Table des matières

I. Introduction

Les concepteurs des premiers ordinateurs, tel Vanevar Bush à la fin de la deuxième guerre mondiale, imaginaient déjà permettre la communication et la collaboration entre les personnes. Quarante anx plus tard, les systèmes collaboratifs commencaient a être formalisés et généralisés : le modèle Dexter a apporté une base solide sur laquelle les systèmes peuvent gagner en flexibilité, en capacité d'organisation, avec les travaux de Borghoff, puis en interopérabilité, avec le Protocole Hypermedia Ouvert.
De nombreux systèmes existent, par exemple CHIPS et XCHIPS, présentés dans l'article 'Organiser l'espace de travail partagé de l'entreprise par le biais d'Hypermedia Coopératif basé sur des composants' (Organizing Shared Enterprise Workspace Using Component-Based Cooperative Hypermedia), qui permettent la gestion dynamique de l'organisation, et l'extension des outils et données lors de l'exécution : leur usage prévu dans des cadres fonctionnels, tel l'entreprise étendue, montre la validité des applications développées.
Pour faire face aux besoins et améliorer le temps et la qualité de la conception, les Motifs de conception (Design Pattern) sont de plus en plus utilisés. Mais ils doivent encore formalisés, et classifiés. L'article 'Communiquer la connaissance de conception avec des motifs de conception pour la technologie des collecticiels' (Communicating Design Knowledge with Groupware) présente une classification de ces motifs de conceptions pour le travail coopératif.
Nous essayerons également d'établir une critique de chacun de ces articles. PP

II. Contexte

A. Le modèle initial : Dexter ([HA94] cité par [RH01])

A la fin des années 1980, les systèmes collaboratifs sont en majorité des systèmes d'édition de documents textuels. Ils se basent sur le principe de l'hypertexte, qui permet la liaison entre documents et permettra le succès du web. L'hypertexte offre aussi la possibilité de manipuler ces liens, et c'est cette propriété qui sera utilisée dans les systèmes collaboratifs.
Le modèle Dexter a été présenté au Workshop “NIST Hypertext Standardization” en 1990, puis dans [HA94]. Il s'agit de formaliser les abstrations présentes dans les systèmes hypertextes existants, pour dégager la structure de tels systèmes. Il en résulte une architecture en trois couches :

- Couche d'éxecution 'runtime', qui supporte l'interaction de l'utilisateur avec l'hypertexte,
- Couche de stockage 'storage', ensemble de liens et de composants, qui forment le réseau hypertexte. Les composants contiennent les données. Il n'y a pas de modélisation du type de document concerné.
- Couche contenu des composants, 'within-component', dans laquelle se trouvent le contenu et la structure des composants, en fonction du type de données.

Le modèle ne définit aucun type de données, mais permet l'utilisation de types extérieurs, tel ODA (Office Document Architecture).
En plus des couches, le modèle contient la définition des interfaces entre ces couches : mécanisme d'ancrage (entre les couches de stockage et contenu de composant, pour adresser le contenu d'un composant), et spécification de présentation (entre la couche contenu de composant et éxecution). Il défini également une terminologie commune (par exemple en utilisant le mot composant plutôt que noeud, dont l'acceptation varie d'un auteur à l'autre).
La figure 1 montre la structure d'une application collaborative selon le modèle Dexter, la figure 2 un exemple de présentation de document.

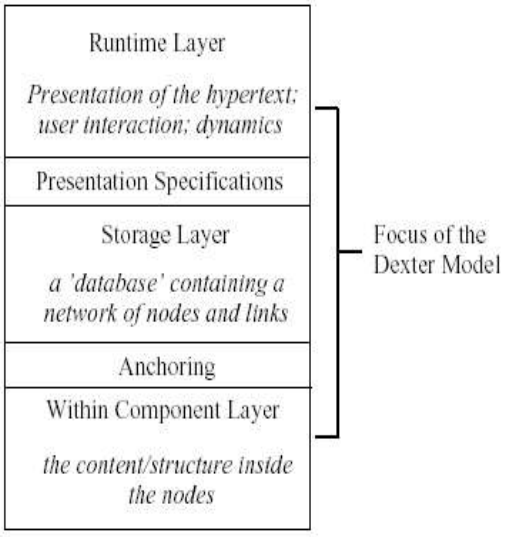


Illustration 1 Structure d'une application collaborative selon le modèle Dexter

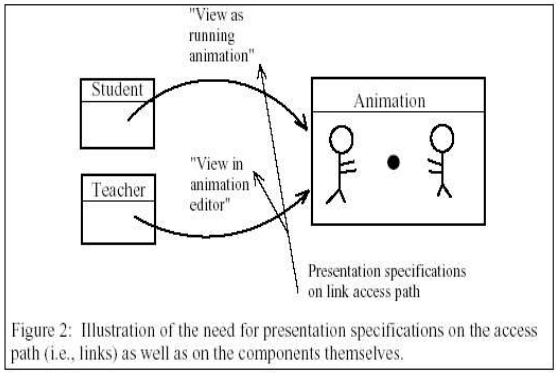


Illustration 2 Exemple de présentation de document

Le modèle Dexter, en combinant les caractéristiques de différentes applications existantes, se révèle plus puissant que nécessaire, et ouvre la voie à des systèmes collaboratifs plus complexes : il contient des liens n-aires, multi-directionnels, pointant vers des liens, introduit la notion de composite (document complexe) qui est encore absent de beaucoup de systèmes. De plus, en réalité, les liens n'ont pas toujours une source et une destination explicite.

B. La gestion dynamique de la structure de données ([BT93] non cité)

Le modèle Dexter introduit une forte formalisation de la structure des systèmes collaboratifs. Il permet de manipuler la structure des documents, en plus de leur édition. Mais un certain nombre de lacunes se font vite sentir : le travail collaboratif a besoin d'être organisé, et d'autres types de données que le texte doivent être supportées.

Pour y remédier, Borghoff [BT93] introduit la possibilité de gérer non seulement la structure de document , mais aussi l'organisation du travail en équipe, par les hyperliens. Il sépare également la structure de l'application, qui est indépendante du média, et l'editeur de contenu, média-dépendante. Bien que le nom ne soit pas cité, on a donc une application hypermedia : les liens ne concernent plus seulement le texte, mais tous les types de documents.

La figure 3 présente la structure de l'application IRIS, qui est le prototype réalisé pour montrer la faisabilité et l'efficacité de ces principes. La structure est faite en deux couches, car l'information de structure est gérée parallèlement aux contenus. Cette structure est éditable, et donne une vision selon plusieurs points de vue des documents et des actions des utilisateurs. Elle donne aussi une documentation de projet explicite, pour gérer l'aspect organisationel. Un historique existe, tant pour l'information de structure que pour le contenu.

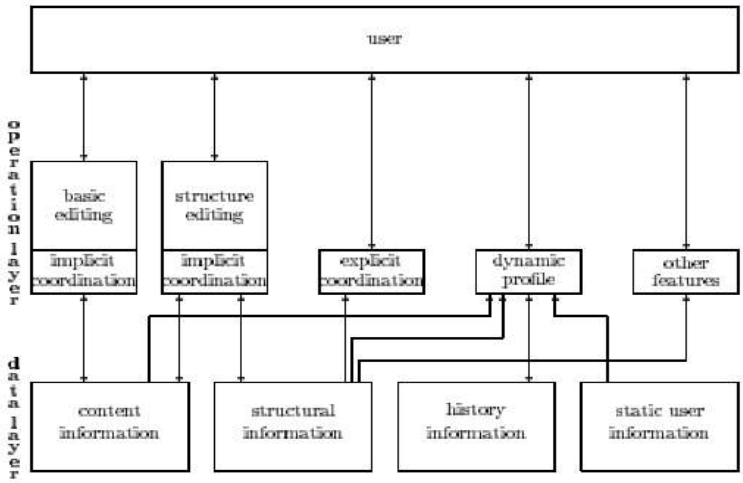


Illustration 3 Structure de l'application IRIS

Bien qu'il se soit pas cité dans les articles proposés, le travail de Borghoff introduit des principes qui sont pertinents pour l'ensemble des applications collaboratives : support de plusieurs types et données, et gestion de l'organisation de travail.

C. Le protocole de communication : l'Hypermedia Ouvert ([RW99] cité par [RH01], [NU97] non cité)

Les applications collaboratives présentées jusqu'à présent permettent de réaliser un travail commun, sur des types de données variées, en organisant ce travail. Mais elles confinent les utilisateurs dans une application donnée. L'avènement du web encourage la création de systèmes qui ne soient plus monolithiques et fermés, mais ouverts et distribués. Il devient nécessaire de mettre à disposition un framework qui soit ouvert, et qui permette l'interopérabilité entre les systèmes hypermedia. Un groupe de travail se met en place dans le cadre les conférences ACM Hypertext, de 1995 à 1999 [RW99],[NU97]. Il propose le protocole hypermedia ouvert (OHP – Open Hypermedia Protocol). L'objectif est d'autoriser la mise à disposition de l'information à un large public, et la réutilisation de logiciels.
La réalisation de ces objectifs implique non tant la réalisation d'un framework disponible, mais surtout la définition d'une interface de communication. La figure 4 montre le principe du protocole OHP. Pour permettre l'usage d'une telle interface, il s'est avéré ensuite nécessaire de préciser la structure des systèmes à intégrer.
Deux types de clients existents : clients légers, qui tirent parti de la diffusion des navigateurs web, et clients lourds, qui permettent un fonctionnement en mode déconnecté, et sont donc adaptés aux systèmes qui se veulent pervasifs.

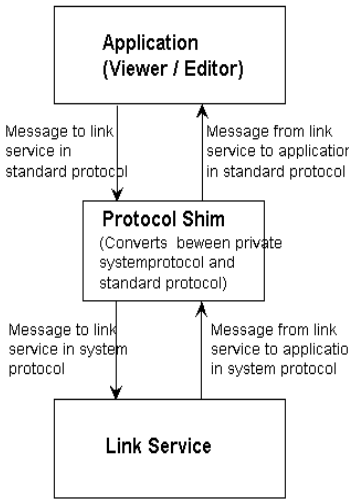


Illustration 4 Principe du protocole OHP

Il n'y a pas de version standardisée du protocole, uniquement un draft. D'après S. Reich, à qui nous avons demandé (contact mail) ce qu'il en était, OHP est vieilli, et n'est plus tout à fait d'actualité. Les travaux se sont ensuite orientés vers la formalisation de la structure des applications, et en particulier vers la définition de Motifs de conceptions (Design Pattern) mettant en oeuvre des composants. L'utopie de l'application des principes du web au monde des applications collaboratives et coopératives aurait alors été abandonné pour favoriser le développement de systèmes cooperatifs pour l'entreprise ou l'enseignement. PP
III. Article 1

Le protocole hypermedia ouvert, s'il permet l'interopérabilité entre systèmes collaboratifs, n'offre pas une intégration suffisante pour un travail efficace entre plusieurs personnes : il est indispensable de mettre à disposition des collaborateurs non seulement les données, mais aussi les outils. L'article proposé [RH01] offre une solution à ce problème.
Il s'agit d'une réponse à un appel d'offre européen, dans le cadre du projet External, qui consiste à créer un framework pour l'entreprise étendue [SO00]. La solution proposée reprend en le complétant le framework hypermedia CHIPS, développé auparavant par les auteurs [WH00]. Il introduit un certain nombre de Motifs de Conception pour le travail collaboratif.
Les problèmes liés à la mise en oeuvre d'une approche hypermedia des espaces d'entreprise partagés sont de quatre types :
- accès et manipulation coopératifs des objets hypermedia partagés,
- modélisation et exécution coopératifs du travail,
- extensibilité des objets hypermedia pendant l'utilisation,
- extensibilité des outils et vues pendant l'utilisation.

Les deux premiers aspects sont résolus par CHIPS, les deux suivants étant l'objet de XCHIPS.

A. CHIPS (présenté dans [WH00])

i. Gestion dynamique de la structure de travail

Le modèle d'Hypermedia collaboratif basé sur composants contient deux niveaux : un Espace de Composants Partagés, où se trouvent les outils de manipulation des données, et un Espace d'Objets partagés, où se trouvent ces données. La figure 5 illustre cette structure. Le travail doit pouvoir avoir lieu de facon synchrone ou asynchrone, ce qui nécessite un grain fin au niveau de la composition des données.
L'Espace de Composants Partagés doit contenir des définitions de processus de travail coopératif, et pouvoir être structuré de manière flexible, et évolutive. Les processus font donc partie des données modifiables, afin d'offrir la modularité (tailorability) de l'environnement de travail, de même que les documents ou les équipes. Les composants contiennent des outils pour manipuler, naviguer entre, rechercher les objets, éditer le contenu des objets partagés.
Les types d'Objets partagés sont des objets de contenu, de processus, ou d'équipes. Ces objets peuvent être des composites, c'est à dire des objets complexes contenant d'autres objets. Des liens peuvent exister entre tous ces objets.
Les objets de contenu contiennent des documents hypermedia avec des liens.
Les processus sont représentés par un ensemble de tâches hypermedia liées par des flux : un noeud commence dans un objet composite de tâche source, et va dans un objet composite de tâche destination quand la tâche source est achevée. Les classes composites associées sont la classe de gestion des tâches (Task) et la classe de lien qui représente les flux (Flow).
Les équipes contiennent les utilisateurs, ainsi que les organisations des équipes.

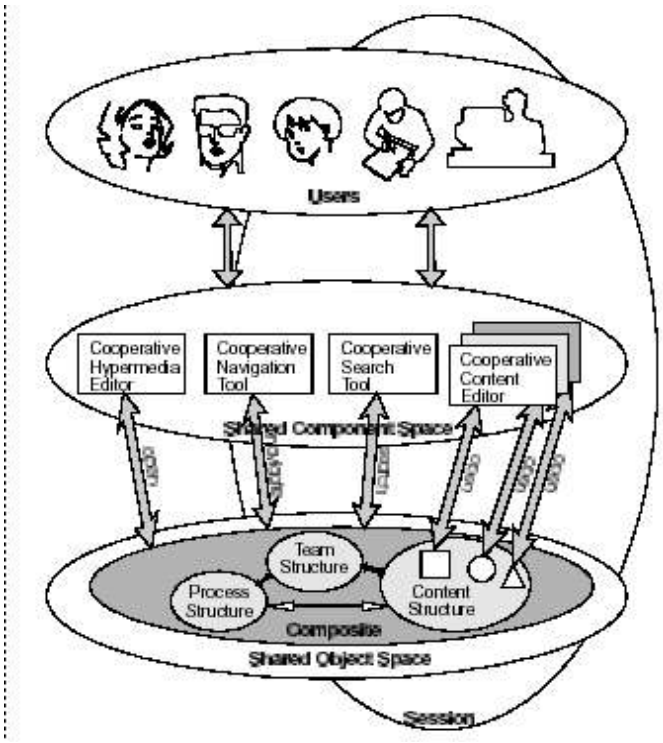


Illustration 5 Hypermedia collaboratif basé composant

Les utilisateurs travaillent par sessions. Ces sessions sont transversales à l'organisation en couche du framework, et lient un ensemble d'utilisateur, des composants de travail collaboratif (outils), ainsi que des Objets partagés (données).

ii. Motifs de Conception

La définition de la structure du système collaboratif n'est pas suffisante. Il est indispensable également de mettre à disposition des Patterns pour son organisation interne, afin de formaliser les fonctionnalités du framework, par exemple la gestion des processus.

Le Pattern 'Support de processus pour Hypermedia partagé' (Shared Hypermedia with process support) offre une structure d'information riche, qui reflète la sémantique souhaitée par l'utilisateur lors de la création de la structure d'information et de la description du processus. Il est représenté dans la figure 6.

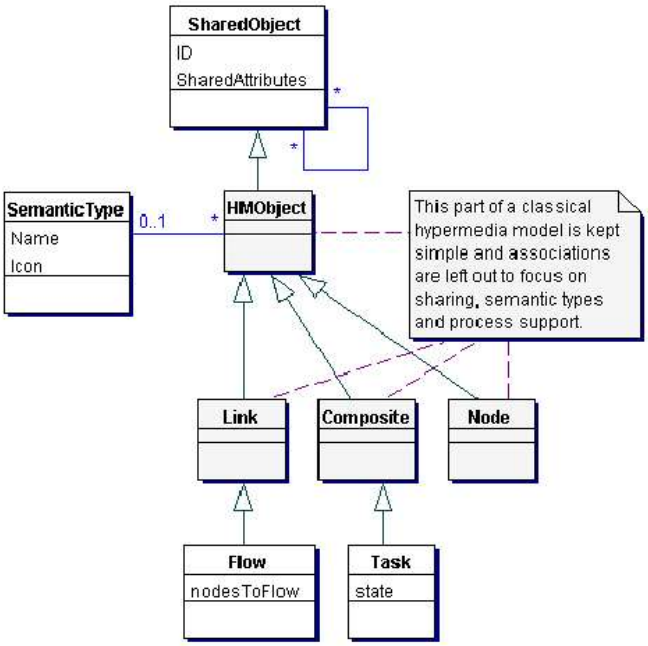


Illustration 6 Pattern pour le support de processus

Le mécanisme de gestion des tâches est présenté dans le paragraphe précédent.

B. XCHIPS

i. Extension à chaud du système

L'originalité de XCHIPS est la possibilité d'étendre à chaud le système, et donc de faire évoluer les applications sans devoir redémarrer le système. Il peut s'agir d'extension des outils, ou des types de données supportées. L'approche orientée composant prend ici tout son sens, car ceux-ci, en tant que packages logiciels exécutables avec interface définie et publiée, peuvent être relativement aisément intégrés pendant l'exécution d'un programme. Les composants orientés Groupware sont situés sur un serveur partagé, et peuvent être téléchargés si besoin par l'utilisateur. Ils sont particulièrement adaptés au travail collaboratif, car ils peuvent être notifiés des modifications dans les objets de données qui les concernent. Ces mécanismes ont été définis par Tietze lors de sa thèse de doctorat [TI01].

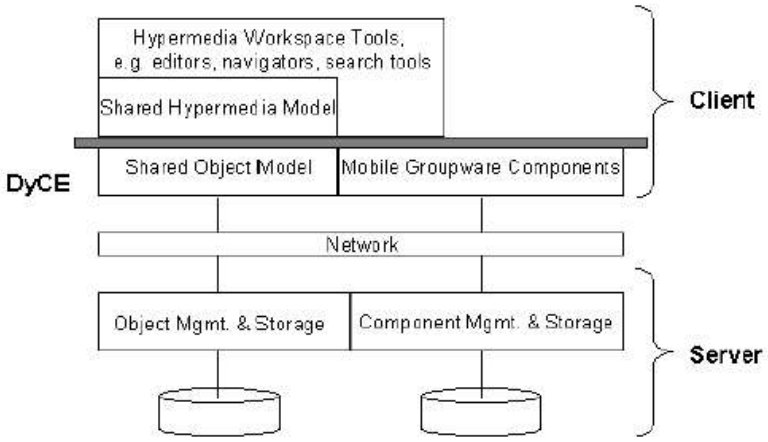


Illustration 7 Architecture de XCHIPS, basé sur DyCE

La figure 7 montre l'architecture utilisée : un serveur DyCE, qui met à disposition les composants orientés travail collaboratif (Groupware), et une implémentation des Motifs de Conception 'Support de processus pour Hypermedia partagé', et Emballage (Wrapper).

ii. Motifs de Conception

Deux Motifs de Conception sont présentés : 'Composant de travail Collaboratif' (Groupware component) [TI01], et Emballage. Le premier (figure 8) permet la mise en place de composants pour le travail collaboratif, le deuxième (figure 9) le chargement à chaud de nouveaux objets ou services.

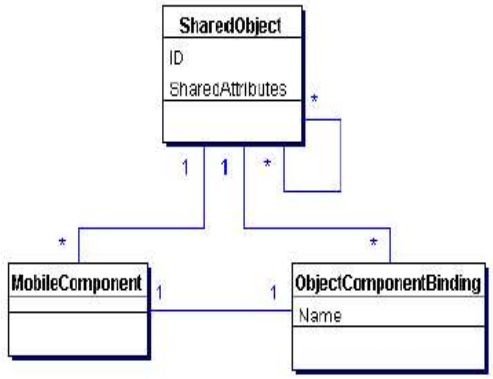


Illustration 8 Motif de Composant de Travail collaboratif

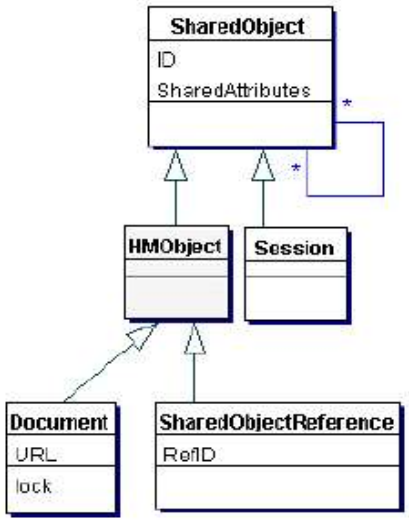


Illustration 9 Motif Emballage pour objet partagé

Le Motif de conception 'Composant pour le Travail collaboratif' a été présenté au paragraphe précédent. Chaque composant mobile pointe vers un objet racine, et accède par ce biais à d'autres

objets partagés. De cette manière, plusieurs instances accèdent au même objet, ce qui permet l'accès à des fonctionnalités, des vues différentes de l'objet. La liaison est réalisée par un lien (ObjectComponentBinding).
Le Motif de Conception 'Emballage' permet à un objet partagé (de type SharedObjectReference) de pointer vers plusieurs objets partagés (données ou session), et l'invocation automatique de l'instance de lien (ObjectComponentBinding) liée à l'objet ouvert. Le mécanisme de cette invocation automatique n'est pas précisée. Le Pattern contient aussi des Documents (pour la gestion de l'accès et de l'awareness), et des sessions.
La figure 10 montre le lien entre les deux patterns ci-dessus.

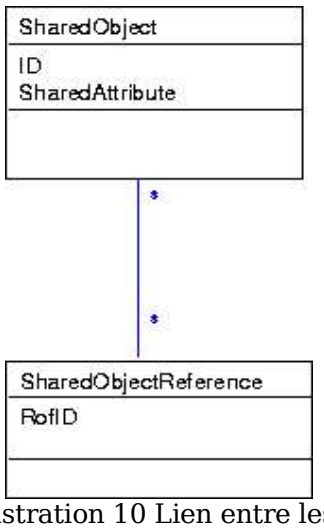


Illustration 10 Lien entre les deux patterns précédents

C. Critique

Ce travail est la présentation d'un prototype qui marque l'aboutissement de dix ans de travaux de la part des auteurs dans le domaine du travail collaboratif. Celà leur permet de présenter une architecture très complète. Le cahier des charges étant fixé par l'appel d'offre auquel ce travail répond, on peut affirmer que les fonctionnalités proposées correspondent à des besoins réels dans le cadre des entreprises étendues.
De plus, les Motifs de Conception proposés sont complétement indépendants de l'implémentation, et peuvent donc être utilisés dans des contextes très variés.
L'originalité de ce prototype réside dans la possibilité d'étendre le système, sans interrompre les sessions de travail collaboratif synchrone.

Toutefois, on peut émettre un certain nombre d'objections. Il ne s'agit pas d'un travail véritablement original, dans la mesure où il constitue l'extension de deux plate-formes réalisées auparavant par les auteurs : CHIPS, pour le support collaboratif, et DyCE, pour les composants orientés Groupware. Il s'agit plus d'un travail de synthèse de modèles existants. Le travail réalisé est néanmoins tout à fait conséquent, et n'est rendu possible que par l'expérience et la pré-existance de ces plate-formes.
Le statut de synthèse rend le prototype proposé peu innovant : le cahier des charges étant fixé, il n'y a pas d'introduction de concepts nouveaux, et aucune perspectives d'évolution n'est présentée, si ce n'est celles dont l'usage aura montré la nécessité.
Une autre limite de cet système est l'absence de possibilité de support des types de fichiers extérieurs en travail synchrone. Le synchrone nécessite que l'application soit réalisée pour ça.

En ce qui concerne les modèles utilisés, on peut se demander si la référence forte au modèle Dexter, qui date de la fin des années 1980, ne limite pas l'introduction d'idées innovantes. De même, les auteurs font référence presque uniquement à des publications de la communauté Hypermedia. La comparaison avec des systèmes coopératifs issus d'autres communautés, par exemple le workflow, serait sans doute pertinente.

Par ailleur, un certain nombre d'éléments pourraient être intégrés dans cette application afin d'étendre son champs d'utilisation : gestion de la mobilité, adaptation de l'interface au terminal de l'utilisateur, etc.

Malgré ces quelques remarques, qui sont parfois inspirées de commentaires des auteurs eux-même, la solution proposée semble à la fois pertinente quand aux objectifs fixés, et évolutif de telles sorte que l'intégration d'aspects complémentaires, telles les interfaces ou la mobilité, doit pouvoir se faire sans remise en cause profonde du modèle.

PP

IV. Article 2

A. Les Motifs de Conception pour la collaboration [LS04]

Même si le développement des applications coopératives est basé sur les même principes que le développement des applications « mono-utilisateur », il y a plusieurs aspects qui n'ont pas leur représentant dans le monde de la conception classique et qui offrent des challenges supplémentaires pour le génie logiciel. Les gros défis du travail coopératif et réparti sont:

1 Contraintes de la **connexion réseau,**

2 **Entrée simultanée/parallèle** de données,

3 **Fonctions** pour travailler en groupe,

4 **Partage d**es données (le plus difficile et le plus important).

Importantes pour le développement des collecticiels sont aussi :

- **Formation** d'anciens et nouveaux développeurs dans la conception et l'implémentation des applications coopératives.
- **Réutilisation** de solutions existantes.

Pour faciliter le développement des applications coopératives et réparties et abstraire (un peu) de ces différences de conception et d'interaction l'approche « framework » est classique. Malheureusement, utiliser ces frameworks comme GroupKit [1], COAST [2] ou DreamObjects [3] n'est pas vraiment facile, apprendre les mécanismes d'un framework est une tâche difficile et de longue haleine. Donc cette approche n'est pas suffisante pour travailler efficacement notamment dans des petits et moyens projets.

Les solutions existantes pour concevoir et implanter une application coopérative à base de frameworks ont aussi des inconvénients forts : La (ré)utilisation des solutions existantes est limité par le **langage de codage** et l'**architecture du système** dans laquelle le framework offre ces services. Pour écrire une application à base de COAST par exemple, les développeurs sont obligé d'utiliser le langage Smalltalk et les librairies des classes associées, mais Smalltalk n'est normalement pas une partie du programme de formation informatique classique. En plus, la plupart des frameworks ne sont **pas compatibles** ou facilement connectables, c'est donc difficile pour les programmeurs de profiter des avantages des divers frameworks. Généralement, les frameworks offrent une API, mais l'implantation est invisible (**boîte noire**) et la **documentation** est rarement suffisante.

Les contraintes, qui sont valables pour toutes les approches orientées codage limitent automatiquement la réutilisation des solutions. Une approche pour surmonter ces difficultés est de déplacer les questions de coopération et répartition de l'implantation dans le cadre de la conception. Les chercheurs Stephan Lukosch et Till Schümmer de l'université de Hagen (Allemagne) proposent d'utiliser une **collection de motifs de conception (MdC)**, (« Design Pattern Language »), spécifiquement adaptée aux challenges de travail coopératif et réparti.

Dans le monde du génie logiciel, un « Motif de Conception » est une famille de solutions pour un problème concret de la conception d'un logiciel. Le motif n'est pas la solution directe (une

pièce de codage par exemple) mais une idée de la construction de la solution (Répartition dans plusieurs classes spécialement connectées et dépendantes, par exemple). Avec MdCs, il plus facile de discuter d'un problème de conception dans un groupe de développement. En utilisant les motifs, on peut éviter une « dérive de conception » et améliorer la qualité et lisibilité du codage. Le but final est notamment de réutiliser la connaissance du génie logiciel. Etant indépendant du langage de codage spécifique, une collection de motifs de conception (une « famille » de MdCs) peut être considérer comme une « Lingua franca » pour la conception.

Les chercheurs de Hagen ont décidé de se focaliser plus dans la réutilisation des solutions de conception que dans la réutilisation du codage existant. Cette approche mène presque inévitablement à la conception d'un langage des motifs de conception et ils ont fait ressortir dix motifs pour la gestion des objets partagés dans les systèmes coopératifs.

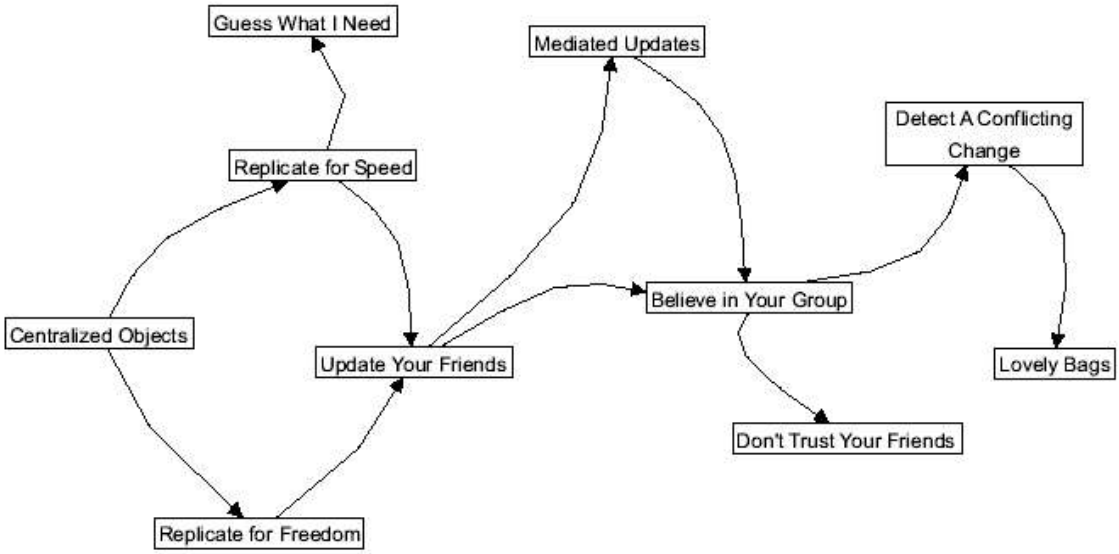


Illustration 11 Carte des Motifs de Conception

Suivant cette figure les motifs sont en détail :

1 **Objets Centralisés** *(Centralized Objects)* : Un serveur central qui est connu par tous les utilisateurs et offre les objets/données. C'est une approche classique qui est par exemple utilisé dans le système WebDAV (Web-based Distributed Authoring and Versioning). Ici, un disque dur en ligne accessible par une extension du protocole HTTP (S) peut supporter l'échange de fichiers entre les travailleurs.

2 **Dupliquer Pour La Liberté** *(Replicates For Freedom)* : Pour avoir un accès aux données indépendant d'une connexion au réseau, les données sont dupliqué localement (cache). Ce principe est connu par le mode « surfer hors ligne » des navigateurs web. Un bon exemple est aussi le iDisk, qui est répliqué localement dans le système Mac OS X actuel , et est donc toujours accessible. Après une réconnection un processus de synchronisation est obligatoire.

3 **Dupliquer Pour La Vitesse** *(Replicate For Speed)* : Augmenter les temps de réponse

d'une application en dupliquant les objets locaux et assurer la consistance avec les motifs *Fournir Les Mises A Jour* ou *Mettre A Jour Vos Amis*. Contraintes : l'Objet doit être duplicable, le délai d'initialisation est grand, les coûts de communication pour la duplication sont grands, la consistence est difficile à maintenir. Un exemple pour ce motif est les serveurs de cache en ligne, comme le système *Oracle 10g Web Cache.*

4 **Deviner Mes Besoins** *(Guess What I Need)* : Non seulement les objets utilisés sont dupliqué localement, mais aussi les objets proche, associé ou connecté. Pour ces objets, la probabilité d'être demandés aussi par l'utilisateur dans un futur proche est très élevée, on peut donc améliorer les temps de réponse pour ces demandes (c'est à dire profiter de la localité des objets en espace et en temps). Ce stratégie est par exemple utilisée dans les systèmes de cache de mémoire où non seulement une donnée est 'cachée' mais le cache complet est échangé.

3 **Mettre A Jour Vos Amis***(Update Your Friends)* : Distribuer les changements locaux pour assurer la consistance globale (update messages). Le principe est de diffuser des messages de mise à jour à tous. Il est par exemple utilisé dans le protocole de routage BGP (Le routage des paquets en ligne est un décision indépendante de chaque routeurs, mais tous les routeurs doivent coopérer pour offrir le service, donc le routage dans le web IP peut être considèré comme une application répartie et coopérative de grande échelle). Contraintes : le groupe de travail doit être connu. Assure la bonne réception du message de mis à jour par chaque membre du groupe. Eviter des conflits par travail parallèle dans le même objet. Non seulement un mode « push » mais aussi un transfert « pull » est utilisable (en combinaison avec les motifs *Dupliquer Pour La Liberté* et *Dupliquer Pour La Vitesse*).

4 **Fournir Les Mises A Jour** *(Mediated Updates)* : Une instance centralisée (le médiateur) gère les mises à jour des objets et informe les clients connectés. Par ce principe, le travail administratif pour les mises à jour peut être minimisé. Un exemple peut être le système de gestion des configurations logiciels (software configuration management system) Perfoce [4], un système pour gèrer le travail réparti et les versions de codage source utilisé par des grands entreprises IT comme SAP AG, Google, palmOne ou la Société Générale.

5 **Croire En Votre Groupe***(Believe In Your Group)* : Mettre en jour vos changements immédiatement et annuler (« undo ») si besoin.

6 **Ne Faire Pas Confiance En Vos Amis** *(Don't Trust Your Friends)* : Pour éviter les conflits de travail, un barrage distribué (distributed lock) est utilisé. Ce stratégie est utilisée pour garantir la consistance des données dans une base de données sans transactions.

4 **Détecter Un Changement Conflictuel** *(Detect A Conflicting Change)* : Dans tous les cas, s'il y a un message de mise à jour pour un objet utilisé et modifié par l'utilisateur, un processus pour détecter les conflits est lancé. C'est fait par exemple très facilement pendant le déroulement de la commande *cvs update* du système de gestion des versions logiciels CVS [5].

6 **Sac Gentil** *(Lovely Bag)* : Utilisation d'un « sac» (un conteneur intelligent) pour

stocker les objets et pour gérer l’accès concurrent dans une structure de données difficile.

Pour tester cette approche MdC dans le monde de collecticiels, les chercheurs Lukosch et Schümmer ont fait une expérimentation dans la quelle deux groupes d'étudiants devaient développer un jeu numérique coopératif (réparti), le premier groupe avec le framework COAST, le deuxième groupe « libre » mais en utilisant les motifs de conception proposés. Ils ont trouvé que la programmation en utilisant les MdCs et en travailler directement en base d’un langage de codage connu par les programmateurs est pour des projets petits et moyens plus rapide que de travailler avec un framework puissante mais complexe.

Pour le futur, les chercheurs ont planifié d’intensifier leur recherches dans les motifs de conception coopératifs et leurs applications.

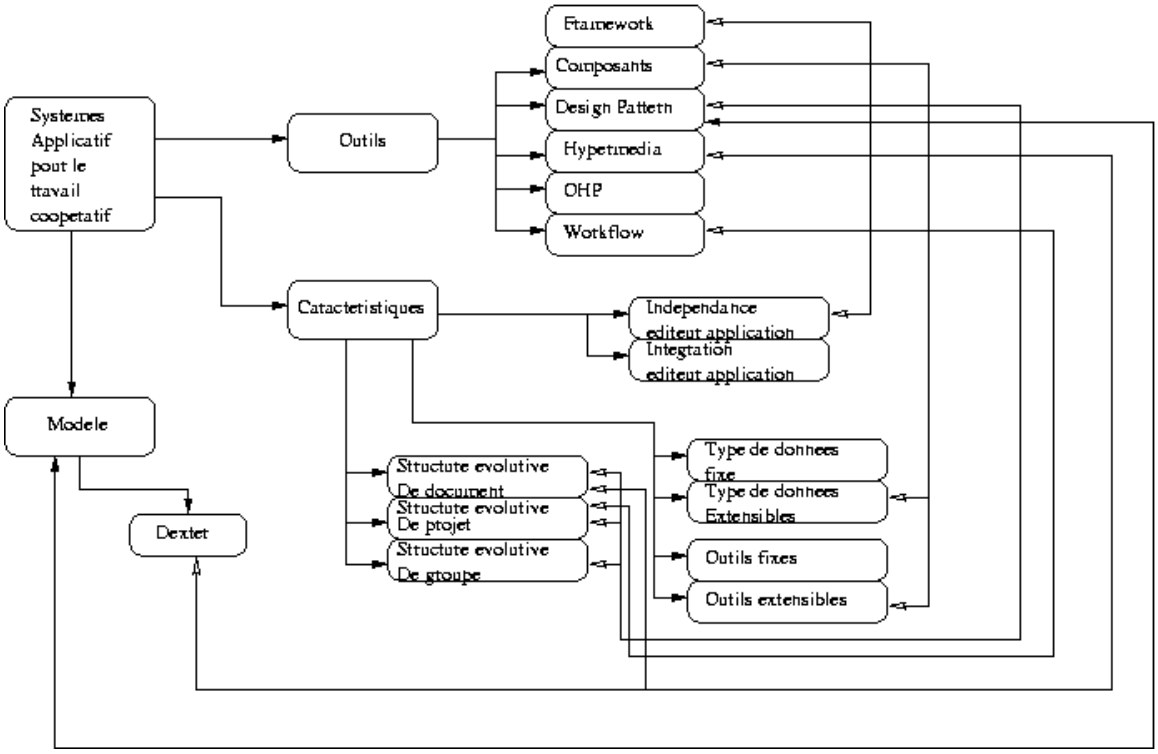
B. Critique

En gros, les problèmes des motifs de conception ici sont les mêmes que dans le monde du génie logiciel classique : les approches sont très bien formalisées dans le cadre de la conception, mais la réalisation ou plutôt l'implantation est pas triviale. Même si la plupart de ces motifs seulement manifeste et formalise un approche logique et intuitive c'est difficile au début de penser en terme de structures de motifs. Une solution (qui a déjà commencé à être réalisée) est l'intégration des idées de ces motifs dans les frameworks existants pour que leur utilisation (et avec ça aussi la possibilité de profiter de leurs avantages) puisse être transparente pour le développeur.

PB

V. Conclusion

Carte de concepts :



Après avoir été concu comme des frameworks rigides, les systèmes collaboratifs et coopératifs sont de plus en plus fondés sur des Motifs de conception, le framework ne fournissant plus que des services élémentaires. Mais ces Motifs ne sont pas encore tous forcément formalisés, et donc pas réutilisables. Les articles présentés illustrent cette tendance à utiliser des motifs, qui sont plus légers, plus maléables. Ils concourent à leur formalisation, et à leur classification, afin de permettre une conception plus facile et plus rapide de plate-formes coopérative flexibles, et extensibles. Mais il n'existe pas encore de recensement des différents motifs, ni par conséquent de méthode de conception formalisée qui permettrait de créer, en fonction des besoins, des applications coopératives et collaboratives à la demande.

De nombreuses plate-formes existent, qui permettent la mise en oeuvre de systèmes collaboratifs et coopératifs, tant en environnement d'apprentissage que dans le cadre de l'entreprise étendue. La limite actuelle de ces systèmes est le faible support de la mobilité, et le manque de pervasivité. C'est dans cette direction que s'orientent les recherchent actuelles.

PP

VI. Bibliographie

Il s'agit des articles et documents utilisés pour la présentation, auquels les articles à présenter font référence, ou complémentaires.

[RH01] Organizing Shared Enterprise Workspace Using Component-Based Cooperative Hypermedia, J. Rubart, J. Haake, D. A. Titze, W. Wang, HT'01.

[HA94] The Dexter Hypertext Reference Model, Halasz, F., and Schwartz, M. In Communications of the ACM 37, 2, 1994, 30-39. cité par [RH01].

[BT93] Structure Management in the collaborative multimedia editing system Iris, Uwe Borghoff and G. Teege, Proc. 1st Int. Conf. on Multi-Media Modeling (MMM,'93). Non cité.

[RW99] Addressing Interoperability in Open Hypermedia: The Design of the Open Hypermedia Protocol, Sigi Reich, Uffe K. Wiil, Peter J. Nürnberg, Hugh C. Davis, Kaj Grønbæk, Kenneth M. Anderson, David E. Millard, Jörg M. Haake, In The New Review of Hypermedia and Multimedia, Volume 5, Taylor Graham, 1999. cité par [RH01].

[NU97] Nürnberg, P. J., HOSS: An Environment to Support Structural Computing, Ph.D. Dissertation, 1997. non cité.

[WH00] Tailoring Groupware: The Cooperative Hypermedia Approach, Weigang Wang and Jörg M. Haake,In Computer Supported Cooperative Work: The Journal of Collaborative Computing, Vol. 9, No. 1, 2000, Kluwer. Cité par [RH01].

[SO00] Strømseng, K., Olsson, N., Haake, J., Scagno, G. Extended Enterprise Requirements, The EXTERNAL project deliverable D1, 2000, http://research.dnv.com/external/PW_Tools/PWD/3/31/ D/3-31-D-2000-01-1. Cité par [RH01].

[TI01] Tietze, D.A. A Framework for Developing Componentbased Co-operative Applications. Dissertation at TU Darmstadt, GMD Research Series, Germany, 2001. cité par [RH01].

[LS04] Stephan Lukosch and Till Schuemmer, Communicating Design Knowledge for Shared Object Management - Groupware: Design, Implementation, and Use, 10th International Workshop, CRIWG 2004, LNCS 3198 , Springer-Verlag Berlin Heidelberg, San Carlos, Costa Rica, 2004, pp. 223-237.

Où trouver des informations complémentaires ?

Conférences :

Conference on Human Factors in Computing Systems
Conférence ED-Media (Educational multimedia, hypermedia and telecommunications)
WWW Conference
International Conference on Distributed Computing Systems
International Conference on Extreme Programming and Agile Processes in Software Engineering
ACM Conference on Hypertext and Hypermedia
ACM Conference CSCW
ACM symposium on User interface software and technology
ACM SIGGROUP Conference on Supporting Group Work
ACM conference on Designing interactive systems
ACM Symposium on Operating Systems Principles

IEEE Symposium on Reliability in Distributed Software and Database Systems
ECOOP (European Conference on Object Programming)
EuroPLoP (European Conference on Pattern Languages of Programs)
International Workshop on Groupware
Workshop Adaptative Hypermedia

Journaux :
ACM Transactions on Computer-Human Interaction
ACM Transactions on Information Systems
ACM Computing Survey
IEEE Computer
IEEE Software
The New Review of Hypermedia and Multimedia
Computer Supported Cooperative Work: The Journal of Collaborative Computing

VII. Outils utilisés

Pour réaliser ce travail, nous avons utilisé les outils de travail collaboratif suivants :
- email,
- messagerie instantannée,
- WebDAV,
- forum (pour des raisons techniques, émulé par un fichier texte présent dans le répertoire WebDav).