Réalisation coopérative de systèmes tuteurs intelligents hypermédias

TALHI Said[1,2] , DJOUDI Mahieddine [2], BATOUCHE Mohamed [3]

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

[2]Laboratoire SIC et équipe ERTe IRMA, Université de Poitiers, France
mahieddine.djoudi@univ-poitiers.fr

[3]Laboratoire Lire, Université de Constantine, 25000 Constantine, Algérie

batouche@wissal.dz

Mots clés : Système auteur coopératif, collecticiel, tuteur intelligent hypermédia.

Résumé :

Dans cet article, nous présentons un système auteur pour la réalisation coopérative de systèmes tuteurs intelligents. Bâti sur une architecture client-serveur, ce système permet à plusieurs auteurs géographiquement dispersés de collaborer pour produire ensemble de tels tuteurs. Nous décrivons dans ce cadre la structure du tuteur engendré par le système, puis le mode auteur coopératif en présentant son architecture logicielle ainsi que les différents niveaux qu'elle recouvre.

Introduction

Les possibilités des technologies d'information et de communication apparues ces dernières années, notamment le réseau mondial Internet et le Web, et les nouveaux besoins d'apprentissage (apprentissage à distance, apprentissage tout au long de la vie, apprentissage coopératif assisté par ordinateur, etc.) modifient considérablement les caractéristiques des environnements d'apprentissage et les questions qui s'en dégagent (BARON, 2001). Un nouveau moyen d'appréhender ces environnements est alors de les considérer comme des environnements dans lesquels coopèrent des agents humains et des agents artificiels (DESPRES, 2001).

Les travaux de recherche que nous menons actuellement, participent à cette mutation. Ils contribuent à proposer de nouvelles situations d'apprentissage en prenant en compte les aspects Distance et Coopération entre apprenants et auteurs/tuteurs. La plate-forme Ibn Sina (ZIDAT, 2004) que nous développons dans ce contexte permet en effet de gérer un apprentissage coopératif à distance intégrant plusieurs fonctionnalités dont : l'assistance à la navigation, l'assistance à l'apprenant par tuteur humain, l'assistance à l'apprenant par tuteurs informatiques et enfin l'assistance aux enseignants auteurs pour réaliser ces tuteurs artificiels. Les deux dernières fonctionnalités sont assurées par un système auteur coopératif baptisé TALHITS (pour Teaching And Learning by Hypermedia Intelligent Tutoring System) qui fait l'objet de ce papier.

Dans la suite de ce papier, nous commençons par introduire le principe des systèmes auteurs en général et discutons de leur évolution vers des systèmes auteurs coopératifs. Nous présentons ensuite brièvement le modèle du système tuteur intelligent hypermédia (Hits), puis nous décrivons, le mode auteur coopératif (Camits) en présentant son architecture logicielle ainsi que les différents niveaux qu'elle recouvre.

Systèmes auteurs : du mode mono-usager vers le mode coopératif

Plusieurs travaux de recherche ont porté sur la réalisation de systèmes auteurs de STI durant cette dernière décennie. Murray (MURRAY, 1999) en a cité plus de deux douzaines dans sa synthèse la plus récente dont figure aussi son système Eon. 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*.

Talhits, le système présenté dans ce papier et qui offre également 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 systèmes auteurs étant apparus en principe pour faciliter la production de STI sans avoir à programmer. La tâche est alors généralement réduite aux experts de domaines à faire couler des connaissances dans un canevas de STI générique prédéterminé par le système. Cependant, nous avons remarqué que malgré les efforts consentis dans ce sens, la réalisation de STI demeure encore une tâche difficile à entreprendre. 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. Ainsi donc et même du point de vue acquisition des différents types de connaissances, cette construction nécessite souvent la coopération entre divers experts du domaine en question. Etant évidemment géographiquement éloignés dans la plupart du temps, il est par conséquent nécessaire de mettre à leur disposition un système de type collecticiel leur permettant de communiquer et coordonner leurs activités.

C'est dans cette optique de coopération que nous avons développé notre système auteur Talhits. Ce système est structuré en deux composants communiquant entre eux : le mode auteur coopératif (Camits) et le mode apprenant (Hits). L'architecture logicielle du mode auteur permet en effet aux auteurs de collaborer pour la production du STI.

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 constituent également une condition capitale quant au succès de ces collecticiels (GREENBERG, 1992). Ainsi, pour éviter les conflits inhérents à la nature humaine, nous avons proposé une organisation qui permet de mener la conduite du projet de construction collective du STI de manière rationnelle et optimale. Nous définissons alors quatre rôles à travers lesquels les participants peuvent intervenir au cours du processus de construction du STI : *auteur*

principal (chef de projet)*, coauteur superviseur* (supervisant un sous groupe d'auteurs)*, coauteur constructeur et coauteur lecteur/commentateur.*

Système tuteur intelligent hypermédia : HITS

Le système tuteur Hits est conçu sur la base de deux paradigmes : celui des hypermédias et celui des systèmes experts. L'analyse critique que nous avons réalisé sur ces deux paradigmes nous a révélé un certain nombre d'insuffisances pour chacun d'eux mais aussi des points forts.

- Un système expert est très limité dans son niveau d'expertise, long à développer, rigide et difficile à modifier, alors qu'un hypermédia est théoriquement illimité dans son expertise, développé rapidement et ses modifications et mises à jour sont faciles ;

- Un système hypermédia, au vu de sa liberté et sa flexibilité, est réputé par les deux problèmes de désorientation et de surcharge cognitive qui fait que l'utilisateur se perd souvent dans la chaîne de liens qui lui sont proposés et part souvent dans des liens inutiles en perdant de vue son objectif initial, alors qu'un système expert est réputé par le fait qu'il guide pertinemment l'utilisateur vers son objectif.

Ainsi, nous avons conjugué les bénéfices des deux paradigmes dans le but d'adapter le cours aux besoins et aptitudes de chaque apprenant. Ceci permettra par conséquent, à ce dernier, d'atteindre les objectifs d'apprentissage fixés par l'auteur du cours.

Le système Hits s'insère donc dans le courant des systèmes tuteurs qui organisent le processus d'instruction autour de composants hypermédias (UAH : pour Unités d'Apprentissage Hypermédias). La gestion des UAH dans le canevas de tuteur, est assurée par un système multi-expert basé sur un ensemble de paquets de règles. 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 tuteur créé par le module auteur 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, représentés sous forme de prédicats, décrivent l'aspect quantitatif de la matière à enseigner (nombre de parties, nombre de chapitres, nombre d'UAH, nombre de questions, nombre d'exercices, etc.).

Les UAH sont censées recevoir, par instanciation, toutes sortes de connaissances du domaine, sous toutes les formes de médias permises par le langage XML (texte, image fixe, image animé, son, vidéo, script, applet) (DE LA PASSARDIERE, 2001). En somme, deux niveaux de connaissances sont donc utilisés dans la définition du curriculum dans le système Hits:

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, estimation des acquis à l'issue de la phase de négociation, planification des enchaînements d'UAH, recherche et filtrage des UAH; diagnostic et évaluation de l'apprenant.

2. Un niveau inférieur correspondant à l'univers des UAH : Cet univers consiste en une structure d'arbre formé de six niveaux d'UAH. Les quatre premiers niveaux correspondent à des UAH de cours et les deux derniers sous-niveaux correspondent à des UAH d'évaluation.

Selon la logique **« dire, montrer, faire »**, les UAH de cours présentent la théorie sur le thème à enseigner **(dire)** et les exemples permettent de **(montrer)** à l'apprenant comment appliquer la théorie sur des exemples concrets. Les UAH d'évaluation quant à elles, permettent de mesurer l'atteignabilité des objectifs opérationnels par l'apprenant, et ce, en le poussant à **(faire)** lui-même des exercices et applications de la théorie qui lui a été présentée dans l'UAH de cours.

L'architecture de Hits (Figure 1) est similaire à celle d'un STI traditionnel (WENGER, 1987) elle comporte :

1. Un *module expert du domaine* qui permet, en utilisant les règles de recherche des UAH, de chercher, de filtrer et d'envoyer sous forme HTML l'UAH sollicitée par le système à un moment donné. Ceci est réalisé grâce aux feuilles de style XSL et l'exploitation des balises sémantiques délimitant les différents UAHs dans les contenus pédagogiques exprimés en XML.

2. 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 de négociation et le *planificateur* utilisant les règles de planification.

3. Un *module de diagnostic de l'apprenant* qui permet d'évaluer l'usager et d'assurer la maintenance d'un modèle-apprenant de type overlay. Ce module comporte à son tour trois sous-modules : un *évaluateur* utilisant les règles d'évaluation, un *déducteur des acquis* utilisant les règles de détermination des acquis et un *gestionnaire du modèle de l'apprenant* manipulant le contenu de ce dernier.

Pour adapter le cours à chaque apprenant le système Hits utilise la technique dite «planification d'enchaînement de cursus» (BRUSILOVSKY, 1997), Cette technique consiste à fournir à l'apprenant l'enchaînement le plus approprié des UAH en tenant compte de son profil et des objectifs pédagogiques à atteindre. Pour cela, XSLT nous permet de transformer le cours générique en un cours spécifique dépendant du modèle de l'apprenant. Le document XSLT spécifique à ce modèle ayant servi à cette transformation est lui-même le résultat d'une autre transformation d'un document XSLT contenant l'ensemble des paramètres possibles et le modèle contenant des valeurs spécifiques de paramètres de cet apprenant.

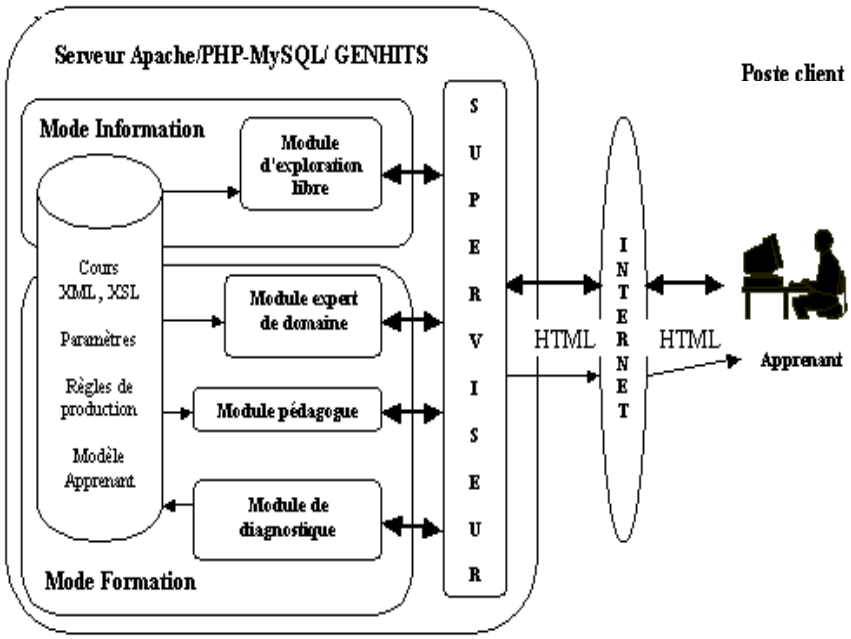

Figure 1. Architecture logicielle de HITS

Mode auteur coopératif : CAMITS

L'éditeur coopératif de connaissances (ou mode auteur coopératif) 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 de prérequis sous forme de graphes orientés, des paramètres du STI sous forme de prédicats, et des connaissances pédagogiques 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 (ZIDANI, 2000).

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* pour les auteurs dans Camits : 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.

Architecture logicielle

L'éditeur coopératif de connaissances est organisé selon une architecture client/serveur centralisée (ORFALI, 1997). Par conséquent, toutes les communications transitent automatiquement par le site central (ou serveur). A chaque site auteur on associe un processus client (PCL) qui accomplit toutes les tâches traitables localement (les tâches d'édition par exemple). Quant au niveau central, nous définissons un processus serveur (PSR) qui gère toutes les communications entre les différents PCL et tient à jour le contenu de la copie centrale et la structure logique du STI. Le PSR 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 PSR prépare aux PCL 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.

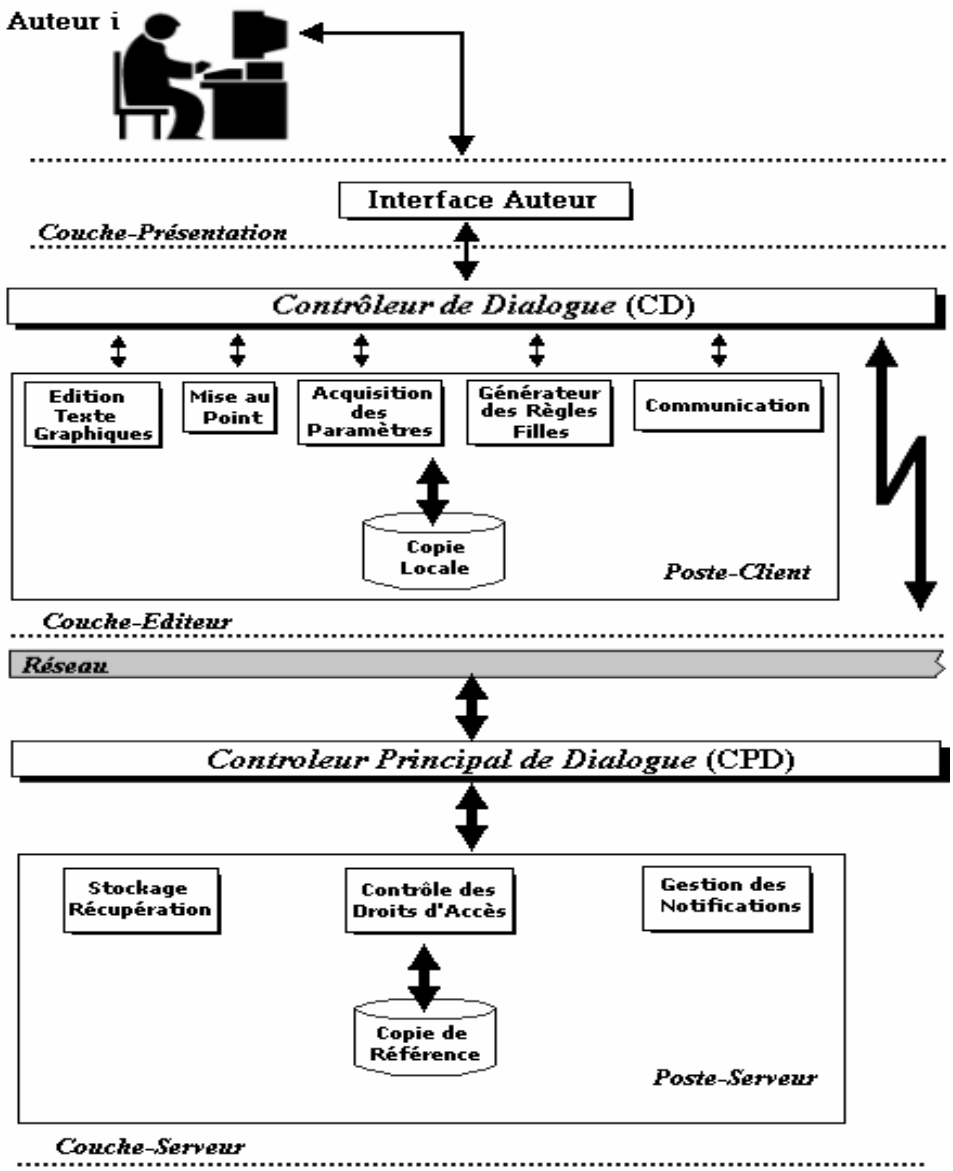

Figure 2. Architecture logicielle en couches du mode auteur

Comme le montre la (Figure 2), l'architecture logicielle 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 approprié. 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 (COUTAZ, 1994).

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 la couche de présentation et la couche d'édition un contrôleur de dialogue (CD), et entre la couche serveur et la couche d'édition le contrôleur principal de dialogue (CPD). L'arrivée d'un événement 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.

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 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.

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 des différents objets du STI. Un premier module permet à l'auteur d'éditer le réseau de 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 utilisés dans le réseau. Un premier niveau montre les prérequis entre parties, un second décrit les prérequis entre les chapitres d'une partie, et le troisième montre les prérequis entre les UAH d'un chapitre. Le deuxième module permet l'édition wisiwig des UAH en utilisant le langage XML.
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. Par exemple le prédicat nbuah (1,2,4) indique que le chapitre 2 de la partie 1 contient 4 UAH.
3. **Outil de mise au point :** Comme la plupart des systèmes auteurs, Talhits 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 STI exécutable sans erreurs, c'est à dire conforme au modèle imposé par Talhits. 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 matière stoquées.

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 déroulant, 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 (MARCUS, 1993). A part les menus déroulant 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.

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. 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.

Conclusion

Dans cet article, nous avons présenté la conception d'un système auteur coopératif de tuteurs intelligents. Ce système, appelé Talhits, 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 par différents apprenants à distance.

Le mode auteur Camits est conçu selon une architecture logicielle centralisée basée sur le concept client-serveur. Il 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 prérequis, 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.

Conscients de l'intérêt d'évaluer ce système dans des situations réelles d'apprentissage, nous souhaitons pouvoir recueillir (à court terme) des informations sur les activités effectives des enseignants et des apprenants. Ceci est d'une importance extrême pour nous et constitue un objectif double. Premièrement nous pouvons valider ou remettre en cause certains choix techniques ou pédagogiques. Ensuite, nous pourrons déterminer avec plus de précisions les ajustements et les adaptations devant être apportés aux outils intégrés. A ce sujet nous avons choisi une architecture modulaire et donc évolutive, dans le sens où elle facilite la conception de nouveaux modules et leur intégration de manière incrémentale.

Références

BARON M. (2001), « *Intelligence Artificielle et EIAH* », Ecole et Sciences Cognitives.

BRUSILOVSKY P. (1997), « *Integrating hypermedia and intelligent tutoring technologies: from systems to authoring tools*», in New media and telematic technologies for education, Twente University Press, Enschede.

DE LA PASSARDIERE B. & GIROIRE H. (2001), « *XML au service des applications pédagogiques*», Revue Sciences et Techniques Educatives, Volume 8, n° 1, pages 99-112.

DESPRES C. & GEORGE S. (2001), « S*upporting learners activities in a distance learning environment* », International Journal of Continuing Engineering Education and Lifelong Learning, Volume 11, n° 3.

GREENBERG S. & ROSEMAN K.& WEBSTER D.(1992) *«Human and technical factors of distributed group drawing tools, interacting with computers»*, Volume 4, n° 3,.pages 364-392.

MARCUS A. (1993), *« Human communications issues in advanced UIs »*, Communications of the ACM, Volume. 36, n° 4, pages. 101-109.

MURRAY T. (1999), *«Authoring intelligent tutoring systems : an analysis of the state of the art »*, International Journal of AI in Education, Volume. 10, pages 98-129.

ORFALI R. & HARKEY D.& EDWARDS J. (1997), *« Essential client/server survival guide »*, John Willeys & Sons Inc Eds, 2nd edition , New York.

WENGER E.(1987), *«Artificial intelligence and tutoring systems»*, Addison-Wesley Eds., USA,

ZIDANI A. & BOUFAIDA M.& DJOUDI M. (2000), *« JamEdit: un outil interactif et coopératif pour l'édition coopérative de documents »*, Technique et Science Informatiques, Volume 19, n°1, pages 1-23, Hermes, Paris.

ZIDAT S. & DJOUDI M. & ZIDANI A. & TALHI S. (2004), *«Collaborative Resolution of Exercises in Situation of Distance Learning»*, International Arab Conference on InformationTechnology (ACIT'2004), Constantine, Algeria.