Une plateforme collecticielle

J. Caelen, H. Kabré
ICP-INPG, 46, Av. Felix Viallet
F-38031 Grenoble Cedex
e-mail : jcaelen@icp.grenet.fr, kabre@icp.grenet.fr

Thème 5 : Présentation de cas et de réalisation
Sous thème : Systèmes multimédias et multimodaux
Communication : orale ou affichée
Portée : organisation d'un système informatique et expérimentation dans le domaine du collecticiel et du multimodal

Résumé

Cet article décrit un environnement informatique — plateforme collecticielle — qui permet de faire collaborer un ensemble d'applications collecticielles et multimodales sur un réseau local. La plateforme est conçue pour les développeurs, l'exploitant du système et bien sûr les utilisateurs des collecticiels. L'architecture est "semi-répliquée" pour faire cohabiter des applications hétérogènes et est constituée d'agents fonctionnels permettant de prendre en compte les problèmes de transport des informations (couche basse), la gestion des ressources multimedia, les communications (couche intermédiaire), les interactions et les règles de collaboration (couche haute). Cette architecture est discutée pour ses aspects liés à la collaboration. Deux collecticiels (ICPdraw et ICPplan) ont été intégrés à la plateforme, qui en montrent l'intérêt. Nous insistons surtout dans cet article sur la gestion des objets et des transactions entre applications dans un tel environnement.

1. Introduction

De plus en plus de tâches effectuées avec l'aide de l'ordinateur se font en groupe (Biel, 1991). Ainsi voit-on se développer un domaine appelé "groupware" en anglais (Ellis et al. 1991) que l'on a traduit par *collecticiel* en français. Ce domaine fait plus généralement partie de celui des interfaces homme-machine, lequel connait actuellement un essor sans précédent, notamment par la publicité qui lui est faite mais surtout par les nombreuses applications maintenant possibles grâce à la puissance des machines : téléconférences, interfaces multimodales, réalité virtuelle, réalité augmentée, etc. (Zarmer, 1992). Les champs d'application ne deviendront vraiment productifs que lorsque ces interfaces (a) permettront à plusieurs personnes de travailler ensemble dans ces univers nouveaux et (b) offriront des moyens de communication "écologiques" avec la machine. Ainsi, un collecticiel, au sens où nous l'entendons, est un logiciel de travail collaboratif et multimodal — c'est-à-dire un logiciel à travers lequel plusieurs utilisateurs peuvent collaborer ou utiliser des ressources informatiques communes en interagissant au moyen de plusieurs modes (parole, écriture, geste, etc.). Sur le plan informatique, un collecticiel s'appuie obligatoirement (du fait de la distance des utilisateurs) sur un système distribué (Andrews, 1991) constitué de postes de travail et de services délocalisés (ressources de calcul, de

données, d'applications, de périphériques, etc.).

Ce contexte étendu — travail collaboratif et multimodalité — soulève de nombreux problèmes :
• la gestion et le contrôle de tous les services offerts sur le réseau dans ce cadre d'une collaboration généralisée,
• la définition des règles de collaboration,
• le contrôle des tâches et des multiples fils d'activité,
• la gestion dynamique des droits, rôles et privilèges des utilisateurs,
• le filtrage des informations méta-communicationnelles (c'est-à-dire des informations échangées par les usagers entre eux),
• les communications multimodales entre les utilisateurs et la machine,
• et secondairement, la "réutilisabilité" de certaines couches du logiciel.

Notre but est également d'offrir un environnement de développement facile à utiliser pour réaliser des collecticiels intégrant des applications hétérogènes. Il doit être conçu de manière à ne pas dépendre de telle ou telle application ni du type de système d'exploitation et à répondre au mieux aux points décrits ci-dessus.

La plateforme décrite ci-après, s'adresse donc à la fois aux utilisateurs, aux développeurs et à l'exploitant du système.
(a) l'utilisateur doit pouvoir disposer d'un poste de travail multimodal, d'un ensemble de collecticiels et de *ressources distantes avec accès transparent*,
(b) le développeur doit pouvoir *intégrer toute nouvelle application* dans la plateforme et disposer de ressources de développement (hardware et software) adaptées,
(c) l'exploitant du système doit pouvoir faire un *suivi automatique de plans d'exécution* et une *gestion des communications et des ressources.*

2. Cahier des charges

2.1. Description d'un poste utilisateur

Un poste vu du côté utilisateur doit comprendre :
– des entrées-sorties multimédia (microphone, haut-parleur, gant numérique et/ou souris, clavier, écran, téléphone) en nombre variable,
– un panneau de contrôle et de visualisation des tâches et des applications en cours,
– des outils de configuration des modes de communication,
– un ensemble de collecticiels (par ex. traitement de texte, dessin, etc.) multimodaux,
- un accès à un arbitre pour régler les conflits dûs à la collaboration et au partage des objets.

2.2. Description du rôle de l'exploitant

L'exploitant du système n'intervient pas directement sur et dans le collecticiel. Il doit mettre à disposition les ressources nécessaires aux utilisateurs, gérer leurs droits en dehors des sessions de travail proprement dites et pouvoir observer le bon déroulement général d'une session. En cas de dysfonctionnement il doit pouvoir intervenir de manière non invasive en réglant les conflits provenant des contraintes informatiques. Il ne joue pas le rôle d'arbitre au cours d'une session.

2.3. Description des attentes du développeur

De son côté le développeur souhaite ré-utiliser le plus possible le code écrit pour une autre application. Il ne devrait pas s'occuper des couches basses et hautes décrites ci-après. Il doit pour cela disposer de spécifications précises, de standards et d'outils pour développer rapidement de nouvelles applications.

2.4. Description d'un collecticiel

Un collecticiel doit pouvoir être manipulé par plusieurs utilisateurs se trouvant dans des lieux et sur des machines distinctes. Il doit aussi offrir des "vues" (Stefik et al., 1987) différentes d'un même objet aux différents utilisateurs. Pour cela, il faut définir des rôles et des règles "sociales" permettant de coordonner les actions effectuées de manière coopérative sur les objets de l'application. De là découleront les vues différentes présentées aux utilisateurs (et non l'inverse).

De façon générale, un collecticiel comprend (Duby et al., 1992) (fig. 1) :
• un niveau de transport, par réseau, des informations aux postes de travail distants,
• des outils d'accès à ces informations circulant sur le réseau,
• le noyau applicatif avec son interface multimodale,
• un niveau de contrôle et de gestion de toutes les applications travaillant en collaboration.

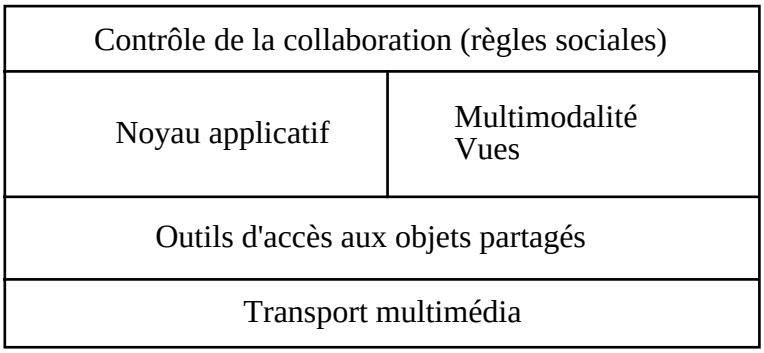

Fig. 1 : décomposition d'une application collaborative en couches fonctionnelles

Les services offerts par les bas niveaux concernent :
• la capture décentralisée des événements multimodaux,
• la gestion des fonctions de reconnaissance (Ye et al. 1990) et de synthèse pour certains modes (parole, geste, etc.),
• la mise en communication des applications devant collaborer.

Les services offerts par les hauts niveaux concernent (Croisy et al., 1992) :
• la gestion des groupes de travail,
• la mise en communication directe des usagers,
• l'établissement et le maintien des règles de collaboration,
• la gestion des droits pendant une session (Pankoke-Babatz, 1989).

Quant aux services de niveau intermédiaire, ils sont classiques dans les interfaces multimodales (Caelen et al. 1992b) :
• fusion et fission des informations multimodales,
• dialogue avec l'usager,
• gestion des vues et contrôle de la présentation,
• exécution de la tâche applicative.

2.5. Pour une plateforme collecticielle

Un utilisateur placé devant son poste de travail doit pouvoir passer d'une application à une autre, voire travailler en temps partagé sur plusieurs applications. L'exploitant doit contrôler toutes ces activités et des ressources diverses, le développeur veut utiliser un environnement de programmation riche et cohérent : pour toutes ces raisons, une plateforme collecticielle nous semble une solution qui peut répondre à ces exigences.

3. Architecture générale de la plateforme

Une architecture distribuée est évidemment nécessaire pour que des personnes puissent travailler à distance; elle permet en outre de :
– concentrer les équipements spécialisés et coûteux sur une station dédiée,
– rendre ces équipements disponibles à de nombreux utilisateurs depuis leur poste banalisé,
– optimiser l'utilisation des ressources, en assurer une meilleure cohérence et faciliter la maintenance.

La plateforme collecticielle doit donc comprendre :
– des postes de travail individuels avec une périphérie multimédia,
– un serveur de ressources software pourt le développeur d'applications,
– un serveur de ressources hardware pour concentrer certains équipements,
– un poste central pour l'exploitant du système.

On supposera dans la suite que le réseau qui supporte cette architecture est suffisamment puissant pour accepter le trafic haut débit nécessaire à tous les échanges de données (nous faisons l'hypothèse que ce problème sera résolu dans l'avenir). La fig 2 montre un schéma de l'architecture matérielle d'une telle plateforme.

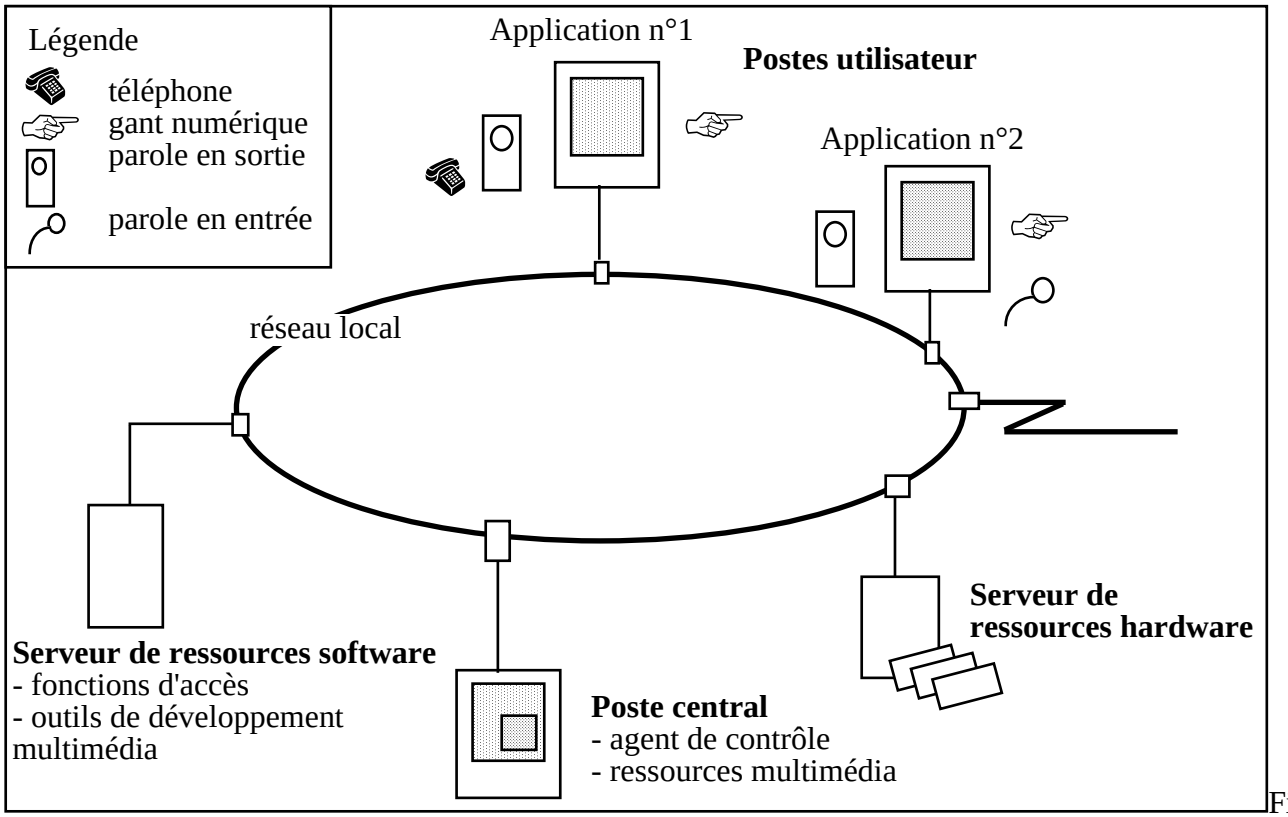

Fig. 2 : Architecture matérielle de la plateforme collecticielle

Du côté software il faut disposer de logiciels :
– de contrôle de l'activité et de la collaboration sur la plateforme,
– d'un gestionnaire de ressources,
– d'outils de développement multimédia,
– et bien sûr d'applications collecticielles et multimodales.

En généralisant la notion de service à toutes les ressources y compris les ressources multimédia et en autorisant l'exploitant du système à observer la boucle de travail, nous arrivons à une architecture distribuée fondée sur la notion client-serveur mais possédant un tableau noir centralisé contenant les informations minimales permettant à l'exploitant de contrôler l'activité générale de la plateforme. Cela nous a conduit à l'architecture logicielle de la fig. 3.

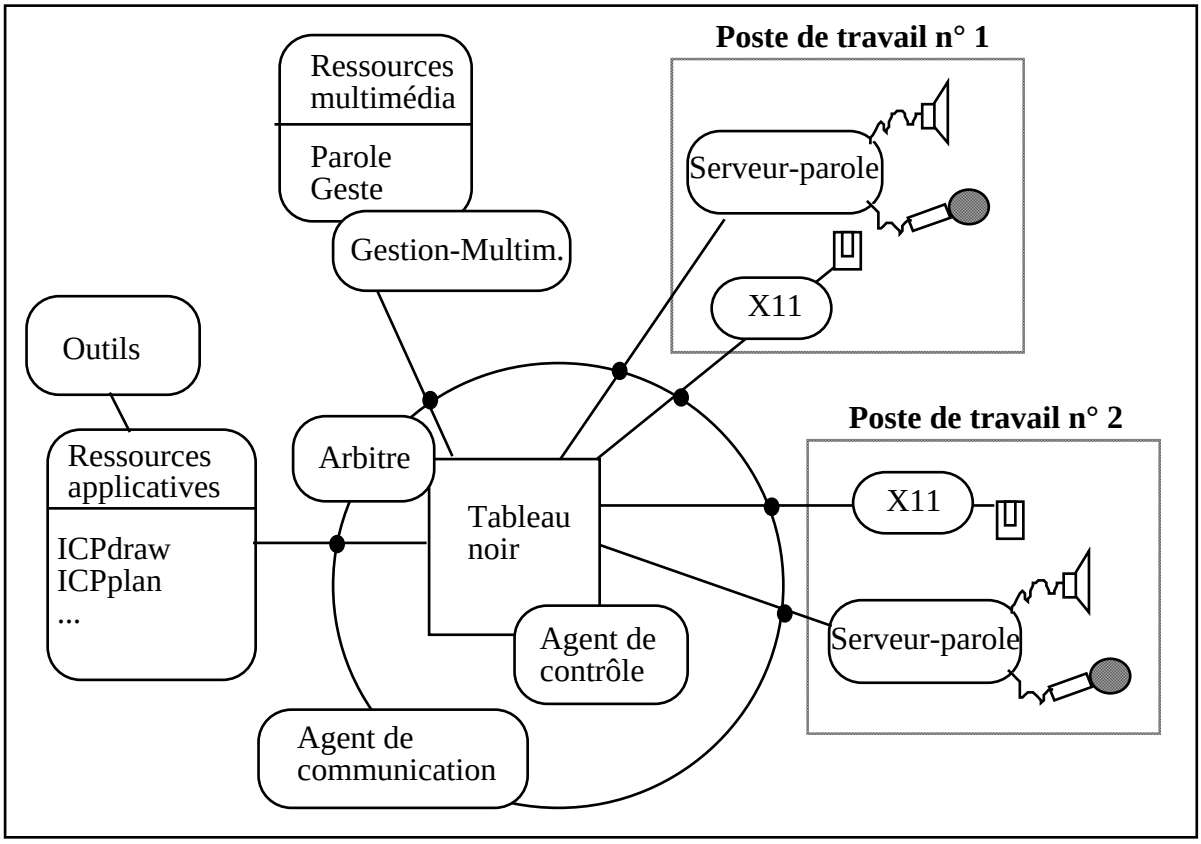

Fig.3 : Architecture logicielle de la plateforme collecticielle multimodale pour le traitement, par exemple, de la parole et du geste-souris. Ce schéma montre notammemnt que toutes les informations ne passent pas par le tableau noir.

Nous décrivons ci-après quelques uns des composants de la plateforme.

3.1. Le gestionnaire multimédia

Le gestionnaire multimédia a pour rôle de détecter les signaux circulant sur le réseau et de les délivrer aux agents de traitement idoines accessibles comme des ressources multimédia. Inversement un ordre de transmission de signal lui étant donné par ceux-ci, il doit mettre des signaux en circulation sur le réseau. Par exemple s'il détecte un signal de parole en provenance d'un poste de travail quelconque et que ce signal est à interpréter, il le capte et l'adresse à l'agent de reconnaissance de la parole (ce dernier tournant par exemple sur une carte spécialisée). Il récupère ensuite la chaîne de caractères issue de la reconnaissance et l'envoie sur le réseau afin qu'un autre agent la capte à son tour pour la traiter.

3.2. L'agent de communication

Nous avons choisi de séparer la gestion de la collaboration en trois niveaux : (a) groupe, (b) membre et (c) session. Les deux premiers sont durables alors que la session est "volatile". Un groupe est un ensemble de personnes travaillant ou ayant travaillé ensemble sur une même application. Un membre est une personne pouvant adhérer à un ou plusieurs groupes. Une session est la séance de travail en cours.

Le rôle de l'agent de communication est de mettre un utilisateur en situation de collaboration c'est-à-dire à lancer une session s'il est le premier membre à vouloir se connecter ou sinon à l'intégrer dans

une session active. Son rôle est ensuite de mettre en communication les applications collecticielles qui ont été entre temps répliquées sur les différents postes des membres de la session, et enfin d'assurer la déconnexion convenable — ou l'abandon en cours de session — des membres les uns après les autres.

Avant de pouvoir travailler de manière collaborative, il faut créer un groupe de travail. Ensuite seulement, des utilisateurs (membres) peuvent décider d'adhérer à ce groupe. Une fois créé, ce groupe a une durée de vie allant jusqu'à son auto-destruction qui doit être proclamée par démission des adhérents. Ces notions sont relativement proches des notions de groupe et d'utilisateur sous UNIX (Banino et al., 1989), (Cheriton et al. 1988), et c'est pour profiter de ses facilités de gestion que nous avons choisi ce découpage (la structure des fichiers est notamment la même). La session a une durée de vie égale à celle de la session de travail,. Cette durée est analogue à un temps de connexion (avec login) sur le système UNIX. Les données propres à une session sont détruites à la fin de la session. Seules sont sauvegardées les données de l'application (figées au moment de l'arrêt de la session) pour une reprise ultérieure éventuelle dans une nouvelle session.

L'agent de communication intervient dans les trois phases suivantes :
• la connexion à une session,
• la déconnexion à une session,
• l'arrêt de la session.

3.2.1. Connexion à une session

La connexion à une session doit s'envisager de plusieurs manières selon qu'il s'agit (a) d'une première activation par un ou plusieurs membres d'un groupe ou selon qu'il s'agit (b) de la demande d'adhésion d'un nouveau membre à une session déjà active.

Examinons ces deux cas :

(a) la nouvelle session est la première à s'activer,
Après vérification de l'identité des membres du groupe qui demandent à activer la session, celle-ci se met à jour à partir de la dernière sauvegarde de l'application effectuée lors de la déconnexion du dernier membre actif. Le travail du gestionnaire consiste à :
• retourner à l'application le nom du dernier membre connecté lors de la précédente session de travail,
• ce qui provoque la duplication sur tous les postes de travail de l'application dans l'état ou elle se trouvait à la fin de la session précédente (il est évident que s'il s'agit d'une toute première activation, un état "initial" est présenté aux adhérents).

(b) la session est déjà active,
Il y a d'autres membres en train de travailler. Après autorisation à se connecter (donnée par l'arbitre) le nouvel adhérent entre dans l'état en cours de l'application. Pour cela, le gestionnaire contacte l'un des membres actifs de la sesion en lui donnant l'adresse du nouveau membre. L'application de ce dernier a en charge sa propre duplication sur le poste de travail du nouveau membre. La procédure de mise à jour étant propre à une application, l'agent n'intervient pas dans cette phase. Pour en permettre l'exécution correcte il bloque cependant l'activite du groupe afin de maintenir la cohérence des informations entre les différents membres (sinon ceux-ci auraient pu continuer à modifier les données de l'application pendant le temps de la connexion).

Ces processus nous garantissent de l'indépendance de l'agent de communication et des applications. Il ne s'occupe pas non plus des conventions et des règles sociales qui régissent le travail des membres. Ce

soin en est laissé à l’arbitre. Cela montre le souci avec lequel nous avons découpé la plateforme en modules indépendants.

3.2.2. Déconnexion d'une session

La connexion à une session suit le processus inverse d'une connexion à une session déjà active : l'adhérent qui souhaite quitter la session doit céder ses droits et privilèges à un autre membre du groupe. Après quoi l'exemplaire de son application est détruite. Dans ce cas le rôle de l’agent de communication se limite au strict minimum : celui de signaler aux autres utilisateurs cet abandon et le destinataire des droits puis d'envoyer un message d'auto-destruction à l'application.

3.2.3. Arrêt d'une session

L'arrêt de la session pend effet lorsque le dernier membre actif se déconnecte. L’agent adresse deux messages à la dernière application active : le premier de sauvegarde des données et le deuxième d'auto-destruction. Il mémorise dans le fichier du groupe le dernier utilisateur et l'application traitée.

3.2.4. Maintenance de la sécurité

Le fait d'introduire des structures permanentes pour les groupes pose la question de la sécurité. Il ne faut pas que n'importe qui ait accès au travail d'un groupe. Pour cela, nous avons instauré trois niveaux de mots de passe :

• le premier est le mot de passe du groupe. La connaissance de celui-ci permet d'adhérer au groupe et de consulter des informations qui y sont attachées (droits, rôles, etc.). Un groupe a des membres ordinaires et un responsable de groupe, sorte de "super-utilisateur" (vocabulaire tiré d'UNIX),
• le deuxième est celui du responsable de groupe, dont la connaissance donne le droit de modifier certaines caractéristiques du groupe sous le contrôle de l'arbitre,
• le troisième est un mot de passe standard d'un membre attribué lors de l'adhésion au système (le même pour tous les groupes si ce membre fait partie de plusieurs groupes). Il lui permettra de se connecter, ou de céder ses droits au moment de la déconnexion, etc.

3.3. L’arbitre

L'arbitre est un module expert en “comportement social”. Il possède un certain nombre de règles de bon ccomportement social nécessaires à la conduite d'un travail en commun.

3.3.1. Les droits attachés aux objets

Lorsqu'une session de travail est active, chaque utilisateur a devant lui une application répliquée qui lui donne une vue qui dépend de son rôle dans le groupe. Il a la possibilté d'agir sur l'interface afin de modifier les objets de l'application. Comme il n'est pas le seul à pouvoir le faire il faut définir ses droits à ce sujet. Pour cela il faut maintenir à jour une table donnant les droits de chaque membre sur tous les objets au fur et à mesure de leur création. Il faut ausi pouvoir modifier ces droits en cours de session (avec l'agrément de l'arbitre). Ainsi apparait la nécessité d'introduire un gestionnaire d'objets qui puisse agir de manière décentralisée. Pour cela, et toujours dans une perspective de généralité maximum et d'indépendance avec les applications, nous avons introduit trois types de droits :

• l'observateur : cet utilisateur n'a le droit que de consulter l'état de l'objet et de le voir,
• le propiétaire : cet utilisateur a tous les droits sur un objet qu'il a créé (modification, destruction) mais il les perd à la déconnexion. Par défaut ils sont attribués au responsable du groupe. En cours de session il peut céder ou partager ses droits avec un tiers,
• le responsable du groupe : il a tous les droits sur tous les objets et en particulier le droit de modifier les droits des autres membres sur l'objet.

La gestion de ces droits conduit à quelques règles permettant de contrôler la cohérence :
• un objet peut avoir plusieurs propriétaires,
• à sa création un objet n'a qu'un propriétaire, son créateur, les autres membres en sont observateurs,
• tous les conflits éventuels sont tranchés en dernier recours par le responsable,
• le responsable peut destituer un membre de son droit de propriété,
• un propriétaire peut retrouver ses droits dans une session ultérieure.

L’arbitre est donc un module qui fait appliquer ces règles. A l'activation d'un exemplaire d'une application il se charge de mettre à jour la table des droits d'accès pour l'utilisateur concerné. Cette table est décentralisée et attachée à l'application instanciée. L'application consulte cette table en local pour valider ses actions sur les objets.

L'utilisation de ce service d’arbitrage des objets n'est pas obligatoire : les applications peuvent développer leur propre gestion, en s'appuyant par exemple sur les mécanismes de messagerie offerts par le gestionnaire. Attardons-nous quelque peu sur ces mécanismes.

3.3.2. La messagerie

Nous avons vu que le rôle principal de l’agent de communication est d'acheminer tous les messages pour le groupe et de créer des “pipe-line” entre les applications. De leur côté les applications (et instances d'applications) n'ont pas à connaître précisément les autres applications connectées simultanément. Ce mécanisme masque aussi le fait que différentes applications peuvent utiliser différents media pour communiquer (sockets, files de messages, événements X, etc), et donc qu'il faut utiliser différents modes d'adressage pour communiquer avec elles. Tout cela est rendu transparent pour le programmeur.

Les services offerts par la messagerie sont de quatre types :

• Diffusion de message avec et sans acquittement,
• Envoi de message avec et sans acquittement.

La diffusion sans acquittement est le service de base du gestionnaire. Lorsque l'on reçoit une demande de diffusion de la part d'une session, on diffuse le message associé vers toutes les autres sessions actives sans attendre une confirmation de bonne réception. Il en est de même pour l’envoi.

La diffusion (ou envoi) avec acquittement repose sur le même principe, mais on demande aux destinataires de s'acquitter lors de la réception du message. Lorsque tous se sont acquittés, le gestionnaire renvoie un acquittement à l'émetteur pour lui confirmer de la prise en compte de son message.

On peut s'interroger sur l'utilité d'une diffusion avec acquittement, puisque les protocoles de communication utilisés sont fiables (par TCP/IP). Cependant, nous avons été confrontés à un problème

délicat, qui survient lorsqu'une demande de connexion arrive alors qu'une diffusion est en cours. Dans ce cas le message risque de passer inaperçu pour la personne qui est en cours de connexion. La solution est donc d'attendre la prise en compte complète du message (qui doit donc être acquittée) avant d'autoriser une nouvelle connexion. Pour les mêmes raisons une demande de connexion doit aussi être acquittée.

Pour éviter tout risque de famine, lorsqu'une connexion a été mise en attente, on diffère également toute demande de diffusion acquittée jusqu'à ce que la connexion soit réalisée. Sinon, on risquerait de ne jamais pouvoir se connecter.

L'envoi de message consiste à acheminer un message depuis une session vers une autre, et non pas vers tout le groupe. Dans son principe il s'apparente donc à une diffusion (acquittée ou non-acquittée). Tous ces services de messagerie sont disponibles directement indépendamment de l'agent arbitre et peuvent être utilisés par le développeur pour mettre en œuvre d'autres services de gestion du travail collaboratif.

3.4. L'agent de contrôle

A chacun des niveaux groupes, membres, sessions, il est possible de consulter l'état de la situation : caractéristiques des groupes existants, des membres et des sessions actives. L'agent de contrôle se réduit pour le moment à ces capacités de visualisation. Cela permet seulement de surveiller l'activité des groupes mais non de les superviser.

3.5. Les applications

Les applications que nous avons développées — ICPdraw éditeur de dessin (Caelen et al., 1992a) et ICPplan conception de plans architecturaux assistée par ordinateur (Bourguet, 1991) — ont été intégrées à la plateforme. Elles ont toutes deux une interface multimodale utilisant la parole et le geste. Il a suffi pour les intégrer de redéfinir les structures de données pour les faire communiquer dans le cadre d'applications répliquées.

Les serveurs (Arons, 1992), (Reichbuach, 1992) et ressources multimédia ont été mis au point et définies à l'issue du projet ESPRIT Multiworks (Caelen et al, 1991). Ils ne présentent pas d'intérêt particulier pour le sujet de cet article.

4. Réalisation

Toute la programmation est orientée autour d'éléments de base appelés Entités (Kabré, 1991) : une entité est un processus possédant la capacité d'envoyer et de recevoir des messages. Par exemple, les entités composant l'agent de communication sont les suivantes :

• le serveur
• le dispatcher
• le superviseur
• le gestionnaire de groupes

Le fonctionnement de ces processus est asynchrone (Guimarlaes, 1992).

Le rôle du serveur est de constituer l'unique interface entre les applications collaboratives et le

gestionnaire de groupes. Le rôle du dispatcher est d'envoyer les messages destinés à l'arbitre vers l'entité concernée de celui-ci. Le superviseur traite toutes les opérations dites d'administration. Il s'agit de tout ce qui concerne la création et la destruction de groupes, adhésion et démission de membres, ainsi que les modifications des mots de passe, etc. C'est egalement lui qui est chargé d'activer les groupes lors de la réception de la première demande de connexion. Une fois le groupe activé, le dispatcher envoie les demandes de connexion directement au gestionnaire du groupe.

Les processus communiquent par des Liens (Mayer, 1991) qui sont des connexions unidirectionnelles implémentées avec des sockets, des événements X et des files de messages. La notion d'adresse a été structurée en différentes couches, distinguant un niveau "physique" et un niveau "logique". A chaque entité est attachée un scénario d'échanges de message.

La programmation a été effectuée en C++, ce qui nous a conduits à définir de nombreuses classes, et autant de couches de logiciel.

5. Résultats et discussion

Une évaluation systématique de la plateforme n'a pas été faite d'un point de vue ergonomique. Des tests généraux de premier niveau sont seulement présentés ici.

5.1. Le travail collaboratif

Il nous est apparu assez rapidement que les règles de la collaboration variaient selon les sessions, les membres, les groupes et même dépendaient des objets manipulés. Il semble qu'il soit nécessaire d'introduire des couches d'arbitrage selon ces niveaux. Pour ce faire, nous modifions dynamiquement, par exemple, les droits sur les objets en cours de session en ôtant systématiquement les droits en fin de session, et en refusant la démission d'un utilisateur qui possède encore des droits de propriété sur des objets Le fait d'avoir introduit des groupes durables nous a également obligés à prévoir des mécanismes de sauvegarde durables (Decouchant et al. 1988) pour les objets pour lesquels il se pose évidemment des problèmes de droit chaque fois qu'un groupe différent reprend une session de travail nouvelle.

5.2. La gestion des communications et l'arbitrage

Nous nous sommes aperçus qu'il serait utile d'introduire une notion d'opérateur attachée aux entités groupe, membre, session, objet, messagerie. Ces opérateurs permetraient de manipuler les differents droits alloués aux utilisateurs, les niveaux de sécurité, les mots de passe, etc.

Sont également apparus des problèmes de droits sur les objets : c'est ainsi que nous avons par exemple, décidé d'ôter systématiquement les verrous protecteurs en fin de session, et de refuser la démission d'un utilisateur qui possède encore des droits de propriétaire sur des objets.

5.3. Les performances

La vitesse de l'ensemble de la plateforme est encore insuffisante. Les causes de cette lenteur sont dûes principalement à :

• la lenteur de X-windows,
• la structuration en couches, qui semble cependant indispensable pour organiser et maintenir les logiciels,
• l'implementation en C++, qui amène à faire des opérations redondantes et rigides.

Le problème principal demeure de loin celui de l'attente active (dite “polling”) qui ralentit toute l'activité de la machine par le temps CPU qu'elle consomme, le plus souvent en pure perte.

Du côté de l’utilisateur, lorsque la machine et le réseau présentent une charge raisonnable, la vitesse apparente semble convenable. En effet, nos applications collaboratives ne sont pas des applications où la vitesse est un facteur réellement capital. Le travail s'effectue à vitesse humaine, entremélé de phases de réflexion et de discussion. Il est seulement nécessaire que le “feedback” sur les objets soit immédiat. Cela est assuré dans notre cas par le fait que les applications sont répliquées sur chaque poste de travail et que donc le “feedback” est pris en compte localement sans une inutile propagation de la commande le long du réseau. Pour un texte entré au clavier sur un poste le temps maximal de visibilité sur un autre poste ne dépasse pas 3 secondes dans le cas d’une charge importante.

6. Conclusion

Une plateforme collecticielle est nécessaire pour faire cohabiter plusieurs applications collecticielles sur une même structure informatique. Les points saillants de l’architecture de la plateforme collecticielle que nous avons mise au point sont :
• des applications semi-répliquées pour permettre un retour local immédiat des informations et des actions,
• une collecte centralisée des informations de haut niveau pour le contrôle de l’activité,
• une implémentation en agents (Hammainen et al., 1990), en séparant les fonctions selon les couches basses à hautes, allant du niveau de transport au niveau de la gestion de la collaboration,
• des services dédiés aux médias et aux communications,
• des agents spécifiques et génériques — comme l’agent de communication qui a été conçu à partir de spécifications très générales qui le rend “réutilisable” (e développeur n’a plus en s’en préoccuper pour intégrer de nouvelles applications dans la plateforme).

La mise en œuvre effective d'applications dans cet environnement a été faite et a montré des performances acceptables côté utilisateur. Le temps de réponse sera amélioré dans le futur grâce aux réseaux à haut-débit. Cette architecture est, dans l’état de l’art actuel, bien adaptée aux systèmes interactifs et multimodaux.

7. Références

(Andrews, 1991) G.R. Andrews, Paradigms for Process Interaction in Distributed Programs, ACM computing Surveys, vol. 23 , No 1, 1991.

(Arons, 1992) B. Arons, Tools for Building Asynchronous Servers to Support Speech and Audio Applications, Proceedings of ACM Symposium on User Interface Software and Technology, California, november 1992, pp. 71-78.

(Banino et al., 1989) J.S. Banino and J.C. Fabre, Distributed Coupled Actors : A CHORUS proposal for reliability, In IEEE 3rd International Conference on Distributed Computing, 1989, pp 128-134.

(Biel, 1991) V. von Biel, Groupware grows up. MacUser, june 1991, pp. 207-211.

(Bourguet, 1991) M.L. Bourguet, Dialogue Multimodal pour la construction de plans architecturaux, Thèse INPG, Grenoble, 1991.

(Caelen et al., 1991) J. Caelen and al., Final report of the Multiworks Project, ESPRIT, 1991.

(Caelen et al., 1992a) J. Caelen, Ph. Garcin, J. Wretö, E. Reynier, Interaction multimodale autour de l'application ICPdraw. Bulletin de la Communication Parlée n°2, 1992, pp. 141-151.

(Caelen et al. 1992b) J. Caelen, J. Coutaz, Interaction homme-machine multimodale : quelques problème. Bulletin de la communication parlée n°2, pp. 125-140.

(Cheriton et al. 1988) D.R Cheriton, The V Distributed System, Comm. ACM, vol.31, No. 3, March 1988, pp. 314-33.

(Croisy et al., 1992) P. Croisy et A. Dericke, Un modèle conceptuel de système interactif pour une application coopérative. Actes du colloque IHM'92, Télécom Paris éd., 1992, pp. 151-156.

(Decouchant et al., 1988) D. Decouchant, A. Duda, A. Freyssinet, M. Riveill, X. Rousset de Pina, R Scioville et G. Vendôme, GUIDE : an implementation of the Commandos object-oriented architecture on UNIX. Proc. of EUUG Autumn Conference, oct. 1988, Lisbon, pp. 181-193.

(Duby et al., 1992) J.C. Duby et B. David, Collecticiel. Compte-rendu des ateliers, Actes IHM'92, Télécom Paris éd., 1992, pp. 59-88.

(Ellis et al. 1991) C.A. Ellis, S.J. Gibbs and G.L. Rein, Groupware : some issues and experiences. Communications of the ACM, 34(1), jan. 1991, pp. 38-58.

(Guimarlaes, 1992) N.M. Guimar\aes, and al., Programming Time in Multimedia User Interfaces, Proceedings of ACM Symposium on User Interface Software and Technology, California, november 1992, pp. 125-134.

(Hammainen et al., 1990) H. Hammainen, J. Alasuvanto and R. Mantyla, Experiences on Semi-Autonomous Agents, Maamans 1989, Departement of Computer Science, Helsinki University of Technology, Espoo, Finland , 1990.

(Kabré, 1991) H. kabré, Distributed Multimodal Applications : LEM system, First report, Institut de la Communication Parlée Grenoble, 1991.

(Mayer, 1991) N.P. Mayer, The WINTERP Widget Interpreter; A lisp prototypind and Extension Environment for OSF-Motif Based Applications and User Interface, lisp pointers, ACM SIGPLAN, vol. IV, No 1, 1991, pp. 45-60.

(Narasimhalu, 1991) A.D. Narasimhalu, Multimedia Information System, The Unfolding of a Reality, Com. ACM, October 1991.

(Pankoke-Babatz, 1989) U. Pankoke-Babatz, Computer based group communication, the AMIGO activity model. Ellis Horwood ed., 1989.

(Reichbuach, 1992) J.D. Reichbuach, R.A. Kemmerer, SoundWorks : An Object-Oriented Distributed System for Digital Sound, Com. ACM, March 1992.

(Stefik et al., 1987) M. Stefik, D. Bobrow, S. Forster, D. Tatar, WYSIWIS : Early experiences with multi-user interfaces. ACM trans. on Office Information System, vol. 5, No 2, april 1987, pp. 147-167.

(Ye et al. 1990) H. Ye and J. Caelen, Duration Constraints for Speech Input Interface in the MULTIWORKS Project, Internationnal Conference on Spoken Language, Processing. Nov. 1990, Kobe, Japan, pp. 18-22.

(Zarmer, 1992) C.L. Zarmer and C. Chew, Frameworks for interactive, extensible, information intensinve applications, Proceedings of ACM Symposium on User Interface Software and Technology, California, november 1992, pp. 33-41.