Un système auteur de tuteurs intelligents : évolution du mono-usager vers la coopération

Said Talhi, Mahieddine Djoudi, Abdelmadjid Zidani

Résumé

Cet article traite de la coopération dans la production de Systèmes Tuteurs Intelligents (STI). Nous partons d'un système auteur de STI initialement mono-usager (Moalim) et le reconsidérons dans un contexte de collecticiel (ou groupware). Nous proposons une approche d'intégration de la dimension coopérative au sein de ce système afin de permettre à plusieurs auteurs géographiquement dispersés de collaborer pour produire ensemble un STI. Nous envisageons, dans le cadre de ce travail, de leur créer un espace de travail partagé rassemblant tous les outils nécessaires à V élaboration coopérative d'un STI. Nous définissons dans ce sens des mécanismes permettant de gérer la notification et rétroaction de groupe et, comme pour tout système distribué, nous définissons un support pour assurer l'intégrité des données dispersées à travers les différents sites impliqués.

Abstract

This paper report about collaboration in the production of Intelligent Tutoring Systems (ITS). We go from a single user authoring system called Moalim and extend it with group tools supporting multi-user functionalities. We used an integration approach centered on taking into account collaboration and thus enabling several remote authors to produce together an ITS. So, we suggest in this framework, to tailor according to their needs, a shared workspace and make available within it appropriate tools. We define in this way mechanisms allowing notification and group awareness and finally, as for all distributed systems, we define a support to guarantee sparse data integrity through the different involved sites.

Un système auteur de tuteurs intelligents

Evolution du mono-usager vers la coopération

Said Talhi* — Mahieddine Djoudi — Abdelmadjid Zidani***

**Département d'informatique, Université de Batna, 05000 Batna, Algérie*
{s_talhi, azidani}@univ-batna.dz

***Laboratoire IRCOM-SIC, UFR SCIENCES - Bat. SP2MI*
Téléport 2, Boulevard Marie et Pierre Curie, BP 30179
86962 Futuroscope Chasseneuil cedex
djoudi@sic.sp2mi.univ-poitiers.fr

RÉSUMÉ. *Cet article traite de la coopération dans la production de Systèmes Tuteurs Intelligents (STI). Nous partons d'un système auteur de STI initialement mono-usager (Moalim) et le reconsidérons dans un contexte de collecticiel (ou groupware). Nous proposons une approche d'intégration de la dimension coopérative au sein de ce système afin de permettre à plusieurs auteurs géographiquement dispersés de collaborer pour produire ensemble un STI. Nous envisageons, dans le cadre de ce travail, de leur créer un espace de travail partagé rassemblant tous les outils nécessaires à l'élaboration coopérative d'un STI. Nous définissons dans ce sens des mécanismes permettant de gérer la notification et rétroaction de groupe et, comme pour tout système distribué, nous définissons un support pour assurer l'intégrité des données dispersées à travers les différents sites impliqués.*

ABSTRACT. *This paper report about collaboration in the production of Intelligent Tutoring Systems (ITS). We go from a single user authoring system called Moalim and extend it with group tools supporting multi-user functionalities. We used an integration approach centered on taking into account collaboration and thus enabling several remote authors to produce together an ITS. So, we suggest in this framework, to tailor according to their needs, a shared workspace and make available within it appropriate tools. We define in this way mechanisms allowing notification and group awareness and finally, as for all distributed systems, we define a support to guarantee sparse data integrity through the different involved sites.*

MOTS-CLÉS : *Systèmes tuteurs intelligents, systèmes auteurs coopératifs, collecticiel, architecture client/serveur.*

KEY WORDS: *Intelligent tutoring systems, cooperative authoring systems, groupware, client/server architecture.*

1. Introduction

Grâce aux systèmes auteurs, les Systèmes Tuteurs Intelligents (STI) [WEN 87] [NIC 88] ont réalisé une avancée considérable ces dernières années. Cependant, leur réalisation demeure une tâche difficile à entreprendre. Cela nécessite souvent la constitution d'équipes pluridisciplinaires dans lesquelles pédagogues, psychologues, experts de domaines et informaticiens coopèrent pour réaliser de tels tuteurs. Si ces auteurs sont regroupés dans un même lieu géographique et ont la volonté de coopérer, ceci ne pose pas de problèmes puisqu'ils peuvent se rencontrer et discuter le temps qu'il faut jusqu'à l'opérationalisation de leur STI. Par contre, s'ils sont géographiquement éloignés, la collaboration deviendrait une tâche difficile pour ne pas dire impossible. Par conséquent, il est nécessaire de mettre à leur disposition des supports coopératifs leurs permettant de communiquer et coordonner leurs activités.

Aujourd'hui, grâce aux réseaux télématiques et aux collecticiels, des rencontres virtuelles à large échelle sont rendues possibles [COC 93] [REY 98]. Plusieurs travaux dans ce sens ont déjà porté sur des domaines tels que l'*édition coopérative de documents* [DEC 95] [ZID 00], les *médiaspaces* [GAV 92], *la conception coopérative*, etc. Le point commun entre tous ces systèmes est qu'ils permettent à plusieurs participants de travailler ensemble de manière synchrone ou asynchrone pour réaliser une tâche commune.

D'autant plus que la technologie le permet donc et est expérimentée avec succès dans beaucoup de domaines, ceci nous conduit à penser qu'il serait souhaitable que les systèmes auteurs de STI du futur intègrent l'aspect coopératif dans leur mode auteur. Pour ce faire, deux approches nous semblent envisageables : l'une, idéale mais coûteuse, consiste à concevoir des systèmes auteurs qui prennent en compte dès leur conception les outils de coopération. L'autre, plus économique, consiste à intégrer des outils de coopération dans des systèmes auteurs mono-usagers déjà opérationnels.

C'est dans l'optique de la deuxième approche que nous présentons cette réflexion à travers un système que nous avons appelé MoalimCo. Notre objectif est de reprendre un système auteur initialement mono-usager (Moalim), dont nous avons développé un prototype en Prolog/C dans nos travaux antérieurs [TAL 97] et de reconsidérer son *mode auteur* dans un contexte coopératif.

Cet article présente une démarche d'intégration des mécanismes de coopération au sein du système Moalim. Dans la section 2, nous commençons par introduire le principe des systèmes auteurs en général et discutons des différentes approches de développement de systèmes auteurs coopératifs. La section 3 présente la structure de Moalim en insistant beaucoup plus sur le mode auteur, partie où la dimension coopérative sera intégrée. Nous décrivons ensuite dans la section 4, le système coopératif MoalimCo en présentant l'architecture logicielle de son *mode auteur* ainsi que les différents niveaux qu'elle recouvre.

2. Les systèmes auteurs de STI

2.1. *Systèmes auteurs mono-usagers : principes de fonctionnement*

Plusieurs travaux de recherche ont porté sur la réalisation de systèmes auteurs de STI durant cette dernière décennie. Dans sa synthèse la plus récente Tom Murray [Mur 99] en a cité plus de deux douzaines et il les a classé en sept catégories relativement au type de STI qu'ils produisent :

1) enchaînement et planification de curriculum ;
2) stratégies tutorielles ;
3) simulation et entraînement ;
4) système expert en domaine ;
5) multiples types de connaissances ;
6) système à but spécial ;
7) système hypermédia intelligent / adaptatif.

Moalim se situe dans la première catégorie de cette classification : *enchaînement et planification de curriculum*. Les systèmes auteurs de cette catégorie structurent en général le *matériau à enseigner* sous forme d'une hiérarchie d'Unités d'Apprentissage (UA) et où chaque UA possède certains objectifs pédagogiques. Les UA sont liés entre elles par des liens de type : *prérequis, partie de, défini par, expliqué par*, etc.

Quoique ces systèmes n'utilisent pas de représentation explicite des connaissances du domaine, ils investissent cependant l'intelligence au niveau de l'enchaînement des UA et au niveau de la modélisation de l'apprenant. Les UA à présenter à l'apprenant sont alors déterminées dynamiquement en se basant sur les performances de ce dernier, les objectifs pédagogiques de la leçon et les relations qui existent entre les différentes UA.

Selon son architecture classique, un STI comporte quatre composants [WEN 87] [NIC 88] (*module expert du domaine, module pédagogique, modèle de l'apprenant et interface de l'apprenant)* et donc les systèmes auteurs doivent théoriquement fournir tous les outils nécessaires permettant de les construire. Cependant, il faut le reconnaître, peu de systèmes permettent aux auteurs de tout construire comme c'est le cas par exemple du système EON [MUR 98], un système classé par son auteur comme faisant partie de la deuxième catégorie (i.e. *stratégies tutorielles*). Les autres systèmes se limitent en général à des outils permettant de construire un, deux ou à la limite trois composants parmi les quatre. Le reste des composants est généralement prédéfini dans un canevas de STI et l'auteur n'est sollicité que pour introduire des paramètres nécessaires à leur fonctionnement. Le système Moalim est l'un des systèmes qui, dans le souci de faciliter la tâche aux auteurs, ne demande que d'instancier les UA. L'auteur devra ensuite introduire certains paramètres nécessaires au fonctionnement des trois autres composants.

2.2. *Systèmes auteurs coopératifs : approches de conception*

Si potentiellement il existe de nombreuses approches pour bâtir un système auteur coopératif, nous pouvons les classer en deux grandes catégories. Une première voie possible consiste à définir une architecture logicielle qui prenne en compte, dès sa conception, les possibilités et besoins des outils de coopération. Cette voie idéale, permet certainement d'édifier une structure souple qui puisse appliquer rigoureusement les concepts de la métaphore de coopération. Cependant, cette voie est coûteuse. La réalisation de systèmes auteurs mono-usagers pour l'EIAO est déjà suffisamment complexe pour ne pas lui rajouter d'autres préoccupations inhérentes à la coopération.

La deuxième approche, plus pragmatique, et à notre avis plus économique en effort de conception, consiste à enrichir un système auteur préexistant d'un certain nombre de modules visant à lui donner la dimension coopérative. Ce pragmatisme s'appuie sur trois idées raisonnables. La première est l'état de faits : la disponibilité d'un nombre appréciable de systèmes opérationnels. La deuxième part du fait que, compte tenu qu'un système auteur est formé d'un *mode auteur* et d'un *mode apprenant*, seul le *mode auteur* va subir des transformations. L'objectif étant évidemment de faire coopérer les auteurs et non pas les apprenants. La troisième et dernière repose sur le fait que de nombreux travaux concernant le travail coopératif dans d'autres domaines telle que l'édition coopérative de documents [DEC 95] [ZID 00], ont abouti à des résultats satisfaisants. Alors, pourquoi ne pas tirer profit de leurs expériences comme l'a suggéré Grudin [GRU 94].

C'est cette deuxième approche que nous avons adopté et dont nous allons décrire l'ossature à travers un modèle de système auteur coopératif que nous avons nommé MoalimCo et qui est construit par enrichissement du système mono-usager Moalim. Ainsi donc, les STI générés par MoalimCo sont de la même norme que ceux générés par Moalim. Par conséquent, nous consacrons l'essentiel de notre travail à l'étude du *mode auteur* coopératif.

Cependant, nous devons rappeler que la partie logicielle ne constitue pas tout dans les collecticiels. La prise en compte des facteurs humains impliqués par les activités de groupe comme l'a constaté Greenberg [GRE 92], constituent également une condition capitale quant au succès des collecticiels. Ainsi, pour éviter les conflits inhérents à la nature humaine, nous proposons une organisation qui permet de mener la conduite du projet de construction collective du STI de manière rationnelle et optimale. Cette organisation facilite également la manipulation des différents constituants du STI pendant toutes les phases du projet et évite ainsi un certain nombre de problèmes tels que les conflits d'accès, la perte de la trace des versions intermédiaires, la non identification des rôles, etc. Nous définissons alors trois rôles à travers lesquels les participants peuvent intervenir au cours du processus de construction du STI : *auteur principal*, *coauteur constructeur* et *coauteur lecteur/commentateur*.

L'*auteur principal* est le chef de projet, il aura pour rôle de coordonner le travail et de vérifier que le calendrier est bien respecté. Il définit la structure logique du STI à produire en le décomposant en plusieurs composants (partie, chapitre, paragraphe,

figure, image, etc.) puis il distribue les rôles aux différents coauteurs. Il a accès libre à tous les composants du STI, il peut ainsi créer, modifier ou détruire toute partie du STI. Aussi il doit spécifier au système tous les paramètres d'ordre organisationnel qui sont nécessaires à la conduite du projet.

Une fois le projet lancé, le système se chargera d'associer les paramètres spécifiés à la structure du STI afin d'assurer la gestion des accès des auteurs et de leurs interactions mutuelles. Par conséquent, un auteur responsable d'une opération sur un objet du STI (création, modification, ...), sera identifié par exemple par son nom ou par la couleur qui lui est associée et l'opération effectuée sera caractérisée par une date, un lieu, etc.

Un *coauteur constructeur* n'aura la permission que de créer, modifier ou détruire la ou les parties auxquelles il sera affecté, sur le reste des parties du STI il n'aura que le rôle de lecteur/commentateur.

Enfin un *coauteur lecteur/commentateur* n'est autorisé qu'à lire et/ou commenter les parties auxquelles il sera affecté .

3. Présentation générale du système auteur Moalim

Moalim est un système auteur dédié aux enseignants universitaires pour la création de STI basés sur la pédagogie par objectifs. Il utilise trois niveaux d'objectifs [HAM 90] : *les objectifs généraux, les objectifs spécifiques et les objectifs opérationnels*. Cette hiérarchie a permis de considérer trois niveaux dans la structuration de la matière à enseigner : les *Parties* (satisfaisant aux objectifs généraux), les *Chapitres* (satisfaisant aux objectifs spécifiques) et les *Paragraphes* (satisfaisant aux objectifs opérationnels). Ces derniers sont évidemment évaluables, grâce aux questions théoriques et/ou des exercices qui leur sont attachés. Moalim comporte deux sous-systèmes : l'*éditeur de connaissances* (lié au mode auteur), et le *tuteur pédagogique* (lié au mode apprenant).

3.1. *Mode apprenant*

Moalim ne traite pas de la représentation explicite des connaissances du domaine. Il s'insère dans le courant des systèmes qui organisent le processus d'enseignement autour de composants pédagogiques [TEN 94]. La gestion des composants dans le canevas de STI, est assurée par un système multi-expert (le tuteur pédagogique) basé sur un ensemble de règles de production. Ces règles complètement paramétrables, dites règles mères, décrivent les différents plans de tutorat relatifs aux différentes situations dans lesquelles peut se trouver l'apprenant. Elles constituent donc une base de connaissances générique prédéfinie qu'il convient d'instancier pour chaque STI créé par Moalim.

L'opération d'instanciation, produisant des règles filles, est effectuée automatiquement par le système en se basant sur des paramètres saisis impérativement par l'auteur. Ces paramètres du STI, représentés sous forme de

prédicats, décrivent l'aspect quantitatif de la matière à enseigner (nombre de parties, nombre de chapitres, nombre de paragraphes, nombre de questions, nombre d'exercices, etc.).

Pour rester indépendantes de tout domaine, les règles mères invoquent des structures abstraites appelées Unités Pédagogiques (UP). Ces UP étant dénuées de toute connaissance du domaine dans le canevas de STI prédéfini. Elles sont censées recevoir, par instanciation, toutes sortes de connaissances du domaine saisies par l'auteur lors de la création d'un STI. Finalement, deux niveaux de connaissances sont donc utilisés dans la définition du curriculum :

– un niveau supérieur correspondant aux plans de tutorat — Ces derniers consistent en cinq paquets de règles qui invoquent des UP du niveau inférieur. Chaque paquet de règles possède une fonction précise, ces fonctions sont respectivement les suivantes :
  - négociation avec l'apprenant du point d'entrée dans le cours,
  - détermination des UP jugées acquis à l'issue d'une négociation,
  - planification des enchaînements d'UP,
  - recherche et affichage des UP,
  - évaluation de l'apprenant ;

– un niveau inférieur correspondant à l'univers des UP — Cet univers consiste en un réseau des prérequis formé d'une hiérarchie de six sous-niveaux d'UP. Les quatre premiers sous-niveaux correspondent à des UP de cours (sommaire, objectifs parties, objectifs chapitres, paragraphe) et les deux derniers sous-niveaux correspondent à des UP d'évaluation (questions théoriques et/ou exercices de résolution de problèmes).

L'architecture logicielle du mode apprenant est similaire à celle d'un STI traditionnel [WEN 87] [NIC 88], elle comporte: *un module d'exploration libre*, un *module expert du domaine*, un *module pédagogue*, un *module de diagnostic de l'apprenant* et une *interface-apprenant*.

Comme nous l'avons mentionné auparavant, cette architecture reste inchangée dans le système MoalimCo. Tous les auteurs peuvent disposer d'une instance du *tuteur pédagogique* localement afin qu'ils puissent l'exécuter partiellement et ainsi pouvoir effectuer aisément la mise au point du STI.

3.2. *Mode auteur*

En mode auteur, Moalim est constitué des composants suivants (figure 1) :

– un éditeur d'unités pédagogiques (UP),
– un module d'acquisition des paramètres,
– un générateur de règles filles qui permet d'instancier les règles mères,
– un outil de mise au point qui permet d'aider l'auteur à mettre au point le STI,
– une interface-auteur permettant de communiquer avec le système.

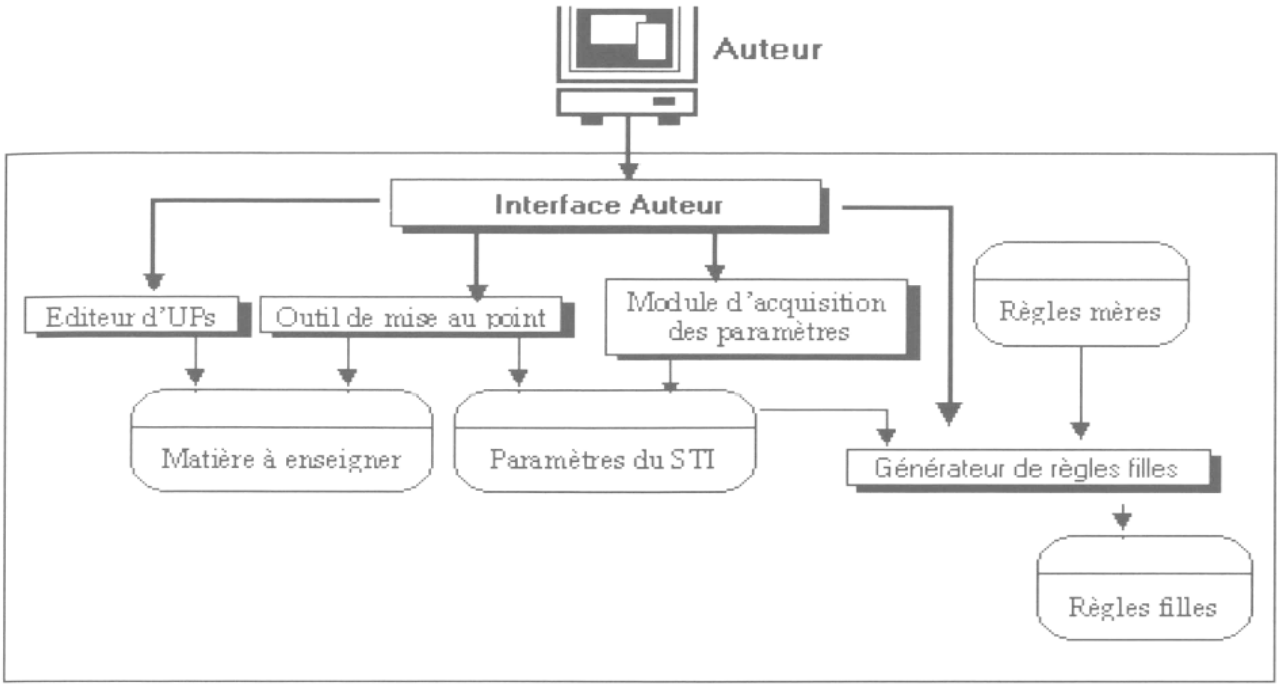

Figure 1. *Architecture du mode auteur*

Pour conclure cette présentation rapide de Moalim nous pouvons dire que du point de vue de l'auteur, construire un STI avec Moalim consiste donc à introduire, via l'éditeur de connaissances, un ensemble d'objets qui seront manipulés par le canevas de STI (le tuteur pédagogique). Ces objets sont constitués (cf. figure 1) de la *matière à enseigner* sous formes d'UP, des *paramètres du STI* sous forme de prédicats, et de la *base pédagogique* sous forme de règles de production. Dans la section suivante, nous montrons les modalités d'intégration des mécanismes de coopération au mode auteur de Moalim.

4. MoalimCo : présentation du mode auteur coopératif

Étant donné que notre objectif est de minimiser les changements au niveau du code initial, nous avons alors adopté une démarche qui consiste à s'interposer entre le *noyau* de l'éditeur de connaissances et son *interface auteur* de telle sorte qu'une action réalisée par un auteur soit diffusée vers tous les autres [ATA 94]. L'introduction de *contrôleurs de dialogue* [COU 94] entre les différentes couches comme nous le verrons (figure 2) permettra certainement d'atteindre cet objectif. Les deux modules qui subiront le plus de changements sont l'*éditeur d'UP* et l'*interface auteur*. Le premier sera découpé en deux composants, une partie qui gère le stockage/récupération et les mises à jour de la copie centrale du STI résidera sur le serveur, l'autre qui regroupe les fonctions d'édition (couper, copier, coller, etc.), de stockage et récupération local résidera sur les sites auteurs. L'interface quant à lui devra fournir, en plus des fonctions actuelles, des mécanismes permettant à l'auteur

de contrôler et d'être avisé de l'évolution des contributions apportées au STI par les autres coauteurs. Les trois autres composants résideront tous sur les sites auteurs.

4.1. *Architecture logicielle*

L'*éditeur coopératif de connaissances* repose sur une architecture client/serveur centralisée [ORF 97]. Par conséquent, toutes les communications transitent automatiquement par le site central (ou serveur). Sur la figure 2, nous pouvons constater qu'à chaque site client on associe deux processus : un *processus de présentation* (PRP) chargé de gérer l'interaction avec l'auteur et un *processus de traitement local* (PRL) qui accomplit toutes les tâches traitables localement (les tâches d'édition par exemple). Quant au niveau central, nous définissons un *processus de traitements central* (PRC) qui gère toutes les communications entre les différents PRL et tient à jour le contenu de la copie centrale et la structure logique du STI. Le PRC détient à son niveau les versions les plus récentes des objets du STI ainsi que sa structure logique, tandis que les stations des différents auteurs peuvent contenir des versions plus anciennes. Par conséquent, il appartiendra à ces auteurs de les récupérer du site central.

L'intérêt de cette approche en terme d'architecture est d'éviter d'alourdir le fonctionnement du serveur central en lui incombant les tâches liées à l'édition de texte et de graphiques qui peuvent être traitées localement. Ceci a pour conséquence d'éviter aux auteurs des attentes inutiles. Ainsi, un PRL peut initier une communication avec le PRC, tout en continuant à exécuter des tâches locales et l'auteur associé mènera par conséquent une attente active.

Au lancement de l'application par un auteur, le PRC prépare aux PRLs toutes les conditions pour leur permettre d'évoluer de façon totalement autonome, il n'interviendra que dans certains cas tels que : l'accès aux données partagées, l'initiation d'une communication, etc.

L'architecture logicielle (figure 2) offre plusieurs types de traitements que nous pouvons décomposer en trois couches : la *couche serveur*, la *couche éditeur* et la *couche présentation*. Chaque couche est structurée comme une collection de modules regroupant chacun plusieurs objets capables de réaliser le type de traitement définis pour ce module. Reposant sur le principe de modularité, ceci suggère que chacune des trois couches ne doit avoir aucune connaissance des deux autres. L'absence de connaissances entre ces couches ne signifie pas absence de communication mais implique des références par indirection [COU 94]. La double nécessité d'assurer à la fois les échanges d'informations au *premier* et au *deuxième* niveau, se traduit par la présence de « *contrôleurs de dialogue* ». Nous interposons donc au *niveau-1* de la figure 2, entre chaque PRP et chaque PRL un *contrôleur de dialogue* (CD), et au *niveau-2* entre le PRC et les différents PRLs le *contrôleur principal de dialogue* (CPD).

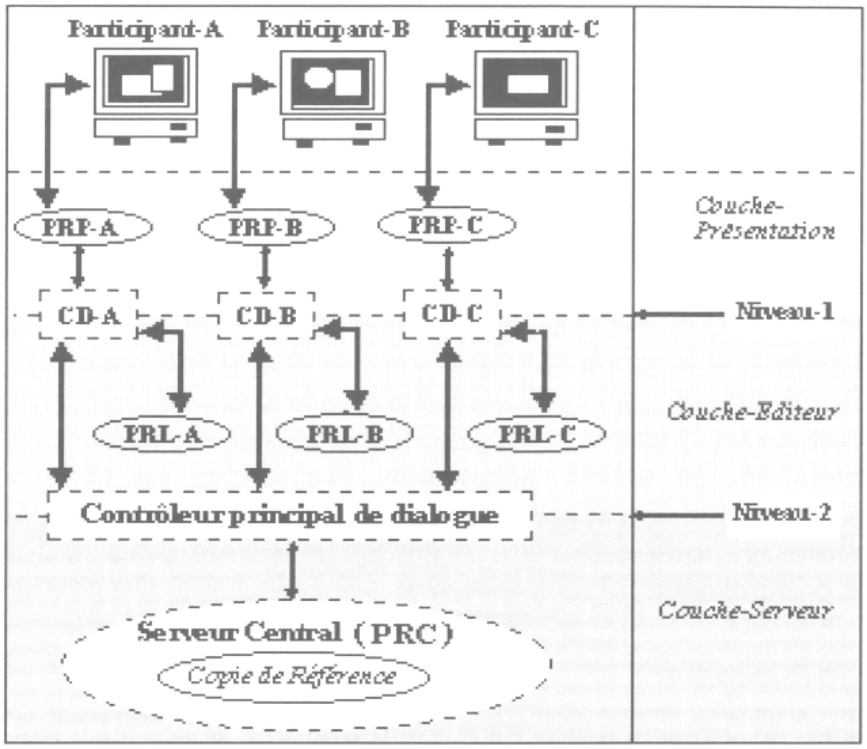

Figure 2. *Architecture logicielle en couches*

Pour décrire les différents composants, nous allons d'abord considérer chacune des couches séparément et présenter ses traitements essentiels. Puis, nous décrivons les contrôleurs de dialogue (CD et CPD) afin de comprendre les interactions entre les différentes couches. L'arrivée d'un événement, comme nous l'avons déjà précisé, implique automatiquement la transition du message associé par ces contrôleurs qui sont les seuls à pouvoir décider des traitements exacts à déclencher parmi ceux qui sont définis au sein d'une couche.

4.2. *Description des couches*

4.2.1. *Couche Serveur*

Le processus central PRC regroupe plusieurs types de traitements, parmi lesquels nous distinguons ceux qui sont liés à la gestion de la structure logique du STI, ainsi que les contenus des composants du STI. Il permet ainsi aux auteurs de stocker et de récupérer les objets du STI dont la structure logique est déclarée aussi bien au niveau central qu'au niveau local. Le PRC est aussi responsable des opérations de contrôle des droits d'accès, du traitement des événements [WII 91] et de la notification de leurs conséquences aux auteurs. Dans le cas de notification des événements, par exemple, le module qui en est chargé gère un ensemble de files d'attentes telles que la *file des engagements*, la *file de blocage*, etc. Chaque fois qu'il est invoqué suite à un événement, ce processus identifie les auteurs destinataires et procède à la structuration des notifications sous forme de messages

transmissibles. Ces messages seront alors mis à la disposition d'un autre module émetteur qui se chargera de l'émission.

4.2.2. *Couche Editeur*

Le processus local PRL regroupe à son tour plusieurs types de traitements permettant à chaque auteur de manipuler les objets composant le STI. Ces traitements incluent aussi bien le support des actions individuelles, que l'aspect de partage et de gestion de la transparence. Par exemple, l'accès à un fichier dans un système mono-usager délivre directement son contenu, Dans notre cas par contre, ce processus déclenchera une suite de traitements tels que la vérification des droits d'accès de l'auteur, de l'état de blocage du composant et enfin l'avertissement des auteurs travaillant sur ce même composant. Au niveau de chaque site, le PRL associé permet à l'auteur d'enregistrer localement les contenus des composants qui lui sont accessibles. Il sollicitera régulièrement le serveur pour maintenir à jour les versions de ces composants.

4.2.3. *Couche Présentation*

Le processus de présentation PRP regroupe un ensemble organisé d'objets interactifs définissant la partie perceptible du système (boutons, curseur, barre de défilement, thermomètres de progression de tâches, etc.). Ainsi à tout objet modélisant une partie du domaine de notre application, nous associons une technique de présentation matérialisée par un objet interactif qui réagit aux actions de l'auteur. Le PRP supporte un ensemble de traitements parmi lesquelles, nous distinguons essentiellement les traitements d'analyse et les traitements d'affichage qui sont chargés de traquer les gestes de l'auteur et réagir en conséquence, ainsi que d'afficher la représentation approprié à la situation courante. L'objectif de la *couche présentation* est de rendre le système facile à manipuler en présentant à l'auteur une vue explicite qui ne comporte pas d'ambiguïtés [MAR 93].

4.3. *Les contrôleurs de dialogue*

Ces contrôleurs situés respectivement aux *niveau-1* et *niveau-2* (figure 2) se composent chacun de trois modules indépendants réalisant respectivement les tâches de *contrôle*, d'*émission* et de *réception* de messages. Le module de *contrôle* regroupe toutes les fonctions qui permettent de coordonner et synchroniser l'exécution des différents modules au sein des trois couches, conformément aux actions des différents auteurs. Il dispose à tout instant de toutes les informations nécessaires pour déterminer exactement quels sont les objets à invoquer au sein des couches dont il est responsable. A chaque fois qu'un événement se manifeste, le message matérialisant cet événement est délivré au module de *contrôle* par le *récepteur* associé. Le module de contrôle réagit alors en suivant trois étapes : analyser l'événement, dresser un plan d'actions puis exécuter le plan établi.

5. Conclusion

Le travail présenté dans ce papier porte sur l'intégration de la dimension coopérative dans un système auteur de STI initialement mono-usager. Cela nous a mené à reconcevoir le mode auteur afin qu'il puisse permettre à des auteurs géographiquement distants de coopérer pour produire un STI de la même norme que le système initial.

Le nouveau système obtenu, appelé MoalimCo, permet à plusieurs auteurs de se connecter à une session de travail caractérisée par un espace de coopération et une politique de contrôle. L'espace de coopération est représenté par un ensemble de composants structurés (UP, paramètres, règles) et des outils d'édition.

La politique de contrôle gère la participation des utilisateurs à la session de travail et la négociation du droit d'intervention sur un composant du STI. Les informations échangées entre les sites sont soit des opérations de contrôle soit des flots de données correspondant aux composants du STI. Ce système est caractérisé par une architecture logicielle centralisée basée sur le concept « client-serveur ».

Pour mieux appréhender le fonctionnement de notre système, nous comptons étendre le prototype actuel dans le cadre de nos futurs travaux. Nous pourrons mettre à l'épreuve l'efficacité du serveur à répondre aux différentes requêtes des auteurs (récupération et stockage des informations, gestion de la structure logique et du contenu du STI, ...) et à un niveau plus haut les performances des contrôleurs de dialogue.

L'implémentation des traitements associés aux modules que nous avons défini est une tâche coûteuse, en particulier pour accroître la tolérance aux pannes transitoires du système et assurer la sécurité. La solution que nous avons proposé consiste à remettre le contrôle aux contrôleurs de dialogue (CD) en cas de panne du serveur. Ils se chargeront alors de maintenir l'application dans un état cohérent au niveau local, jusqu'à la reprise en service du serveur. Cette solution bien qu'utile demeure insuffisante. Par conséquent, nous projetons dans le futur d'étendre cette architecture vers une architecture répliquée.

6. Bibliographie

[COC 93] COCKBURN. A., GREENBERG. S., « Making contact Getting the group communicating with groupware », *Proceedings of the ACM conference on organizational computing systems*, november 1993, p. 1-4.

[DEC 95] DECOUCHANT D., QUINT V., ROMERO M., « Structured Cooperative Editing and Group Awareness », *Proceedings of the Sixth International Conference on Human-Computer Interaction*, Tokyo, July 9-14 1995, vol. 20A, p. 403-408.

[GAV 92] GAVER W., MORAN T., MACLEAN A., LÖVSTRAND L., DOURISH P., CARTER K., BUXTON W., « Realizing a Video Environment : Europarc's RAVE System »,

Proceedings of the Conference on Human Factors in Computing Systems CHI'92, Montery, 1992, p. 27.

[GRU 94] GRUDIN. J., « Groupware and social dynamics : Eight challenges for developers », *Communication of the ACM*, vol. 37, n° 1, January 1994, p. 92-105.

[HAM 90] HAMELINE. D., « Les objectifs pédagogiques en formation initiale et en formation continue », *Edition ESF, 8ième édition*, Paris, 1990.

[MAR 93] MARCUS. A., « Human Communications Issues », Advanced UIs », *Communications of the ACM*, vol. 36, n° 4, April 1993, p. 101-109.

[MUR 98] MURRAY T., « Authoring knowledge-based Tutors : Tools for content, instructional strategy, student model and interface design », *Journal of the Learning Sciences*, vol. 7,.n° 1, 1998.

[MUR 99] MURRAY. T., « Authoring Intelligent Tutoring Systems : An analysis of the state of the art » , *International Journal of AI in Education*, vol. 10, p. 98-129, 1999.

[NIC 88] NICAUD J.F., VIVET M., « Les tuteurs intelligents, réalisations et tendances de recherche », *Technique et Science Informatiques*, vol. 7, n°1, 1988, p. 21-45, Hermes, Paris.

[ORF 97] ORFALI. R., HARKEY. D., EDWARDS. J., « Essential Client/Server Survival Guide », *John Willeys &Sons Inc Eds*, 2nd edition , New York, 1997.

[PAL 94] PALMER. J., FIELDS. N., « Guest editors introduction : computer supported cooperative work », *Communications IEEE Computer*, vol. 27, n° 5, 1994, p. 15-18.

[REY 98] REYNARD, G., BENFORD, S., GREENHALGH, C., HEATH, C., « Awareness driven video quality of service in collaborative virtual environments », *Proceedings of the ACM conference CHI'98*, 1998.

[TAL 97] TALHI S., « MOALIM : vers un environnement indépendant du domaine pour le développement de tutoriels basés sur la pédagogie par objectifs », *Thèse de Magister*, Université de Constantine, Algérie, Mai 1997.

[TEN 94] TENACHI. A. , « Modélisation de connaissances pédagogiques - systèmes d'aide à l'élaboration de cursus personnalisés » . Thèse de Doctorat de l'Université Henri Poincaré, Nancy, 1994.

[WEN 87] WENGER. E., « Artificial Intelligence and Tutoring Systems », Addison-Wesley Eds., USA, 1987.

[WII 91] WIIL U.K., « Using Events as Support for Data Sharing in Collaborative Work », *Proceedings of the International Workshop on CSCW*, Berlin, 1991.

[ZID 00] ZIDANI. A., BOUFAIDA M, DJOUDI M., « JamEdit : un outil interactif et coopératif pour l'édition coopérative de documents », *Revue TSI*, vol. 19, n°1, 2000, p. 1-23, édition Hermes, Paris.