CoPAC, notre modèle d'architecture pour les systèmes multi-utilisateurs

C'est dans les vieilles marmites
qu'on fait les meilleures soupes.

Dicton populaire

CoPAC, notre modèle d'architecture
pour les systèmes multi-utilisateurs

7.1. Introduction

Notre présentation au chapitre précédent des modèles d'architecture pour les systèmes multi-utilisateurs nous a montré plusieurs insuffisances. Vis-à-vis de notre espace problème, aucun modèle n'apporte une solution couvrant toutes les dimensions. Vis-à-vis du modèle du trèfle du chapitre 1, les modèles tendent à privilégier une des trois dimensions du trèfle, souvent l'espace de production, au détriment des autres. Nous nous fixons donc comme objectif dans ce chapitre l'élaboration d'un modèle d'architecture qui intègre harmonieusement les trois dimensions du trèfle et qui réponde le plus largement possible aux dimensions de notre espace-problème des modèles d'architecture. Comme les interfaces multi-utilisateurs sont aussi en un sens des interfaces utilisateur classiques, nous partons de modèles proposés pour les systèmes mono-utilisateurs. Nous verrons que cette approche nous permet de préserver les propriétés de ces modèles tout en les étendant aux systèmes multi-utilisateurs. Nous présentons d'abord PAC et PAC-Amodeus qui servent de base à notre modèle et nous justifions ce choix. Nous présentons ensuite CoPAC, extension du modèle PAC-Amodeus aux systèmes multi-utilisateurs. Puis nous détaillons plusieurs exemples de mise en œuvre du modèle. Chaque exemple met en avant une dimension particulière du trèfle et nous permet de vérifier la validité du modèle pour chacune des composantes de l'espace fonctionnel des systèmes multi-utilisateurs.

7.2. Nos modèle de base : PAC, Arch et PAC-Amodeus

Notre modèle CoPAC repose sur le modèle PAC-Amodeus, lui-même issu des modèles PAC et Arch. Ces trois modèles d'architecture sont adaptés aux systèmes mono-utilisateurs. Nous avons ainsi voulu profiter des acquis, en particulier afin de bénéficier des propriétés de ces modèles. Nous présentons maintenant brièvement ces trois modèles et les propriétés de qualité du logiciel qu'ils permettent d'assurer.

7.2.1. Le modèle PAC

Le modèle PAC est un modèle multi-agent pour la conception logicielle des systèmes interactifs [Coutaz 1987]. Il repose sur deux principes directeurs : le concept d'agents réactifs à facettes, et l'organisation hiérarchique de ces agents.

Un agent réactif est un système de traitement de l'information. Il est constitué d'un ensemble d'opérations, de mécanismes d'entrée/sortie et d'un état interne. A la différence des agents cognitifs de l'intelligence artificielle, un agent réactif n'a pas à un but qu'il cherche à satisfaire. Il réagit à des stimuli, génère des réponses et modifie son état en

fonction de ces stimuli. Un agent PAC est constitué de trois facettes, chacune ayant une compétence particulière (figure 7.1).

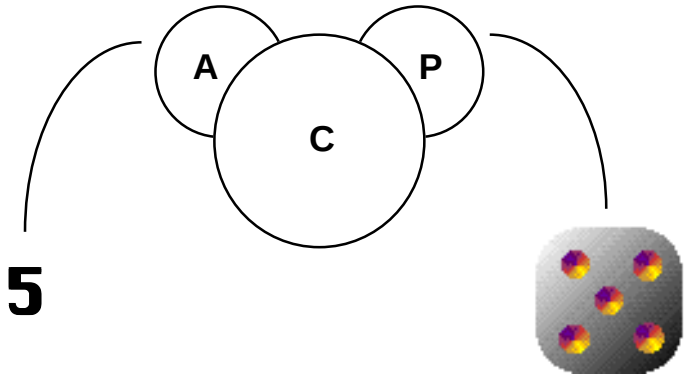

Figure 7.1. Un agent PAC. L'agent gère ici un dé à jouer. Son abstraction (A) maintient la valeur courante sous forme d'un entier. Sa présentation (P) gère la représentation visible du dé. Dans ce cas il s'agit d'une image stockée en ressource et représentant une face du dé. Le contrôleur de dialogue (C) assure la traduction de formalismes entre A (un entier, valeur du dé) et P (un entier, numéro de ressource de l'image à afficher).

- La facette P, Présentation, définit le comportement perceptible de l'agent pour l'utilisateur. Cette facette gère à la fois les communications de l'utilisateur vers l'agent (entrées) et de l'agent vers l'utilisateur (sorties).

- La facette A, Abstraction, définit la compétence propre de l'agent, indépendamment de toute considération de présentation.

- La facette C, Contrôle, a deux rôles : il exprime les dépendances et assure la traduction de formalismes entre les facettes A et P. Il gère aussi la communication de l'agent avec son environnement, c'est-à-dire d'autres agents PAC.

Un système interactif est constitué d'un ensemble d'agents PAC organisés en une hiérarchie. Cette hiérarchie traduit des relations de coopération dynamique entre les agents et reflète un continuum de niveaux d'abstraction depuis le noyau fonctionnel jusqu'aux éléments d'interaction (figure 7.2).

[Nigay 1994] identifie trois propriétés de qualité du logiciel que l'utilisation du modèle PAC permet de satisfaire : l'indépendance entre le noyau fonctionnel et la présentation qui est un précepte fondamental de la conception d'interfaces, la réutilisabilité des composants de l'interface grâce à l'indépendance des facettes P et A, la modifiabilité indispensable à la conception itérative des systèmes interactifs. Notons aussi que la granularité fine des agents PAC apporte deux avantages : elle permet une conception

logicielle détaillée et il est possible de réfléchir de façon précise à la modifiabilité et à la réutilisabilité.

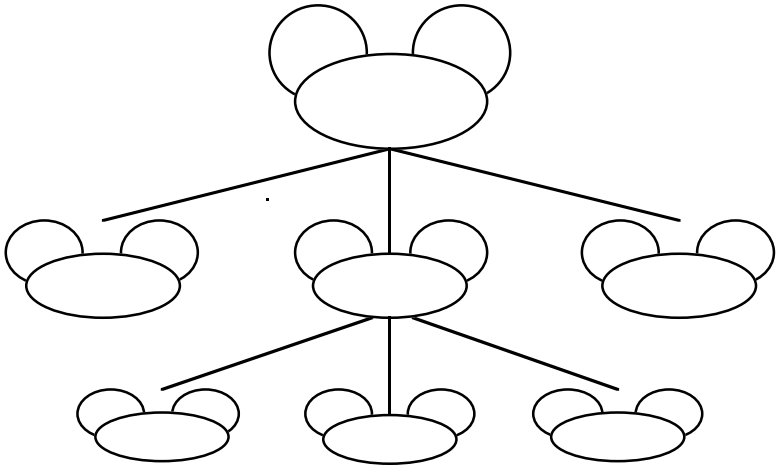

Figure 7.2. Une hiérarchie d'agents PAC. Les agents communiquent par leur composant Contrôle.

7.2.2. Le modèle Arch

Le modèle Arch [Bass 1992] est un affinement du classique modèle de Seeheim. Il distingue cinq composants organisés sous forme d'une arche : noyau fonctionnel, adaptateur de domaine, contrôleur de dialogue, présentation et interaction (figure 7.3).

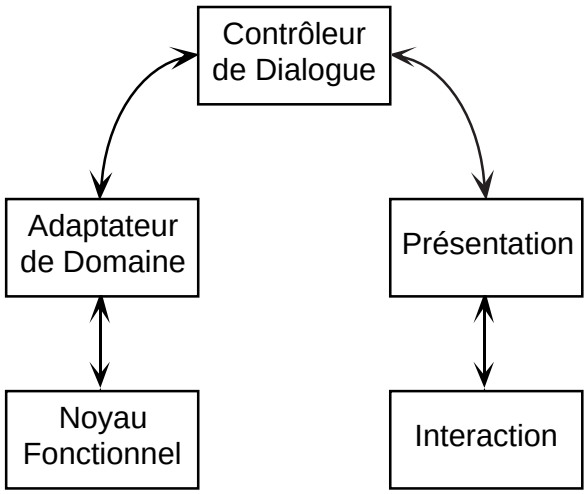

Figure 7.3. Le modèle Arch.

- Le noyau fonctionnel implémente les concepts du domaine.

- Le composant interaction est en contact direct avec l'utilisateur et peut être mis en œuvre grâce à une boîte à outils.

- Le contrôleur de dialogue est la clé de voûte du modèle : il gère l'enchaînement des tâches et assure le lien entre les composants présentation et adaptateur de domaine.

- L'adaptateur de domaine établit un pont entre le contrôleur de dialogue et le noyau fonctionnel. Ce composant permet d'ajuster les différences de modélisation des objets conceptuels entre les deux composants qui l'entourent.

- Le composant présentation a un rôle similaire à l'adaptateur de domaine pour l'autre branche de l'arche : il permet de définir une boîte à outils virtuelle qui est concrétisée par le composant interaction.

Nous retenons deux points forts du modèle Arch : la présence de l'adaptateur de domaine et le composant présentation.

L'*adaptateur de domaine* joue un rôle de médiateur entre le contrôleur de dialogue et le noyau fonctionnel. En tant que tel, il implémente un protocole de communication entre ces deux composants. Un tel protocole est caractérisé par une stratégie temporelle et la nature des données échangées. La *stratégie temporelle* définit les règles de coordination entre les deux entités communicantes : la communication peut être synchrone, asynchrone ou alterner entre ces deux modes. Dans le mode synchrone, la communication est du style requête-réponse. De ce fait, du point de vue de l'utilisateur, le dialogue est soit à fil unique, soit dans le meilleur des cas à fils multiples entrelacés. Dans le mode asynchrone en revanche, le dialogue à fils multiples parallèle est possible. Ces deux modes correspondent donc à deux interprétations différentes de la propriété de dialogue à fils multiples. Notons que pour implémenter le mode asynchrone, il est nécessaire que le noyau fonctionnel et le contrôleur de dialogue soient implémentés dans deux processus séparés, ou au minimum, qu'ils soient exécutés dans deux *threads* distincts d'un même processus. Dans le cas multi-utilisateur, cette stratégie temporelle prend une nouvelle importance comme nous le verrons au paragraphe 7.3.1. Les *données échangées* via l'adaptateur de domaine sont des objets du domaine de l'application, connus par le noyau fonctionnel. Si la nature de ces objets n'est pas adaptée à leur présentation à l'utilisateur, l'adaptateur du noyau fonctionnel permet d'enrichir la sémantique de ces objets de façon à les faire correspondre aux attentes de l'utilisateur. Il peut s'agir de combiner plusieurs structures du noyau fonctionnel en un unique objet du domaine, ou au contraire de segmenter un concept du noyau fonctionnel en un ensemble d'objets du domaine. Cette notion de délégation sémantique confiée à l'adaptateur du noyau fonctionnel, dont nous avons déjà parlé au chapitre 6, participe à la satisfaction de deux propriétés : la

configurabilité pour laquelle on peut envisager différents types de délégation sémantique adaptés aux différents rôles, et l'observabilité publiée. Les filtres requis pour la satisfaction de l'observabilité publiée sont typiquement implémentés par l'adaptateur du noyau fonctionnel.

Le *composant présentation* garantit l'indépendance du contrôleur de dialogue vis-à-vis du composant interaction qui recouvre la boîte à outils de la plate-forme d'accueil. Ce composant permet de définir une boîte à outils virtuelle qui permet d'assurer la propriété de portabilité. Certains outils, comme les squelettes d'application intègrent explicitement ce composant et masquent au réalisateur les particularités de la plate-forme d'accueil. Nous avons dit toutefois au chapitre 4 que la satisfaction de la propriété de portabilité impose des contraintes au réalisateur. Elle exige souvent l'utilisation exclusive des services qui constituent le plus petit dénominateur commun entre les plates-formes d'accueil et interdit l'exploitation de toute la richesse d'une plate-forme donnée. Avec le composant présentation, il est cependant possible d'enrichir une boîte à outils suivant un mécanisme analogue à la délégation sémantique pour l'adaptateur du noyau fonctionnel.

Le modèle Arch est un modèle de référence : avec le métamodèle Slinky qui permet de faire varier l'importance relative de ses cinq composants, il peut être adapté aux contraintes d'un environnement particulier. Un des intérêts du modèle Arch consiste en la présence des composants adaptateur de domaine et présentation qui permettent de garantir la satisfaction de propriétés.

7.2.3. Le modèle PAC-Amodeus

PAC-Amodeus combine la décomposition fonctionnelle canonique du modèle Arch et affine le contrôleur de dialogue en agents PAC [Nigay 1994] (figure 7.4).

Dans PAC-Amodeus, le contrôleur de dialogue d'Arch est organisé en une hiérarchie d'agents PAC. Chaque agent PAC établit plusieurs liaisons :

- la facette A de chaque agent est en relation avec un objet du domaine situé dans l'Adaptateur du Noyau Fonctionnel (ANF, équivalent de l'adaptateur de domaine de Arch). La facette P d'un agent pointe sur un objet de présentation dans le Composant Techniques de Présentation (CTP, équivalent du composant présentation de Arch). Comme dans PAC, le contrôle de chaque agent réalise la traduction entre le objets de présentation et les objets du domaine.

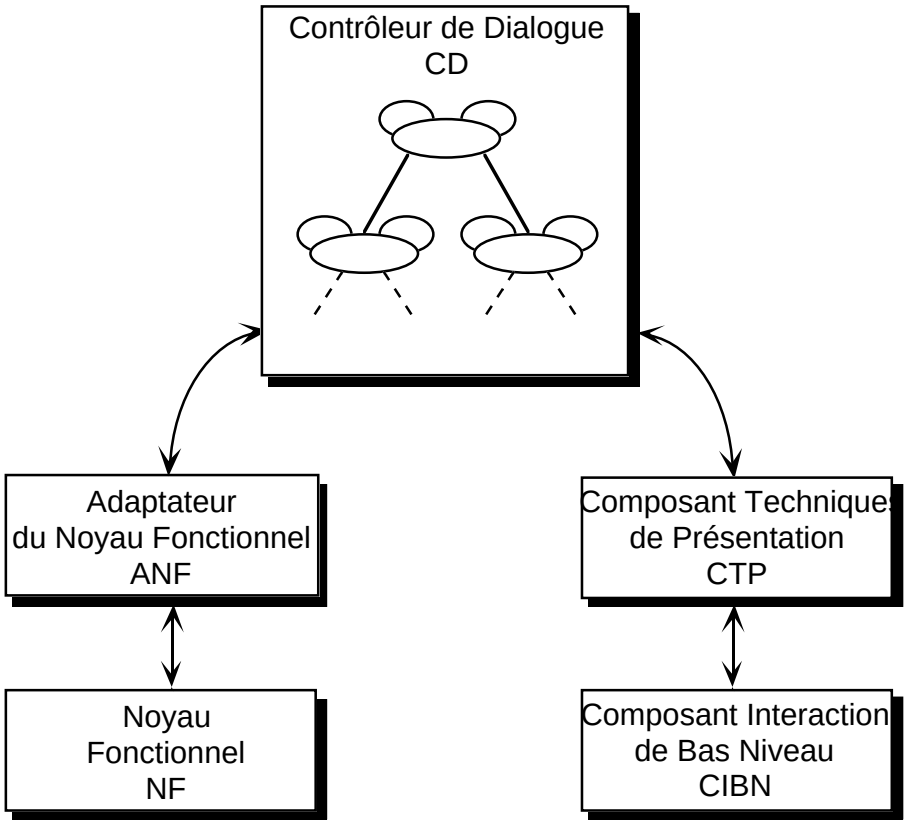

Figure 7.4. Le modèle PAC-Amodeus. Reposant sur la structure du modèle Arch, il affine le contrôleur de dialogue en une hiérarchie d'agents PAC.

- Chaque agent est relié à des agents fils et parents dans la hiérarchie. Comme dans PAC, la hiérarchie définit les relations de coopération entre agents. Cette décomposition appliquée au Contrôleur de Dialogue (CD) de Arch est particulièrement adaptée au rôle d'enchaînement des tâches du CD. On peut identifier dans la hiérarchie d'agents des grappes d'agents qui auront la charge d'une tâche donnée. La composition de ces grappes d'agents alliée au fonctionnement indépendant des agents de la hiérarchie permet d'implémenter les relations entre tâches comme le parallélisme ou l'entrelacement. Le modèle PAC-Amodeus permet ainsi de garantir la satisfaction de la propriété de dialogue à fils multiples.

Le modèle PAC-Amodeus conserve les propriétés intrinsèques du modèle Arch comme la portabilité et la décomposition fonctionnelle qu'il préconise. Grâce à la décomposition du Contrôleur de Dialogue en agents PAC, il permet aussi de satisfaire les propriétés de qualité du logiciel de modifiabilité et réutilisabilité, ainsi que les propriétés d'utilisabilité comme le dialogue à fils multiples.

7.3. Le modèle CoPAC

CoPAC est l'extension du modèle PAC-Amodeus aux systèmes multi-utilisateurs. Il augmente à la fois les agents PAC et le modèle Arch. CoPAC est un modèle d'architecture conceptuel et nous verrons comment il répond aux différentes dimensions de l'espace-problème que nous avons présenté au chapitre précédent. CoPAC étend PAC-Amodeus de deux façons :

- il prend en compte les systèmes multi-utilisateurs en répondant à l'espace-problème des modèles d'architecture des systèmes multi-utilisateurs. Il apporte aussi des réponses aux caractéristiques requises d'un modèle d'architecture pour les systèmes multi-utilisateurs énoncées au chapitre précédent,

- Il introduit de nouveaux composants et connecteurs pour répondre aux demandes particulières des services de communication. En effet, ces services exigent de manipuler de nouveaux types de données, les flots de médias continus, pour lesquels les modèles multi-agents n'ont pas de solution explicite.

7.3.1. Le partage des composants dans CoPAC

Un moyen simple de prendre en compte les systèmes multi-utilisateurs avec PAC-Amodeus pourrait consister à répliquer le modèle pour chaque utilisateur et à introduire des mécanismes de synchronisation pour garantir par exemple la cohérence des données. Mais cette approche mène à une duplication inutile de composants logiciels et à des difficultés pour identifier les mécanismes de synchronisation.

L'approche que nous avons choisie consiste à donner la possibilité de partager de différentes façons chacun des cinq composants du modèle Arch sur lequel repose PAC-Amodeus. Plusieurs cas sont envisageables : chacun des composants peut être découplé, couplé, ou commun. La figure 7.5 résume les différents modes de partage d'un composant de l'arche.

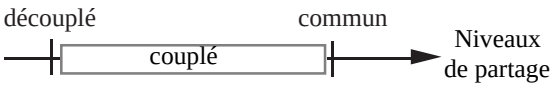

Figure 7.5. Les différents niveaux de partage d'un composant d'Arch dans le modèle CoPAC. Entre le niveaux découplé et le niveau commun, on trouve un continuum de possibilités de couplage.

Un composant du modèle est mis en *commun* lorsqu'il est partagé par plusieurs utilisateurs. C'est par exemple le cas du noyau fonctionnel pour les éditeurs partagés : tous les utilisateurs manipulent le même document et le noyau fonctionnel qui gère ce

document est alors partagé. On retrouve ici l'approche d'ALV [Hill 1992]. Cependant, la granularité plus fine des composants de Arch permet une plus grande souplesse. L'adaptateur du noyau fonctionnel (ANF) peut être chargé de gérer différentes politiques d'accès au document en fonction de différents rôles. Par exemple, si l'on doit implémenter deux rôles, lecteur et rédacteur, on utilisera deux composants ANF, chacun implémentant l'un des rôles. Tous les utilisateurs ayant un rôle de lecteur partageront l'ANF correspondant. Les utilisateurs rédacteurs partageront l'ANF rédacteur. Les rôles doivent aussi être pris en compte au niveau du contrôleur de dialogue mais celui-ci peut être mis en commun entre différents rôles. Cette configuration permet en particulier la modification dynamique des rôles en cours de session par la reconfiguration de la partie de l'arche contenant l'ANF et la modification des connexions entre composants, à l'initiative du contrôleur de dialogue. Cette possibilité permet de satisfaire la propriété de liaison dynamique entre le noyau fonctionnel et l'interface énoncée par [Dewan 1992]. La stratégie temporelle, que nous avons définie au paragraphe 7.2.2, doit être ici considérée. Elle définit la façon dont l'ANF fait communiquer le noyau fonctionnel et le contrôleur de dialogue. Lorsque l'ANF est mis en commun entre plusieurs utilisateurs, la stratégie temporelle choisie a une influence sur la façon dont les utilisateurs peuvent accéder au noyau fonctionnel. Le mode de communication synchrone impose des accès séquentiels au noyau fonctionnel, tandis qu'un mode asynchrone permet des accès parallèles. Cet aspect doit être examiné en liaison avec les compétences du noyau fonctionnel, en particulier afin d'éviter le classique problème de l'interblocage. Si le noyau fonctionnel inclut des mécanismes de contrôle d'accès, l'ANF peut utiliser un mode de communication asynchrone. Dans le cas contraire, par exemple lorsque l'on veut réutiliser le noyau fonctionnel d'un système mono-utilisateur, la délégation sémantique permet de prendre en compte le contrôle d'accès dans l'ANF.

A l'autre extrême de l'arche, un composant comme le composant interaction de bas niveau (CIBN) peut aussi être partagé : dans le mode souris partagée de Timbuktu par exemple, les utilisateurs partagent le même pointeur. La mise en commun du CIBN est un moyen simple d'assurer le partage des dispositifs physiques. La mise en commun d'éléments d'interaction, tels des formulaires ou des formes graphiques peut être réalisé en mettant en commun le composant techniques de présentation (CTP). Dans ce cas cependant, on pourra souhaiter affiner à un niveau de détail plus précis. Un modèle tel que SLICE [Karsenty 1994] fournit une décomposition plus fine que le couple CIBN/CTP et pourrait être utilisé pour structurer plus précisément cette partie de l'arche. La conception logicielle des applications multi-utilisateurs permettant la manipulation directe et le partage fin d'éléments d'interaction bénéficierait particulièrement de la combinaison de SLICE et CoPAC.

Il faut remarquer que lorsqu'un composant est mis en commun dans l'architecture conceptuelle, cela ne se traduit pas nécessairement par un composant unique dans l'implémentation. Bien que l'existence d'un composant unique dans l'implémentation soit justifiée pour le noyau fonctionnel (comme dans le modèle centralisé), une telle approche risque de poser des problèmes de performance pour le CIBN. Dans l'implémentation, le CIBN peut en fait être répliqué et la synchronisation entre les différents CIBN peut être assurée par le partage d'états entre les CIBN. Cette approche ressemble au modèle à états partagés de Patterson que nous avons présenté au chapitre précédent.

Les composants d'un niveau de l'arche sont *couplés* s'ils doivent maintenir une forme de cohérence entre eux. La cohérence requise ici est plus lâche que dans les cas précédents et ne nécessite pas une mise en commun comme précédemment. Un bon exemple en est la rétroaction de groupe, lorsque les actions d'un utilisateur sur un élément d'interaction doivent être répercutées vers les autres utilisateurs. Par exemple, dans l'éditeur partagé SASSE [Baecker 1992], chaque utilisateur peut suivre la position des autres utilisateurs dans le document partagé grâce à un ensemble de scrollbars. Chacune des scrollbars reflète la position d'un autre utilisateur et chaque utilisateur ne peut déplacer que sa propre scrollbar. Si l'effet de la manipulation des scrollbars est partagé, la manipulation des scrollbars ne l'est pas. Les scrollbars ne sont partagées que pour l'affichage et pas pour la manipulation. Il s'agit de ce que nous appelons cohérence lâche d'un élément d'interaction. Dans ce cas, il n'est pas nécessaire de mettre en commun le composant CTP mais les différents composants CTP doivent quand même respecter un couplage. La modification de la position d'une scrollbar implique la répercussion de cette modification sur les "images" de cette scrollbar chez les autres utilisateurs. Dans ce cas, la cohérence lâche est assurée par l'intermédiaire de l'agent PAC du contrôleur de dialogue en charge de la gestion de la scrollbar. Nous verrons au paragraphe suivant comment cet agent PAC répercute son changement d'état vers ses "cousins", c'est-à-dire les agents PAC des autres utilisateurs qui gèrent l'affichage de l'image de la scrollbar concernée. Notons qu'ici encore, il s'agit d'une synchronisation par partage d'état.

Enfin, les composants de différents utilisateurs peuvent être *découplés*. C'est le cas lorsqu'il n'y a aucun partage et que les composants sont indépendants. Les composants sont répliqués et ne communiquent pas. Par exemple dans un éditeur graphique partagé, le document est géré par un noyau fonctionnel commun comme expliqué plus haut, mais les composants CIBN sont découplés : chaque utilisateur a une copie locale du CIBN.

Par rapport à notre espace-problème des modèles d'architecture, ce mécanisme de partage permet de considérer la dimension distribution et en partie la dimension responsabilité. Le partage permet en effet de réfléchir à la réplication des composants et aux éventuels

mécanismes de synchronisation qu'ils requièrent. Toutefois la granularité du partage pourra être modulée suivant les possibilités du système d'exploitation utilisé. En particulier un système d'exploitation à objets permettant un partage fin nécessitera d'affiner les parties des composants qui doivent être partagées. La dimension responsabilité considérée au niveau de granularité des utilisateurs correspond à la notion de rôle. Dans ce cas, le partage ou le découplage de composants et en particulier de l'adaptateur du noyau fonctionnel, comme nous l'avons vu, permet d'implémenter différents rôles.

Les stratégies de partage et de réplication de composants exposées dans ce paragraphe nous permettent aussi de répondre à la règle ❶ sur la factorisation que nous avons énoncée au paragraphe 6.3 du chapitre 6. Le modèle CoPAC permet d'identifier les composants répliqués afin de factoriser la réalisation.

Nous allons maintenant détailler la façon dont le modèle CoPAC permet à différents composants sur différents sites de communiquer et la façon dont il intègre les services de communication homme-homme médiatisée.

7.3.2. La communication dans CoPAC

Deux aspects de la communication importants pour les systèmes multi-utilisateurs sont pris en compte par le modèle CoPAC :

- la communication entre composants, que ceux-ci soient mis en commun et répliqués, ou qu'ils doivent maintenir une cohérence lâche. En général, ces composants communiquent en synchronisant leurs états.

- La facette communication du modèle du trèfle du chapitre 1, c'est-à-dire la communication homme-homme médiatisée. Dans ce cas, l'architecture doit pouvoir intégrer des flots de données correspondant aux médias continus comme le son et la vidéo.

7.3.2.1. Communication entre composants

En règle générale dans CoPAC, la communication entre composants s'effectue par l'intermédiaire du contrôleur du dialogue (CD). En effet, le CD, constitué d'une hiérarchie d'agents PAC, est le seul composant de l'architecture dans lequel on peut identifier précisément l'élément qui doit communiquer : c'est un agent PAC et nous verrons que nous distinguons dans ce cas une nouvelle facette dans un agent PAC pour gérer la communication. Cette règle permet de localiser précisément les éléments communicants et permet de conserver la modularité du modèle PAC-Amodeus. Pour

prendre en compte la coordination entre les rôles, un contrôleur de dialogue peut être privilégié. Dans ce cas, les autres contrôleurs lui sont subordonnés. Nous en verrons un exemple avec la mise en œuvre du jeu $(421)^n$. La partie du contrôleur de dialogue qui gère la coordination entre les rôles peut aussi être isolée dans un "super-contrôleur de dialogue" qui contrôle tous les autres contrôleurs de dialogue. Nous détaillerons l'exemple du super-contrôleur de dialogue de NEIMO au paragraphe 7.4.3.

Cependant, dans certains cas et principalement pour des raisons de performance (c'est là la règle de "validité vis-à-vis du monde réel" qui joue), des composants de l'arche de CoPAC peuvent communiquer directement entre eux sans passer par le contrôleur de dialogue. C'est le cas par exemple lorsque le CIBN est mis en commun comme dans Timbuktu. Dans les autres cas, la communication passe par un agent PAC et nous distinguons alors explicitement dans l'agent une facette communication, COM, comme le montre la figure 7.6. Lorsque la communication est établie directement entre composants, le facette COM joue un rôle légèrement différent que nous expliquons plus loin.

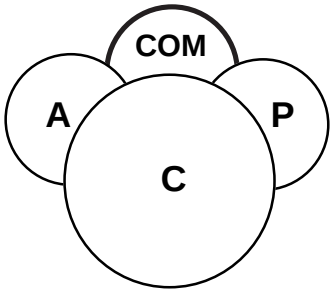

Figure 7.6. Les agents du modèle CoPAC. La facette communication (COM) rajoutée à l'agent PAC a la charge de la communication avec les agents d'autres sites.

La nouvelle facette, COM, est exactement sur le même plan que les facettes A et P. De même que la facette A d'un agent PAC communique avec l'ANF du modèle PAC-Amodeus et la facette P est en relation avec le CTP, la facette COM est en relation avec un nouveau composant spécialisé dans la communication (figure 7.7). De même que le contrôle (C) assure la communication entre les facettes A et P, c'est encore le contrôle qui a la charge des échanges entre COM et les facettes A et P. En règle générale, une modification de la facette P (respectivement A) sera répercutée par C à la fois vers A (respectivement P) et COM. L'arrivée d'un message en provenance d'un autre site sur la facette COM sera répercutée à la fois vers A et P par le contrôle.

Le modèle CoPAC ne détaille pas le contenu du Composant Communication avec lequel la facette COM est en relation. Ceci permet de préserver la diversité des mécanismes de communication existants. Toutefois, il existe quelques règles simples pour structurer le Composant Communication. Avec la plupart des systèmes d'exploitation, le Composant

Communication s'appuiera sur les services de communication fournis par le système (par exemple, les sockets ou les Apple Events [Apple 1991]). Si nécessaire, on pourra envisager une structuration classique en couches comme le modèle OSI pour organiser le Composant Communication. Certains mécanismes de communication inter-applications nécessitent toutefois une approche particulière. C'est par exemple le cas du protocole de communication des Apple Events : en effet, les messages inter-applications envoyés sous forme d'Apple Events sont reçus par la queue d'événements de l'application, c'est-à-dire dans le CIBN. Dans ce cas particulier, le Composant Communication communique avec le CIBN. Nous verrons des exemples de cette mise en œuvre au paragraphe 7.4.

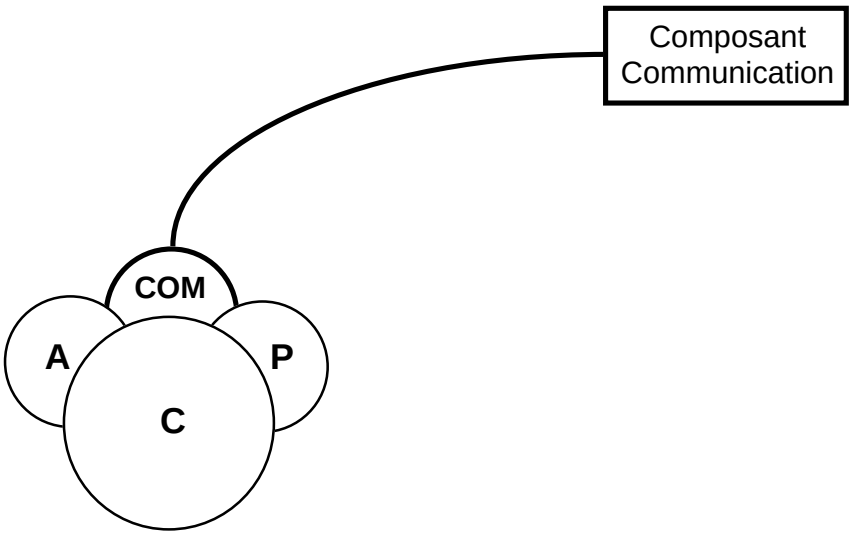

Figure 7.7. La facette COM de l'agent PAC communique avec les autres sites via le Composant Communication.

Notons que l'idée de rajouter des facettes spécialisées à un agent PAC n'est pas une idée nouvelle. Cette approche permet de détailler la structure d'un agent PAC et participe au respect des propriétés de modifiabilité et de réutilisabilité. Par exemple, [Ouadou 1994] avec le modèle AMF propose d'affiner les agents PAC dans le cas d'une application mono-utilisateur en rajoutant des facettes gérant l'aide à l'utilisateur ou les erreurs. La particularité de la facette COM que nous proposons est qu'elle réalise un service qui ne peut, au contraire de l'aide ou des erreurs, être pris en charge par les autres facettes. Ainsi la facette COM ne précise pas seulement la structure de l'agent PAC, mais elle lui rajoute une compétence qui n'était pas requise pour les systèmes mono-utilisateurs.

7.3.2.2. Communication homme-homme médiatisée

Le second type de communication modélisé par CoPAC correspond à la communication homme-homme médiatisée via des médias continus comme l'audio ou la vidéo. Cette communication impose de considérer un nouveau type de données : les *flots*. Les flots sont particuliers à plusieurs titres : ils sont produits de façon constante par une *source*

(par exemple une caméra vidéo) et sont consommés par un *puits* (par exemple une région d'affichage sur un écran). Leur nature continue pose problème dans une architecture multi-agents : en effet, les agents savent bien manipuler des informations discrètes, sous formes de messages ou d'événements. Mais ils ne peuvent manipuler des données continues sans les tronçonner. Pour assurer la propriété de régularité des flots, il est nécessaire d'envisager des communications directes entres sources et puits, comme nous l'avons vu au chapitre précédent avec le modèle Gemma. Ces communications directes court-circuitent les agents et peuvent donc être établies directement entre des composants de l'architecture, typiquement les Composants Techniques de Présentation. Mais la mise en relation a toujours lieu à l'initiative d'un Contrôleur de Dialogue d'un site partie prenante de la communication. Plus précisément, un agent du Contrôleur de Dialogue, via sa facette COM, peut ouvrir et fermer une communication et gérer l'aspect contrôle du flot de médias continus. Cet aspect contrôle inclut en particulier la gestion de la qualité de service (QoS) du réseau utilisé. Via sa facette abstraction et en faisant appel au noyau fonctionnel, l'agent peut aussi effectuer des traitements sur les médias continus reçus ou envoyés. Un traitement peut être par exemple un retournement d'image pour satisfaire la propriété de réversibilité vidéo. Nous verrons un exemple de mise en œuvre au paragraphe 7.4.2 avec le mediaspace VideoPort. Notons enfin que pour tenir compte de la diversité des outils de communication existants, on peut observer un phénomène analogue au "slinky" du modèle Arch : suivant la plate-forme d'accueil, l'importance relative du Composant Communication et du Composant Techniques de Présentation doit être modulée. Par exemple, pour utiliser un outil de communication qui fait circuler une partie des informations de contrôle dans le flot de médias continus, il convient de donner une compétence accrue au Composant Techniques de Présentation au détriment du Composant Communication.

En résumé, la communication dans le modèle CoPAC est prise en compte de deux façons. La première repose sur la facette COM qui communique avec le Composant Communication. Cette première possibilité est adaptée à l'échange de messages entre agents situés sur différents sites. La communication proprement dite s'appuie sur les compétences du Composant Communication qui reposent sur les possibilités de la plate-forme d'accueil. Le deuxième type de communication est adapté aux médias continus utilisés principalement dans la communication homme-homme médiatisée. Dans ce cas, la communication distingue le transfert d'informations de contrôle et le transfert des médias continus. Le transfert des flots de médias est réalisé par le Composant Techniques de Présentation sous le contrôle d'un agent. Les informations de contrôle sont échangées entre les agents via la facette COM et le Composant Communication.

7.3.3. Évaluation du modèle CoPAC

Après la présentation du modèle d'architecture CoPAC, il nous faut maintenant l'évaluer vis-à-vis de notre espace problème et des caractéristiques attendues d'une architecture que nous avons présentées au chapitre précédent.

L'espace problème que nous avons proposé comporte cinq dimensions : services, responsabilité, niveaux d'abstraction, parallélisme, distribution. En ce qui concerne les services, le modèle CoPAC les considère au niveau conceptuel, tels que définis par le modèle du trèfle du chapitre 1. L'espace de production est à la charge du noyau fonctionnel. L'espace de coordination est géré principalement par le contrôleur de dialogue. La communication est prise en compte par le composant communication. Nous avons déjà remarqué au chapitre 6 ces équivalences entre les composantes de l'espace fonctionnel du trèfle et les composants du modèle Arch pour la production et la coordination. Avec CoPAC, nous avons montré que cette équivalence pouvait être étendue aux systèmes multi-utilisateurs, et qu'un nouveau composant chargé de la communication permet de couvrir la facette communication du trèfle. Nous verrons plus précisément dans les exemples de mise en œuvre comment CoPAC prend en compte chacune des composantes de notre espace fonctionnel. Pour la dimension responsabilité, CoPAC bénéficie de l'apport de Arch pour la délégation entre composants fonctionnels. Nous avons notamment mentionné la délégation sémantique au niveau de l'adaptateur du noyau fonctionnel. Nous avons aussi vu que l'adaptateur du noyau fonctionnel permet d'implémenter différents rôles et que la stratégie de partage contribue également à la dimension responsabilité. En ce qui concerne les niveaux d'abstraction, là encore, CoPAC exploite les niveaux d'abstraction présents dans Arch. Cependant, CoPAC ne propose pas de structuration en niveaux d'abstraction pour le composant communication, à part la séparation entre communication de contrôle et communication de médias continus. Mais il est possible d'utiliser un modèle classique comme le modèle OSI pour organiser ce composant. Pour la dimension parallélisme, deux aspects interviennent : la structuration du contrôleur de dialogue en une hiérarchie d'agents PAC fonctionnant de façon indépendante et les stratégies temporelles que nous avons expliquées en 7.2.2 et 7.3.1 permettent toutes deux de considérer la dimension parallélisme. Enfin nous avons vu que la stratégie de partage permet de réfléchir à la dimension distribution.

La satisfaction des caractéristiques attendues d'une architecture est également assurée, à une nuance près. Les quatre premières règles, la modifiabilité, la réduction de la complexité, la décomposition du travail et la règle de validité vis-à-vis du monde réel sont satisfaites principalement grâce au modèle PAC-Amodeus, sous-jacent dans CoPAC. La règle de validité vis-à-vis du monde réel a fait l'objet d'une attention particulière pour le

composant communication. Elle a en particulier inspiré la séparation entre messages de contrôle et médias continus. Elle a aussi conduit à ne pas préconiser une structuration du composant communication pour tenir compte de la diversité des outils de communication. Les deux autres règles que nous nous sommes fixées concernent la factorisation de la réalisation des composants et la prise en compte du facteur d'échelle. Nous avons vu au paragraphe 7.3.1 que la première règle est vérifiée. En revanche, il nous faut être prudents avec la règle de facteur d'échelle. Nos expérimentations avec le modèle n'ont concerné jusqu'à présent que des petits groupes d'utilisateurs (4-5 personnes) et nous manquons ici d'expériences à plus grande échelle.

En résumé, CoPAC couvre de façon satisfaisante les dimensions de notre espace problème. On note en particulier qu'il intègre de façon équilibrée les trois facettes de notre espace fonctionnel du trèfle. Pour la satisfaction des caractéristiques attendues d'une architecture, CoPAC bénéficie de l'apport du modèle PAC-Amodeus et garantit aussi la possibilité de factoriser la réalisation. En revanche, nous manquons d'éléments pour apprécier sa validité au-delà d'un petit groupe d'utilisateurs.

7.4. Exemples de mise en œuvre de CoPAC

Nous présentons maintenant trois exemples de mise en œuvre du modèle d'architecture CoPAC. Le premier exemple, un jeu multi-utilisateur, met l'accent sur la synchronisation. Le second, le mediaspace numérique VideoPort, privilégie la facette communication. Enfin le troisième exemple d'application, la plate-forme d'utilisabilité NEIMO que nous avons présentée au chapitre 4, est plus orientée vers la production, mais inclut aussi des composantes synchronisation et communication importantes. Tous ces exemples ont été réalisés sur Macintosh et nous verrons comment nous avons tenu compte des particularités de cet environnement.

7.4.1. $(421)^n$, un jeu de 421 multi-utilisateur

Les jeux sont traditionnellement un terrain privilégié pour les systèmes multi-utilisateurs. Non seulement ils sont un exemple d'application pertinent pour ces systèmes car ils permettent de sortir du classique paradigme du "jouer contre l'ordinateur", mais ils constituent un mode d'interaction sociale particulier. Un jeu repose en effet sur des règles écrites et connues par tous les joueurs, même s'il accepte des variantes. Le contrôle des interactions entre les joueurs est ainsi clairement défini et est approprié à une mise en œuvre informatique. La facette coordination du modèle du trèfle est privilégiée. Notons toutefois que la mise en œuvre d'un jeu sous forme informatique conduit souvent à perdre un élément social important : tricher n'est plus possible. Par exemple, dans notre jeu de 421, il n'est pas possible de passer le tour de quelqu'un "par inadvertance" ou de

modifier subrepticement la valeur d'un dé. On retrouve ici une caractéristique du contrôle technique par rapport au contrôle social : une plus grande rigidité et l'impossibilité de transgresser les règles intégrées au système.

Le jeu de 421 est un classique des bistros français et se joue avec trois dés et un nombre illimité de joueurs. Le but est de faire sortir des combinaisons de dés, la plus recherchée étant 421 (un dé sur 4, un autre sur 2, le dernier sur 1). Chaque joueur joue à tour de rôle. Les dés sont lancés tous en même temps au premier lancer, le joueur a droit à deux lancers supplémentaires et peut reprendre tout ou partie des dés jetés à chaque lancer. L'interface de notre réalisation informatique du jeu de 421 est présentée figure 7.8.

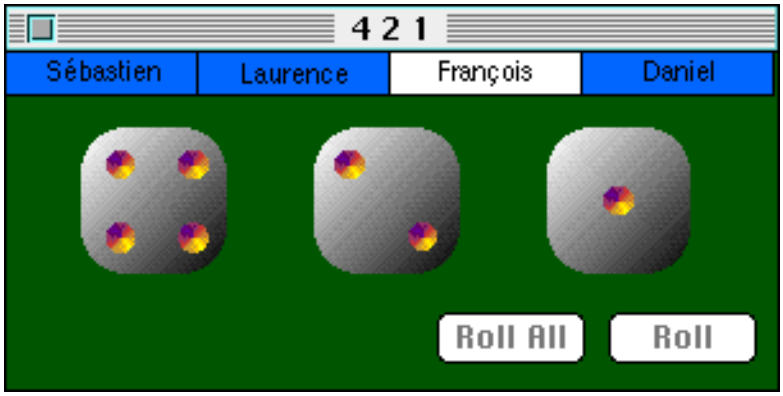

Figure 7.8. L'interface de $(421)^n$. Les trois dés viennent d'être lancés et sont affichés au centre de la fenêtre. Les noms des joueurs sont affichés en haut et le nom du joueur qui a la main (ici, François), est mis en évidence. Les boutons en bas à droite permettent de lancer tous les dés ou uniquement les dés sélectionnés. Lorsqu'un autre joueur a la main (comme ici), ces boutons sont grisés.

L'architecture choisie privilégie la réplication de composants. Deux rôles sont identifiés : un joueur est l'hôte, les autres joueurs sont les invités. L'architecture correspondante est présentée figure 7.9.

Le noyau fonctionnel (NF) est répliqué pour tous les joueurs : il maintient la liste des joueurs. Cette liste est affichée sur tous les sites dans le bandeau visible sur la figure 7.8. Lorsqu'un nouveau joueur se présente, il établit une connexion avec le joueur hôte et l'état du NF hôte est transmis au NF invité. Les rôles sont distingués grâce à deux ANF différents. L'ANF hôte permet la modification de la liste des joueurs, l'ANF invité ne permet l'accès à la liste des joueurs qu'en mode lecture. Chaque modification de l'état du NF hôte est répercuté vers les NF invités. Ce sont les contrôleurs (eux aussi répliqués) qui gèrent la coordination des différents joueurs. Le joueur dont c'est le tour de jouer est marqué par le NF hôte dans la liste des joueurs. Lorsqu'un NF voit son état modifié, il transmet son changement au contrôleur de dialogue qui vérifie si c'est le tour de son site. Dans ce cas, il active les boutons de jeu et donne la main au joueur.

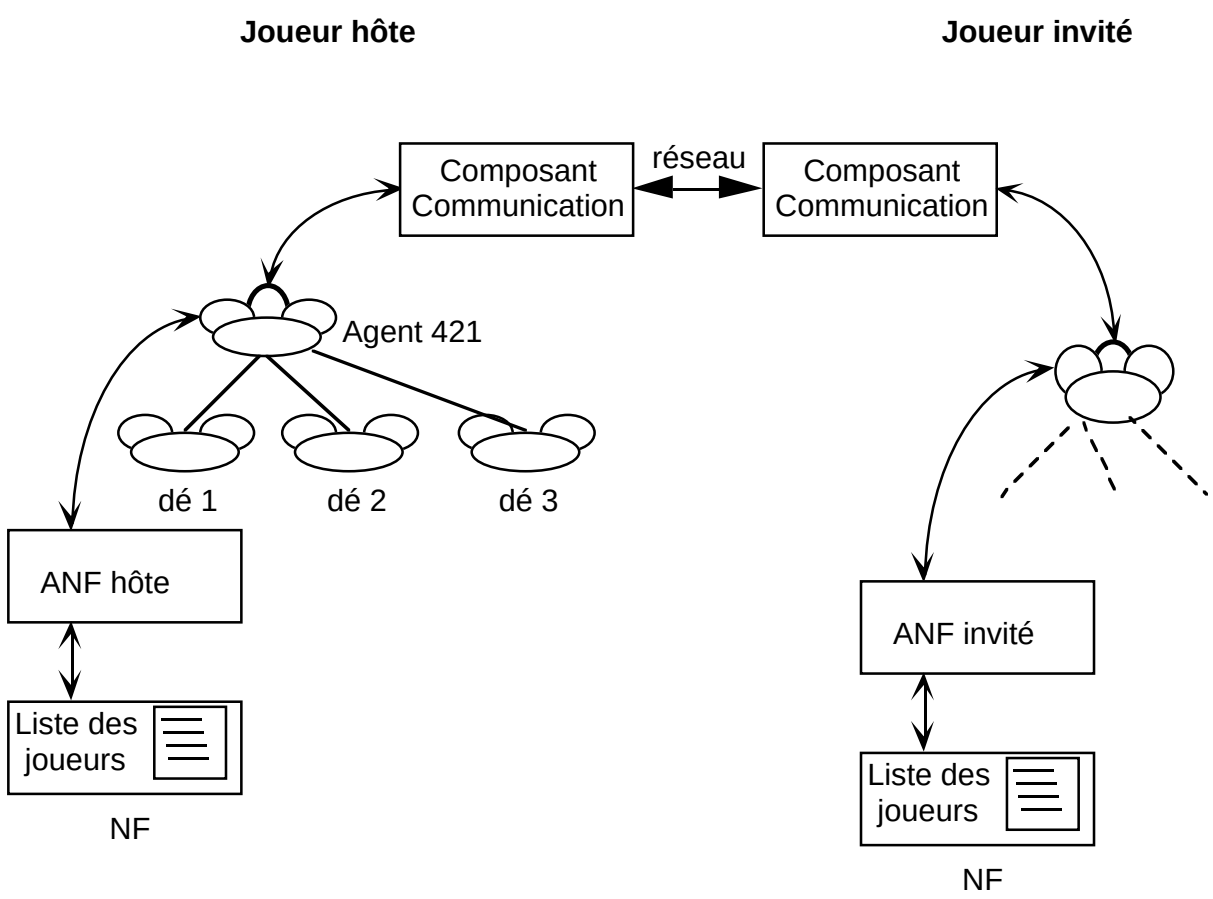

Figure 7.9. Architecture logicielle du jeu $(421)^n$. Deux sites sont représentés. L'un des joueurs (à gauche) joue le rôle d'hôte, les autres celui d'invités.

Cet exemple montre que l'on peut assurer la coordination avec le modèle CoPAC à l'aide du contrôleur de dialogue. Il montre aussi que les ANF permettent d'implémenter simplement différents rôles avec le même NF.

7.4.2. VideoPort, un mediaspace numérique

VideoPort est notre mediaspace que nous avons déjà mentionné au chapitre 2. Cette réalisation nous a permis d'étudier la mise en œuvre de la communication de médias continus dans le modèle CoPAC. L'interface utilisateur est très simple : les connexions vidéo sont affichées dans une fenêtre, un menu permet d'ouvrir et de fermer une connexion et de modifier des réglages comme le choix du son joué lors de la réception d'un appel (figure 7.10). Dans les premières versions dont nous présentons ici l'architecture, le seul média de communication implémenté est la vidéo. En effet, le son est plus exigeant en termes de propriété de régularité des flots et la première version qui ne traitait pas explicitement les médias continus n'était pas à même de gérer le son. Le son est maintenant en cours d'intégration dans la seconde version dont nous présentons aussi

l'architecture, grâce à la bibliothèque UserLink de gestion des médias continus (présentée au chapitre 8).

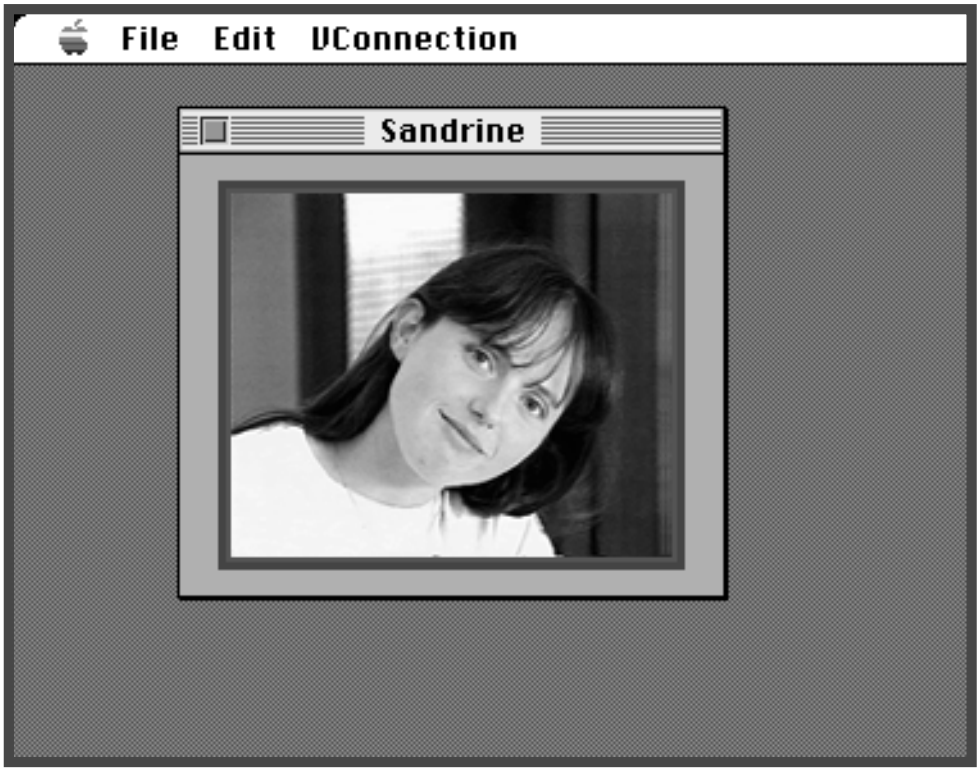

Figure 7.10. Copie d'écran de l'interface de VideoPort.

Deux versions successives de VideoPort ont été réalisées, toutes deux conservant la même interface utilisateur. La première version utilise comme mécanisme de communication les Apple Events. Il s'agit d'une communication reposant sur des messages qui sont indifféremment des messages de contrôle ou qui contiennent des données comme par exemple des images.

La figure 7.11 montre une vue partielle de l'architecture de deux applications VideoPort en cours de communication. Pour alléger le schéma, nous avons conservé de l'architecture uniquement les composants significatifs : le composant communication, le composant techniques de présentation (CTP, représenté pour un seul des deux sites) et, à l'intérieur du contrôleur de dialogue, les agents VP qui gèrent les fenêtres comme celle montrée figure 7.10.

L'acquisition vidéo se fait au niveau du CIBN. Dans cette première version de VideoPort, nous ne transmettons pas un flot vidéo, mais juste des images successives. Dans notre cas, nous utilisons en fait la boîte à outils QuickTime qui fournit une interface standardisée et qui permet de s'affranchir des particularités du dispositif d'acquisition

vidéo. Nous considérons donc que QuickTime nous fournit une surcouche garantissant l'indépendance vis-à-vis du matériel vidéo et l'acquisition se fait donc au niveau du CTP et non du CIBN. Notons qu'il s'agit là d'un cas typique d'application du métamodèle "slinky" de Arch, tenant compte des particularités de la plate-forme d'accueil. La facette P de l'agent VP (pour VideoPort) acquiert l'image capturée et la transmet, via le contrôleur de dialogue, à la facette COM. Si l'on voulait afficher cette image "en local", par exemple pour fournir une vue miroir (au sens du chapitre 4), la facette P ferait appel au CTP pour l'affichage de l'image dans une fenêtre.

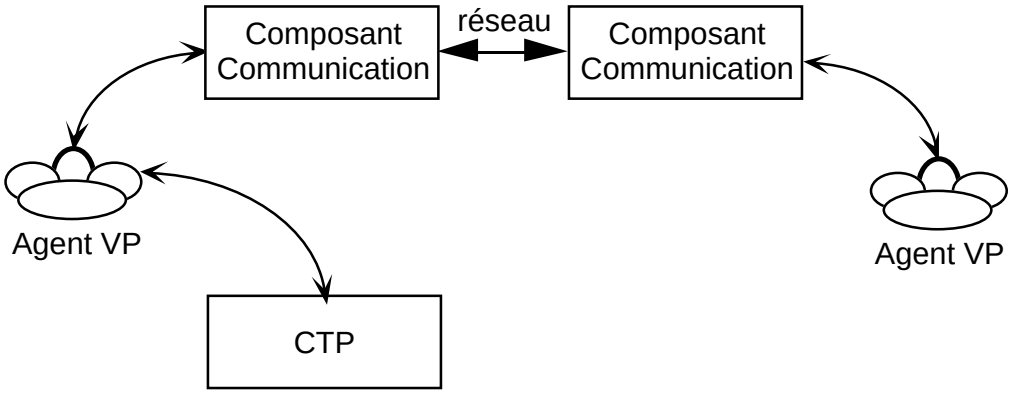

Figure 7.11. Architecture logicielle de VideoPort version Apple Events. Chaque site dispose d'un agent VP. Un agent VP gère une fenêtre comme celle de la figure 7.10.

La facette COM transmet l'image au composant communication qui se charge de l'encapsuler dans un Apple Event. Le composant communication se charge ensuite de l'envoi de l'Apple Event à travers le réseau vers l'autre site VideoPort.

Le mécanisme de réception est analogue. Un Apple Event est reçu par le CIBN (non représenté pour ne pas alourdir le dessin), comme déjà expliqué pour l'exemple de $(421)^n$. Il est immédiatement transmis au composant communication qui en extrait l'information pertinente, c'est-à-dire l'image reçue, et la transmet à la facette COM de l'agent VP. Celle-ci est alors transmise à la facette P par le contrôle et P la transmet au CTP pour affichage.

Nous n'avons pas parlé du rôle de la facette A de l'agent VP. Son rôle est en fait extrêmement réduit : elle se contente de conserver le titre de la fenêtre, c'est-à-dire le nom du correspondant. Mais elle pourrait être le lieu de traitements sur l'image, par exemple un retournement pour garantir la propriété de réversibilité vidéo.

La deuxième version de VideoPort vise à intégrer les médias continus. En effet nous avons vu que la première version ne gère pas des flots vidéo, mais juste des images

successives et qui sont traitées séquentiellement. La gestion des flots vidéo a été rendue possible par l'utilisation de la bibliothèque de communication UserLink décrite au chapitre suivant. Cette évolution de VideoPort a été aussi l'occasion de vérifier les propriétés de l'architecture en suivant les recommandations de la méthode SAAM (Software Architecture Analysis Method) [Kazman 1994]. Cette méthode recommande notamment de choisir une tâche concrète et d'éprouver les propriétés de l'architecture dans le cadre de la réalisation de cette tâche. Cette tâche est typiquement une modification ou une extension du logiciel. Dans notre cas, il s'agissait de modifier le moyen de communication utilisé dans VideoPort : de l'envoi d'images successives par Apple Events, nous voulions passer aux médias continus fournis par la bibliothèque UserLink. La propriété de modifiabilité a donc pu être étudiée.

L'architecture de la deuxième version de VideoPort est présentée figure 7.12.

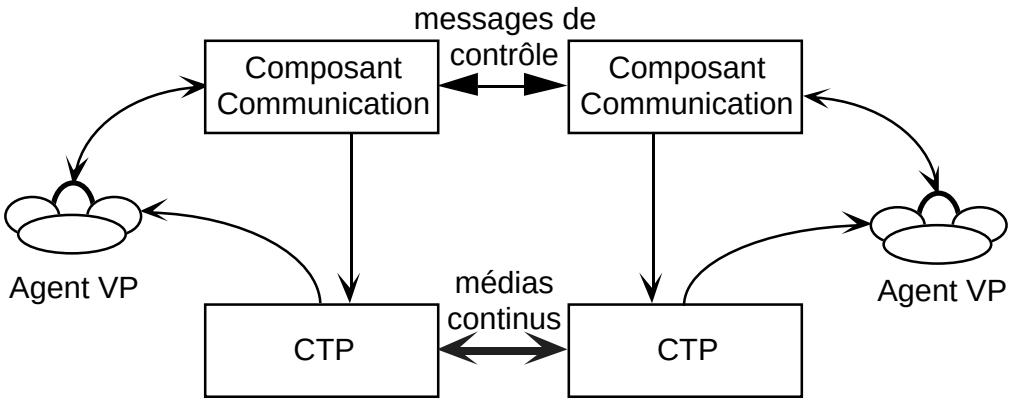

Figure 7.12. Architecture logicielle de VideoPort version UserLink. la communication a désormais lieu à deux niveaux dans l'architecture : les messages de contrôle au niveau du composant communication, les médias continus au niveau du CTP.

UserLink prend en charge plusieurs services qui étaient réalisés par l'architecture dans la première version : la communication, mais aussi l'acquisition vidéo et l'affichage. Le rôle de l'agent VP se voit donc réduit. En fait, dans cette deuxième version, son rôle consiste surtout à faire appel à UserLink pour contrôler le flot vidéo. Avec UserLink nous avons introduit une couche supplémentaire qui nous masque la plupart des services. Lors de l'établissement d'une connexion, l'agent VP contacte le composant communication, c'est-à-dire la partie contrôle de UserLink. UserLink établit ensuite la connexion entre les deux sites et instancie le flot vidéo entre les CTP. UserLink se charge aussi de l'acquisition du flot et de son affichage via le CTP.

Les modifications à réaliser ont pu être localisées précisément. Sur le schéma d'architecture de la figure 7.12, on note les modifications suivantes :

- la facette P d'un agent VP se contente de recevoir des informations depuis le CTP. En effet, l'affichage est géré par UserLink. Le lien du CTP vers l'agent VP ne fait que recopier l'affichage réalisé par UserLink vers la facette P de l'agent. Cette façon de faire permet de prévoir d'éventuels traitements qui devront être réalisés par l'agent, par exemple la réversibilité vidéo.

- Le composant communication contrôle directement le CTP. Ce contrôle a lieu au sein de UserLink.

En résumé, les modifications introduites par l'utilisation de UserLink ont été au nombre de cinq :

- suppression du code d'aiguillage des Apple Events dans le CIBN,

- suppression du code d'acquisition vidéo dans le CTP et des appels correspondants dans la facette P de l'agent VP,

- suppression du code d'affichage dans le CTP et des appels correspondants dans la facette P de l'agent VP,

- modification de l'établissement et de la rupture des connexions dans la facette COM de l'agent VP. Cette modification a été la plus importante à cause du changement du mode d'adressage des sites entre les Apple Events et UserLink qui utilise les adresses IP,

- modification du contrôle de dialogue. Il est maintenant réduit au contrôle de la facette COM.

L'architecture nous a ainsi permis de localiser précisément les modules à modifier. On peut donc considérer que la propriété de modifiabilité est vérifiée, au moins pour la partie communication du modèle CoPAC.

La mise en œuvre des deux versions successives de VideoPort nous a permis d'expérimenter divers moyens de communication pour la vidéo et d'éprouver le modèle CoPAC pour la communication. Il s'est révélé adapté aussi bien à la prise en compte de succession d'images que de flots vidéo en utilisant la bibliothèque spécialisée UserLink. De plus, l'évolution de VideoPort a permis de vérifier la propriété de modifiabilité de l'architecture prônée par le modèle.

7.4.3. NEIMO, notre plate-forme d'observation du comportement des utilisateurs

NEIMO est la plate-forme d'étude de l'utilisabilité présentée au chapitre 5. L'architecture globale de NEIMO est complexe et nous n'en donnerons qu'un aperçu en faisant ressortir les points intéressants dans la mise en œuvre de CoPAC.

L'architecture de NEIMO repose sur un serveur central, NeimoCom. La raison de ce choix est que les participants à une session d'expérimentation contribuent tous à la production d'un unique fichier historique de la session. D'autre part, tous les enregistrements de ce fichier sont datés. La nécessité d'avoir une horloge unique fiable a imposé la solution centralisée. Notons que des problèmes de performances nous amènent maintenant à considérer une solution où la capture est répartie. Mais cette nouvelle solution étant en cours de réalisation, nous nous intéressons ici à la solution centralisée. Chacun des rôles identifié dans NEIMO dispose d'une application locale qui lui est propre et se connecte à NeimoCom via une bibliothèque adaptée à son rôle. Ces bibliothèques jouent le rôle d'ANF et NeimoCom joue le rôle du NF (figure 7.13). Le principal service de NeimoCom vu comme NF est la capture vers le fichier historique.

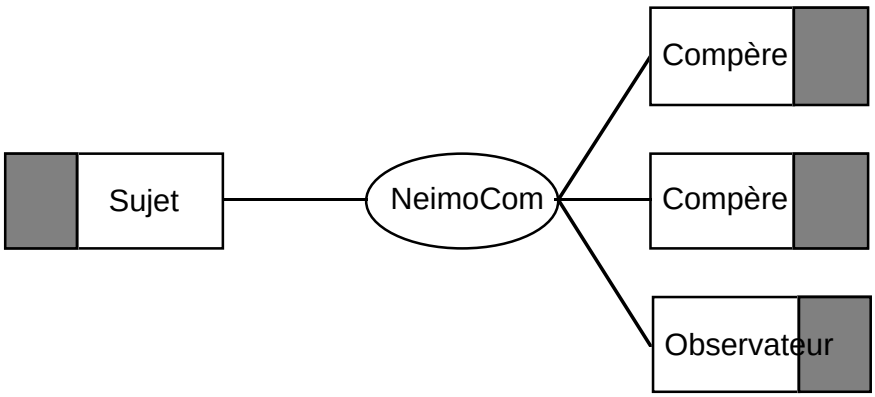

Figure 7.13. Architecture globale de NEIMO. Au centre le processus serveur central NeimoCom. A droite, trois applications : deux sont dédiées à un compère, l'autre à un observateur. A gauche, l'application du sujet. Les parties grisées représentent les interfaces et contrôleurs de dialogue locaux développés pour l'expérimentation considérée. Les rectangles blancs sont des bibliothèques d'accès à NeimoCom et jouent le rôle d'ANF vis-à-vis du NF NeimoCom.

Mais NeimoCom a aussi un rôle de contrôleur de dialogue global. A ce titre, cette architecture est proche de celle de Suite qui distingue aussi un contrôleur de dialogue global pour la coordination entre les rôles et des contrôleurs de dialogue locaux pour la gestion de l'interface utilisateur [Dewan 1992]. Ce contrôleur de dialogue est assez particulier : il n'établit pas une communication entre un noyau fonctionnel et une interface mais entre un noyau fonctionnel et un ensemble de contrôleurs de dialogue locaux, répartis sur différents sites. Il peut être vu comme un "super-contrôleur de dialogue"

auquel tous les contrôleurs de dialogue locaux sont subordonnés. La figure 7.14 détaille l'architecture du contrôleur de dialogue de NeimoCom pour un sujet et deux compères.

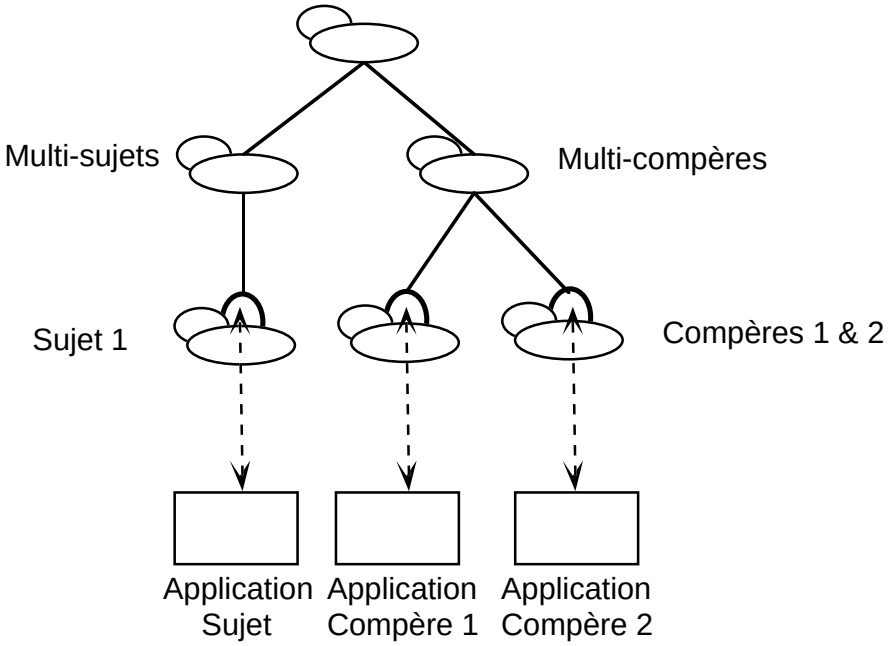

Figure 7.14. Architecture logicielle de NeimoCom. Les flèches pointillées indiquent des communications réseau. Sur la figure, seuls les agents du contrôleur de dialogue global sont représentés pour le cas d'un sujet et deux compères. Les agents multi-sujets et multi-compères gèrent l'enchaînement des tâches des sujets et des compères sous le contrôle de l'agent de plus haut niveau.

Les agents du super-contrôleur de dialogue (super-CD) ont une caractéristique remarquable : ils ne comportent pas de facette P. En effet, ils ne remplissent que des services de coordination, et l'interface de NEIMO ne permet pas d'agir sur la coordination. Celle-ci est "câblée" dans NeimoCom. Chaque sujet et chaque compère est représenté dans le super-CD par un agent qui communique avec le contrôleur de dialogue local du sujet ou du compère par sa facette COM. Les agents multi-sujets et multi-compères ainsi que l'agent de plus haut niveau gèrent la coordination entre les rôles des sujets et des compères. Par exemple, on peut considérer l'échange suivant : le sujet émet une commande qui nécessite une action de simulation de la part de chacun des compères (par exemple, un compère simule la reconnaissance d'une phrase dite par le sujet, et l'autre compère vérifie la cohérence de la réponse). L'agent multi-sujets reçoit la requête du sujet, et la transmet, via l'agent de plus haut niveau et l'agent multi-compères, aux deux compères. Le premier compère effectue l'action correspondant à la requête du sujet et sa réponse est transmise à l'agent compère 1, puis à l'agent compère 2 via le multi-compères. Le deuxième compère vérifie la cohérence de la réponse et la valide. L'agent compère 2 reçoit la validation et la transmet à l'agent multi-compères qui envoie alors la réponse à l'agent de plus haut niveau. Cette réponse est ensuite transmise au sujet.

Notons deux possibilités d'extension de l'exemple que nous avons présenté. L'agent multi-sujets permet potentiellement d'assurer la coordination entre plusieurs sujets, mais cette possibilité n'a pas encore été mise en œuvre. On peut envisager de fournir une interface pour contrôler la coordination : par exemple, le deuxième compère pourrait désactiver la validation. Dans ce cas, l'agent compère 2 aurait une facette P.

L'exemple de NeimoCom nous a permis de voir la conception d'un noyau fonctionnel centralisé et d'une coordination forte entre plusieurs rôles. Dans ce cas, un super-contrôleur de dialogue assure la coordination entre les rôles. En règle générale, les agents de ce super-CD sont regroupés sous un agent contrôle de plus haut niveau et les agents feuilles de la hiérarchie communiquent avec les contrôleurs de dialogue locaux via leur facette COM. Les agents du super-CD n'ont pas de facette P, sauf si l'interface permet d'agir sur la coordination.

7.5. Synthèse

Dans ce chapitre, nous avons présenté le modèle d'architecture CoPAC pour les systèmes multi-utilisateurs et nous avons détaillé des exemples de mise en œuvre. CoPAC repose sur le modèle PAC-Amodeus, lui-même composé de Arch et PAC. Il préserve les qualités de ces modèles quant à la structuration et la vérification de propriétés comme la modifiabilité, la portabilité et la réutilisabilité. Il répond aussi à l'espace problème des architectures des systèmes multi-utilisateurs et aux caractéristiques attendues d'une architecture, au moins pour des groupes de petite taille. CoPAC distingue plusieurs niveaux de partage des composants de Arch, et étend à la fois les agents PAC et le modèle Arch. Il rajoute aux agents PAC une facette COM pour gérer la communication et un composant communication au modèle Arch. L'ajout de ces nouveaux composants permet à CoPAC de prendre explicitement en compte les médias continus pour la communication homme-homme médiatisée.

Références

[Apple 1991] Apple. *The Apple Event Manager*, Apple Computer Inc., Cupertino, California, USA, 1991.

[Baecker 1992] R. M. Baecker, D. Nastos, L. R. Posner et K. L. Mawby. *The user-centred iterative design of collaborative writing software*, Workshop on Real Time Group Drawing and Writing Tools (CSCW'92, ACM Conference on Computer-Supported Cooperative Work), Toronto, Canada, 1992.

[Bass 1992] L. Bass, R. Little, R. Pellegrino, S. Reed, R. Seacord, S. Sheppard et M. R. Szczur. *The UIMS Tool Developers' Workshop: A Metamodel for the Runtime Architecture of an Interactive System*, in *SIGCHI Bulletin*, 24(1), janvier 1992. pp. 32-37.

[Coutaz 1987] J. Coutaz. *PAC, an Implementation Model for Dialog Design*, Interact'87, IFIP Conference on Human-Computer Interaction, Stuttgart, 1987. pp. 431-436.

[Dewan 1992] P. Dewan. *Principles of Designing Multi-User User Interfaces Development Environments*, IFIP TC2/WG2.7 Working Conference on Engineering for Human-Computer Interaction, Ellivuori, Finland, 1992. pp. 35-50.

[Hill 1992] R. D. Hill. *The Abstraction-Link-View Paradigm: Using Constraints to Connect User Interfaces to Applications*, ACM CHI'92 Conference on Human Factors in Computing Systems, Monterey, California, USA, 1992. pp. 335-342.

[Karsenty 1994] A. Karsenty. *GroupDesign : un collecticiel synchrone pour l'édition partagée de documents.* Thèse de doctorat, Université d'Orsay Paris-Sud, 1994.

[Kazman 1994] R. Kazman, L. Bass, G. Abowd et M. Webb. *SAAM: A Method for Analyzing the Properties of Software Architectures*, ICSE-16, Sorrento, Italie, 1994.

[Nigay 1994] L. Nigay. *Conception et réalisation des systèmes interactifs: Application aux Interfaces Multimodales.* Thèse de doctorat, Université Joseph Fourier Grenoble I, 1994.

[Ouadou 1994] K. E. Ouadou. *AMF : un modèle d'architecture multi-Agents Multi-Facettes pour interfaces homme-machine et les outils associés.* Thèse de Doctorat, Ecole Centrale de Lyon, 1994.