Christophe GRAVIER

MRI-RST3 : Systèmes coopératifs : services et usages

pour Bertrand DAVID & Frank TARPIN-BERNARD

Synthèse de l'article :

"Reusing single-user applications to create multi-user Internet applications"
de Stephan Lukosch & Joerg Roth (2001)

Présentation du 2 février 2005

Introduction

La proposition des auteurs est de réutiliser les applications mono-utilisateur existantes (pour autant dire la quasi-totalité de l'existant applicatif aujourd'hui) pour les transformer en applications collaboratives. L'article offre une méthodologie ainsi qu'un « toolkit » dans le but de proposer une couche d'abstraction supplémentaire au développement prenant en charge le caractère collaboratif que l'on veut donner à l'application.

L'enjeu est bien évidemment double :

- Exploiter l'existant mono-utilisateur par la transformation
- Dans les applications futures, ne développer que la partie mono-utilisateur (i.e. le cœur de l'application) pour n'introduire la notion de groupe que par la suite.

Il convient de cadrer les débats : nous parlons ici du développement d'applications mettant en place le travail collaboratif et non pas sur l'organisation de celui-ci.

Ce papier vous permettra dans un premier temps de prendre connaissance de l'existant en terme de concept de développement des applications mettant en place le travail collaboratif. Par la suite, nous exposerons les propositions des auteurs, puis nous apporterons une conclusion sur l'apport de cet article pour la communauté de recherche sur le travail llaboratif assisté par ordinateur (noté TCAO par la suite).

1. L’être ou le paraître

Il est cependant très intéressant de noter que l’existant applicatif en terme de TCAO, ne suit pas l’idée de transformation des applications. En réalité, deux idées s’affrontaient déjà :

- le « collaboration awareness » que je traduirais par « être collaboratif »
- le « collaboration transparent » que je traduirais par « paraître collaboratif »

a. L’être

La première pensée qui vient à l’esprit lorsque l’on envisage de développer une application mettant en place un travail collaboratif, c’est d’établir un développement spécifique, appliquant les règles du travail collaboratif que l’on se fixe.

Cela a pour avantage indéniable de répondre précisément aux besoins fonctionnels, du fait de l’expression précise des besoins en terme de travail collaboratif. On espère donc que le résultat obtenu n’auront pas subit de forte dérive en regard des besoins « de collaboration » formulés.

De plus, il est clair que la conscience de groupe (« group awareness ») n’en sera que plus forte : les développements qui se seront attachés à respecter les règles fixées dans l’expression des besoins permettrons aux utilisateurs de se sentir agir dans un groupe (à la condition que les besoins exprimés conjugue correctement les besoins réels en terme de travail collaboratif).

Enfin, d’un point de vue ingénierie, la gestion des données distribuées sera intrinsèque : des modules auront été développés spécifiquement pour gérer la concurrence d’accès à une ressource (il convient d’utiliser le grain minimal « une donnée » dans ce cas d’une application collaborative).

A l’inverse, une telle solution supporte difficilement le coût financier engendré dû aux développements spécifiques, que ce soit initial ou en terme de maintenance de l’applicatif.

De plus, réutiliser les briques logicielles développées spécifiquement se révèle une tâche complexes du fait que ces briques logicielles répondent à des besoins très précis en terme de travail collaboratif, et ne seront donc pas forcément facilement transposable pour une autre application.

b. Le paraître

A l'inverse d'un développement spécifique, on pourrait imaginer de ne pas partager l'application mais l'environnement dans lequel s'exécute celle-ci.

La grande force de cette idée séduisante est que sa mise en place est triviale. De plus son coût est réellement très faible par rapport à un solution « collaboration awareness ».

De plus, il est évident que les utilisateurs ne seront pas dépaysés (l'application restant la même, ils ne se sentiront pas déroutés et conserveront leur habitudes de travail).

Une telle proposition n'existe pas sans quelques désagréments : si l'environnement est partagé, comment éviter que les utilisateurs ne soient pas des concurrents à l'accès à un contrôle graphique (e.g. ascenseur dans une fenêtre). On peut alors s'interroger si l'on a mis en place une solution collaborative ou une solution compétitive, qui diminuerait fortement la conscience de groupe, au profit d'une conscience de la concurrence.

Enfin, il est clair que l'enfouissement des données au sein de l'application mono-utilisateur ne permet en aucun cas de contrôler l'intégrité, la disponibilité, la non répudiation ou encore la propriété d'une donnée, ce qui peut avoir de très graves conséquences pour un travail collaboratif (en plein essor de la notion de traçabilité de l'information).

2. Proposition des auteurs

a. Prendre le meilleur des deux mondes

La pensée des auteurs est de transformer les applications mono-utilisateur en applications supportant l'accès multi-utilisateurs dans le cadre d'un travail de groupe, dit travail collaboratif.

L'idée revient donc à proposer de prendre le meilleur des mondes « être » et « paraître » (originellement nommés « collaboration awareness » et « collaboration transparent »).

Les moyens offerts par l'article sont :

- un contexte d'exécution (« *runtime system* »)
- une API pour automatiser la transformation
- une méthodologie pour effectuer la transformation

Le « runtime system» système est en fait un panneau applicatif sur lequel l'utilisateur a la liste des applications qu'il détient (nom, version …). Cela lui permet également de créer/rejoindre des sessions de travail collaboratif.

L'API est la traditionnelle librairie de composants permettant d'élaborer la transformation de l'application en une application prenant en compte un contexte collaborative.

La méthodologie est probablement l'aspect le plus intéressant de l'article et ce sera elle qui fera l'objet d'une étude dans le paragraphe suivant.

a. Une méthodologie pour transformer les applications.

i) Etape 1: Intégrer l'application dans le « runtime system » de la proposition

Cela a pour but de permettre de lancer l'application via le panneau des applications du « runtime system ». Cela permet également de déclarer les sessions de travail collaboratif.

Le but avoué de ce « runtime system » est double :

- résoudre les problématiques de gestion de versions.
- découverte des acteurs potentiels
- rejoindre, via l'interface, les sessions ouvertes

ii) Etape 2: Organiser la gestion des données partagées distribuées

Le but de cette étape est d'éviter d'enfouir la gestion des données dans l'application mono-utilisateur. Cela permet de gérer l'intégrité et l'accès concurrentiel aux données.

Les différentes étapes de transformation pour gérer ces données distribuées sont:

- Identifier les données susceptibles d'être partagées. Cela est un travail trivial si l'on a pris soin de bien séparer les données et leur traitement, de la présentation des données. Il est probable que cela ne constitue pas un gros problème si l'on utilise un modèle Arch, MVC, PAC …
- Utiliser un substitut offrant un point d'entrée sur la donnée. Le substitut est un fait un composant simplifié de la représentation d'une donnée. En pratique, ce substitut est le parent de toutes les classes représentant une grappe de donnée. Le but de cette étape est de masquer le mécanisme de distribution de données dans le framework.
- Configurer le substitut. Il s'agit d'implémenter la donnée elle-même au travers du substitut pour autoriser ou non l'accès concurrentiel aux méthodes d'un objet.

iii) Etape 3: Ajouter les services pour la conscience de groupe

Dans cette dernière étape de la méthodologie proposée, les auteurs cherchent à ajouter cette conscience de groupe via des « widgets » graphiques. Les auteurs présentent des « widgets » présents dans leurs librairies allant avec leur proposition comme des « distributed mouse pointers ». Les auteurs soulignent qu'il reste possible de créer sont propre « widget » à partir de la librairie de base.

Conclusion

L'apport de cet article est double: il amène une idée de réutilisation des interfaces hommes machines, qui est tout de même un concept original, séduisant … et ambitieux.

De plus, le concept est couplé à des outils et une méthodologie pour opérer le transformation, via une plateforme opérationnelle, qui constitue une couche d'abstraction de la prise en compte de l'aspect collaboratif dans les applications multi-utilisateurs, ce qui est non négligeable pour le développeur. Cela n'a pas été rapporté dans cette synthèse (dû à la nature de « compression » de l'exercice), mais les auteurs ont également présentés des exemples (néanmoins triviaux) de transformation.

Malheureusement, il est clair qu'une telle proposition engendre une surcouche applicative. On peut alors légitimement se poser la question de la dégradation des performances dans un contexte collaboratif, où le facteur temps est non négligeable (TCAO est intrinsèquement synchrone dans le cadre d'un applicatif multi-utilisateurs).

De plus, l'originalité technique reste à démontrer : il s'agit là d'un assemblage de briques logicielles mettant en place une idée de transformation des interfaces.

Du reste, les exemples de transformation proposés par les auteurs restent des exemples triviaux et il serait intéressant de procéder aux mêmes transformations dans le cadre d'applications plus complexes et plus vétustes. On pourrait ainsi contrôler que le travail à fournir reste linéaire en fonction de la taille de l'application à transformer (i.e. ne prenne pas un envol exponentiel).

Pour conclure, je suis convaincu que cette solution reste adapté à des cas plus ou moins simple, mais pas trop simple. Je m'explique : les extrêmes que constituent le « collaboration awareness » et le « collaboration transparent » forment probablement des concepts très adaptés au cas d'applications spécifiques et complexes pour l'un, et très simple sans un besoin d'une trop grande conscience de groupe pour l'autre. Il serait judicieux de considérer cet article comme une proposition s'insérant dans le gouffre séparant ces deux mondes, non pas en prenant le meilleur de ces deux mondes, mais en permettant un développement d'applications collaboratives relativement complexes mais pas forcément trop spécifiques : ce serait un curseur pour naviguer en le « collaboration awareness » et le « collaboration transparent ».

Bibliographie complémentaire

G. Calavary, J. Coutaz, L. Nigay. “from single-user architectureal Design to PAC: a generic Software Architecture Model for CSCW”, in Human Factors in Computing Systems: CIH’97 Conference Proceedings, pages 242-249, ACM, 1997

H. Gajewska, M. Manasse, D. Redell. “Argohalls:Adding Support for Group Awareness to the Argo Telecollaboration System”. In Proceedings of the 8th annual ACM sympossiumon User interface software and technology, pages 157-158, 1995

G. Krasner and Stephen T. Pope. “A cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80”. Journal of Object-Oriented Programming 1(3):26-49, 1988

S. Lukosch and Claus Unger. “Flexible Synchronisation of Shared Groupware Objects”. In Proceedings of the second International Network conference (INC 2000, pages 209-219, University of Plymouth, Great Britain, July 2000

M. Roseman and S. Greenberg. “Building Real-time Groupware with GroupKit, a Groupware Toolkit”. ACM Transactions on Computer-Human Interaction, 3-(1):66-106, March 1996.