Le Modèle d'Architecture Clover pour les Collecticiels

Yann Laurillau

Laboratoire CLIPS-IMAG, Equipe IIHM
BP 53 38041 Grenoble Cedex 9 France
Yann.Laurillau@imag.fr

Laurence Nigay[1]

Computing Science Dept, University of Glasgow
17 Lilibank Gardens, glasgow G128RZ Scotland
laurence@dcs.gla.ac.uk

RÉSUMÉ
Dans cet article, nous présentons le modèle d'architecture logicielle Clover dédié aux collecticiels. Le modèle Clover résulte de la combinaison de l'approche en couches répliquées et partagées de Dewan avec la décomposition fonctionnelle du trèfle des collecticiels (Clover). Le trèfle des collecticiels définit trois classes de fonctionnalités, intitulées production, communication et coordination. Ces trois classes sont présentes dans chaque couche fonctionnelle de notre modèle. Nous illustrons la discussion au moyen du système CoVitesse, un système de navigation collaborative sur le WWW : l'architecture logicielle de CoVitesse découle de l'application de notre modèle Clover.

MOTS CLÉS : Modèle d'Architecture Logicielle, Architecture Conceptuelle, Architecture Implémentationnelle, Collecticiel, Modèle du Trèfle.

ABSTRACT
In this paper we present the Clover architectural model, a new conceptual architectural model for groupware. Our model results from the combination of the layer approach of Dewan's generic architecture with the functional decomposition of the Clover design model. The Clover design model defines three classes of services that a groupware may support, namely, production, communication and coordination services. The three classes of services can be found in each functional layer of our model. Our model is illustrated with the CoVitesse system, its software being organized according to our Clover architectural model.

KEYWORDS: Software Architectural Model, Conceptual Architecture, Implementation Architecture, Groupware, Clover Design Model.

INTRODUCTION
Cet article a pour thème d'étude la conception logicielle des collecticiels. Nous constatons que les collecticiels, plus particulièrement sur le réseau Internet, sont de plus en plus nombreux. Citons des systèmes comme ICQ pour discuter en ligne, comme StarCraft pour jouer en réseau ou encore Napster pour échanger des fichiers musicaux. Cette multiplicité des collecticiels ne s'est pas accompagnée d'outils d'aide à la conception et à la réalisation logicielles : la réalisation logicielle d'un collecticiel reste une tâche complexe et non systématique. Tandis qu'un développement ad hoc d'un système est acceptable pour un prototype jetable, il est admis que la conception logicielle d'un système complexe comme un collecticiel ne peut reposer que sur le savoir artisanal du développeur. Dans cet article, nous focalisons sur la conception logicielle systématique d'un collecticiel qui s'appuie sur un modèle d'architecture logicielle.

Un modèle d'architecture logicielle est utilisé lors de la phase de conception logicielle dans le cycle de vie du logiciel pour concevoir l'architecture logicielle du système : l'usage commun assimile une architecture logicielle à un ensemble organisé de composants dont les interactions sont médiatisées par des entités spécialisées ou connecteurs. Le processus de conception d'une architecture logicielle recouvre les activités suivantes : 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. Le résultat de ces quatre activités constitue la structure modulaire ou architecture conceptuelle. Selon les cas, il convient ensuite de focaliser sur 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 pour un collecticiel, système par définition réparti. La structure physique est alors conçue : elle complète la structure modulaire de l'étape précédente pour constituer ensemble l'architecture implémentationnelle. Pour une description détaillée des concepts architecturaux et du processus de développement d'une architecture logicielle, nous encourageons le lecteur à se référer à [4] ; dans la suite de ce papier, nous utilisons les termes architecture conceptuelle et architecture implémentationnelle.

Les modèles d'architecture pour collecticiels sont principalement conceptuels comme Clock [6], PAC* [2] ou le modèle de Dewan [5]. Ces modèles définissent des structures modulaires normalisées. Par exemple, le modèle de Dewan préconise un empilement de couches partagées et répliquées communiquant entre elles par échanges d'événements. L'architecture conceptuelle issue de l'application d'un modèle est ensuite traduite en architecture implémentationnelle dépendante des outils de développement utilisés. Les outils de développement pour collecticiels comme GroupKit [15] ou COCA [9] n'éliminent pas la nécessité d'une architecture logicielle conceptuelle. L'architecture logicielle est un artefact incontournable tant que les outils impliquent une phase de programmation. Sans une architecture logicielle, le collecticiel est plus difficile à développer, à modifier, à étendre et à maintenir. Les modèles d'architecture logicielle conceptuelle existants préconisent une décomposition du collecticiel à gros grains. Notre approche vise à définir une décomposition plus fine ; nous sommes donc complémentaires avec les modèles existants. Pour cela, nous nous appuyons sur le trèfle des

[1] En année sabbatique, de l'Université de Grenoble 1, laboratoire CLIPS-IMAG, équipe IIHM.

collecticiels [16], qui définit trois classes de fonctionnalités qu'un collecticiel peut offrir : les fonctionnalités de production, de communication et de coordination.

Cet article est structuré comme suit : nous décrivons d'abord le trèfle des collecticiels et soulignons son impact sur la conception du collecticiel et de son IHM. Nous rappelons ensuite les principes de base de deux modèles d'architecture logicielle, le modèle de Dewan et le modèle PAC*, deux modèles sur lesquels repose notre modèle Clover. La partie suivante est alors dédiée à notre modèle Clover. Ce dernier est un métamodèle : plusieurs modèles d'architecture peuvent être dérivés du métamodèle Clover. Nous présentons en premier un modèle d'architecture Clover, puis sa généralisation, le métamodèle Clover. En dernière partie, nous illustrons le métamodèle en décrivant l'architecture Clover de notre système CoVitesse, un système de navigation collaborative sur le WWW. Nous présentons les caractéristiques du système CoVitesse dans le paragraphe suivant.

ILLUSTRATION : LE SYSTÈME COVITESSE

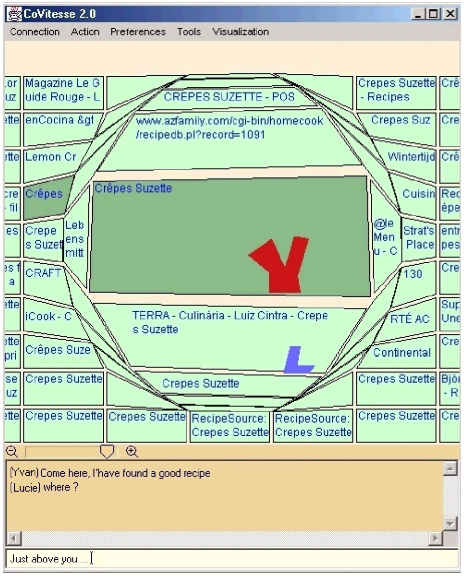

Figure 1: Fenêtre principale du système CoVitesse.

CoVitesse [3][7] est un système de navigation collaborative sur le WWW. Les utilisateurs naviguent dans un même espace d'information pour collecter collaborativement de l'information. Ce système a été réalisé à partir du système Vitesse [12], une application mono-utilisateur qui visualise l'ensemble des réponses d'une requête soumise à un moteur de recherche sur le WWW. Comme le montre la Figure 1, CoVitesse présente l'ensemble des pages affichées sous forme de polygones. Les utilisateurs sont représentés par des formes géométriques colorées (un Y rouge pour Yvan et un L bleu pour Lucie). Un utilisateur peut naviguer dans l'espace, observer les autres utilisateurs présents, organiser son panier contenant l'ensemble des pages collectées, lire une page en sélectionnant un polygone (la page correspondante étant automatiquement téléchargée dans un navigateur), communiquer via un forum de discussion situé sous l'espace d'information, créer ou joindre un groupe.

Un utilisateur peut créer son propre groupe de travail, identifié par un nom et une couleur. Par exemple dans la Figure 1, Yvan, représenté par la forme en Y, est membre d'un groupe identifié par la couleur rouge. Dans la version actuelle, il existe quatre types de groupe détaillés dans [8].

CoVitesse maintient une base de donnée de toutes les informations collectées ou modifiées au cours d'une session, comme les informations concernant les groupes et les utilisateurs, et les résultats collectés individuellement ou en groupe. Ces données sont protégées par un simple mécanisme de mot de passe. Ainsi, lorsqu'un utilisateur démarre une nouvelle session, il peut récupérer ses données personnelles.

MODÈLE DU TRÈFLE DES COLLECTICIELS

Description

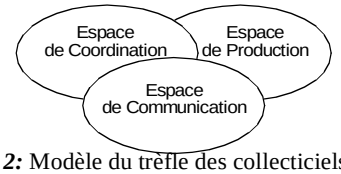

Figure 2: Modèle du trèfle des collecticiels.

Le modèle du trèfle [16] propose une classification fonctionnelle des collecticiels, selon trois espaces présentés à la Figure 2 : l'espace de production (ou espace ontologique) recouvre les objets et les fonctionnalités qui sont liés à l'activité de groupe, par exemple dans CoVitesse, l'ensemble des pages collectées dans le caddy d'un groupe ; l'espace de communication offre aux utilisateurs la possibilité d'échanger des messages (Communication Homme-Homme Médiatisée), par exemple dans CoVitesse, les échanges de messages textuels grâce au forum de discussion ; l'espace de coordination modélise l'organisation des acteurs, leurs rôles, leurs droits, les procédures et protocoles à respecter vis-à-vis des entités de l'espace de production, par exemple dans CoVitesse, seul le chef d'un groupe de type coordonné a la responsabilité d'accepter un nouveau membre. Néanmoins, tous les collecticiels n'offrent pas nécessairement les trois types de fonctionnalités. Ce point est développé dans le paragraphe suivant.

Lors de la phase de conception du système, en amont de la phase de conception logicielle, les fonctionnalités offertes par le collecticiel sont identifiées et leurs accès sont organisés au sein de l'interface homme-machine (IHM). Le trèfle des collecticiels est donc un modèle de conception.

Conception des Collecticiels

Le modèle du trèfle des collecticiels identifie trois types de fonctionnalités qu'un collecticiel peut offrir. En cela, le trèfle des collecticiels est un modèle de conception, utile lors de la phase de spécification fonctionnelle du collecticiel. Il convient néanmoins de noter que tous les collecticiels ne proposent pas nécessairement les trois types de fonctionnalités. Par exemple, un mediaspace, dédié à la communication informelle afin de renforcer la conscience de groupe, repose principalement sur des services de communication et quelques services de coordination tels que joindre ou quitter une session. En

l'occurrence, il n'y a aucun support pour la production. De plus, plusieurs approches visent à considérer la coordination indépendamment des aspects de production et de communication. Par exemple, dans [5], le gestionnaire de session dédié à la coordination est considéré comme un composant externe au collecticiel, à l'opposé des composants dédiés à la production et à la communication qui constituent le corps du collecticiel. Autre exemple, la plate-forme COCA [9] est dédiée à la coordination : elle régule l'utilisation simultanée par plusieurs utilisateurs d'applications, telles qu'un tableau blanc ou un forum de discussion, grâce à un ensemble de règles spécifiées dans un langage du type Prolog. Ces règles sont interprétées à l'exécution par la machine virtuelle COCA. L'objectif affiché de cette plate-forme est de rendre collaboratives des applications mono-utilisateur, en greffant un module de coordination externe.

En conclusion, le modèle du trèfle identifie trois types de fonctionnalités d'un collecticiel, certaines études considérant la coordination comme un type particulier de fonctionnalités. Le modèle du trèfle est donc utile lors de la phase de spécification fonctionnelle d'un collecticiel. Nous étudions maintenant l'impact du modèle du trèfle sur la conception de l'interface du collecticiel.

Conception de l'Interface Utilisateur

L'application du modèle du trèfle à la conception de l'interface homme-machine (IHM) peut aboutir à un collecticiel peu utilisable : par exemple une application directe consiste à concevoir une fenêtre par facette du trèfle. C'etait le cas de la première version de notre système CoVitesse. Ainsi les trois principales fenêtres étaient : une fenêtre dédiée à la production contenant l'ensemble des pages collectées, une autre réservée à la communication avec le forum de discussion en mode texte et une troisième dédiée à la coordination incluant l'espace d'information. Une évaluation empirique menée avec des groupes de quatre utilisateurs devant réaliser un scénario (collecter les dix pages les plus significatives sur l'interaction homme-machine) a montré qu'il était important de combiner, au sein d'une même zone interactionnelle, l'espace d'information avec le forum de discussion (combinaison de la coordination avec la communication). En effet, les utilisateurs ont exprimé le besoin de pouvoir à la fois observer le déplacement des avatars dans l'espace d'information et lire les messages échangés.

De plus, de nombreux collecticiels proposent des objets graphiques d'interaction combinant les trois facettes du modèle du trèfle. Par exemple, la barre de défilement collaborative [2] combine les trois types de fonctionnalités. Une barre, dédiée à la production personnelle, est une barre de défilement ordinaire utilisée pour se déplacer dans le texte. L'autre barre, dédiée à la coordination et à la communication, contient plusieurs ascenseurs, un par utilisateur qui peut être vu à travers une vignette vidéo contenue à l'intérieur même de l'ascenseur. Du point de vue fonctionnel, nous concluons que la position courante d'un utilisateur dans le texte contribue à la coordination, tandis que le service vidéo concerne la communication. Du point de vue de l'interface, un unique objet d'interaction (la barre de défilement) est utilisé à la fois pour la coordination et la communication.

En synthèse, au regard de notre expérience avec CoVitesse et de l'existant, le modèle du trèfle est utile pour identifier et classer les fonctionnalités du collecticiel, mais ne doit pas servir de guide à la conception de l'interface du collecticiel. Ce constat motive l'accent mis sur les couches du niveau sémantique et non du niveau interface dans notre étude sur la conception logicielle des collecticiels.

ARCHITECTURES POUR LES COLLECTICIELS

Nous présentons deux modèles d'architecture logicielle pour les collecticiels qui sont complémentaires, l'un étant de style "machines abstraites en couche", le métamodèle de Dewan, et l'autre de style à agents, le modèle PAC*. Notre métamodèle Clover exploite des principes de structuration sur ces deux modèles.

Métamodèle d'Architecture de Dewan

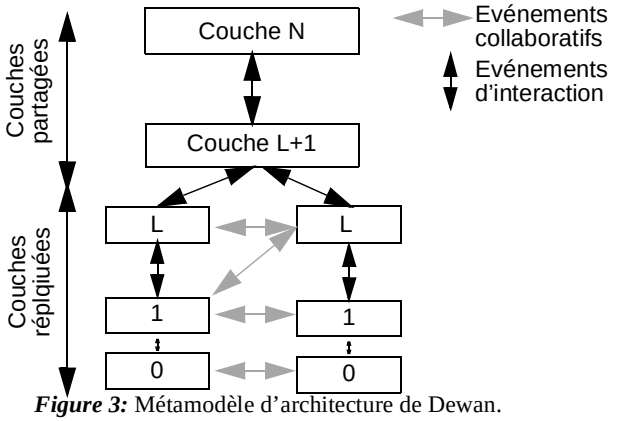

Figure 3: Métamodèle d'architecture de Dewan.

Le métamodèle d'architecture de Dewan [5] est une généralisation du modèle Arch [1] et du modèle de la fermeture éclair (*zipper model*) [14]. Ce métamodèle préconise une décomposition en plusieurs couches d'un collecticiel. Comme le montre la Figure 4, le nombre de couches est variable, chaque couche correspondant à un niveau d'abstraction. Au niveau le plus haut, la couche N à la Figure 4 est de nature sémantique (Noyau Fonctionnel dans le modèle Arch [1]), tandis que la couche de niveau le plus bas (couche 0 à la Figure 4) est dépendante du matériel (Composant Interaction Physique dans le modèle Arch [1]).

L'architecture globale est constituée d'une racine et de plusieurs branches. La racine, comme le montre la Figure 3, est composée de couches partagées (niveaux L+1 à N). Les objets gérés par les couches partagées sont publiques. Les branches sont composées de couches répliquées (niveaux 0 à L), reliées à la racine au niveau de la couche L, le point de branchement. Les objets gérés par les branches sont privés. L'ensemble des objets du domaine, partagés et privés, constitue l'état de l'interaction pour un utilisateur donné.

Les couches communiquent entre elles selon deux types d'événements : les événements d'interaction, échangés verticalement à travers les couches d'une branche, et les événements de collaboration, échangés entre les couches appartenant à deux branches différentes.

Dans le reste de la discussion, nous utiliserons de manière interchangeable le terme de couche et de composant. Certes, un composant peut englober plusieurs couches, néanmoins pour simplifier le discours nous considérons qu'il n'y a qu'une couche par composant.

Modèle PAC*

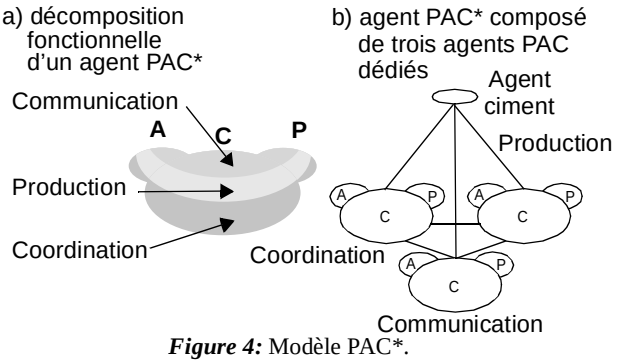

Figure 4: Modèle PAC*.

Le modèle PAC* [2] est la version collaborative du modèle d'architecture hybride PAC-Amodeus [11]. PAC-Amodeus préconise un découpage fonctionnel selon les couches du modèle Arch [1] ainsi que la structuration du Contrôleur de Dialogue en agents PAC. Un agent PAC est composé de trois facettes : une facette Présentation qui définit l'interface utilisateur de l'agent, une facette Abstraction qui définit la compétence de l'agent et une facette Contrôle qui maintient les liens entre les deux autres facettes (Présentation et Abstraction) et qui assure la communication avec les autres agents.

Dans le modèle PAC*, le Contrôleur de Dialogue est peuplé d'agents PAC*, agents PAC tronçonnés selon les trois types de fonctionnalités du modèle du trèfle, comme le montre la Figure 5(a). Une forme hybride d'un agent PAC* est présentée à la Figure 5(b) : celle-ci est composée de trois agents PAC, chacun dédié à l'une des facettes du modèle du trèfle. Les trois agents communiquent entre eux via leur facette Contrôle, et un agent ciment gère la communication avec l'extérieur, avec d'autres agents PAC*.

MODÈLE D'ARCHITECTURE CLOVER

Le modèle d'architecture Clover reprend des principes de structuration des deux modèles d'architecture précédents. De notre métamodèle Clover, plusieurs modèles d'architecture Clover peuvent être définis. Ce paragraphe est dédié à un modèle d'architecture dérivé du métamodèle, que nous présentons ensuite comme une généralisation.

Description

Le modèle d'architecture Clover est une instance du métamodèle de Dewan en appliquant les cinq couches préconisées par le modèle Arch [1]. Néanmoins, comme le montre la Figure 6, notre modèle compte six couches fonctionnelles : en effet, le Noyau Fonctionnel (NF) du modèle Arch est divisé en deux couches : le Trèfle Fonctionnel (TF) **répliqué** et privé, et le Trèfle Fonctionnel (TF) **partagé** et publique. Nous n'avons pas représenté à la Figure 5 les composants Interaction Logique et Physique, situés sous le Contrôleur de Dialogue. De plus, nous n'avons pas représenté toutes les flèches symbolisant la communication entre les deux branches afin d'alléger la figure.

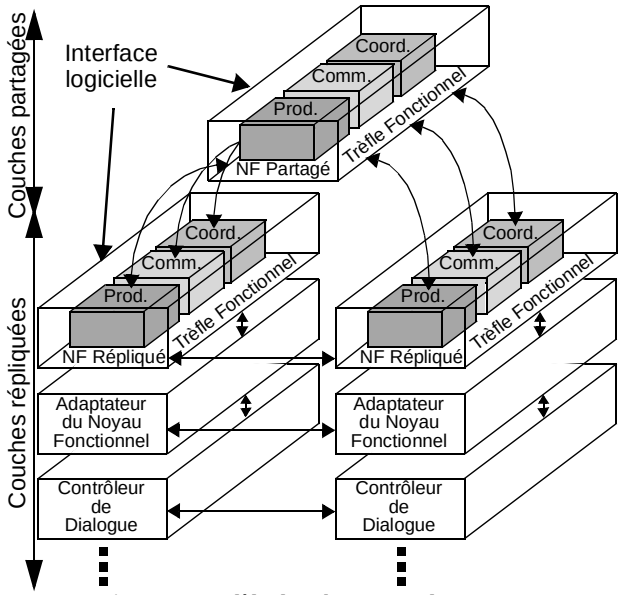

Figure 5: Modèle d'architecture Clover..

Tous les composants du modèle Arch sont répliqués à l'exception du **TF partagé**. Le **TF partagé** permet à tous les utilisateurs de manipuler les objets du domaine et d'accéder aux différents services au cours de l'interaction. A l'opposé, le **TF répliqué** gère un ensemble d'objets du domaine et l'état du système qui sont privés à un utilisateur. Nous illustrons cette distinction "partagé/ privé" lors de la présentation de l'architecture Clover du système CoVitesse. Dans ce modèle Clover, le point de branchement est donc le **TF répliqué**. Ce point de branchement implique un degré de réplication, comme défini dans [5], élevé. Un degré de réplication élevé permet un mode WISIWYS relâché (*What I See Is What You See*). A l'opposé, un degré de réplication faible, défini par un point de branchement au niveau des composants Interaction, correspond à un mode WISIWYS stricte. D'autres points de branchement et donc des degrés de réplication différents sont certes possibles. Cette possibilité est explicitée par le métamodèle Clover.

L'originalité du modèle Clover réside dans la décomposition selon les facettes du trèfle des collecticiels des composants fonctionnels : **TF partagé** et **TF répliqué.** Chaque composant **TF** est composé de trois sous-composants, chacun dédié à une facette du trèfle des collecticiels. Chaque sous-composant fournit des services et gère des objets spécifiques à la production, à la communication ou à la coordination. Ces trois sous-composants sont englobés par une interface logicielle, similaire à une approche par composants comme CORBA [13]. Le rôle de cet interface logicielle est triple :

- L'interface logicielle englobe les trois sous-composants. Son rôle est de masquer la structuration interne du composant selon les facettes du trèfle. Cet interface permet donc la communication avec des couches voisines qui ne seraient pas structurées selon les facettes du trèfle. Ainsi, à la Figure 6, grâce à l'inter-

face logicielle, le **TF répliqué** peut communiquer avec l'Adaptateur de Noyau Fonctionnel qui n'est pas structuré selon les facettes du trèfle.

- L'interface logicielle gère les objets communs aux trois sous-composants. Ainsi les sous-composants Production, Communication et Coordination peuvent utiliser des objets en commun et leur appliquer des traitements qui leur sont propres.
- L'interface logicielle définit aussi un ensemble de fonctionnalités incluant les services systèmes ainsi que les fonctionnalités relevant de l'activité mono-utilisateur. En effet, ces fonctionnalités ne sont pas intrinsèquement collaboratives et ne relèvent donc pas de l'une des trois facettes du modèle du trèfle.

La communication entre couches d'une branche ou entre couches de différentes branches suit les mêmes règles prescrites par le modèle de Dewan. Cependant, les événements d'interaction et de collaboration sont différenciés selon les trois facettes : production, communication et coordination. Cet affinement permet une communication directe entre composants spécialisés. Le modèle maintient néanmoins un événement générique qui assure la communication avec les couches qui ne seraient pas structurées selon les trois facettes.

Avec ce modèle Clover, nous nous sommes principalement intéressés aux composants fonctionnels et nous n'avons émis aucune hypothèse sur la structuration des autres composants, les composants Interaction Logique et Physique et le Contrôleur de Dialogue. Comme nous l'avons expliqué précédemment, le trèfle des collecticiels est difficilement applicable au niveau de l'interface ; aussi la décomposition selon les trois facettes du trèfle des composants Interaction Logique et Physique ne semble pas adaptée. Nous revenons sur la structuration du Contrôleur de Dialogue dans le paragraphe suivant en comparant le modèle Clover avec le modèle PAC*.

Extension du Modèle PAC*

Dans le modèle PAC*, le Contrôleur de Dialogue est composé d'une hiérarchie d'agents PAC et PAC* qui échangent des événements avec le Noyau Fonctionnel via l'Adaptateur du Noyau Fonctionnel. Ainsi, notre modèle d'architecture Clover permet aux trois facettes Abstraction d'un agent PAC* (voir Figure 4) de communiquer avec le sous-composant correspondant du Trèfle Fonctionnel répliqué. Cette communication n'est pas directe puisque les événements transitent via l'Adaptateur du Noyau Fonctionnel (qui n'est pas nécessairement structuré selon les trois facettes du modèle du trèfle), puis via l'interface logicielle du Trèfle Fonctionnel répliqué**.**

De plus, comme le montre la Figure 4 (a), un agent PAC* contient un agent ciment. Ce dernier gère la communication entre d'une part les trois agents Production, Communication et Coordination et d'autre part le monde extérieur. Le rôle de cet agent ciment est comparable à un des rôles assurés par l'interface logicielle du Trèfle Fonctionnel de notre modèle : la communication avec les autres couches.

Propriétés

Le découpage fonctionnel du Noyau Fonctionnel selon le modèle du trèfle véhicule plusieurs propriétés. Du point de vue conceptuel, comme nous l'avons expliqué en introduction, ce découpage permet d'expliciter au niveau de l'architecture logicielle des concepts manipulés lors de la phase de conception du collecticiel et en particulier lors de la phase de spécification fonctionnelle. Du point de vue implémentationnel, ce découpage fonctionnel se traduit par une modularité accrue du logiciel qui à son tour implique une plus grande modifiabilité. La modifiabilité est une propriété importante dans le cadre d'une conception itérative centrée sur l'utilisateur. Par exemple, dans le cas du système CoVitesse, il serait aisé d'ajouter un service de communication vidéo en ajoutant un module dédié à la communication et sans affecter les autres modules existants. De plus, une forte modularité diminue la complexité de programmation. Par expérience, le développement du système CoVitesse a été facilité grâce à l'architecture Clover. L'application du modèle a permis de développer chaque partie sans être obligé d'avoir une vue globale de l'ensemble du logiciel. Par exemple, les trois sous-composants des Trèfles Fonctionnels répliqués et partagés ont été développés séparément, diminuant très significativement la complexité de la programmation.

MÉTAMODÈLE D'ARCHITECTURE CLOVER

Après avoir présenté un modèle d'architecture Clover et ses propriétés, nous présentons une généralisation du modèle, le métamodèle Clover.

Description

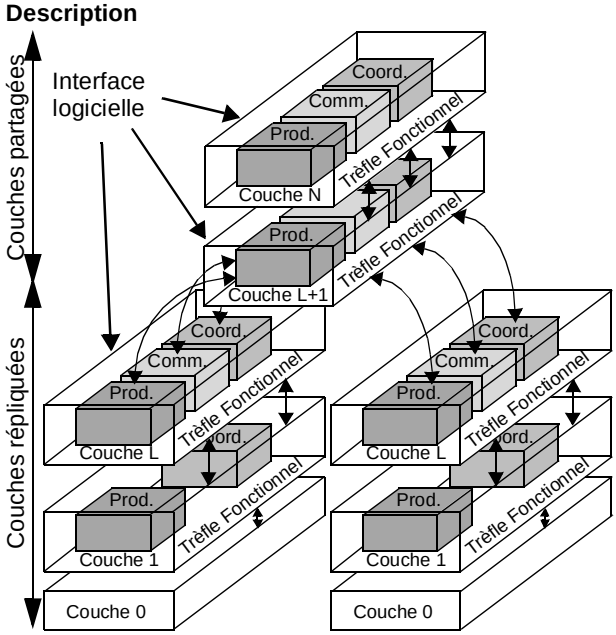

Figure 6: Métamodèle d'architecture Clover.

Comme le métamodèle de Dewan, le métamodèle Clover, présenté à la Figure 7, structure un collecticiel en un nombre variable de couches. Si nous comparons le métamodèle Clover (Figure 7) avec le modèle Clover (Figure 6) du paragraphe précédent, la couche L désigne le Trèfle Fonctionnel répliqué tandis que les couches L+1 à N correspondent au Trèfle Fonctionnel partagé.

L'originalité du métamodèle réside dans la structuration des unités logicielles constitutives de la racine et des branches, une unité étant notée Trèfle Fonctionnel (TF). Un TF est généralement décomposé en trois sous-composants (production, communication et coordination) englobés par une interface logicielle. La communication entre ces unités est réalisée grâce à des événements qui sont soit génériques, soit dédiés à la production, à la communication ou à la coordination. Notons que pour ne pas surcharger la Figure 7, nous n'avons pas représenté l'échange d'événements entre des unités de branches distinctes.

Le rôle principal de l'interface logicielle d'un composant TF est de masquer la structuration interne du composant. Ainsi un composant TF peut être empilé avec des composants non structuré selon les facettes du trèfle des collecticiels. En particulier, les couches les plus basses dépendantes du matériel ne sont généralement pas dédiées au développement de collecticiels et donc ne sont pas structurées selon les trois facettes du trèfle des collecticiels. Par exemple à la Figure 7, la couche 0 est un composant qui n'est pas détaillé en sous-composants qui seraient dédiés à la collaboration. De plus, la décomposition en trois sous-composants d'une unité TF n'est pas obligatoire. Un TF peut comporter que deux sous-composants, par exemple Production et Coordination comme la couche 1 de la Figure 7.

En synthèse, l'unité logicielle constitutive du métamodèle, un TF, contient une interface logicielle qui à son tour englobe éventuellement un, deux ou trois sous-composants, respectivement dédiés à la production, communication et coordination. La cohabitation de styles offre l'avantage de choisir le style le mieux adapté à telle ou telle couche de l'organisation structurelle. Inversement, l'hétérogénéité implique de maintenir une interface logicielle au niveau de chaque couche.

Propriétés

Le métamodèle ne fixe pas le nombre de couches. Cette flexibilité autorise l'ajout de couches pour mieux répartir les fonctionnalités, et permet ainsi d'accroître la modularité du code. Le découpage fonctionnel interne à une couche selon les trois facettes du trèfle des collecticiels contribue aussi à augmenter la modularité. Comme expliquée dans la section précédente, la modularité augmente la modifiabilité du code et réduit la complexité de programmation. De plus la modularité concourt à une meilleure réutilisabilité, réutilisabilité d'une couche complète ou d'une facette d'une couche, comme un service de communication.

Le métamodèle autorise un découpage fonctionnel partiel d'une couche : les trois sous-composants dédiés à la production, la communication et la coordination ne sont pas nécessairement tous présents. Cette flexibilité permet l'empilement de couches hétérogènes, dont le découpage fonctionnel est partiel. Par exemple, les systèmes de messagerie instantanée (*Instant Messaging*) sont souvent utilisés pour coordonner différents utilisateurs [17]. Pour réaliser un tel système de coordination reposant sur de la communication, l'architecture sera composée, par exemple, d'une couche dédiée à la coordination empilée au-dessus d'une couche dédiée à la communication.

De plus, le métamodèle considère la production, la communication et la coordination sur le même plan, à l'opposé de certaines approches, déjà mentionnées dans ce papier, qui examinent la coordination indépendamment des aspects de production et de communication. Par exemple dans [5], le gestionnaire de session dédié à la coordination est un composant externe à l'architecture du collecticiel. Ce composant externe gère les utilisateurs et les groupes au cours d'une session, et donc a la responsabilité de créer dynamiquement des branches dans l'architecture logicielle. Dans le métamodèle Clover, le gestionnaire de session est conséquemment situé dans une couche partagée de la racine. Au niveau de l'architecture conceptuelle, nous concevons toujours au moins une couche partagée. Cette dernière n'implique pas une architecture implémentationnelle centralisée. La dimension "partagé-répliqué" du niveau conceptuel est orthogonale à la dimension "centralisé-réparti" du niveau implémentationnel. Ainsi une couche partagée de l'architecture conceptuelle peut se traduire par un processus unique centralisé ou par un ensemble de processus répliqués synchronisés au niveau de l'architecture implémentationnelle.

Enfin, le métamodèle étend celui de Dewan [5]. Une architecture selon le métamodèle de Dewan est caractérisée par un degré de conscience de groupe (*awareness degree*). Ce degré désigne le niveau de la couche la plus haute dans l'architecture qui est dépendante de l'aspect collaboratif. Avec le métamodèle Clover, nous déclinons ce degré de conscience de groupe en trois mesures : degré de production collaborative, degré de communication et degré de coordination. Cet affinement permet par exemple d'établir une classification des systèmes existants plus précise que celle décrite dans [5].

ARCHITECTURE CLOVER EN PRATIQUE : ARCHITECTURE DU SYSTÈME COVITESSE

L'illustration de notre métamodèle Clover repose sur notre système Co-Vitesse (Figure 1).

La Figure 7 présente l'architecture Clover du système CoVitesse. Pour l'architecture conceptuelle, nous avons appliqué le modèle Clover de la Figure 6, composé des Trèfles Fonctionnels répliqués et partagés. Au niveau implémentationnel, CoVitesse est écrit en Java, et son architecture implémentationnelle est de type client/ serveur. Le Trèfle Fonctionnel partagé est centralisé sur le serveur et le Trèfle Fonctionnel répliqué est décomposé en deux parties, une au niveau du serveur et l'autre au niveau du client.

Nous décrivons l'architecture en partant de la couche la plus haute vers la couche la plus basse. Au plus haut niveau d'abstraction, le Trèfle Fonctionnel partagé, est composé d'un NF dédié à la production (NF_{prod}), d'un NF dédié à la communication (NF_{comm}), d'un NF dédié à la coordination (NF_{coord}) et d'une interface logicielle notée méta-serveur. Ce dernier gère l'accès aux trois Noyaux Fonctionnels spécialisés et les appels à l'infrastructure réseau.

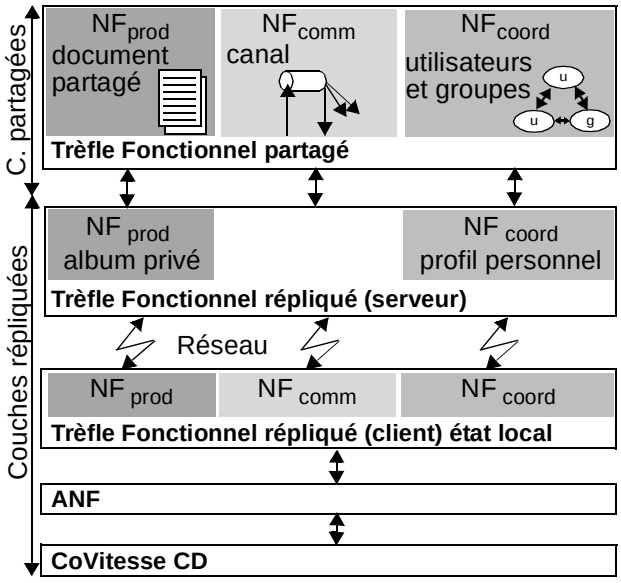

Figure 7: Architecture implémentationnelle de CoVitesse.

- *Composant partagé dédié à la production (NF_{prod})* : ce composant maintient l'ensemble des documents partagés, représentés par des pages à la Figure 7. Les principaux concepts manipulés sont les documents, les références, les albums et les catalogues. Un document, associé à une référence, est l'entité d'information élémentaire manipulée par les utilisateurs. Un album, créé et géré par des auteurs, contient des documents et des références. Un catalogue est une collection de références. De nombreuses fonctions sont disponibles telles que créer ou récupérer un catalogue, créer un album, ajouter ou retirer un auteur d'un album, ajouter un document dans l'album, modifier le contenu d'un album. Dans CoVitesse, une référence est une URL et un document, le contenu de la page web, c'est-à-dire un document HTML. Un catalogue est un ensemble d'URL construit à partir de l'ensemble des réponses d'une requête soumise à un moteur de recherche. L'album est aussi un ensemble d'URL.

- *Composant partagé dédié à la communication (NF_{comm})* : ce composant gère différents canaux de communication, représentés par un tube à la Figure 7. Le canal de communication est l'unique concept manipulé par le NF_{comm}. Un client peut s'abonner ou se désabonner à un canal de communication et peut y poster des messages. Dans CoVitesse, l'unique canal de communication est celui du forum de discussion textuel, comparable à une liste de diffusion : quand un utilisateur poste un message sur un canal, celui-ci est diffusé à l'ensemble des abonnés.

- *Composant partagé dédié à la coordination (NF_{coord})* : Ce composant a la responsabilité de coordonner les utilisateurs et les groupes. Il dispose de mécanismes gérant les accès concurrents aux données partagées comme les préférences d'un groupe et notifie tous les clients des changements d'état. Les principaux concepts manipulés par NF_{coord} sont les concepts d'utilisateur et de groupe :
  - Utilisateur : deux fonctions permettent respectivement de lire et de modifier les informations personnelles d'un utilisateur, tandis que deux autres fonctions sont dédiées à la lecture et à la modification des droits d'accès à ces informations. La réalisation actuelle offre la possibilité de définir son propre concept d'utilisateur en fonction du collecticiel. Dans CoVitesse, un utilisateur est identifié par un nom et un mot de passe. La description d'un utilisateur contient aussi une forme géométrique, des préférences de filtrage, une adresse mél, etc.
  - Groupe : un groupe est défini par un ensemble de membres, des droits d'accès au groupe et un ensemble minimal de fonctions. Ces fonctions permettent de modifier ou d'obtenir les informations relatives à un groupe, de créer et de joindre un groupe (un utilisateur soumettant une requête à un des membres pour une acceptation éventuelle), de quitter un groupe. Dans CoVitesse, un groupe est défini par un nom, une couleur, des préférences de filtrage et un type de navigation.

Comme le montre Figure 7, le Trèfle Fonctionnel partagé est empilé au-dessus du Trèfle Fonctionnel répliqué. Ce dernier est, au niveau implémentationnel, divisé en deux parties, l'une au niveau client et l'autre au niveau du serveur. Le Trèfle Fonctionnel répliqué localisé sur le client maintient un état local à l'application et une copie de l'état privé situé sur le serveur. Un mécanisme de cache permet d'optimiser les performances à l'exécution. Le Trèfle Fonctionnel répliqué situé sur le serveur est constitué de deux sous-composants, dédiés à la production et à la coordination (NF_{prod} et NF_{coord}). Ces composants maintiennent les données privées d'un utilisateur. L'implémentation actuelle du système permet de conserver ces données de manière persistante sur le serveur. D'une part, le NF_{prod} répliqué gère l'album privé de l'utilisateur contenant les URL qu'il a collectés au cours de la session. De plus ce composant a accès à l'album du groupe, maintenu et protégé par le NF_{prod} partagé. D'autre part, le NF_{coord} répliqué maintient les préférences personnelles de l'utilisateur. Les deux NF_{coord}, répliqués et partagés, manipulent les mêmes données relatives à l'utilisateur. Cependant, le composant répliqué est l'unique instance autorisée à modifier les données privées comme le mot de passe. Le composant partagé peut uniquement lire ces données quand celles-ci sont publiées.

Dans la version actuelle de CoVitesse, il n'y a pas de sous-composant dédié à la communication dans le Trèfle Fonctionnel répliqué. Par conséquent, l'interface logicielle de ce Trèfle Fonctionnel transfère directement les événements dédiés à la communication entre le Trèfle Fonctionnel partagé et l'Adaptateur du Noyau Fonctionnel. Cependant, des développements sont en cours pour intégrer la facette manquante. Ce composant gérera par exemple des listes de destinataires (*aliases*). De plus, nous avons prévu d'étendre le sous-composant dédié à la communication du Trèfle Fonctionnel partagé pour permettre la discussion uniquement entre membres d'un même groupe.

Les autres couches de l'architecture de CoVitesse ne sont pas décomposées selon les trois facettes du trèfle des collecticiels. Ces couches sont au nombre de quatre : l'Adaptateur du Noyau Fonctionnel (ANF), le Contrôleur de Dialogue (CD) et les deux composants Interaction Logique et Physique. Ces deux derniers composants ne sont pas dessinés à la Figure 8, ils se situent en dessous du Contrôleur de Dialogue.

CONCLUSION

Nous avons proposé un métamodèle d'architecture logicielle, Clover, pour les collecticiels. Ce métamodèle résulte de la combinaison de l'approche en couches répliquées et partagées du métamodèle de Dewan avec la décomposition fonctionnelle du trèfle des collecticiels. Les propriétés notables du métamodèle Clover sont les suivantes : D'une part, le découpage fonctionnel des composants du métamodèle Clover, selon le modèle de conception du trèfle, permet de rendre explicites des éléments de conception du collecticiel au sein de l'architecture logicielle. Ce point est crucial si l'on considère la situation pivot de l'architecture logicielle au sein de cycle de vie du logiciel. En effet, l'architecture logicielle constitue le point de passage entre deux mondes : celui de la conception du système interactif et celui de la réalisation logicielle. Aussi il est primordial que les modèles d'architecture, comme le métamodèle Clover, intègrent des concepts et véhiculent des propriétés des deux mondes, conception et réalisation logicielle du collecticiel. D'autre part, le métamodèle Clover résulte d'une combinaison motivée de modèles d'architecture existants, sélectionnés pour leurs propriétés complémentaires. En particulier, notre métamodèle hérite des propriétés de généricité du modèle en couches de Dewan. De plus notre métamodèle affine les composants et les événements échangées selon les trois facettes du trèfle des collecticiels : par conséquent la modularité et donc la modifiabilité du code obtenu sont accrues.

Un modèle d'architecture Clover, instance du métamodèle, a été appliqué à la conception logicielle du système CoVitesse. Ce modèle affine le Noyau Fonctionnel en deux composants, l'un partagé, l'autre répliqué. Ces deux composants constitutifs du Noyau Fonctionnel sont à leur tour décomposés selon les trois facettes du trèfle des collecticiels, pour obtenir trois sous-composants encapsulés par une interface logicielle. Cette interface a pour rôle de masquer la décomposition interne du composant selon les facettes du trèfle. Nous sommes en cours de développement d'une plate-forme logicielle, la plate-forme Clover, pour la réalisation de collecticiels construits selon notre modèle d'architecture Clover. Cette plate-forme repose sur la réutilisation de fonctions et de composants organisés suivant les trois facettes du trèfle des collecticiels.

RÉFÉRENCES

1. Bass, L., and al., A Metamodel for the Runtime Architecture of an Interactive System, The UIMS Tool Developers Workshop, dans *SIGCHI Bulletin*, 1992, p. 32-37, ACM Press.
2. Calvary, G., Coutaz, J., Nigay, L., From Single-User Architectural Design to PAC*: a Generic Software Architecture Model for CSCW, dans *Proceedings of CHI'97*, 1997, p. 242-249, ACM Press.
3. CoVitesse, http://iihm.imag.fr/demos/CoVitesse/
4. Coutaz, J., Nigay, L., dans *éditeur C. Kolski, Analyse et conception de l'IHM : Interaction Homme-Machine pour les SI*, 2001, Hermès.
5. Dewan, P., Architecture for Collaborative Applications, dans *M. Beaudoin-Lafon editor, Computer Supported Cooperative Work*, 1999, John Wiley & Sons Ltd.
6. Graham, T.C.N., Urnes, T., Nejabi., R., Efficient distributed implementation of semi-replicated synchronous groupware, dans *Proceedings of UIST'96*, 1996, p. 1-10, ACM Press.
7. Laurillau, Y., Synchronous Collaborative Navigation on the WWW, dans *Extended Abtracts of CHI'99*, 1999, p. 308-309, ACM Press.
8. Laurillau, Y., Nigay, L., Modèle de Navigation Collaborative Synchrone pour l'Exploration des Grands Espaces d'Information, dans *les actes de la conférence Francophone IHM'00*, 2000, p. 121-128, CRT ILS&ESTIA .
9. Li, D., Muntz, R., COCA: Collaborative objects coordination architecture. dans *Proceedings of CSCW'98*, 1998, p. 179-188, ACM Press.
10. Nigay, L., Coutaz, J., Software architecture modelling: Bridging Two Worlds using Ergonomics and Software Properties, dans *P. Palanque & F. Paterno editors, Formal Methods in Human-Computer Interaction*, 1997, p. 49-73, Springer Verlag.
11. Nigay, L., Coutaz, J., A Generic Platform for Addressing the Multimodal Challenge, dans *Proceedings of CHI'95*, 1995, p. 98-105, ACM Press.
12. Nigay, L., Vernier, F. Design Method of Interaction Techniques for Large Information Spaces, dans *Proceedings of AVI'98*, 1998, p. 37-46, ACM Press.
13. OMG, Object Management Group & CORBA, http://www.omg.org/
14. Patterson, J.F., A taxonomy of Architectures for Synchronous Groupware Applications, dans *Workshop on Software Architectures for Cooperative Systems of CSCW'94*, 1994.
15. Roseman, M., Greenberg, S., GroupKit: A Groupware Toolkit for Building Real-Time Conferencing applications, dans Proceedings of CSCW'92, 1992, p. 43-50, ACM Press.
16. Salber, D., De l'interaction homme-machine individuelle aux systèmes multi-utilisateurs, *Thèse de l' Université de Grenoble 1*, France, 1995, 305 pages.
17. Whittaker, S., Nardi, B., Bradner, E., Interaction and Outeracion Instant Messaging In Action, dans *Proceedings of CSCW'00*, 2000, p. 79-88, ACM Press.