Collecticiel : modèle d'architecture et validation sur un exemple

Franck TARPIN-BERNARD, Bertrand DAVID, Kamel OUADOU

Laboratoire LISPI
Département MIS
Ecole Centrale de Lyon
36, avenue Guy de Collongue
B.P. 163 69131 Ecully Cedex
Tél: 72.18.64.43 Fax: 78.33.16.15
E-mail: tarpin@cc.ec-lyon.fr, david@cc.ec-lyon.fr

Résumé

Cet article a pour but de présenter un modèle d'architecture pour des collecticiels synchrones permettant une forte collaboration entre participants. Les trois fonctions principales d'un collecticiel (communiquer, coopérer et coordonner) prennent place dans cette architecture et sont mises à la disposition des participants via l'interface utilisateur. Après un bref aperçu bibliographique, nous présentons les différents mécanismes et leur organisation sous forme de modèle d'architecture. Un cas concret est ensuite décrit pour montrer la mise en oeuvre possible du modèle. Il s'agit d'un environnement d'apprentissage de la conduite automobile. Des ouvertures et des nouvelles orientations constituent la conclusion de l'article.

Mots clés

interface homme-machine, interface de coopération, travail coopératif, CSCW, collecticiel synchrone, architecture, point de vue, rôle.

Abstract

This paper aims to present an architectural model for synchronous groupwares that allows a strong collaboration between users. The main three functions of a groupware (i.e. communication, co-operation and co-ordination) are embedded in this architecture and are proposed to the members through the user interface. After a short bibliographical survey, we present the different mecanisms and their organization in form of an architectural model. We detail a concrete example (a driving learning environment) so as to show the possible use of the model. Openings and new orientations frame the conclusion of this paper.

Key words

human-computer interaction, co-operation interface, collaborative work, CSCW, synchronous groupware, architecture, point of view, role.

I. Introduction

L'informatique dans son évolution historique prend progressivement en compte les souhaits des utilisateurs en s'adaptant à leurs exigences dans leur contexte de travail. C'est ainsi que, du travail par lot, on est passé au conversationnel, d'abord textuel, puis graphique et maintenant multimédia, voire multimodal. On a également évolué de la vision individualiste des utilisateurs vers une approche plus collective du travail. L'ignorance mutuelle totale, entretenue par les machines virtuelles protégeant entièrement les utilisateurs, leurs données et leurs applications, a d'abord été rompue par le partage des données, puis par la possibilité de communiquer via des messageries. Maintenant, les collecticiels permettent de raisonner au niveau d'une équipe de collaborateurs et de prendre en compte le processus de travail, avec ses étapes, ainsi que les rôles des différents intervenants.

Les formes du travail coopératif peuvent être décrites par les caractéristiques géographiques et temporelles suivantes [Blair 91] :

du point de vue géographique :
- co-situé (salle de conférence)
- virtuellement co-situé (vidéoconférence)
- à distance (messagerie, ftp).

du point de vue temporel :
- travail asynchrone (systèmes de gestion de fichiers ou systèmes de gestion de documents incluant des fonctionnalités du type : accès, recherche, version, état)
- travail synchrone (partage dynamique des données et du contrôle).

Les collecticiels proposent donc de compléter les fonctionnalités habituelles des logiciels par des fonctions qui permettent de coopérer, coordonner et communiquer [Grudin 94] au sein d'un collectif de participants.

La communication

L'échange d'information est prépondérant dans le travail coopératif. Les progrès des technologies de réseaux et de multimédia ont permis de voir apparaître une multitude de modes de communication :

- applications distribuées (client-serveur)
- messagerie (*mail* classique ou multimédia)
- *scheduler* (accès aux agendas des coopérants)
- *chat* (conversation textuelle)
- *stick-up* (message bref type *post-it* sur l'écran)
- type vidéophone (conversation face à face)
- coup d'oeil (initiation furtive d'une conversation face à face)

Notons que certains produits comme *Montage* [Tang 94] intègrent déjà plusieurs de ces fonctionnalités. Par ailleurs, de multiples travaux sont entrepris pour mettre au point des modèles de messages multimédia, soit à des fins purement structurelles [Borovoy 94], soit dans l'objectif de synchroniser leur propagation dans les réseaux [Chen 92].

La coordination

La coordination prend en compte l'organisation des différentes tâches du travail coopératif dont un des aspects est le *Work Flow Management* [Levan 94]. Il s'agit en effet de gérer les flux de travaux en utilisant des transferts et des parallélisations automatiques des tâches entre les postes de responsabilité. Cet aspect du travail coopératif présente l'intérêt d'obliger les personnes responsables du travail à mener une analyse très fine du processus coopératif afin d'optimiser les temps de traitement et la qualité.

La coopération

La coopération traduit le fait de partager des informations afin de faire ou produire ensemble, c'est à dire de créer une cohésion de groupe tout en renforçant l'implication des individus. Cette coopération peut soit s'effectuer en face à face (éventuellement virtuel), comme dans le cas de téléconférence ou de téléamphi, soit en partageant un espace de production comme dans le cas d'édition partagée ou de conception coopérante. Pour assurer une bonne coopération il est souvent nécessaire de proposer un moyen de conversation (textuel, vocal, ou visuel) pour rendre l'implication des individus plus efficace et renforcer la cohésion du groupe (voire des sous-groupes).

II. Quelques définitions et repères bibliographiques

Dans les applications multi-utilisateurs coopératives, de nouvelles notions apparaissent :

- **le groupe et l'individu** : chaque participant est un individu personnalisé qui appartient à un ou plusieurs groupes de participants. Un groupe est un ensemble d'individus travaillant sur un même domaine. Chaque individu peut avoir conscience du travail des autres membres du groupe.
- **les rôles** : chaque participant, dans chaque groupe et à un instant donné, joue un rôle [David 92]. Ce rôle est caractérisé par l'ensemble des droits et des devoirs du participant vis à vis de ses partenaires et des données (entités) partagées. Ce rôle peut évoluer au cours du temps.
- **les vues** : chaque participant peut avoir sa propre perception (vue) des entités manipulées collectivement. En fait, une vue peut être considérée comme un **filtre** sur un ensemble d'entités partagées définissant un choix de **représentation**. La notion de vue inclue à la fois des choix de représentation, mais aussi des choix d'interaction [Kobialka 91] [Mariani 94]. Une vue peut être publique (accessible par d'autres utilisateurs), privée (accessible seulement par le propriétaire) ou semi-privée (publique pour un groupe, privée seulement sur certains éléments, etc.).
- **le WYSIWIS** (What You See Is What I See) : a pour but d'assurer la **rétroaction du groupe**, c'est-à-dire de permettre à chaque participant de voir ce qu'un autre participant voit ou fait. Cette notion est en fait modulable. Lorsque la vision est vraiment identique chez plusieurs participants, on parle de WYSIWIS strict, tandis que lorsque la vision d'une même situation est différente, on parle de WYSIWIS relâché.

Architectures et mécanismes

Un rapide tour d'horizon des architectures et des mécanismes utilisés pour réaliser des applications multi-utilisateurs nous conduit à citer ici un certain nombre de travaux. Une description plus complète apparaît dans la bibliographie et en particulier dans [Tarpin 95].

L'atelier *Collecticiel* qui a eu lieu à l'occasion d'IHM'92 a structuré en 5 **couches** une application de travail coopératif [David 92] :

- une couche interface utilisateur assurant la communication entre l'homme et la machine, et la communication médiatisée entre les hommes,
- une couche contrôle assurant le filtrage d'information en fonction de l'état de l'application et conformément aux rôles et aux vues des participants,
- une couche noyau applicatif avec la structuration des informations propres à l'application et les rôles et vues associées,
- une couche gestion d'accès assurant l'accès à l'information partagée et gérant la synchronisation et le verrouillage,
- une couche de transport chargée de l'acheminement des informations entre les différents postes.

La couche de gestion d'accès aux données joue un rôle très important car elle gére le partage. Dans *Colab* [Stefik 87], différents **systèmes de contrôle** ont été testés :

- modèle centralisé (trop lent),
- modèle distribué à verrou centralisé (trop lent avec un système d'exploitation non préemptif),
- modèle dupliqué avec propagation asynchrone des changements,
- modèle à détection de dépendance (détection des conflits dont la résolution est à la charge des utilisateurs),
- modèle à verrous errants (les verrous liés à chaque donnée sont gérés par le dernier utilisateur).

Sur le plan de l'architecture, nous rappelons que [Dewan 93] propose une classification en huit catégories des systèmes multi-utilisateurs, dans laquelle sont positionnés les systèmes les plus connus comme *Suite* [Choudhary 92] et *Rendezvous* [Patterson 90].

On peut remarquer qu'il existe une certaine contradiction entre les systèmes distribués, qui tendent à rendre transparente la localisation des utilisateurs, des données et des traitements, et le travail coopératif, qui doit permettre d'avoir conscience de ce que font les autres participants. Une architecture hybride intéressante consiste à centraliser les données et à distribuer les traitements et les interfaces [Labrosse 94]. En effet, la centralisation des données assure un maintien facile de leur cohérence, tandis que la distribution des traitements et des interfaces permet d'accélérer les temps de traitement et d'affichage, et de limiter le nombre de messages circulant à travers le réseau.

Au niveau de l'interface, un des concepts les plus intéressants est celui d'agent de présentation (User Display : UD) proposé dans [Bentley 93]. Un UD gère trois types d'opérations : la focalisation qui est une limitation dynamique du champ de données, la représentation qui est le choix de représentation des objets et la composition qui est l'arrangement spatial des représentations.

Ainsi, un UD est un triplet UD = [Sélection, Représentation, Composition] avec

Sélection = { objets ou attributs} / { critères de sélection }
Représentation = { vues } / { critères de représentation }
Composition = { compositions } / { critères de composition }

La propagation des modifications sur les objets se fait via un serveur.

Enfin, signalons que dans le cadre des recherches sur les applications réparties, un concept de haut-niveau très intéressant pour la gestion des applications coopératives est celui des objets fragmentés [Gourhant 91]. Les différents champs et méthodes d'un objet sont répartis dans le réseau et des mécanismes de gestion "transparents" rendent son comportement similaire à celui d'un objet classique.

Le besoin de dégager un modèle d'architecture constitue une étape naturelle pour industrialiser la démarche de production de ce type d'application.

III. Architecture proposée

Nous présentons ici, un modèle d'architecture pour des applications coopératives. Il s'agit d'une architecture hybride destinée principalement aux applications dites synchrones (temps réel).

Définitions

Dans le monde des applications mono-utilisateur, le terme **agent** est utilisé pour définir une entité autonome. Il est préféré à **objet**, utilisé tantôt dans un sens similaire à celui d'agent (ex: PAC), tantôt dans un contexte d'implémentation.

Dans le cadre des collecticiels, la répartition conduit soit à la duplication soit à la fragmentation des agents. Dans le cas des collecticiels permettant un WYSIWIS relâché, une réplication au moins partielle semble nécessaire. Elle conduit à distinguer la notion d'**agent de référence** de celle d'**agent local**. En effet, si une entité est manipulée par *n* participants, un agent de référence constitue comme son nom l'indique, l'unique référence pour les autres entités. Les *n* agents locaux répartis sur les postes de travail des participants supportent les manipulations qui dépendent des spécificités exprimées par des rôles et des points de vue. Les agents locaux d'un même agent de référence sont dits **agents frères**.

La stratégie de contrôle conduit à choisir la façon de gérer le parallélisme dans le travail. Ainsi, pour maintenir une cohérence stricte du système, nous choisissons d'adopter la stratégie suivante : pour chaque agent de référence, à un instant donné, on ne peut délivrer qu'à un seul agent local le droit de modifier les caractéristiques partagées avec ses agents frères. Cet agent local est appelé **agent maître**. Les autres agents frères sont temporairement les **esclaves** de l'agent maître.

Chaque participant travaille sur un **poste** dit **client** sur lequel s'exécute la partie de l'application coopérative correspondante, c'est-à-dire l'ensemble des agents locaux. Ceux-ci permettent d'accéder aux agents manipulés par ses collaborateurs sur d'autres postes clients. Nous envisageons deux modes d'accès sur lesquels nous reviendrons plus tard : soit via le *serveur* de coopération, soit directement. Dans la Figure 1, nous présentons un schéma de principe de notre architecture dans lequel apparaît la répartition des différents agents.

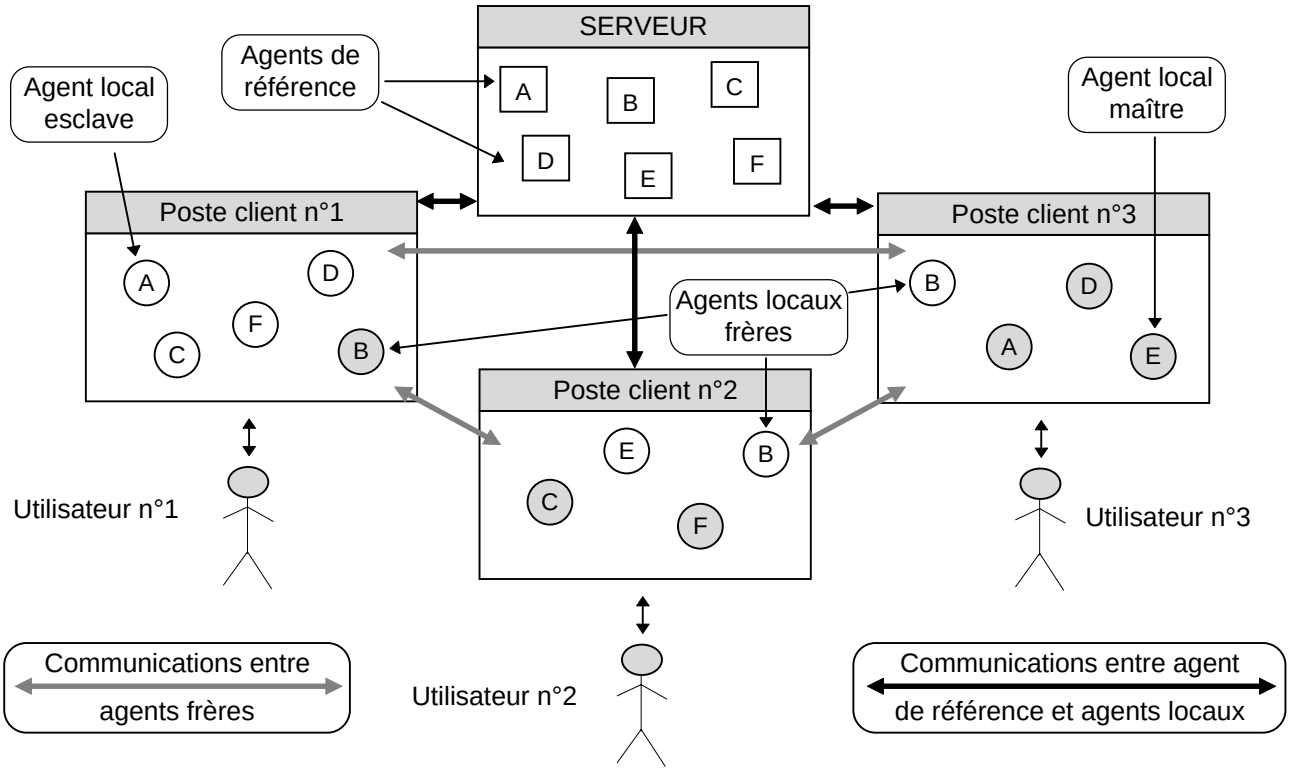


Figure 1 : Schéma de principe de l'architecture

Modèle AMF

Pour la modélisation des agents et de l'organisation architecturale d'une application coopérative, nous avons choisi de prendre comme point de départ le modèle multi-agents multi-facettes, appelé AMF, qui a fait l'objet de la thèse de K. Ouadou [Ouadou 94]. Ce modèle inspiré du modèle PAC [Coutaz 87] propose d'une part une décomposition plus fine des agents interactifs, et d'autre part, un formalisme puissant de représentation des gestions du contrôle. Ainsi chaque agent est structuré en facettes. Parmi ces facettes, on retrouve les composants classiques du modèle PAC, auxquels s'ajoutent des facettes issues, soit d'une décomposition fine du composant *contrôle*, soit d'une duplication des facettes classiques (plusieurs facettes *présentation*), soit de l'identification de nouveaux aspects des agents (ex: la coopération). Pour un agent AMF classique, d'une application mono-utilisateur, on trouve généralement les facettes suivantes :

- *présentation* (la commande des E/S avec l'utilisateur),
- *abstraction* (les données logiques informatiques),
- *contrainte* (les relations entre les facettes de l'objet ou avec d'autres objets),
- *erreur* (les procédures et messages de traitement d'erreurs),
- *aide* (les aides en ligne et contextuelles),
- *contrôle* (les administrateurs assurant les communications entre les autres facettes et certaines liaisons avec les autres objets).

La transposition du modèle AMF dans l'environnement coopératif peut se faire de deux façons. Dans une première approche, on fragmente les agents AMF en distribuant leurs facettes (dont par exemple une facette présentation par utilisateur). Cette approche convient particulièrement pour le WYSIWIS strict. L'autre approche, plus particulièrement souhaitable pour le WYSIWIS relâché, conduit à dupliquer les agents et à s'appuyer sur les notions abordées précédemment d'agent de référence, agents locaux et agents frères.

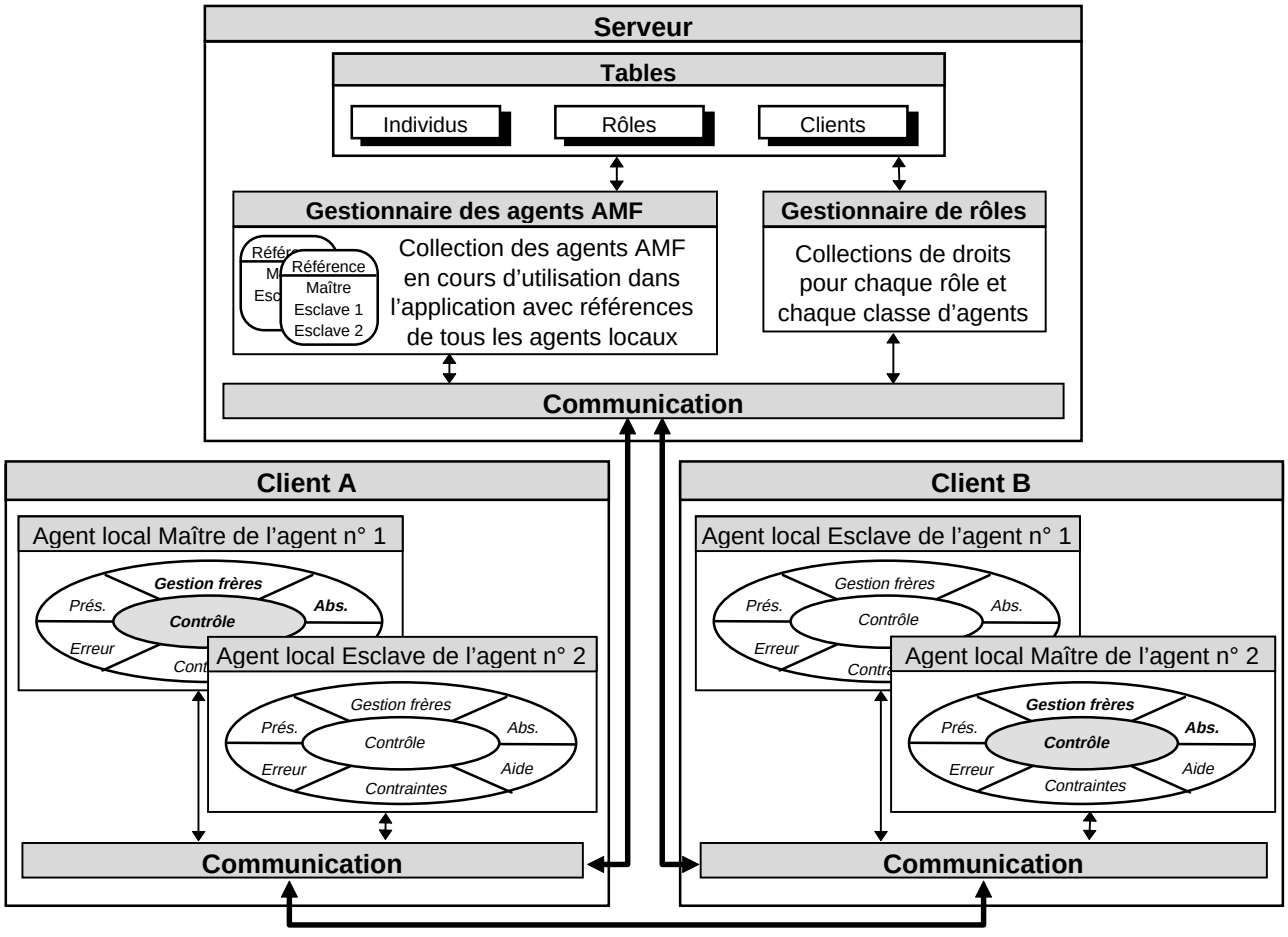


Figure 2 : Schéma général de l'architecture

Structure du serveur

Le serveur est l'interlocuteur initial des clients lors des phases de connexion. Il a en charge les trois fonctions vues plus haut (communication, coopération, coordination).

- Gestion de la coordination : il a pour rôle de gérer les individus (liste des participants) et leurs rôles (définitions et règles d'évolution).
- Gestion de la communication : il assure l'émission et la réception des informations. Il gère donc la localisation des participants sur les postes clients (liste des adresses et références des clients). Pour communiquer avec les agents AMF, il utilise un système de messages propagés par un protocole standard (TCP/IP par exemple).
- Gestion de la coopération : le **gestionnaire des agents AMF** gère les objets partagés manipulés par les différents participants (création et accès aux objets partagés). La gestion des droits liés à chaque classe d'agent et à chaque rôle d'utilisateur est assurée par le **gestionnaire de rôles**.

Organisation des agents locaux

Pour gérer le parallélisme d'action des différents utilisateurs, tout en maintenant la cohérence du système, nous pouvons adopter différentes stratégies de contrôle. Nous avons choisi une des plus simples en ne donnant, à un instant donné, le droit de modifier une entité qu'à un seul utilisateur. Les contrôles à mettre en place doivent choisir la granularité des entités ainsi que la façon de gérer la propagation des autorisations de modification (sollicitation explicite ou détection implicite).

Sur ce point, nous sommes soucieux de privilégier le participant qui est en train de manipuler une entité. C'est pourquoi, nous avons choisi de promouvoir momentanément l'agent local, support d'action du participant, en agent maître. Toute modification effectuée par un agent local nécessite donc préalablement une prise de pouvoir de l'agent, qui passe ainsi du statut d'esclave à celui de maître. L'agent maître informe de toute modification ses agents frères. En fonction de la topologie des réseaux supports, et pour des raisons d'efficacité, nous souhaitons que les agents locaux frères puissent communiquer directement entre eux. Ils conservent donc à jour, via le serveur, une liste des points d'entrées des agents frères. Tout agent local est donc capable de s'adresser à son agent de référence et à ses frères.

La phase de création des agents est une phase critique car elle impose de construire des liens entre des références dynamiques. Ainsi, les éléments *communication* présentés dans le schéma général précédent sont des agents qui utilisent des techniques de transfert de structures dynamiques semblables à celles utilisées dans les RPC (Remote Procedure Call) [Birrell 84]. La création d'un agent de référence (ex: un document), nécessite des échanges entre le contexte d'un utilisateur et le Gestionnaire des agents AMF du serveur. La requête correspondante va aboutir, d'une part, à la création de l'agent local maître au niveau du poste client, et d'autre part, à l'enregistrement du point d'entrée (référence) de cet agent dans la table des agents du serveur. Ce nouvel agent local maître est bien sûr une instance d'une classe d'agents dont les droits ont été définis initialement par le gestionnaire de rôles du serveur. De fait, lorsque l'on désire manipuler ou modifier un agent local, les droits liés au rôle du participant lui sont notifiés localement. A chaque instant, on peut demander au gestionnaire d'agents AMF du serveur la liste des agents de référence (voire des agents locaux) qu'un participant peut atteindre. Cette liste dépend bien sûr du (ou des) rôle(s) joué(s) par le participant. Lorsque l'on désire accéder à un agent local situé sur un autre poste client, le gestionnaire d'agents AMF du serveur reçoit une demande d'accès. Celle-ci conduit, d'une part, à la création d'un nouvel agent local (avec les droits du participant demandeur), et d'autre part, à la mise à jour de la table d'agents du serveur et de celles de ses agents frères.

Exemple de fonctionnement

Pour illustrer le fonctionnement de notre modèle, nous proposons d'observer, sur la Figure 3, les circuits d'information dans le cas de la manipulation d'un représentant local d'un agent de référence accessible à plusieurs participants (existence d'au moins deux agents locaux).

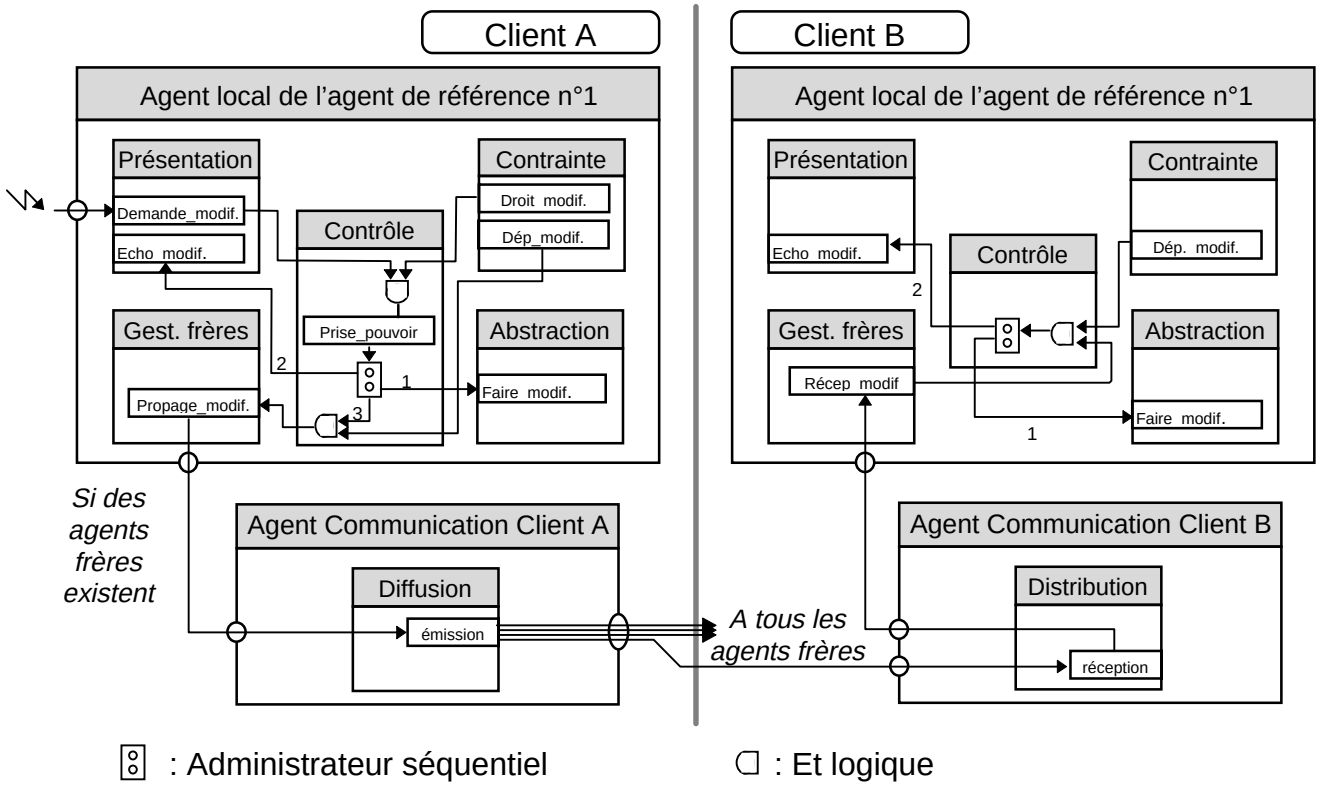


Figure 3 : Exemple de gestion du contrôle lors de la manipulation d'un agent local

On notera que, dans cette situation, l'agent local recevant le stimulus du participant doit tout d'abord prendre le pouvoir (bloc *Prise_pouvoir*) pour devenir un agent maître (si l'action est autorisée, c'est-à-dire si *Droit_modif* est vrai). La répercussion de l'action du participant est alors localement transmise aux fonctions adéquates (*Faire_modif* et *Echo_modif*).

De plus, si l'on ne relâche pas le WYSIWIS pour cette fonction (c'est-à-dire si l'indicateur de dépendance *Dép_modif* est vrai) et si d'autres agents frères existent sur d'autres postes clients, l'appel de la fonction est propagé. Quand le message arrive à l'agent frère, la fonction concernée est invoquée si l'indicateur de dépendance est vrai. Cette fonction peut être liée au rôle de l'utilisateur de l'agent frère. D'une façon générale, les différentes **vues** peuvent être exprimées soit par des fonctions alternatives, soit par la définition de facettes *présentation* interchangeables choisies lors de la construction des agents locaux ou lors de l'évolution du rôle des utilisateurs.

La modification de l'état des indicateurs de dépendance permet de modifier dynamiquement le relâchement de la rétroaction de groupe.

La technique de contrôle proposée ici est surtout intéressante dans des cas de fonctionnement dans lesquels pendant une période donnée, les actions entreprises sur un agent sont surtout produites par un seul participant. En effet, une fois qu'un agent est devenu maître, il peut agir très rapidement sur ses facettes.

IV. Application

Afin de montrer l'intérêt du modèle d'architecture proposé et de le valider, nous avons défini une application comportant les caractéristiques suivantes :

Caractéristiques	Intérêts
Utilisateurs typés	Gestion de rôles
Présence d'objets multiformes	Utilisation de plusieurs facettes *présentation*. Gestion de vues différentes.
Présence de processus coopératifs synchrones	Validation des techniques de propagation. WYSIWIS strict ou relâché.
Extension possible vers les multimodalités	Perspectives d'enrichissement

Le domaine de l'EAO, sur lequel nous travaillons par ailleurs, constitue un vaste champ applicatif pour les collecticiels, notamment dans le cadre de la coopération synchrone. Il nous semble intéressant d'utiliser une application de ce type pour illustrer nos travaux, même si celle-ci ne prétend en aucun cas constituer un archétype de *groupware*.

L'application que nous avons choisie d'étudier est une application destinée à l'**apprentissage du code de la route**.

Présentation de l'application

Notre application d'apprentissage du code de la route fait intervenir deux grandes catégories d'utilisateurs : les moniteurs et les élèves. Le principe de fonctionnement que nous avons adopté est le suivant : chaque moniteur soumet au groupe d'élève qu'il supervise des situations de test. Il répond à leurs questions et juge leurs résultats.

Objectifs principaux de l'application :

- Permettre à un élève d'être supervisé par un moniteur.
- Permettre à un moniteur de travailler avec plusieurs élèves.
- Proposer à l'élève un environnement de conduite adapté à son niveau.
- Proposer à l'élève des situations de test élaborées par le moniteur.
- Permettre au moniteur d'intervenir dynamiquement dans la situation de test en cours d'un élève.
- Permettre au moniteur d'avoir accès à l'environnement visuel de l'élève.
- Permettre au moniteur d'avoir une possibilité d'évaluation des élèves à partir de comptes-rendus d'activité générés par le système.
- Proposer des outils de communication tels que télépointeur [Primet 95] et conversation textuelle, puis orale, voire multimodale.

Objectifs futurs de l'application :

- Permettre à un élève de collaborer avec un autre élève voire de le rencontrer sur un circuit routier.
- Permettre à un superviseur d'analyser la coopération entre moniteurs et élèves.

Nous avons choisi de représenter les différents environnements de travail sous forme de fenêtre de type MDI (Multiple Document Interface), c'est-à-dire de fenêtres contenant d'autres fenêtres filles iconisables. Mais, avant de proposer une illustration de ces environnements, nous allons détailler les principaux éléments de cette application.

La situation

Chaque élève est confronté à une situation proposée par le moniteur. Structurellement, une situation peut être définie par trois éléments :

- Un **problème** : Il s'agit de la question qui est posée à l'élève. Cette question peut être du type " que faites-vous dans la situation suivante : ...? " ou " comment réalisez-vous la manoeuvre suivante : ... ? " ou plus simplement " choisissez parmi *n* propositions ".
- Un **contexte** : ce sont les éléments qui décrivent la situation. Dans le cadre de la réalisation d'une manoeuvre, le contexte peut être facultatif. On peut considérer dans un premier temps, qu'un contexte est décrit par un ensemble de trois types d'éléments :
  - un décor (image bitmap ou objets structurés tels que routes, maisons, voitures garées, etc.) pouvant être, soit une vue à travers le pare-brise pour le conducteur (type diapositive auto-école), soit un schéma (type gestion de priorités),
  - des éléments statiques porteurs de sens pour la conduite (panneaux de signalisation, panneaux indicateurs, marquages au sol, etc.).
  - des éléments potentiellement dynamiques (autres véhicules, 2 roues, piétons, feux tricolores, agent de circulation, etc.).
- Une **solution** : Il s'agit d'une part de l'ensemble des actions devant répondre au problème, et d'autre part des commentaires automatiques pouvant être associés aux erreurs de l'élève. Suivant l'intelligence associée à cette solution, on peut envisager une analyse pas à pas des actions de l'élève ainsi qu'une supervision du timing des réponses de ce dernier.

Plutôt que de chercher à développer un automate intelligent capable de fournir l'analyse pédagogique des actions de l'élève, il nous semble intéressant de déléguer cette tâche à la personne la plus apte à la réaliser, c'est à dire le moniteur. En effet, si le moniteur peut suivre de façon synchrone les actions de l'élève, il lui sera possible non seulement de les commenter en direct, mais aussi d'agir directement sur les éléments potentiellement dynamiques de la scène. Ainsi, si un élève est confronté à une situation dans laquelle il doit effectuer un dépassement, et qu'il déboîte sans avoir regardé dans son rétroviseur ou contrôlé son angle mort, le moniteur peut agir sur un autre véhicule pour provoquer un accident et ainsi prouver à l'élève que cette opération est indispensable.

L'éditeur de situation du moniteur doit donc permettre de construire des problèmes, des contextes, ainsi que d'enregistrer les solutions et les commentaires automatiques d'erreurs.

NB : Il est intéressant de constater que les éléments du véhicule sont actionnables par l'élève, tandis que le moniteur peut seulement les voir. D'un autre côté, les éléments potentiellement dynamiques sont actionnables par le moniteur (ou par l'application), tandis que l'élève peut seulement les voir. Par ailleurs, certains outils de communication pourront être actionnables par les deux types d'utilisateurs.

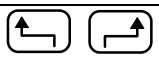
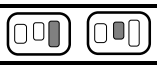
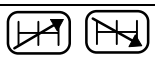
Le véhicule

Nous nous intéressons à la modélisation du poste de pilotage du véhicule dans le cadre de situations où l'environnement n'est pas réaliste, c'est-à-dire dans lesquelles l'objectif n'est pas de fournir un simulateur de conduite dans un environnement virtuel.

Nous imaginons deux types de représentation de la vision du véhicule par l'utilisateur.

Représentation symbolique

Le poste de conduite est représenté par un ensemble de boutons correspondant aux actions élémentaires à la disposition du conducteur.

Type d'action	Action	Représentation
Manoeuvre	Tourner à gauche, à droite	
	Déboîter par la gauche, la droite	
Action sur la vitesse	Accélérer, freiner	
Action sur les vitesses	Avancer, reculer	
	Monter, descendre un rapport	
	Passer une vitesse	
Action sur les manettes	Clignotant à gauche, à droite	
	Allumer veilleuses, codes, phares	
Action visuelle	Rétroviseur intérieur, extérieur	
	Contrôle angle mort	

Cette représentation est surtout destinée aux élèves débutants confrontés à des situations de bases (manoeuvres avec contexte très limité).

On peut naturellement imaginer à terme une liste plus exhaustive des actions incluant les commandes des éléments suivants : embrayage, essuie-glaces, avertisseur, warning, huile, température, compte-tours, dégivrage, etc.

Représentation réaliste

Le poste de conduite est représenté par un tableau de bord réaliste, un pare-brise et des rétroviseurs. Comme nous l'avons précisé en préambule, il n'est pas immédiatement nécessaire de construire un simulateur de conduite dans lequel la vitesse et les variations angulaires du volant seraient à prendre en compte pour réaliser une véritable conduite virtuelle.

Le compte rendu

Il s'agit d'un élément permettant de mémoriser les actions de l'élève face aux situations. Cette zone est principalement alimentée automatiquement par l'application, mais on peut imaginer que l'élève comme le moniteur peut y ajouter des commentaires.

La fiche d'évaluation

Il s'agit d'une zone permettant d'évaluer les performances et les progrès de l'élève. Elle est remplie par les moniteurs, et est bien sûr consultable par l'élève.

Les outils de communication

On peut imaginer pour ces outils, des solutions complexes intégrant des caractéristiques multimédia. Dans une première réalisation, nous nous intéressons uniquement à l'utilisation d'un canal de conversation textuelle ou vocal et à un outil de télépointage.

Dans les Figures 4 et 5, nous présentons une illustration de ce que pourraient être les environnements des élèves et des moniteurs.

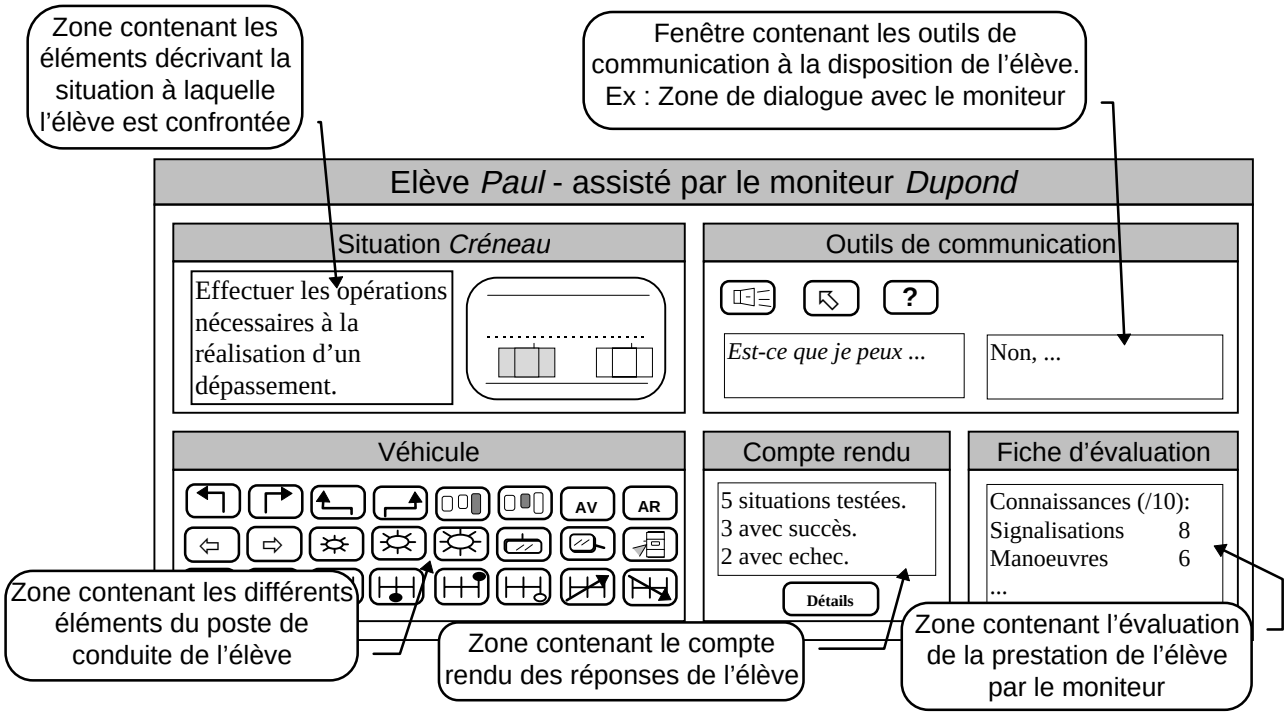


Figure 4 : Environnement d'un élève

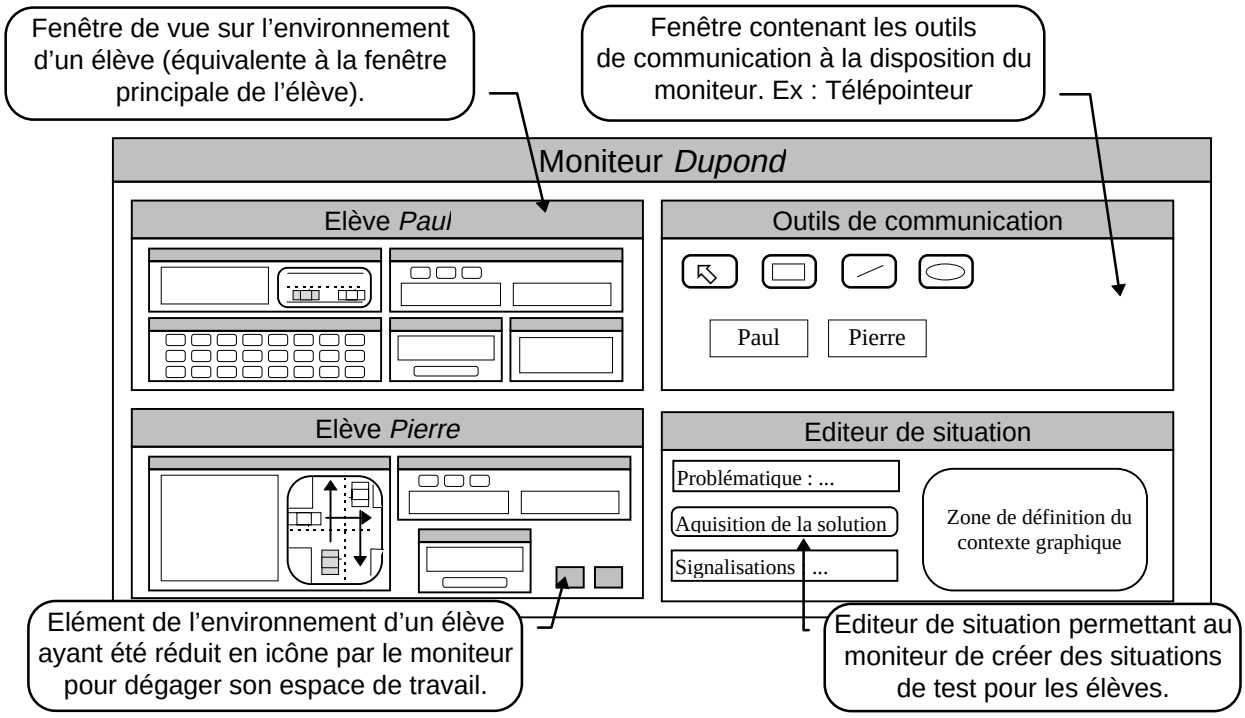
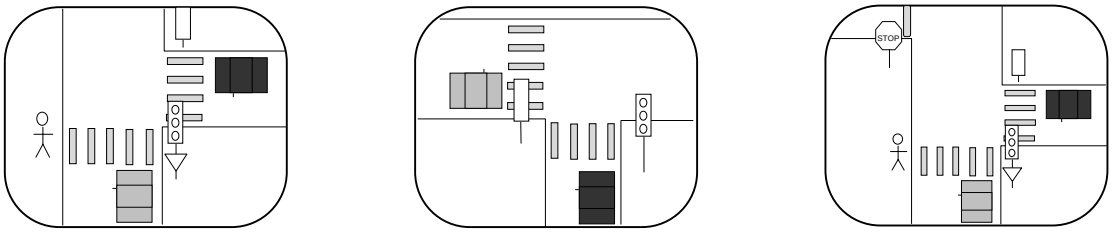


Figure 5 : Environnement d'un moniteur

Comme nous l'avons indiqué précédemment, nous envisageons de faire partager des situations dynamiques à plusieurs élèves. Ainsi, il est possible de faire évoluer les élèves sur un même " circuit ", ce qui les amènera à se rencontrer et donc à faire face à des situations " réelles " sous la surveillance du moniteur. En fait, cette application peut évoluer vers un simulateur coopératif de conduite.

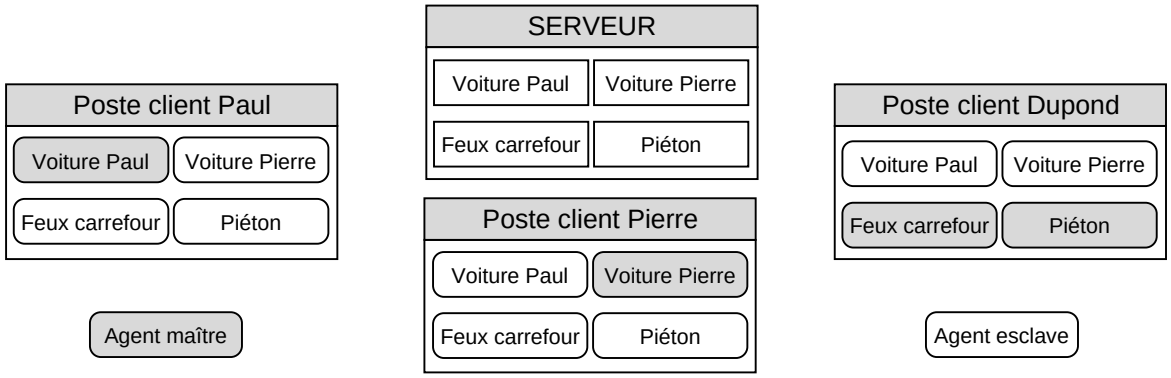
Un tel contexte peut d'ailleurs nous permettre d'illustrer sur un cas réel les répartitions possibles d'agents interactifs. Dans l'exemple qui suit, seuls les participants possèdant les agents maîtres (ici toujours situés chez les mêmes clients) peuvent agir sur leurs comportements. Ainsi, chaque élève ne peut agir que sur son véhicule, tandis que le moniteur ne peut directement agir que sur les éléments dynamiques de la situation.

Représentations graphiques d'une situation de conduite mettant en jeu plusieurs élèves



Elève Paul | Elève Pierre | Moniteur Dupond

Répartition des principaux agents intervenant dans cette situation



Pour valider les mécanismes de propagation et de hiérarchisation des droits, on peut imaginer l'existence d'un superviseur étudiant les sessions d'apprentissage des équipes moniteurs-élèves. Son environnement pourrait comporter une zone décrivant les différents groupes moniteurs-élèves en train de travailler avec l'application tandis qu'une autre zone permettrait d'évaluer l'aspect coopératif de l'enseignement, c'est-à-dire notamment d'observer les flux de dialogue à l'intérieur des groupes.

V. Conclusion et ouvertures

Nous avons rapidement esquissé l'évolution conduisant au CSCW et aux collecticiels, avec quelques repères bibliographiques. Puis nous avons proposé un modèle d'architecture d'applications coopératives applicable principalement aux systèmes coopératifs pour applications synchrones.

Notre modèle d'architecture repose sur le modèle AMF, un modèle multi-agents d'applications interactives mono-utilisateur, qu'il généralise. Il s'agit d'une architecture répliquée pour laquelle nous proposons un contrôle " semi-centralisé ". Ce terme signifie, qu'à un instant donné, pour chaque agent de référence, le système se comporte comme un système centralisé puisque seul l'agent local maître peut modifier son état et celui de ses agents frères. Cependant, à la différence d'un système purement centralisé, l'agent maître peut varier au cours du temps. De plus, les agents maîtres des agents de référence ne se situent pas tous sur le même poste client.

Dans le cadre de l'étude du relâchement de la rétroaction de groupe, nous envisageons d'étudier d'autres stratégies de contrôle (contrôles proactifs ou réactifs avec différents niveaux de granularité de verrouillage). Après des évaluations fonctionnelles et ergonomiques de ces stratégies, nous analyserons les possibilités d'évolution de notre modèle, vers un système auto-adaptable.

Pour rendre notre modèle utilisable, nous travaillons sur l'extension des outils d'AMF, et nous étudions la mise en oeuvre d'un système de programmation visuelle permettant d'exprimer les règles de partage et de dépendance conduisant à une gestion efficace des niveaux de relâchement des rétroactions.

Enfin, nous préparons des extensions de notre modèle dans le domaine de la gestion des interactions multimédia et multimodales. Cet axe de recherche est étudié, dans le cadre de l'évolution de notre application test d'apprentissage du code de la route, en intégrant des possibilités de dialogues oraux et de la vidéo, ainsi que des ouvertures vers des systèmes de commande plus adaptés au domaine.

VI. Bibliographie

[Bentley 93] *Richard Bentley, Tom Rodden, Pete Sawyer, Ian Sommerville,* **Architectural Support for Cooperative Multi-User Interfaces,** Université de Lancaster. 1993. 22 pages.

[Birrell 84] *A. Birrell, B. Nelson,* **Implementing remote procedure calls,** ACM TCS, Vol.2 n°1, p. 39-50, fév. 1984.

[Blair 91] *Gordon Blair, Tom Rodden,* **CSCW and Distributed Systems : The Problem of Control,** Rapport Université de Lancaster. 1991. 19 pages.

[Borovoy 94] *R. D. Borovoy, E. B. W. Cooper, R. K. E. Bellamy,* **Media Fusion : An Application of Model-Based Communication,** CHI '94 Proceedings. p 17-18.

[Chen 92] *Mon-Song Chen, P. V. Rangan, Harrick M. Vin,* **System Support for Computer Mediated Multimedia Collaboration,** CSCW '92 Proceedings. p 203-209.

[Choudhary 92] *R. Choudhary, P. Dewan,* **A high-level and flexible framework for implementing multiuser interfaces,** ACM Transactions on information systems, 10 (4), 1992. 345-380.

[Coutaz 87] *Joëlle Coutaz,* **PAC, an Implementation Model for Dialog Design,** Interact'87. Stuttgart. Septembre 1987. p 431-436.

[David 92] *Bertrand David, Jean-Claude Duby (coordinateurs),* **Collecticiel,** IHM'92. Compte rendu des ateliers. 4° Journées sur l'ingénierie des IHM. Telecom Paris 1992. p 59-88

[Dewan 92] *Prasun Dewan,* **Principle of Designing Multi-User Interface Development Environments,** Engineering for Human-Computer Interaction, J. Larson & C. Unger (Editors) Elsevier Science Publishers B. V. (North Holland), IFIP, 1992. p 35-50.

[Dewan 93] *Prasun Dewan,* **User Interface Software. Ch 8 : Tools for implementing Multi-User Interfaces,** Ed L. Bass, P. Dewan, John Wiley & Sons. 1993. p 149-193.

[Foster 86] *Greg Foster,* **Collaborative Systems and Multi-User Interfaces,** Rapport de Thèse n°UCB/CSD 87/326. University of California. Berkeley. 1986. 190 pages.

[Gourhant 91] *Yvon Gourhant, M. Makpangou, J-P. Le Narzul, M. Shapiro,* **Structuring Distributed applications as fragmented objects,** Rapport INRIA n° 1404. Janvier 1991. 30 pages.

[Grudin 94] *Jonathan Grudin, Steven Poltrock,* **Computer-Supported Cooperative Work and Groupware,** CHI '94 Proceedings. p 355-356.

[Kobialka 91] *Hans-Ulrich Kobialka,* **Modelling Cooperative Work in integrated Environments,** Rapport Gesellschat für Mathematik und Davenverarbeitung (GMD). 1991. 11 pages.

[Labrosse 94] *Frédéric. Labrosse,* **Coopératisation d'une application mono-utilisateur existante. Application à Autocad R 12 pour Windows,** Mémoire CNAM. Lyon. 1994. 150 pages.

[Levan 94] *Serge K. Levan, Anne Liebman,* **Le groupware, informatique, management et organisation,** Paris, Hermès 1994.

[Mariani 94] *John Mariani, Tom Rodden, Jonathan Trevor,* **The Use of Adapters to Support Sharing,** Rapport Université de Lancaster. 1994. 13 pages.

[Ouadou 94] *Kamel Ouadou,* **AMF : Un modèle d'architecture multi-agents multi-facettes pour Interfaces Homme-Machine et les outils associés,** Mémoire de Thèse, Ecole Centrale de Lyon. 1994. 210 pages.

[Patterson 90] *J.F. Patterson, R.D. Hill, S.L. Rohall, W.S. Meeks,* **Rendezvous: An architecture for synchronous multi-user applications,** CSCW'90 Proceedings. Oct. 1990. p 317-328.

[Primet 94] *P. Primet, S. Akkouche,* **Réflexions autour du concept de télépointeur,** IHM'94, Lille, p. 176-182, déc. 1994.

[Rodden 94] *Gareth Smith, Tom Rodden,* **An Access Model for Shared Interfaces,** Rapport Université de Lancaster. 1994. 22 pages.

[Shen 93] *HongHai Shen, Prasun Dewan,* **Access Control for Collaborative Environments,** CSCW '92 Proceedings. p 51-58.

[Stefik 87] *Mark Stefik, Gregg Foster, D.G. Borrow, Kenneth Kahn, Stan Lanning, Lucy Suchman,* **Beyond the ChalkBoard : Computer support for Collaboration and Problem Solving in Meetings,** Communication of the ACM. Janvier 1987. 10 (1). p. 32-47.

[Tang 94] *John C. Tang, Monica Rua,* **Montage : Providing Teleproximity for Distributed Groups,** CHI '94 Proceedings. p 37-43.

[Tarpin 95] *F. Tarpin-Bernard,* **Travail coopératif : état de l'art,** Rapport de recherche LISPI-ECL, janvier 1995.