CAMITS : Système Auteur Coopératif de Tuteurs Intelligents

Said Talhi[1] — Mahieddine Djoudi[2] — Abdelmadjid Zidani[1]

[1]*Département d'informatique, Université de Batna, 05000 Batna, Algérie*
s_talhi@yahoo.fr, azidani@yahoo.com

[2]*Laboratoire IRCOM-SIC, UFR SCIENCES - Bat. SP2MI*
Téléport 2, Boulevard Marie et Pierre Curie, BP 30179
86962 FUTUROSCOPE CHASSENEUIL Cedex France
djoudi@sic.sp2mi.univ-poitiers.fr

Résumé :

Dans cet article, nous présentons le système CAMITS pour la conception coopérative de systèmes tuteurs intelligents. Bâti sur une architecture client-serveur, le système permet à plusieurs auteurs géographiquement dispersés de collaborer pour produire ensemble de tels tuteurs. 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 tuteur intelligent. 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.

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

1. Introduction

Malgré les efforts consentis ces dernières années, la réalisation de Systèmes Tuteurs Intelligents (STI) [WEN87] [NIC88] demeure une tâche difficile à entreprendre. Cette tâche exige souvent la constitution d'équipes pluridisciplinaires dans lesquelles pédagogues, psychologues et experts de domaines doivent coopérer avec

des informaticiens pour réaliser de tels systèmes. En vue d'améliorer la productivité dans ce domaine et permettre à une communauté plus large de s'impliquer, des systèmes auteurs sont apparus et ont permis de développer des STI parfois sans taper une seule ligne de code. La tâche est alors réduite aux experts de domaines à faire couler des connaissances dans un canevas de STI générique prédéterminé par le système. Certains de ces systèmes auteurs sont discutés dans [MUR99], cependant, ils ont tous été conçus pour fonctionner en mono-usager.

De nos jours, nous pensons que la disponibilité de ces systèmes ne résout pas totalement le problème. En effet, avec la complexité et l'interdépendance des sciences actuelles, la difficulté de construction d'un STI par un seul auteur reste toujours à l'ordre du jour. Pour produire un STI dans un domaine donné, cela nécessite souvent la coopération de divers experts qui, dans la plupart du temps, se trouvent géographiquement éloignés. 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 collecticiels, des rencontres virtuelles à large échelle sont rendues possibles [COC93] [REY98]. Plusieurs travaux dans ce sens ont déjà porté sur des domaines tels que l'édition coopérative de documents [DEC95] [PAC94], [ZID00], les médiaspaces [GAV92] [MAN91], 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 (spécialement leur mode auteur) intègrent l'aspect coopératif à travers un réseau informatique. C'est dans cette optique que nous présentons cette réflexion à travers un système auteur que nous avons appelé CAMITS (Collaborative Authoring Model for Intelligent Tutoring Systems).

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. Nous présentons en particulier le système CAMITS sous son aspect organisationnel. La section 3 présente la structure du STI engendré par CAMITS. Nous décrivons ensuite dans la section 4, le mode auteur coopératif CAMITS en présentant son architecture logicielle ainsi que les différents niveaux qu'elle recouvre. Nous concluons ce document en donnant les perspectives de ce travail.

2. 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. Outre notre expérience avec le système ***Moalim*** [TAL97], Murray [MUR99] en a cité plus de deux douzaines dans sa synthèse la

plus récente. Parmi ces systèmes nous pouvons citer : ***Cream-tools*** [NKA96], ***Eon*** [MUR98], ***Iris*** [ARR97], ***Training Express*** [CLA88], etc. L'auteur en a fait une classification en sept catégories relativement au type de STI qu'ils produisent, à savoir : *enchaînement et planification de curriculum, stratégies tutorielles, simulation et entraînement, système expert en domaine, types de connaissances multiples, système à but spécial* et, *hypermédia intelligent / adaptatif*.

Etant donné que les STI se trouvent souvent structurés en quatre composants (*expert du domaine, module pédagogue, modèle-apprenant, et interface apprenant)*, les systèmes auteurs doivent donc théoriquement fournir tous les outils nécessaires permettant de les construire. Néanmoins, il faut le reconnaître, très peu de systèmes exigent des auteurs de tout construire comme c'est le cas par exemple du système ***Eon*** [MUR98]. 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.

CAMITS, le système présenté dans ce papier, offre des fonctionnalités de collaboration aux auteurs, génère des STI qui se situent dans la première et septième catégorie de cette classification : *enchaînement et planification de curriculum, hypermédia intelligent/adaptatif*. Les systèmes auteurs de cette catégorie structurent généralement la *matière à enseigner* sous forme d'un réseau 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, la manipulation des liens hypertextes et l'adaptation du cursus par l'utilisation d'un modèle 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. Le système CAMITS 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 le réseau de prérequis et les paramètres nécessaires au fonctionnement des trois autres composants.

2.2. Systèmes auteurs coopératifs : approches de conception

Si en réalité il existe de nombreuses approches pour bâtir un système auteur coopératif, nous pouvons néanmoins les classer en deux grandes catégories. Une première voie, pragmatique, et à notre avis plus économique en effort de réalisation, consiste à enrichir un système auteur préexistant d'un certain nombre de modules visant à lui donner la dimension coopérative. Cependant la rigidité induite par les modules d'acquisition de connaissances des systèmes auteurs mono-usagers rend très difficile la prise de contrôle de la conscience de groupe et la gestion partagée des bases de connaissances. Les systèmes produits manqueront certainement

d'efficacité et utiliseront des mécanismes de coopération seulement à un degré élémentaire.
Une deuxième voie possible, celle que nous avons adopté, consiste à définir une architecture logicielle qui prenne en compte, dès sa conception, les possibilités et besoins des outils de coopération. Cette voie idéale, quoique coûteuse, permet certainement d'édifier une structure souple qui puisse appliquer rigoureusement les concepts de la métaphore de coopération.
Nous devons fournir, à travers cette architecture, un espace de travail commun à l'ensemble des auteurs impliqués dans la construction coopérative d'un STI. Cet espace doit permettre, à travers une interface graphique interactive, de leur apporter suffisamment de connaissances sur l'état courant du STI partagé ainsi que sur leurs contributions mutuelles.
Toutefois, il est à remarquer que la partie logicielle ne constitue pas tout dans les collecticiels. La prise en compte des facteurs humains impliqués par les activités de groupe constitue également une condition capitale quant au succès des collecticiels [GRE92]. 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 quatre rôles à travers lesquels les participants peuvent intervenir au cours du processus de construction du STI : *auteur principal*, *coauteur superviseur*, *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, UA, figure, image, etc.) puis il distribue les rôles aux différents co-auteurs. Il a accès libre à tous les composants du STI, il peut ainsi créer, modifier ou détruire toute partie du STI.
- Un *coauteur superviseur* est désigné par l'auteur principal pour superviser l'avancement du travail sur certaines parties du STI. Ses prérogatives sont restreintes aux parties auxquelles il est affecté.
- 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. Mode apprenant

Pour structurer la matière à enseigner, CAMITS utilise une hiérarchie à trois niveaux d'objectifs pédagogiques définie dans [HAM 90] et [TAG 91] : *les objectifs généraux, les objectifs spécifiques et les objectifs opérationnels*. Cette hiérarchie a

permis de considérer trois niveaux d'abstraction du contenu : les *parties* (satisfaisant aux objectifs généraux), les *chapitres* (satisfaisant aux objectifs spécifiques) et les *unités d'apprentissage hypermédias* (UAH) (satisfaisant aux objectifs opérationnels). Ces derniers sont les unités de transfert évaluables. Deux modes d'apprentissage sont offerts aux apprenants : le *mode formation* (i.e. apprentissage avec évaluation) et le *mode exploration libre*.
Dans la section suivante, nous décrivons le modèle de STI sous-jacent au système CAMITS puis nous décrivons dans la suite son architecture logicielle.

3.1. Modèle du STI créé par CAMITS

CAMITS s'insère dans le courant des systèmes qui organisent le processus d'enseignement autour de composants hypermédias. La gestion des composants dans le canevas de STI, est assurée par un système multi-expert 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 qu'il convient d'instancier pour chaque STI créé par CAMITS.
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 d'unités d'apprentissage, nombre de questions, nombre d'exercices, etc.).
Pour rester indépendantes de tout domaine, les règles mères invoquent des structures abstraites appelées UAH. Ces UAH 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, sous toutes les formes de médias permises par le langage HTML (texte, image fixe, image animé, son, vidéo, applet).
En somme, deux niveaux de connaissances sont donc utilisés dans la définition du curriculum :

1. *Un niveau supérieur correspondant aux plans de tutorat* — Ces derniers consistent en cinq paquets de règles qui invoquent des UAH 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 UAH jugées acquis à l'issue d'une phase de négociation; planification des enchaînements d'UAH; recherche et affichage des UAH; évaluation de l'apprenant.
2. *Un niveau inférieur correspondant à l'univers des UAH* — Cet univers consiste en un réseau de prérequis formé d'une hiérarchie de six sous-niveaux d'UAH. Les quatre premiers sous-niveaux correspondent à des UAH de cours (sommaire, résumé de partie, résumé de chapitre, UAH de cours) et les deux derniers sous-niveaux correspondent à des UAH d'évaluation (questions et exercices).

3.2. Architecture logicielle du STI

L'architecture logicielle du STI est similaire à celle d'un STI traditionnel [WEN87] [NIC88], elle comporte :

1. Un *module d'exploration libre* qui permet à l'apprenant de naviguer librement à travers les UAH.
2. Trois modules représentant le mode *d'apprentissage formation* :
   - Un module expert du domaine qui permet, en utilisant les règles filles de recherche des UAH, de chercher et d'afficher l'UAH sollicitée par le système à un moment donné.
   - Un module pédagogue qui permet de négocier avec l'apprenant le point d'entrée dans le cours et de planifier l'enchaînement des UAH sur la base du résultat de cette négociation. Ces deux fonctions étant assurées par deux sous-modules : Le négociateur utilisant les règles filles de négociation et le planificateur utilisant les règles filles de planification.
   - Un module de diagnostic de l'apprenant qui permet d'évaluer l'usager et d'assurer la maintenance d'un modèle de l'apprenant de type overlay. Ce module comporte à son tour trois sous-modules : un évaluateur utilisant les règles filles d'évaluation, un déducteur des acquis utilisant les règles filles de détermination des acquis et un gestionnaire du modèle de l'apprenant manipulant le contenu de ce dernier.
3. Un module superviseur qui permet d'une part de communiquer avec l'apprenant et d'autre part, de coordonner la communication entre les trois modules : expert, pédagogue et celui du diagnostic de l'apprenant. La communication entre ces trois modules étant assurée par envoi de message.

La différence avec un STI classique est le fait que notre STI réside sur un serveur et peut donc être sollicité à distance par les apprenants. Son implémentation en PHP/MySQL contribue certainement en faveur de son utilisation comme système d'apprentissage à distance.

4. Mode auteur coopératif

L'*éditeur de connaissances* (ou mode auteur) présente aux auteurs tous les outils nécessaires à l'élaboration collaborative d'un STI. Du point de vue d'un auteur, construire un STI avec CAMITS consiste à introduire, via cet *éditeur*, un ensemble d'objets qui seront manipulés par le *mode apprenant*. Ces objets sont constitués de la *matière à enseigner* sous formes d'UAH, du *réseau des prérequis* sous forme de graphes orientés, des *paramètres du STI* sous forme de prédicats, et de la *base pédagogique* sous forme de règles de production.

La coopération dans CAMITS est introduite au niveau de l'édition de la *matière à enseigner* et au niveau de l'édition du *réseau de prérequis*. Ces deux composants sont bien structurés : la *matière* étant hiérarchisée en *parties*, *chapitres* et *UAH*, et le *réseau de prérequis* en sous-réseaux (*prérequis-parties*, *prérequis-chapitres* et *prérequis-UAH*). Cette structure s'adapte bien pour la fragmentation de ces deux

composants et constitue de ce fait la base de notre approche d'édition coopérative comme dans JamEdit [ZID00].
Le principe d'édition coopérative que nous avons exploité repose en fait sur les deux concepts clés utilisés dans la plupart des éditeurs coopératifs : la fragmentation et l'attribution de rôles d'édition sur les différents fragments. Nous avons défini quatre rôles dans CAMITS (cf. 2.2): *auteur principal, superviseur, constructeur et lecteur/commentateur.*
Avant tout projet de construction d'un STI, une phase de négociation via des outils de communication s'impose. Les rôles sont alors attribués par l'auteur principal et des sous-groupes de travail sont constitués autour de différents fragments conformément aux compétences et disponibilités des participants. Chaque sous groupe est alors dirigé par un superviseur. Tout auteur d'un sous groupe peut détenir les rôles de superviseur ou de constructeur, toutefois, une politique d'exclusion mutuelle n'autorise à tout instant qu'un seul superviseur ou constructeur.

4.1. Modes de coopération

La définition d'un modèle d'interaction coopératif efficace repose sur le choix de modes d'interaction naturels et avec le minimum de contraintes possibles. Le processus de construction coopérative d'un STI est caractérisé par un enchaînement de phases au cours desquelles les auteurs peuvent travailler individuellement ou collectivement. Dans l'editeur coopératif JamEdit [ZID00] nous avons défini quatre modes de coopération que nous avons jugé conserver dans CAMITS : *responsabilité individuelle, échange dynamique, version alternative* et *responsabilité collective*.
Les trois premiers modes sont des modes de coopération typiquement asynchrones. Le troisième s'inspire de la démarche réelle suivante : *réfléchissons séparément à la question puis comparons nos résultats*. Le quatrième est un mode synchrone. L'objectif visé à travers ce mode n'est pas d'offrir le mode synchrone en lui-même, qui reste très délicat à réaliser, mais de permettre à un nombre relativement réduit d'auteurs (tels que les responsables hiérarchiques), lorsque le projet arrive à terme, de mettre au point la version finale du STI.
Ainsi, à travers ces modes, les auteurs accèdent aux différents fragments du STI et la coopération prend place à travers des annotations et des commentaires liés aux fragments développés par les autres

4.2. Architecture logicielle

L*'éditeur coopératif de connaissances* est organisé selon une architecture client/serveur centralisée [ORF97]. Par conséquent, toutes les communications transitent automatiquement par le site central (ou serveur). A chaque site client on associe un processus local (PRL) qui accomplit toutes les tâches traitables localement (les tâches d'édition par exemple). Quant au niveau central, nous définissons un processus 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 qui ne sont pas forcément à jour. Par conséquent, il appartiendra à ces auteurs de les récupérer du site central.
Au lancement de l'application par un auteur, le PRC prépare aux PRL 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 1) 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 par ailleurs absence de communication mais implique des références par indirection [COU94].
La double nécessité d'assurer à la fois les échanges d'informations entre les deux couches du poste client et entre le client et le serveur, se traduit par la présence de *contrôleurs de dialogue*. Nous interposons donc entre chaque couche de présentation et chaque couche d'édition un *contrôleur de dialogue* (CD), et entre la couche serveur et chaque couche d'édition le *contrôleur principal de dialogue* (CPD).
Chacune des couches est structurée comme une collection de modules regroupant chacun plusieurs objets capables de réaliser le type de traitement définis pour ce module. L'arrivée d'un événement, comme nous l'avons déjà précisé, implique automatiquement la transition du message associé par les contrôleurs de dialogue qui sont les seuls à pouvoir décider des traitements exacts à déclencher parmi ceux qui sont définis au sein d'une couche.

4.2.1. Couche Serveur

Cette couche 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. Ils permettent 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. Cette couche 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.

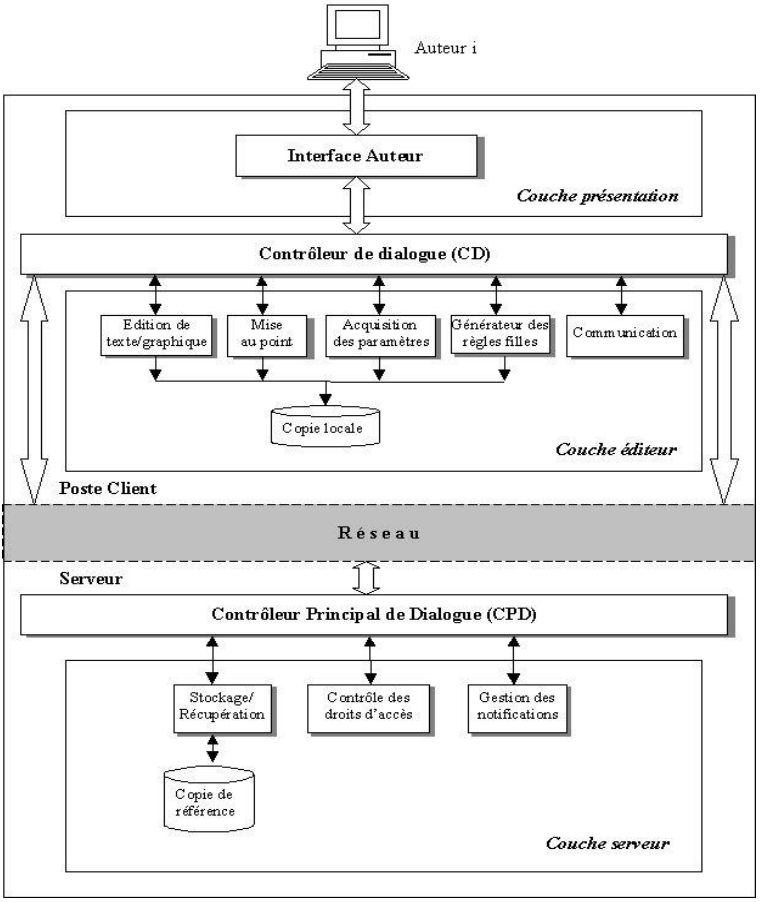

Figure 1 : Architecture logicielle en couches du mode auteur

4.2.2. Couche Editeur

Cette couche 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. Par contre dans notre cas, 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, les traitements associés permettent 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. Les différents composants de la couche éditeur sont :

1. Editeur d'UAH/réseau de prérequis : Ce composant est constitué de deux modules permettant la création/mise à jour de différents objets du STI. Un premier module permet à l'auteur d'éditer le réseau des prérequis sous forme graphique. Le réseau est formé de nœuds et d'arcs liant ces nœuds et indique les différents cheminements possibles entre les constituants de la matière à enseigner. Trois niveaux sont utilisées dans le réseau. Un niveau montre les prérequis entre *parties*, un autre montre les prérequis entre les *chapitres* d'une partie, et un troisième niveau montre les prérequis entre les UAH d'un chapitre. Le deuxième module permet l'édition WISIWIG des UAH en utilisant le langage HTML.
2. Module d'acquisition des paramètres du STI : Via ce module, l'auteur doit spécifier au système la manière avec laquelle il a décomposé la matière d'enseignement (nombre de parties, nombre de chapitres par partie, nombre d'UAH par chapitre, etc.). Ces paramètres sont mémorisés sous forme de prédicats puis utilisés pour instancier les règles mères. Par exemple le prédicat *nbuah(1,2,4)* indique que le chapitre 2 de la partie 1 contient 4 UAH.
3. Générateur des règles filles : Ce module permet à l'auteur de générer les cinq paquets de règles filles qui représentent les différents plans de tutorat. Cette génération est effectuée par instanciation de cinq paquets de règles mères et est réalisée sur la base des paramètres du STI introduits via le module précédent.
4. Outil de mise au point : Comme la plupart des systèmes auteurs, CAMITS offre un outil permettant d'assister l'auteur dans la détection des erreurs et des incohérences qui peuvent se produire durant la construction du STI. Il s'agit essentiellement d'une mise au point technique qui consiste à rendre le tutoriel exécutable sans erreurs, c'est à dire conforme au modèle imposé par CAMITS. Le contrôle de compatibilité matière/paramètres par exemple permet de vérifier si les paramètres indiqués par l'auteur correspondent réellement aux contenus des différentes entités de la matière stoquées.

4.2.3. Couche Présentation

Cette couche regroupe un ensemble organisé d'objets interactifs définissant la partie perceptible du système (boutons, curseur, barre de défilement, thermomètres de progression de tâches, menus déroulants, 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.
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 [MAR93]. A part les menus déroulants traduisant les différentes fonctions, nous trouvons spécialement une boite à outils contenant des icônes référençant les fonctions les plus utilisées, et des palettes de widgets permettant la construction graphique du *réseau de prérequis.*

4.2.4. Les contrôleurs de dialogue CD et CPD

Ces contrôleurs se composent chacun de trois modules indépendants réalisant respectivement les tâches de *contrôle*, d'*émission* et de *réception* de messages (figure 2).

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.

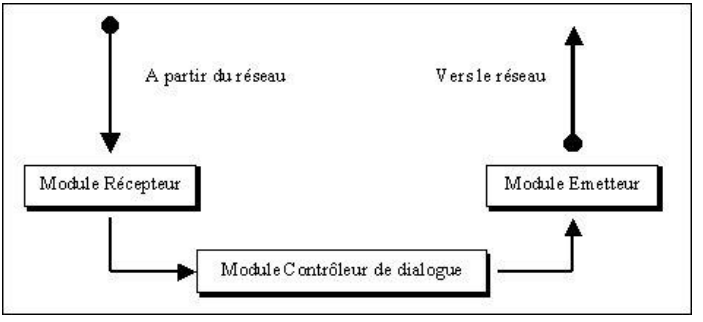

Figure 2 : Structure des contrôleurs de dialogue CD et CPD

Un événement généré par un auteur à partir de l'interface, aboutit à son CD qui l'analyse et détermine s'il est traitable localement ou s'il nécessite l'intervention du serveur. S'il s'agit d'un traitement local , le CD commence à invoquer les modules de la couche éditeur et si l'événement exige une réponse alors le CD élabore le message approprié et le délivre à l'auteur via son interface.

Par contre si l'événement nécessite l'intervention du serveur, le CD déclenche les traitements pour charger toutes les informations constituant les messages à émettre et déclenche le *module émetteur* associé. Le *module récepteur* du serveur reçoit les messages émis et les délivre au *module de contrôle* du CPD qui les analyse à son tour et élabore un plan d'actions. Il procède ensuite à l'exécution de ce plan en invoquant les modules de la couche serveur concernés par les traitements de cet événement. Une fois les traitements achevés, la réponse reprend le chemin inverse sous forme d'un ou plusieurs messages selon la taille des informations renvoyées.

4.3. Notification/Rétroaction de groupe

La fonction de notification et de rétroaction de groupe constitue un point clé dans la conception d'une application coopérative. Celle-ci inclut toutes les fonctions d'interface et de systèmes qui permettent aux utilisateurs de percevoir les activités

des autres utilisateurs, ainsi que de contrôler et d'agir sur l'environnement partagé [DEC94].
Pour développer le support informatique approprié, nous avons dégagé les principaux facteurs concernés par la transparence au sein de l'espace de travail. La table suivante résume les éléments considérés ainsi que les questions que peuvent se poser les auteurs participants.

	Interrogations des auteurs participants
Identification	*Qui participe à l'activité ?*
Lieu	*Où est-il ?*
Présence	*Est-il présent en session ?*
Rôle	*Avec quel rôle intervient-il ?*
Modifications	*Qu'est ce qu'il a modifié ?*
Fragments	*Sur quels fragments travaille t-il ?*

CAMITS exploite un mécanisme de notification qui lui permet de diffuser aux auteurs les différents événements qui se produisent au sein de l'espace de travail partagé. Au cours du processus de collaboration, les participants prennent alors connaissance de "*Qui fait quoi, Où et Quand ?* ". C'est ainsi que CAMITS peut fournir à chaque participant des informations concernant ses collègues (rôle, identité, etc.), les interactions au sein de l'espace de travail (activités courantes, parties manipulées, modifications réalisées, etc.), ainsi que les états de chaque fragment du STI (bloqué, libre, auteurs le manipulant, instant de blocage, etc.).
Le processus de gestion des événements est à la charge du serveur central (figure 1). Comme le serveur encapsule la copie de référence, la notification d'événements liés aux différents fragments du STI se trouve ainsi facilitée. Chaque auteur peut demander au serveur central d'être informé à chaque fois qu'un événement se produit au sein de l'espace partagé. Par exemple, il peut établir un engagement afin de recevoir les modifications apportées à un fragment spécifique du STI, qui lui seront alors automatiquement délivrées. Les engagements sont établis manuellement par les participants comme le montre la figure 3.
Par exemple, en ce qui concerne la *partie « L'API Java » (figure 3)*, l'auteur *Aladin* peut configurer les événements dont il souhaite être notifié, il spécifie au système qu'il souhaite être notifié à chaque fois qu'elle est bloquée ou libérée. Il sollicite également le droit de blocage sur cette partie ainsi que les mises à jour réalisées par les autres auteurs. *Aladin* peut également spécifier la manière selon laquelle il désire être informé par des marques sur les objets, des messages à l'écran ou des messages sonores. Enfin, il peut être concentré et décider, pour ne pas être perturbé, de fermer son espace de travail. De cette manière, nous permettons aux participants de contrôler eux-mêmes le degré d'ouverture de leur espace de travail.
A chaque fois qu'un auteur enregistre ses modifications, il force leur apparition sur les écrans des autres. Par exemple, sur la figure 4, l'auteur *Aladin* est notifié de l'arrivée de modifications associées au chapitre *« Dessiner en Java »* conformément à son engagement de la figure 3.

Figure 3 : Engagements des auteurs participants

Figure 4 : Notification des événements à un auteur

5. Conclusion

Le travail présenté dans ce papier porte sur la conception d'un système auteur coopératif de tuteurs intelligents. Ce système, appelé CAMITS, permet à des auteurs géographiquement distants de collaborer pour produire un STI selon un canevas prédéfini dans le système. Le STI produit, écrit en PHP/MySQL, réside sur un serveur et peut donc être utilisé parallèlement par différents apprenants à distance.
CAMITS 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 (UAH, réseaux de rérequis, paramètres du tutoriel, règles tutorielles) et des outils d'édition et de communication.
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 définis est une tâche coûteuse, en particulier pour accroître la tolérance aux pannes transitoires du système et assurer la sécurité. La solution que nous avons proposée 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.

Bibliographie

[ARR97] Arruarte A., Fernandez-Castro I., Ferrero B., Greer J., « The IRIS shell : How to build ITSs from pedagogical and design requisites », *International Journal of Artificial Intelligence in Education*, vol. 8, 1997, n° 3-4, p. 341-381.

[CLA88] Clancey W., Joerger K., « A Practical Authoring Shell for Apprenticeship Learning », *Proceedings of ITS-88*, Montreal, June 1988, p. 67-74.

[COC93] 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.

[DEC94] Decouchant. D. , « Rétroaction de groupe et édition coopérative de documents structurés ». *Actes d'IHM'94*.

[DEC95] 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.

[GAV92] 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.

[GRE92] Greenberg. S., Roseman. K., Webster. D. , « Human and Technical Factors of Distributed Group Drawing Tools, Interacting with Computers », vol 4, n° 3, December 1992, p. 364-392.

[HAM90] Hameline. D. , « Les objectifs pédagogiques en formation initiale et en formation continue », Edition ESF, 8ième édition, Paris, 1990.

[MAN91] Mantei M., Backer R. M., Sellem A., Buxton W., Milligan T., Wellman, *Experience of the CHI'91 Conference on Human Factors in Computing Systems*, Nouvelles Orleans, 1991, p. 203.

[MAR93] Marcus. A. , « Human Communications Issues », Advanced UIs », *Communications of the ACM*, vol. 36, n° 4, April 1993, p. 101-109.

[MUR98] 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.

[MUR99] 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.

[NIC88] 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.

[NKA96] Nkambou. R., Gauthier. R., Frasson M.C. , « CREAM-Tools: an authoring environment for curriculum and course building in an ITS », *Proceedings of the Third International Conference on Computer Aided Learning and Instructional Science and Engineering*, New York, Springer-Verlag, 1996.

[ORF97] Orfali. R., Harkey. D., Edwards. J. , « Essential Client/Server Survival Guide », John Willeys &Sons Inc Eds, 2nd edition , New York, 1997.

[PAC94] Pacull F., Sandoz A., Schiper A. , « Duplex : A Distributed Collaborative Editing Environment in Large Scale », *Proceedings of the ACM Conference CSCW'94*, ACM Press, October 1994.

[REY98] 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.

[TAG91] Tagliante. C. , « L'évaluation » , Édition Clé international , France, 1991.

[TAL97] 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.

[WEN87] Wenger. E. , « Artificial Intelligence and Tutoring Systems », Addison-Wesley Eds., USA, 1987.

[WII91] Wiil U.K. , « Using Events as Support for Data Sharing in Collaborative Work », *Proceedings of the International Workshop on CSCW*, Berlin, 1991.

[ZID00] Zidani. A., Boufaida. M., Djoudi. M. , « JamEdit: un outil interactif et coopératif pour l'édition coopérative de documents », *Technique et Science Informatiques*, vol. 19, n°1, 2000, p. 1-23, Hermes, Paris.