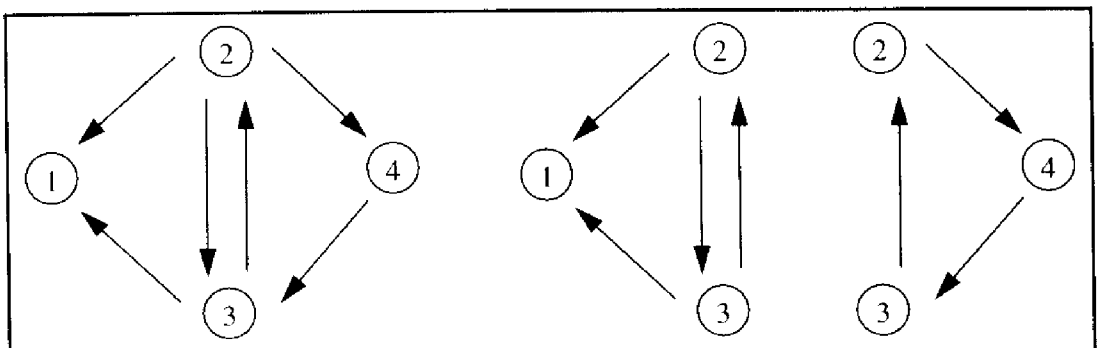
Les modules «Utilisateur i» représentent les utilisateurs du groupe coopératif. Les modules fils des utilisateurs contiennent les actions qu'un utilisateur peut effectuer dans un domaine auquel il appartient. Ainsi, l'utilisateur 1 n'a qu'un module fils. Il ne coopère que dans un seul domaine. Les utilisateurs 2 et 3 ont chacun deux modules fils car ils appartiennent tous deux aux domaines DOM1 et DOM2. Le rôle des modules «Utilisateur i» est d'instancier les modules fils qui correspondent à la participation aux domaines de coopération. Le comportement permis par chacun de ces modules fils est identique à celui donné par les modules utilisateurs du service d'appartenance dynamique de la figure III.9.

Les modules «Protocole i» permettent de réaliser le service qui a été défini. Ces modules possèdent également des modules fils qui s'occupent plus précisément de la réalisation du service pour chacun des domaines de la coopération. Le rôle des modules «Protocole i» est donc d'instancier les modules fils qui fournissent les services d'appartenance dynamique pour chacun des domaines de coopération. Le code de chacun des modules fils est le même que celui des modules protocole du service d'appartenance dynamique de la figure III.9.

Les modules gestionnaires de coopération contrôlent l'évolution dynamique de la coopération. Leur code est similaire à celui des modules gestionnaires de la figure III.9 : ils ne diffèrent que par le vote choisi. De la même façon, les sous-modules moniteur sont identiques à celui de la figure III.9. Seuls les modules «Consistance du groupe DOMi» sont différents, car chacun contient les critères de validité des graphes instantanés, qui sont liés à chacun des domaines de coopération et qui diffèrent suivant ces domaines.

La conception d'un protocole pour plusieurs domaines indépendants est donc une ré-utilisation directe du service d'appartenance dynamique pour un seul domaine. Le rôle des entités de protocole est donc limité à instancier les sous-modules provenant de la spécification précédente.

Par exemple, la configuration de la figure III.17 qui correspond à l'architecture Estelle de la figure III.16 a été étudiée. Le groupe coopératif est formé de quatre agents : 1, 2, 3 et 4. La structure de la coopération est donnée par la figure III.17(a). Le domaine DOM1 est formé des agents 1,2 et 3, organisés suivant la figure III.17(b), le domaine DOM2 contient les agents 2, 3 et 4 dont les relations sont données par la figure III.17(c). Pour qu'une coopération ait lieu dans le domaine DOM1, il faut que deux agents soient présents dans ce domaine. Il s'agit de la règle d'obtention des graphes de domaine instantanés valides pour DOM1. Le vote choisi est la majorité. Pour DOM2, les graphes valides retenus sont composés des agents : {2, 3}, {3, 4}, {2, 3, 4}. Le vote est l'unanimité.



(a) : Graphe de coopération. (b) : Graphe du domaine DOM1. (c) : Graphe du domaine DOM2.

Figure III.17. Configuration pour une coopération multi-domaines.

4.2. Domaines fortement couplés

L'autre hypothèse est de considérer que le travail effectué dans un domaine est fortement relié au travail d'un autre domaine. Dans ce cas, les domaines sont fortement couplés entre eux, et les règles logiques d'un domaine sont reliées et dépendantes de celles d'un autre domaine. Un agent faisant partie des deux domaines doit participer en même temps au travail des deux domaines fortement couplés et doit donc entrer en même temps dans ces derniers. Pour offrir cette possibilité, le choix des graphes instantanés valides se révèle plus complexe : en effet, les critères de sélection doivent permettre à un agent de participer à la fois aux deux domaines. Comme les deux domaines sont inter-dépendants, les critères de sélection des graphes valides sont aussi inter-dépendants. La prise de décision pour accepter une modification de la structure valide de coopération dans chacun des domaines pose également un autre problème : si un agent faisant partie des deux domaines se voit accepté dans un domaine et refusé dans l'autre, un choix doit être fait. Il est donc nécessaire de mettre en place une sorte de «méta-décision» qui permette de lever cette contradiction : soit l'agent est accepté dans les deux domaines qui voient leur configuration instantanée modifiée ensemble, soit son entrée est refusée et les configurations valides des deux domaines ne changent pas. Cette prise de décision peut être laissée au service de gestion des domaines, mais peut également être soumise à l'ensemble des agents des deux domaines fortement couplés.

Le service pour gérer l'appartenance dynamique d'un seul domaine peut cependant être réutilisé pour la gestion de domaines fortement couplés. Un agent faisant partie des deux domaines doit cependant utiliser une fois la requête d'entrée pour le premier domaine, et une fois pour le deuxième, tout en sachant que son entrée effective ne pourra se faire qu'après sa demande de participation dans les deux domaines. L'ajout d'une nouvelle primitive qui effectue à la fois une requête d'entrée ou de sortie dans l'ensemble des domaines fortement couplés offrirait une plus grande facilité d'utilisation du service. De même, on peut créer une primitive pour demander à sortir en même temps de plusieurs domaines. Par contre, dans le cas où une contradiction d'entrée ou de sortie est laissée à la charge des participants, deux primitives doivent être ajoutées, la première pour soumettre la contradiction aux agents coopérants, la deuxième pour recueillir leur réponse.

Au niveau de la conception du protocole, plusieurs possibilités dérivées de la conception originelle du service d'appartenance dynamique à la coopération peuvent être envisagées. Soit l'on garde un gestionnaire par domaine et ces gestionnaires se coordonnent et se synchronisent soit directement s'ils sont sur une même machine, soit par l'intermédiaire d'un service de communication s'ils sont distribués. Soit l'on crée un super-gestionnaire qui coordonne l'ensemble des domaines fortement couplés entre eux.

La gestion de l'évolution dynamique de domaines fortement couplés se révèle beaucoup plus complexe que celle de domaines indépendants. Elle a été moins approfondie dans notre travail et elle n'a pas été étudiée plus en détail car elle correspond moins au modèle de coopération qui utilise les domaines pour former des tâches indépendantes. De plus, pour éviter cette gestion ardue, il est possible de considérer un ensemble de domaines fortement couplés comme un seul et même domaine contenant un ensemble de règles communes. Dans ce cas, le service de gestion dynamique de la coopération peut être utilisé avec l'ensemble de ces règles communes.

Conclusion

Le modèle OSI se révèle finalement très bien adapté à la conception de services et de protocoles généraux coopératifs. En effet, son cadre d'application initial était la spécification de services et de protocoles de communication point à point, mais son extension vers le multipoint et vers les communications de groupe se fait sans trop de difficulté.

Par contrecoup, Estelle s'applique bien à la spécification formelle de ces services et de ces protocoles coopératifs qui supportent des groupes d'utilisateurs. La modularité d'Estelle permet la description d'un module type, qui donne le comportement général d'une entité, chacun des membres du groupe étant une instance de cette entité générique.

Estelle possède également de nombreux avantages. Le formalisme sur lequel il est basé, c'est-à-dire les machines à états, est simple à apprendre. Les extensions fournies par le langage PASCAL complètent bien le formalisme initial, surtout pour la description des types de données, des fonctions de test ou de calcul des messages échangés. PASCAL est également un langage algorithmique classique, relativement répandu. Le fait de pouvoir récupérer des parties des spécifications Estelle à des fins d'implantation offre un autre avantage non négligeable.

Cependant, le formalisme des machines à états possède un gros inconvénient : son faible niveau sémantique, ce qui nécessite un grand nombre d'états pour modéliser un comportement. Cela se traduit le plus souvent par une explosion combinatoire du nombre d'états de l'ensemble du système. A titre d'exemple, une spécification globale pour l'entrée et la sortie dynamique de la coopération de trois agents possède plus de six cent mille états résultant de la combinaison des états de chacune des entités du système. Il est donc très difficile, voire impossible d'établir des vérifications exhaustives de spécifications Estelle représentant des cas réels. L'approche retenue alors est la simulation avec un grand nombre de pas. Les traces de simulation sont mémorisées, et un ensemble d'observateurs sont définis. Ces observateurs sont des entités qui contiennent un certain nombre de propriétés. A chaque étape, l'observateur contrôle la validité des propriétés et arrête la simulation en cas de viol de ces propriétés. Cependant, cette approche de simulation ne donne qu'un «certain degré de confiance» de la spécification et ne fournit pas de preuve exhaustive sur l'ensemble des états du système.

Le chapitre suivant va donc montrer l'approche utilisée pour la vérification du protocole d'appartenance, puis va présenter l'implantation réalisée à partir du code Estelle.

Chapitre IV

Vérification et implantation du protocole d'appartenance

Introduction

La spécification d'un système distribué permet de vérifier qu'il se comporte correctement vis-à-vis des propriétés attendues, et qu'il ne possède pas de situation anormale conduisant à des blocages ou à des comportements erronés.

Deux types d'approches sont utilisées pour la vérification de logiciels et d'algorithmes distribués. La première repose sur la vérification de propriétés générales, qui doivent être vraies pour tout système. La non-existence de blocages, de réception non spécifiée... constituent des propriétés de ce type. La deuxième approche étudie les propriétés spécifiques du système à analyser. Elle fait souvent appel à l'utilisation d'invariants [RAYN85], technique très répandue dans le domaine des systèmes répartis. Les propriétés que le système doit respecter et vérifier sont décrites sous la forme de formules logiques qui représentent les invariants. Le principe est de démontrer que l'évolution du système dans le temps ne viole pas ces invariants, c'est-à-dire que toutes ses transformations respectent et conservent la validité des invariants. Les démonstrations sont effectuées comme des preuves mathématiques : chaque démonstration est spécifique d'un algorithme distribué, et est réalisée manuellement.

Pour coupler ces deux approches, il est nécessaire de représenter l'ensemble des comportements du système. Souvent, cet ensemble est obtenu à partir d'une description formelle sous la forme d'un système de transition étiqueté. Chaque état instantané du système est représenté par un sommet du graphe, une transition entre deux états modélisant l'action nécessaire pour le passage d'un état vers un autre. Il est alors possible, lors de la construction du graphe, d'observer si les propriétés générales ou les invariants sont satisfaits : l'analyse peut s'effectuer directement à partir du graphe. Cependant, la représentation de systèmes réels comprend souvent plusieurs dizaines de milliers d'états et de transitions. Une autre technique d'analyse est alors l'utilisation de bissimulations ou projections. Le graphe étiqueté est partitionné en événements observables et inobservables, puis est réduit au moyen d'une relation d'équivalence. L'automate quotient ne

présentant plus que quelques dizaines d'états en général est de ce fait beaucoup plus facilement analysable et interprétable.

Ces méthodes peuvent être automatisées : en effet, le système est d'abord décrit au moyen d'un langage basé sur des modèles de machines à états, ou de réseaux de Petri. L'ensemble des états et leurs relations sont alors obtenus automatiquement. Dans le cas des techniques de bissimulation, la partition du graphe et ses projections sont également supportées par des outils. Ce support de la vérification qui simplifie les fonctionnements et les rapproche des services à remplir, nous a fait opter pour son utilisation dans le cadre de la vérification du protocole d'appartenance dynamique à la coopération. De plus, l'établissement du graphe nous permet de vérifier, lors de sa construction, la véracité des propriétés générales.

Une fois la correction du système formel établie, son adaptation en vue d'une implantation peut être effectuée. Ceci a été fait à partir des spécifications Estelle du chapitre précédent. La démarche générale pour implanter une spécification Estelle est de conserver le comportement des modules définis. Par contre, les communications simulées par certains canaux sont remplacées par des appels de primitives de communication réelles. Cette étape nécessite de prendre en compte les caractéristiques des primitives de communication disponibles qui vont intervenir. La spécification des modules est également étendue par des fonctions qui interfacent le code Estelle et ces primitives.

Ce chapitre est donc composé des deux parties suivantes :

La section 1 détaille le processus de vérification utilisé pour le protocole d'appartenance dynamique, et analyse les projections obtenues. La section 2 décrit l'implantation de la spécification Estelle dans un environnement UNIX, en transformant les modules Estelle en processus UNIX pouvant être distribués sur plusieurs machines.

1. Vérification

Cette partie présente l'approche utilisée pour la vérification du protocole d'appartenance dynamique. L'architecture de la spécification Estelle initiale a été portée dans l'environnement multi-agents VAL pour être vérifiée.

1.1. Principe de vérification utilisé

La méthode qui a été retenue pour l'analyse des propriétés du protocole et pour sa vérification est celle de l'analyse du graphe d'accessibilité, en particulier au moyen de projections.

Le *graphe d'accessibilité* représente le graphe de tous les états du système global. Chaque sommet de ce graphe représente un état du système. Au cours d'une simulation d'un système, le passage d'un état dans un autre état se fait lors du tir d'une transition au préalable sensibilisée. Cette action se traduit par une flèche, souvent étiquetée par l'identificateur de la transition tirée, entre l'ancien état et le nouvel état, dans le graphe d'accessibilité. Finalement, on peut voir une simulation comme le parcours d'un chemin dans le graphe d'accessibilité.

Le graphe d'accessibilité d'un système réel peut être très volumineux, souvent de l'ordre de plusieurs dizaines ou centaines de milliers d'états et de transitions, et son analyse sera d'autant plus compliquée et difficile à mettre en oeuvre que sa taille est importante. La première technique pour limiter sa taille, qui découle de la sémantique du parallélisme par entrelacement, est de limiter le plus possible le nombre de transitions dans le système. En effet, si un système qui est composé de deux modules faiblement couplés entre eux pouvant effectuer chacun dix transitions, peut être transformé en un système équivalent dont chaque module ne possède plus que

cinq transitions plus puissantes, le graphe résultant se trouve beaucoup plus réduit. Ceci n'est possible que si chaque transition élémentaire possède des fonctionnalités plus puissantes, et de plus haut niveau sémantique que les transitions initiales.

Estelle est basé sur un formalisme de machines à états. Ce formalisme, bien que très facile d'utilisation, ne possède que peu de puissance d'expression. En effet, chaque module Estelle nécessite souvent un grand nombre d'états et de transitions intermédiaires, ce qui provoque une explosion combinatoire du nombre d'états du graphe d'accessibilité. De ce fait, la création du graphe d'accessibilité d'une spécification Estelle pure est le plus souvent impossible car le stockage du graphe dépasse la capacité des ordinateurs. Pour procéder à l'analyse d'une telle spécification, il est préférable de la modifier en y ajoutant des mécanismes plus puissants et plus compacts, tels que des communications par rendez-vous ou bien en la réécrivant avec un langage basé sur un formalisme de plus haut niveau sémantique tel que les réseaux de Petri.

Les propriétés à vérifier peuvent être des propriétés générales comme l'absence d'interblocage, ou l'absence de cycle infini ne faisant pas progresser le système. Dans le cas d'un protocole de communication, les propriétés plus spécifiques recherchées doivent également montrer que le protocole fournit bien le service attendu et qu'il ne conduit pas à des situations aberrantes, inattendues ou contradictoires avec la définition du service proposé.

L'analyse de propriétés peut se faire directement sur le graphe d'accessibilité, si celui-ci n'est pas trop gros. Cependant, même avec des langages de description de haut niveau sémantique, il n'est pas rare qu'un graphe d'accessibilité fasse quelques centaines de milliers d'états. Pour réduire sa taille et pour pouvoir l'interpréter et l'analyser plus facilement, on utilise alors des techniques de bissimulations, ou projections. Pour vérifier une propriété, on sélectionne les transitions qui correspondent à cette propriété, et on banalise les autres. Cette sélection partitionne le graphe d'accessibilité en un ensemble d'événements observables et en un ensemble d'événements internes. Par la suite, le graphe partitionné est réduit en un automate quotient au moyen d'une relation d'équivalence. Tous les événements observables font partie de l'automate quotient. Par contre, suivant la projection choisie, les événements internes sont réduits et seuls ne demeurent que leurs effets sur l'ensemble des événements observables. L'automate quotient obtenu ne comporte souvent que quelques états : son analyse est alors simplifiée. Grâce à la relation d'équivalence, toute propriété vérifiée sur l'automate réduit est également vérifiée sur le graphe d'accessibilité initial.

L'analyse du protocole d'appartenance dynamique s'est fortement appuyée sur les projections des graphes d'accessibilité obtenus.

1.2. Présentation de VAL

VAL est un environnement de programmation multi-agents qui permet la mise au point, la simulation, puis la vérification d'un système composé d'instances d'agents qui communiquent entre eux. Une spécification VAL [VERN93], [VILL94], [DIAZ94d], [VILL95] est composée d'un ensemble d'agents communicants dont les comportements sont basés sur le formalisme des réseaux de Petri prédicats/transitions. Les places et les transitions du réseau de Petri modélisant le comportement de chacun des agents sont représentées en utilisant des prédicats PROLOG (figure IV.1). De même, les fonctions de test ou de manipulation des messages reçus ou transmis sont écrites en PROLOG.

Chaque agent est une instance d'une classe d'agents. Cette classe donne le comportement type et les fonctionnalités type de chacune des instances. Les agents peuvent créer dynamiquement d'autres agents en indiquant la classe du nouvel agent créé.

Les agents VAL ne partagent aucune mémoire commune et communiquent entre eux au moyen de messages. Un agent envoie un message vers un autre agent destinataire en utilisant l'identification du destinataire et en indiquant chez ce destinataire un identificateur de port. Si le port n'existe pas chez l'agent destinataire, il est dynamiquement créé et le message est placé en tête de la file d'attente associée à ce port. Si ce port existe et contient des messages, le nouveau message est placé à la fin de la file d'attente liée au port. Un port disparaît lorsqu'il est vide.

Des primitives de communication très puissantes et variées sont utilisables par les agents VAL. Un agent VAL peut effectuer un envoi asynchrone vers un ou plusieurs destinataires de messages identiques ou différents au moyen d'une seule primitive : il s'agit d'une émission asynchrone multiple. De même, un agent peut recevoir plusieurs messages envoyés de façon asynchrone en attendant sur plusieurs ports à la fois. Dès que les messages attendus sont tous en tête des files d'attente des ports de communication souhaités, l'agent peut recevoir et traiter ces messages. Une seule primitive autorise ainsi une réception asynchrone multiple. Des communications synchrones sont aussi possibles par rendez-vous entre deux ou plusieurs agents.

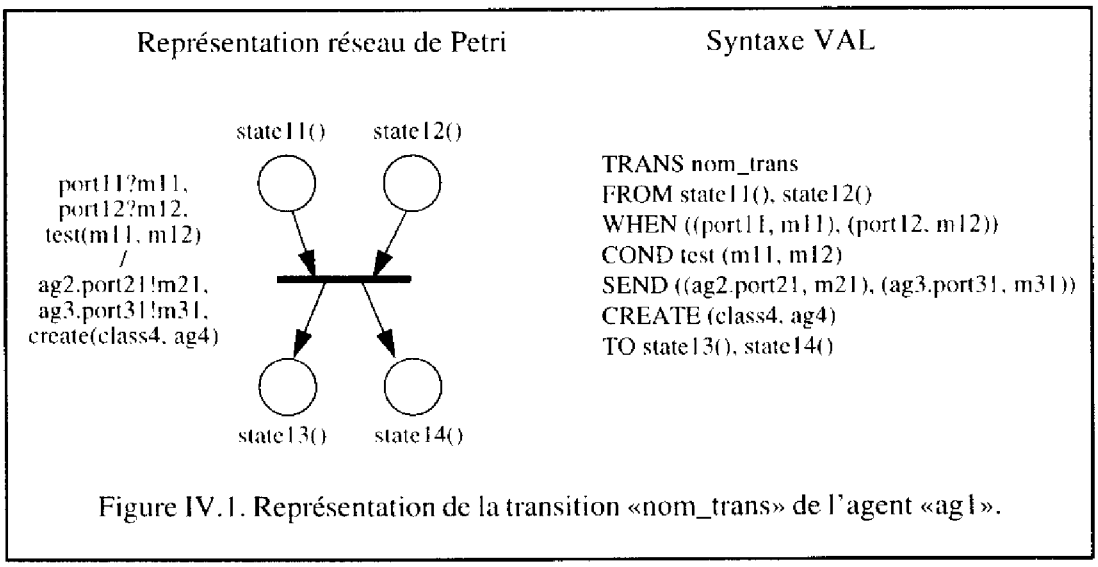


Figure IV.1. Représentation de la transition «nom_trans» de l'agent «ag1».

Pour illustrer les possibilités de VAL, l'exemple de la figure IV.2 a été choisi. A un instant donné une spécification VAL est composée de trois agents «ag1», «ag2» et «ag3». L'agent «ag1» se trouve dans l'état «state11(), state12()». Cet agent reçoit deux messages «m11» et «m12» respectivement sur les ports «port11» et «port12» (figure IV.2(a)). La transition «nom_trans» de la figure IV.1 fait partie du comportement de l'agent «ag1».

La transition «nom_trans» de l'agent «ag1» se trouve sensibilisée lors de l'arrivée des deux messages «m11» et «m12» en tête des files des deux ports si ces messages vérifient le prédicat «test» (figure IV.2(a)). Supposons que cette transition soit tirée. Dans ce cas, l'agent «ag1» envoie deux messages «m21» et «m31» vers les agents «ag2» et «ag3». Ces derniers sont stockés respectivement dans les files d'attente associées aux ports «port21» et «port31» des agents «ag2» et «ag3». Puis l'agent «ag4» dont le comportement est donné par la classe «classe4» est créé. Finalement, l'agent «ag1» passe dans l'état «state13(), state14()» et l'état global du système est celui donné par la figure IV.2(b).

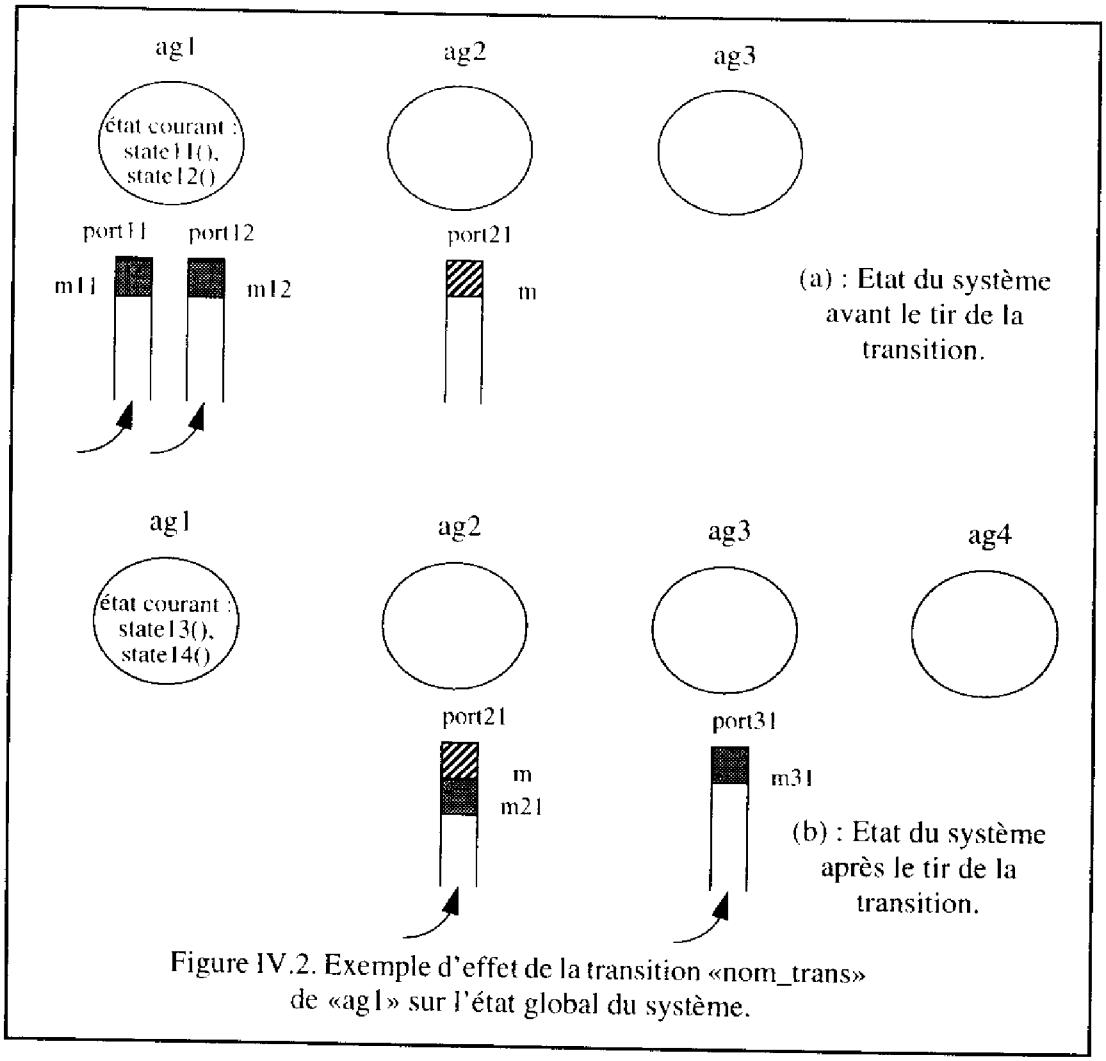


Figure IV.2. Exemple d'effet de la transition «nom_trans» de «ag1» sur l'état global du système.

1.3. Adaptation de la spécification Estelle en Estelle* et en VAL

La spécification Estelle initiale a été remaniée et a conduit à deux spécifications Estelle*, afin de hausser le niveau sémantique de la spécification et de diminuer le nombre d'états global du système pour faciliter sa vérification. Pour cela, l'environnement VEDA [VEDA91] a été utilisé. Estelle* est une extension d'Estelle qui permet, en plus des communications asynchrones, des échanges synchrones par rendez-vous en déclarant des canaux spéciaux avec le mot réservé «rendez_vous».

1.3.1. Première adaptation en Estelle*

La spécification Estelle initiale suivant le modèle OSI a été transformée en une spécification Estelle*, en introduisant des rendez-vous entre les utilisateurs et le service d'appartenance dynamique afin de rendre plus abstraits les mécanismes d'interface utilisateur/service (figure IV.3). L'architecture générale de la spécification reste cependant identique. L'utilisation des rendez-vous simplifie fortement le comportement des utilisateurs qui deviennent fortement synchronisés avec le service qu'ils utilisent. De plus, ils réduisent le nombre d'états du système global, car ils fusionnent l'émission d'une primitive de service avec sa réception et sa prise en compte : on n'a plus d'état intermédiaire provenant de l'émission d'une primitive de service suivie de sa réception.

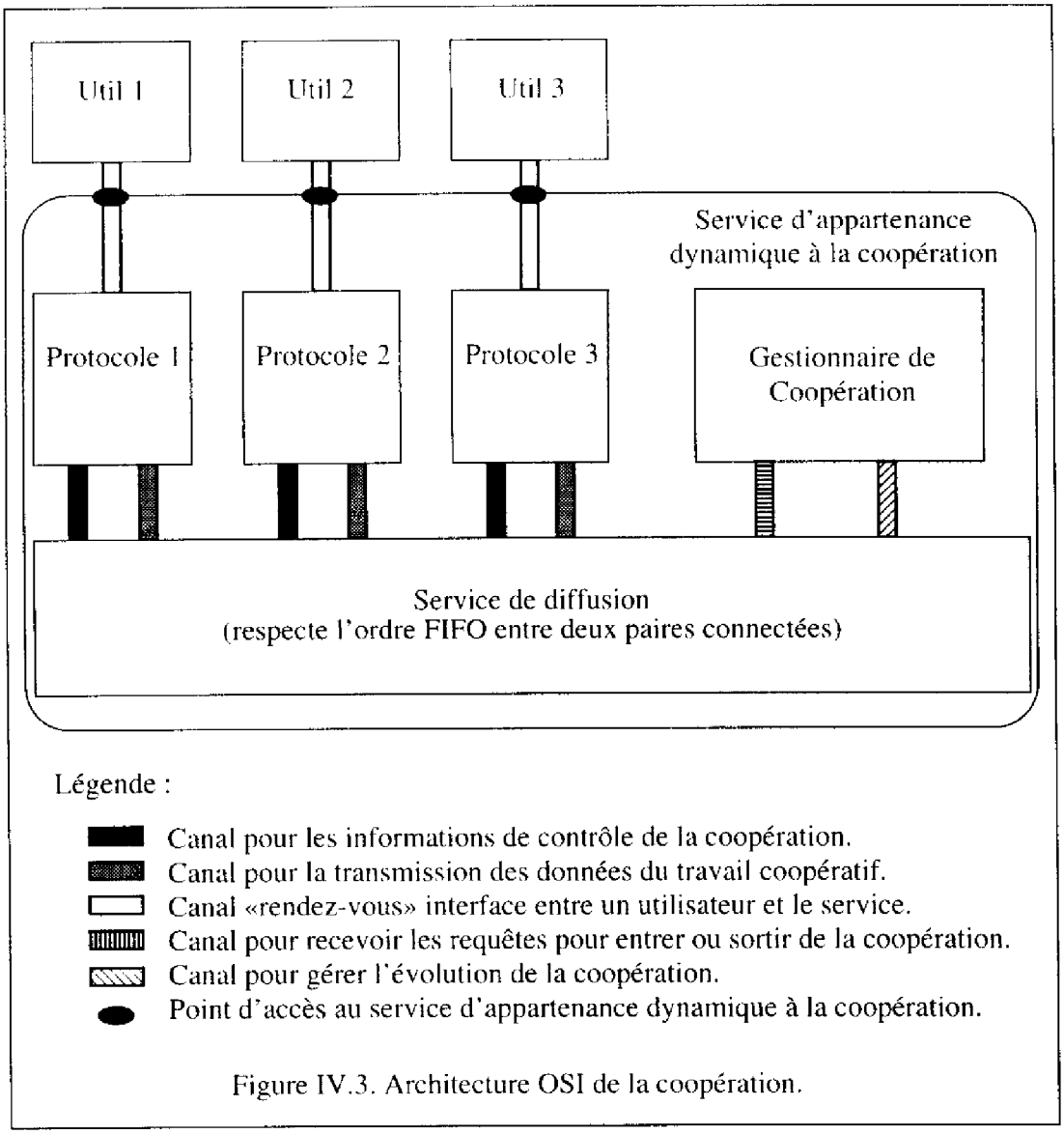


Figure IV.3. Architecture OSI de la coopération.

1.3.2. Deuxième spécification Estelle*

L'architecture de la spécification Estelle* suivant le modèle OSI introduit également des états intermédiaires au niveau du service de diffusion sous-jacent. En effet, un PDU échangé entre deux entités de protocole nécessite trois états : le premier état est l'émission du PDU vers le service de diffusion, le second est la réception du PDU par le service et son émission vers le destinataire, le dernier état est la réception du PDU par l'entité destinataire. Les inconvénients provenant de ces trois états sont levés en supprimant le module service de diffusion et en représentant ce service par un ensemble de files Estelle interconnectant directement les entités de protocole. Les propriétés requises par le service de diffusion, essentiellement de respecter l'ordre FIFO entre deux paires connectées sont garanties par les propriétés des files Estelle. Cette transformation permet de supprimer l'état intermédiaire. Le service défini par l'ensemble des files Estelle ne nécessite plus que deux états : un pour l'émission d'un PDU, l'autre pour sa réception. L'architecture de cette deuxième spécification est donnée par la figure IV.4.

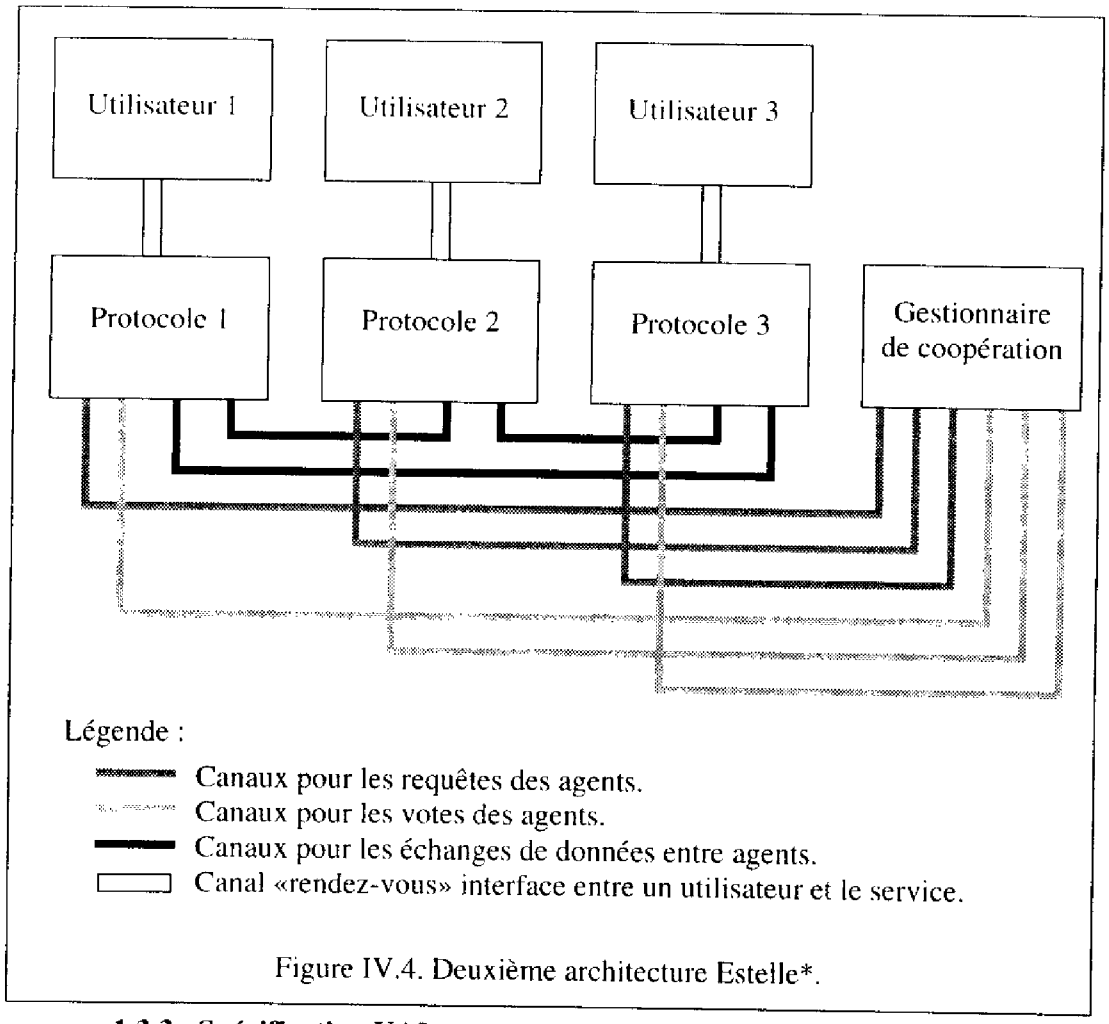


Figure IV.4. Deuxième architecture Estelle*.

1.3.3. Spécification VAL

En utilisant Estelle*, il n'est pas possible de diminuer davantage le nombre des états du système. Il s'est avéré que l'environnement VAL, plus riche et plus puissant que Estelle*, permet de rehausser encore le niveau sémantique de la spécification. VAL possède des primitives de diffusion et de réception multiples. L'idée est donc de transcrire la spécification Estelle* en une spécification VAL et d'utiliser ces primitives pour réaliser le service de diffusion nécessaire aux entités de protocole [VILL94], [DIAZ94d].

Chaque module protocole est représenté par une instance d'agent VAL. Pour chacun de ces modules Estelle, l'état courant de la machine à états est représenté par une clause PROLOG. Les données et variables internes aux modules deviennent également chacune une clause PROLOG. Les agents protocole possèdent deux catégories de ports pour envoyer, recevoir et échanger les PDUs entre eux. La première catégorie comprend le port de contrôle qui est utilisé pour recevoir tous les messages de contrôle en provenance du gestionnaire de coopération. La deuxième catégorie contient les ports de données qui sont utilisés pour recevoir les données en provenance des autres agents protocole. Chaque protocole possède N-1 ports de données pour recevoir potentiellement des données des autres N-1 coopérants. Cette architecture a été retenue pour éviter de sérialiser arbitrairement sur un même port les messages provenant des autres sites. Cette multiplication des ports de communication présente deux avantages pour la vérification. Le fait de ne pas sérialiser des messages indépendants dans une seule file d'attente évite de générer dans

le graphe d'accessibilité du système des états intermédiaires inutiles résultant des combinaisons des différents ordonnancements de ces messages. De plus, l'utilisation de ports séparés pour les données permet d'utiliser les réceptions multiples, qui contribuent également à diminuer le nombre d'états du graphe d'accessibilité de la spécification VAL. En effet, lorsqu'un agent attend plusieurs messages sur plusieurs ports séparés, un seul nouvel état sera nécessaire pour prendre en compte la réception de tous les messages attendus. Sans réception multiple, i.e. en recevant et en traitant les messages un par un, il aurait fallu autant d'états intermédiaires que de messages attendus, sans toutefois prendre en compte l'entrelacement possible de ces états intermédiaires avec les modifications possibles provenant d'autres agents.

Le module gestionnaire de la coopération devient aussi un agent VAL. Cet agent possède également deux catégories de ports de communication. La première catégorie contient N ports de communication, un par coopérant, et elle sert à recevoir les requêtes des agents protocole. La deuxième catégorie est formée de N ports de communication pour recevoir les votes des agents en coopération, lorsqu'un changement de coopération est possible.

L'ensemble des agents protocole et gestionnaire représente un système distribué dont les composants ne sont pas fortement synchronisés entre eux. Par conséquent, seules les communications asynchrones ont été utilisées entre les divers agents de la spécification. L'ensemble des primitives asynchrones fournies par l'environnement VAL suffisent pour représenter le service de diffusion utilisé par les entités de protocole et par le gestionnaire de coopération.

L'architecture de la spécification VAL est donnée par la figure IV.5. Cette spécification a été simplifiée par rapport aux spécifications Estelle en supprimant les utilisateurs. En effet, il est possible de représenter les utilisateurs par des agents VAL interconnectés par rendez-vous aux entités de protocole. Cependant, chaque transition atomique VAL peut contenir à la fois un rendez-vous, une réception et une émission asynchrone. Le fait d'ajouter ces utilisateurs ne fait qu'ajouter des rendez-vous dans le contenu de transitions déjà existantes des modules protocole. On n'a aucune création de nouvelle transition. Par conséquent, cet ajout ne modifie ni la taille globale du graphe d'accessibilité du système, ni même les relations entre ces états. La structure du graphe d'accessibilité restant la même, les utilisateurs sont donc optionnels pour la vérification.

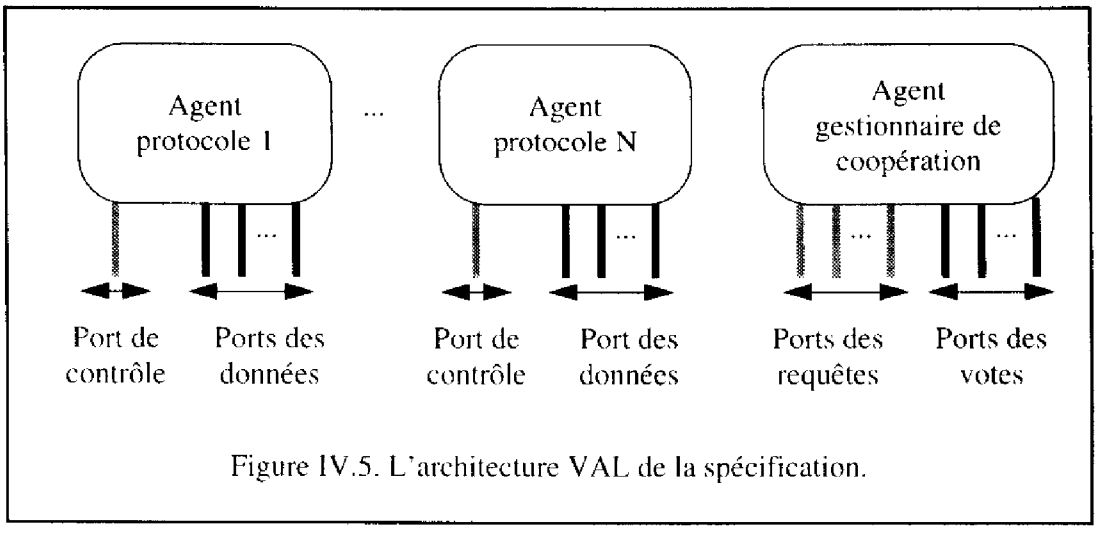


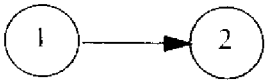
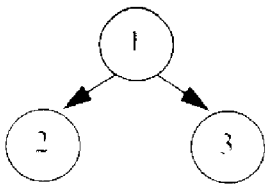
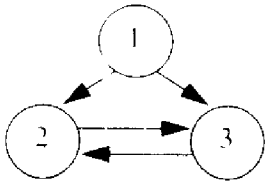
Figure IV.5. L'architecture VAL de la spécification.

1.3.4. Comparaison des tailles des graphes d'accessibilité

La figure IV.6 montre l'évolution de la taille des graphes d'accessibilité suivant le niveau sémantique des spécifications [VILL95]. En prenant plusieurs exemples de groupes coopératifs comprenant deux et trois coopérants, nous voyons que les spécifications VAL sont finalement les mieux adaptées à la vérification, la deuxième spécification Estelle* étant un peu moins bonne, et finalement la spécification Estelle* respectant le modèle OSI étant mal adaptée. Dans tous les exemples suivants, les graphes instantanés valides doivent comporter au moins deux sites. Le vote retenu est le vote majoritaire. Lorsque deux agents n'ont pas de flèche les reliant, cela signifie qu'ils ne peuvent pas échanger de donnée entre eux. Bien entendu, certains exemples sont uniquement théoriques et ne servent que pour voir l'évolution de la taille des graphes d'accessibilité. Les ratios obtenus sont de 5 et de 10 pour une coopération à deux sites entre la spécification VAL et les spécifications Estelle*. Par contre, ces ratios sont de 10 et de 40 pour les coopérations à trois sites.

La taille maximum des graphes d'accessibilité qui peuvent être gérés et manipulés dépend bien entendu de la configuration des machines qui génèrent ce graphe, ainsi que du volume de stockage disponible pour sauvegarder ces graphes. Cependant, au delà de cent mille états, les outils disponibles ne peuvent que très difficilement procéder à l'analyse de ces graphes qui ne peuvent pas non plus, du fait de leur taille trop importante, être stockés sur disque.

Figure IV.6. Taille des graphes d'accessibilité.

Graphe de coopération	Spécification VAL	Spécification Estelle* 2	Spécification Estelle* 1
	48 états 72 arcs	328 états 759 arcs	490 états 1163 arcs
	69 états 123 arcs	251 états 495 arcs	495 états 1070 arcs
	126 états 302 arcs	600 états 1455 arcs	1865 états 5166 arcs
	6282 états 15223 arcs	62009 états 182329 arcs	147967 états 462223 arcs
	10643 états 31077 arcs	103893 états 332875 arcs	plus de 688128 états plus de 2581316 arcs (VEDA n'a pas pu calculer le graphe)

1.4. Analyse des graphes d'accessibilité. Projections obtenues

L'analyse du protocole d'appartenance dynamique s'est poursuivie en utilisant des projections des graphes d'accessibilité des spécifications VAL [VILL94]. La sélection des transitions observables lors de la partition du graphe d'accessibilité est faite au moyen d'un filtre VAL. Ces filtres sont décrits par des clauses PROLOG. Les transitions du graphe d'accessibilité sont étiquetées par des prédicats PROLOG. Les étiquettes des transitions qui s'unifient avec les clauses du filtre VAL sont conservées telles quelles, et font partie des événements observables. Les autres étiquettes sont banalisées, et deviennent des événements inobservables. La syntaxe des clauses des filtres est la suivante :

show_trans (classe_agent, id_agent, prédicat_prolog, nouveau_prédicat_prolog).

Cette clause sélectionne les événements effectués par l'agent identifié par la classe «classe_agent» et «id_agent» dont l'identification s'unifie avec le terme «prédicat_prolog». La transition sélectionnée est renommée par le terme «nouveau_prédicat_prolog» qui est le plus souvent identique à «prédicat_prolog». Les transitions qui ne s'unifient pas avec au moins une des clauses du filtre sont renommées en «i» qui signifie inobservable. Il s'agit alors des événements inobservables ou internes.

Une fois le graphe partitionné obtenu, ce dernier est réduit en utilisant l'outil Aldebaran [FERN88], [FERN90] en utilisant les projections observationnelle et sûreté. Par la suite, les principales propriétés observées, ainsi que les graphes réduits obtenus sont présentés.

Certaines transitions des automates quotients provenant de projections observationnelles sont étiquetées par «i». Ces transitions correspondent aux effets des événements inobservables sur l'ensemble des événements observables de l'automate quotient résultant. Cela signifie qu'un événement interne «i» a eu une influence sur une action non masquée et observable, et que cet événement interne n'a pu ni être réduit ni supprimé par la projection observationnelle.

1.4.1. Propriété locales. Projections sur un agent

Cette section contient un ensemble de propriétés qui concernent l'étude du comportement d'un seul agent par rapport à l'évolution de la structure globale de la coopération. La façon dont évolue un agent par rapport à l'ensemble des configurations de la coopération est analysée.

Appartenance à la coopération

Le graphe quotient de la figure IV.7 est une projection sûreté du système. Cette projection montre le comportement d'un agent pour participer à la coopération et pour quitter cette dernière. Le graphe obtenu correspond exactement à la description du fonctionnement du service d'appartenance dynamique proposé aux utilisateurs, dans le chapitre III.

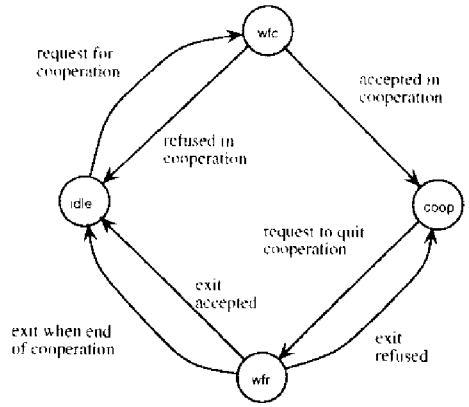


Figure IV.7. Fonctionnement du service d'appartenance dynamique pour un utilisateur.

Il apparaît sur la figure IV.7 qu'un utilisateur du service peut donc se trouver dans quatre états différents :

- IDLE : Il ne coopère pas.
- WFC (ou «Wait For Cooperation») : L'utilisateur a demandé à participer au travail coopératif.
- COOP (ou «Cooperate») : L'utilisateur participe à la coopération et souhaite continuer sa participation.
- WFR (ou «Wait For Release») : L'utilisateur participe à la coopération, mais il souhaite abandonner dès que possible.

Lorsqu'un utilisateur demande à entrer en coopération, il passe de l'état IDLE à l'état WFC. Puis, il attend jusqu'à ce que sa requête ait été prise en compte et qu'un nouveau schéma instantané valide qui le contient soit possible. Si le changement de coopération est accepté ou si aucun agent n'était en coopération, il passe dans l'état COOP. Sinon, son entrée en coopération est refusée et il retourne à l'état IDLE. Le comportement d'un utilisateur pour quitter la coopération est symétrique à celui pour y entrer. Lorsqu'un utilisateur désire quitter la coopération, il passe dans l'état WFR. Puis, il attend jusqu'à ce que sa requête ait été prise en compte et qu'un nouveau schéma instantané valide, dont il ne fait pas partie, soit possible. Si le changement de coopération est accepté, l'utilisateur quitte effectivement la coopération et passe dans l'état IDLE. De la même façon, si tous les utilisateurs désirent quitter la coopération et si aucun nouvel agent ne désire y participer, alors tous les coopérants sortent de la coopération et passent dans l'état IDLE. Par contre, si le changement est refusé, la sortie de l'utilisateur est refusée et il retourne dans l'état COOP.

Relations entre agents

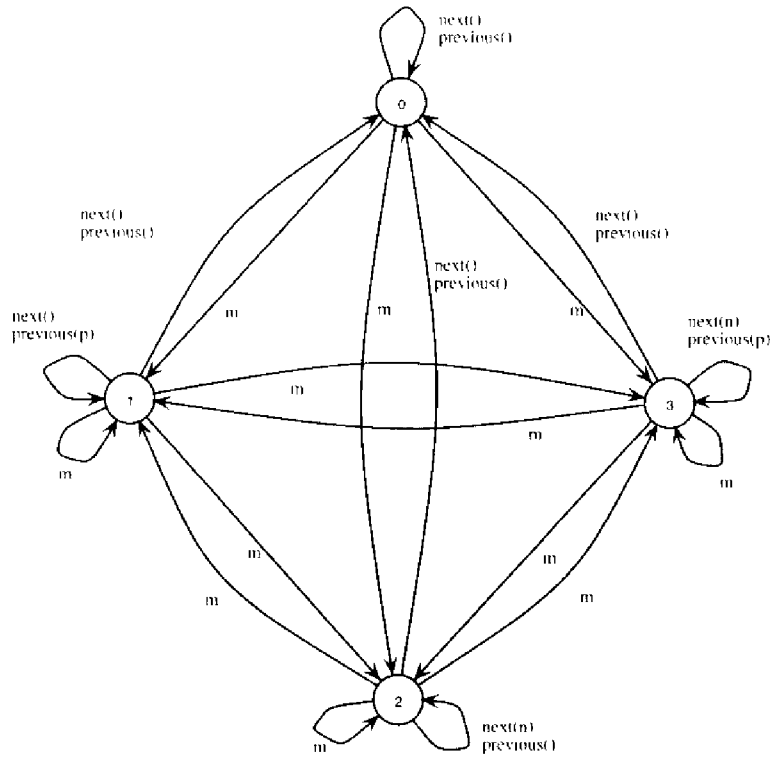


Figure IV.8. Possibilités de partage d'information d'un agent coopérant.

L'automate quotient de la figure IV.8 est une autre projection sûreté qui donne toutes les relations qu'un agent peut avoir avec les autres agents dans un groupe coopératif. Un agent peut connaître des informations provenant d'autres agents qui le suivent dans le graphe de coopération. Ce cas est représenté par le prédicat «next(n)». «next()» signifie qu'un agent n'a pas de suivant, à un instant donné. De la même façon, «previous(p)» signifie qu'un agent fait connaître les informations dont il est propriétaire vers d'autres agents. «previous()» indique qu'aucun agent ne précède l'agent considéré. Lorsque ces deux possibilités sont mises ensemble, quatre cas apparaissent :

- Dans l'état «0», un agent n'est en relation avec personne. Cela correspond au cas où il ne participe pas au travail coopératif.
- L'état «1» est un état dans lequel l'agent est en coopération. Il fait connaître ses informations, mais aucun autre agent ne lui laisse entrevoir de l'information.
- L'état «2» donne le cas où un agent coopérant a connaissance d'information détenue par d'autres agents, mais aucun autre agent n'accède à sa connaissance.
- Dans l'état «3», un agent est en coopération. Il partage ses informations et a connaissance de l'information d'autres agents.

Le graphe décrit le maximum de possibilités qu'un agent peut avoir. Il est en effet possible de passer d'un état vers tous les autres. Le graphe résultant est donc totalement connecté. La transition «m» correspond à un changement de coopération. Plus précisément, il s'agit de l'ordre donné aux agents coopérants d'échanger mutuellement leurs contextes avant de changer de coopération.

Les relations d'un agent avec les autres agents sont bien entendu dépendantes de sa position dans un graphe de coopération. Or, suivant la place d'un agent dans la structure de coopération, les possibilités de communication peuvent être réduites. En effet, dans une structure de coopération hiérarchique par exemple, l'agent racine peut être dans l'état «0» où il ne coopère pas, ou dans l'état «2», où il surveille d'autres agents. Suivant les graphes valides de coopération retenus, certains états ou certaines transitions entre ces états peuvent être supprimés.

Echanges des contextes

L'analyse du protocole a ensuite été menée en regardant le comportement de chaque agent suivant les phases de la coopération. Pour cela, la dernière configuration de la figure IV.6 a été prise, où trois agents sont en coopération. Les graphes instantanés valides sont formés par les ensembles {1, 2}, {1, 3}, {2, 3} et {1, 2, 3}. Le vote utilisé est la majorité.

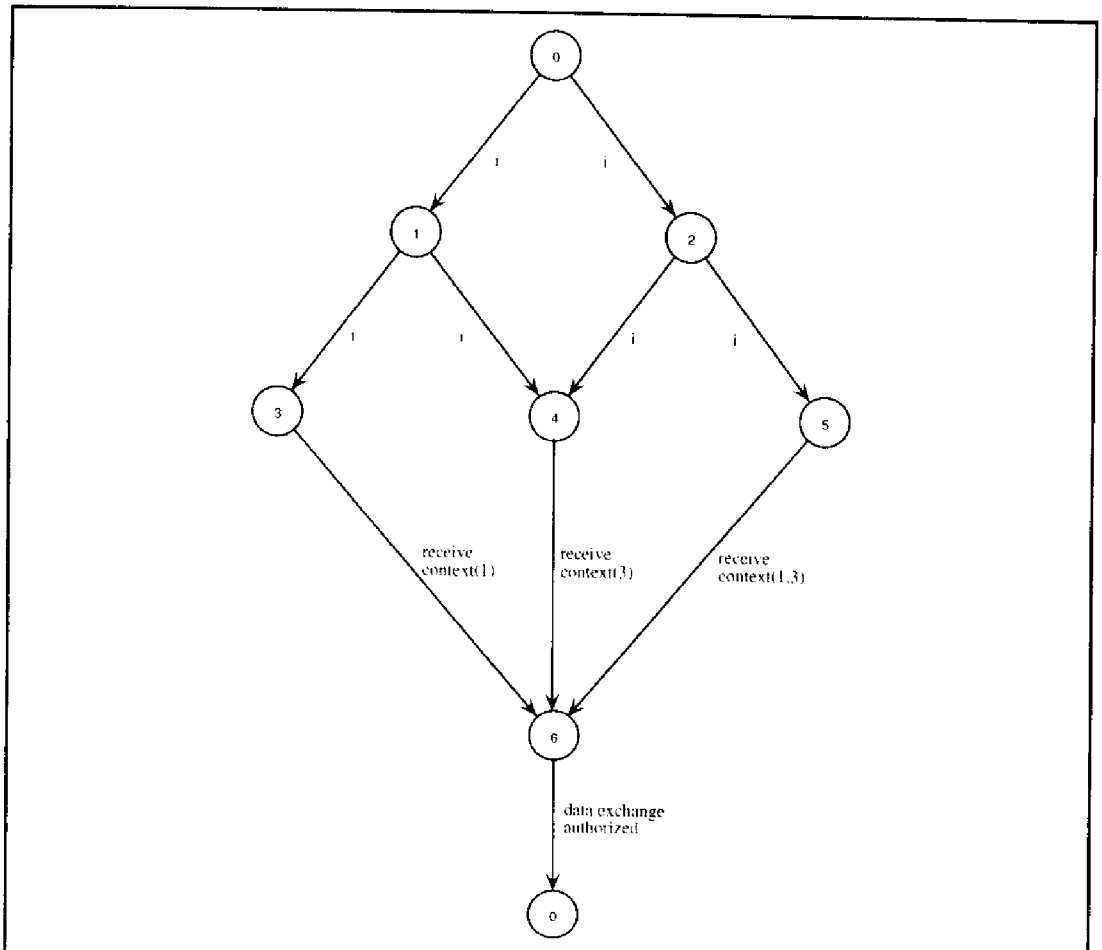


Figure IV.9. Réception des contextes pour l'agent «2».

Le graphe de la figure IV.9 est une projection observationnelle qui montre l'ensemble des possibilités que le site «2» a de recevoir les contextes, c'est-à-dire les informations détenues par les autres agents avec lesquels il est en relation pour chacune des coopérations possibles. Ainsi, il peut recevoir des informations de «1» lorsque «1» et «2» coopèrent, ou bien il peut recevoir des informations de «3» lorsque «2» et «3» coopèrent, ou bien des informations de «1» et «3» lorsque «1», «2», et «3» sont en coopération. Chaque possibilité correspond à une configuration valide dont fait partie l'agent «2». Les transitions internes «i» donnent le passage d'une configuration valide à une autre lors d'un changement de coopération. L'entrelacement de ces tran-

sitions internes «i» limite à chaque étape les choix possibles et introduit des niveaux d'indéterminisme. Il masque en fait une action du système qui a réduit les possibilités de réception de l'agent «2». En effet, à partir de l'état «0», l'agent «2» peut recevoir les contextes de l'agent «1» seul, de «3» seul, ou de «1» et «3». Pour l'état «1», un événement interne, et donc masqué, a fait que cet agent ne peut plus recevoir que les contextes de l'agent «1» seul ou de l'agent «3» seul. De la même façon, l'agent «2» ne peut plus recevoir dans l'état «2» que les contextes de l'agent «3» seul ou des agents «1» et «3». Les états «3», «4», et «5» réduisent encore chacune des possibilités finales. Les événements internes «i» correspondent à des suites d'actions masquées et de ce fait ne sont pas interprétables.

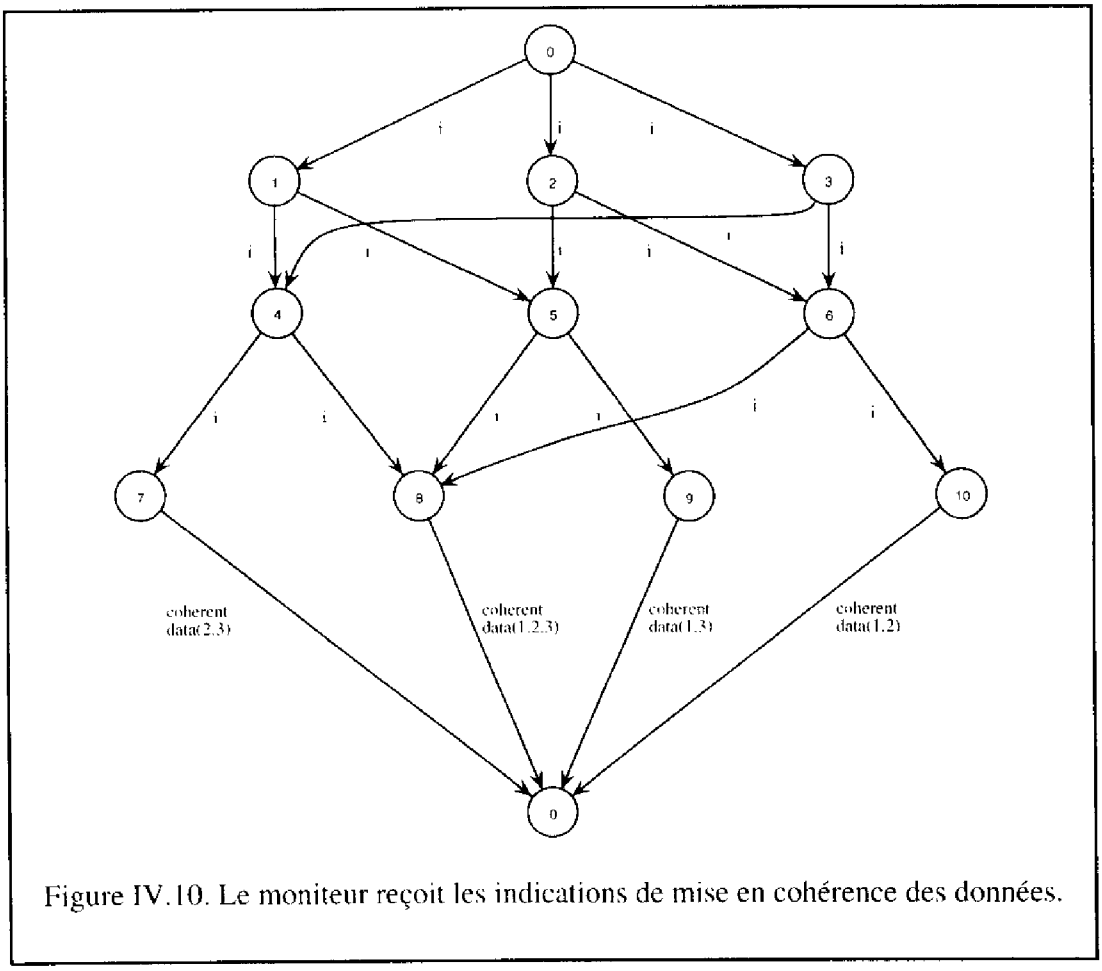


Figure IV.10. Le moniteur reçoit les indications de mise en cohérence des données.

Avant de commencer une coopération, les agents échangent entre eux leurs contextes. Le gestionnaire de coopération reçoit alors de la part de chacun des agents en coopération un message lui indiquant que ses données sont à jour. La figure IV.10 est une projection observationnelle qui montre tous les cas possibles suivant les schémas de coopération. Lorsque «2» et «3» commencent une nouvelle coopération, chaque nouveau coopérant émet vers les agents suivants les informations représentant son contexte propre. Une fois cet envoi terminé, lorsqu'un site a reçu l'ensemble des informations provenant des agents qui le précèdent dans le graphe, il envoie un message vers le gestionnaire de la coopération pour lui indiquer que ses données sont à jour. Lorsque le gestionnaire reçoit le message «coherent_data (2, 3)», alors les agents coopérants «2» et «3» ont leurs données à jour. De même pour «1», «2», et «3». De même pour «1» et «3». De même pour «1» et «2». Chaque possibilité correspond à une nouvelle configuration valide. Le gestionnaire de coopération a donc autant de choix que l'on a de configurations valides. Les

transitions «i» représentent les événements internes qui provoquent les changements de coopération. L'entrelacement des transitions internes «i» limite à chaque étape les choix possibles et introduit des niveaux d'indéterminisme. Il masque en fait une action du système qui a réduit les possibilités futures de réception des données cohérentes pour le moniteur. En effet, à partir de l'état «0», le moniteur peut recevoir les quatre possibilités de cohérence. Puis, un événement interne, et donc masqué, survient et réduit le nombre de possibilités finales à trois parmi les quatre possibles, pour les états «1», «2», et «3». De la même façon, uniquement deux choix parmi les quatre restent possibles pour les états «4», «5», et «6». Le choix devient unique pour les états «7», «8», «9», et «10». Comme les événements internes «i» correspondent à des ensembles d'événements masqués, leur interprétation n'est pas possible.

Emissions de données

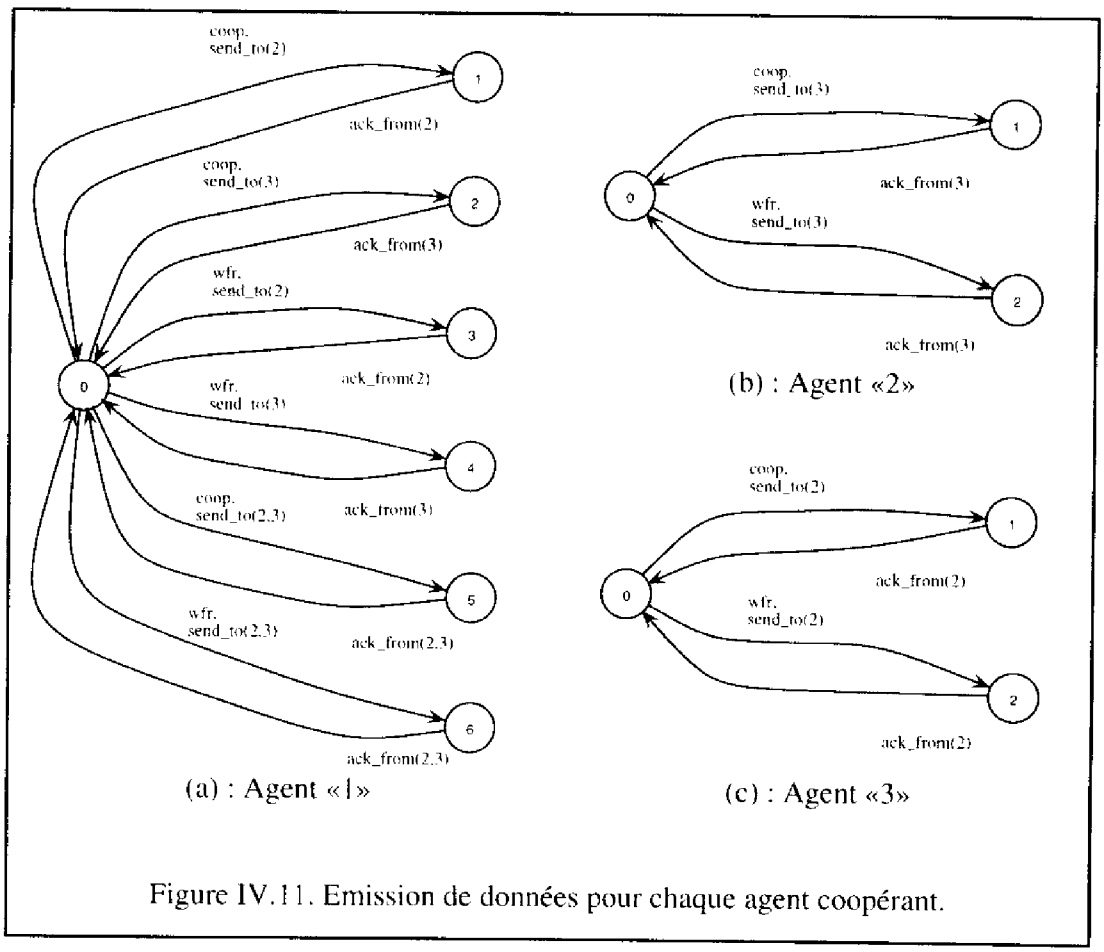


Figure IV.11. Emission de données pour chaque agent coopérant.

La figure IV.11 montre trois projections observationnelles de chacun des agents de la coopération pour l'envoi de données vers les sites avec lesquels ils sont en relation. Ces envois de données sont dépendants de leur état. En effet, les émissions ne peuvent se faire que si et seulement si les sites coopèrent, c'est-à-dire qu'ils sont dans l'état «coop», qui signifie en coopération ou «wfr», qui signifie que l'agent est en coopération, mais qu'il a demandé à sortir. Pour chacun des sites, les émissions respectent la structure courante de la coopération, c'est-à-dire les sites avec lesquels ils sont en relation à l'instant considéré. Pour chaque agent, on a donc deux possibilités pour chacune des configurations valides de la coopération, l'une étant reliée à l'état «coop», l'autre à l'état «wfr».

1.4.2. Vues globales de la coopération

Après avoir considéré les agents seuls, cette section présente un ensemble de projections qui montre l'évolution globale des agents suivant la coopération. Plus précisément, ces vues détaillent le comportement du protocole pour une structure de coopération, les informations que chaque agent possède sur la coopération et les données qu'ils peuvent échanger dans une structure de coopération. Ces projections sont dites globales car elles ne considèrent pas le comportement d'un agent pour toutes les configurations de coopération possibles, mais elles mettent en évidence le comportement de l'ensemble des agents pour une structure de coopération particulière.

La configuration de coopération choisie est la même que celle de la section précédente, avec trois agents dans le graphe de coopération.

Configurations valides de la coopération

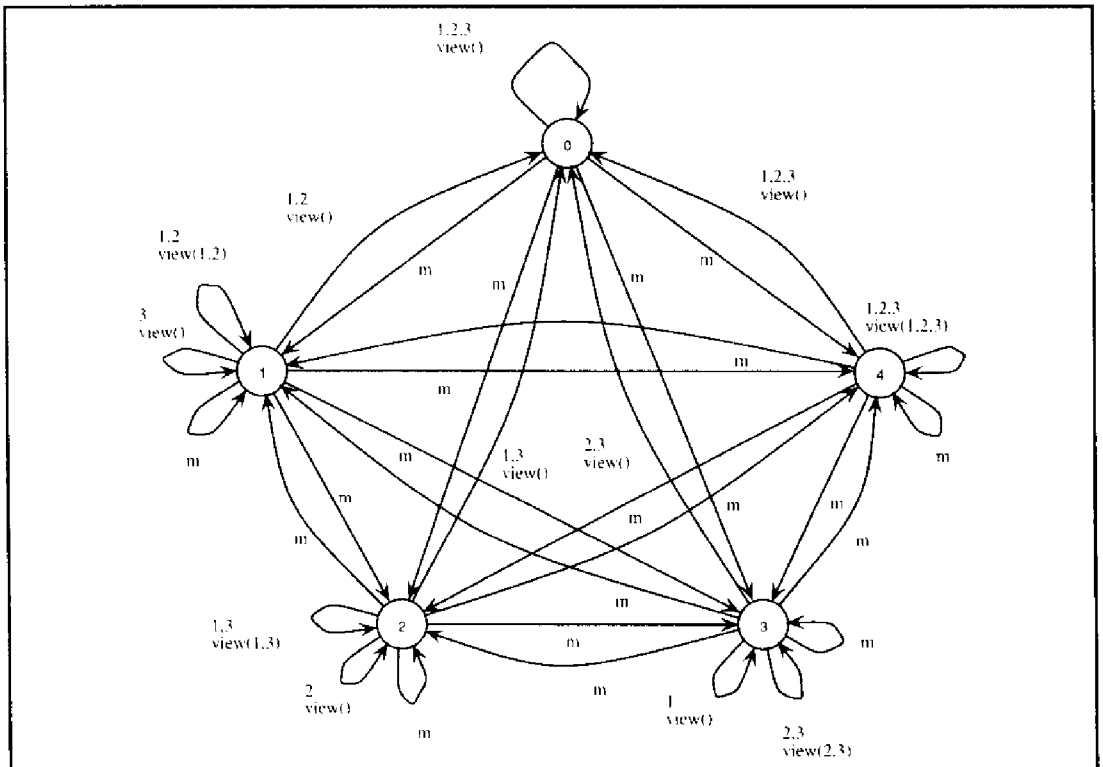


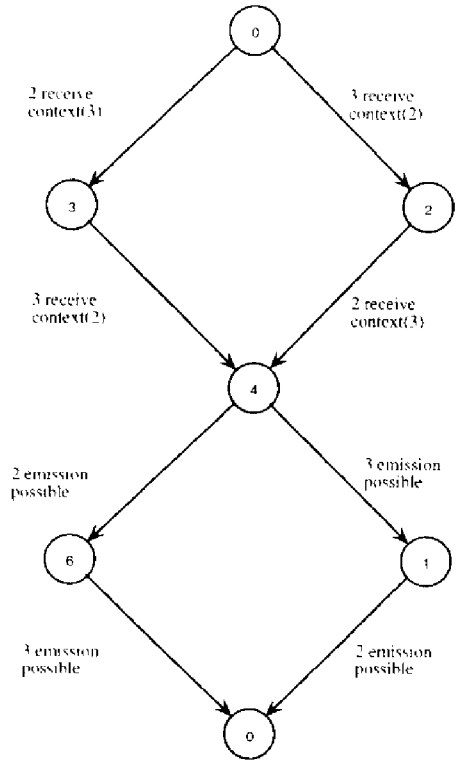
Figure IV.12. Connaissance que les agents ont des diverses coopérations.

La figure IV.12 est une projection sûreté qui donne toutes les configurations de coopération qui peuvent être obtenues à partir du schéma coopératif choisi. L'automate quotient obtenu est composé de cinq états : quatre de ces états représentent une configuration instantanée valide. L'état «0» représente en plus le cas où personne ne coopère. Le graphe est totalement connecté, car il est possible de passer, selon les requêtes, depuis n'importe quelle configuration courante, vers n'importe quelle autre configuration. La transition «m» permet de passer entre certaines configurations valides. Plus précisément, elle correspond à l'ordre donné aux agents d'une nouvelle configuration valide d'échanger leurs contextes initiaux avant de commencer à coopérer. Les flèches qui bouclent sur chacun des états donnent la connaissance de chaque agent sur la configuration valide courante de la coopération. Par exemple, dans l'état «1», les agents «1» et «2» savent qu'ils sont mutuellement en coopération. L'agent «3», qui ne coopère pas, n'a pas

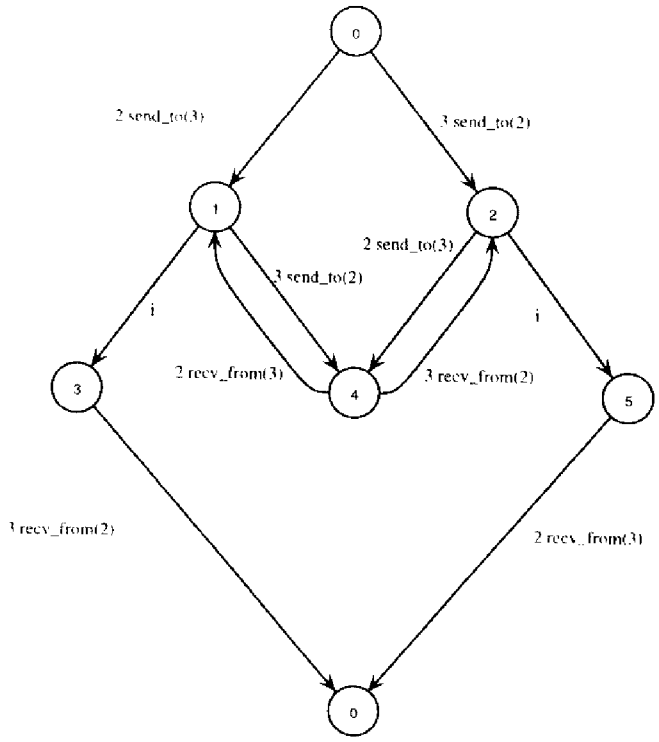
d'information ni de connaissance sur la coopération courante. L'automate obtenu permet de vérifier que les connaissances des agents sur chacun des états ne sont pas incompatibles. Par exemple, deux agents ne croient pas faire partie de deux configurations distinctes en même temps.

Phases de la coopération

Les projections suivantes ont été réalisées lorsque les agents «2» et «3» sont en coopération et que l'agent «1» ne coopère pas. Elles reflètent le comportement des agents pour chacune des quatre phases de la coopération, c'est-à-dire l'échange de contexte initial, la coopération, le vote et la mise en cohérence des données avant le changement de structure coopérative.



(a) : Acquisition des contextes lorsque le nouveau groupe coopératif est {2, 3}.

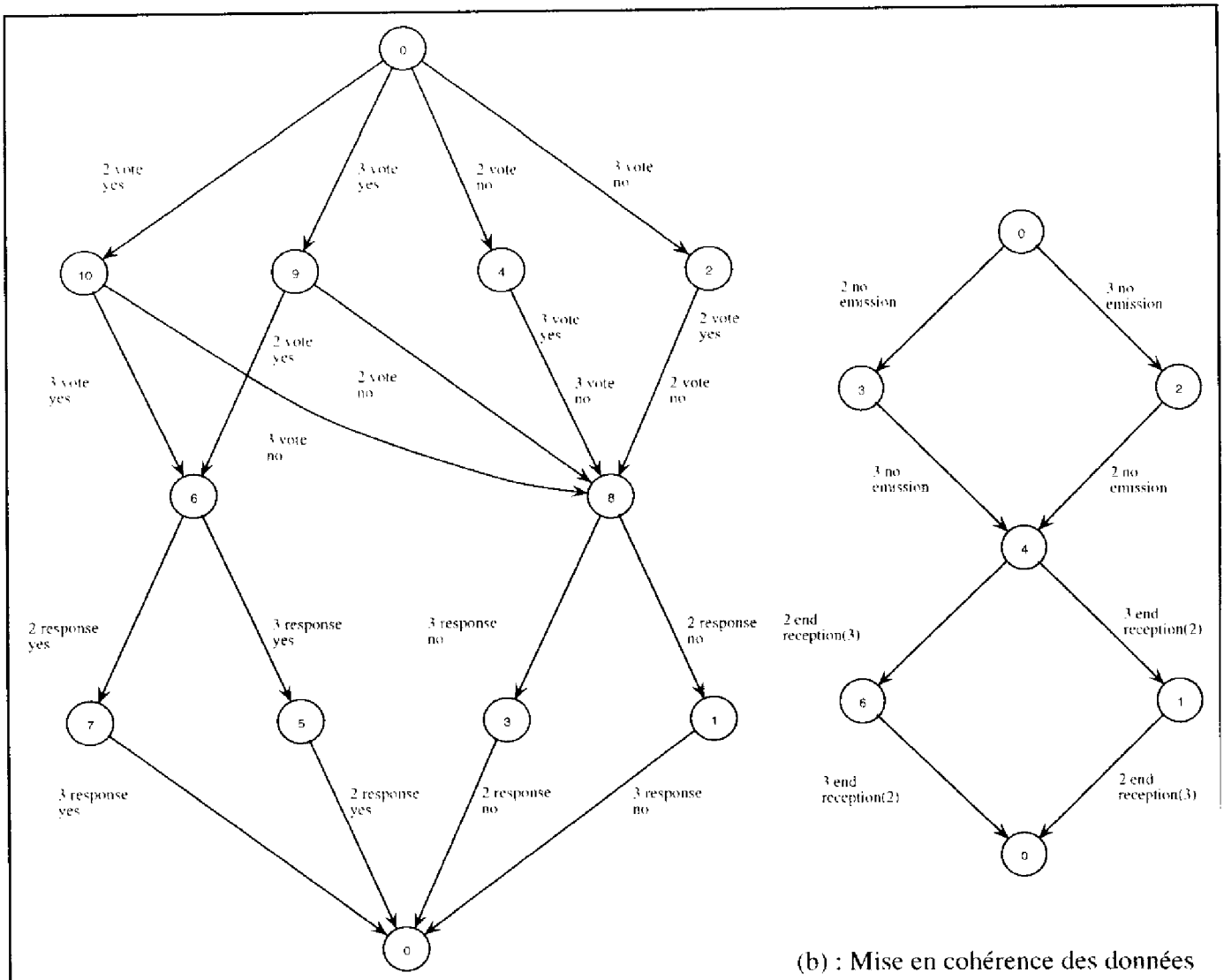


(b) : Echanges de données lorsque le groupe coopératif est {2, 3}.

Figure IV.13.

La figure IV.13(a) est une projection observationnelle qui montre le déroulement de la phase d'échange de contextes, lorsque les agents «2» et «3» forment une nouvelle coopération. D'après la structure de la coopération, comme chacun de ces deux agents a connaissance du contexte de l'autre, ils échangent mutuellement leurs données contextuelles. Cet échange se termine à l'état «4». Puis, chaque agent prend connaissance que l'ensemble des données de la coopération sont cohérentes et reçoit donc l'ordre de commencer la coopération.

Les agents peuvent émettre de nouvelles données traduisant la progression du travail coopératif. La figure IV.13(b) est une autre projection observationnelle qui montre tous les échanges de données possibles lorsque les agents «2» et «3» coopèrent. Les agents «2» et «3» peuvent s'envoyer mutuellement des données. Le graphe résultant montre les combinaisons et les entrelacements entre les émissions et les réceptions des agents coopérants.



(a) : Vote lorsque le groupe coopératif est {2, 3}.

(b) : Mise en cohérence des données lorsque le groupe coopératif est {2, 3}.

Figure IV.14.

L'automate quotient de la figure IV.14(a) est une projection sûreté du système lorsque les agents «2» et «3» sont en coopération. Cette projection donne le déroulement du vote. Depuis l'état «0», les agents «2» et «3» votent pour accepter ou pour refuser un changement de configuration valide. Comme le vote se fait à la majorité, il suffit qu'un des deux agents coopérants refuse pour que l'ensemble du changement soit refusé. Le changement est accepté lors de l'arrivée à l'état «6», il est refusé dans l'état «8». Les agents attendent alors le verdict du vote qui leur est communiqué. On vérifie que le vote se déroule bien à la majorité, c'est-à-dire que l'état «6» n'est atteint que si les deux agents ont bien accepté le changement et on regarde également que le verdict du vote communiqué aux agents n'est ni contradictoire entre les agents, ni avec le résultat du vote.

La figure IV.14(b) est une projection observationnelle qui décrit la mise en cohérence des données entre les agents «2» et «3», avant de changer de configuration de coopération. Les agents reçoivent chacun l'ordre de mettre en cohérence leurs données. Cet ordre est pris en compte par les deux agents à l'état «4». Puis, comme chaque agent est en relation avec l'autre, l'agent «2» indique à l'agent «3» qu'il ne lui enverra plus de donnée, et réciproquement. En retournant à l'état «0», les agents savent alors que leurs données sont dans un état cohérent.

Remarques :

Les projections effectuées et les propriétés vérifiées dépendent fortement de la configuration de coopération choisie, du choix des graphes instantanés valides et du type de vote utilisé. Le but essentiel était cependant de vérifier que le système respectait bien les différentes phases de la coopération et que son comportement suivait bien le protocole d'appartenance à la coopération. Les automates quotient obtenus reflétaient indirectement la configuration de coopération choisie. Une autre direction de vérification aurait été de chercher à vérifier des propriétés globales indépendantes de la structure choisie, bien que la principale préoccupation de notre vérification était de tester si le système programmé respectait bien le protocole défini.

La vérification par génération du graphe d'accessibilité et par bissimulation est cependant une démarche lourde, car les graphes manipulés ont un grand nombre d'états. L'analyse a pu s'effectuer pour des configurations de trois agents coopérants mais, très vite, pour des graphes coopératifs plus importants, la taille de ces graphes subit une explosion combinatoire, ce qui les rend trop volumineux pour pouvoir être stockés et étudiés. A titre d'exemple, le graphe d'accessibilité utilisé pour le groupe coopératif de trois coopérants, décrit sous format texte, nécessite un fichier de 7,5 mégaoctets pour stocker les états et un fichier de 6 mégaoctets qui contient la description des transitions. Avec de tels volumes de données, le filtrage et le renommage des transitions nécessitait une demi-heure, les projections (quand les machines ne tombaient pas en panne pour insuffisance de mémoire) prenant parfois plusieurs heures.

L'analyse effectuée a cherché à couvrir les principales phases du fonctionnement du protocole. La spécification VAL, au moyen de ces diverses projections, nous semble suffisamment étudiée.

2. Réalisation d'une implantation semi-automatique

Le langage Estelle peut être compilé. En particulier, l'environnement de programmation EDT utilisé possède un générateur de code C [EC92], c'est-à-dire qu'il est capable de traduire une spécification Estelle en C. De plus, il est possible, à l'intérieur de la spécification Estelle, d'inclure des parties de code écrites directement en C et de faire appel à des routines ou fonctions C. Cette propriété se révèle intéressante lorsque l'on désire utiliser et récupérer la spécification réalisée alors à des fins d'implantation. Par la suite, la démarche retenue pour dériver la spécification vers une implantation est détaillée et analysée.

2.1. Principe d'implantation

La technique de modification de la spécification Estelle en vue de son implantation consiste essentiellement à modifier les connexions utilisées entre les modules Estelle. Dans une spécification Estelle, les modules sont reliés entre eux par des files bidirectionnelles. Il s'agit de remplacer certaines files Estelle par un canal de communication réel, canal qui permet de faire effectivement communiquer les modules qui peuvent être répartis sur plusieurs sites physiques. Les créations, les connexions, et les communications réelles ne se font plus avec les primitives

d'envoi et de réception fournies par Estelle, mais font appel à des fonctions interface qui réalisent les appels système pour l'envoi et la réception de messages sur les connexions établies. Dans un environnement UNIX, les communications sur une même machine peuvent être basées sur les «Inter-Process Communications» comme les tubes ou les files de message. Par contre, les canaux Estelle entre machines peuvent être remplacés par l'utilisation de primitives de transport comme les «sockets» ou la «Transport Layer Interface» s'appuyant sur les protocoles TCP ou UDP, pour des communications distantes.

Chaque instance de module Estelle communiquant par un moyen de communication réel est alors incluse dans un processus UNIX. Certaines transitions font alors appel à des fonctions externes interface écrites en C. Puis, le code du moteur d'implantation est alors ajouté au processus pour assurer l'exécution de ce dernier. Dans un cas plus général qui n'a pas été nécessaire ici, un ensemble de modules Estelle peuvent être regroupés dans un même processus UNIX.

La figure IV.15 donne de façon plus détaillée la transformation effectuée sur deux modules Estelle qui communiquaient initialement par un canal, qui se trouvent séparés et reliés par un moyen réel de communication.

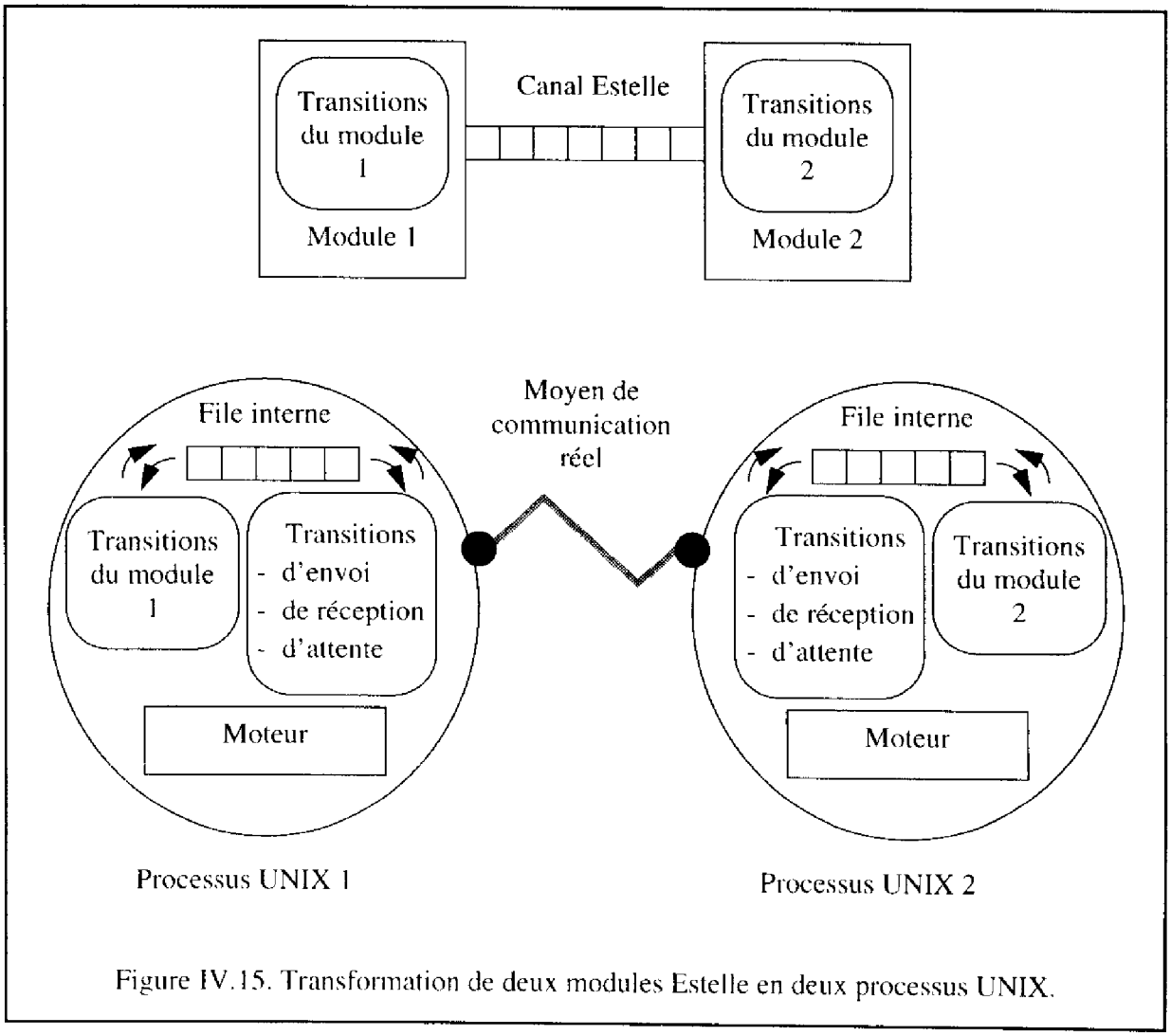


Figure IV.15. Transformation de deux modules Estelle en deux processus UNIX.

La file de communication externe Estelle est transformée en une file de communication interne au processus. Cette file permet de stocker temporairement les messages à émettre vers l'extérieur du processus, ainsi que les messages reçus, en attente d'être pris en compte par le moteur d'implantation. L'utilisation de cette file interne présente un gros avantage : elle permet de garder identique le code Estelle initial décrivant les transitions. Les transitions du module sont ainsi réutilisées telles quelles, sans aucun changement. Cette approche, bien que pouvant être pénalisante en performances, est donc un gage de ne pas ajouter d'erreur dans le module récupéré car son contenu n'est absolument pas modifié.

Pour réaliser les communications externes, trois groupes de transitions supplémentaires sont utilisés : le premier lit un message depuis la file interne lorsque celle-ci est non vide, et le retransmet sur le port de communication réel. Le deuxième groupe lit le premier message disponible sur le port réel, et écrit ce message dans la file interne. La troisième transition est une transition de plus basse priorité. Elle met le processus en attente d'un événement externe tel que l'arrivée d'un message, lorsque plus aucune action interne n'est possible dans ce processus.

Cette organisation permet une séparation stricte entre les transitions réalisant le comportement du module et celles chargées de l'interface avec le monde extérieur au processus UNIX. De plus, cette séparation permet de remplacer l'ensemble des transitions décrivant un module par un autre ensemble, sans toucher aux transitions d'interface, ce qui évite nombre d'erreurs de réécriture ou d'adaptation. Plusieurs versions d'un même module peuvent être facilement testées et implantées si leur architecture extérieure ne change pas.

L'ensemble des transitions d'un processus est sous la responsabilité du moteur d'implantation inclus sous la forme d'une librairie binaire. A chaque étape, ce moteur teste la ou les transitions tirables, en sélectionne une, et exécute le code qui lui est associé.

Une autre avantage de cette organisation est de pouvoir facilement mettre le processus en attente bloquante d'événements externes, et d'éviter des attentes actives qui font appel inutilement au processeur et de ce fait dévastent les performances de la machine hôte. Au départ, l'ensemble de la spécification Estelle forme un système fermé. En effet, aucun événement extérieur ne peut survenir et influencer le comportement du système global. Cependant, le découpage et la répartition dans les processus transforment chaque module Estelle en un système ouvert. Chaque module inclus dans un processus peut recevoir des informations externes au processus via les canaux de communication externes. Le moteur d'implantation sélectionne une transition sensibilisée du processus et l'exécute. Cependant, il peut arriver un instant où plus aucune transition n'est activable. Seul un événement externe au processus comme l'arrivée d'un message pourra activer de nouveau une transition. Une première implantation naïve est de laisser tourner le moteur. Ce dernier continue de tester les transitions, en effectuant alors une attente active. La solution retenue pour l'implantation est d'ajouter une transition interface qui réalise une attente bloquante au moyen de l'appel système «poll». Cette transition, de plus basse priorité que les autres, n'est activée par le moteur que lorsque plus aucune autre transition n'est sensibilisée. Son exécution bloque le processus. Le moteur cesse alors son activité qui ne reprendra que lors de l'arrivée d'un message externe.

Le principal reproche que l'on peut faire à cette approche est l'utilisation de files de messages comme tampons internes avant de réaliser toute communication externe. En effet, ces files entraînent une recopie qui peut s'avérer pénalisante pour des messages de grande taille. Cependant, dans l'implantation réalisée par la suite, les messages transmis sont relativement courts, moins d'une centaine d'octets chacun, et ne nécessitent pas des débits de transmission importants. Une solution pour éviter les recopies est de supprimer les files internes de messages et de modifier directement le code de certaines transitions du module initial. Cette approche est toutefois une source d'erreur non négligeable. Pour un envoi, la transformation est assez facile. L'envoi dans une file d'attente est simplement remplacé par la primitive permettant un envoi réel. Par contre, la réception est beaucoup plus difficile à substituer directement. En effet, les primitives système réelles pour recevoir un message sont beaucoup moins puissantes que celles offertes par Estelle pour recevoir un message depuis une file d'attente. Estelle permet par exemple d'analyser le contenu d'un message avant de le recevoir effectivement et de le traiter. En utilisant des primitives réelles, il est nécessaire de recevoir et de stocker un message avant de pouvoir l'analyser. De plus, pour ne pas pénaliser le tir d'autres transitions, les attentes de messages doivent être non bloquantes, ce qui entraîne des attentes actives qui gaspillent les ressources du processeur. Une autre solution plus simple est de ne pas utiliser ces primitives très puissantes en Estelle, et de se contenter dès le départ de réceptions simples. Cependant, la spécification initiale est plus ardue. En dernier lieu, la modification directe ne permet pas de bien séparer les parties de code faisant interface et celles décrivant le comportement du module initial. Toute modification de ce module ne permet pas de garder intactes les transitions interface.

2.2. Architecture générale de l'implantation

La spécification Estelle initiale sert de contrôle pour l'entrée et la sortie en coopération des agents utilisateurs. Cette partie a été conservée telle quelle. Par contre, la partie échange des données a été découplée et gérée par d'autres entités du niveau utilisateur. Cette séparation a été faite pour deux raisons : tout d'abord, les messages pour le contrôle de la coopération et les messages d'échanges de données sont sémantiquement différents. Les premiers sont courts et doivent être transportés sans perte, de façon fiable. Les données peuvent être de granularités diverses, provenir de médias différents, et, suivant leur sémantique, leur transport nécessite des services de communication possédant des propriétés différentes. Le deuxième avantage est que la partie contrôle peut être modifiée sans influencer la partie qui s'occupe du transport des données, et vice-versa. Par conséquent, les données ne transitent plus par le service d'appartenance dynamique de la coopération, pour éviter d'être pénalisées par la traversée des entités de protocole. Cependant, la partie qui contrôle les données reste sous la dépendance des entités du service d'appartenance dynamique.

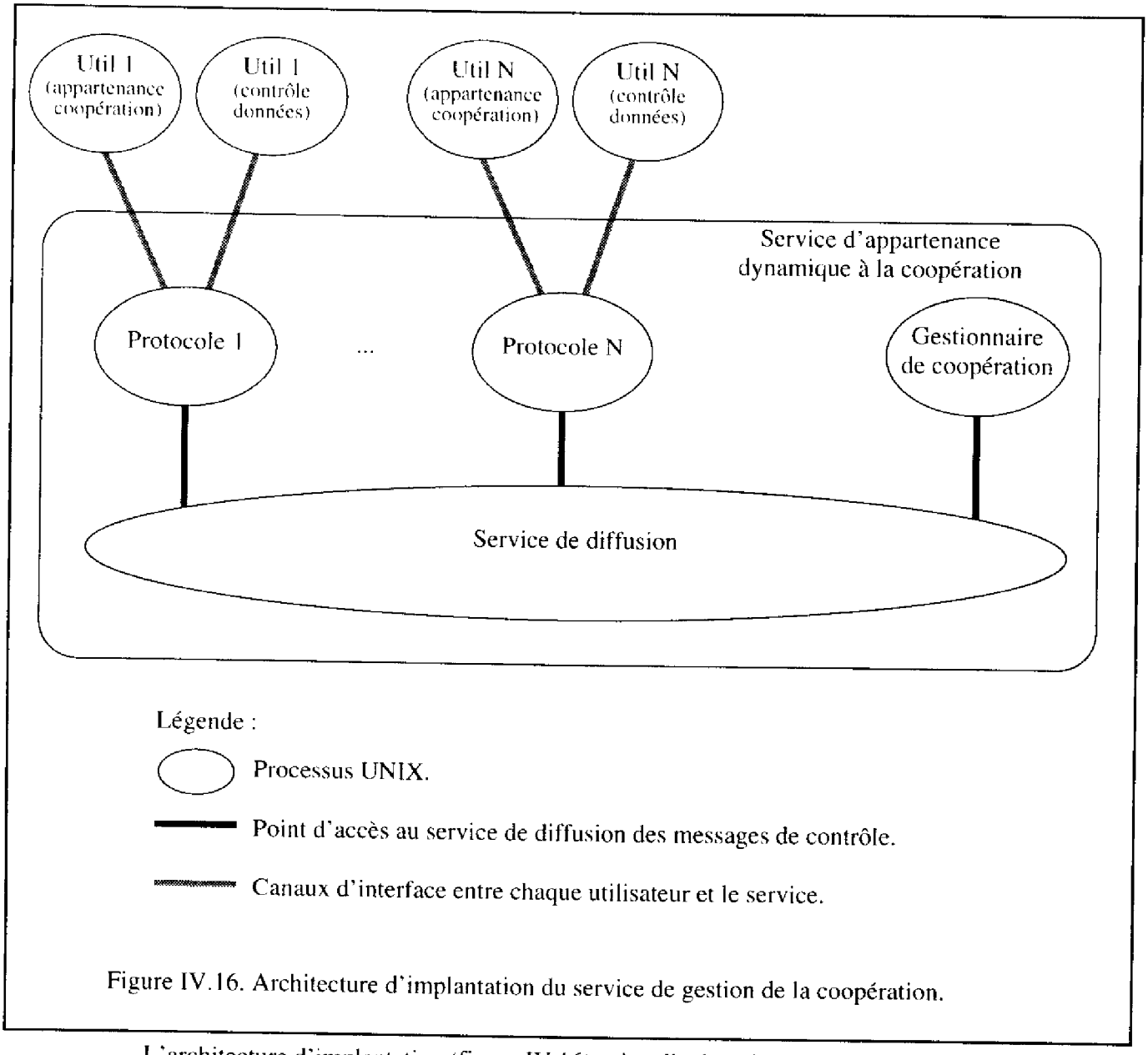


Figure IV.16. Architecture d'implantation du service de gestion de la coopération.

L'architecture d'implantation (figure IV.16) suit celle donnée par la spécification Estelle.

Le service de diffusion est réalisé au moyen d'un processus centralisateur UNIX. Il offre un service de diffusion fiable aux entités du protocole d'appartenance dynamique et véhicule ainsi entre elles les PDUs nécessaires au contrôle de la coopération. Ce service utilisé est quelque peu générique et pourrait être substitué par celui fourni par une plate-forme distribuée. A terme, nous supposons qu'un tel service fiable sera disponible dans la technologie ATM.

Les entités du protocole d'appartenance dynamique et le gestionnaire de coopération se connectent au service de diffusion au moyen d'un ensemble de primitives de service détaillées par la suite. Cette connexion est basée sur l'utilisation de l'interface de transport normalisée «Transport Layer Interface» (TLI) et fait appel au protocole fiable «Transport Control Protocol» (TCP).

Les processus protocole contiennent chacun une entité de protocole Estelle. Le processus gestionnaire de la coopération regroupe l'ensemble des transitions contenues dans le module gestionnaire de la spécification Estelle.

Pour assurer une plus grande modularité entre la partie concernant les échanges de données et les requêtes d'entrée et de sortie de la coopération, chaque utilisateur a été découpé en deux processus. Le premier processus utilisateur se charge uniquement des demandes pour participer ou pour quitter la coopération, affiche si ces demandes sont acceptées ou refusées, et permet d'effectuer un vote lors d'un changement potentiel de groupe. Le deuxième processus se charge directement du contrôle des données dont un utilisateur est propriétaire. Il se charge de l'acheminement des données vers les membres en relation avec l'utilisateur concerné. Ce processus est toutefois sous le contrôle du service d'appartenance dynamique qui lui ordonne d'effectuer un échange de contexte initial lors d'une nouvelle coopération et d'obtenir un état cohérent à la fin de chaque configuration de coopération.

Les échanges de primitives de service entre les deux processus utilisateurs et leur unité de protocole respective se font au moyen de deux files bidirectionnelles réalisées à partir des «Inter-Process Communications» tubes nommés.

2.3. Détail des entités de l'implantation

Cette partie présente en détail chacun des types de processus créés et utilisés pour l'implantation. Elle précise également les parties de code Estelle récupérées, et les comportements utilisés.

2.3.1. Service de diffusion

Description

La plate-forme utilisée est composée de machines SUN fonctionnant avec le système d'exploitation SOLARIS2. Dans ce système, un service de diffusion fiable n'est pas fourni en standard, bien que des extensions du protocole Internet accessibles permettent d'effectuer des diffusions non fiables. Par conséquent, ce service a été réalisé au moyen d'un processus UNIX centralisateur représenté sur la figure IV.17. Ce processus est un processus serveur, suivant le mode de connexion TCP. Il se connecte au réseau avec les primitives de la TLI et attend sur un port que d'autres processus clients entrent en relation avec lui. Puis, il établit une connexion TCP avec le client pris en compte, et peut alors effectuer un dialogue. Pour profiter des facilités offertes par SOLARIS2, le serveur a été «multithreade». Un «thread», ou processus léger, se charge des demandes de connexion des clients. A chaque nouveau client connecté au service, un nouveau «thread», qui se charge de la connexion, est créé. Le rôle de ce «thread» est très simple : il attend un message sur la connexion qu'il gère, puis il rediffuse ce message vers les connexions des destinataires. Puis, lorsqu'un client se déconnecte, il libère les ressources allouées à la connexion et se termine. L'ensemble de ce service a été codé en C, mais son fonctionnement est très proche de celui donné par le module service de diffusion Estelle. L'utilisation d'un service de diffusion fiable plus réel comme dans ATM se ferait assez aisément.

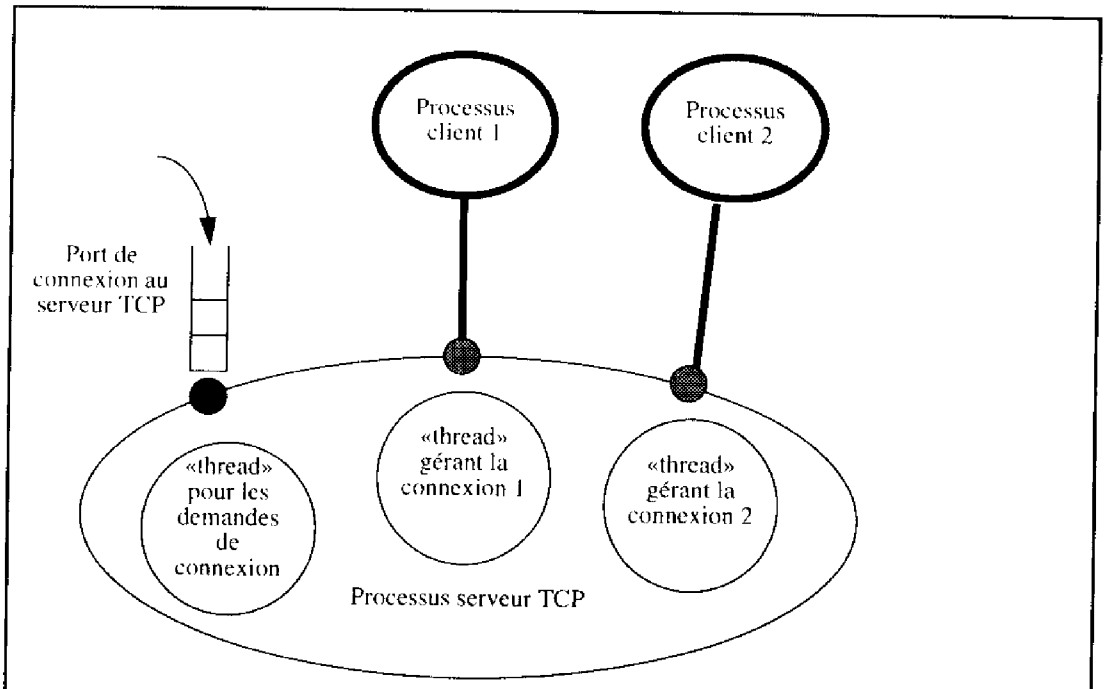


Figure IV.17. Implantation du service de diffusion avec un serveur TCP.

Accès

Pour permettre la connexion des clients au service de diffusion, un ensemble de primitives écrites en C utilisant la TLI ont été définies. Ces fonctions forment une librairie pour l'accès au service de diffusion. Le rôle de ces fonctions est le suivant :

- int connexion_service (int identite);

  /* cette primitive permet de se connecter au service de communication. Un processus client donne son identité. Le résultat retourné est le descripteur de fichier de la connexion établie, s'il est positif. S'il est négatif, il indique la raison pour laquelle la connexion a été refusée. */

- int deconnexion_service (int tfd);

  /* ferme une connexion. tfd est le descripteur de fichier de la connexion */

Un deuxième type de fonctions interface est utilisable pour les communications avec le service de diffusion, mais aussi pour tout descripteur de fichier d'un processus. Notamment, ces fonctions serviront également pour les communications à travers les tubes.

- BOOLEAN attente_message (int *tab_fd, int nb_fd);

  /* réalise une attente bloquante sur un ensemble de descripteurs de fichier. Retourne TRUE si aucune erreur n'est survenue. */

- int recupere_port_actif ();

  /* renvoie le descripteur de fichier sur lequel un message peut être lu. */

- int lecture_bloquante (int fd, void* buffer, int nb);

  /* lit de manière bloquante nb octets depuis le descripteur de fichier fd, et les place dans la zone buffer. Souvent utilisé après une attente de message. */

- int write (int fd, void* buffer, int nb);

  /* appel système pour écrire vers le descripteur de fichier fd, nb octets depuis la zone buffer. */

2.3.2. Entités de protocole

Description des processus protocole

L'ensemble du code Estelle provenant de la spécification initiale et décrivant le comportement d'un module protocole a été récupéré sans aucune modification, en suivant le principe développé dans la section précédente. L'adaptation est présentée sur la figure IV.18.

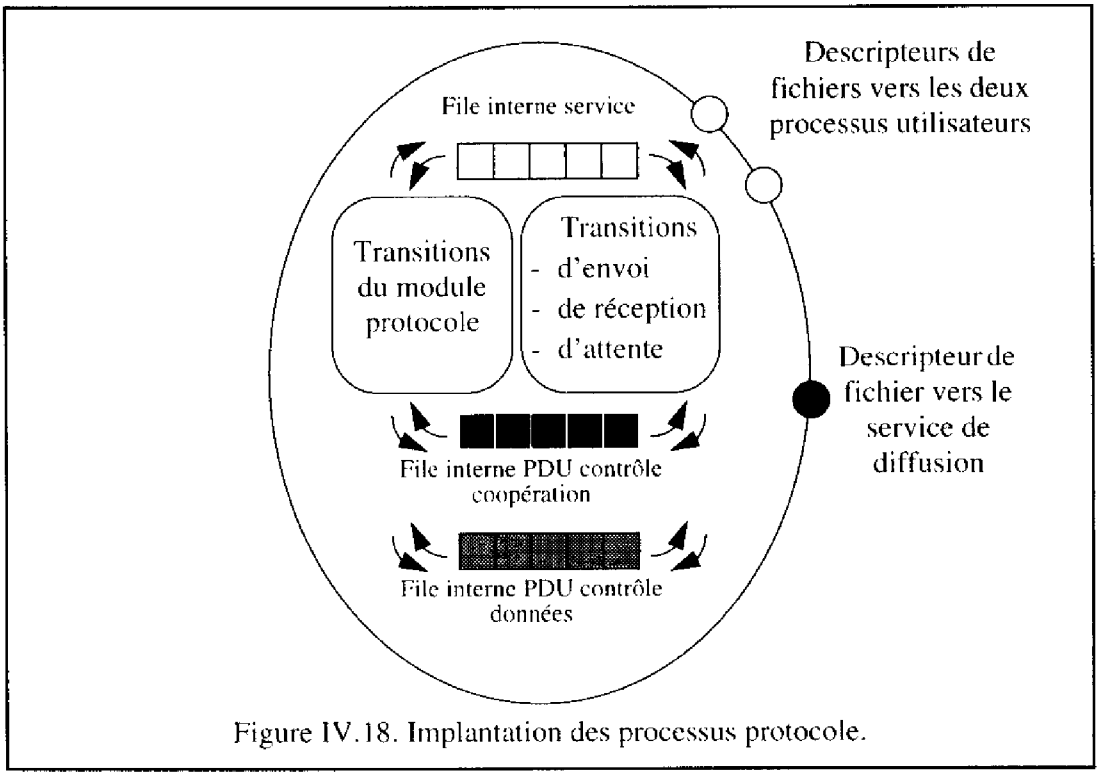


Figure IV.18. Implantation des processus protocole.

Les trois canaux initiaux sont devenus trois canaux internes. Par contre, les transitions qui font office d'interface sont un peu plus complexes. Les transitions qui s'occupent de l'interface avec les deux processus utilisateurs analysent les primitives à diffuser vers l'utilisateur et transmettent cette primitive soit vers l'un des deux processus concernés, soit vers les deux. A l'inverse, toute primitive provenant de l'un des deux processus utilisateurs est regroupée dans la même file d'attente. Pour les échanges de PDUs, un multiplexage a été effectué. Deux grandes catégories de PDUs sont échangées : les premiers contrôlent les entrées et sorties de la coopération ainsi que les votes, les deuxièmes s'occupent de la synchronisation des échanges de données avec l'évolution de la coopération. Chacun de ces types de PDUs possède à l'origine un canal Estelle. Comme tous ces PDUs sont courts, afin de ne pas gaspiller les connexions réelles, leurs échanges ont été regroupés sur la même connexion physique reliée au service de diffusion. Les transitions interface se chargent donc de multiplexer et de démultiplexer les PDUs provenant de la même connexion.

Description du processus gestionnaire

L'adaptation effectuée pour le module gestionnaire de la coopération est similaire à celle utilisée pour les entités de protocole (figure IV.19). Le découpage en deux sous-modules est cependant supprimé. Le sous-module consistance du groupe, qui donne les règles de validité des nouveaux graphes instantanés est finalement codé directement en C, pour pouvoir être étendu par la suite, en changeant les configurations acceptées. L'ensemble des transitions du moniteur sont récupérées telles quelles. Les deux canaux faisant le lien avec le service de diffusion deviennent deux canaux internes. Comme une seule connexion est utilisée vers le service de diffusion, les transitions interface se chargent du multiplexage.

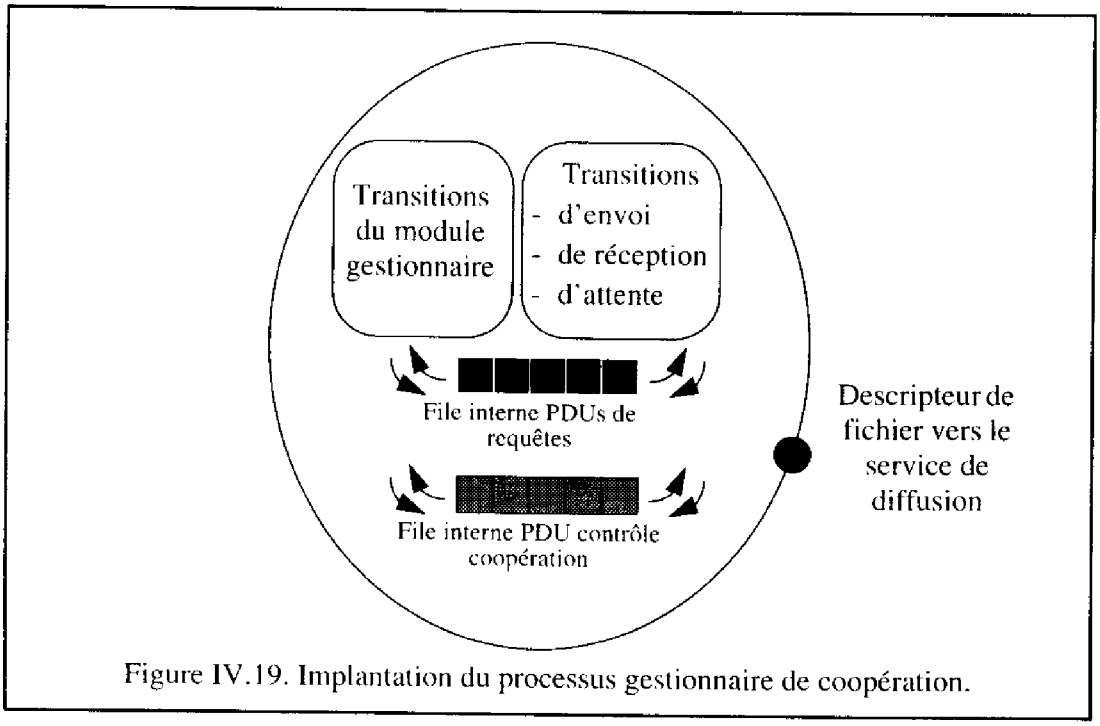


Figure IV.19. Implantation du processus gestionnaire de coopération.

Accès au service d'appartenance

Les processus utilisateurs et leurs entités de protocole respectives se trouvent toujours sur le même hôte. Par conséquent, leurs échanges de messages se font en utilisant les possibilités de communication inter-processus sur une même machine. Le moyen retenu est l'utilisation de tubes nommés. L'avantage des tubes est leur forte similarité avec le fonctionnement des connexions TCP : les processus y accèdent au travers de descripteurs de fichiers et de ce fait il est possible d'utiliser les mêmes primitives système d'attente de message, de lecture et d'écriture que pour les connexions TCP. Notamment, les fonctions «attente_message», «recupere_port_actif», «lecture_bloquante» et «write» sont réutilisables directement. Les tubes fournissent cependant un moyen de communication unidirectionnel. Une librairie composée des fonctions suivantes a donc été créée pour les associer par paires afin de former des canaux bidirectionnels.

- int creation_double_tube (char *nom);

  /* crée un canal formé de deux tubes nommés. Le champ nom sert à identifier ce tube. */

- int destruction_double_tube (char *nom);

  /* détruit le double tube du nom considéré */

- connexion_double_tube (char *nom, int extremite, int *fd_read, int *fd_write);

  /* connexion à une extrémité d'un double tube. fd_read est le descripteur de fichier d'où proviennent les données lues, fd_write est descripteur de fichier pour écrire. */

- deconnexion_double_tube (int fd1, int fd2);

  /* déconnexion d'un double tube */

Les processus protocole créent les double tubes qui vont servir à la connexion des processus utilisateurs, puis se connectent à ces double tubes. De l'autre côté, chaque processus utilisateur se connecte également. Les communications entre les utilisateurs et le service peuvent alors avoir lieu.

Les primitives de service sont codées sous la forme de messages avec un champ spécial pour identifier la primitive, sont transmises avec la primitive système «write» et sont reçues au moyen de la fonction «lecture_bloquante».

2.3.3. Processus utilisateur

Les utilisateurs sont chacun représentés par deux processus, le premier se chargeant du contrôle des participations de la coopération, le second gérant les données qui appartiennent à un utilisateur.

Processus utilisateur pour l'appartenance

Ce processus permet à un utilisateur d'effectuer une requête pour entrer en coopération ou pour quitter cette coopération, affiche un ensemble d'informations qui renseignent sur la configuration courante du groupe et sur les relations avec les participants, puis permet d'effectuer un vote lors d'une modification possible de la coopération. L'ensemble de ces actions et de ces informations sont représentées au moyen de trois fenêtres X Window dont il est propriétaire.

La première fenêtre est le panel de coopération, dont un exemple est donné sur la figure IV.20. La zone la plus haute donne le nom du coopérant, la machine sur laquelle il est connecté, ainsi qu'un numéro identificateur unique. La zone de gauche donne l'état courant de l'utilisateur (attente de coopération, coopération, attente de sortie, vote...). La zone de droite donne la liste des coopérants à un instant donné. Les astérisques à côté des noms indiquent vers quel coopérant on envoie ses informations propres. Plus bas, une autre zone affiche un message d'information associé à l'état courant. En bas de la fenêtre, un ensemble de boutons, actifs ou inactifs suivant l'état dans lequel on se trouve, donne les possibilités d'action de l'utilisateur.

La deuxième fenêtre, présentée sur la figure IV.21, est une fenêtre d'information qui indique si l'entrée ou la sortie de coopération a été refusée ou non.

La dernière fenêtre, sur la figure IV.22, permet de réaliser un vote. La zone centrale montre la nouvelle configuration potentielle en indiquant le nom des coopérants de la nouvelle coopération. L'astérisque indique toujours les coopérants avec lesquels on reste en relation. Les boutons du dessous permettent de valider son choix.

L'ensemble du code de ce module a été programmé en C, car l'interfaçage entre Estelle et les primitives X Window s'avère difficile à gérer. Cependant, le comportement codé dans le module Estelle utilisateur a pu être entièrement réutilisé. En effet, le principe de programmation des fenêtres X Window, plus précisément celui du «Toolkit» OpenLook, est de décrire un système état / transition. L'état global du programme est donné par l'ensemble des variables globales, et par l'état du contenu des fenêtres associées au programme. Le système attend un événement externe provenant du serveur X Window auquel il est connecté, par exemple suite à une action sur une fenêtre. Une fois l'événement reçu, ce dernier est traité de façon atomique par un ensemble

de fonctions C (appelées aussi «callbacks») associées à l'événement. Ces fonctions modifient l'état courant en changeant les valeurs des variables globales et en modifiant le contenu des fenêtres. Estelle est aussi un langage basé sur le formalisme état / transition. En Estelle, l'état courant d'un module est donné par la valeur de l'état majeur de la machine à états et par les valeurs de l'ensemble des variables globales au module.

L'état courant de la machine à états du module utilisateur Estelle a donc été codé sous la forme d'une variable C. Les variables globales telles que la liste des utilisateurs ont été également transcrites en variables globales C. Les événements pour passer d'un état vers un autre ont été déduits du comportement du module Estelle, c'est-à-dire de la machine à états. Pour simplifier, ces événements sont de deux types :

- Arrivée d'un événement X, suite à une action sur une fenêtre, tel qu'une action sur un bouton sensible : la fonction associée modifie l'état des variables globales internes, émet des primitives vers le serveur X pour modifier l'état des fenêtres, et, en général, envoie une primitive de service vers le service d'appartenance à la coopération.
- Réception d'une primitive de service depuis la connexion vers le service d'appartenance : la fonction associée au traitement modifie l'état courant du système, change l'état des fenêtres en envoyant des messages vers le serveur X, et éventuellement renvoie une primitive de service réponse.

Le processus d'appartenance réalise donc deux connexions, l'une vers le service gérant la coopération, l'autre vers le serveur X gérant les fenêtres de l'application. Puis, il réalise une boucle infinie qui effectue une attente bloquante sur les deux ports de connexion. Dès qu'un événement survient sur l'un des deux ports, il exécute de façon atomique le traitement associé.

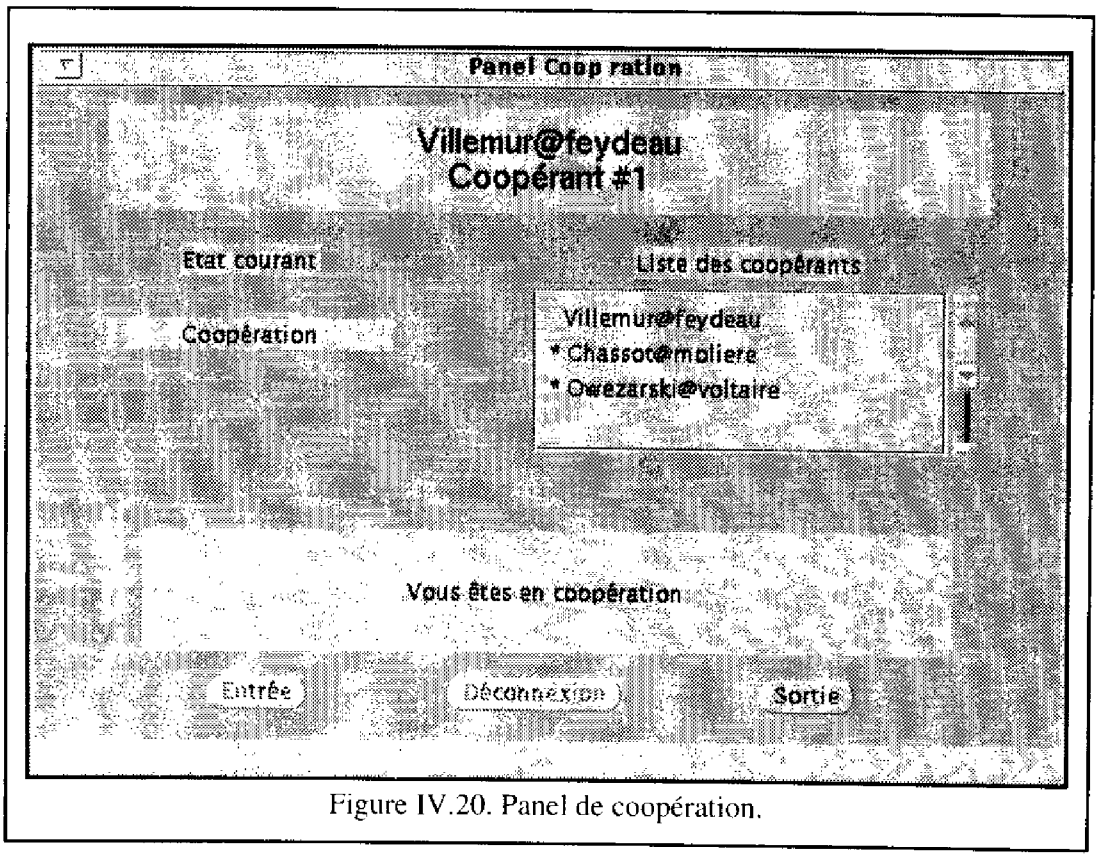


Figure IV.20. Panel de coopération.

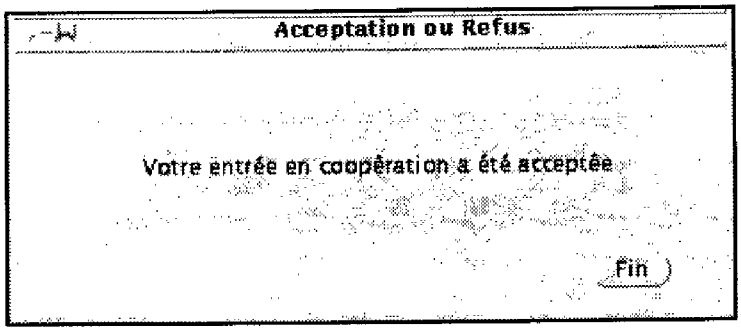


Figure IV.21. Fenêtre d'information.

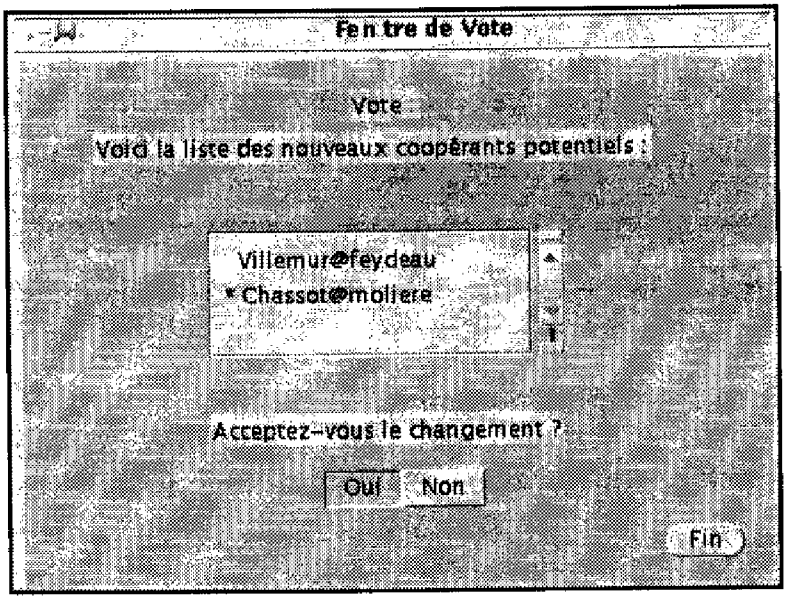


Figure IV.22. Fenêtre de vote.

Processus utilisateur pour les données

Le codage du processus utilisateur qui gère spécifiquement les données dépend bien entendu du type de données échangées. Ce processus est connecté au service de communication pour savoir quand réaliser les phases d'échange de contexte et de mise en cohérence des données. Lors de la phase d'échange des contextes initiaux, le service informe ce processus de la liste des coopérants courants. Le processus se charge alors de faire parvenir son propre contexte aux coopérants avec lesquels il est en relation, puis il informe le service lorsque cette opération est terminée. De la même façon, lors d'une demande de mise en cohérence des données par le service, ce processus se charge de la mise en oeuvre de la cohérence et de son déroulement. Dès qu'il a terminé, il informe le service.

2.3.4. Répartition sur plusieurs machines

Les processus définis peuvent être répartis sur une ou plusieurs machines. Par contre, les deux processus qui représentent un utilisateur et leur processus protocole associé doivent absolument se trouver sur une même machine. Le gestionnaire de la coopération peut se trouver sur une même machine qu'un utilisateur, ou bien peut être placé sur une machine dédiée. A titre d'exemple, la figure IV.23 donne une répartition possible d'un système comprenant trois utilisateurs.

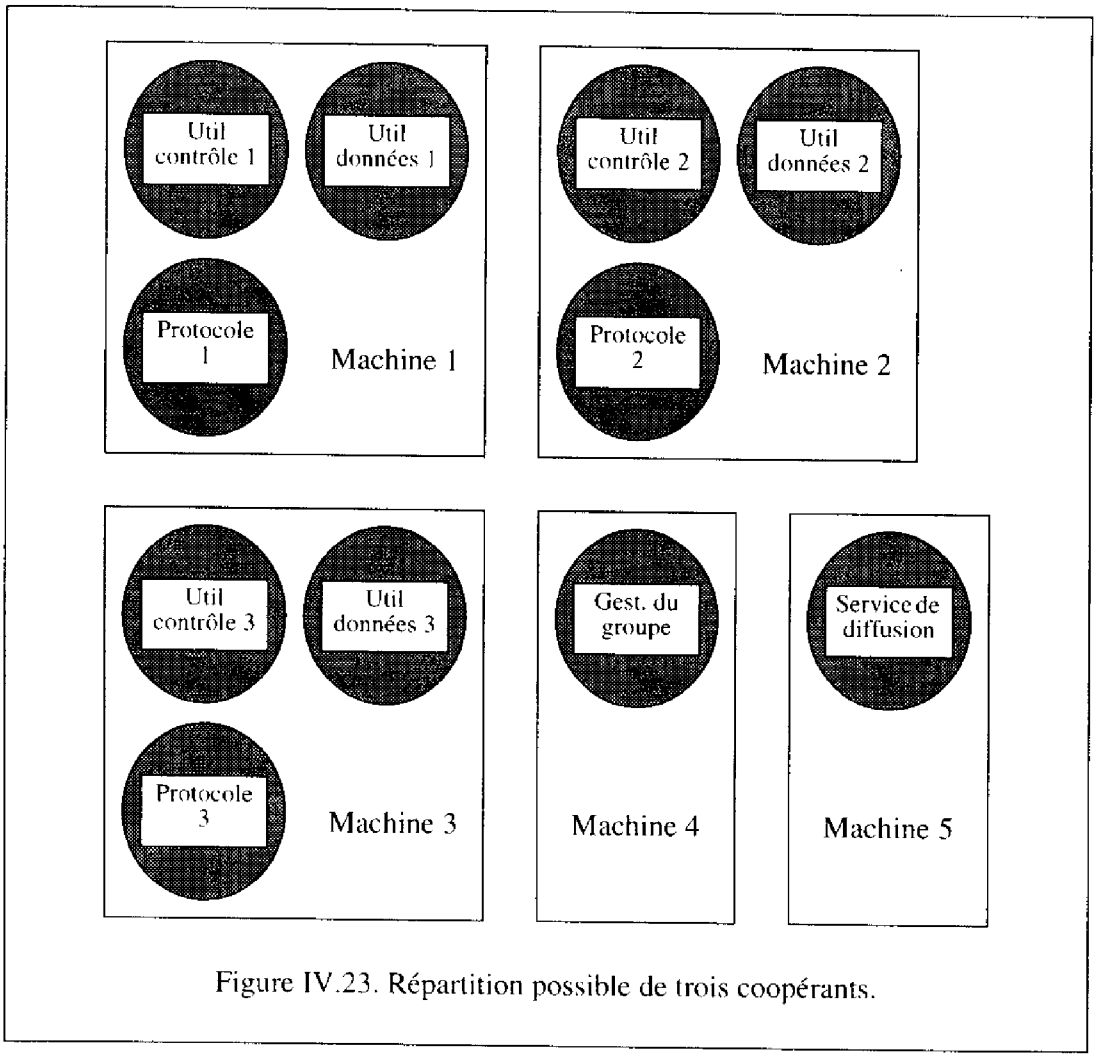


Figure IV.23. Répartition possible de trois coopérants.

2.4. Exemple d'exécution

Pour tester l'exécution de l'implantation, une configuration composée de trois coopérants a été choisie. L'organisation de ces coopérants est donnée par la figure IV.24. Les graphes instantanés valides doivent contenir au moins deux coopérants. Le système de vote choisi est le vote majoritaire.

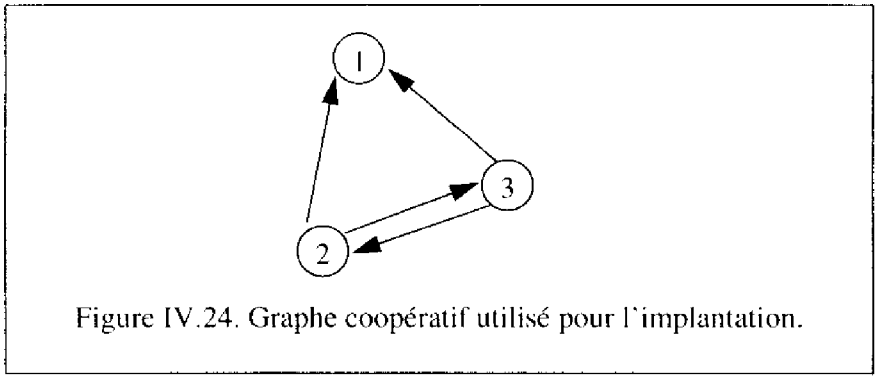


Figure IV.24. Graphe coopératif utilisé pour l'implantation.

Dans cet exemple, pour illustrer le déroulement de la coopération, les données manipulées sont constituées par une fenêtre texte, gérée par le protocole X. Chaque processus contrôlant les données se charge, au travers du protocole X, de l'affichage distant de la fenêtre texte vers l'ensemble des coopérants avec lesquels il est en relation. La phase d'échange de contexte initial consiste dans ce cas à réaliser cet affichage distant vers les autres coopérants. La phase de cohérence des données a été choisie plus simple : l'utilisateur se voit interdire de taper de nouvelle donnée.

L'exemple choisi sert essentiellement à montrer l'utilisation de la partie contrôle de la coopération. Du fait du découplage entre le contrôle de la coopération et l'acheminement des données, le processus gérant les données peut se voir remplacé sans toucher à la partie contrôle.

Actuellement, la configuration est choisie de façon statique, et est compilée dans les processus C. Cette approche étant un peu lourde, une extension est prévue pour stocker les configurations valides dans un fichier, pour pouvoir les changer dynamiquement, sans recompiler le système.

La configuration décrite nécessite environ 6800 lignes de code Estelle et C.

Conclusion

L'utilisation d'Estelle à des fins de conception, puis d'implantation de services et de protocoles de communication se révèle très prometteuse. En effet, l'adaptation du système spécifié dans un environnement réel de communication permet de réutiliser sans modification des parties de code provenant de la spécification initiale. Cette possibilité est très intéressante pour éviter nombre d'erreurs dues à la modification, à la réécriture, ou à la transcription de ce code.

Estelle est basé sur un formalisme de machines à états. Or, le faible niveau sémantique de ce formalisme le rend peu approprié à la vérification, à cause du grand nombre d'états nécessaires pour décrire un système. La démarche proposée pour pallier à ce problème a été de rehausser le niveau de spécification en Estelle*, puis en VAL. Cette transformation fournit des niveaux d'abstraction supérieurs à ceux de la spécification Estelle initiale et plus propices à l'analyse.

La vérification s'appuyant sur l'analyse de graphes d'accessibilité nécessite en effet de coder le système initial dans un langage de haut niveau sémantique comme VAL, possédant des primitives de communication très puissantes comme les réceptions asynchrones multiples, les diffusions asynchrones, des communications avec des synchronisations fortes comme les rendez-vous simples ou multiples. L'utilisation de ces primitives à l'intérieur de transitions compactes qui réalisent en une seule étape des réceptions multiples avec le traitement des messages reçus, des rendez-vous multiples et des diffusions permet de spécifier des entités les plus compactes possibles, ayant chacune peu d'états. La combinaison des états de ces entités avec le comportement d'autres entités fait que le nombre d'états globaux du système se trouve le plus réduit possible. Le graphe d'accessibilité peut rester ainsi de taille raisonnable pour permettre sa manipulation, son stockage et son analyse automatique.

Finalement, la démarche suivie pour la vérification et pour l'implantation semble fructueuse et intéressante à appliquer à d'autres services et protocoles. Une extension possible de ces travaux serait de proposer une méthodologie ou un outil qui permette de passer de spécifications Estelle en spécifications VAL, ou vice-versa. La vérification formelle se ferait alors dans l'environnement VAL, qui est le mieux adapté pour cela. L'implantation pourrait partir d'Estelle, un outil faisant le lien entre les deux environnements.

Chapitre V

Services et protocoles pour les apartés et pour les dépendances de données

Introduction

Les extensions du modèle de coopération, notamment les apartés et le lien entre les données et les communications, ont également servi de base pour la conception en Estelle de deux autres services et protocoles.

Les apartés sont des sous-groupes très dynamiques et temporaires. Ils sont formés entre deux agents coopératifs ou plus pour effectuer des courtes interactions privées et pour procéder à de brefs échanges de messages. La structure d'un aparté suit celle de la configuration courante de la coopération. Le service proposé va donc permettre aux coopérants de créer des apartés, d'y échanger des données, puis de quitter ces apartés. Le protocole associé au service va également gérer les interactions entre l'évolution dynamique de la coopération et les apartés.

Les données manipulées à l'intérieur d'un groupe coopératif peuvent dépendre les unes des autres à travers la structuration du groupe. Un agent coopératif peut manipuler des données dont la valeur dépend d'autres données locales ou exportées par d'autres agents. Ces données sont ainsi séparées en deux catégories : les variables libres ne dépendent d'aucune autre donnée et leurs valeurs sont données directement par les coopérants. Les variables liées dépendent fonctionnellement d'autres variables locales ou exportées par des agents. Les liens de dépendance entre les variables libres et les variables liées forment des graphes acycliques qui fournissent des indications sur la structure du graphe de communication sous-jacent utilisé pour les changements de valeur. Dans un cas particulier de dépendances simples, c'est-à-dire lorsque chaque variable liée ne dépend que d'une seule autre variable libre ou liée, un service a été proposé pour créer, détruire et modifier des dépendances. Le graphe de communication retenu cherche à être le plus compact possible, de manière à privilégier les changements de valeur sur les changements de structure de coopération.

Le plan de ce chapitre est donc le suivant :

La section 1 présente le service de gestion des apartés, avec le protocole spécifié en Estelle. La section 2 décrit le service de gestion des dépendances simples de données, donne son cadre d'utilisation, et présente le protocole associé, écrit en Estelle, qui a été réalisé.

1. Service et protocole pour des apartés

Le service et le protocole proposés dans cette section permettent de créer et de gérer les apartés en relation avec l'évolution dynamique du groupe coopératif [DIAZ94b], [DIAZ94c]. A l'intérieur des domaines de la coopération, des sous-ensembles d'agents peuvent former des apartés pour échanger de l'information privée entre eux. Les apartés créés sont également structurés et cette structuration respecte celle de la coopération. Les agents ne faisant pas partie d'un aparté ne peuvent pas connaître l'information qui y est échangée.

1.1. Service pour les apartés

Le service proposé permet de créer de nouveaux apartés à l'intérieur des groupes coopératifs, d'échanger des données dans ces apartés en suivant la structure de l'aparté, et de clore les apartés lorsqu'ils deviennent inutiles.

Les apartés sont toujours formés entre des agents faisant partie d'un même domaine car ils permettent des interactions brèves entre certains agents réalisant une même tâche. Chaque aparté fait donc partie et est rattaché à un domaine de coopération. Pour simplifier par la suite, on considère que la coopération ne comprend qu'un seul domaine et on identifie la coopération à ce domaine. Cette restriction peut facilement être levée pour une coopération multi-domaines, si chaque domaine est pris comme une entité autonome et indépendante.

1.1.1. Rôle du service

A un instant donné, nous supposons que des agents sont en coopération et sont organisés suivant un graphe instantané valide. Un aparté est défini par l'ensemble des agents coopérants qu'il contient et par la structure du graphe coopératif entre ces agents. Un agent coopératif décide de former un nouvel aparté : il donne au service la liste des agents qui vont appartenir à ce nouvel aparté. Le service vérifie si ce nouvel aparté n'a pas été créé par un autre agent à cet instant et refuse la création s'il existe déjà. Par conséquent, le service interdit la formation de deux apartés identiques au même moment.

Lorsque tous les agents de la liste sont informés qu'ils font partie d'un nouvel aparté, les échanges de données à l'intérieur de cet aparté peuvent commencer. Bien entendu, ces échanges doivent respecter la structure de l'aparté, structure qui dépend de l'actuelle coopération, comme vu précédemment.

Pendant l'évolution de la coopération, les membres peuvent décider de quitter un aparté. Ils indiquent au service leur intention de partir et le service enregistre ces demandes. Lorsque tous les membres veulent quitter l'aparté, le service indique la fin de l'aparté et ses membres le quittent effectivement.

1.1.2. Fonctionnement du service

Deux comportements différents peuvent être distingués dans ce service. Le premier offre la possibilité de créer de nouveaux apartés, le second permet de participer à un aparté, d'y échanger de l'information et de le quitter.

Service pour la création d'un aparté

Deux états pour chaque agent servent à la création de nouveaux apartés :

- IDLE : L'agent ne veut pas créer de nouvel aparté.
- WAIT_FOR_RESP : L'agent a fait une requête pour créer un nouvel aparté et attend la réponse pour savoir si sa création a été acceptée.

Service pour être membre d'un aparté

Le comportement d'un agent dans un aparté est le même que celui dans un autre aparté. Dans ce sens, ce comportement est générique. Par conséquent, le comportement d'un agent est donné pour un seul aparté dans ce qui suit. Un agent peut être dans trois états différents pour un même aparté :

- IDLE : L'agent n'est pas dans l'aparté.
- COMMUNICATE : L'agent a été accepté dans le nouvel aparté créé et il peut échanger de l'information avec les autres membres de cet aparté.
- WAIT_FOR_QUIT : L'agent a fait une requête pour quitter l'aparté et attend que tous les autres membres soient d'accord pour quitter cet aparté.

Un agent est membre d'un aparté spécifique lorsqu'il se trouve dans les états COMMUNICATE ou WAIT_FOR_QUIT. Le fait que le comportement d'un aparté soit le même pour tous les apartés ne signifie pas pour autant que cet agent soit dans le même état en même temps pour tous les apartés. Par exemple, un agent peut communiquer dans un aparté alors qu'il veut en quitter un autre. Cela signifie seulement qu'il possède les mêmes capacités et possibilités pour chaque aparté.

Fonctionnement

Le service attend qu'un agent génère un nouvel aparté (cet agent passe dans l'état WAIT_FOR_RESP). Lorsqu'il reçoit la requête, le service vérifie que cet aparté n'existe pas déjà et retourne la réponse à l'agent demandeur. Nous supposons que cet aparté n'est pas déjà créé. Le service prévient tous les membres du nouvel aparté (les membres passent dans l'état COMMUNICATE). Puis, les membres peuvent échanger des informations entre eux dans l'aparté. Le service attend les demandes pour quitter l'aparté. Lorsque tous les agents ont demandé à quitter l'aparté (tous les membres sont dans l'état WAIT_FOR_QUIT), le service notifie à tous les membres la fin de l'aparté (les membres passent dans l'état IDLE). Comme le créateur de l'aparté en est également membre, il reçoit aussi l'indication de fin d'aparté.

1.1.3. Echange de données

Jusqu'à présent, les échanges de données dans les apartés n'ont pas été pris en compte. Chaque membre est propriétaire d'un ensemble de données dans un aparté. Dès qu'il modifie une de ses propres données, il doit envoyer la nouvelle valeur à tous les membres de l'aparté avec lesquels il est en relation, c'est-à-dire les agents de l'aparté qui le précèdent dans le graphe. Ceci nécessite aussi des mécanismes de diffusion pour échanger les nouvelles données entre membres.

Lorsque tous les membres veulent quitter l'aparté, le service doit laisser les données échangées dans l'aparté dans un état cohérent. Ceci implique que tous les agents doivent voir les mêmes valeurs de données et qu'aucune donnée de l'aparté ne doit être en transit dans le réseau. Pour cela, le service ordonne aux membres de l'aparté de suspendre toutes les transmissions de données. Il attend jusqu'à ce que toutes les données en transit soient arrivées à destination. Puis, il indique aux membres que l'aparté est terminé.

Lorsque le problème de la cohérence des données est ajouté à l'aparté, trois phases sont nécessaires durant l'existence d'un aparté. Ces trois phases sont données par la figure V.1.

- La première phase correspond à la création du nouvel aparté. Elle contient les requêtes de création et de réception par tous les membres de l'indication de création du nouvel aparté.
- La seconde phase est la phase active de l'aparté et correspond aux échanges de données entre les membres.
- La troisième phase commence lorsque tous les membres veulent quitter l'aparté. Le service suspend les échanges de données et signale la fin de l'aparté.

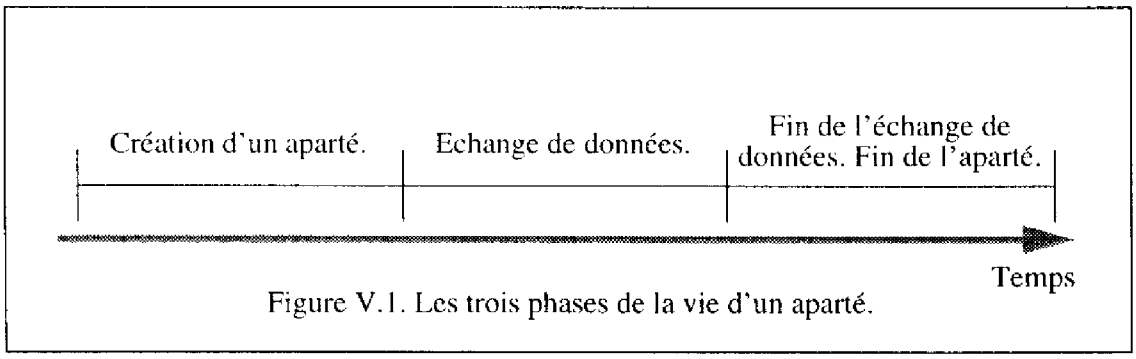


Figure V.1. Les trois phases de la vie d'un aparté.

1.1.4. Primitives de service

Ce paragraphe présente les différentes primitives de service nécessaires à la réalisation du service précédemment décrit. Ces primitives ont été regroupées par unités fonctionnelles dans les figures V.2 et V.3.

Primitives utilisées pour la création

La figure V.2 contient les primitives utilisées pour la création des nouveaux apartés, CREATE_PCS_REQ servant à émettre la requête, CREATE_PCS_CONF donnant la réponse.

Figure V.2. Primitives de service.

Nom et paramètres des primitives.	↘ Utilisateur vers Service ↗ Service vers Utilisateur	Rôle de la primitive.
CREATE_PCS_REQ (domain: domain_type; list: list_site_type)	↘	Requête pour la création d'un nouvel aparté. «list» représente la liste des membres du nouvel aparté.
CREATE_PCS_CONF (domain: domain_type; status : boolean; pcs: pcs_type; list: list_site_type)	↗	Signal indiquant si la requête a été acceptée, suivant la valeur de la variable «status». Dans la négative, la création a été refusée car l'aparté est déjà créé par un autre agent. «pcs» est l'identificur du nouvel aparté créé. «list» représente la liste des membres de l'aparté.

Primitives utilisées par les membres de l'aparté

La figure V.3 contient les primitives utilisées par les membres de l'aparté.

La primitive PCS_IND de la première unité fonctionnelle est reçue par tous les membres d'un nouvel aparté. Elle les informe de sa création et de leur appartenance.

Les primitives DATA_REQ et DATA_IND servent au transport des données entre les membres d'un aparté, DATA_REQ servant à l'envoi, DATA_IND étant l'indication de réception d'un message.

La primitive NO_NEW_DATA_IND de la troisième unité fonctionnelle indique aux membres d'un aparté de mettre leurs données dans un état cohérent et de cesser toute transmission d'information liée à cet aparté. Ceci intervient avant la fin d'un aparté.

Les primitives QUIT_PCS_REQ et QUIT_PCS_IND servent à quitter un aparté, QUIT_PCS_REQ étant la requête émise par chaque agent membre de l'aparté, QUIT_PCS_IND étant reçue par les membres lors de la fin de l'aparté, après que tous ses membres ont demandé à sortir.

Figure V.3. Primitives de service.

Nom et paramètres des primitives.	↘ Utilisateur vers Service. ↗ Service vers Utilisateur.	Rôle de la primitive.
PCS_IND (domain: domain_type; pcs: pcs_type; pure_receiver: boolean; list: list_site_type)	↗	Signal indiquant que le nouvel aparté identifié par «pcs» a été créé et que cet agent en fait partie. «list» représente la liste des membres de l'aparté. «pure_receiver» indique si cet agent envoie des données vers d'autres membres.
DATA_REQ (domain: domain_type; pcs: pcs_type; data: data_type)	↘	Cette primitive sert à envoyer une nouvelle valeur vers d'autres membres de l'aparté.
DATA_IND (domain: domain_type; pcs: pcs_type; data: data_type; sender: site_type)	↗	Un agent reçoit une nouvelle valeur de la part de l'agent «sender».
NO_NEW_DATA_IND (domain: domain_type; pcs: pcs_type)	↗	Cette primitive donne l'ordre aux membres de l'aparté de ne plus envoyer de nouvelle donnée.
QUIT_PCS_REQ (domain: domain_type; pcs: pcs_type)	↘	Un membre demande à quitter l'aparté.
QUIT_PCS_IND (domain: domain_type; pcs: pcs_type)	↗	Cette primitive signale la fin de l'aparté.

1.2. Evolution du groupe coopératif et des apartés

Plusieurs problèmes apparaissent lorsque l'évolution de la coopération dans le temps est considérée avec la formation et la destruction des apartés. Dans cette section, les différents problèmes rencontrés sont présentés et les solutions adoptées pour les résoudre sont données. La formation des apartés doit être suspendue avant tout changement de coopération. Les apartés formés sont synchronisés avec le changement de structure coopérative pendant la phase de restructuration. Puis à la fin du changement, la création d'apartés est à nouveau autorisée. Cela met en place une exclusion entre la formation de nouveaux apartés et le changement de structure de coopération.

1.2.1. Problème lors d'un changement de coopération

Précédemment, nous avons vu que lorsqu'un changement de coopération est possible, les agents font un choix pour accepter ou refuser le changement de coopération. Supposons qu'un agent crée un aparté pendant une phase de changement de coopération. De plus, supposons que le changement soit accepté. L'agent a créé un nouvel aparté lorsque les agents étaient organisés suivant l'ancien graphe de coopération. Il peut donc se produire que des agents aient quitté la coopération et que ces agents fassent partie des nouveaux apartés demandés. Clairement, cette situation n'est pas acceptable. Dès qu'un agent quitte la coopération, il ne peut pas être dans un aparté.

La solution retenue pour résoudre ce problème est d'interdire la création de nouveaux apartés pendant la phase de changement de coopération et d'autoriser les créations lorsque le changement est terminé. Ceci nécessite l'addition d'une primitive au service de création, présentée sur la figure V.4. La primitive CONF_MODIF_IND interdit à l'ensemble des utilisateurs en coopération de créer de nouvel aparté, puis elle les informe du nouveau groupe coopératif lorsqu'elle autorise de nouveau les créations.

Figure V.4. Primitives de service.

Nom et paramètres des primitives.	↘ Utilisateur vers Service ↗ Service vers Utilisateur.	Rôle de la primitive.
CONF_MODIF_IND (domain: domain_type; status: boolean; new_group: list_site_type)	↗	Cette primitive interdit ou autorise la création de nouveaux apartés lorsqu'un changement de coopération est possible, suivant la valeur de la variable «status». «new_group» est la liste des nouveaux agents coopérants.

1.2.2. Problème des apartés créés avant le changement

Un autre problème apparaît avec les apartés créés avant le changement de coopération. Lorsqu'un agent quitte la coopération et appartient à un aparté, le sous-graphe représentant l'aparté peut être déconnecté. Soit un agent qui désire quitter la coopération et qui appartient aussi à un aparté. Nous supposons également que cet agent n'est pas dans le nouveau graphe potentiel de coopération. De plus, si le changement de coopération est accepté, l'agent quitte donc la coopération, puis doit quitter l'aparté dont il fait partie. Mais, en quittant l'aparté, l'agent peut déconnecter cet aparté, comme présenté sur la figure V.5, où l'agent «A3» quitte le groupe coopératif.

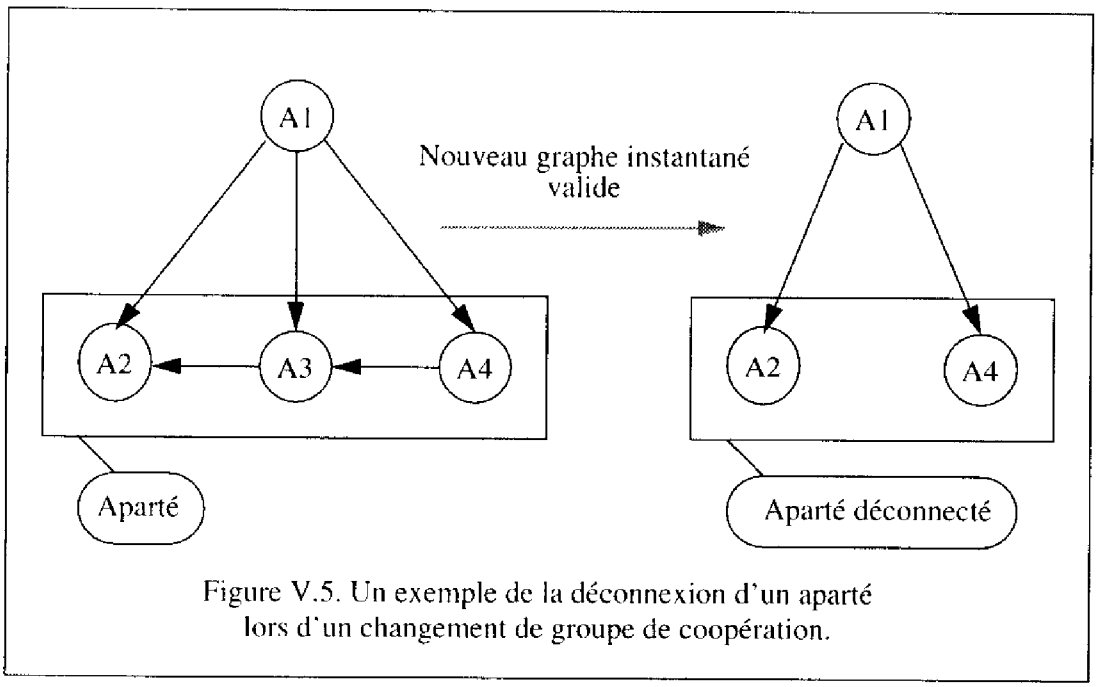


Figure V.5. Un exemple de la déconnexion d'un aparté lors d'un changement de groupe de coopération.

Bien entendu, la possibilité de déconnecter un aparté n'est pas acceptable. Ce problème semble difficile à résoudre dans le cas général. Dans notre service, nous avons choisi d'attendre la fin de tous les apartés qui peuvent être déconnectés lors du changement de coopération. Le service qui gère les changements de coopération doit également contrôler les créations et les terminaisons des apartés. Avant tout changement de coopération, le service doit attendre la fin des apartés problématiques, qui pourraient être déconnectés. Cette solution paraît raisonnable si l'on suppose que les agents qui veulent quitter la coopération ont fini la tâche qui leur a été assignée et que, par conséquent, ils n'ont plus besoin de rester dans les apartés. Le service pourrait aussi choisir une autre politique, par exemple clore les apartés arbitrairement après un certain délai, ceci afin d'éviter que la phase de changement de coopération n'attende trop longuement la fin des apartés.

1.2.3. Phases d'évolution pour un changement de coopération

La figure V.6 représente les phases nécessaires pour changer de structure coopérative. Lorsqu'un changement est possible, c'est-à-dire lorsqu'un nouveau graphe instantané valide peut être obtenu, le service qui gère le changement de coopération doit suivre quatre phases :

- La première phase interdit de créer de nouveaux apartés. Le service envoie une primitive d'interdiction vers tous les agents en coopération.
- Lorsque tous les agents coopérants ne peuvent plus créer de nouveaux apartés, le service qui gère les changements de coopération regarde s'il existe des apartés pouvant être déconnectés par le changement. Ces apartés possèdent un agent qui peut quitter la coopération. Dans l'affirmative, le service attend la fin de ces apartés problématiques.
- La troisième est celle du changement de structure coopérative. Cette phase est détaillée dans la présentation du service d'appartenance dynamique [DIAZ93d], [DIAZ93e].
- Après le changement de structure, le service qui gère les changements donne de nouveau le droit aux coopérants de former de nouveaux apartés.

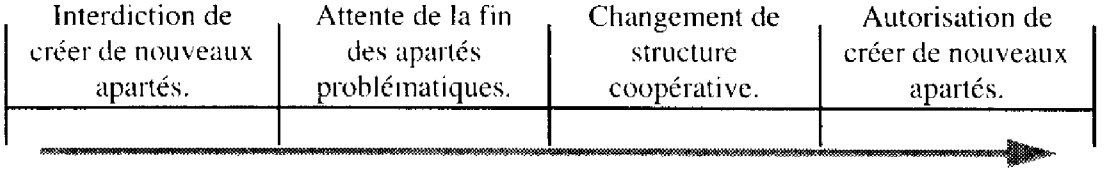


Figure V.6. Les quatre phases pour un changement de coopération.

1.3. Description formelle du protocole

Un protocole a été associé au service précédent. L'ensemble du système a été décrit et mis au point en Estelle avec le simulateur et le «debugger» Estelle [EDB92] de l'environnement de programmation «Estelle Development Toolset» (EDT).

1.3.1. Description de la spécification

L'architecture de la spécification, présentée sur la figure V.7, suit le modèle de référence en couches OSI. Les utilisateurs sont connectés au service de création des apartés au moyen des files Estelle qui représentent les SAPs de ce service. Les communications entre le service et les utilisateurs se font en utilisant les primitives de service définies dans les sections 1.1.4 et 1.2.1. Le service d'appartenance dynamique à la coopération et de formation des apartés est composé d'un ensemble de modules «Protocole i», chacun connecté à un utilisateur, d'un module gestionnaire de coopération, et d'un module gestionnaire de l'ensemble des apartés de la coopération. Les communications entre les entités qui fournissent le service d'appartenance dynamique et le service de gestion des apartés, c'est-à-dire les échanges de PDUs, sont faites en utilisant le service de diffusion sous-jacent.

Le protocole proposé est également un protocole semi-centralisé. Toutes les demandes d'entrée et de sortie de la coopération sont centralisées et contrôlées par le gestionnaire de coopération. Les requêtes pour les créations et les destructions dynamiques des apartés sont également centralisées vers un module gestionnaire des apartés. Les gestionnaires communiquent entre eux pour se synchroniser lors des modifications de la structure de coopération. Par contre, les échanges de données sont distribués en respectant la structure de la coopération. Cette conception permet de séparer les rôles. Le gestionnaire de coopération s'occupe de l'évolution dynamique de la coopération, le gestionnaire des apartés dirige la formation des apartés, tandis que chaque agent coopérant gère les données qui lui appartiennent. Les coopérants sont ainsi responsables des données dont ils sont propriétaires, qu'ils communiquent au sein de la coopération et au sein des apartés dont ils font partie.

Les modules utilisateur représentent les agents du groupe coopératif. A l'intérieur de chaque utilisateur se trouvent plusieurs sous-modules. Les sous-modules «Comportement en coopération» donnent les possibilités des utilisateurs pour la coopération, les opérations qu'ils peuvent réaliser en coopération et les données qu'ils peuvent y échanger. Les modules «Comportement pour la création d'apartés» permettent la création de nouveaux apartés. Les sous-modules «Comportement dans un aparté» sont instanciés dynamiquement lorsqu'un utilisateur fait partie d'un nouvel aparté. Ils donnent les possibilités de chaque utilisateur pour un aparté. En particulier, ils décrivent les opérations qu'un utilisateur peut réaliser pour chaque aparté, ainsi que les données qu'il peut y transmettre.

Les modules «Protocole i» fournissent aux utilisateurs le service pour faire partie du groupe coopératif et pour créer et faire partie des apartés. Pour cela, ils sont composés de plusieurs sous-modules. Le sous-module «Protocole de coopération» s'occupe des transmissions des requêtes pour participer ou pour quitter le travail coopératif, des échanges de données pour le travail coopératif et des changements de structure de coopération. Le sous-module «Protocole de création d'apartés» fournit le service pour créer de nouveaux apartés. Il manipule les requêtes et les réponses de demande de création. Bien entendu, les créations peuvent être interdites lors d'un changement de coopération puis autorisées de nouveau à la fin de ce changement. Les sous-modules «Protocole d'un aparté» sont instanciés dynamiquement pour chaque membre d'un aparté. Ile fournissent le service pour qu'un utilisateur fasse partie d'un aparté. Ces sous-modules permettent aux utilisateurs d'échanger des données à l'intérieur des apartés et de transmettre les requêtes pour quitter les apartés.

Le module de gestion de la coopération a été défini pour gérer l'évolution de la coopération dans le temps. En particulier, il reçoit les requêtes pour rejoindre ou pour quitter le travail coopératif. Lorsqu'un changement de coopération est possible, il organise les élections. Il envoie le résultat aux agents en coopération. Il contrôle les changements de structure coopérative après que la création d'apartés a été interdite et après que les apartés qui peuvent être déconnectés sont terminés. A l'intérieur du gestionnaire de coopération, un sous-module «Consistance du groupe» communique avec un moniteur. Le moniteur indique l'état courant des agents du groupe coopératif. Le sous-module «Consistance du groupe» contient les conditions de validité des graphes instantanés, conditions qui sont dépendantes de l'application coopérative réalisée. Les règles contenues permettent de vérifier s'il est possible de former de nouveaux graphes instantanés valides. Ce sous-module envoie alors le résultat au moniteur. La vérification de validité des graphes instantanés a été séparée du moniteur pour augmenter la modularité de la spécification et pour pouvoir changer facilement les règles caractérisant les graphes instantanés valides.

Le module gestionnaire des apartés contrôle et supervise l'évolution de tous les apartés. En particulier, il enregistre l'état courant des apartés créés, i.e. les membres qu'ils contiennent. Le sous-module «Superviseur» interdit ou autorise la création de nouveaux apartés et permet d'éviter la création de deux apartés identiques. Lorsqu'un changement de coopération est possible, il reçoit un signal provenant du gestionnaire de coopération qui interdit la création de nouveaux apartés. Il vérifie alors si des apartés peuvent être déconnectés par le changement de structure coopérative et attend la fin de ces apartés. Puis, il signale au gestionnaire de coopération que le changement peut être effectué. Après la mise en place de la nouvelle structure de coopération, il autorise à nouveau la création d'apartés. Chaque sous-module gestionnaire d'aparté est instancié dynamiquement lorsque l'aparté correspondant est créé ; il contrôle alors l'évolution de cet aparté. Lorsqu'un nouvel aparté est formé, le gestionnaire d'aparté correspondant avertit les membres de cet aparté, enregistre les demandes pour quitter cet aparté, et donne l'ordre de mettre les données dans un état cohérent à la fin de cet aparté. En fait, le sous-module «Superviseur» contrôle tous les apartés pour les synchroniser avec l'évolution de la coopération, alors que chaque gestionnaire d'aparté n'est concerné que par l'évolution d'un seul aparté.

Le service de diffusion est défini dans la spécification par un module Estelle et possède des propriétés similaires à celui utilisé pour l'appartenance dynamique à la coopération. Il doit toujours respecter un ordre FIFO entre les paires connectées, c'est-à-dire qu'un message transmis par une entité ne doit pas dépasser les messages précédents émis par cette même entité. Un service de diffusion fiable est nécessaire pour transmettre les PDUs gérant l'évolution de la coopération et ceux de la formation d'apartés, qui ne doivent absolument pas être perdus par le réseau. Les échanges de données n'ont pas besoin d'être fiables en permanence : seuls ceux de l'échange des contextes initiaux pour la coopération et ceux de la mise en cohérence pour la coopération et pour les apartés doivent être réalisés sans perte. Le protocole d'appartenance dynamique à la coopération et celui de la formation des apartés ne nécessitent donc pas un service fiable de façon continue pour les échanges de données.

La figure V.7 donne l'architecture de la spécification formelle. Les sous-modules en trait gras sont instanciés dynamiquement. Ils permettent l'existence dynamique de l'aparté «A» entre les utilisateurs «1» et «2». Les sous-modules en trait fin sont des sous-modules statiques. Ils existent tant que la coopération existe.

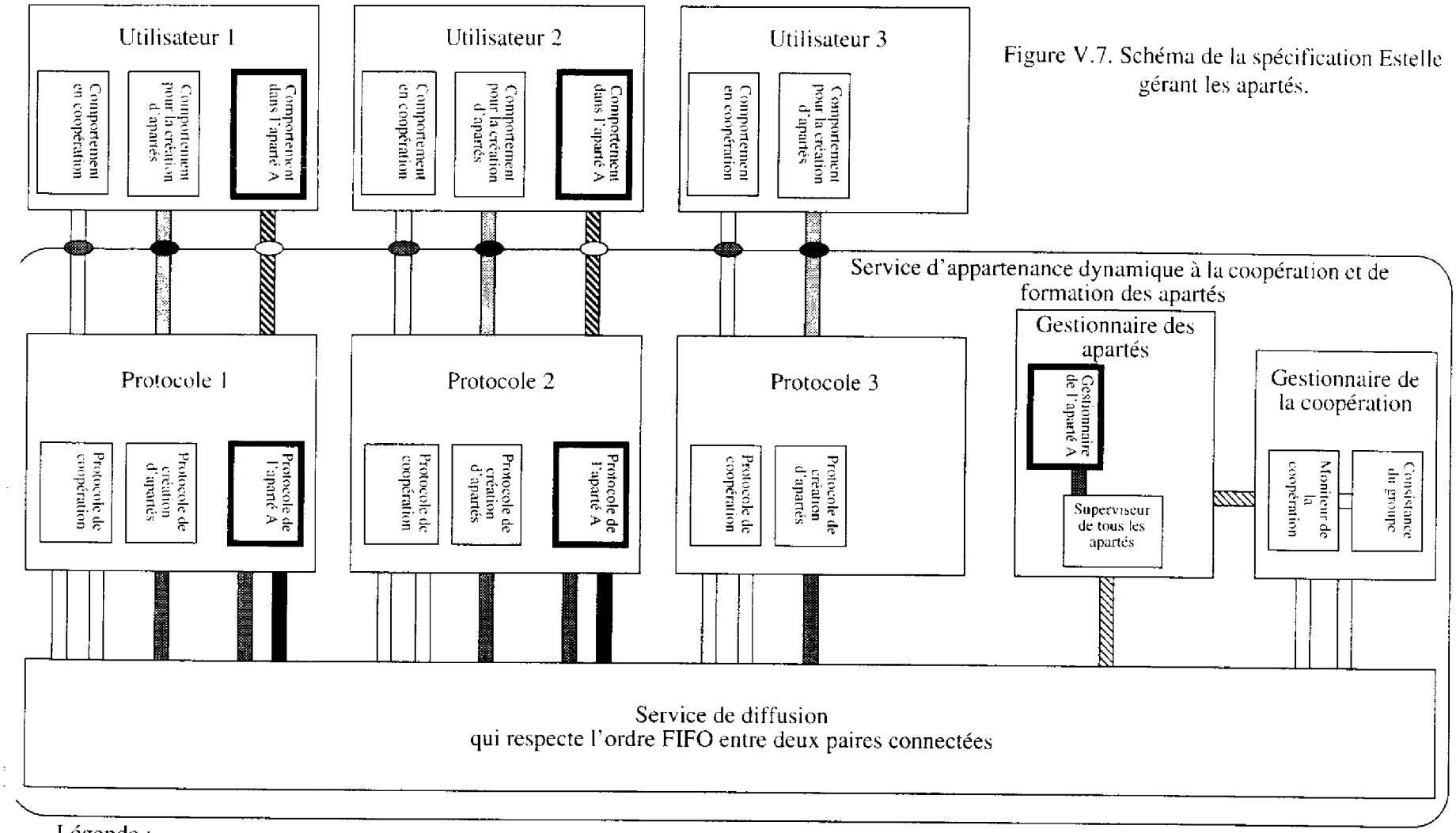


Figure V.7. Schéma de la spécification Estelle gérant les apartés.

Légende :

- Canal pour les informations de contrôle des apartés.
- Canal pour les transmissions des données des apartés.
- Canal interface entre un utilisateur et le service pour créer des apartés.
- Canal interface entre un utilisateur et le service pour faire partie d'un aparté.
- Canal pour les informations de contrôle de tous les apartés.
- Canal pour la coopération.
- Point d'Accès au Service pour créer des apartés.
- Point d'Accès au Service pour faire partie d'un aparté.
- Point d'Accès au Service pour participer à la coopération.

1.3.2. Exemple d'application

Les simulations et les tests du protocole ont été réalisés en prenant plusieurs configurations. A titre d'exemple, une configuration choisie est celle de la figure V.8, dont le graphe coopératif est formé de trois agents. Les graphes valides sont composés par au moins deux agents. Le vote utilisé pour changer de configuration est le vote majoritaire.

Chaque agent a la possibilité de créer un aparté. En fait, comme la coopération choisie est composée de trois agents, au plus trois apartés peuvent exister simultanément. La description de cet exemple nécessite environ 3300 lignes de code Estelle.

Sur la figure V.8, les agents «1» et «2» sont dans un aparté. Cette configuration courante correspond exactement à la situation de la figure V.7.

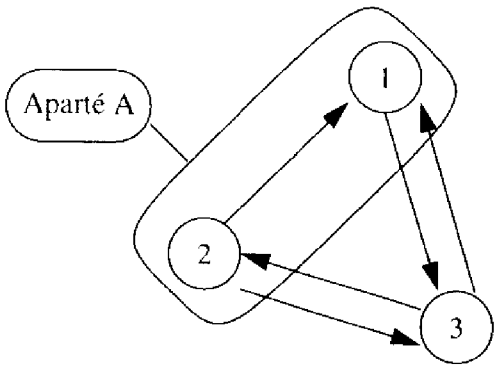


Figure V.8. Exemple de groupe coopératif.

2. Service et protocole de dépendances de données

Les services précédents n'ont pas pris en compte les échanges des données manipulées par les agents. Cette partie présente donc la conception d'un service et d'un protocole permettant de gérer des dépendances entre données. Chaque agent est propriétaire d'un ensemble de données dont les valeurs sont connues par d'autres agents suivant les relations définies entre les membres du groupe coopératif. Des dépendances entre données peuvent être créées, c'est-à-dire que certaines valeurs sont fonction des valeurs d'autres données dont un agent a connaissance.

2.1. Problématique

Le service proposé fonctionne dans le cas de dépendances simples de données. Une *dépendance simple* est définie lorsqu'une donnée liée a sa valeur qui ne dépend que d'une seule autre donnée, détenue par un autre agent.

Lorsqu'un agent a connaissance d'une valeur d'une donnée «x», qui peut être locale ou exportée par un autre agent, il peut créer une nouvelle donnée «y» liée à la valeur de la première donnée «x». Cet agent est propriétaire de la nouvelle donnée «y» créée et il peut par conséquent décider d'exporter cette nouvelle donnée vers d'autres agents avec lesquels il est en relation dans les divers domaines ou apartés dont il fait partie.

Pour illustrer cette possibilité, l'exemple de la figure V.9 est choisi : à un instant donné, cinq agents sont en coopération et sont organisés en deux domaines, DOM1 et DOM2. L'agent A2 possède une donnée «x» qu'il exporte vers les agents du domaine DOM1. Comme le domaine DOM1 est un domaine total de coopération, l'ensemble de ses agents a connaissance de «x» : les agents A1 et A3 connaissent donc la valeur de «x». L'agent A3 décide alors de créer une nouvelle donnée liée «y», à partir de la valeur de «x». La relation de dépendance simple est donnée par la formule :

$$y = fa3\,(x)$$

De plus, A3 exporte la donnée liée «y» dont il est propriétaire dans le domaine DOM2. Comme DOM2 est un domaine hiérarchique dont A3 est la racine, les agents A4 et A5 ont connaissance de la valeur de «y». Sur la figure V.9, les données dont les agents sont propriétaires sont placées dans les sommets du graphe de coopération, les données exportées étant mises sur les flèches du graphe.

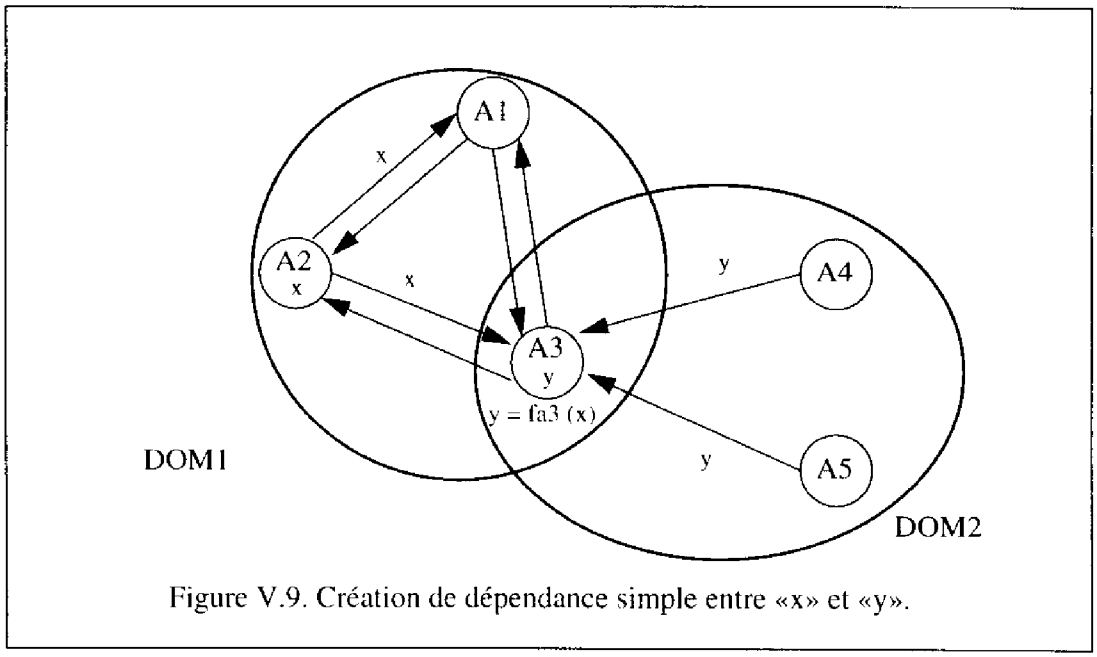


Figure V.9. Création de dépendance simple entre «x» et «y».

Un exemple concret d'application des dépendances simples est donné un peu plus loin dans la section 2.3 qui décrit une situation de télé-enseignement.

Remarques :

Dans le cas général, le domaine vers lequel est exportée la nouvelle variable liée créée peut être le même que celui de la donnée initiale ayant servi à la dépendance simple. Ainsi, dans l'exemple de la figure V.9, A3 aurait pu décider d'exporter «y» dans le domaine DOM1. Dans ce cas, «y» aurait été connue des agents A1 et A2 de DOM1.

La création de dépendances peut aussi s'appliquer pour des apartés qui peuvent être considérés comme des domaines dynamiques particuliers. Par la suite, afin de simplifier les explications, on identifie les domaines et les apartés. Tous deux sont appelés domaines.

2.2. Service pour des dépendances simples

Le service présenté va donc permettre de gérer des dépendances simples entre données dans le cadre de groupes coopératifs structurés et comportant plusieurs domaines.

2.2.1. Description du service

Ce service comporte quatre fonctionnalités.

La première variable à l'origine d'une série de dépendances est une variable libre. Le service permet donc à l'agent propriétaire de modifier la valeur de cette variable et répercute automatiquement cette modification sur l'ensemble des variables liées dépendant directement ou indirectement de la variable libre initiale.

Le service permet également aux agents coopérants de créer de nouvelles variables liées dépendant d'une autre donnée libre ou liée connue. Pour cela, les agents doivent donner au service la fonction donnant la relation de dépendance entre les deux données.

La troisième possibilité est de modifier une relation de dépendance, c'est-à-dire de changer dynamiquement la fonction de dépendance entre deux données. Le service répercute alors cette modification sur les autres variables liées qui dépendent directement ou indirectement de la variable calculée à partir de la relation de dépendance modifiée.

Finalement, le service permet de supprimer des relations de dépendance. Dans le cas d'un dépendance simple, la suppression d'une dépendance revient à dire que la variable liée devient constante. En effet, dans la relation «y = f (x)», la variable liée «y» est égale à une fonction d'arité zéro, donc constante, lorsque l'on supprime la variable «x». Par conséquent, «y» ne peut plus apporter de nouvelle information aux agents qui ont connaissance de cette donnée exportée. Or, «y» peut également intervenir dans d'autres relations de dépendance, qui, par récursivité, sont constantes. Le service répercute alors cette suppression de dépendance en supprimant également les dépendances de toutes les variables qui dépendent directement ou indirectement de «y».

Remarque : Une autre stratégie, lorsqu'une dépendance est supprimée, aurait été de conserver les dépendances partant de la dépendance supprimée. Cependant, la variable liée de la dépendance supprimée, pour ne pas demeurer constante, doit devenir une variable libre que l'agent propriétaire peut alors modifier. Cette possibilité n'a pas été retenue car elle nécessite de changer la sémantique de la donnée et de la transformer en variable libre. Rien n'empêche cependant un agent de créer une autre variable libre qu'il peut exporter à sa guise.

La séparation entre données libres et données liées correspond à la séparation entre les données dont les valeurs vont dépendre des utilisateurs et vont être communiquées au service, et celles qui vont être sous la responsabilité du service et dont les valeurs vont être calculées par le service et communiquées aux utilisateurs en fonction des valeurs initiales des données libres et des fonctions appliquées. Les données libres voient leurs valeurs changées par les utilisateurs. Par contrecoup, et du fait des relations de dépendances, les nouvelles valeurs des données liées sont calculées par le service à partir des valeurs des variables libres. Suivant les modifications apportées aux données libres et aux relations de dépendance caractérisant les données liées, le service va donc provoquer des changements sur l'ensemble des variables liées qu'il contrôle.

2.2.2. Formalisme de représentation

Le formalisme décrit dans cette section représente l'organisation logique d'un ensemble de données liées exportées à l'intérieur d'une coopération, en considérant les dépendances directes ou indirectes provenant d'une variable libre. En fait, cette structure forme une arborescence, appelée *arborescence de dépendance*, dont la racine est la donnée libre à l'origine de toutes les dépendances.

Le formalisme est présenté à partir de l'exemple de la figure V.10. L'agent A2 possède une donnée «x» exportée dans le domaine DOM1. Les agents A1 et A3 ont connaissance de la valeur de «x». L'agent A3 crée deux variables «y» et «z» liées à «x» par les fonctions «fa31» et «fa32» qu'il exporte dans le domaine DOM2. Les agents A4 et A5 ont donc connaissance de ces deux variables liées. L'agent A1 crée à son tour une variable «t» liée à «x» par la fonction «fa11» qu'il exporte dans le domaine DOM1. «t» appartient à A1, mais sa valeur est aussi connue par A2 et A3. Finalement, A3 décide de créer une dernière donnée liée «u», dépendant de «t» au moyen de la fonction «fa33» et de l'exporter dans DOM2.

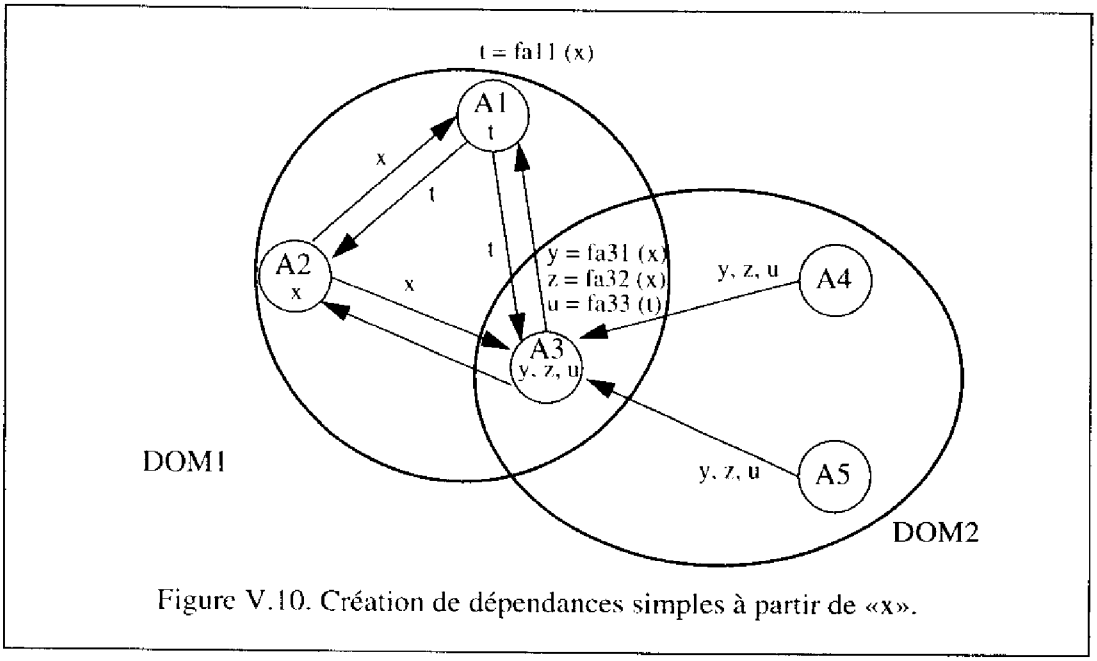


Figure V.10. Création de dépendances simples à partir de «x».

Du point de vue mathématique, ces quatre dépendances se traduisent par :

$y = fa31\ (x)$

$z = fa32\ (x)$

$t = fa11\ (x)$

$u = fa33\ (t) = fa33\ (fa11\ (x))$

L'arborescence de dépendance obtenue est celle de la figure V.11.

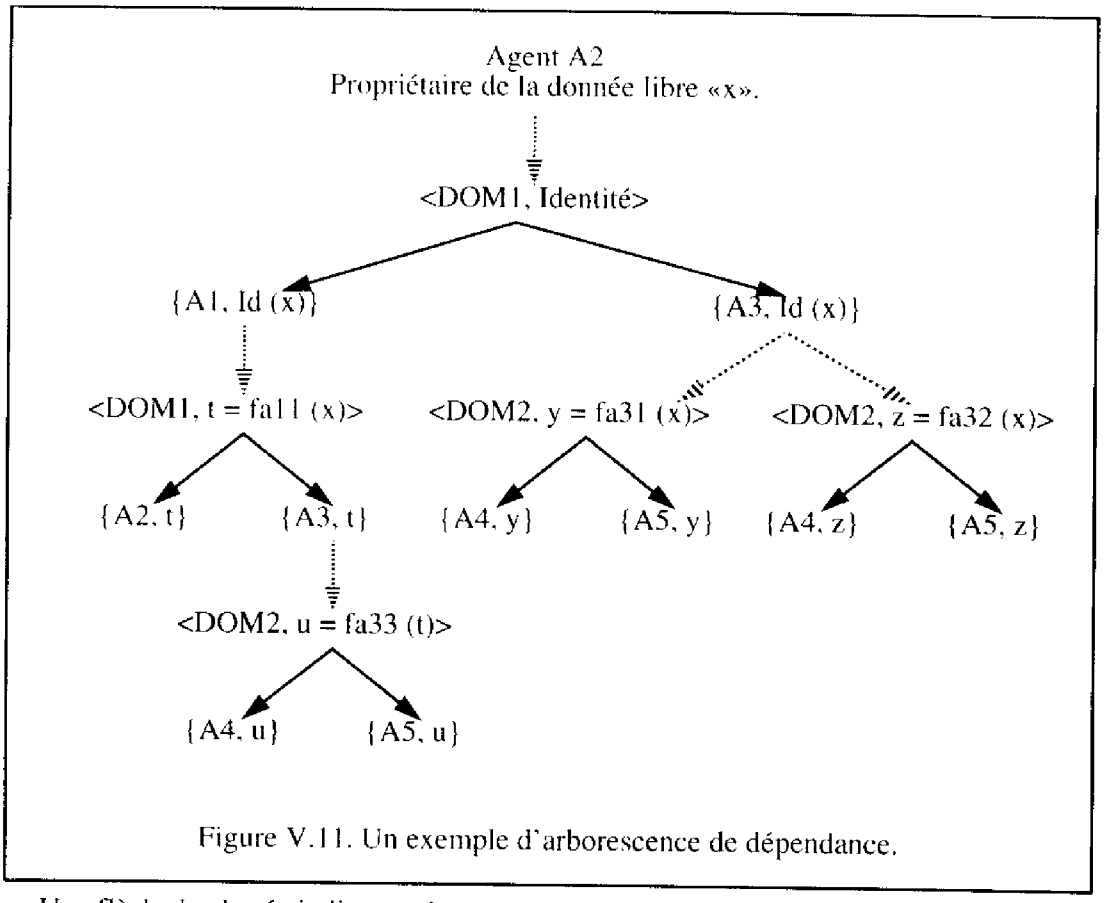


Figure V.11. Un exemple d'arborescence de dépendance.

Une flèche hachurée indique qu'un agent crée une nouvelle donnée liée à partir d'une variable qu'il connaît. La nouvelle donnée crée est exportée dans un domaine spécifique.

Un noeud du type <Domaine, Formule de dépendance> donne le domaine dans lequel la nouvelle donnée est exportée. La formule de dépendance permet de calculer sa valeur.

Les flèches noires donnent le nom des agents qui ont connaissance de la nouvelle donnée exportée, mais qui n'en sont pas propriétaires.

Un noeud du type {Agent, Nom de variable} donne l'identification de l'agent qui a connaissance de la variable exportée «Nom de variable», dont il n'est pas propriétaire.

La racine de l'arborescence est la seule donnée libre de la structure. Toutes les autres variables dépendent directement ou indirectement de sa valeur.

L'arborescence de dépendance donne les relations entre les données liées, la façon dont elles sont partagées et exportées à l'intérieur d'un groupe coopératif.

2.2.3. Lien entre le service et les modifications de l'arborescence

Cette section va montrer qu'il existe une équivalence directe entre les fonctionnalités du service proposé et un ensemble de modifications de l'arborescence de dépendance.

La création d'une nouvelle relation de dépendance revient à créer une nouvelle variable liée dont la valeur est fonction de celle d'une autre variable. Dans l'arborescence de dépendance, cette opération est équivalente à l'ajout d'une nouvelle branche avec une flèche hachurée depuis un noeud du type {Agent, Nom de variable}. La sous-arborescence ajoutée est composée d'une flèche hachurée dont l'extrémité est un noeud du type <Domaine, Formule de dépendance>. Depuis ce noeud partent un ensemble de flèches noires au bout desquelles se trouvent des noeuds du type {Agent, Nom de variable}, un pour chaque agent qui a connaissance de la nouvelle variable exportée, mais qui n'en n'est pas propriétaire. A titre d'exemple, la figure V.12 montre la création de la nouvelle variable liée «v» par le noeud A3 à partir de la variable «t» avec la fonction «fa34», «v» étant exportée dans le domaine DOM1.

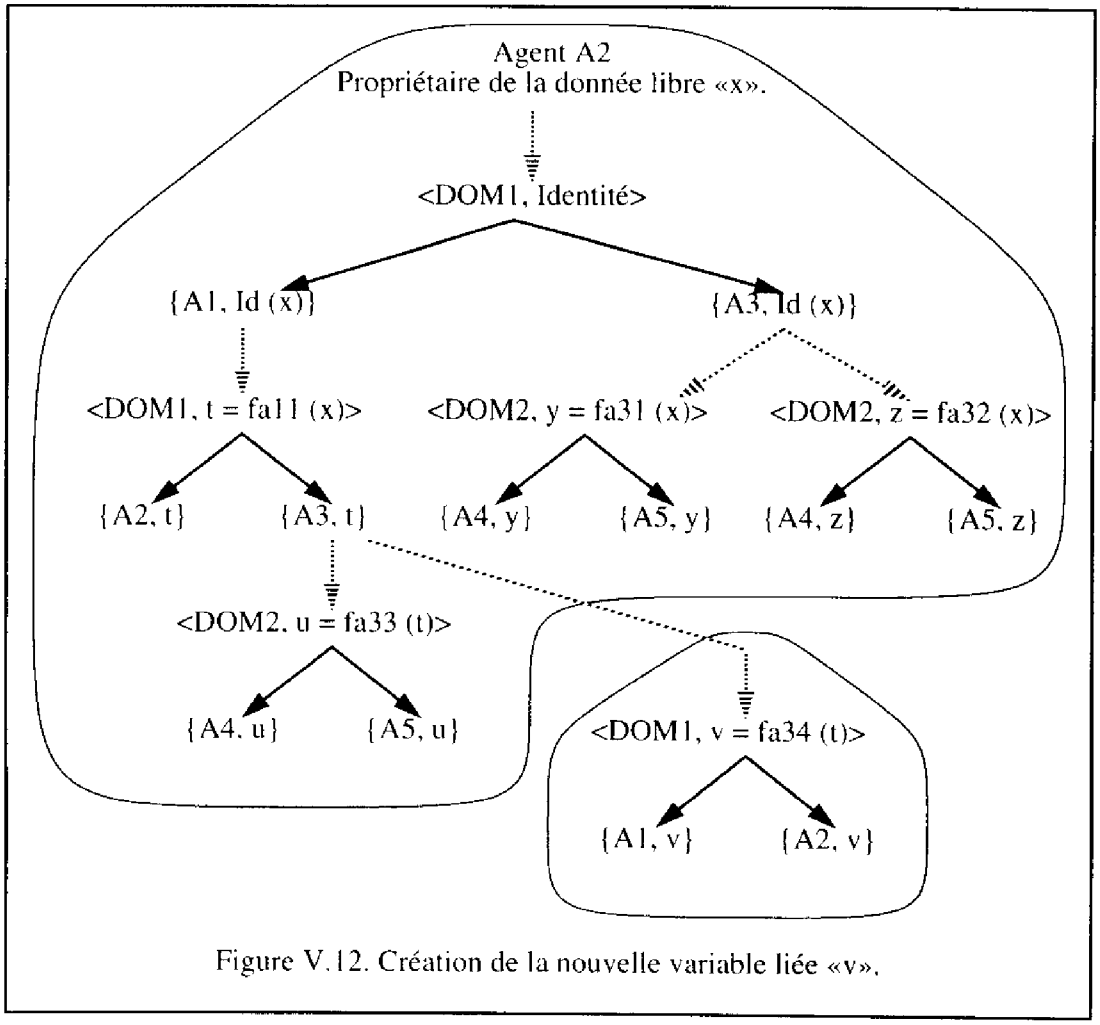


Figure V.12. Création de la nouvelle variable liée «v».

La suppression d'une dépendance de donnée correspond à la destruction de la totalité d'une sous-arborescence qui se trouve sous une flèche hachurée. Suite au fait qu'une donnée liée non dépendante devient constante, ce qui rend constantes toutes les autre variables qui dépendent de cette donnée constante, la sous-arborescence détruite part d'une flèche grise et comprend tous les noeuds intermédiaires et toutes les feuilles de l'arborescence initiale situés sous cette flèche hachurée. Contrairement à la sous-arborescence de la création, celle de la suppression peut contenir plusieurs noeuds du type <Domaine, Formule de dépendance>. Par exemple, si l'agent A1 supprime la variable liée «t», avec sa formule de dépendance, du domaine DOM1, toute la sous-arborescence sous la flèche hachurée est supprimée sur la figure V.12. La branche qui venait d'être ajoutée est également détruite. L'agent A2 peut aussi supprimer la première dépendance de «x». Dans ce cas, l'arborescence est réduite à un simple noeud, ce qui signifie qu'aucun autre agent ne peut connaître la valeur de «x», qui devient alors privée à A2.

Le changement de la variable libre racine, uniquement possible par son propriétaire, correspond à changer un attribut de la racine de l'arborescence.

L'opération de changement de la fonction d'une relation de dépendance correspond à changer l'attribut fonction d'un noeud du type <Domaine, Formule de dépendance> de l'arborescence. Par exemple, si l'agent A3 change la fonction «fa31» en «fa35» dans la relation de dépendance «y = fa31 (x)», alors le noeud <DOM2, y = fa31 (x)> devient <DOM2, y = fa35 (x)>.

Deux niveaux de modification de l'arborescence peuvent être identifiés : le premier concerne les modifications de la structure de l'arborescence de dépendance, le second concerne les changements de ses attributs. Les opérations de création et de suppression de dépendances réalisent des modifications de la structure. Par contre, les changements de valeur de la variable libre racine ou les changements de fonction modifient les attributs, mais pas la structure.

2.2.4. Principe d'utilisation du service

Chaque modification est faite en suivant les quatre étapes suivantes :

Un agent envoie vers le service une requête correspondant à l'opération qu'il veut réaliser. De l'état IDLE, il passe dans l'état C_WAIT_FOR_RESP, D_WAIT_FOR_RESP, V_WAIT_FOR_RESP, N_WAIT_FOR_RESP respectivement pour une création de dépendance, une suppression de dépendance, un changement de la valeur de la variable racine s'il en est propriétaire, ou un changement de fonction de dépendance.

Il attend la réponse. Sa modification peut être acceptée ou refusée s'il se produit une opération concurrente à la sienne.

Si la modification a été acceptée, il la réalise en informant le service des changements effectués.

L'agent attend la fin de ses changements qui peuvent produire des modifications sur les autres données liées qu'il connaît de l'arborescence.

Pour synchroniser les différentes modifications de l'arborescence qui peuvent survenir, le service peut suspendre certains agents du groupe coopératif. Les agents suspendus attendent les effets des modifications effectuées par d'autres agents du groupe coopératif. Ils ne peuvent pas réaliser de modification jusqu'à ce que le service les active de nouveau.

2.2.5. Primitives de service

Ce paragraphe donne l'ensemble des primitives de service nécessaires pour la réalisation du service de gestion des dépendances de données. Ces primitives sont regroupées par unités fonctionnelles sur la figure V.13.

Les primitives MODIFICATION_REQ et MODIFICATION_CONF de la première unité fonctionnelle servent aux requêtes pour créer, détruire, changer des dépendances de données, ou changer la valeur de la variable libre racine, MODIFICATION_REQ transmettant la requête, MODIFICATION_CONF contenant la réponse.

Les primitives SUSPEND_IND, SUSPEND_RESP et UNSUSPEND_IND de la deuxième unité fonctionnelle suspendent des agents et les empêchent de réaliser des opérations de modification des dépendances, ceci afin de les synchroniser avec des modifications concurrentes. SUSPEND_IND est l'indication de suspension, SUSPEND_RESP est la réponse lorsqu'un agent a pris en compte la demande, UNSUSPEND_IND est la primitive qui active de nouveau un agent.

Les primitives DEPEND_CREATE_REQ et DEPEND_CREATE_IND valident une nouvelle création de dépendance. DEPEND_CREATE_REQ transmet les paramètres de la nouvelle dépendance créée, DEPEND_CREATE_IND est reçue par les agents qui ont connaissance de la nouvelle valeur liée créée.

Les primitives DEPEND_DELETE_REQ, DEPEND_DELETE_IND et END_DEPEND_DELETE_IND valident la destruction d'une dépendance. DEPEND_DELETE_REQ est utilisée par l'agent qui détruit la dépendance, DEPEND_DELETE_IND informe tous les agents qui ont accès à des variables liées détruites, END_DEPEND_DELETE_IND indique à l'agent qui réalise la suppression si des variables liées auxquelles il a accès sont également détruites.

Les primitives CHANGE_VALUE_REQ, CHANGE_VALUE_IND et END_CHANGE_VALUE_IND servent à changer la valeur de la variable libre racine. L'agent propriétaire de cette variable change sa valeur avec CHANGE_VALUE_REQ. Les agents qui accèdent à des variables liées connaissent leurs nouvelles valeurs avec CHANGE_VALUE_IND. END_CHANGE_VALUE_IND signale à l'agent qui a modifié la variable libre si des variables liées qu'il connaît sont également modifiées.

Les primitives CHANGE_FUNCT_REQ, CHANGE_FUNCT_IND et END_CHANGE_FUNCT_IND permettent de modifier une relation de dépendance. CHANGE_FUNCT_REQ modifie la relation, CHANGE_FUNCT_IND indique aux agents les répercussions sur les autres variables liées. END_CHANGE_FUNCT_IND donne à l'agent qui a effectué la requête la liste des variables liées modifiées auxquelles il accède.

Figure V.13. Primitives de service.

Nom et paramètres des primitives.	↘ Utilisateur vers Service. ↗ Service vers Utilisateur.	Rôle de la primitive.
MODIFICATION_REQ (modif: modif_type; id: id_data_type; dom_dest: domain_type; func: id_function_type)	↘	Un agent demande une modification dans l'arborescence. «modif» est le type de modification requise. «id» est l'identificateur de la donnée à partir de laquelle la modification est appliquée, «dom_dest» et «func» sont les domaines et fonctions permettant de définir la nouvelle donnée liée.
MODIFICATION_CONF (modif: modif_type; status: boolean)	↗	Confirmation d'une requête de modification de l'arborescence. «status» indique si la modification est acceptée.
SUSPEND_IND	↗	Signal pour suspendre un agent.
SUSPEND_RESP	↘	Réponse à une indication de suspension.
UNSUSPEND_IND	↗	Signal pour réactiver (suppression de la suspension) un agent.
DEPEND_CREATE_REQ	↘	Notification pour créer une nouvelle donnée liée avec une relation de dépendance, et exportée.
DEPEND_CREATE_IND (id: id_data; val: data_type)	↗	Signal indiquant qu'un agent a connaissance d'une nouvelle donnée liée exportée, provenant d'une nouvelle dépendance. «id» représente l'identifiant de la nouvelle donnée, «val» sa valeur.
DEPEND_DELETE_REQ	↘	Notification pour supprimer une relation de dépendance de données.
DEPEND_DELETE_IND (list_id: list_elts_subtree_type)	↗	Cette primitive signale aux agents concernés qu'une relation de dépendance a été supprimée. «list_id» contient les identificateurs des données qui dépendent (directement ou indirectement) de la relation de dépendance supprimée.
END_DEPEND_DELETE_IND (list_id: list_elts_subtree_type; status: boolean)	↗	Cette primitive signale à l'agent qui a initialisé la modification si la suppression de dépendance a supprimé indirectement d'autres variables liées qu'il connaît.
CHANGE_VALUE_REQ (val: data_type)	↘	Requête utilisée par l'agent racine de l'arborescence pour changer la valeur de la racine.
CHANGE_VALUE_IND (list_val: list_val_data_type)	↗	Cette primitive signale que toutes les valeurs des données liées ont été changées. «list_val» contient les identificateurs des données et leurs nouvelles valeurs.
END_CHANGE_VALUE_IND (list_val: list_val_data_type; status: boolean)	↗	Cette primitive signale à l'agent racine de l'arborescence qui a initialisé la modification si la modification de la donnée libre racine a indirectement changé les valeurs des autres données dépendantes qu'il connaît.

Figure V.13. Primitives de service.

Nom et paramètres des primitives.	↘ Utilisateur vers Service. ↗ Service vers Utilisateur	Rôle de la primitive.
CHANGE_FUNCT_REQ (func: id_function_type)	↘	Requête pour changer une fonction dans une relation de dépendance.
CHANGE_FUNCT_IND (list_val: list_val_data_type)	↗	Cette primitive signale qu'un changement de fonction de dépendance a modifié (directement ou indirectement) les valeurs des données liées de la liste «list_val».
END_CHANGE_FUNCT_IND (list_val: list_val_data_type; status: boolean)	↗	Cette primitive signale à l'agent qui a réalisé le changement de fonction si sa modification de la relation de dépendance a indirectement changé les valeurs d'autres variables liées qu'il connaît.

2.3. Cadre d'utilisation du service des dépendances

En s'appuyant sur un exemple simple, ce paragraphe décrit les hypothèses d'application du service de gestion des dépendances simples.

2.3.1. Exemple

Le service proposé peut servir pour un cours de télé-enseignement. Dans cette situation, un enseignant donne un cours à un ensemble d'étudiants. Les différents membres de ce groupe ne sont pas physiquement situés au même endroit. Chacun possède une station de travail connectée à un réseau qui permet aux membres de communiquer entre eux. La structure coopérative retenue pour représenter le cours est décrite par la figure V.14. Tous les membres sont placés dans un seul domaine de coopération DOM1. Le professeur «P» a une position centrale et il a connaissance du contexte propre de chacun des élèves. Les élèves «Ei» voient tout ce que fait l'enseignant mais ils ne peuvent pas communiquer entre eux.

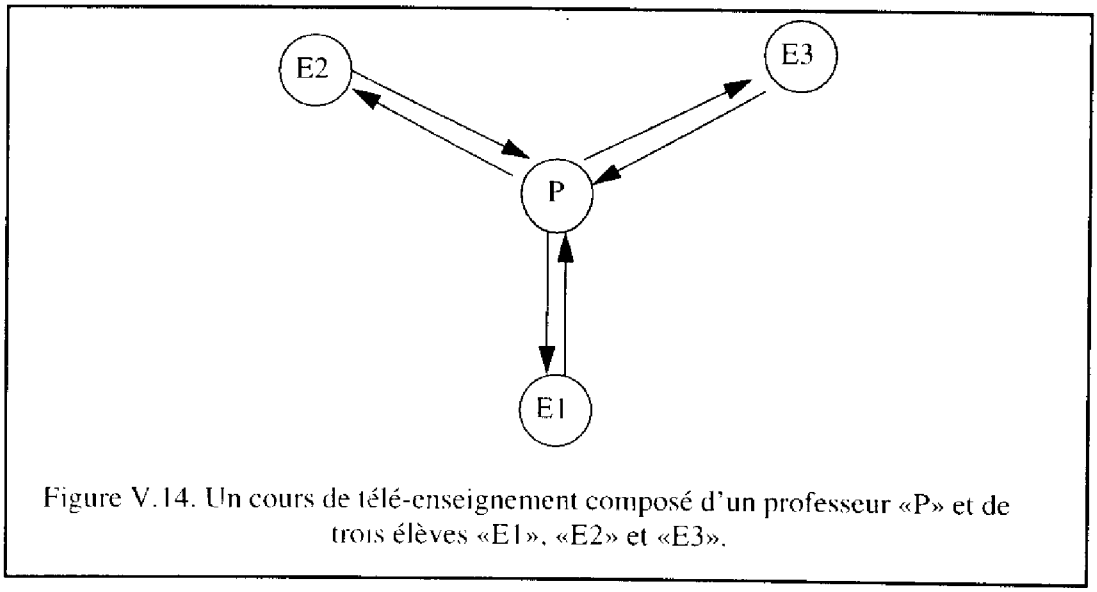


Figure V.14. Un cours de télé-enseignement composé d'un professeur «P» et de trois élèves «E1», «E2» et «E3».

L'étudiant «E1» désire poser une question «qe1» par exemple. Le professeur prend en compte l'intervention de «E1». Cette intervention peut se révéler intéressante pour l'ensemble de la classe : le professeur décide donc de la faire connaître à l'ensemble de la classe. Pour cela, il crée une donnée «qp1» liée à la question de l'élève par la fonction d'identité, qu'il porte à la connaissance de l'ensemble de la classe. Du point de vue de la question, le professeur joue le rôle de relais vers l'ensemble de la classe. La donnée liée «qp1» appartient au professeur et est connue de tous les étudiants suivant la structure de coopération. La figure V.15 donne l'arborescence de dépendance obtenue.

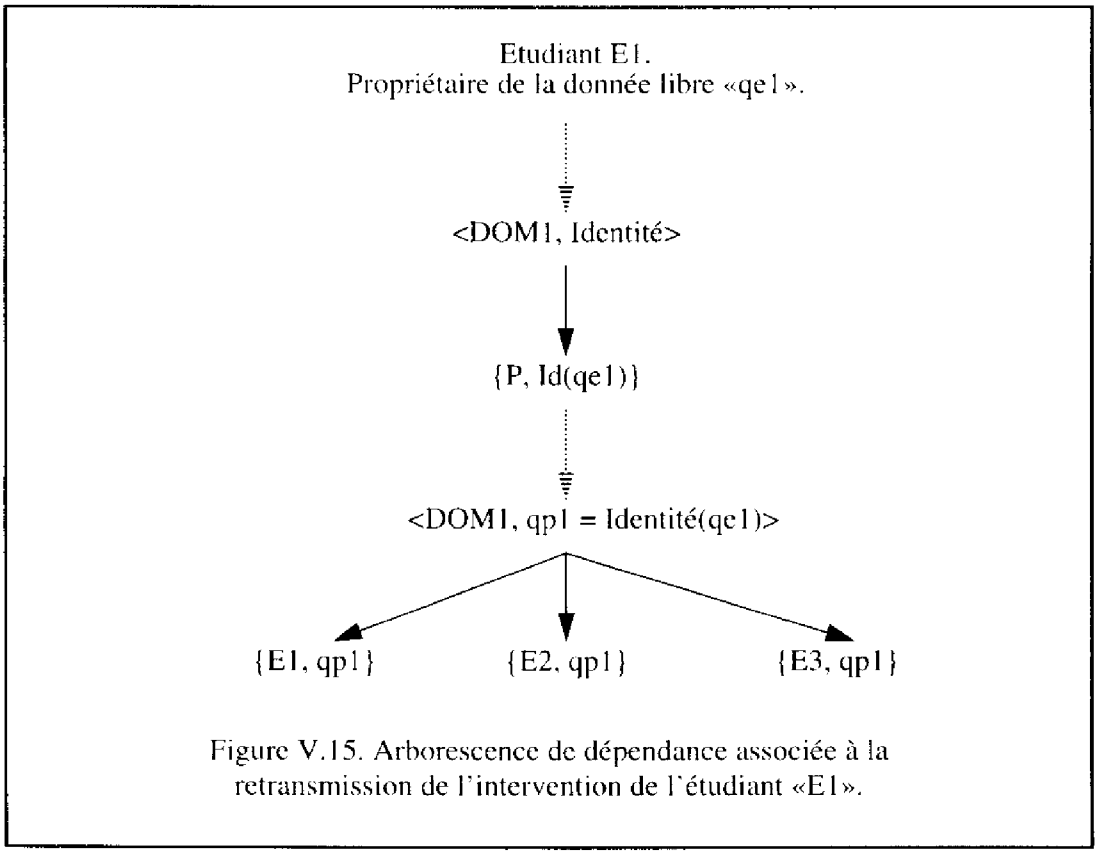
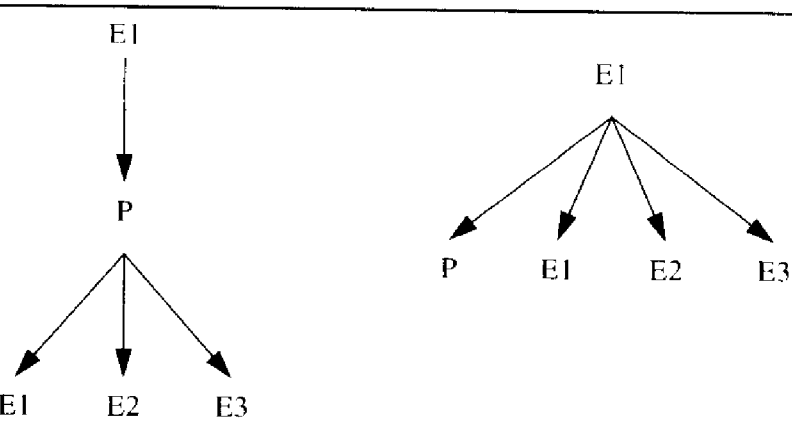


Figure V.15. Arborescence de dépendance associée à la retransmission de l'intervention de l'étudiant «E1».

Plusieurs schémas de communication sont possibles : les communications peuvent correspondre directement à l'arborescence de dépendance. Dans ce cas, le professeur stocke la question puis il la rediffuse vers les élèves (figure V.16(a)). La deuxième possibilité est que l'élève diffuse directement son intervention vers l'ensemble de la classe (figure V.16(b)). La dépendance créée par le professeur peut alors se voir comme une autorisation de transmission vers la classe. Les flèches des deux arborescences de diffusion de la figure V.16 indiquent vers qui un agent diffuse ses informations.



(a) : Arborescence de diffusion suivant celle de dépendance.

(b) : Arborescence de diffusion «directe».

Figure V.16. Arborescences de diffusion associées à la retransmission de l'intervention de l'étudiant «E1».

2.3.2. Cadre d'utilisation

Le choix qui a été fait pour le protocole correspond à la deuxième possibilité de l'exemple, c'est-à-dire la diffusion directe. L'agent racine de l'arborescence diffuse directement son intervention vers l'ensemble de tous les agents de l'arborescence. Le graphe de diffusion «direct» obtenu est donc une arborescence composée d'une racine, qui est le propriétaire de la variable libre racine, et d'un ensemble de feuilles qui sont les agents propriétaires de toutes les données liées de l'arborescence. Les flèches de l'arborescence de diffusion indiquent vers qui un agent envoie directement les nouvelles valeurs.

Cette stratégie s'applique pour une donnée libre racine dont la valeur change fréquemment ou dont le contenu est volumineux, et pour laquelle il est coûteux de faire des stockages et réémissions sur des noeuds intermédiaires. Un cas classique de ce type de donnée est la vidéo. Une image vidéo fait environ une centaine de kilo-octets et est modifiée au moins une dizaine de fois par seconde pour avoir un rendu acceptable. Ce genre de donnée est très coûteux à stocker et à réémettre : il vaut mieux la diffuser directement, surtout si l'on s'appuie sur un service de communication offrant des diffusions efficaces.

Un autre cas pour lequel la transmission directe est intéressante est lorsque l'arborescence de dépendance est très profonde et comporte beaucoup de noeuds intermédiaires. La modification des variables feuilles nécessite la traversée de nombreux noeuds. Si l'arborescence de diffusion est calquée sur celle des dépendances, il faut beaucoup d'étapes pour répercuter les modifications de la variable libre initiale. Cet inconvénient est évité avec une arborescence de diffusion réalisant une diffusion directe.

Le protocole de communication proposé favorise donc les changements de valeurs de la variable libre en répercutant au plus vite ses effets sur l'ensemble des variables liées de l'arborescence. Il s'applique lorsque la fréquence de changement de la valeur de la variable libre est élevée par rapport à celle de la modification des dépendances. Par contre, le processus sera plus lourd pour effectuer les modifications de la structure de l'arborescence ou des dépendances par rapport au changement de valeur des données.

2.4. Protocole pour la réalisation du service

Le protocole utilise l'arborescence de dépendance associée à la variable libre racine. Il maintient l'arborescence de diffusion «directe» en relation avec l'évolution dynamique de l'arborescence de dépendance. Cette partie montre comment les modifications de l'arborescence de dépendance sont réalisées en prenant en compte les demandes des agents en coopération.

2.4.1. Connaissance nécessaire à chaque entité de protocole

La connaissance minimale qu'un agent possédant une variable liée doit avoir dans l'arborescence de dépendance est la liste des agents d'un domaine vers lequel il exporte cette variable liée. Il ne sait alors rien sur le comportement des agents fils, notamment s'il créent ou non d'autres dépendances à partir de sa variable exportée. Avec cette connaissance minimale, l'agent racine doit parcourir récursivement l'arborescence de dépendance pour répercuter sur l'ensemble des données liées la modification de la valeur de la variable libre. Ce principe est utilisé dans les protocoles transactionnels [TRAV92a], [AYOU90]. Chaque noeud intermédiaire stocke la donnée reçue, l'analyse et la retransmet si nécessaire vers ses fils. Cette stratégie, pénalisante avec des volumes de données importants ou avec une arborescence profonde, peut être justifiée pour des protocoles transactionnels, qui ne nécessitent pas des temps de réponse très rapides et qui ne manipulent pas en général des données volumineuses.

La connaissance des agents a donc été étendue pour faciliter les changements rapides de valeurs. L'agent racine possède la liste de tous les agents ayant une variable liée dans l'arborescence de dépendance. De ce fait, il a connaissance de l'arborescence de diffusion «directe» et il peut transmettre directement les modifications effectuées sans parcourir récursivement l'arborescence de dépendance. Les changements de valeur sont donc plus rapides que pour le parcours récursif, surtout s'ils sont réalisés au dessus d'un service de diffusion efficace. Par contre, les changements de structure de l'arborescence de dépendance, qui correspondent aux créations et suppressions de dépendances, sont plus complexes à gérer que dans le cas d'un parcours récursif.

La connaissance des agents possédant des variables liées a également été étendue, pour pouvoir se synchroniser avec d'éventuelles modifications concurrentes. Un agent propriétaire d'une variable liée connaît la liste des agents propriétaires de l'ensemble des variables dont sa variable dépend. Dans le cas de dépendances simples, cet ensemble correspond à la liste des agents rencontrés depuis la racine de l'arborescence de dépendance jusqu'à la variable considérée. L'ensemble des noeuds rencontrés depuis la racine de l'arborescence de dépendance jusqu'à un noeud contenant une variable liée est appelé «*chemin-racine*». De plus, un agent propriétaire d'une variable liée connaît la liste des agents qui interviennent dans la sous-arborescence partant de la variable liée considérée.

2.4.2. Protocole de verrouillage à deux phases

Pour réaliser les modifications des dépendances, c'est-à-dire les modifications de la structure de l'arborescence, les agents intervenant dans l'arborescence doivent se synchroniser entre eux. Cette synchronisation a été faite au moyen du protocole de verrouillage à deux phases [TRAV92b], [ANCI90], [AYOU90]. Un agent qui veut effectuer une modification de l'arborescence de dépendance prévient l'ensemble des agents qui peuvent réaliser une modification concurrente en leur envoyant une requête de suspension. Tout agent recevant une requête de suspension cesse toute opération qui peut modifier l'arborescence et envoie une réponse à l'émetteur pour lui signaler que sa requête a bien été prise en compte et que plus aucune modification concurrente ne peut survenir de la part de cet agent. Lorsque toutes les réponses sont reçues par

l'émetteur de la requête, il effectue la modification de l'arborescence et envoie un nouveau message vers les agents concernés pour les prévenir de la modification réalisée et pour les réactiver. La figure V.17 montre le schéma de déroulement de cette procédure.

Les suspensions et les notifications de modifications sont liées à la connaissance que chaque agent a de l'arborescence de dépendance. Cette connaissance a été étendue pour que chaque entité de protocole puisse communiquer avec l'ensemble des agents concurrents au moyen d'une seule diffusion.

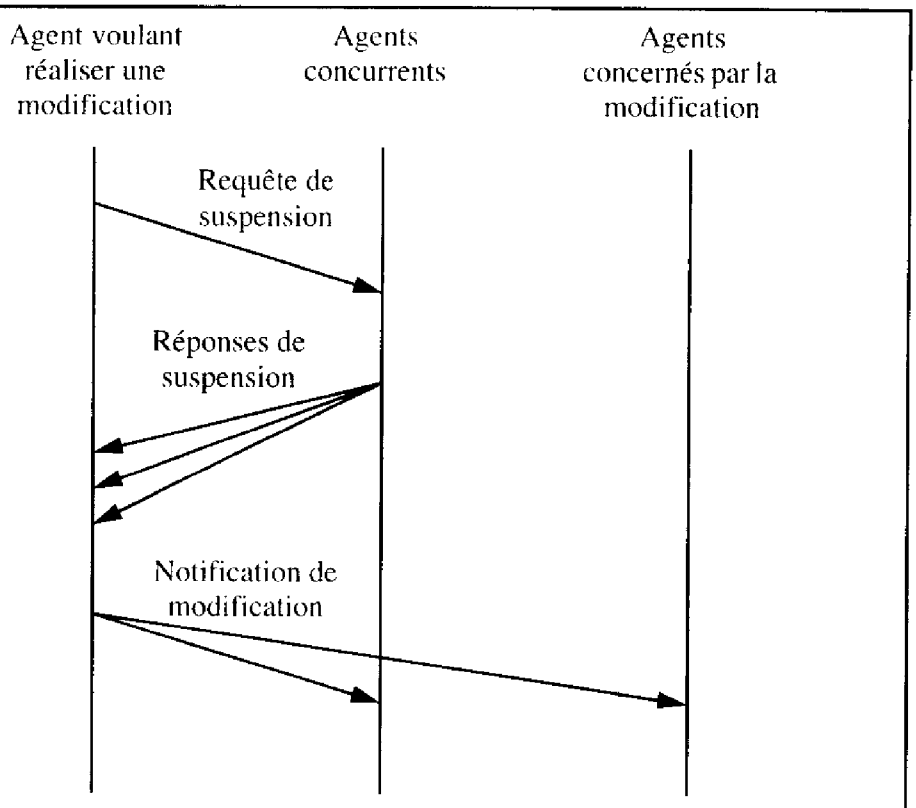


Figure V.17. Le protocole de verrouillage à deux phases utilisé pour réaliser les changements de la structure de l'arborescence de dépendance.

2.4.3. Agents atteints par chaque changement

Pour chaque modification, ce paragraphe donne la liste des agents concurrents et celle des agents concernés par le changement de structure de l'arborescence de dépendance.

Création d'une nouvelle dépendance

Un agent qui désire créer une nouvelle variable liée exportée, et donc une nouvelle dépendance, suspend tous les agents qui ont des données sur le chemin-racine. Ces agents forment l'ensemble des agents concurrents à la création d'une nouvelle dépendance. Puis, il attend leur réponse. Si la modification est acceptée, il transmet le résultat aux agents du chemin-racine et aux agents ayant connaissance de la nouvelle variable exportée, qui forment la sous-arborescence à partir de cette nouvelle variable liée. Ces derniers forment les agents concernés par la modification. Si la création est refusée, l'agent envoie un message d'annulation vers les agents du chemin-racine pour les réactiver.

Suppression de dépendance

Un agent désirant supprimer une donnée dépendante suspend tous les agents du chemin-racine et tous les agents de la sous-arborescence qui part de la donnée à détruire. L'ensemble de ces agents est l'ensemble concurrent à la destruction. Puis il attend leur réponse. Si la modification est acceptée, il envoie le résultat vers tous les noeuds suspendus. L'ensemble des agents concernés est le même que celui des agents concurrents. Si la modification est refusée, l'agent envoie un message d'annulation pour réactiver les agents suspendus.

Changement de la valeur de la variable libre racine

Cette opération n'est possible que par l'agent propriétaire de la donnée racine. Cet agent suspend alors l'ensemble des agents qui ont une donnée liée dans l'arborescence afin d'éviter que l'arborescence en cours ne soit modifiée lors de la diffusion. Il attend alors leur réponse. Puis, il envoie la nouvelle valeur vers l'ensemble des agents de l'arborescence. Les ensembles des agents concurrents et des agents concernés sont tous deux formés par les agents ayant des données liées dans l'arborescence.

Changement de fonction

Un agent désirant modifier une relation de dépendance suspend tous les agents du chemin-racine et tous les agents de la sous-arborescence qui part de la donnée concernée. L'ensemble de ces agents est l'ensemble concurrent au changement. Puis il attend leur réponse. Si la modification est acceptée, il envoie la nouvelle fonction vers les agents de la sous-arborescence et réactive également les agents suspendus du chemin-racine. L'ensemble des agents concernés comprend les agents de la sous-arborescence. Il est inclus dans l'ensemble des agents concurrents. Si la modification est refusée, l'agent envoie un message d'annulation pour réactiver les agents suspendus.

2.4.4. Résolution des conflits

Des situations de conflit peuvent apparaître lorsque plusieurs agents veulent modifier l'arborescence de dépendance en même temps et lorsque ces opérations s'appliquent sur un ou plusieurs mêmes noeuds. Plusieurs opérations peuvent avoir lieu en même temps, comme deux créations de dépendances avec deux nouvelles variables liées. Mais d'autres opérations doivent être réalisées en exclusion si elles manipulent les mêmes noeuds en même temps.

Requêtes concurrentes

Lorsqu'un agent veut réaliser une opération de modification, il envoie une requête de suspension vers l'ensemble des agents concurrents. Ces agents forment l'ensemble des agents pouvant créer un conflit avec la modification souhaitée. Lorsque l'émetteur d'une requête a reçu toutes les réponses des agents concurrents, il est sûr qu'ils sont tous suspendus et qu'ils n'effectuent plus aucune opération sur l'arborescence. Par conséquent, aucune opération ne peut être en conflit avec la requête. Mais, avant que l'émetteur de la requête n'ait reçu toutes les réponses, des agents concurrents non encore suspendus peuvent entamer des opérations conflictuelles, à cause de l'asynchronisme des communications (figure V.18). Ces opérations conflictuelles peuvent annuler ou retarder la requête de modification initiale. Pendant l'attente des réponses des agents concurrents, si l'émetteur de la requête reçoit une requête de la part d'un agent concurrent, il analyse cette requête pour détecter si elle peut avoir lieu en même temps que celle qu'il a entamée, ou si sa propre requête doit être annulée ou retardée.

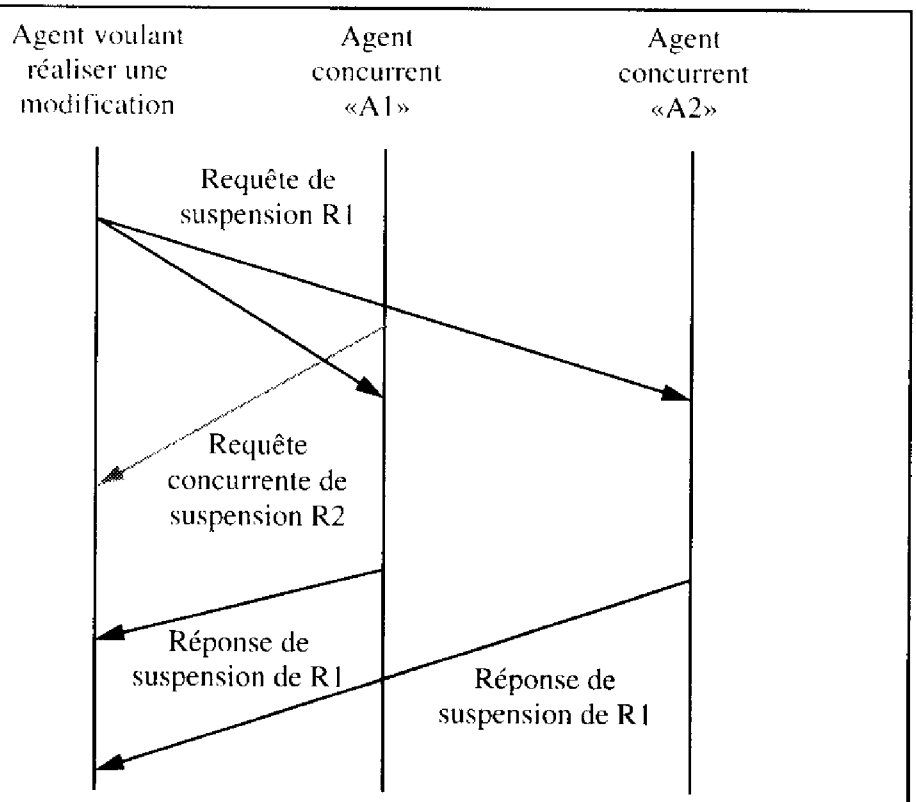


Figure V.18. L'agent «A1» a commencé la requête concurrente R2 avant de prendre en compte la requête R1 et avant d'envoyer la réponse à R1.

Deux types de conflits, présentés dans les deux paragraphes suivants, ont été rencontrés.

Conflit de destruction

Ce conflit apparaît lorsqu'un agent commence une opération de destruction de dépendance, ce qui va entraîner la destruction de tout une sous-arborescence de l'arborescence de dépendance, pendant qu'un autre noeud réalise une autre opération comme créer un nouvelle dépendance, détruire une autre sous-arborescence, ou changer une fonction à partir d'un noeud faisant partie de la sous-arborescence concernée par la destruction. La deuxième opération est inutile, car elle est annulée par la destruction du dessus. La solution retenue est donc d'annuler l'opération inutile.

L'agent qui a commencé l'opération annulée attend l'ensemble des réponses des agents concurrents. Puis, il leur envoie un message pour leur indiquer que sa requête a été annulée et pour les réactiver.

Conflit de changement de structure

Ce conflit apparaît lorsqu'un agent modifie la structure de l'arborescence de dépendance en créant ou en supprimant des dépendances, pendant qu'un autre agent modifie les attributs de l'arborescence, soit en changeant la valeur de la donnée racine, soit en changeant une fonction de dépendance. Supposons qu'un agent veuille créer une nouvelle donnée liée exportée et qu'en même temps, l'agent propriétaire de la donnée racine décide de modifier la valeur de sa donnée. L'agent racine ne peut pas envoyer directement la nouvelle valeur vers les agents qui voient la nouvelle donnée exportée, car il ne connaît pas ces agents. Ce conflit peut être plus sérieux si la

deuxième opération est une destruction. L'agent racine risque d'envoyer la nouvelle valeur vers des agents qui ne font plus partie de l'arborescence. Pour éviter ce conflit, ces deux opérations parallèles ont été synchronisées. Toute nouvelle opération de création ou de destruction est retardée jusqu'à ce que le changement de valeur de la donnée racine soit terminé.

Un problème similaire arrive lorsqu'une fonction d'une dépendance est modifiée. Si une autre opération concurrente de modification de la structure se déroule à partir d'un noeud faisant partie de la sous-arborescence située en dessous du changement de fonction, le changement de fonction risque de ne pas être répercuté vers tous les noeuds, ou bien risque d'être répercuté vers des noeuds détruits. Là aussi, ces deux opérations concurrentes doivent être synchronisées. Toute opération de création ou de destruction est retardée si un changement de fonction concurrent se produit sur un noeud de son chemin-racine.

2.5. Description formelle du protocole

Le protocole défini et le service assuré ont été décrits et mis au point en Estelle en utilisant le simulateur et le «debugger» Estelle [EDB92] de l'environnement de programmation «Estelle Development Toolset» (EDT).

2.5.1. Description de la spécification

L'architecture de la spécification, présentée sur la figure V.19, suit le modèle de référence en couches OSI. Les utilisateurs du service de gestion des dépendances sont connectés via un ensemble de SAPs, représentés par des files Estelle, à ce service. Les utilisateurs et le service dialoguent au moyen des primitives de service définies dans la section 2.2.5. Un ensemble de modules «Protocole i», chacun connecté à un utilisateur, compose le service de gestion des dépendances. Les échanges de PDUs entre les entités de protocole sont réalisés en utilisant le service de diffusion sous-jacent.

Le protocole présenté est totalement distribué. Chaque agent est responsable des données qu'il possède et des dépendances qu'il crée entre les données. La connaissance de l'arborescence de dépendance est partagée entre les diverses entités de protocole. De plus, la gestion des modifications est faite en faisant intervenir l'ensemble des entités de protocole et non en centralisant les requêtes.

Les modules «Util i» sont les utilisateurs du service, c'est-à-dire les agents du groupe coopératif. En utilisant les primitives du service de gestion des dépendances, chaque utilisateur envoie à son module protocole respectif, des requêtes pour créer de nouvelles dépendances entre données, pour détruire des dépendances, ou pour les modifier en changeant de fonction. L'agent propriétaire de la donnée libre racine peut aussi modifier la valeur de cette dernière.

Les modules «Protocole i» fournissent le service défini. Ces modules contiennent la connaissance que chaque agent a de l'arborescence de dépendance, c'est-à-dire les noeuds des chemins-racine et les sous-arborescences de chaque donnée que possède chaque agent. Chacun transmet et gère les requêtes de modification des dépendances provenant de son utilisateur respectif vers les autres modules de protocole concernés. Chaque module réalise ces modifications en utilisant le protocole de verrouillage à deux phases. Il gère aussi les requêtes provenant des autres agents et suspend ou active son utilisateur suivant ces requêtes concurrentes. Finalement, il détecte les conflits entre les opérations de son utilisateur et celles faites par les autres. L'ensemble des modules protocole synchronise également les modifications de l'arborescence de dépendance.

Le service de diffusion est représenté par un module Estelle. Il doit respecter un ordre causal entre les entités connectées car les modifications de l'arborescence de dépendance doivent être synchronisées et ordonnées, et doivent donc être perçues dans le même ordre par toutes les entités connectées. Les PDUs transmis entre les entités de protocole ne doivent pas non plus être perdus par le réseau, ce qui nécessite un service de diffusion fiable.

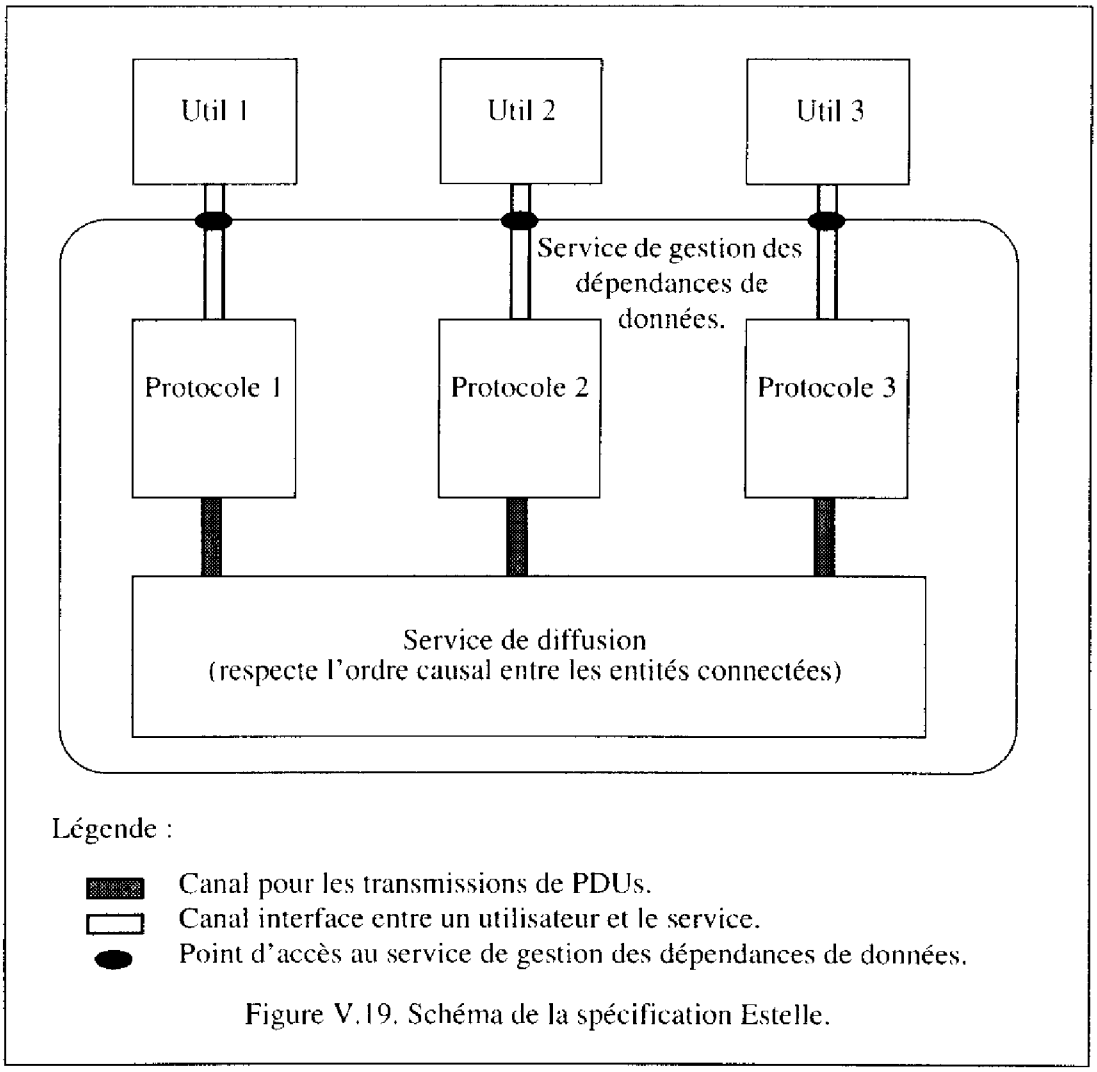


Figure V.19. Schéma de la spécification Estelle.

2.5.2. Application à une structure de coopération spécifique

Les simulations et les tests du protocole ont été réalisés en prenant plusieurs configurations. A titre d'exemple, la figure V.20 donne un graphe coopératif qui comprend trois agents organisés en deux domaines, DOM1 et DOM2. Pour simplifier la simulation, une seule donnée libre, appelée «x», est détenue par l'agent «1». La taille de cette spécification Estelle est d'environ 3700 lignes de code. Bien entendu, une spécification plus complète peut être facilement réalisée avec plusieurs données libres, car ces données sont considérées comme indépendantes et n'interfèrent pas entre elles au niveau des entités de protocole : il suffit d'instancier un module Estelle de plus chez chaque agent pour gérer une nouvelle arborescence de dépendance provenant d'une nouvelle variable libre.

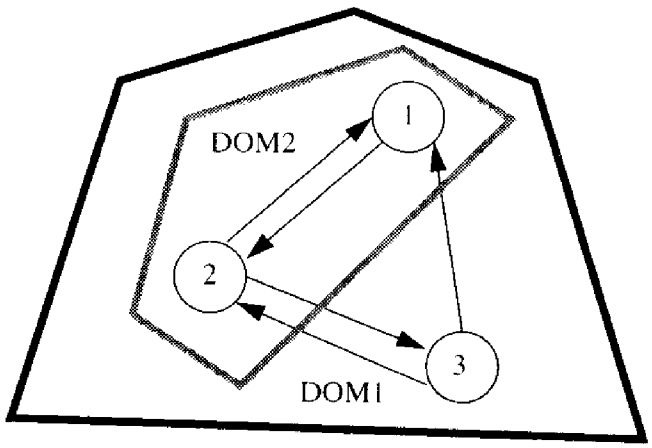


Figure V.20. Un exemple retenu pour tester la spécification.

2.6. Liens avec les services précédents. Extensions

Pour simplifier la conception, le service de gestion des dépendances a été spécifié lorsque la configuration du groupe coopératif est statique et ne change pas dans le temps. Le lien doit cependant être fait avec les services et protocoles précédents qui manipulent des groupes dynamiques, dont le nombre d'agents et la configuration du groupe coopératif évoluent au cours du temps. Cette évolution dynamique du groupe va également influer sur la configuration de l'arborescence de dépendance. Les principales modifications qu'elle entraîne sont :

Une donnée est créée par un agent et est exportée dans un domaine (ou un aparté) particulier. Lorsque l'agent quitte le domaine (ou l'aparté) considéré, la donnée dépendante disparaît, et avec elle toutes les données liées qui dépendent d'elle. Cette modification est finalement identique à celle provoquée par une suppression de dépendance.

De plus, les agents qui exportaient des données vers les agents qui quittent un domaine ne doivent plus le faire. Cela entraîne des suppressions de flèches noires et des noeuds de type {Agent, Nom de variable} dans l'arborescence de dépendance.

Lorsque des agents entrent dans un nouveau domaine, ils doivent prendre connaissance des variables exportées par les autres agents avec lesquels ils sont en relation. Cela induit des modifications dans l'arborescence de dépendance au niveau des flèches noires, et provoque des ajouts de flèches noires et des noeuds de type {Agent, Nom de variable}.

Les agents qui restent dans un domaine peuvent garder intactes leurs dépendances, s'il reste cependant des agents qui ont encore accès à leurs variables exportées. Sinon, il est inutile qu'ils les conservent, car personne d'autre que le propriétaire n'a connaissance de ces variables.

Finalement, l'adaptation de ce protocole peut être faite en ajoutant une phase de synchronisation supplémentaire. Lorsqu'une modification de la structure de la coopération peut avoir lieu, l'ensemble des opérations sur l'arborescence de dépendance sont annulées, et l'ensemble des agents utilisateurs sont suspendus. Le protocole est en quelque sorte bloqué. Puis le protocole attend la phase de reconfiguration de groupe coopératif. Une fois cette reconfiguration terminée, le protocole prend connaissance de la nouvelle structure coopérative, modifie la structure de l'arborescence de dépendance en fonction des entrées et des sorties des agents, puis il déblo-

que les utilisateurs en excluant les agents qui quittent un domaine, en signalant aux agents qui restent en coopération les données exportées qui disparaissent et dont ils n'ont plus connaissance, et en portant à la connaissance des nouveaux entrants les données exportées qu'ils voient.

Le service et le protocole proposés s'appliquent dans certains cas particuliers. Tout d'abord, le graphe de diffusion est imposé par le protocole et est choisi de façon à permettre le plus possible une diffusion directe vers tous les agents concernés. La première extension proposée est de faire négocier par chaque agent créant une nouvelle variable liée avec une nouvelle dépendance le choix de la diffusion de cette donnée liée : soit directement comme précédemment, soit au moyen d'une retransmission par l'agent. L'arborescence de diffusion est ainsi construite de façon coopérative et la décision de transmission est laissée à chacun des propriétaires des données. On peut aussi imaginer qu'un agent puisse renégocier son choix de retransmission. La dernière possibilité serait d'étendre ce service vers la gestion de dépendances multiples, qui apparaissent comme beaucoup plus complexes à manipuler.

Conclusion

Ce chapitre a présenté deux services et protocoles coopératifs qui complètent la gestion d'appartenance, l'un gérant la formation dynamique d'apartés, l'autre assurant la création de dépendances simples entre données.

La suite de ces travaux serait de procéder à la vérification des services et des protocoles proposés. La démarche adoptée pourrait s'inspirer de celle utilisée dans le chapitre IV pour la vérification du protocole d'appartenance dynamique. La spécification Estelle qui gère les apartés serait transcrite en VAL, puis les graphes d'accessibilité obtenus seraient alors analysés. Quelques exemples des vérifications à effectuer seraient de regarder que le protocole fournit bien le service attendu, de voir si la structure des apartés suit effectivement celle de la coopération. Il serait également intéressant de voir si la synchronisation entre les créations d'apartés et l'évolution de la coopération se fait effectivement correctement, en observant que tous les membres d'un aparté soient tous en coopération et que l'on n'ait pas d'aparté contenant des agents non coopérants.

A plus long terme, l'implantation de ces protocoles peut être envisagée.

L'étude des services et des protocoles pour les dépendances de données doit être poursuivie car elle se révèle incomplète. En particulier, il est nécessaire de définir des services et des protocoles plus généraux que celui présenté, notamment pour offrir plus de possibilités dans les relations entre les dépendances de données et les communications, puis à terme pour gérer des dépendances multiples. Pour de tels services et protocoles, l'utilisation d'Estelle est certainement à reconsidérer car ce langage ne fournit pas un niveau d'abstraction suffisant pour s'adapter assez facilement à leur complexité. De plus, Estelle ne se révèle pas suffisamment dynamique car il n'est pas possible de manipuler dynamiquement ou même de passer en argument des fonctions ou des comportements. VAL, dérivé de PROLOG, offre de telles possibilités qui pourraient s'avérer utiles pour la transmission directe de relations de dépendances entre agents coopérants.

Conclusion

Le travail présenté dans ce mémoire a contribué à définir et à réaliser un ensemble de services et de protocoles coopératifs génériques qui seraient de ce fait utilisables par de multiples applications coopératives. Leur conception s'est appuyée sur les approches formelles développées dans le cadre des systèmes distribués et des réseaux, ce qui a permis de montrer comment ces approches pouvaient être étendues et pouvaient s'adapter à la spécification de ces services et de ces protocoles coopératifs prévus pour des groupes d'utilisateurs.

Tout d'abord, l'analyse de la coopération et du travail coopératif a mis en évidence l'aspect pluri-disciplinaire de ce domaine, qui inclut notamment les sciences sociales, l'intelligence artificielle et les systèmes distribués. Les apports de chacune de ces disciplines sont intéressants car ils montrent des vues et des approches différentes de la coopération et du travail coopératif. Cependant, quelques nuances doivent être apportées : en effet, la fusion de ces disciplines n'est pas triviale et s'effectue souvent difficilement. Les travaux qui proviennent des sciences humaines veulent réutiliser les théories sociologiques, les théories des organisations et du travail de groupe prévues initialement pour l'étude des sociétés humaines, pour les appliquer dans un domaine plus technologique, celui du travail de groupe assisté par ordinateur. Ces études doivent donc être adaptées, transformées et modifiées pour pouvoir effectivement s'appliquer au CSCW. Jusqu'à présent, la plupart d'entre elles se contentent d'évaluer l'efficacité du travail de groupe par rapport au travail individuel, ou bien d'évaluer les performances de groupes d'utilisateurs pour des travaux spécifiques. La dimension humaine a du mal à s'adapter et à rejoindre la dimension technologique. L'intelligence artificielle distribuée étudie les méthodes de raisonnement, de planification, de contrôle pour que le travail de groupe soit mené à son terme. Mais, souvent, les travaux présentés manquent de généralité et ils ne sont appliqués que dans des cas spécifiques. Leur utilisation dans des applications coopératives générales s'avère de ce fait délicate. De plus, les réalisations ne sont souvent pas distribuées, et les agents d'un groupe s'exécutent sur une même machine. Les systèmes distribués et les réseaux interviennent pour supporter des communications entre des entités séparées physiquement. Cette partie qui est la plus technologique comprend les études des services et des protocoles qui assurent les communications entre des groupes d'utilisateurs. Ces études permettent également la conception d'architectures distribuées et de plate-formes coopératives effectivement utilisables par des groupes de coopérants. Les systèmes distribués et les réseaux traitent les échanges, les stockages et les transferts d'information entre entités. Cependant, bien que des réalisations conséquentes soient faites dans ce domaine, les systèmes proposés n'effectuent que le transport de l'information sans procéder à son analyse ou à son interprétation, ce qui est l'opposé des travaux de l'intelligence artificielle distribuée. De ce fait, bien que plus réalistes, ces systèmes sont beaucoup moins formels et bien

moins modélisés que ceux qui proviennent des sciences cognitives. Cette constatation nous a d'abord poussé à proposer un modèle pour représenter les interactions entre les membres de groupes coopératifs puis à s'appuyer sur ce modèle pour définir des services et des protocoles généraux de communication.

Le modèle défini sert à représenter un groupe coopératif structuré et repose à l'origine sur la logique modale. Il permet de décrire une coopération sous la forme d'un graphe dont les noeuds sont les entités coopératives et dont les flèches donnent les relations entre ces entités. Ce modèle, finalement assez simple, donne les liens qui définissent le partage et l'échange de l'information et de la connaissance entre les entités coopératives. Les relations entre les coopérants sont fixes et figées car elles sont liées à un type de travail coopératif. Une autre possibilité serait d'avoir des relations dynamiques. Cependant, ceci n'a pas été retenu car les relations sont définies à priori par le travail coopératif à exécuter. Modifier ces relations est équivalent à changer de travail coopératif. De plus, cette possibilité offrirait trop de niveaux dynamiques qui s'ajouteraient à ceux déjà proposés et qui, finalement, risqueraient de devenir ingérables : les entités passeraient leur temps à gérer les évolutions dynamiques possibles au détriment de l'avancée du travail coopératif. Chacune de ces relations n'est pas non plus liée à un type particulier de donnée. Un modèle plus fin serait de mettre une relation par donnée ou par type de donnée entre les coopérants. Cependant, ce choix n'a pas été fait car le modèle ne cherche pas à caractériser chacun des échanges de données, mais il veut plutôt montrer les relations globales entre les entités coopérantes : ainsi, ces relations influent sur le partage des données dont les coopérants sont propriétaires, mais elles ne dépendent pas des données. En effet, ce sont les coopérants qui sont finalement responsables des informations qu'ils connaissent et qu'ils échangent avec d'autres coopérants. La structuration du groupe influe sur la vision de la connaissance par les entités coopératives, ainsi que sur la façon dont les données sont communiquées et transmises entre ces entités. Un deuxième niveau de structuration a été proposé par l'utilisation des domaines de coopération qui servent à contenir la connaissance commune à des sous-ensembles d'agents coopératifs. En plus des relations entre agents, les domaines définissent un découpage du groupe coopératif. Bien que statique pour la définition des relations entre agents, le modèle de coopération a été étendu par les graphes instantanés valides qui donnent les configurations acceptables d'agents pendant le déroulement d'un travail coopératif. La dynamique du groupe se fait donc par la présence ou par l'absence des coopérants pour réaliser à un instant donné la tâche coopérative. Des groupes très dynamiques, ou apartés, sont également introduits pour modéliser les courtes interactions qui ne sont pas représentables par le seul modèle de coopération et par les domaines. Le modèle pour les groupes coopératifs découpés en domaines, augmenté des structures dynamiques de coopération valide et des apartés paraît suffisamment riche pour pouvoir s'adapter à des situations réelles et diversifiées. Son application à la formalisation de groupes coopératifs de télé-enseignement est actuellement étudiée et devrait déboucher sur une implantation des configurations proposées.

Les services et protocoles d'appartenance dynamique ont été définis pour gérer l'évolution dans le temps de la coopération et pour assurer, suivant les requêtes d'entrée et de sortie des agents coopérants, le passage d'une configuration valide vers une autre configuration. Les prises de décision se font au moyen d'un vote. Ces services utilisent le modèle formel de la coopération pour représenter les groupes et les configurations valides. Par contre, leur conception a été faite en utilisant les techniques de description formelles Estelle et VAL, basées sur les modèles de machines à états et de réseaux de Petri. Ceci introduit deux degrés de formalisation, le premier assurant la représentation des groupes coopératifs, le deuxième servant à la conception et à la réalisation des services et des protocoles. A l'origine, les techniques de description formelles ont été mises au point pour la spécification de services et de protocoles point à point. Le but poursuivi a été de voir si elles pouvaient s'appliquer à la conception de services prévus pour des grou-

pes d'utilisateurs et pour des groupes multipoints. Finalement, ces méthodes formelles Estelle et VAL s'adaptent relativement bien à décrire des comportements de groupe grâce au concept de classe ou de module type qui permettent de représenter le comportement générique d'une entité du groupe, les diverses instances de ces classes ou de ces modules type formant alors la dynamique du groupe par l'entrelacement des comportements élémentaires de ces instances.

La vérification par l'obtention du graphe d'accessibilité et par l'utilisation des techniques de bissimulation est aussi intéressante pour observer le comportement d'un membre du groupe par rapport à l'évolution globale du système et pour voir si les comportements instantanés du système suivent ceux définis par le protocole. Ils permettent de vérifier si le comportement du système suit bien celui attendu. Cependant, cette méthode souffre de quelques inconvénients : la sémantique de représentation des événements est l'entrelacement et elle entraîne très rapidement une explosion combinatoire du nombre d'états globaux du système qui devient trop élevé pour être gérable et manipulable par ordinateur. De plus, la représentation des groupes se heurte également à des facteurs d'échelle pour la vérification. Cette dernière peut s'appliquer à des groupes de petite taille, c'est-à-dire de trois ou quatre membres, mais nécessite trop d'états pour des groupes plus gros. De plus, les graphes d'accessibilité reflètent la structure choisie pour le groupe et sont dépendants de cette structure. D'autres techniques de vérification devraient aussi être considérées et évaluées, avec par exemple l'utilisation de la logique pour décrire un système coopératif afin d'en étudier ses propriétés.

Un autre avantage de la technique de description formelle Estelle est de pouvoir réutiliser des parties de la spécification pour réaliser une implantation. Une démarche possible a été présentée pour distribuer un groupe coopératif spécifique. La partie de contrôle, assurée par des entités Estelle a été découplée de la partie des échanges de données, codée directement en C, pour permettre une plus grande facilité d'adaptation à des données diverses. Actuellement, une nouvelle implantation a été entamée et est en cours de réalisation avec des données multimédias.

Les services et les protocoles de gestion des apartés et de gestion des dépendances entre données sont dans une phase bien moins avancée que celle des services et des protocoles d'appartenance. Le service pour les apartés s'occupe de la création, de la destruction et de la participation de certains coopérants aux apartés. Il contrôle également l'évolution des apartés qui est synchronisée avec l'évolution dynamique de la coopération. Le service et le protocole ont été spécifiés en Estelle. La suite du travail pour la gestion des apartés est de vérifier le protocole, probablement avec la même démarche que celle adoptée pour le protocole d'appartenance dynamique, c'est-à-dire le codage en VAL et l'utilisation des bissimulations à partir du graphe d'accessibilité initial. L'implantation à partir du code Estelle doit aussi être réalisée. Le service pour les dépendances simples permet de créer des relations de dépendances fonctionnelles entre les données des coopérants, de détruire ou de modifier ces relations, puis se charge également des transmissions des valeurs des données au travers des diverses dépendances. Il ne s'applique que pour des dépendances simples pour lesquelles la valeur d'une donnée ne dépend que de celle d'une autre donnée. Ce service pour les dépendances doit donc être étendu à des cas plus généraux de dépendances multiples et, à plus long terme, sa vérification et son implantation pourront être envisagées. En dernier lieu, il est également prévu de distribuer la gestion de l'appartenance et celle des apartés en proposant de nouveaux protocoles.

Malgré quelques inconvénients, l'utilisation des techniques de description formelles pour la spécification, la conception et la réalisation de services et de protocoles pour des groupes coopératifs d'utilisateurs se révèle fort prometteuse. Certaines améliorations sont nécessaires, par exemple pour la vérification exhaustive, mais, dans l'ensemble, les langages de spécification formelle tels Estelle et VAL s'adaptent bien à la description de comportements de groupe et de connexions multipoints. Dans ce mémoire, ils nous ont permis de couvrir toutes les phases du cycle de développement logiciel de services et de protocoles coopératifs, depuis la spécification

initiale s'appuyant sur un modèle pour représenter des groupes coopératifs, en passant par la conception formelle décrite en Estelle, puis par la vérification dans le langage VAL, et en terminant par l'implantation à partir du code Estelle lié à des primitives système réelles de communication. Dans un cadre plus général, cette méthodologie devrait s'inscrire dans une approche multi-paradigmes et multi-outils qui pourrait ainsi être utilisable pour l'ingénierie du protocole et être appliquée à une hiérarchie complète de divers services et protocoles coopératifs.

Références bibliographiques

[AALS94] M. P. van de Aalst, K. M. van Hee et G. J. Houben. Modelling Workflow Management with High Level Petri Nets. Workshop on Computer Supported Cooperative Work, Petri Nets and related formalisms, Saragosse, pages 31-50, Juin 1994.

[AGOS94a] A. Agostini, G. de Michelis et K. Petruni. Keeping Workflow Models as Simple as Possible. Workshop on Computer Supported Cooperative Work, Petri Nets and related formalisms, Saragosse, pages 11-29, Juin 1994.

[AGOS94b] A. Agostini, G. de Michelis, S. Patriarca et R. Tinini. A Prototype of an Integrated Coordination Support System. Computer Supported Cooperative Work (CSCW). An International Journal, 2(4):209-238, Kluwer Academic Publishers, 1994.

[AHAM91] M. Ahamad, M. H. Ammar et S. Y. Cheung. Multidimensionnal Voting. ACM Transactions on Computers Systems, 9(4):399-431, Novembre 1991.

[AHUJ88] S. R. Ahuja, J. R. Ensor et D. N. Horn. The Rapport Multimedia Conferencing System. ACM SIGOIS Bulletin, 9(2&3):1-8, Avril/Juillet 1988.

[ANCI90] P. Ancilotti, B. Lazzerini, C. A. Prete and M. Sacchi. A Distributed Protocol for a Multicomputer System. IEEE Transactions on Computers, 39(5):718-724, Mai 1990.

[ANDE92] COMET: A Toolkit for Multiuser Audio/Video Applications. Proceedings of the IEEE: 12th International Conference on Distributed Computing Systems, pages 555-562, Juin 1992.

[AYOU90] M. Ayoub Dit Ayadi. Contribution à la Spécification Formelle et Vérification d'Architectures de Communication pour les Transactions Distribuées. Thèse de Doctorat de 3ème cycle, Spécialité Informatique, Université Paul Sabatier, Toulouse III, Juillet 1990.

[BANN89] L. J. Bannon et K. Schmidt. CSCW: Four Characters in Search of a Context. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[BAPT90] M. Baptista, S. Graf et C. Rodriguez. Formal Specification and Verification of a Network Independant Atomic Multicast Protocol. Proceedings of the 3rd International Conference on Formal Description Techniques FORTE'90, pages 425-434, 1990.

[BAUD93] V. Baudin, G. Cicchelero, M. Diaz, T. Villemur, F. Baudin et J. F. Schmidt. Intégration des Techniques de Synchronisation et de Coopération dans l'Application de Téléenseignement Aéroformation VACBI. Projet CESAME. Rapport du marché CNET France Télécom 92 1B 178 - Lot 3. Rapport interne LAAS no. 93348, 40 pages, Juin 1993.

[BEAR90] D. Beard, M. Palaniappan, A. Humm, D. Banks, A. Nair et Y. P. Shan. A Visual Calendar for Scheduling Group Meetings. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 279-290, Octobre 1990.

[BECK93] E. E. Beck et V. M. E. Bellotti. Informed Opportunism as Strategy: Supporting Coordination in Distributed Collaborative Writing. Proceedings of the Third European Conference on Computer Supported Cooperative Work, pages 233-248, Eds. G. de Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Septembre 1993.

[BENF91] S. Benford, A. Bullock, P. Harvey, H. Howidi, A. Shepherd et H. Smith. The Grace project. Group Communication in an Open Systems Environment. Final Report. Communications Research Group. Nottingham University. Nottingham NG7 2RD. Septembre 1991.

[BENF92] S. Benford, H. Smith et A. Shepherd, A. Bullock, H. Howidi. Information Sharing Approach to CSCW: The Grace Project. Computer Communications, 15(8):502-508, Octobre 1992.

[BENF93] S. Benford et J. Palme. A Standard for OSI Group Communication. Computer Networks and ISDN systems, 25(8):933-946, Mars 1993.

[BIGN89] C. Bignoli et C. Simone. AI Techniques for Supporting Human to Human Communications in CHAOS. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[BIRM87] K. P. Birman et T. A. Joseph. Reliable Communication in the Presence of Failures. ACM Transactions on Computers Systems, 5(1):47-76, Février 1987.

[BIRM91] K. Birman, A. Schiper et P. Stephenson. Lightweight Causal and Atomic Group Multicast. ACM Transactions on Computers Systems, 9(3):272-314, Août 1991.

[BLAI94] G. S. Blair et T. Rodden. The Opportunities and Challenges of CSCW. Journal of the Brazilian Computer Society, 1(1):3-14, Juillet 1994.

[BODK89] S. Bødker et K. Grønbæk. Cooperative Prototyping Studies - users and designers envision a dental case record system. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[BOND92] A. H. Bond et L. Gasser. A Subject-Indexed Bibliography of Distributed Artificial Intelligence. IEEE Transactions on Systems, Man, and Cybernetics, 22(6):1260-1281, Novembre/Décembre 1992.

[BONF89] A. Bonfiglio, G. Malatesta et F. Tisato. Conference Toolkit: A Framework for Real-Time Conferencing. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[BORE92] N. S. Borenstein. Computational Mail as Network Infrastructure for Computer-Supported Cooperative Work. Proceedings of the ACM SIGCHI & SIGOIS: Conference on Computer Supported Cooperative Work CSCW'92, pages 67-74, Octobre 1992.

[BROTH90] L. Borthers, V. Sembugamoorthy et M. Muller. ICICLE: Groupware for Code Inspection. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 169-181, Octobre 1990.

[BUDK87] S. Budkowski and P. Dembinski. An Introduction to Estelle: A specification Language for Distributed Systems. Computer Networks and ISDN Systems, 14(1):3-23, 1987.

[CHAI92] B. Chaib-Draa, R. Mandiau et P. Millot. Distributed Artificial Intelligence: An Annotated Bibliography. ACM SIGART Bulletin, 3(3):20-37, Août 1992.

[CHER85] D.R. Cheriton et W. Zwaenepoel. Distributed Process Groups in the V Kernel. ACM Transactions on Computers Systems, 3(2):77-107, Mai 1985.

[CHER88] D.R. Cheriton. The V Distributed System. Communications of the ACM, 31(3):314-333, Mars 1988.

[CHEU90] S. Y. Cheung, M. Ahamad et M. H. Ammar. Multi-Dimensional Voting: A General Method for Implementing Synchronization in Distributed Systems. Proceedings of the IEEE: 10th International Conference on Distributed Computing Systems, pages 362-369, Mai 1990.

[CHUN94] G. Chung, K. Jeffray et H. Abdel-Wahab. Dynamic Participation in a Computer-Based Conferencing System. Computer Communications, 17(1):7-16, Janvier 1994.

[CONR91] S. E. Conry, K. Kuwabara, V. R. Lesser et R. A. Meyer. Multistage Negociation for Distributed Constraint Satisfaction. IEEE Transactions on Systems, Man, and Cybernetics, 21(6):1462-1477, Novembre/Décembre 1991.

[COUR87a] J. P. Courtiat, P. Dembinski, R. Groz and C. Jard. Estelle: An ISO language for distributed algorithms and protocols. Technology and Science of Informatics, 6(5):311-324, 1987.

[COUR87b] J. P. Courtiat. Contribution à la Description Formelle de Protocoles. Thèse de Doctorat d'état, Spécialité Informatique, Université Paul Sabatier, Toulouse III, Décembre 1987.

[COUR92] J. P. Courtiat et P. de Saqui-Sannes. ESTIM: An Integrated Environment for the Simulation and Verification of OSI Protocols Specified in Estelle*. Computer Networks and ISDN systems, 25(1):83-98, 1992.

[CROW90] T. Crowley, P. Milazzo, E. Baker, H. Forsdick et R. Tomlinson. MMConf: An Infrastructure for Building Shared Multimedia Applications. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 329-342, Octobre 1990.

[DELT91] Delta-4. A Generic Architecture for Dependable Distributed Computing, Ed. D. Powell, Springer-Verlag, 1991.

[DEMA90] Y. Demazeau et J. P. Müller. Decentralized Artificial Intelligence. In Decentralized A.I., Eds. Y. Demazeau and J. P. Müller, North-Holland, 1990.

[DERM93] G. Dermler, T. Gutekunst, B. Plattner, E. Ostrowski, F. Ruge et M. Weber. Constructing a Distributed Multimedia Joint Viewing and Tele-Operation Service for Heterogeneous Workstation Environments. Proceedings of the IEEE: Fourth Workshop on Future Trends of Distributed Computing Systems, pages 8-15, Septembre 1993.

[DESP93] T. Desprats. Conception des Systèmes Coopératifs : Maîtrise de la Complexité par Modélisation de l'Interaction. Thèse de Doctorat de 3ème cycle, Spécialité Informatique, Université Paul Sabatier, Toulouse III, Septembre 1993.

[DIAZ82] M. Diaz. Modeling and Analysis of Communication and Cooperation Protocols Using Petri Net Based Models. Computer Networks 6(6):419-442, Décembre 1982.

[DIAZ89] M. Diaz and C. Vissers. SEDOS: Designing Open Distributed Systems. IEEE Software 6(6):24-33, Novembre 1989.

[DIAZ92] M. Diaz. A logical model of cooperation. Proceedings of the IEEE: Third Workshop on Future Trends of Distributed Computing Systems, pages 64-70, Avril 1992.

[DIAZ93a] M. Diaz. Coopération, Logique et Partage de Données. Actes de l'AFCET et de l'AFIA, Premières Journées Francophones Intelligence Artificielle Distribuée et Systèmes Multi-Agents, pages 253-262, Avril 1993.

[DIAZ93b] *M. Diaz et T. Villemur. Eléments de Synthèse sur la Coopération et sur le Travail Coopératif. Projet CESAME. Rapport du marché CNET France Télécom 92 1B 178 - Lot 3. Rapport interne LAAS no. 93165, 17 pages, Avril 1993.*

[DIAZ93c] *M. Diaz et T. Villemur. Présentation et Classification des Applications Coopératives. Projet CESAME. Rapport du marché CNET France Télécom 92 1B 178 - Lot 3. Rapport interne LAAS no. 93164, 25 pages, Avril 1993.*

[DIAZ93d] *M. Diaz et T. Villemur. Services et Protocoles d'Adhésion à des Groupes Coopératifs Dynamiques. Projet CESAME. Rapport du marché CNET France Télécom 92 1B 178 - Lot 3. Rapport interne LAAS no. 93274, 48 pages, Juillet 1993.*

[DIAZ93e] *M. Diaz et T. Villemur. Membership Services and Protocols for Cooperative Frameworks of Processes. Computer Communications, 16(9):548-556, Septembre 1993.*

[DIAZ94a] *M. Diaz, P. Owezarski et T. Villemur. Une Définition Logique de la Coopération Basée sur le Partage des Données. Projet CESAME. Rapport du marché CNET France Télécom 92 1B 178 - Lot 3. Rapport interne LAAS no. 94008, 19 pages, Janvier 1994.*

[DIAZ94b] *M. Diaz et T. Villemur. Formation d'Apartés dans des Groupes Coopératifs d'Agents. Projet CESAME. Rapport du marché CNET France Télécom 92 1B 178 - Lot 3. Rapport interne LAAS no. 94172, 20 pages, Avril 1994.*

[DIAZ94c] *M. Diaz et T. Villemur. Communications dans des Groupes Structurés et Dynamiques d'Agents. Actes des 6èmes Rencontres Francophones du Parallélisme RenPar'6, page 325, Juin 1994.*

[DIAZ94d] *M. Diaz, T. Villemur, F. Vernadat et P. Azéma. Verification of Services and Protocols for Dynamic Membership to Cooperative Groups. Rapport interne LAAS no. 94172, 8 pages, Juin 1994.*

[DIAZ94e] *M. Diaz et T. Villemur. Formation of Private Conversation Subgroups in a Cooperative Group of Processes. Journal of the Brazilian Computer Society, 1(1):46-58, Juillet 1994.*

[DOLL89] J. Dollimore et S. Wilbur. Experiences in Building a Configurable CSCW System. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[DURF87] E. H. Durfee, V. R. Lesser et D. D. Corkill. Coherent Cooperation Among Communicating Problem Solvers. IEEE Transactions on Computers, 36(11):1275-1291, Novembre 1987.

[DURF89] E. H. Durfee, V. R. Lesser et D. D. Corkill. Trends in Cooperative Distributed Problem Solving. IEEE Transactions on Knowledge and Data Engineering, 1(1):63-83, Mars 1989.

[EC92] EC ESTELLE to C compiler. Version 3.0. User reference manual. BULL S.A Copyright 1989, 1990. INT Copyright 1991, 1992.

[EDB92] EDB ESTELLE simulator debugger. Version 3.0. User reference manual. BULL S.A Copyright 1989, 1990. INT Copyright 1991, 1992.

[ELLI91] C. Ellis, S. Gibbs et G. Rein. Groupware. Some issues and experiences. Communications of the ACM, 34(1):38-58, Janvier 1991.

[EMER89] E. A. Emerson, J. Srivashana. Branching Time Temporal Logic. Lecture Notes in Computer Science, Volume 354, 1989.

[FERN88] J.C. Fernandez. Aldébaran, Un système de vérification par réduction de processus communicants. Thèse de Doctorat de 3ème cycle, Université de Grenoble, 1988.

[FERN90] J. C. Fernandez et L. Mounier. Verifying Bisimulations on the fly. Proceedings of the Third International Conference on Formal Description Techniques FORTE'90, pages 91-105. Novembre 1990.

[FILI93] G. Filippi et J. Theureau. Analyzing Cooperative Work in an Urban Traffic Control Room for the Design of a Coordination Support System. Proceedings of the Third European Conference on Computer Supported Cooperative Work, pages 171-186, Eds. G. de Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Septembre 1993.

[FISH90] R. S. Fish, R. E. Kraut et B. L. Chalfonte. The VideoWindow System in Informal Communications. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 1-11, Octobre 1990.

[FOX81] M. S. Fox. An Organizational View of Distributed Systems. IEEE Transactions on Systems, Man, and Cybernetics, SMC-11(1):70-80, Janvier 1981.

[FRAI90] Th. Fraichard et Y. Demazeau. Motion Planning in a Multi-Agent World. In Decentralized A.I., Eds. Y. Demazeau and J. P. Müller, North-Holland, 1990.

[FUKS94] H. Fuks et L. Mendonça de Moura. A Document Based Approach for Cooperation. Journal of the Brazilian Computer Society, 1(1):36-45, Juillet 1994.

[GALE92] S. Gale. Desktop Video Conferencing: Technical Advances and Evaluation Issues. Computer Communications, 15(8):517-525, Octobre 1992.

[GARC85] H. Garcia-Molina et D. Barbara. How to assign votes in a distributed system. Journal of the ACM, 32(4):841-860, Octobre 1985.

[GREI88] I. Greif. Computer Supported Cooperative Work: A Book of Readings, Ed. I. Greif, Morgan Kaufman, San Mateo, California, 1988.

[GUIL92] Y. Guillot, J. Guivarc'h, T. Morin et J. Seguin. Télé-Amphi : un service de communication de groupe. L'écho des RECHERCHES (149):3-12, 3ème trimestre 1992.

[HAAK92] J. M. Haake et B. Wilson. Supporting Collaborative Writing of Hyperdocuments in SEPIA. Proceedings of the ACM SIGCHI & SIGOIS: Conference on Computer Supported Cooperative Work CSCW'92, pages 139-146, Octobre 1992.

[HAHN89] U. Hahn, M. Jarke, S. Eherer et K. Kreplin. CoAUTHOR - A Hypermedia Group Authoring Environment. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design. Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[HAHN91] U. Hahn et M. Jarke. Teamwork Support in a Knowledge-Based Information System Environment. IEEE Transactions on Software Engineering, 17(5):467-482, Mai 1991.

[HALP86] J. Y. Halpern. Reasoning about Knowledge: an Overview. Proceedings of the Conference of Theoretical Aspects of Reasoning about Knowledge, Ed. J. Y. Halpern, 1986.

[HARP89] R. R. harper, J. A. Hughes et D.Z. Shapiro. Harmonious Working and CSCW: Computer Technology and air traffic control. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[HELA94] J. M. Hélary, A. Mostefaoui et M. Raynal. Déterminer un état global dans un système réparti. Annales des Télécommunications, 49(7-8), 1994.

[HENN89] P. Hennessy. Information Domains in CSCW. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[HENN92] P. Hennessy, T. Kreifelts et U. Ehrlich. Distributed Work Management: Activity Coordination within the EuroCoOp Project. Computer Communications, 15(8):477-488, Octobre 1992.

[HEWI91] C. Hewitt et J. Inman. DAI Betwixt and Between: From "Intelligent Agents" to Open System Science. IEEE Transactions on Systems, Man, and Cybernetics, 21(6):1409-1419, Novembre/Décembre 1991.

[HOSH92] T. Hoshi, Y. Takahashi et K. Mori. An Integrated Multimedia Desktop Communication and Collaboration Platform for Broadband ISDN: The Broadband ISDN Group Tele-Working System. Computer Communication Review, 22(3):14-15, Juillet 1992.

[HSU93] J. Hsu et T. Lockwood. Collaborative Computing. Byte, 18(3):113-120, Mars 1993.

[HUGH68] G. E. Hughes, M. J. Cresswell. An Introduction to Modal Logic. Methuen, 1968.

[HUGH89] L. Hughes. Multicast Response Handling Taxonomy. Computer Communications, 12(1):39-46, Février 1989.

[INTO91] D. J. Mac Intosh, S. E. Conry et R. A Meyer. Distributed Automated Reasoning: Issues in Coordination, Cooperation, and Performances. IEEE Transactions on Systems, Man, and Cybernetics, 21(6):1307-1316, Novembre/Décembre 1991.

[ISHII90] H. Ishii. TeamWorkStation: Towards a Seamless Shared Workspace. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 13-26, Octobre 1990.

[ISHI91] H. Ishii et N. Miyake. Toward an Open Shared Workspace: Computer and Video Fusion Approach of Teamworkstation. Communications of the ACM, 34(12):37-50, Décembre 1991.

[ISHI93] H. Ishii, K. Arita et T. Yagi. Beyond Videophones: TeamWorkStation-2 for Narrowband ISDN. Proceedings of the Third European Conference on Computer Supported Cooperative Work, pages 325-340, Eds. G. de Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Septembre 1993.

[ISO10021] Message Oriented Text Interchange System (MOTIS), IS 10021 (CCITT X.400), 1990.

[ISO7498] ISO/TC 97/SC 16/WG1. Basic Reference Model for Open System Interconnection, ISO IS 7498, 1983.

[ISO9074] ISO/IEC ISO 9074 : 1989 (E). Information processing systems. Open System Interconnection. Estelle: A formal description technique based on an extended state transition model, ISO IS 9074, Juin 1989.

[ISO9594] The Directory, IS 9594 (CCITT X.500), 1990.

[ISON2031] ISO/TC97/SC21 WG1 N2031 (1987). Working Draft Addendum to ISO 7498-1 on Multipeer Data Transmission, 1987.

[JOHN93] P. M. Johnson et D. Tjahjono. Improving Software Quality through Computer Supported Collaborative Review. Proceedings of the Third European Conference on Computer Supported Cooperative Work, pages 61-76, Eds. G. de Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Septembre 1993.

[JUAN88] G. Juanole, O. Amyay, P. Azema, R. Gustavsson, B. Pershon. Towards a Knowledge-Base for Design of (N) Service-Protocol Pairs -An Epistemic Approach. In Research into Networks and Distributed Applications, Ed. R. Speth, North Holland, 1988.

[KAAS91] M.F. Kaashoek et A.S. Tanenbaum. Group communication in the AMOEBA distributed system. Proceedings of the IEEE: 11th International Conference on Distributed Computing Systems, pages 222-230, Mai 1991.

[KAPL92] S. M Kaplan et A. M. Caroll. Supporting Collaborative Processes with Conversation Builder. Computer Communications, 15(8):489-501, Octobre 1992.

[KARS93] A. Karsenty et M. Beaudouin-Lafon. An Algorithm for Distributed Groupware Applications. Proceedings of the IEEE: 13th International Conference on Distributed Computing Systems, pages 195-202, Mai 1993.

[KARS94] A. Karsenty. Le collecticiel : de l'interaction homme-machine à la communication homme-machine-homme. Technique et science informatiques, 13(1):105-127, 1994.

[KELL94] R. K. Keller, X. Shen et G. v. Bochmann. Macronet - A Simple yet Expressive and Flexible Formalism for Business Modelling. Workshop on Computer Supported Cooperative Work, Petri Nets and related formalisms, Saragosse, pages 51-55, Juin 1994.

[KLEI91] M. Klein. Supporting Conflict Resolution in Cooperative Design Systems. IEEE Transactions on Systems, Man, and Cybernetics, 21(6):1379-1390, Novembre/Décembre 1991.

[KNIS90] M. J. Knister et A. Prakash. DistEdit: A Distributed Toolkit for Supporting Multiple Group Editors. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 343-355, Octobre 1990.

[KRAE88] K. L. Kraemer et J. L. King. Computer-Based Systems for Cooperative Work and Group Decision Making. ACM Computing Surveys, 20(2):115-146, Juin 1988.

[KRAS91] H. Krasner, J. McInroy et D. B. Walz. Groupware Research and Technology Issues with Application to Software Process Management. IEEE Transactions on Systems, Man, and Cybernetics, 21(4):704-712, Juillet/Août 1991.

[KREI89] T. Kreifelts, F. Victor, G. Woetzel et M. Woitass. A Design Tool for Autonomous Group Agents. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[KUMA91] A. Kumar. Hierarchical Quorum Consensus: A New Algorithm for Managing Replicated Data. IEEE Transactions on Computers, 40(9):996-1004, Septembre 1991.

[LARO77] Petit Larousse en couleurs, édition de 1977.

[LEE90] J. Lee. SIBYL: A tool for Managing Group Decision Rationale. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 79-92, Octobre 1990.

[LESS80] V. R. Lesser et L. D. Ermann. Distributed Interpretation: A Model and Experiment. IEEE Transactions on Computers, C29(12):1144-1163, Décembre 1980.

[LESS81] V. R. Lesser et D. D. Corkill. Functionally Accurate, Cooperative Distributed Systems. IEEE Transactions on Systems, Man, and Cybernetics, SMC-11(1):81-99, Janvier 1981.

[LIAN90a] L. Liang, S. T. Chanson et G.W. Neufeld. Process Groups and Group Communications: Classifications and Requirements. IEEE Computer, 23(2):56-66, Février 1990.

[LIAN90b] L. Liang, G.W. Neufeld et S. T. Chanson. Avoiding Name Resolution Loops and Duplications in Group Communications. Proceedings of the ACM: SIGCOMM'90. Communications, Architectures and Protocols, pages 220-230, Septembre 1990.

[MALL91] R. Mallubhatla, K. R. Pattipati, D. L. Kleinmann et Z. B. Tang. A Model of Distributed Team Information Processing under Ambiguity. IEEE Transactions on Systems, Man, and Cybernetics, 21(4):713-725, Juillet/Août 1991.

[MALM93] P. S. Malm. The unOfficial Yellow Pages of CSCW. Disponible par «ftp anonymous» sur le site «gorgon.tft.tele.no» sous le répertoire «pub/groupware». A paraître dans la thèse «Classification of Cooperative Systems from a technological Perspective. Groupware in Local Government Administration». Université de Tromsø. Norvège. Octobre 1993.

[MALO90] T. W. malone et K. Crowston. What is Coordination Theory and How Can It Help Design Cooperative Work Systems?. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 357-370, Octobre 1990.

[MANN89] Z. Manna, A. Pnueli. The Anchored Version of the Temporal Framework, Lecture Notes in Computer Science, Volume 354, 1989.

[MART90] F. Von Martial. Interactions among Autonomous Planning Agents. In Decentralized A.I., Eds. Y. Demazeau and J. P. Müller, North-Holland, 1990.

[MARU90] T. Maruichi, M. Ichikawa et M. Tokoro. Modelling Autonomous Agents and their Groups. In Decentralized A.I., Eds. Y. Demazeau and J. P. Müller, North-Holland, 1990.

[MASA92] S. Masaki, T. Arikawa, H. Ichihara, M. Tanbara et K. Shimamura. A Promising Groupware System for Broadband ISDN: PMTC. Computer Communication Review, 22(3):55-56, Juillet 1992.

[MAZE91] M. S. Mazer. Reasoning About Knowledge to Understand Distributed AI Systems. IEEE Transactions on Systems, Man, and Cybernetics, 21(6):1333-1346, Novembre/Décembre 1991.

[MELO94] W. L. Melo et N. Belkhatir. Collaborating Software Engineering Processes in Tempo. Journal of the Brazilian Computer Society, 1(1):24-35, Juillet 1994.

[MINO93] S. Minör et B. Magnusson. A model for Semi-(a)Synchronous Collaborative Editing. Proceedings of the Third European Conference on Computer Supported Cooperative Work, pages 219-231, Eds. G. de Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Septembre 1993.

[MITT91] R. Mittman. Dans cinq ans, toutes les applications seront des groupwares. Télécom Magazine (9):63-65, Octobre 1991.

[MURA90] T. Murata, V. S. Subrhamanian, T. Wakayama. A High Level Petri Net Model for Reasoning in the Presence of Inconsistency. Proceedings of the 11th International Conference on Application and Theory of Petri Nets, Paris, June 1990.

[MURA91] T. Murata V. S. Subrahmanian et T. Wakayama. A Petri Net Model for Reasoning in the Presence of Inconsistency. IEEE Transactions on Knowledge and Data Engineering, 3(3):281-292, Septembre 1991.

[NAVA92] L. Navarro, W. Prinz et T. Rodden. Open CSCW Systems: Will ODP help?. Proceedings of the IEEE: 12th International Conference on Distributed Computing Systems, pages 547-554, Juin 1992.

[NAVA93] L. Navarro, W. Prinz et T. Rodden. CSCW Requires Open Systems. Computer Communications, 16(5):288-297, Mai 1993.

[NEIL92] M. L. Neilsen, M. Mizuno et M. Raynal. A general Method to Define Quorums. Proceedings of the IEEE: 12th International Conference on Distributed Computing Systems, pages 657-664, Juin 1992.

[NEUW90] C. M. Neuwirth, D. S. Kaufer, R. Chandhok et J. H. Morris. Issues in the Design of Computer Support for Co-authoring and Commenting. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 183-195, Octobre 1990.

[NGOH91] L. H. Ngoh. Multicast support for group communications. Computer Networks and ISDN systems, 22(3):165-178, Octobre 1991.

[OHMO92a] T. Ohmori, K. Maeno, S. Sakata, H. Fukuora et K. Watabe. Distributed Cooperative Control for Sharing Applications Based on Multiparty and Multimedia Desktop Conferencing System: MERMAID. Proceedings of the IEEE: 12th International Conference on Distributed Computing Systems, pages 538-546, Juin 1992.

[OHMO92b] T. Ohmori, K. Maeno, S. Sakata, H. Fukuora et K. Watabe. Distributed Cooperative Control for Sharing Applications Based on Multiparty and Multimedia Desktop Conferencing System: MERMAID. Computer Communication Review, 22(3):39-40, Juillet 1992.

[OWEZ93] P. Owezarski. Etude et Spécification de Systèmes de Communications de Groupes pour les Applications Coopératives. DEA d'Informatique, Université Paul Sabatier, Juin 1993.

[PAGA93] D. S. Pagani et W. E. Mackay. Bringing Media Spaces into the Real World. Proceedings of the Third European Conference on Computer Supported Cooperative Work, pages 341-356, Eds. G. de Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Septembre 1993.

[PALM92] J. Palme et T. Tholerus. SuperKOM-Design Considerations for a Distributed, Highly Structured Computer Conferencing System. Computer Communications, 15(8):509-516, Octobre 1992.

[PAN91] J. Y. C. Pan et J. M. Tenenbaum. An Intelligent Agent Framework for Enterprise Integration. IEEE Transactions on Systems, Man, and Cybernetics, 21(6):1391-1408, Novembre/Décembre 1991.

[PATT90] J. F. Patterson, R. D. Hill et S. L. Rohall. Rendezvous: An Architecture for Synchronous Multi-User Applications. Proceedings of the ACM SIGCHI & SIGOIS: 3rd Conference on Computer Supported Cooperative Work, pages 317-328, Octobre 1990.

[PEHR91] B. Pehrson et S. Pink. Multimedia and High Speed Networking in Multi G. Computer Networks and ISDN Systems, 21(4) 315-319, Juin 1991.

[PRIN89] W. Prinz et P. Pennelli. Relevance of the X.500 Directory to CSCW Applications. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[PRIN93] W. Prinz. TOSCA Providing Organisational Information to CSCW Applications. Proceedings of the Third European Conference on Computer Supported Cooperative Work, pages 139-154, Eds. G. de Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Septembre 1993.

[RAYN85] M. Raynal. Algorithmes distribués et protocoles. Eyrolles, 1985.

[RAYN94] M. Raynal. Synchronisation et état global dans les systèmes répartis. Eyrolles, Collection EDF. Janvier 1992.

[ROBI89] M. Robinson. Pay Bargaining in a Shared Information Space. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[ROBI91] J. Robinson, E. Rubinov, C. Toulson, B. Prasada, S. Sabri, N. Goldberg et G. Vonderweidt. A Multimedia Interactive Conferencing Application for Personal Workstations. IEEE Transactions on Communications, 39(11):1698-1708, Novembre 1991.

[ROBI94] J. A. Robinson. Communications Services Architecture for CSCW. Computer Communications, 17(5):339-347, Mai 1994

[RODD89] T. Rodden et I. Sommerville. Building Conversations Using Mailtrays. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[RODD92] T. Rodden et G. S. Blair. Distributed Systems Support for Computer Supported Cooperative Work. Computer Communications, 15(8):527-538, Octobre 1992.

[RODR93] L. Rodrigues, P. Verissimo et J. Rufino. A Low-Level Processor Group Membership Protocol for LANS. Proceedings of the IEEE: 13th International Conference on Distributed Computing Systems, pages 541-550, Mai 1993.

[ROSE92] M. Roseman et S. Greenberg. GroupKit. A Groupware Toolkit for Building Real-Time Conferencing Applications. Proceedings of the ACM SIGCHI & SIGOIS: Conference on Computer Supported Cooperative Work CSCW'92, pages 43-50, Octobre 1992.

[ROUS92] W. B. Rouse, J. Cannon-Bowers et E. Salas. The Role of Mental Models in Team Performance in Complex Systems. IEEE Transactions on Systems, Man, and Cybernetics, 22(6):1296-1308, Novembre/Décembre 1992.

[SANT93] A. Santos et A. Marcos. An Algorithm and Architecture to Support Cooperative Multimedia Editing. Proceedings of the IEEE: Fourth Workshop on Future Trends of Distributed Computing Systems, pages 2-7, Septembre 1993

[SANT94] A. Santos et B. Trish. Cooperative Multimedia Editing Tool for Enhanced Group Communication. Computer Communications, 17(4):277-287, Avril 1994.

[SARI85] S. Sarin et I Greif. Computer-Based Real-Time Conferencing Systems. IEEE Computer, 33-45, Octobre 1985.

[SASS93] M. A. Sasse, M. J. Handley et S. C. Chuang. Support for Collaborative Authoring via Email: The MESSIE Environment. Proceedings of the Third European Conference on Computer Supported Cooperative Work, pages 249-264, Eds. G. de Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Septembre 1993.

[SCHI93] A. Schiper et A. Sandoz. Uniform Reliable Multicast in a Virtually Synchronous Environment. Proceedings of the IEEE: 13th International Conference on Distributed Computing Systems, pages 561-568, Mai 1993.

[SCRI94] S. A. R. Scrivener, S. M. Clark et N. Keen. The LookingGlass Distributed Shared Workspace. Computer Supported Cooperative Work (CSCW). An International Journal, 2(3):137-157, Kluwer Academic Publishers, 1994.

[SHU92] L. Shu et W. Flowers. Groupware Experiences in Three-Dimensional Computer-Aided Design. Proceedings of the ACM SIGCHI & SIGOIS: Conference on Computer Supported Cooperative Work CSCW'92, pages 179-186, Octobre 1992.

[SMIT81] R. G. Smith et R. Davis. Frameworks for Cooperation in Distributed Problem Solving. IEEE Transactions on Systems, Man, and Cybernetics, SMC-11(1):61-70, Janvier 1981.

[SMIT89] H. Smith, P. Hennessy et G. Lunt. An Object-Oriented Framework for Modelling Organisational Communication. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[STEE81] R. Steeb et S. C. Johnson. A Computer-Based Interactive System for Group Decisionmaking. IEEE Transactions on Systems, Man, and Cybernetics, SMC-11(1):544-552, Janvier 1981.

[STEF87] M. Stefik, G. Foster, D. G. Bobrow, K. Kahn, S. Lanning et L. Suchman. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. Communications of the ACM, 30(1):32-47, Janvier 1987.

[TAKE92] H. Takemura et F. Kishino. Cooperative Work Environment Using Virtual Workspace. Proceedings of the ACM SIGCHI & SIGOIS: Conference on Computer Supported Cooperative Work CSCW'92, pages 226-232, Octobre 1992.

[TANN90] A. Tanenbaum. Réseaux, Architectures, protocoles, applications. InterEditions. 1990.

[TOUZ94] P. Touzeau. Formalism for Documents Circulation Application. Workshop on Computer Supported Cooperative Work, Petri Nets and related formalisms, Saragosse, pages 1-9, Juin 1994.

[TRAV92a] B. Traverson. Transaction Réparties et Mode Client-Serveur. Application dans les Normes OSI. Réseaux et Informatique répartie, 2(4):275-303, 1992.

[TRAV92b] B. Traverson. Le Protocole de Validation en Deux Phases. Stratégies d'Optimisation et Evaluation de Performances. Réseaux et Informatique répartie, 2(4):305-345, 1992.

[TREH93a] M. Tréhel. Présentation Générale de l'Enseignement Assisté à Distance. Projet CESAME. Rapport du marché CNET France Télécom 92 1B 178 - Lot 4. 21 pages, Octobre 1993.

[TREH93b] M. Tréhel. Méthodologie de l'Enseignement Assisté à Distance. Actes du Colloque International de l'Enseignement Supérieur : Stratégies d'apprentissage appropriées, pages 1-19, Août 1993.

[TUEN89] M. Tueni, J. Li et J. Ang. Knowledge-Based Office Automation and CSCW. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[TWID93] M. Twidale, T. Rodden et I. Sommerville. The Designers' Notepad: Supporting and Understanding Cooperative Design. Proceedings of the Third European Conference on Computer Supported Cooperative Work, pages 93-108, Eds. G. de Michelis, C. Simone and K. Schmidt, Kluwer Academic Publishers, Septembre 1993.

[VARD86] M. Y. Vardi. On Epistemic Logic and Logical Omniscience. Proceedings of the Conference of Theoretical Aspects of Reasoning about Knowledge, Ed. J. Y. Halpern, 1986.

[VEDA91] VEDA Release 2.0. Administrator guide. Reference manual. VERILOG. Octobre 1991.

[VERI89] P. Veríssimo, L. Rodrigues et M. baptista. AMp: A Highly Parallel Atomic Multicast Protocol. Proceedings of the ACM: SIG COMM'89 symposium. Communications, architectures and protocols, pages 83-93, Septembre 1989.

[VERN93] F. Vernadat et P. Azéma. Prototypage d'agents communicants. Actes de l'AFCET et de l'AFIA. Premières Journées Francophones Intelligence Artificielle Distribuée et Systèmes Multi-Agents, pages 129-140. Avril 1993.

[VICT89] F. Victor et E. Sommer. Supporting the design of Office Procedures in the DOMINO System. In Studies in Computer Supported Cooperative Work: Theory, Practice and Design, Eds. J. M. Bowers and S. D. Benford, Elsevier, 1991.

[VILL94] T. Villemur, M. Diaz, F. Vernadat et P. Azéma. Verification of Services and Protocols for Dynamic Membership to Cooperative Groups. Workshop on Computer Supported Cooperative Work, Petri Nets and related formalisms, Saragosse, pages 73-91, Juin 1994.

[VILL95] T. Villemur, M. Diaz et F. Vernadat. Une Approche pour la Conception Validée de Services et de Protocoles Coopératifs. A paraître dans le Colloque Francophone sur l'Ingénierie des Protocoles CFIP'95, Mai 1995.

[WANG92] F. J. Wang, S. Lu et J. Liang. Constructing an X-based Teleconferencing System. Proceedings of the IEEE: Third Workshop on Future Trends of Distributed Computing Systems, pages 370-376, Avril 1992.

[WERN88] E. Werner. Towards a Theory of Communication and Cooperation for Multiagent Planning. Proceedings of the 2nd Conference of Theoretical Aspects of Reasoning About Knowledge, Ed. M. Y. Vardi, pages 129-144, 1988.

[WILL94] N. Williams et G. S. Blair. Distributed Multimedia Applications: A Review. Computer Communications, 17(2):119-132, Février 1994.

[WILS91] P. Wilson. Computer Supported Cooperative Work (CSCW): origins, concepts and research initiatives. Computer Networks and ISDN Systems, 23(1-3):91-95, Novembre 1991.

[YOKO92] M. Yokoo et E. H. Durfee. Distributed Constraint Satisfaction for Formalizing Distributed Problem Solving. Proceedings of the IEEE: 12th International Conference on Distributed Computing Systems, pages 614-621, Juin 1992.

[ZIMM80] H. Zimmerman. OSI reference model, the ISO model of architecture for open systems interconnection. IEEE Transactions on Communications, vol. COM-28, Avril 1980.

[ZLOT91] G. Zlotkin et J. S. Rosenschein. Cooperation and Conflict Resolution via Negotiation Among Autonomous Agents in Noncooperative Domains. IEEE Transactions on Systems, Man, and Cybernetics, 21(6):1317-1324, Novembre/Décembre 1991.

Thèse de Monsieur VILLEMUR Thierry

«Conception de services et de protocoles pour la gestion de groupes coopératifs.»

RÉSUMÉ :

Le travail coopératif est un domaine qui étudie le travail de groupes d'utilisateurs de façon générale. Sa finalité est la conception de collecticiels, ensembles logiciels qui contiennent les outils, les applications, et les plate-formes qui supportent les activités de groupes d'utilisateurs. La gestion de ces groupes et les échanges d'information entre leurs membres nécessitent la définition de nouveaux services de communication adaptés aux besoins des agents en coopération.

Les travaux menés dans ce mémoire ont consisté à définir, à concevoir et à gérer la structuration des groupes coopératifs. Un modèle à base de graphes a été proposé à partir du partage de données, pour représenter les relations entre les divers membres d'un groupe coopératif. A partir de ce modèle, un service pour l'entrée et la sortie en coopération des agents coopérants a été défini. Un protocole de communication sous-jacent a été spécifié en utilisant le langage de description formelle Estelle. Le protocole proposé a été vérifié en utilisant l'environnement à base de réseaux de Petri VAL, puis a été implanté en langage C sous UNIX à partir du code Estelle généré. Une extension de ce travail permet la formation d'apartés qui sont des sous-groupes très dynamiques créés à l'intérieur de la coopération. Un autre protocole spécifié en Estelle a été proposé pour gérer la formation de ces apartés et leur évolution au sein de la coopération. En plus de la structuration des groupes, une étude des données qui peuvent être échangées entre des agents coopérants a mené à la définition d'un service de gestion des dépendances de données. Ce service, spécifié également en Estelle, permet de créer, supprimer ou modifier des dépendances entre données, et répercute les modifications de valeurs vers l'ensemble des données dépendantes.

MOTS-CLÉS : Systèmes distribués, Travaux coopératifs, Communications de groupe,
Services et protocoles de communication, Spécifications formelles, Estelle.

"Conception of services and protocols for managing cooperative groups."

ABSTRACT:

Cooperative work is a domain that studies work of user groups in a general way. Its finality is the design of groupware, software sets that contain tools, applications and platforms for supporting the activities of user groups. The management of these groups and the information exchanges between their members require the definition of new communication services and protocols adapted to the needs of cooperative agents.

Work presented inside this dissertation consist in defining, conceiving and managing the structuration of cooperative groups. A model based on graphs has been proposed based on the data sharing to represent the relationships between the different members of a cooperative group. From this model, a service has been defined to manage the entry and exit inside cooperation of the cooperative agents. An underlying communication protocol has been specified using the formal specification language Estelle. The proposed protocol has been verified using the Petri net environment VAL and has been implemented with the C language inside an UNIX environment from the generated Estelle code. An extension of this work allows the formation of private conversation subgroups (PCSs) that are dynamic subgroups created inside cooperations. Another protocol specified in Estelle has been proposed to manage the formation of PCSs and their evolution inside cooperation. In parallel with the group structuration, a study of the data that can be exchanged between cooperative agents has led to the definition of a service that manages data dependencies. This service, specified also in Estelle, allows to create, suppress or modify dependencies between data, and communicates the value changes to the set of dependent data.

KEYWORDS: Distributed systems, Cooperative work, Group communications,
Communication services and protocols, Formal specifications, Estelle.