PAC^{+3} : un modèle d'architecture générique pour systèmes multi-utilisateurs

Gaëlle Calvary, Joëlle Coutaz, Laurence Nigay, Daniel Salber
CLIPS-IMAG, B.P. 53
38041 Grenoble Cedex 9, France
{gaelle.calvary, joelle.coutaz, laurence.nigay, daniel.salber}@imag.fr

RÉSUMÉ

Cet article a pour thème la modélisation d'architecture logicielle applicable aux systèmes multi-utilisateurs. Après une analyse synthétique des problèmes de la conception des architectures logicielles en général, nous abordons le cas particulier des systèmes multi-utilisateurs. L'analyse de l'état de l'art et l'identification des lacunes en la matière nous conduisent à une nouvelle proposition : PAC^{+3} . PAC^{+3} concilie, d'une part, la décomposition fonctionnelle du trèfle des collecticiels et d'autre part, l'indépendance entre les trois facettes du modèle PAC. Le modèle est illustré par un exemple fondé sur une extension imaginée au système SASSE.

MOTS CLÉS modèle d'architecture, collecticiel, communication, coordination, production, PAC, PAC-Amodeus, CoPAC, PAC^{+3} .

INTRODUCTION

L'étude formelle des aspects architecturaux des logiciels fait, depuis peu, l'objet de recherches actives. S'il est vrai qu'une maquette à jeter ne justifie pas une conception logicielle poussée, la maintenabilité et la modifiabilité de systèmes de plus en plus complexes ne peuvent plus se contenter d'une organisation logicielle artisanale : les choix architecturaux doivent être justifiés et s'appuyer sur un ou plusieurs modèles d'architecture idoines.

En Interaction Homme-Machine, la modélisation des architectures n'est pas un phénomène nouveau : le caractère itératif du processus de développement des interfaces utilisateur l'impose de facto. Si les modèles semblent bien formulés pour les interfaces graphiques mono-utilisateur, l'analyse de la pratique appelle deux remarques : 1) ces modèles occultent des problèmes généraux importants de l'activité de conception d'une architecture, 2) les propositions sont encore incertaines, imprécises ou mal comprises dès que le caractère "multi" entre en scène. C'est le cas notamment des interfaces multimodales et des systèmes multi-utilisateurs.

Après une analyse synthétique des problèmes de la conception des architectures logicielles en général, nous abordons dans cet article le cas particulier des systèmes multi-utilisateurs. L'analyse de l'état de l'art et les lacunes en la matière nous conduisent à une nouvelle proposition, PAC^{+3}, que nous illustrons sur une extension imaginée au système SASSE [1].

La conception d'architecture logicielle

L'usage commun assimile une architecture logicielle à un ensemble organisé d'entités de calcul (ou *composants*) dont les interactions sont médiatisées par des entités spécialisées (ou *connecteurs*). L'appel procédural, la communication par événements et les flots de données sont des exemples de connecteurs. Dans les modèles d'architecture, les composants sont généralement représentés par des boîtes tandis que les connecteurs sont dénotés par des flèches ou de simples traits. En vérité, une architecture revêt d'autres perspectives que cette simple décomposition structurelle. Chaque perspective est le reflet d'une activité particulière du processus de conception d'une architecture et traduit un aspect pertinent de la solution. Nous précisons ces étapes dans le paragraphe suivant puis nous analysons les problèmes laissés sans réponse.

Les activités du processus de conception d' une architecture

Le processus de conception d'une architecture logicielle recouvre au moins les activités suivantes, l'ordre de leur mise en œuvre étant laissé au savoir-faire du concepteur : définition de la décomposition fonctionnelle du système, identification de son organisation structurelle, allocation des fonctions à la structure et définition de la coordination entre les entités de la structure. Selon les cas, il faudra aussi se préoccuper de l'association entre entités structurelles et processus d'exécution du système, voire l'allocation des processus aux processeurs. Cette dernière activité prend tout son sens en informatique ubiquitaire avec notamment la migration dynamique d'agents sur le réseau.

Décomposition fonctionnelle. La décomposition fonctionnelle consiste à exprimer les besoins fonctionnels du système en des unités conceptuelles plus simples. Contre toute apparence, cette activité n'est pas aisée. Il faut des années d'expérience pour produire une décomposition fonctionnelle correcte, c'est-à-dire *conforme* à des critères précis. En Interaction Homme-Machine, nous disposons, avec les modèles

Seeheim et Arch, de décompositions fonctionnelles canoniques mais ces offres ne sont applicables telles quelles qu'aux systèmes mono-utilisateurs.

Organisation structurelle L'organisation structurelle, un agencement statique, souvent artisanal, de composants et de connecteurs, s'inspire au mieux d'un ou de plusieurs styles. Rappelons qu'un style est un véhicule générique qui aide à l'expression de solutions structurelles. Selon Garlan [7], il comprend un vocabulaire d'eléments conceptuels : par exemple, dans PAC, le concept d'agent et ses facettes fonctionnelles P, A, C et la communication par événements. Il impose des règles de configuration entre ces éléments : par exemple, les facettes P et A d'un agent PAC ne communiquent que par la facette C de cet agent. Il véhicule une sémantique qui donne un sens à la description structurelle. C'est cette sémantique, partagée par tous les acteurs du processus de développement, qui permet d'exploiter la description structurelle sans ambiguïté.

Allocation des fonctions à la structure L'allocation des fonctions à la structure consiste à associer les éléments de la décomposition fonctionnelle aux entités de la structure. Cette mise en correspondance n'est pas unique et nous conseillons de justifier explicitement les choix en faisant référence aux critères de conception retenus pour le système. Par exemple, dans PAC-Amodeus, les concepts du domaine peuvent être associés au noyau fonctionnel mais peuvent aussi, pour une rétro-action immédiate et sémantiquement informative, se diluer dans les facettes A des agents PAC qui peuplent le contrôleur de dialogue.

Définition de la coordination La définition de la coordination entre les entités de la structure décrit le comportement dynamique de l'architecture : typiquement, les conditions de création et de destruction des entités structurelles, le protocole de communication et notamment la synchronisation, voire la migration fonctionnelle entre les composants. Cette perspective est orthogonale à la définition structurelle. Toutes deux expriment ce qui est important dans un contexte de développement donné.

Problèmes ouverts

Parmi les problèmes que nous avons identifiés [3], nous en retenons trois : le maintien de la cohérence entre les multiples vues d'une architecture, la granularité et l'hétérogénéité. La granularité d'une architecture désigne le niveau d'affinement adéquat de la description. L'hétérogénéité dénote la cohabitation de plusieurs styles au sein d'une même architecture.

Granularité d'une architecture Une architecture doit, en tant que support au raisonnement, exprimer ce qui est important. Outre la structure du système, elle doit aussi dénoter des problèmes de mise en œuvre. En particulier, si la plate-forme ne fournit pas un service au niveau d'abstraction voulu, ce service doit impérativement figurer dans l'architecture proposée.

Hétérogénéité des architectures logicielles Shaw démontre que chaque style véhicule des propriétés logicielles spécifiques [13]. En conséquence, il convient de choisir le style en fonction des critères retenus pour le système. A l'évidence, le style "taille unique" qui remplirait tous les critères est illusoire, l'hétérogénéité étant un mal nécessaire : elle peut survenir au cours du processus de réification. Par exemple, un modèle logique de partage de données peut s'affiner à l'implémentation en mémoires locales assorties d'une communication par messages. Mais, l'hétérogénéité peut aussi apparaître au sein d'un même niveau de réification. Par exemple, PAC-Amodeus utilise le style PAC pour structurer le contrôleur de dialogue mais hérite aussi, d'une part, des styles de la décomposition structurelle du noyau fonctionnel et, d'autre part, des styles des outils choisis pour les aspects logique et physique de l'interaction. Plus généralement, les systèmes ouverts et la réutilisation rendent l'hétérogénéité incontournable.

La cohabitation de styles offre l'avantage de choisir le style le mieux adapté à telle ou telle entité de l'organisation structurelle. Inversement, l'hétérogénéité peut être la cause d'incompatibilités : incompatibilités entre les structures de contrôle, entre les représentations de données, entre les protocoles, etc. Une solution générale à ce type de problèmes reste à trouver. Pour le cas particulier de l'IHM, nous utilisons dans PAC-Amodeus les deux adaptateurs du modèle Arch comme points de résolution de ces conflits.

Ayant précisé les principes généraux et les problèmes qui sous-tendent le processus de conception des architectures logicielles, nous étudions maintenant les offres en systèmes multi-utilisateurs.

Architectures pour les systèmes multi-utilisateurs

La conception d'architecture pour systèmes multi-utilisateurs doit concilier les exigences des systèmes répartis mais aussi les propriétés générales de l'interaction homme-machine augmentées des requis du travail coopératif. La diversité des sources de savoir-faire, voire les tensions entre les pratiques des domaines impliqués dans la conception des collecticiels, expliquent la difficulté de l'entreprise. De fait, la littérature offre de nombreuses solutions mais la couverture et la généricité de ces propositions sont insuffisantes pour servir de modèles canoniques.

Couverture et généricité

Nous caractérisons la *couverture d'un modèle d'architecture* par sa capacité à véhiculer le trèfle fonctionnel des collecticiels [12]:

- l'espace de production (ou espace ontologique) recouvre les concepts et les opérations qui motivent l'activité de groupe (par exemple les concepts et opérations relatives à l'édition de documents ou au pilotage d'avion, etc.) ;
- l'espace de coordination modélise l'organisation des acteurs, leurs rôles, leurs droits, devoirs et

responsabilités, les procédures et protocoles à respecter vis-à-vis des entités de l'espace de production (par exemple, le chef a la responsabilité d'allouer telle opération d'édition à tel membre du groupe) ;

- l'espace de communication offre aux acteurs humains la possibilité d'échanger des messages dont le contenu sémantique échappe au système (typiquement par courrier électronique ou tunnel audio-vidéo).

La *généricité d'un modèle d'architecture* s'analyse par la faculté d'adaptation du modèle à une large variété de contraintes. Un modèle générique est instanciable au cas particulier et cette opération d'incarnation doit être justifiée. Dans notre revue de l'état de l'art, nous nous limitons aux modèles dont nous pressentons un fort potentiel de généricité. Nous les regroupons en deux catégories en utilisant le style comme critère de classification : d'une part, le style à agents avec ALV, le modèle de Croisy et CoPAC, et d'autre part, le style "machines abstraites en couche" avec SLICE et le modèle de Dewan.

Modèles génériques à agents

Dans ALV [5], chaque utilisateur dispose d'un composant interface qui lui est propre et adapté à son rôle ("Personal View"). Des composants de liaison, "Link", font le lien entre l'Abstraction partagée (Shared Abstraction) et chaque vue. Les Link maintiennent des systèmes de contraintes et propagent les modifications de l'un des deux composants vers l'autre. ALV vise les systèmes multi-utilisateurs synchrones construits autour d'un noyau fonctionnel centralisé. Les exemples d'utilisation d'ALV indiquent que la couverture fonctionnelle privilégiée du trèfle est l'espace de production avec manipulation directe et, dans une moindre mesure, l'espace de coordination. On relèvera qu'un Link ALV sert à exprimer toutes sortes de dépendances que celles-ci relèvent de la production ou de la coordination.

Croisy [4] propose un affinement du modèle ALV dans lequel le contrôleur de dialogue est explicite. Ce modèle distingue un composant "multi-dialogue" géré par un superviseur et composé de "micro-dialogues". Un micro-dialogue est dédié à l'abstraction et communique avec autant de micro-dialogues de présentation qu'il y a de vues. Afin d'assurer la rétroaction de groupe, les micro-dialogues communiquent directement entre eux. Mais, en l'absence d'un mécanisme d'abonnement comme celui de GroupKit, l'indépendance des micro-dialogues n'est assurée que s'ils sont tous identiques. La flexibilité du modèle risque donc d'être compromise. Par ailleurs, comme ALV et UDA [11], ce modèle cible les systèmes privilégiant l'espace de production.

CoPAC reprend les principes directeurs de PAC-Amodeus qu'il étend pour les besoins des systèmes multi-utilisateurs [12]. Il définit des règles de couplage entre les composants pairs distants de l'arche et peuple le contrôleur de dialogue d'agents CoPAC lorsque ceux-ci contribuent à l'espace de communication. Un agent CoPAC est un agent PAC enrichi d'une facette explicite dédiée à la communication. Cette facette est en liaison avec le service chargé des médias continus. A l'inverse des modèles considérés jusqu'ici, CoPAC vise à couvrir l'espace de communication.

Modèles génériques à couches

SLICE (Sharing Layers In Cooperative Editing) propose une structuration en sept couches□ document abstrait, représentation du document, manipulation directe, représentation des vues, manipulation des vues, manipulation indirecte, curseurs. Chacune des couches peut être partagée entre les utilisateurs de façon indépendante. Cette décomposition fournit un cadre de réflexion pour le partage d'informations des systèmes d'édition coopérative. SLICE est donc concerné principalement par l'espace de production.

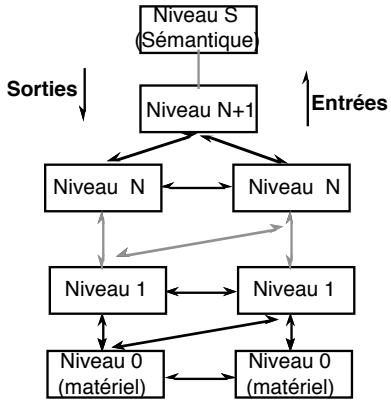

Figure 1 : Le modèle de Dewan.

Le modèle de Dewan [2] peut se voir comme une généralisation du modèle à "fermeture éclair" de Patterson [10]. Comme le montre la Figure 1, un système est découpé en niveaux d'abstraction allant du niveau le plus haut de nature sémantique, au niveau le plus bas dépendant du matériel. Certaines couches sont partagées et constituent la base du système (niveaux S à N+1) jusqu'à un point de branchement (niveau N+1) à partir duquel les couches sont répliquées sur chaque station utilisateur. La communication entre niveaux se fait par événements selon l'axe vertical entrée/sortie des niveaux d'abstraction mais aussi entre niveaux répliqués frères notamment pour la synchronisation d'état. Ce modèle offre un bon cadre de réflexion pour l'allocation des niveaux fonctionnels aux processus et permet de raisonner sur la granularité du parallélisme.

Analyse comparative entre style à agents et style en couches

Le style à agents offre deux mécanismes de généricité : l'affinement en niveaux d'abstraction et le multi-facettage.

1) Un style à agents permet au concepteur d'affiner l'organisation structurelle de l'architecture au niveau d'abstraction voulu et de jouer de manière rationnelle sur l'allocation (et la migration) des fonctions au bon niveau de granularité. Le problème pour le concepteur est d'identifier le

niveau d'abstraction idoine.

2) Le multi-facettage fonctionnel peut être exploité à façon en fonction de la pertinence des problèmes à traiter. Dans l'absolu, peu importe la nature des facettes de tel ou tel modèle pourvu qu'elles reflètent ce qui est important et qu'elles permettent de raisonner sur les choix de conception. Ainsi, dans le modèle CoPAC, les agents PAC impliqués dans l'espace de communication sont augmentés d'une facette de communication qui rend explicite le problème complexe des connexions entre postes de travail distants. De même, dans AMF [9], les agents PAC sont augmentés de facettes pour exprimer l'aide contextuelle et la capture d'événements en vue d'étudier l'utilisabilité du système. En résumé, il ne suffit pas d'affirmer que les facettes d'un modèle à agents sont "meilleures" que celles de tel autre modèle mais il faut s'interroger sur leur adéquation au problème posé.

Si une modélisation systématique d'un système en termes d'agents peut se voir comme un élément simplificateur, elle suppose un milieu homogène. Or, dans le cas général, l'hétérogénéité est un fait dont il faut tenir compte. Sur ce point, le style à couches est plus souple, chaque couche pouvant faire usage d'un sous-style spécifique. Dans le style à couches, la généricité tient au concept de couches dont la nature et le nombre peuvent être adaptés au cas à traiter. Selon le cas, une couche peut être vue comme un niveau d'abstraction comme dans le modèle de Dewan, ou bien comme un service qui se superpose aux autres comme dans SLICE.

En résumé, les modèles d'architecture décrits ici offrent un bon pouvoir de généricité mais présentent deux inconvénients : leur homogénéité et leur couverture fonctionnelle partielle du trèfle des collecticiels. Dans PAC^{+3}, nous comblons ces deux lacunes tout en restant génériques.

Le modèle PAC^{+3}

Pour satisfaire au critère de généricité, PAC^{+3} s'appuie sur PAC-Amodeus. Augmenté du trèfle des collecticiels, pour l'aspect couverture fonctionnelle, il est aussi inspiré du modèle de Dewan pour raisonner sur le partage et la réplication.

Généricité, hétérogénéité et PAC-Amodeus

Le fondement de PAC^{+3} est PAC-Amodeus [8] dont la généricité a fait ses preuves pour la conception logicielle d'interfaces graphiques et multimodales [8]. Dans PAC-Amodeus, le découpage fonctionnel en couches ainsi que la structuration du contrôleur de dialogue en agents PAC offrent des avantages qu'il convient de conserver [8].

Couverture fonctionnelle du trèfle

Un agent PAC définit une unité de compétence décomposable en trois facettes fonctionnelles : l'Abstraction (son noyau fonctionnel), la Présentation (son rendu perceptible et manipulable par l'utilisateur) et le Contrôle (arbitre local qui gère aussi la communication avec d'autres agents). Dans PAC^{+3}, cette décomposition est maintenue et poussée plus avant de manière orthogonale avec les facettes fonctionnelles du trèfle. La matrice de la figure 2 montre les deux axes orthogonaux de décomposition fonctionnelle d'un agent PAC.

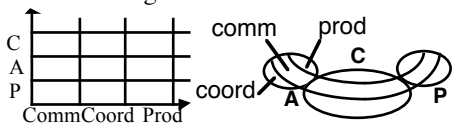

Figure 2: Décomposition fonctionnelle d'un agent PAC dans PAC^{+3}.

La compétence locale d'un agent est affinée selon les trois dimensions du trèfle des collecticiels. Ainsi l'abstraction d'un agent se décompose en trois parties, dédiée chacune à l'une ou l'autre des facettes du trèfle : partie abstraite pour soutenir la production (notée A.Prod), partie abstraite pour la coordination (A.Coord) et partie abstraite pour la communication (A.Com). La même opération est appliquée aux facettes C et P. Typiquement, A.Prod a une présentation P.Prod reliée par un C.Prod. Comme le montre la figure 2, notre stratégie revient à tronçonner en trois tranches l'agent PAC originel.

Ce "PAC Napolitain", de représentation compacte, masque, sous cette visualisation, les mécanismes sous-jacents de communication et de contrôle. Nous proposons trois vues sur ce modèle, pour rendre compte des stratégies envisageables.

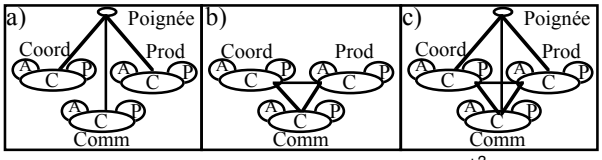

Figure 3 : La triple vue d'un agent PAC^{+3}.

Version a) de la figure 3 : un agent poignée (un agent PAC restreint à sa fonction de contrôle) centralise les échanges entre facettes fonctionnelles. Toutes les relations de dépendance entre celles-ci y sont exprimées. Concentrées dans cet agent, elles facilitent évolutions et maintenance. Cet agent poignée, seule interface avec les agents pères, redirige les événements, suivant leur nature, vers l'une ou l'autre des facettes. Ces indirections simplifient certes l'implémentation mais ne sont pas sans conséquence pour l'interface : traitements ralentis, temps de réponse augmentés, le prix classique de toute centralisation.

Version b) : à l'opposé des versions centralisées, la version b) opte pour une répartition des dépendances entre facettes. Ces facettes communiquent entre elles sans intermédiaire : elles ciblent directement la facette pertinente. En découlent rapidité, efficacité, robustesse mais évolutivité plus limitée, de part l'éparpillement du code.

Version c) : cette version est une version hybride qui concilie les deux premières. Son intérêt réside dans la

souplesse qu'elle procure : le spectre des combinaisons possibles est accessible dans sa globalité, au concepteur de jouer, par une stratégie fine, sur les avantages de chacune d'elles pour ne pas en cumuler les inconvénients. D'implémentation largement plus complexe, elle est, en pratique, plus délicate à mettre au point.

Quant au choix maintenant entre l'une ou l'autre de ces trois sous-architectures ? Peut-on en préconiser une de façon systématique ? La réponse à cette question est non. Aucune architecture n'est bonne ou mauvaise dans l'absolu. Chacune présente ses avantages, ses inconvénients, qu'il faut peser dans le contexte d'une application donnée. C'est toujours par rapport à des critères que s'évaluent, de façon comparative, différentes alternatives. Typiquement, disposant d'un langage d'expression de contraintes, un agent poignée serait simple à réaliser.

Répartition et Partage : modèle de Dewan

Nous avons jusqu'ici raisonné sans évoquer les problèmes de partage et de réplication propres aux systèmes répartis. Notre réponse s'inspire du modèle de Dewan : certaines couches de l'arche sont communes d'autre pas et les couches répliquées communiquent entre pairs par un mécanisme sous-jacent (dans un monde à objets, la technique des proxy par exemple). Nous pensons pousser le grain de partage, non pas au niveau de la couche comme chez Dewan, mais au niveau de l'agent ou grappe d'agents. La figure 4 illustre une communication à différents niveaux d'abstraction au sein d'une couche donnée, ici celle du contrôleur de dialogue.

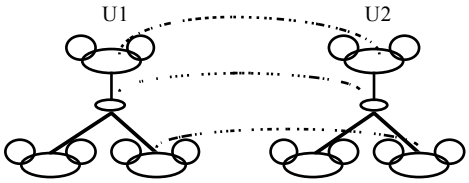

Figure 4 : Communication entre agents ou grappes d'agents pairs de deux contrôleurs de dialogue répartis sur deux stations U1 et U2. Les autres couches de l'arche ne sont pas représentées.

Des liens horizontaux peuvent en effet, conformément au modèle de Dewan, être établis à différents niveaux dans la modélisation. Il faut néanmoins savoir que chaque communication signifie ouverture d'une connexion et donc, facturation associée. Par conséquent, même si la plate-forme d'accueil le permet, l'ouverture de connexions intempestives ne peut être décidée sans une étude de rentabilité : tout est question de compromis. Précisons que ces liens peuvent être permanents ou établis, au contraire, à la demande. Il faut, dans ce cas, spécifier ces conditions de connexion. Ces solutions techniques sont à discuter dans un contexte donné et ne peuvent, en aucun cas, être préconisées de façon systématique.

Nous ne sommes pas certains (ni Dewan) du bien fondé du modèle de la fermeture éclair comme base de raisonnement sur le partage. Intuitivement, nous préférons la généralité du feston qui permet de varier entre réplication et partage tout au long des niveaux d'asbtraction.

En résumé

En résumé, PAC^{+3} est un PAC-Amodeus dont les agents du contrôleur de dialogue véhiculent les 3 facettes fonctionnelles des collecticiels et dont la réplication s'inspire du modèle de Dewan. Comme dans PAC-Amodeus, nous ne faisons aucune hypothèse sur la structuration interne des couches fonctionnelles autres que celle du contrôleur de dialogue : cette structure est héritée de ou adaptée à l'environnement d'accueil. Nous conservons les règles de communication de PAC-Amodeus et comme dans PAC, un agent PAC^{+3} n'est pas toujours agrémenté de toutes ses facettes.

ILLUSTRATION

L'illustration que nous proposons repose sur une extension de l'éditeur collaboratif SASSE : la barre de défilement associée aux utilisateurs distants est augmentée de photos "actives" identifiant leur propriétaire et permettant d'entrer en communication audio/vidéo avec ces participants.

Description

Chaque utilisateur dispose d'une interface ainsi structurée : une barre de menus (offrant notamment l'accès aux fichiers), une zone d'édition et une barre de défilement multi-utilisateur (figure 5). Notons que cette barre est composée de deux sous-barres, l'une personnelle; l'autre, dédiée aux utilisateurs distants, servant à la fois de support à la coordination et à la communication. Cette augmentation de SASSE permet de raisonner sur un exemple complet : les trois facettes fonctionnelles du trèfle y sont représentées.

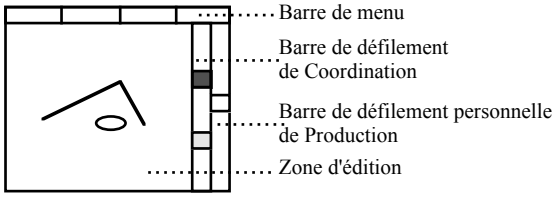

Figure 5 : SASSE accomodé.

Modélisation

En appliquant les règles d'identification des agents de PAC-Amodeus [8], nous associons un agent à la barre de menus, un agent à la zone d'édition et un autre à la barre multi-utilisateur. Nous obtenons la modélisation de la figure 6 à incarner pour chaque utilisateur.

En voici plus précisément la substance pour un utilisateur Ui :

Agent Barre multi-utilisateur :

- A.prod : position de Ui dans le texte
- P.prod : ascenseur dédié à Ui

- A.coord : liste des autres participants (Uj) et leur position dans le texte
- P.coord: ascenseurs des Uj
- A.comm : facteur de résolution d'image
- P.comm : photo, audio/vidéo d'un Uj

Agent Zone d'édition : exclusivement dédié à la production, il ne contient qu'une seule facette :

- A : pointeur vers le document (NF/ANF), centralisé ou non. Précisons que ce lien peut, pour des raisons d'évolutivité, être plutôt maintenu au niveau de l'agent père : rajout par exemple du plan du document.
- P : widget prédéfini (canvas Motif par exemple).

Agent haut-niveau :

- A : identificateur du fichier
- P : widget de type barre de menus.

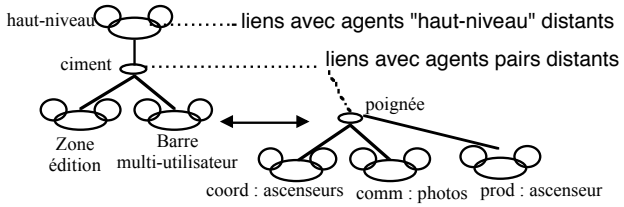

Figure 6 : modélisation du contrôleur de dialogue pour SASSE. Affinement de l'agent barre multi-utilisateur en 3 sous-agents reliés par une poignée.

Reste maintenant à animer cette modélisation. Pour des raisons de place, nous n'en présentons ici que l'essence : les événements, captés en local par les widgets, sont transmis à l'agent associé qui effectue alors, si besoin est, une rétro-action locale. Il envoie ensuite à ses supérieurs et/ou son agent pair cet événement enrichi. Inversement tout événement reçu d'un autre agent est traité avec reflet sur les facettes locales et récursivement vers les fils. Attention aux bouclages.

CONCLUSION

Cet article consigne notre réflexion actuelle sur la conception d'architecture en général et sur le cas des collecticiels en particulier. Au vu de l'expérience de PAC-Amodeus, nous estimons que PAC^{+3} constitue un bon cadre de réflexion offrant généricité, hétérogénéité et couverture fonctionnelle complète des collecticiels. Il nous reste, comme pour PAC-Amodeus, à définir des patterns, c'est-à-dire des configurations d'agents PAC^{+3} correspondant à des problèmes récurrents [6]. Il nous reste aussi, comme pour PAC-Amodeus, à définir des heuristiques explicitant le passage entre le modèle conceptuel et la mise en œuvre en fonction des outils d'implémentation disponibles.

Remerciements

Nous tenons à remercier France Télécom-CNET pour le soutien partiel apporté à cette recherche.

BIBLIOGRAPHIE

1. R. Baecker, D. Nastos, I. R. Posner, K. L. Mawby. The User-centred Iterative Design of Collaborative Writing Software. In Proc. INTERCHI'93, Human Factors in Comupting Systems, ACM, 1993, pp. 399-405.
2. P. Dewan, "Multiuser Architectures", in Engineering for Human-Computer Interaction, Chapman&Hall Publ., Bass&Unger Eds., 1995, pp. 247-270.
3. J. Coutaz, L. Nigay, D. Salber. Software architecture modelling for interactive systems. A paraître dans encyclopedia of HCI, Wiley&son, J. Nielsen eds.
4. P. Croisy. Collecticiel temps réel et apprentissage coopératif: des aspects sociaux et pédagogiques jusqu'au modèle multi-agent de l'interface de groupe. Thèse de l'université de Lille, 1995, 200 pages.
5. R. Hill, "The Abstraction-Link-View Paradigm: Using Constraints to Connect User Interfaces to Applications. In proceedings CHI'92, ACM:New York, 1992, pp. 335-342.
6. E. Gamma, R. Helm, R. Johnson, J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software". Addison Wesley, Reading, MA. 1995.
7. D. Garlan, M. Shaw, "An Introduction to Software Architecture", Advances in Software Engineering and Knowledge Engineering, V. Ambriola and G. Tortora Eds., Vol. 1, World Scientific Publ., 1993, pp. 1-39.
8. L. Nigay, "Conception et modélisation logicielles des systèmes interactifs : application aux interfaces multimodales". Thèse de l'Université Joseph Fourier, Grenoble 1, janvier 1994, 330 pages.
9. 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.
10. J.F. Paterson, "A taxonomy of Architectures for Synchronous Groupware Applications". Workshop on Software Architectures for Cooperative Systems, CSCW'94, ACM Conference on Computer Supported Cooperative Work, Chappel Hill, NC, 1994.
11. W. Reinhard, J. Schweitzer, G. Völksen. CSCW Tools: concepts and architecture, IEEE Computer, May 1994, 27(5), pp. 28-36.
12. D. Salber, "De l'interaction homme-machine individuelle aux systèmes multi-utilisateurs, L'exemple de la communication homme-homme médiatisée". Thèse de l'Université Joseph Fourier, Grenoble 1, septembre 1995
13. M. Shaw, D. Garlan, Software Architecture. Perspectives on an Emerging Discipline, Prentice Hall, 1995.