Université de Montréal
Faculté des Études Supérieures

Spécifications en XML d'un langage de génération d'exercices pédagogiques

Par :
El Bachir BOUKHEROUAA

Département d'informatique et de recherche opérationnelle
Faculté des art et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès Sciences (M.Sc.)
en informatique

Octobre, 2000

Université de Montréal
la Faculté des études supérieures

Ce mémoire intitulé:

Spécifications en XML d'un langage de génération d'exercices pédagogiques

présenté par:

El Bachir Boukherouaa

A été évalué par un jury composé des personnes suivantes :

Nadia El Mabrouk	Présidente-rapporteuse
Claude Frasson	Directeur de recherche
François Lustman	Membre du jury

Mémoire Accepté le : 12 décembre 2000

SOMMAIRE

La formation à distance est un domaine qui évolue en parallèle avec l'évolution des hautes technologies. En effet, depuis les vingt cinq dernières années, l'éducation à distance est l'une des seules disciplines de l'éducation où la technologie est devenue la partie centrale de la tâche de l'enseignement. Les systèmes familiers de correspondances comme le téléphone, la radio, la télévision ont été rapidement remplacés avec l'addition des fibres optiques, des réseaux à grande échelle, le World Wide Web et une variété d'autres technologies qu'offrent les ordinateurs. Les apprenants d'un système d'éducation à distance peuvent maintenant recevoir électroniquement leurs tâches assignées, faire des exercices et interagir avec leurs professeurs et les autres apprenants.
L'arrivée de la vidéo conférence au sein d'un système d'éducation à distance peut créer un environnement d'apprentissage où les apprenants sont activement engagés [Collins et al., 1996]. La technologie avancée permet de créer un environnement d'apprentissage qui développe la confiance, la collaboration et la coopération [Collins et al., 1996]. Un apprentissage et une intégration plus élevés sont effectivement accomplis grâce au dialogue qui se produit avec l'usage de la vidéo conférence [Lauzon, 1992]. Les apprenants aiment le fait que l'anonymat facilite leur usage de la vidéo conférence. Ils sentent avoir plus de pouvoirs ce qui leur permet de défier, de confronter et d'exprimer leurs idées [Lauzon, 1992]. Les apprenants ne sont plus jamais mis sur-le-champ et peuvent réfléchir sur le contenu [McMahen et al., 1995]. Comparativement aux classes traditionnelles, les rôles des professeurs et des apprenants changent dans un système d'éducation à distance utilisant la vidéo conférence. Des études démontrent que le professeur contribue jusqu'à 80% de la discussion dans une classe traditionnelle et seulement 10% à 15% dans un système de télé-apprentissage [Jonassen et al., 1995]. Les apprenants doivent donc se renseigner, s'informer et faire des recherches qui dépassent

les quatre murs d'une classe [McMahen et al. 1995]. Des conférenciers peuvent aussi participer à la vidéo conférence ce qui enrichit l'expérience d'apprentissage par l'entremise d'experts [Collins et al., 1996]. Finalement, les frontières internationales sont abolies grâce à l'Internet et au World Wide Web. Un des points positifs de l'apprentissage à distance est la capacité, qu'a ce genre d'apprentissage, de pourvoir à un enseignement et à un accès à l'information au plus grand nombre d'apprenants possibles. Comparativement aux classes traditionnelles où les apprenants ont accès à un nombre limité de cours, les apprenants ayant accès à un ordinateur et à l'Internet sont connectés, eux, à une variété de cours sur différents sites, pourvu que ces cours soient accessibles au public.

L'arrivée des Systèmes Tutoriels Intelligents (STI), qui ont pour objet de réaliser, à l'aide d'un ordinateur, un enseignement individualisé et convivial, a révolutionné le monde de l'éducation. Ces systèmes s'adaptent dynamiquement à la formation en utilisant diverses stratégies pédagogiques selon les performances et le modèle de l'apprenant.

La réalisation d'exercices pédagogiques, dans un STI, est d'une extrême importance. En effet, sans outils d'évaluation, un système éducationnel ne peut déterminer avec précision les performances et les connaissances acquises par un apprenant. Notre travail consiste à réaliser des exercices pédagogiques variés, dans un STI, pour la diffusion de la formation spécifique à travers le World Wide Web. Pour que ces exercices puissent assurer une intégration parfaite dans une architecture STI, ils devraient présenter un certain nombre de caractéristiques, à savoir :

- une variété de choix,
- une validation exacte de la maîtrise du concept enseigné,
- une difficulté ascendante en fonction de l'évolution dans le cours,
- une présentation sous un format standard,
- une spécificité en fonction du cours traité,
- une portabilité sur plusieurs plate-formes.

Pour la manipulation de ces exercices à travers le réseau Internet et leur échange avec d'autres STI, nous avons mis en place un agent mobile appelé *'Gestionnaire d'exercices'* capable d'effectuer parfaitement ces tâches. En ce qui concerne la portabilité et l'interopérabilité de ces exercices entre les STI à travers Internet, le choix du langage

extensible markup language (XML) s'impose. Nous verrons plus loin que les possibilités offertes par ce standard nous permettent une grande souplesse quant à l'échange de données à travers le Web.

Mots clefs : Système Tutoriel Intelligent, exercices pédagogiques, XML, enseignement intelligemment assisté par ordinateur, représentation de connaissance, modèle de l'apprenant.

TABLE DES MATIÈRES :

LISTE DES FIGURES

Numéro

Page

REMERCIEMENTS

Tout d'abord, je voudrais remercier sincèrement M. Claude Frasson, mon directeur de recherche, de m'avoir fait confiance en m'intégrant au sein du laboratoire Héron.

Par la présente, je viens exprimer ma gratitude à Monsieur Claude Frasson. Si ce document a pu voir le jour, c'est grâce à ses conseils précieux, son soutien technique pertinent et sa disponibilité continue durant toute la période de mon projet.

La mise en œuvre de ce mémoire est le fruit d'un soutien financier de la part de M. Frasson.

Je tiens à remercier toute l'équipe du projet ainsi que tous les employés de la société Virtuel-Âge International inc.

Je tiens à remercier également ma femme Kenza de m'avoir soutenu tout le long de mes études et de m'avoir apporté ses précieux commentaires.

Je tiens à remercier également les membres du jury pour le temps consacré à ce mémoire.

Finalement, je remercie toutes les personnes qui ont contribué directement ou indirectement à la réalisation de ce travail.

CHAPITRE 1

INTRODUCTION

L'Enseignement Intelligemment Assisté par Ordinateur (EIAO) est un domaine de recherche très dynamique qui a pour objectif de développer des tuteurs informatiques utilisant des techniques d'Intelligence Artificielle (IA) pour dispenser l'enseignement. Ces derniers simulent la démarche pédagogique d'un tuteur humain et comportent certains aspects intelligents. L'EIAO a ajouté à l'EAO (Enseignement Assisté par Ordinateur), une individualisation de l'enseignement qui entraîne un accroissement du rendement pédagogique [Barr et Feigenbaum, 1989]. Les Systèmes Tutoriels Intelligents (STI) [Frasson et Gauthier, 1989] constituent des systèmes mettant en œuvre un tel enseignement. Ils varient beaucoup dans les techniques utilisées, dans leur approche pédagogique, et bien sûr selon la matière enseignée. L'architecture de tels systèmes n'est pas standard. Néanmoins, elle comporte généralement les principales composantes suivantes : une base de connaissances du domaine à enseigner, un modèle de l'apprenant, un module tuteur et une interface conviviale.

Le modèle de l'apprenant constitue une partie fondamentale sur laquelle le STI va se baser pour déterminer ses actions. En effet, il permet au STI de connaître l'état des connaissances de l'apprenant, par conséquent d'adapter ses interventions et en particulier de choisir des stratégies pédagogiques adéquates. Il évolue constamment et représente la

partie dynamique du système à partir de laquelle nous pouvons bâtir une interaction plus appropriée aux processus d'apprentissage de l'apprenant.

L'évolution de la technologie informatique couplée à l'évolution dans les technologies de l'éducation a entraîné une évolution dans les STI. L'informatique, à travers les concepts développés en intelligence artificielle, a permis d'adapter les résultats des recherches en éducation et en psychologie cognitive. Malgré tous ces développements, les STI ne répondent pas encore à toutes les préoccupations de l'homme. Les connaissances enseignées ont toujours eu un caractère individualiste. Ces connaissances ne nécessitent pas la présence de plusieurs humains dans le processus d'apprentissage ou dans sa réalisation effective. Ces STI présentent aussi une faiblesse quant aux moyens d'évaluation des apprenants.

1.1 Définition du problème

La réalisation d'exercices dans un environnement pédagogique constitue un point fondamental de l'apprentissage. En effet, une fois le cours donné, nous souhaitons tester adéquatement les apprenants sur les connaissances acquises. Pour qu'un STI soit complet et utilisable, il doit inclure une partie non négligeable sur la façon d'évaluer les apprenants. Pour cela, plusieurs questions se posent :

- Comment doit-on aider les concepteurs de cours à générer rapidement des exercices variés ?
- Quel est le meilleur moyen pour venir en aide à un apprenant en difficulté ?
- Quel système de notation choisir pour un type d'exercice donné ?
- Comment doit-on permettre la réutilisation de ces exercices entre plusieurs concepteurs ?
- Comment peut-on échanger des exercices entre plusieurs STI à travers le monde ? Sachant que ces STI n'ont pas la même plate-forme, ne parlent pas le même langage…

Pour répondre convenablement à toutes ces questions, nous avons subdivisé notre projet en deux grandes parties. La première partie concerne la conception et la réalisation d'un outil de développement d'exercices pédagogiques variés au sein d'un STI.

La seconde partie concerne l'accès à distance à ces exercices, et l'interopérabilité entre un réseau de STI à travers le monde. En effet, le partage de la connaissance entre un groupe d'individus ayant des intérêts communs permet l'enrichissement et la diversification de leur savoir-faire.

1.2 Cadre de réalisation

Notre projet de maîtrise s'insère dans le cadre d'un projet de recherche « Développement d'un STI basé sur les nouvelles technologies » développé au sein du laboratoire Heron. Le principal objectif du projet est de concevoir et réaliser un prototype d'un STI basé sur une architecture simplifiée à base d'agents intelligents.

Compte tenu de la complexité du projet, quatre étudiants en maîtrise ont pris part à sa réalisation.

Ce prototype a été implémenté en JAVA, ceci pour deux raisons, d'abord la forte diffusion du langage dans le monde, ensuite ses avantages techniques : masquage de l'hétérogénéité des machines et typage fort pour la sécurité.

Les contraintes rencontrées sont énormes puisque les individus qui vont échanger les exercices travaillent dans des environnements hétérogènes, parlent différents langages et utilisent des médias de stockage variés.

Pour ce faire, nous avons utilisé le standard XML qui permettra un stockage universel et une interopérabilité efficace.

CHAPITRE 2

LES SYSTÈMES TUTORIELS INTELLIGENTS

2.1 Introduction

L'apprentissage peut être considéré comme étant l'ensemble des activités amenant un individu à acquérir de nouvelles connaissances et de nouveaux comportements qu'il est capable soit de mettre en œuvre plus tard, soit d'exploiter ou de redéfinir [Péninou, 1993].

L'utilisation de l'ordinateur pour l'éducation et la formation a connu un important essor pendant la dernière décennie. Les recherches effectuées ont visé à comprendre les comportements humains et à les reproduire dans l'ordinateur. Les premiers systèmes conçus et développés dans ce contexte sont appelés systèmes d'Enseignement Assisté par Ordinateur (EAO) [Bestougeff et Fargette, 1982] [Lefevre, 1984]. L'objectif de ces systèmes est de favoriser l'apprentissage d'un domaine de connaissances par un apprenant. L'EAO suppose que la relation entre apprenants et enseignants se trouve

remplacée par une relation apprenant/machine, cette dernière jouant le rôle de l'enseignant.

Une forme plus récente de l'instruction par ordinateur est celle des Systèmes Tutoriels Intelligents (STI) [Frasson et Gauthier, 1989]. Les STI sont un exemple d'application des technologies de l'intelligence Artificielle (IA) à l'instruction. Bien que leur développement soit limité aux études de laboratoires, ils ont été offerts comme solutions prouvées aux problèmes techniques et fonctionnels dans la formation et la technologie éducative.

Dans ce chapitre, nous décrivons brièvement d'abord le développement historique de l'instruction assistée par ordinateur et les principaux aspects théoriques de l'Enseignement Intelligemment Assisté par Ordinateur (EIAO) [Sleeman et Brown, 1982]. Nous examinons en second lieu les composants des systèmes tutoriels intelligents. Troisièmement, nous passons en revue les stratégies tutorielles et les typologies de STI. Nous décrivons ensuite le développement des systèmes représentatifs. Finalement, nous présentons une architecture générale des STI.

2.2 Développement des systèmes tutoriels : de L'EAO vers L'EIAO

L'EAO, qui est l'utilisation interactive de l'ordinateur comme support d'enseignement, est apparue dans les années 1950. Comme son nom le suggère, il s'agit de donner à l'ordinateur un rôle de tuteur, de précepteur. Un échange didactique va s'instaurer entre un apprenant et un enseignant simulé par l'ordinateur. L'EAO se présente en gros comme une succession de 'information-question-réponse-commentaire' entièrement prévus et rédigés à l'avance par les auteurs du didacticiel [Arsac, 1987].

La majorité des systèmes d'EAO sont critiqués du fait qu'ils sont développés sans aucune base théorique. Pour leur part, les programmes d'EAO incorporent quelques principes d'enseignement sous une forme ou une autre. L'approche suivie pour la conception et la réalisation des systèmes d'EAO consiste à voir comment un bon professeur enseigne sa matière et ensuite écrire un programme avec des réponses toutes prêtes à une série de problèmes ou d'exercices relatifs au cours.

Les systèmes d'EAO présentent certaines limites et faiblesses comme par exemple :

- Le problème d'établir un dialogue avec l'apprenant (dans sa langue naturelle)
- L'incapacité d'améliorer la stratégie d'enseignement pour réduire la distance entre l'apprenant et l'expert du domaine
- La difficulté de décider ce qui devrait être enseigné par la suite
- L'incapacité de traiter les réponses de l'apprenant
- Le problème de diagnostiquer et de comprendre les erreurs et les incompréhensions de l'apprenant.

Depuis le début des années 70, les progrès réalisés en intelligence artificielle et le développement d'outils qui y sont associés, ont aidé les chercheurs à réaliser des systèmes d'EIAO, qui sont une nouvelle génération des systèmes d'EAO. Plusieurs techniques d'IA sont utilisées pour la réalisation de tels systèmes dont notamment, les méthodes de représentation des connaissances, le traitement du langage naturel, l'utilisation de processus inférentiels et aussi plusieurs autres techniques plus spécifiques à l'IA comme : l'intégration symbolique, les diagnostics médicaux, les preuves de théorèmes, etc.

En fait, les systèmes d'EIAO ont été conçus pour combler certaines lacunes des systèmes conventionnels d'EAO, par exemple : en plus d'avoir une représentation des connaissances de la matière à enseigner, en utilisant les techniques développées en IA, les systèmes d'EIAO sont capables d'engager un dialogue individuel avec l'étudiant et d'utiliser ses erreurs pour diagnostiquer les éléments incompris de la matière à enseigner.

Ils tiennent compte des réactions de l'étudiant, de ses erreurs et de ses intérêts. Les réponses et les interventions de ces systèmes sont plus 'intelligents' et plus individualisées. L'étudiant bénéficiera d'un cheminement personnalisé et d'une approche pédagogique plus adaptée à ses besoins. Ce comportement simule en fait celui d'un

expert-pédagogique humain avec tous les aspects intelligents que cette tâche requiert. Ces systèmes répondent au moins à trois critères [Burns et Capps, 1988] :

1. Ils comprennent et connaissent le domaine;
2. Ils implantent des stratégies d'enseignement pour réduire la distance entre les connaissances de l'apprenant et celles de l'expert du domaine.

2.3 Système tutoriel intelligent : définition et objectifs

Nous allons dans la suite donner la définition de STI du point de vue de certains auteurs.

2.3.1 Définition

Plusieurs définitions des STI sont données dans la littérature, certaines définitions insistent sur le diagnostic cognitif des actions de l'apprenant, la correction adaptative de ses erreurs, d'autres, sur la présentation des activités pédagogiques, l'environnement d'apprentissage, etc.

VanLehn (1988) par exemple définit un STI comme un programme informatique qui instruit l'apprenant d'une manière intelligente.

D'après Mandl et Lesgold (1988), un STI est un programme informatique qui peut faire des inférences sur les connaissances de l'apprenant et interagir intelligemment avec lui (sur la base d'une ou plusieurs représentations individuelles de ses connaissances).

D'après Greg Kearsley (1982), un STI est un système qui utilise les techniques d'IA pour aider une personne à apprendre.

Pour nous, le STI est un programme de formation complet (cours et exercices), basé sur les techniques de l'IA qui a pour but d'aider intelligemment un apprenant à acquérir des connaissances et pouvoir s'assurer de la maîtrise des dites connaissances.

L'individualisation de l'enseignement est alors une des caractéristiques et l'un des avantages les plus évidents des STI. Voyons ce qu'elle apporte à l'apprenant :

- *Rythme individuel* : dans le cadre des STI, chaque apprenant doit travailler à son rythme.

- *Participation active* : tous les messages sont adressés à l'apprenant et toutes les questions lui sont posées. Il doit y répondre et ses réponses sont analysées et commentées.

- *Correction immédiate et systématique* : dans le cadre des STI, la correction vient immédiatement. Le système permet également de diminuer la répétition systématique du même type d'erreur (durant un exercice) par l'apprenant.

- *Augmentation de la capacité de concentration de l'apprenant* : la nécessité de répondre à toutes les questions de l'apprenant et de lui envoyer systématiquement des commentaires par le système lui impose de se confronter à toutes les étapes du processus didactique et de constater la qualité de sa performance.

- *Intimité et anonymat* : dans un travail individualisé, l'apprenant se trouve seul et libre, il se sent donc à l'aise pour répondre aux questions.

2.3.2 Le but des STI

Le but d'un STI est de présenter la connaissance de façon claire, simple et accessible à l'apprenant, en faisant intervenir des éléments médiatiques et diverses stratégies pédagogiques, pour essayer de placer l'étudiant dans un environnement cognitif lié au domaine d'apprentissage.

2.3.3 Les domaines utilisés pour la conception des STI

La figure 1 montre les domaines utilisés pour la conception des STI. La conception et le développement de tels systèmes se trouvent à l'intersection de *l'informatique*, de la *psychologie cognitive* et de la *recherche en éducation*. Ce domaine est désigné souvent sous le nom de *sciences cognitives* [Polson et Richardson, 1988].

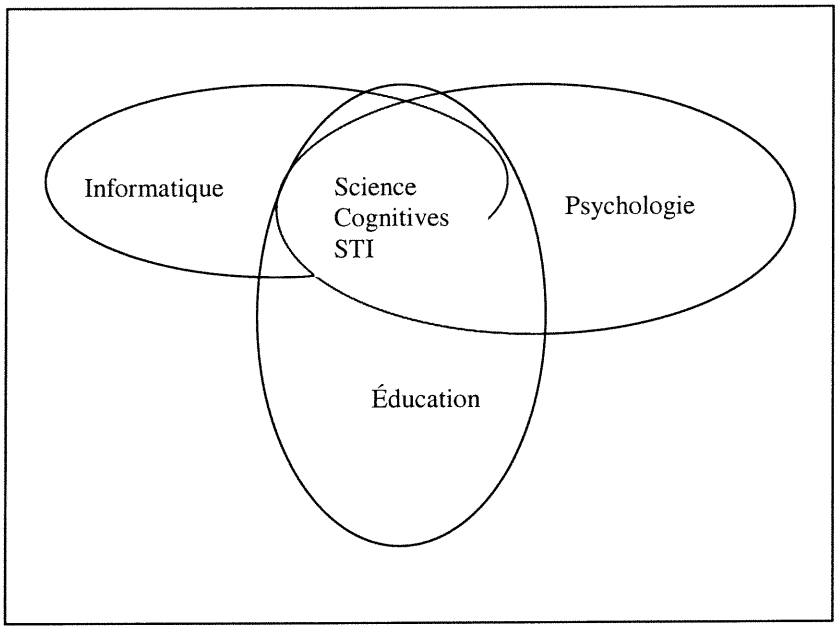


Figure 1 : Domaine de Sciences Cognitives

2.4 Architecture de base d'un STI

L'architecture de base d'un STI fait ressortir quatre modules de base [Burns et Capps, 1988] : *le modèle du domaine(module expert)*, *le modèle de l'apprenant*, *le modèle tuteur*, et *l'interface* (voir Figure 2).

2.4.1 Le modèle du domaine

Le modèle du domaine est le composant du STI qui modélise l'expertise du domaine. Il contient une représentation des connaissances du domaine à enseigner et un système qui permet de manipuler ces connaissances. Ce module permet au STI de résoudre les problèmes que le système pose à l'étudiant pour être ainsi en mesure d'interpréter les actions de ce dernier lorsqu'il résout un exercice. Plusieurs techniques sont utilisées pour représenter le domaine à enseigner [Barr et Feigenbaum, 1982]. Les principales sont les réseaux sémantiques, les graphes génétiques, les représentations procédurales et les représentations multiples.

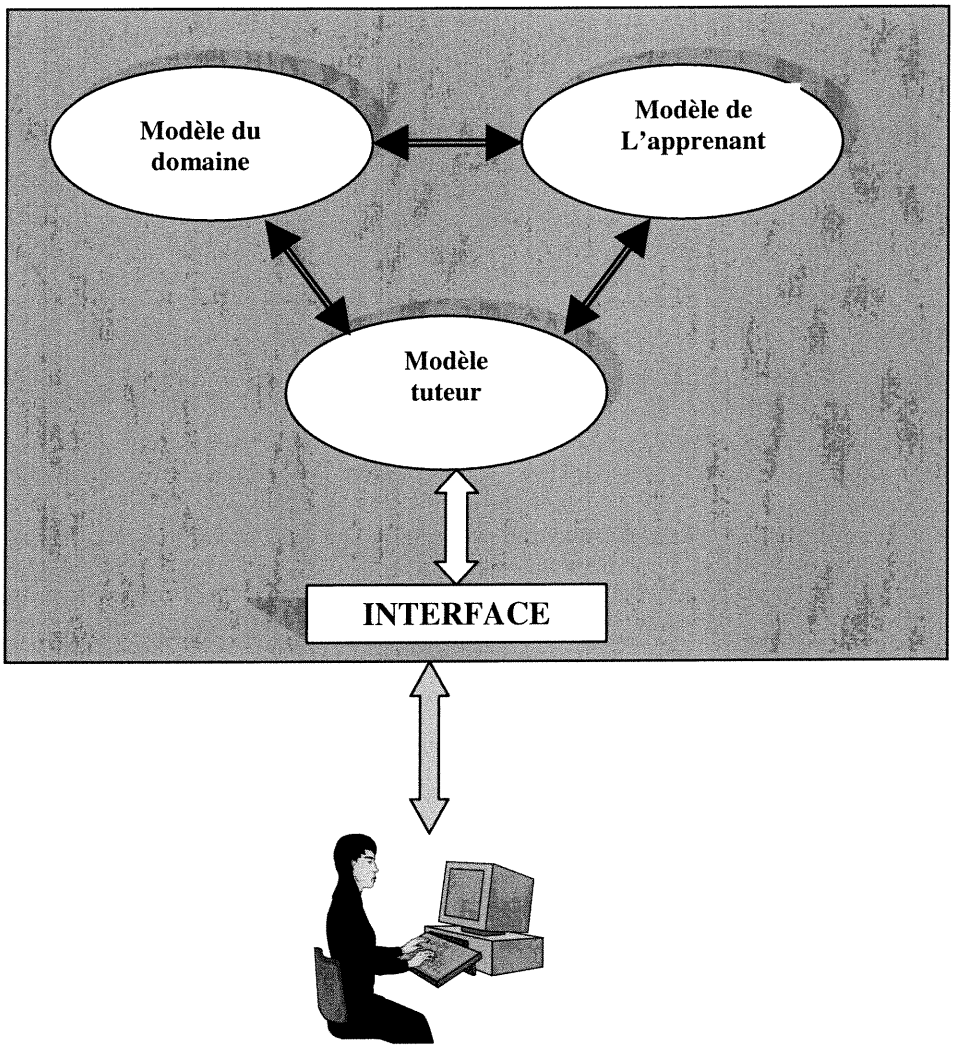


Figure 2 : Architecture classique de STI

2.4.2 Le modèle de l'apprenant

Le modèle de l'apprenant est utilisé pour évaluer l'état des connaissances de l'étudiant et le STI utilise les informations contenues dans ce modèle pour ajuster ses stratégies pédagogiques.

En général, un *modèle de l'apprenant* contient une représentation de l'état des connaissances de l'étudiant et des règles permettant de diagnostiquer cet état. Il est à

noter que, le processus qui forme et met à jour le modèle de l'apprenant en analysant des données rendues disponibles au système, s'appelle souvent le diagnostic.

En effet, la séparation de deux composantes du modèle de l'apprenant permet au STI de présenter à l'étudiant une plus grande variété d'activités d'apprentissage, certaines permettant l'acquisition de connaissances et d'autres simulant l'acquisition de nouvelles compétences au regard de la mise en pratique des connaissances acquises.

2.4.3 Le modèle tuteur

Le *module tuteur* est la composante principale d'un STI qui gère le fonctionnement global du système. Il est chargé de communiquer avec l'étudiant, de sélectionner les problèmes qu'il va lui poser, d'analyser ses réponses, de fournir une rétroaction quand c'est nécessaire et de présenter à l'étudiant la solution des problèmes posés [Barr et Feigenbaum, 1982].

C'est lui qui doit décider à chaque instant quelle action spécifique il y a lieu d'entreprendre dans la situation présente, en appliquant les techniques d'enseignement appropriées, au moment opportun durant la séance d'apprentissage. Ces techniques d'enseignement sont connues sous le nom de stratégies tutorielles du STI. Le tuteur doit guider l'enseignement sans pour autant paraître trop directif et/ou autoritaire.

2.4.4 Le module interface

Pour être en mesure de fournir une interface diversifiée, un STI dispose d'une composante (Interface) pour l'interaction avec l'usager. En général, un environnement d'apprentissage comprend un micro-monde et/ou une Interface. Cet environnement d'apprentissage, comme son nom l'indique, supporte les apprentissages de l'étudiant en permettant la mise en application des concepts appris et des interventions pédagogiques de la part du STI. Des études ont montré que l'Interface Homme-Machine joue un rôle majeur dans l'apprentissage. En particulier, une interface pauvre et confuse influence négativement l'apprentissage et l'enseignement, en revanche une interface bien conçue améliore la formation [Miller, 1988]. En ce sens, l'interface doit être la plus transparente

et conviviale possible dans le but de faciliter l'apprentissage. Elle doit inclure, en plus du texte, les images, le son, la vidéo, etc.

2.5 Les stratégies tutorielles

Dans le domaine des STI, plusieurs stratégies pédagogiques, couvrant une variété d'interactions entre le tuteur et l'apprenant, ont été développées.

Définition : Une stratégie pédagogique tente de reproduire le comportement d'un spécialiste en pédagogie qui adapte le rythme de son enseignement à celui de l'apprenant [Holt et Wood, 1990].

Nous allons décrire ici quelques stratégies tutorielles mises en œuvre par des systèmes existants. Il faut noter que ces stratégies permettent au tuteur de réagir de la bonne façon au bon moment [Frasson et Gauthier, 1994]. Pour chaque stratégie, nous allons énumérer les agents impliqués, leurs rôles respectifs, et les étapes éducatives touchées.

2.5.1 Tuteur classique (ou mode directif)

Dans cette stratégie on retrouve deux agents : un tuteur simulé par l'ordinateur et un étudiant humain, le tuteur démontre des concepts et des tâches, et l'apprenant résout des problèmes simples sous la supervision du tuteur qui le conseille. Enfin, le tuteur présente à l'apprenant une critique de sa performance.

2.5.2 Co-apprenant

Cette stratégie implique deux agents : une personne qui étudie et un co-apprenant simulé par l'ordinateur [Self, 1988]. Le co-apprenant devient un démonstrateur de niveau intermédiaire, d'un point de vue conceptuel, le co-apprenant a récemment acquis la présente leçon, il a donc le modèle explicite du processus encore en mémoire et il aide l'apprenant à suivre le même chemin de construction. L'accent est mis ici sur la construction de la connaissance et non sur sa transmission [Aïmeur et Frasson, 1996].

2.5.3 Compagnon d'apprentissage

Cette stratégie implique trois agents : l'apprenant, un ou plusieurs compagnons d'apprentissage (qui peuvent être simulés ou humains), et un tuteur lui aussi simulé. En général, le compagnon possède à peu près le même niveau de connaissances que celui de l'apprenant, seuls ses processus de raisonnement diffèrent. L'apprenant et le compagnon travaillent ensemble sur la même tâche, sous la supervision du tuteur. Pendant la phase d'exercices, l'apprenant peut être conseillé par le compagnon, ou peut lui-même le conseiller, ce qui augmente sa motivation [Chan et Baskin, 1990].

2.5.4 Tuteur inversé

C'est une autre stratégie qui implique deux agents : l'apprenant et un compagnon simulé par l'ordinateur. Dans cette approche, c'est l'apprenant qui est supposé jouer le rôle du tuteur et explorer ses connaissances de manière claire et précise pour que le compagnon simulé puisse les comprendre [VanLehn et al., 1994]. De plus, il doit corriger les fautes du compagnon, ce qui l'aide à formuler clairement ses connaissances.

2.5.5 Le perturbateur

Le perturbateur est un agent (qui a l'apparence d'un compagnon) qui conseille l'apprenant pendant son travail; ses conseils peuvent être valides ou non. En réalité c'est un tuteur déguisé qui est spécialisé dans la pédagogie et s'entend avec le tuteur sur les points d'intervention. Le but de cette stratégie est de tester la confiance de l'apprenant en ses connaissances [Aïmeur et Frasson, 1996]. Pour être pédagogiquement efficace, le perturbateur se doit de proposer des conseils-pièges qui serviront à mettre en valeur certaines subtilités de la tâche, qui passeraient sinon inaperçues.

2.5.6 Apprentissage par double test

La stratégie du 'Double Test Learning' (DTL) [Aïmeur et Frasson, 1998] est une nouvelle stratégie tutorielle qui s'appuie sur la théorie de l'apprentissage social. Elle comporte quatre étapes : le Pré-Test, la Formation, le Post-Test1 et le Post-Test2. Dans la première étape, le tuteur pose à l'apprenant une série de questions qui lui permettent

d'évaluer ses réponses afin de créer le modèle de l'apprenant. Ensuite, lors de la formation, le tuteur présente à l'apprenant et au co-apprenant une série de problèmes et leurs solutions afin de permettre une acquisition de connaissances. Dans la troisième étape, l'apprenant observe la co-apprenant qui interagit avec le tuteur et il écoute les questions du tuteur et les réponses du co-apprenant. Lors de la dernière étape, l'apprenant répond aux questions (certaines de ces questions sont les mêmes que celles du post-test1) ce qui permet au système de détecter s'il a pu profiter des erreurs commises par le co-apprenant lors du Post-Test1. À la fin, le tuteur donne des explications aux questions non réussies par l'apprenant.

La figure 3 montre l'évolution des stratégies tutorielles dans les STI.

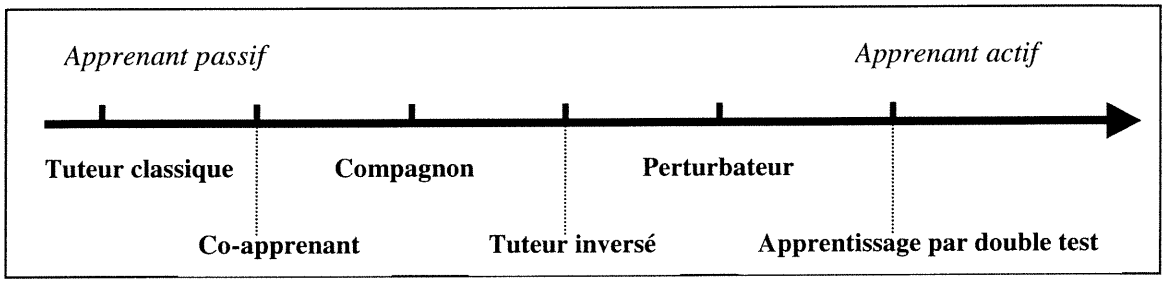


Figure 3 : Évolution des stratégies tutorielles [Aïmeur et Frasson, 1996]

2.6 Typologie des STI

Plusieurs types de systèmes tutoriels intelligents sont développés, chaque type peut utiliser une ou plusieurs stratégies tutorielles. Les différents types de STI accessibles dans la littérature sont les suivants :

2.6.1 Les STI socratiques

Par la méthode Socratique, le système engage l'apprenant dans un dialogue qui lui permet de découvrir ses erreurs et ses contradictions. Une décision pédagogique est prise en fonction du résultat du dialogue. Ce type de STI est très approprié pour présenter les informations factuelles et aussi pour l'apprentissage de règles de principe ou de stratégies de résolution de problèmes [Alessi et Trollip, 1985]. WHY et SCHOLAR sont des

systèmes qui utilisent un dialogue socratique pour assurer l'enseignement [Stevens et Collins, 1977].

2.6.2 Les STI de type guidage (Coach)

Dans ce cas, pour amener l'apprenant à s'apercevoir de ses propres erreurs, la présence d'un tuteur intelligent (coach) s'avère nécessaire afin d'évaluer constamment les décisions de l'apprenant pour le guider plus ou moins explicitement en lui faisant percevoir les conséquences. Cette technique consiste à guider l'apprenant plutôt que de lui dicter ce qu'il doit faire. Le système ne doit pas intervenir trop souvent et doit laisser l'apprenant se 'débrouiller' tout en lui donnant quelques informations utiles quand cela est nécessaire.

2.6.3 Les STI démonstrateurs

Le but de ces systèmes est de présenter des séquences de tâches à effectuer dans un contexte de formation [Alessi et Trollip, 1985]. Ils n'enseignent pas vraiment au sens des STI socratiques et des STI procéduraux, mais se contentent de montrer quelque chose à l'apprenant. Ils permettent à l'apprenant de voir et de comprendre le contexte réel étudié, en se basant sur une simulation (souvent interactive) du monde modélisé (on ne peut parler vraiment de STI pour ces systèmes, car l'intelligence est encapsulée dans la simulation). Nous trouvons ce principe dans certains systèmes experts, comme MYCIN [Shortliffe, 1976].

2.6.4 Les systèmes critiques

Le but de ces systèmes est de critiquer les apprenants dans la résolution de problèmes (et d'amener l'utilisateur vers une solution correcte) [Silverman, 1992]. Le mode critique analyse les diverses étapes que l'étudiant a suivi pour résoudre le problème et critique ses actions pour 's'assurer' qu'il a vraiment acquis la connaissance du monde. SOPHIE [Brown et al., 1982] et STEAMER [Hollan et al., 1987] sont des exemples des systèmes critiques.

2.6.5 Les systèmes sociaux

Les systèmes sociaux sont des STI qui expérimentent différentes combinaisons d'agents (par exemple : un ou plusieurs étudiants, un ou plusieurs enseignants, etc.). Ces agents communiquent lors d'une session d'apprentissage. Chaque agent par exemple, peut donner son point de vue pendant la résolution d'un problème. Un exemple de ce type de système est le système compagnon [Chan et Baskin, 1990].

Au cours des dernières années, des dizaines de systèmes tutoriels intelligents ont été développés. La liste est tellement longue qu'on ne peut tous les citer.

2.7 Les STI intégrés

Les différents types de STI discutés précédemment ont chacun des aspects que d'autres n'ont pas, et qui sont pourtant des aspects nécessaires pour un enseignement ou un apprentissage de qualité ; ce sont donc des systèmes complémentaires. Leur intégration dans un même système peut s'avérer très utile pour augmenter la puissance des STI. L'environnement modulaire proposé par Frasson et al. (1996a) va dans ce sens.

2.7.1 Architecture générale d'un STI

Plusieurs architectures de STI existent dans la littérature [Mengelle et Frasson., 1996] [Herzog, 1992] et [Frasson et al., 1997]. Les modules similaires dans ces architectures peuvent être différents mais ils effectuent souvent des tâches similaires (souvent certains modules sont combinés).

Dans ce qui suit, nous présentons l'architecture SAFARI qui a été proposée par le groupe de recherche Héron [Frasson et al., 1996a] . Cette architecture est décrite à la figure 4.

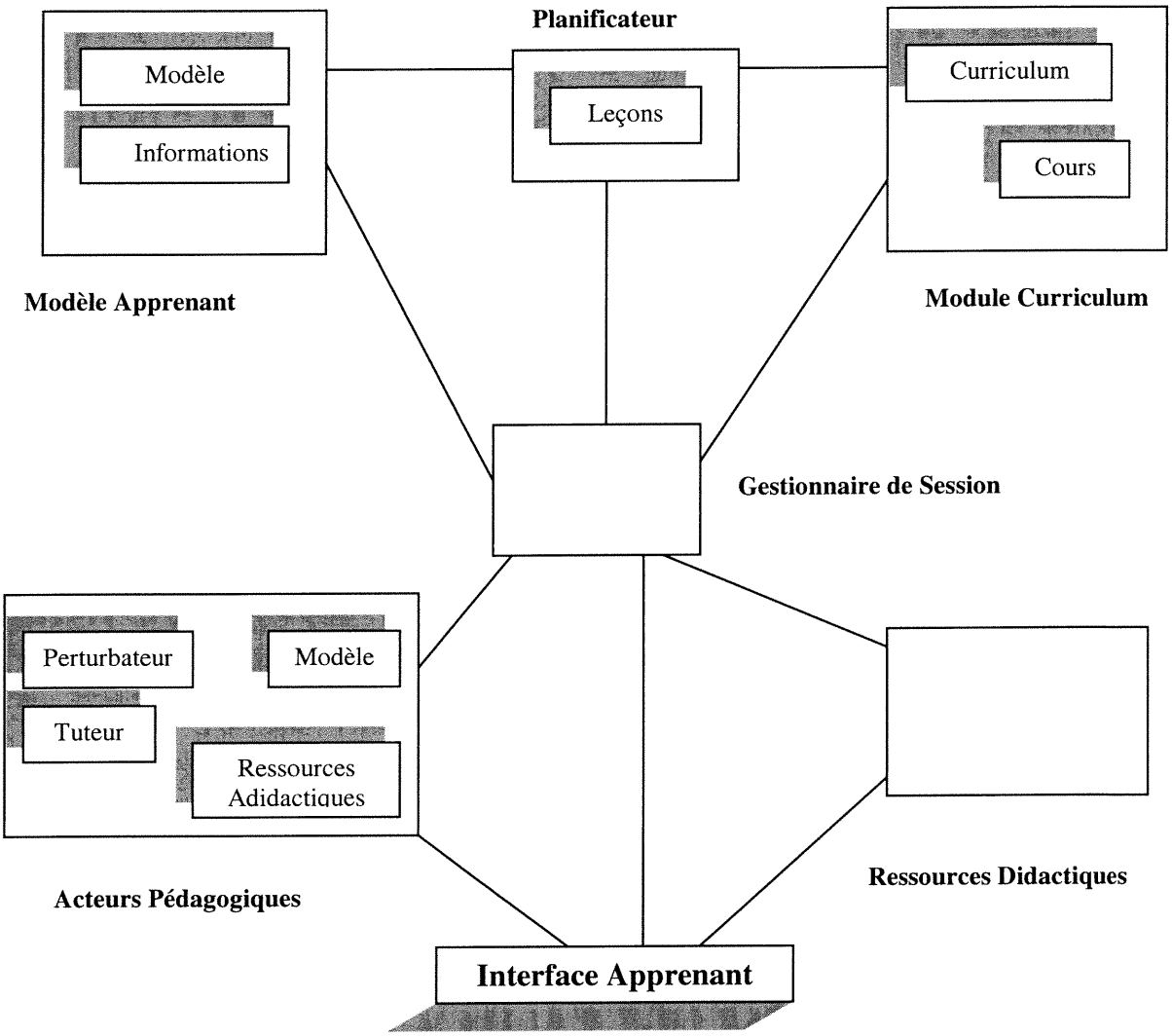


Figure 4 : Architecture Safari [Frasson et al., 1996a].

2.8 Conclusion

Ce chapitre nous a permis de faire un tour d'horizon des STI et de leur évolution depuis qu'on a pensé à introduire les techniques de l'informatique dans la formation et l'apprentissage. Mais des difficultés liées au développement des STI persistent de nos jours. Nous en énumérons quelques unes :

- Acquisition des connaissances d'un ou plusieurs enseignants dans un domaine donné.
- Encodage de ces connaissances de manière à les communiquer aux apprenants.
- Présentation d'exercices adaptés pour les différents contextes d'un concept enseigné.

Tous les STI développés jusqu'à présent n'ont pas accordé beaucoup d'importance quant à la manière d'évaluer les apprenants.

Les exercices pédagogiques sont d'une importance capitale dans un environnement d'apprentissage. Prenons comme exemple l'architecture Safari (Figure 4), nous avons remarqué que beaucoup de travail reste à faire dans la partie qui concerne les ressources didactiques. L'architecture se limite à souligner la nécessité d'évaluer les apprenants pour savoir s'ils ont bien acquis les connaissances, mais plusieurs questions restent sans réponse :

- Comment doit-on aider les concepteurs de cours à générer rapidement des exercices variés ?
- Quel est le meilleur moyen pour venir en aide à un apprenant en difficulté ?
- Quel système de notation choisir pour un type d'exercice donné ?
- Comment doit-on permettre la réutilisation de ces exercices entre plusieurs concepteurs ?

- Comment peut-on échanger des exercices entre plusieurs STI à travers le monde, sachant que ces STI n'ont pas la même plate-forme, ne parlent pas le même langage et ne possèdent pas le même système d'exploitation ?

CHAPITRE 3

GÉNÉRATION D'EXERCICES AU SEIN DE NOTRE STI

3.1 Architecture de base de notre STI

3.1.1 La notion de concept

La notion de concept est très importante pour la compréhension de notre projet. Nous avons défini le concept comme étant une idée simple, une connaissance ou une notion théorique indivisible à partir de laquelle, par composition, nous pouvons exprimer des idées plus complexes. Prenons comme exemple un cours de bases de données relationnelles, le terme « Champs dans une table ou colonne » peut représenter un concept avec comme définition : L'attribut qui désigne une propriété relative à un lot d'information (nom de la personne = Bouchard).

3.1.2 La notion de l'unité cognitive

Nous appelons une unité cognitive toute idée complexe composée d'un ensemble de concepts utiles et nécessaires à la définition de cette unité cognitive. Cette abstraction

s'est avérée très intéressante, elle nous a permis de découpler les exercices et les explications des cours, favorisant ainsi la réutilisation des ressources développées. En effet, deux cours partageant une même unité cognitive partagent également les exercices et les explications concernant cette unité cognitive. Le fonctionnement par unité cognitive nous a également permis d'adapter le cheminement de l'apprenant en fonction des connaissances acquises dans les autres cours. Comme exemple, nous considérons encore une fois les bases de données. Nous prenons la notion de « **Jointure de deux tables**» comme unité cognitive à enseigner. Cette même unité cognitive aura d'autres sous unités cognitives afférentes comme « **Equijointure** », « **Jointure naturelle**», etc. La figure 5 ci-dessous montre bien les relations qui existent entre un cours, une unité cognitive et un concept.

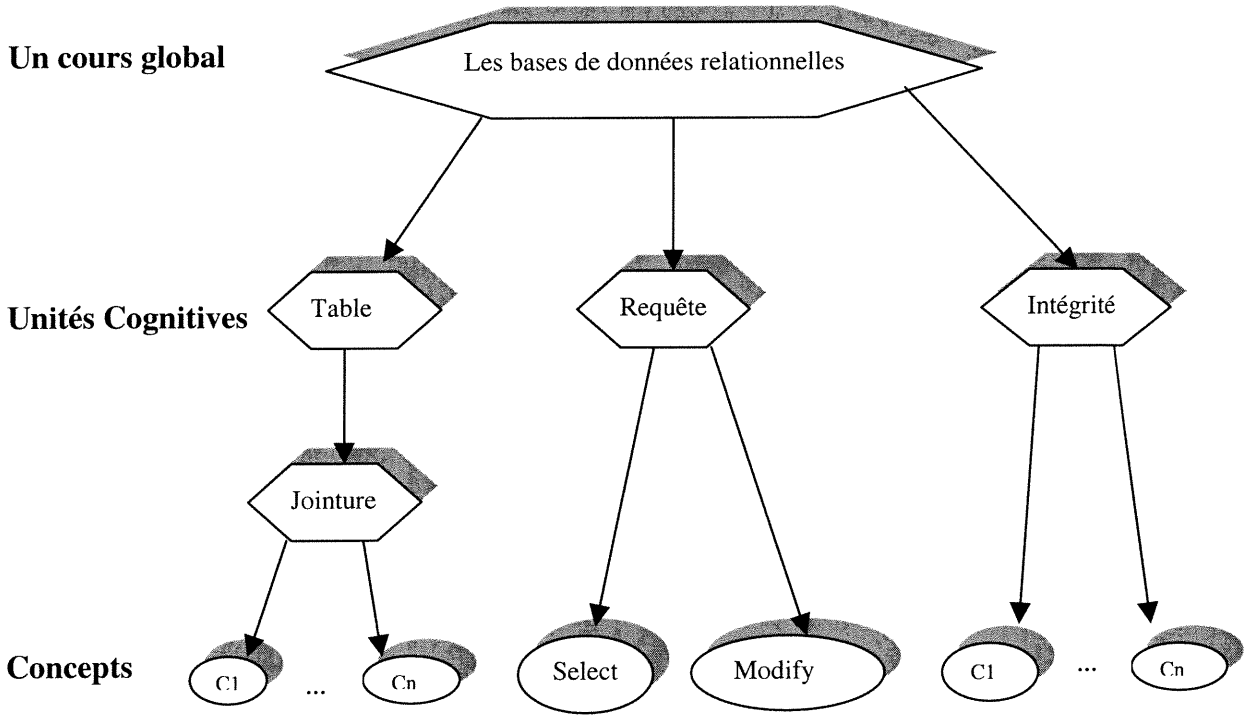


Figure 5 : Les composantes d'un cours.

Comme nous pouvons le constater à la figure précédente, le cours lui-même représente une unité cognitive globale constituée de l'ensemble des autres unités cognitives. Donc, une unité cognitive peut contenir une ou plusieurs autres unités cognitives. Chaque unité cognitive possède plusieurs niveaux de maîtrise qui déterminent son acquisition.

- ***Ressources d'une unité cognitive***

Une unité cognitive est caractérisée par un nom ou titre qui identifie l'unité et une ressource qui donne le contenant et l'aboutissant de l'unité en question. Cette ressource est sous la forme d'un fichier Html local. Le choix de l'Html a été fait d'une part, pour pouvoir enrichir le contenu en exploitant toutes les ressources multimédia à savoir texte, images, son, vidéo, etc., et d'autre part, pour faciliter l'utilisation de ces ressources à travers le World Wide Web (le Web).

Pour l'exemple précédent de l'unité cognitive « **Jointure de deux tables**», on aura une page Html ressource correspondante contenant les informations suivantes :

➔ *La jointure de deux relations R et S suivant la relation i θ j dans laquelle θ est un opérateur (=, #, <, >) i et j les rangs des attributs respectivement dans R et S, est l'ensemble des tuples du produit cartésien R × S qui satisfont la relation i θ j. On la note :* $\mathbf{R} \underset{i\,\theta\,j}{\textcircled{\Large\ast}} \mathbf{S}$.

i et j peuvent être remplacés par le nom des attributs correspondants.

Exemple : R :

A	B	C
1	2	3
4	5	6
7	8	9
0	1	2

S :

D	E
3	1
6	2

➔ Les relations suivantes R1 et R2 sont le résultat des jointures $\mathbf{R} \underset{B < D}{\textcircled{\Large\ast}} \mathbf{S}$ et $\mathbf{R} \underset{C = D}{\textcircled{\Large\ast}} \mathbf{S}$

Respectivement.

R1 :

A	B	C	D	E
1	2	3	3	1
1	2	3	6	2
4	5	6	6	2
0	1	2	3	1
0	1	2	6	2

R2 :

A	B	C	D	E
1	2	3	3	1
4	5	6	6	2

3.1.3 La notion de groupe d'explication

Un groupe d'explications est un ensemble d'explications relatives à une seule unité cognitive ou à un concept. La notion de groupe a été mise en place afin de pouvoir donner des explications selon un degré de raffinement. L'explication concernant un niveau de difficulté sera donnée en fonction du modèle de l'apprenant et l'avancement dans le cours. Une explication peut être textuelle saisie par le concepteur, fichier Html local conçu par le concepteur contenant texte, images, son, vidéo, etc. ou un lien vers un site distant dans lequel se trouvent toutes les notions de base à la compréhension de ladite unité cognitive ou dudit concept.

3.1.4 Architecture globale de notre projet

Nous décrivons à la figure 6 suivante l'architecture globale de notre système tutoriel Intelligent développé au sein de la société Virtuel-Âge International inc. C'est une architecture basée sur la technologie des agents, qui a pour avantage de réduire le va-et-vient entre le client et le serveur dans le cas d'une architecture orientée réseau. Comme nous pouvons le constater, nous faisons collaborer un certain nombre d'agents pour assister au mieux l'apprenant. Il s'agit de l'agent pédagogique qui contrôle la progression de l'apprenant dans le cours, l'agent explicatif qui est chargé d'apporter l'explication la plus efficace et qui correspond au mieux au profil de l'apprenant, l'agent gestionnaire d'exercices qui a pour mission de présenter les exercices les plus pertinents, afin d'avoir une meilleure évaluation des concepts non maîtrisés, et enfin l'agent de service qui s'occupe des accès aux données pour assurer l'intégrité et la cohérence des données de la base.

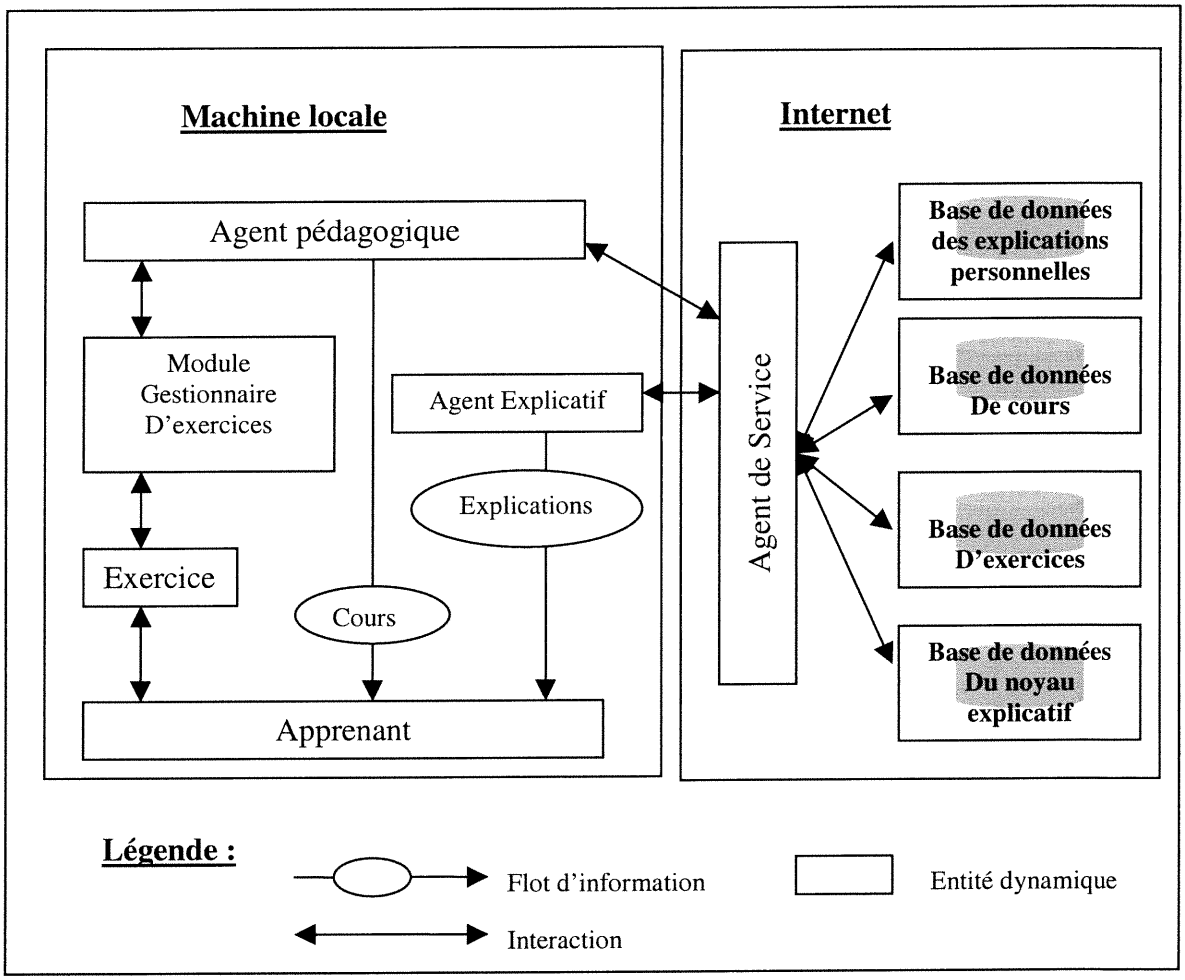


Figure 6 : Architecture globale de notre STI.

L'apprentissage au sein d'un réseau de télécommunication mettant en jeu des apprenants et un tuteur peut être amélioré considérablement en faisant appel aux systèmes tutoriels intelligents et à la technologie des agents. Dans notre projet, nous avons établi les composants systèmes utiles pour permettre à un tuteur intelligent de multiplier ses possibilités. Ceci est réalisable si nous confions les tâches d'explications et sessions d'exercices relatives à un cours à des agents spécialisés.

3.1.5 Agent pédagogique

Le rôle de l'agent pédagogique est de superviser en local, c'est à dire sur l'ordinateur de l'usager, l'apprentissage d'un apprenant. Pour ce faire, l'agent pédagogique utilise un profil de l'apprenant. En fonction de ce profil, cet agent gère efficacement la progression de l'apprenant dans son cours. Conceptuellement, l'agent pédagogique est doté d'un certain nombre de stratégies pédagogiques utilisées pour favoriser l'apprentissage de l'apprenant selon son profil. Afin d'adopter un tel comportement, l'agent pédagogique doit être en mesure d'observer chaque action prise par l'apprenant. À chaque observation, l'agent pédagogique met à jour le profil de l'apprenant. Au fur et à mesure que le profil de l'apprenant s'affine, les connaissances exactes de l'apprenant sont mieux reflétées et les décisions prises par l'agent pédagogique sont meilleures.

Dans le prochain chapitre, nous donnons une description plus détaillée sur la théorie des agents.

3.1.6 Le profil de l'apprenant

Le profil de l'apprenant est la composante qui représente l'état des connaissances de l'apprenant. Le profil de l'apprenant est *un regroupement d'informations identifiant et modélisant un apprenant sous divers aspects*. Concrètement, il s'agit d'une structure de données complexes, associées à chaque apprenant qui travaille avec le système. Dans cette section, nous introduirons la modélisation du profil de l'apprenant telle que conçue et implémentée dans le cadre de notre projet.

- **Un modèle de représentation du profil de l'apprenant dans un STI**

Conceptuellement parlant, le profil de l'apprenant est le point de départ du développement de notre STI. Il permet à ce dernier de modifier dynamiquement son comportement en fonction de l'apprenant avec lequel il travaille, satisfaisant ainsi le critère d'adaptabilité au niveau apprenant. De manière générale, disons simplement que notre STI mémorise, à l'aide du profil de l'apprenant, toutes les informations nécessaires afin de pouvoir déterminer statistiquement quelles sont les interventions du système qui

favorisent l'apprentissage de l'apprenant. Ceci permet une meilleure convivialité du système.

Les informations contenues dans le profil de l'apprenant se divisent en 2 catégories, soient l'information d'identification et l'information sur le cheminement. Voici une brève description de chaque catégorie d'information.

Information d'identification:

Ces informations permettent au système d'identifier de façon unique chaque apprenant :

- nom complet de l'apprenant
- prénom de l'apprenant
- nom d'usager utilisé pour la reconnaissance dans le système
- mot de passe obligatoire pour mesure de sécurité.

Information sur le cheminement:

Ces informations permettent au système de modéliser la progression de l'apprenant par rapport à l'ensemble des cours offerts par le STI :

- liste des cours suivis
- degré de complétude de chaque cours
- degré de maîtrise des unités cognitives suivies
- séquence des explications reçues
- liste des exercices déjà reçus avec les scores obtenus.

3.2 Module gestionnaire d'exercices (MGE)

Le module gestionnaire de l'exercice permet, d'une part d'aider le concepteur de cours à construire des exercices adaptés et variés, d'autre part, présenter ces exercices à l'apprenant en analysant bien ses réponses afin de communiquer les résultats obtenus à l'agent pédagogique. Dans notre projet, l'MGE observe l'apprenant et intervient auprès de lui lors de la réalisation d'exercices. Pour ce faire, l'MGE doit pouvoir extraire des exercices les informations pertinentes pour les communiquer à l'agent pédagogique. Ce dernier utilise les données sur les performances de l'apprenant pour décider de la prochaine étape à suivre.

3.2.1 Le module gestionnaire d'exercices orienté concepteur

C'est un module appelé à chaque fois qu'un concepteur désire construire un exercice relatif à une unité cognitive. L'exercice doit appartenir à l'un des niveaux de maîtrise définis dans l'UC. En effet, chaque niveau de maîtrise possède un certain nombre d'exercices avec un degré de difficulté ascendant. Le passage d'un niveau à un autre se traduit par la réalisation de tous les exercices définis dans ce niveau. Ce sont ces exercices qui permettent à l'agent pédagogique de tester le niveau de l'apprenant dans l'UC courante. Les exercices sont classés selon deux principales catégories : *les exercices génériques et les exercices spécifiques*. Les premiers sont applicables à tout genre de cours développé par les concepteurs (choix multiples, problèmes à étapes,...) nous verrons plus loin l'ensemble des exercices génériques qui ont été développés dans le cadre de notre projet. Les seconds sont des exercices réalisés sur mesure pour un cours particulier. Dans le cadre d'un cours sur SQL (Structured Querry Language, langage d'interrogation des bases de données) nous avons mis en place un exercice dans lequel l'apprenant doit saisir une requête SQL pour répondre à un besoin spécifique.

3.2.2 Caractéristiques communes des exercices

Plusieurs types d'exercices ont été définis et réalisés dans ce projet. Ils possèdent des données communes, tout exercice doit avoir :

- Titre : un titre unique pour chaque exercice
- Description : une brève description décrivant l'objectif de l'exercice
- Niveau : le niveau de maîtrise de l'UC auquel l'exercice est rattaché
- Seuil d'échec : un score en dessous duquel l'apprenant échoue dans le niveau
- Seuil de réussite : un score minimal pour réussir le niveau de maîtrise
- Temps estimé : le concepteur définit un temps de résolution de chaque exercice.

Pour savoir si l'apprenant a acquis les concepts d'une unité cognitive, cette dernière possède des niveaux de maîtrise auxquels sont rattachés des exercices. Donc un apprenant est tenu de faire tous les exercices d'un niveau pour le passer. La figure 7 suivante montre la relation qui existe entre unité cognitive, niveau de maîtrise et exercice.

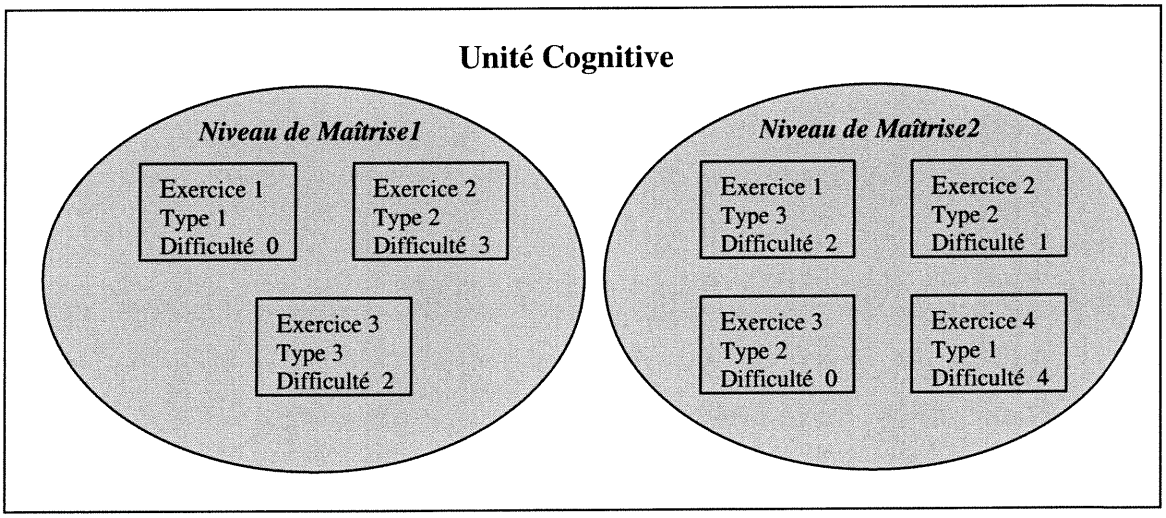


Figure 7 : Relation entre unité cognitive, niveau de maîtrise et exercice.

3.2.3 Les exercices spécifiques

Ce sont les exercices conçus ad hoc (sur mesure) pour enrichir le contenu du cours en question. En effet, pour un cours particulier, nous avons jugé que les exercices génériques (voir section ci-dessous) sont moins adaptés aux besoins ou répondront moins aux attentes des apprenants. C'est pourquoi des exercices spécifiques à ce cours s'imposent. Prenons un exemple d'un cours sur le SQL (Structured Query Language langage pour l'interrogation et la mise à jour des bases de données), un exercice permettant la saisie d'une requête SQL en ligne pour vérifier la syntaxe de l'apprenant et visualiser le résultat de la requête aiderait énormément l'apprenant à acquérir les commandes du langage.

3.2.4 Les exercices génériques

Contrairement à la catégorie d'exercices cités précédemment, les exercices génériques peuvent être appliqués à tous les cours, ils sont générés automatiquement sous format Html et représentent un standard dans le domaine de l'éducation. À partir des éléments clés textuels saisis par le concepteur, à savoir la question et les choix possibles, le système génère une page html standard présentable correspondant à cette exercice. Toute la mise en forme de l'exercice est faite à ce stade mais nous avons laissé le choix au concepteur d'éditer cette page pour pouvoir apporter d'éventuels changements.

Dans le cadre de ce projet, plusieurs types d'exercices génériques ont été mis en place. Lors de leur création, chaque type d'exercice possède sa propre interface, ses propres données et son propre traitement. Pour définir ces types d'exercices, nous nous sommes inspirés de tout se qui se passe chez les grands industriels du monde de l'informatique à savoir Microsoft, Oracle, IBM, etc. En outre, nous avons usé de nos expériences dans le domaine de l'éducation pour en réaliser d'autres. Nous les présentons ci-après avec à droite le symbole de chaque type d'exercice dans le prototype.

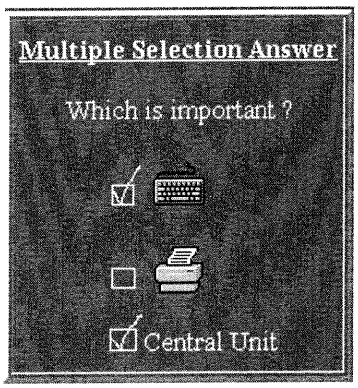
- ***Single Selection Answer (SSA)***

Ce sont des questions à choix multiples, mais une seule réponse parmi les choix est correcte. Après avoir défini la bonne réponse, pour chaque mauvaise réponse, on doit définir un groupe d'explications associé. Ceci permet à l'agent pédagogique de choisir quelle explication donnée en cas d'erreur faite par l'apprenant.

Une fois que le nombre de choix qui composent l'exercice est connu, on génère un fichier Html qui sera édité plus tard par le concepteur afin d'y inclure des rubriques de choix sous différents formats (images, sons, vidéos, etc).

- ***Multiple Selection Answer (MSA)***

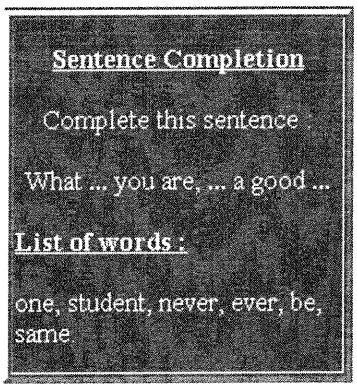
Ce sont des questions à choix multiples, mais plusieurs réponses parmi les choix peuvent être correctes. Après avoir défini la bonne réponse (un ou une combinaison de choix), pour chaque combinaison de mauvaises réponses, on doit définir un groupe d'explications associé. Ceci permet à l'agent pédagogique de choisir quelle explication donner en cas d'erreur faite par l'apprenant.



Ce type d'exercice est un peu plus délicat à gérer dans la mesure où nous devons raisonner en terme de combinaison de choix pour le groupe d'explications. Ceci complique énormément la tâche quand il s'agit de définir un groupe d'explications pour chaque combinaison possible. En effet, pour **n** choix possibles, le nombre de combinaison possible est $\mathbf{2^n - 1}$. Pour cela, nous avons établi des mécanismes pour définir un groupe d'explications pour tout un ensemble de combinaisons de choix.

- *Sentence Completion (SC)*

Ce sont des exercices qui permettent de compléter un texte où plusieurs paragraphes contiennent des mots vides. Pour chaque mot vide, une liste de choix de mots est fournie dans laquelle se trouve le mot juste. Le texte peut être une formule mathématique, une citation célèbre, une définition, etc.



Comme le nombre de combinaisons possibles croit rapidement en fonction du nombre de mots vides à remplir, nous avons donc défini un symbole générique (*) pour désigner n'importe quel mot choisi dans la liste. Ce type d'exercice est plus délicat du fait que la réponse donnée par l'apprenant peut être partiellement correcte, donc il a fallu décomposer l'exercice en plusieurs étapes pour ne pas pénaliser les apprenants et suivre leurs réponses étape par étape.

La figure ci-dessous montre l'exemple d'un exercice de ce type concernant la syntaxe d'une requête SQL à saisir. Comme, nous pouvons le constater, les mots cachés des phrases se trouvent dans *'combo box'* , l'utilisateur doit choisir le mot juste parmi la liste des mots proposés.

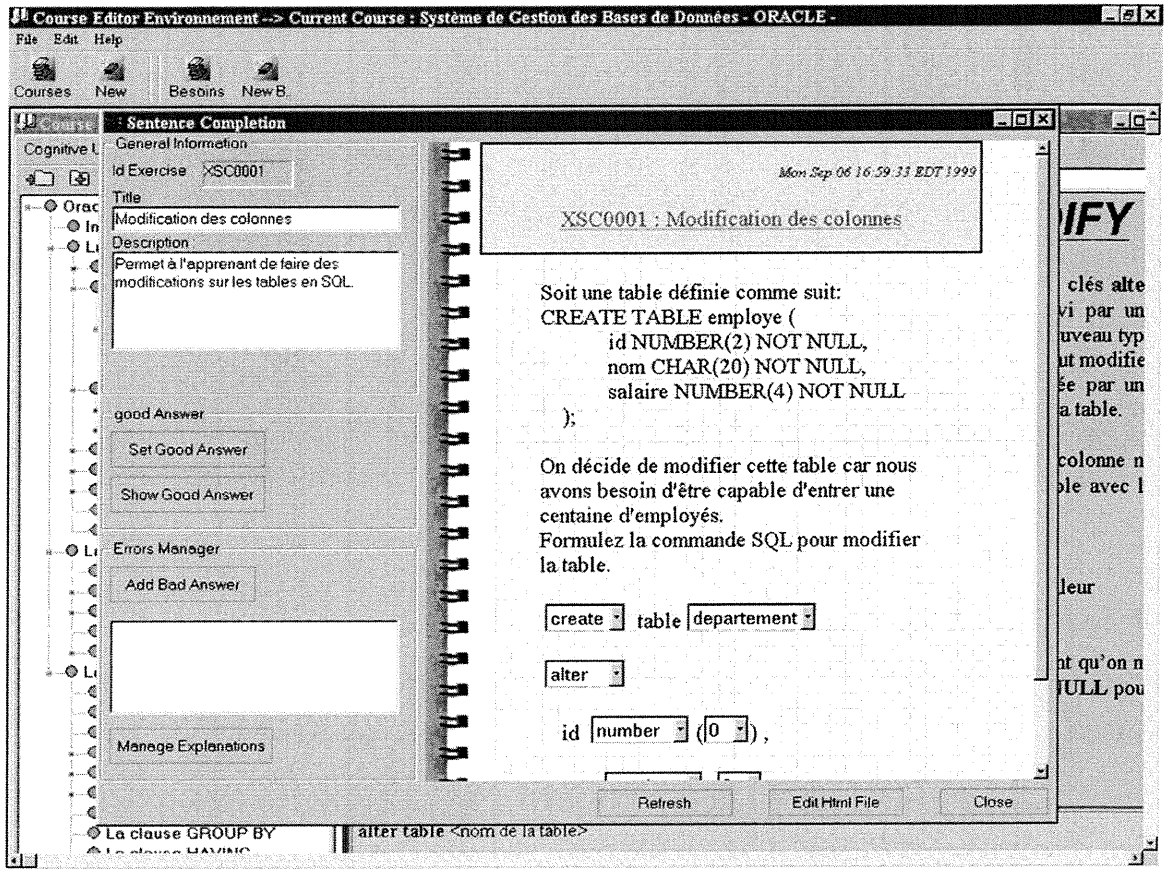
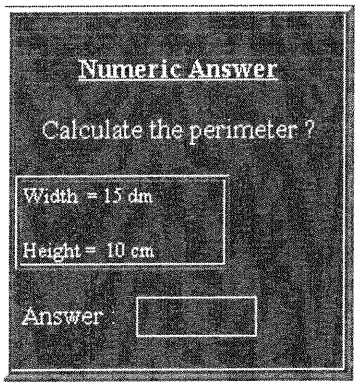


Figure 8 : Schéma d'un exercice de type Sentence Completion.

Le concepteur de cours doit saisir les caractéristiques de l'exercice, puis définir les mots à cacher dans le texte. Une fois terminé, l'exercice est généré dans une page HTML et présenté dans le browser intégré à l'application comme l'indique la figure 8. Ensuite, le concepteur définit la bonne solution et les explications pour chaque cas d'erreur.

- *Numéric Answer (NA)*

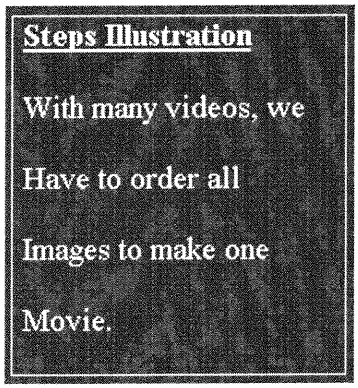
Ce sont des exercices où la réponse est soit un nombre unique (ex. 24.05) ou un intervalle fermé de nombres (ex. [11.5,25.4]). Ce genre d'exercices sert spécialement dans les cours qui demandent beaucoup de calcul de précision comme les mathématiques, physiques…



Les cas d'erreur dans ce type d'exercices sont infinis. Au départ, le concepteur décide lui même des erreurs jugées potentiellement fréquentes chez les apprenants pour leurs associer des groupes d'explications. Ces cas d'erreurs peuvent être des nombres exacts ou des intervalles de nombres. Mais le MGE est doté d'une fonctionnalité qui permet de détecter les cas d'erreurs fréquentes chez les apprenants et qui n'ont pas été prévues au départ par le concepteur. En effet, dans le cas où un cas d'erreur, sans aucun groupe d'explications associé, arrive plus souvent chez les apprenants, le MGE en informe le concepteur.

- *Steps illustration (SI)*

Il s'agit de présenter à l'apprenant une séquence vidéo sectionnée en plusieurs vidéos séparées et en désordre. L'apprenant doit classer ces clips dans l'ordre afin de retrouver le film en entier. C'est le type d'exercice qui a demandé le plus d'effort dans sa réalisation à cause de la complexité du problème.



Au départ, nous disposons d'un film vidéo complet sous format AVI. Ensuite il a fallu réaliser un séquenceur automatique pour sectionner les vidéos à la seconde près. Chaque nouvelle séquence générée doit être sauvegardée dans un fichier séparé. Enfin, nous déterminons l'ordre selon lequel les vidéos seront présentées à l'apprenant. Une fois, toutes les séquences définies, leur manipulation est extrêmement conviviale, grâce à la technique de glisser-déplacer « drag and drop », aussi bien pour le concepteur que l'apprenant. Pour le moment, ce type d'exercice représente un seul inconvénient qui est le temps de transmission de ces vidéos en temps réel du côté de l'apprenant pendant la session de formation pour résoudre un exercice de ce type. En effet, les fichiers prennent beaucoup de place.

La figure 9 ci-dessous montre un exemple d'exercice, de ce type, concernant les étapes à suivre pour effectuer une descente lors d'un vol d'avion. Il s'agit d'images vidéos prises du logiciel « Flight Simulator » pour l'apprentissage du pilotage d'avion. Toutes les vidéos présentes représentent une seule séquence vidéo sectionnée en plusieurs séquences. Chaque séquence représente un certain nombre de secondes de la vidéo originelle est sectionnée grâce au séquenceur cité précédemment et représenté dans la figure 10. Dans le même exercice, nous pouvons avoir plusieurs séquences issues de plusieurs fichiers vidéos différents.

Ce type d'exercice est très utile dans le cadre de cours qui demandent plus de manipulations. En effet, pour l'enseignement de la mécanique des pièces à monter ou démonter, il suffit de filmer numériquement la procédure pour en faire une série d'exercices de ce type.

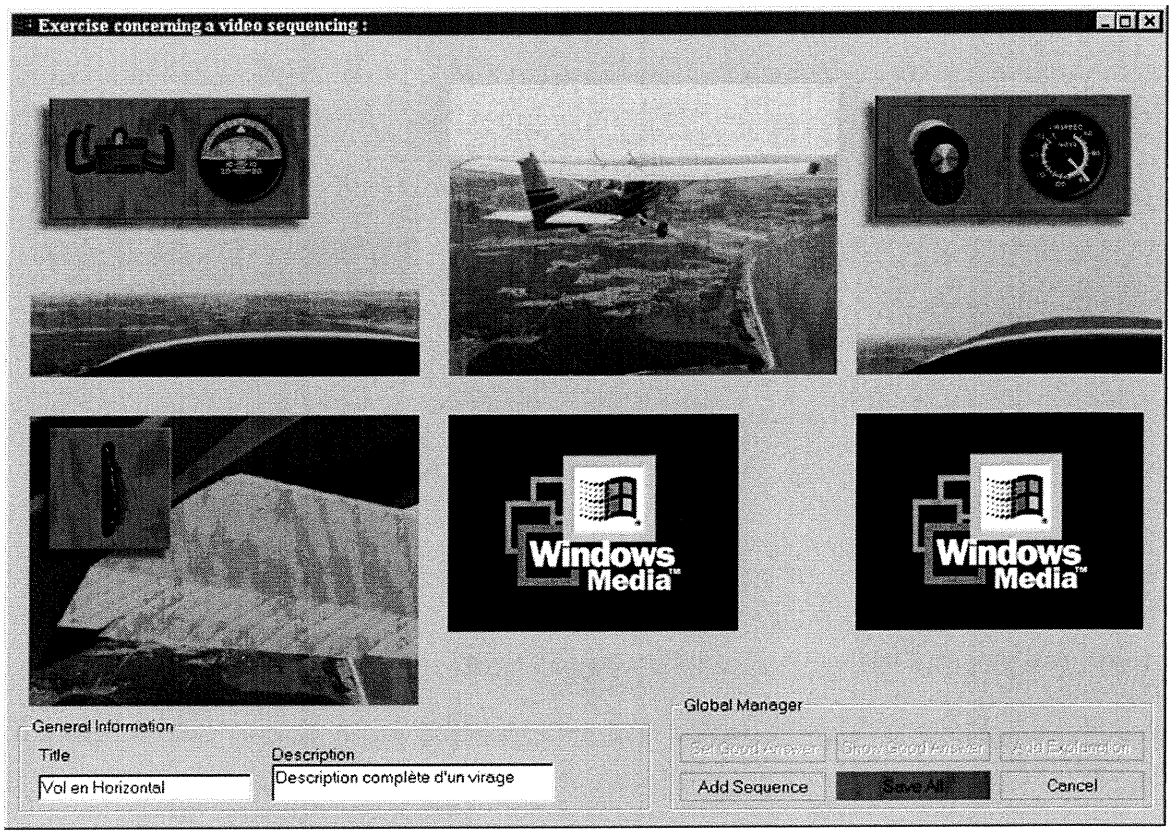


Figure 9 : Schéma d'un exercice de type Steps Illustration.

Comme nous pouvons le constater, le concepteur doit inclure au maximum six séquences vidéos par exercice. Cette limitation est nécessaire, d'une part, pour préserver l'esthétique globale de l'interface et d'autre part, parce que la taille des vidéos est tellement grande qu'il nous serait impossible de les transmettre chez l'apprenant en un temps raisonnable.

Ensuite le concepteur indique la bonne séquence par des glisser-déplacer « drag and drop », il indique aussi les mauvaises séquences possibles pour leur associer des groupes d'explications.

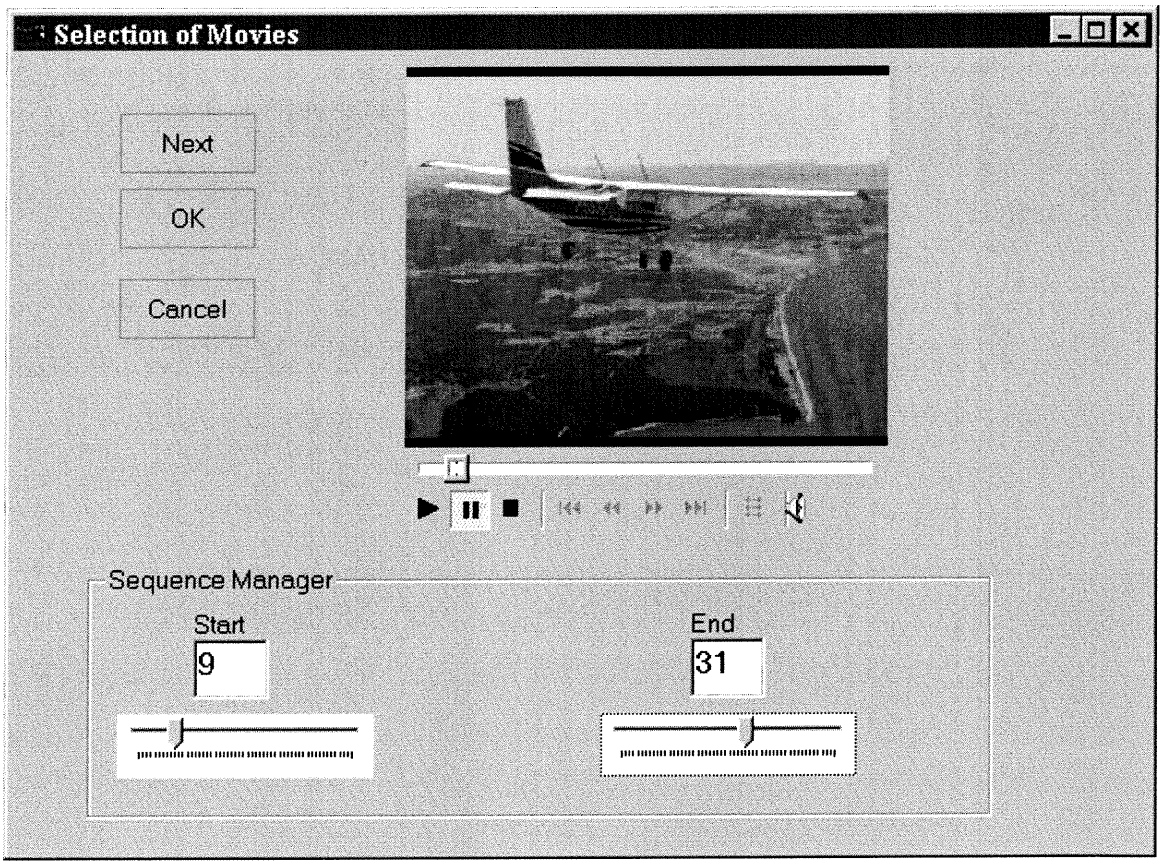
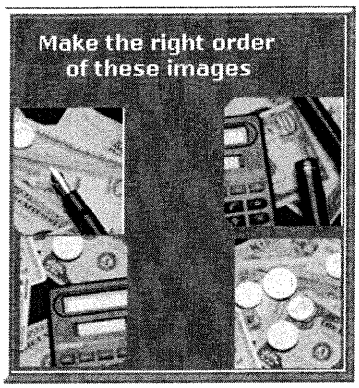


Figure 10 : Séquenceur permettant de partitionner une vidéo en plusieurs sections.

Une fois la vidéo choisie, le concepteur commence par le visionnement et ensuite sélectionne une séquence bien précise en indiquant la seconde de départ et celle de la fin. Pour la réalisation de ce séquenceur, nous avons été obligé de mettre en place une API (Application Programming Interface) qui nous a permis d'accéder aux fonctionnalités de base de l'outil multimédia de Microsoft qui nous permet de jouer la vidéo. Cette API nous a été d'une grande utilité dans la mesure où nous l'avons utilisée pour l'édition de ce type d'exercices et pour la présentation de ces mêmes exercices chez l'apprenant. Enfin, cette même API pourrait être utilisée dans le cadre d'un exercice de type « Steps Illustration » dans lequel les séquences représentent des morceaux de fichiers son (audio).

3.2.4.6 Images to order (ITO)



C'est un exercice qui consiste à classifier un ensemble d'images selon le bon ordre. C'est le même type que précédemment sauf que cette fois ci, il s'agit d'images au lieu de vidéos. Ces images peuvent être des écrans d'un logiciel permettant d'illustrer les étapes à suivre pour réaliser une manipulation.

Prenons comme exemple, un cours sur Powerpoint de Microsoft, le concepteur veut savoir si l'apprenant maîtrise les étapes à faire pour insérer un pied de page dans une diapositive. Il suffit de prendre les écrans de chaque étape et de les mettre en désordre. La technique du drag and drop a aussi été utilisée dans ce type d'exercice.

Tous les exercices cités précédemment ont chacun une bonne réponse et un ensemble de mauvaises réponses ou cas d'erreurs. Lors de la réalisation d'exercice, le concepteur peut associer un groupe d'explications à chaque cas d'erreur. Les explications vont de la plus ou moins détaillée à la plus raffinée et élaborée. Une explication peut être textuelle ou HTML. Cette dernière s'utilise lorsqu'on veut associer à une mauvaise réponse une explication qui est en fait une référence à une page Web se trouvant dans un site Internet distant. Toutes ces explications vont servir à l'agent explicatif pour venir en aide à un apprenant en difficulté.

3.2.5 Le module Gestionnaire d'exercices orienté apprenant

Cette partie concerne la présentation de l'exercice à l'apprenant pour le résoudre. Selon le type d'exercice et le modèle de l'apprenant, le MGE détermine l'interface correspondante. En effet, le MGE interagit avec l'apprenant par l'intermédiaire d'une interface personnalisée relative au type d'exercice présentement en traitement, le modèle de l'apprenant et l'état de l'exercice.

- *Les types d'exercices* sont tous ceux énumérés précédemment, chacun possède sa propre interface, son propre système d'évaluation et son propre processus de correction.

- *Le modèle de l'apprenant*, étant donné que chaque exercice appartient à un niveau de maîtrise d'une unité cognitive donnée, si l'apprenant est supposé être au niveau de maîtrise n dans une unité cognitive U1, le MGE va lui proposer l'exercice le plus difficile dans le niveau n, avec l'explication la moins détaillée, afin de valider ses connaissances.

- *L'état de l'exercice,* dans notre projet, l'évolution d'un exercice exécuté par un apprenant est définie à partir de quatre états :

    1. Exercice Complété Vérifié Correct
    2. Exercice Complété Vérifié Partiellement Correct
    3. Exercice Complété Vérifié Incorrect
    4. Exercice Complété Non Vérifié

Selon l'état de l'exercice et le type d'exercice, un ensemble de choix est fourni à l'apprenant afin de l'aider pédagogiquement à trouver la solution de l'exercice. La figure 11 suivante résume les choix offerts :

ÉTATS	SSA	MSA	Others
Complété			
Vérifié			
• Correct	• Next step	• Next step	• Next step
• Partiellement Correct		• Help • Show Errors • Solution	• Help • Show Errors • Correct an Error • Solution
• Incorrect	• Help • Solution	• Help • Show an Error • Solution	• Help • Show an Error • Correct an Error • Solution
Non Vérifié			
	• Verify my answer • Help • Solution	• Verify my answer • Help • Solution	• Verify my answer • Help • Solution

Figure 11 : Choix contextuel fourni à l'apprenant durant l'exercice

3.2.6 Système de notation adéquat

Pour tout nouvel exercice présenté à l'apprenant, la note 100 est attribuée automatiquement. Si l'apprenant donne la bonne réponse dès le premier coup, il obtient 100 % de la note. Mais, au fur et à mesure qu'il fait des erreurs le MGE retranche des points selon le type de l'exercice courant, le nombre de choix possible dans l'exercice et le nombre d'explications fourni dans chaque cas d'erreur. Nous avons pris en considération ce dernier critère car nous avons jugé que si l'apprenant n'est pas très aidé dans le processus de résolution de l'exercice, il ne faut pas le pénaliser davantage.

- **SSA** :

Entier CalculSSAScore(Entier ScoreActuel)

Début

Si bonne réponse **alors** return ScoreActuel

Sinon return ScoreActuel – (100 / Nombre de réponses - 1);

Fin

Où ScoreActuel est le *score* actuel de l'apprenant et *Nombre de réponses* est le nombre de réponses possibles existantes dans l'exercice courant. Au départ ScoreActuel est initialisé à 100%.

- **MSA, SC, SI, ITO**

Dans ce type d'exercice, comme le nombre de combinaisons possibles peut être très grand, nous avons déterminé un mode de calcul qui tient compte des explications fournies à l'apprenant pour chaque erreur commise. Quand un cas d'erreur n'a pas de groupe d'explications associé, le système retranche 5% (nombre donné dans le but de ne pas pénaliser l'apprenant) du score actuel. La procédure qui permet de calculer le score dans ce type d'exercice est la suivante :

Entier CalculMSAScore(Entier ScoreActuel)

Début

Si bonne réponse **alors** return ScoreActuel

Sinon

Début

Nb_Erreur = Lire_Nb_Erreur() pour lire le nombre de mauvaises réponses qui ont un groupe d'explications associé pour l'exercice courant.

Nb_Expl = Lire_Nb_Explication() pour lire le nombre d'explications associé au cas d'erreur couramment traité.

ScoreActuel = ScoreActuel - ((100 / Nb_Erreur) / Nb_Expl);

Fin

Fin

- **NA**

Comme dans ce type d'exercices, la réponse peut être un nombre ou un intervalle de nombres, l'évaluation exige une certaine précision par rapport au résultat final. En effet, le concepteur doit préciser la pénalité en fonction de la distance qui sépare la réponse de l'apprenant et le résultat exact. Prenons comme exemple l'exercice ci-dessous :

Calculer la surface (en mètre) d'un rectangle ayant comme paramètres :

Largeur = 23.5 cm et Longueur = 26.3 cm. La réponse est S = (23.5 * 26.3) = 6.1805 m

Pour cet exercice la réponse est une valeur exacte, donc les réponses envisageables sont décrites dans la figure 12 suivante :

Réponse de l'apprenant	Évaluation faite
618.05 cm	Réponse exacte mais pas de conversion faite, donc pénalité de 10 %
Réponse > 620 ou Réponse < 615	Complètement faux, donc est 0
615 > Réponse < 620	Erreur dans le calcul, donc pénalité de 20 %
6.1805 m	Juste, 100 % de la note
99.6 cm	Confusion entre Surface et Périmètre d'un rectangle, donc pénalité de 50 %

Figure 12 : Évaluations possibles lors de la résolution d'un exercice de type Numeric Answer

3.3 Conclusion

Dans ce chapitre, nous avons abordé la conception et la réalisation de l'agent gestionnaire d'exercices, qui permet à la fois aux concepteurs d'éditer une série variée d'exercice, et aux apprenants de résoudre intelligemment et pédagogiquement les exercices pour mieux assimiler les concepts d'un cours particulier. Nous avons aussi montré que dans les STI, les exercices sont d'une grande importance car ils permettent aux concepteurs d'une part, de savoir si les connaissances ont été acquises ou pas, d'autre part, de remédier aux lacunes et points non détaillés dans le contenu du cours. Pour cela un système d'évaluation rigoureux s'impose.

Nous avons également mis l'accent sur la personnalisation de l'interface graphique, car c'est la clé du succès d'une application informatique. Dans le développement des STI, le concepteur et l'apprenant vont passer un temps considérable devant la machine, donc un grand effort a été consacré à la convivialité et à la souplesse de l'interface graphique.

Dans le prochain chapitre, nous montrerons l'ouverture sur le monde extérieur pour l'échange et l'interopérabilité de nos bases de données générées jusqu'à présent.

CHAPITRE 4
ARCHITECTURE MULTI-AGENTS POUR L'INTEROPÉRABILITÉ

4.1 Stockage et interopérabilité en XML

4.1.1 Objectif à atteindre

Les STI sont développés à travers le monde dans différents centres de recherches, organismes de formations, unités de production de logiciels, etc. Chacun de ces STI possède sa propre base de cours et d'exercices. Le but est de pouvoir créer un réseau commun entre ces STI pour pouvoir échanger les données, éviter la redondance dans la conception d'exercices portant sur l'enseignement d'un concept précis dans un cours précis. Ces STI sont développés dans différentes plates-formes et parlant des langages différents. Donc l'objectif principal est de créer une sorte de bibliothèque distribuée d'exercices qui permet à différents usagers de partager et réutiliser le contenu de ces exercices. Avec l'aide de l'éditeur d'exercices cité précédemment, nous avons pu mettre en place une série d'exercices de différents types concernant plusieurs cours. Il serait souhaitable de faire profiter d'autres concepteurs de cette base de données. Pour ce faire, il va falloir trouver un langage standard qui puisse répondre exactement à nos attentes. Cette solution miracle s'appelle XML (eXtensible Markup Language), un standard

reconnu pour le stockage et l'interopérabilité. Voyons ce que ce langage nous apporte comme richesse et solution dans le monde d'Internet d'aujourd'hui.

4.1.2 Tour d'horizon global sur le standard XML

XML (eXtensible Markup Language) est un langage de description de structures de documents recommandé par l'organisme de normalisation W3C [W3C 98]. Une structure est, par définition, un ensemble de sous-structures appelées éléments. Un document peut être un livre, avec comme éléments un titre, des chapitres, sections, paragraphes, figures, légendes, etc. Un document peut aussi être un message électronique (e-mail), ses éléments étant alors une en-tête (émetteur, destinataire, date...), un corps de texte et des pièces jointes. Il peut, enfin, s'agir d'un message entre deux applications, par exemple entre une application commerciale et une application de comptabilité. Rédigé à l'aide du langage informatique XML, un document est comme un code source de programme : il doit respecter les règles syntaxiques du langage.

4.1.3 Document XML et DTD

Sur le plan conceptuel, un document XML a deux parties :

Le document proprement dit, obligatoirement présent, (DTD). Cette description contient les règles syntaxiques que doit respecter le document. Tout se passe comme pour un programme que l'on écrit dans un langage de programmation : le langage a une grammaire, qui décrit les structures syntaxiques permises pour un programme et les mots réservés que l'on emploiera. Le DTD contient aussi, dans cette grammaire, la liste des éléments (concepts) du document ; chaque concept est représenté par un mot réservé, comme le mot <titre d'exercice>. L'ensemble de ces mots constitue le vocabulaire autorisé ou, en langage savant « l'espace de noms ». Comme HTML, XML dérive de SGML (Structured Generalized Markup Language); SGML aussi utilise des DTD.

Accompagné de son DTD, un document XML est comme un programme : il doit se conformer strictement aux règles de sa grammaire. On dit qu'il est « valide » lorsqu'on peut en vérifier la syntaxe avec cette grammaire. Sans DTD, un document peut quand

même être « bien formé » : sa structure est alors réputée décrite par elle-même et peut se contenter de respecter les règles générales du langage XML.

4.1.4 Comparaison entre XML et HTML

Différence entre SGML et XML : SGML a été conçu pour le stockage de documents dans de grandes bibliothèques centralisées, alors que XML a été fait pour une utilisation dans des environnements répartis, aussi bien pour le *stockage de documents* que pour *l'interopérabilité par échange de données.*

Le langage XML permet de définir et de nommer des éléments, avec leurs sous-éléments. Un document a forcément une structure arborescente, accessible en entier à partir de son élément de plus haut niveau. Un arbre décrit la composition de chaque élément (ses sous-structures) ; un élément peut être un hyperlien pointant vers n'importe quel objet informatique n'importe où, y compris vers un élément de document XML (le même document ou un autre).

- HTML définit le format de mise en page (affichage ou impression) d'un document, alors que XML en définit la structure, le contenu, la sémantique, indépendamment de la mise en page. Les documents XML ont un DTD, les documents HTML n'en ont pas.

- La grammaire de HTML est fixe, définie par le standard, avec ses mots réservés et ses structures entre balises (tags). XML, au contraire, permet de définir n'importe quelle structure dans la mesure où elle est arborescente, avec notamment les balises que l'on veut.

- Les documents HTML ont une structure séquentielle avec une en-tête (header) et un corps (body). Les documents XML, eux, sont des hiérarchies.

- En HTML, un hyperlien désigne un objet externe pour incorporation à la page en cours, remplacement de cette page ou affichage dans une fenêtre séparée. XML est un des standards de stockage et de partage de documents

De ce qui précède on peut tirer une première conclusion : XML est bien adapté au stockage de documents quelconques : ouvrages techniques illustrés, emails, lignes d'une table en Bases de données, etc. Les données stockées en XML sont indépendantes des systèmes d'exploitation de stockage ou d'accès, des langages de programmation et des formats de mise en page. *XML est donc un standard universel de stockage.*

4.1.5 XML : une solution d'interopérabilité intéressante

XML est aussi une solution d'interopérabilité intéressante. Le langage peut être utilisé pour créer des messages auto-descripteurs destinés aux échanges entre des agents provenant d'applications aussi différentes que la conception de cours et la présentation des exercices. De tels messages sont indépendants des systèmes d'exploitation, des langages de programmation et des formats d'affichage. Des applications qui utilisent ces messages comprennent la structure et la sémantique des données qu'elles reçoivent, ce qui est impossible avec HTML.

XML permet à un agent consommateur d'échanger des données avec beaucoup de sources de données ou d'agents producteurs hétérogènes, sans connaître d'avance les structures, les types et les divers formats des données qu'il reçoit. Cette transparence est intéressante, dans notre cas, pour la coopération. Elle est inconcevable dans une architecture client serveur traditionnelle.

4.2 Notre approche de Génération d'exercices en XML

Tous les exercices des cours sont sauvegardés dans une base de données relationnelle. Pour pouvoir échanger ces données, nous allons les extraire de la base et générer un ou plusieurs XML correspondants. Nous verrons plus loin l'agent responsable de l'acheminement de ces fichiers chez le solliciteur.

Examinons un ensemble d'exercices généré sous le format XML, avec une DTD définie :

```
<?xml version="1.0" ?>
<!DOCTYPE Base SYSTEM "Exercise.dtd"><Base>
<Exercise>
  <IDExercise>XSA0002</IDExercise>
  <Title>Calcul de la taille d'une jointure</Title>
  <Description>Comment l'apprenant peut determiner la taille d'une
table resultat d'une jointure.</Description>
  <Content>www.iro.umontreal.ca/~boukhere/XML/XSA0002.htm</Content>
  <FailThreshold>25</FailThreshold>
  <SuccessThreshold>85</SuccessThreshold>
  <Time>15</Time>
</Exercise>
<Exercise>
  <IDExercise>XMA0001</IDExercise>
  <Title>Les commandes invalides</Title>
  <Description>Permettre à l'apprenant de vérifier les commandes
invalides lors de la création.</Description>
  <Content>www.iro.umontreal.ca/~boukhere/XML/XMA0001.htm</Content>
  <FailThreshold>30</FailThreshold>
  <SuccessThreshold>80</SuccessThreshold>
  <Time>10</Time>
</Exercise>
<Exercise>
  <IDExercise>ORA0001</IDExercise>
  <Title>Oracle Exercie (Order Distinct)</Title>
  <Description>Lancer la question 1 dans Rank</Description>
  <Content>www.iro.umontreal.ca/~boukhere/XML/ORA0001.htm</Content>
  <FailThreshold>10</FailThreshold>
  <SuccessThreshold>90</SuccessThreshold>
  <Time>15</Time>
</Exercise>
<Exercise>
  <IDExercise>XMA0002</IDExercise>
  <Title>Définition des Vues</Title>
  <Description>Les possibilités offertes par les Vues en
SQL.</Description>
  <Content>www.iro.umontreal.ca/~boukhere/XML/XMA0002.htm</Content>
  <FailThreshold>25</FailThreshold>
  <SuccessThreshold>85</SuccessThreshold>
  <Time>25</Time>
</Exercise>
<Exercise>
  <IDExercise>XSC0001</IDExercise>
  <Title>Modification des colonnes</Title>
  <Description>Permet à l'apprenant de faire des modifications sur les
tables en SQL.</Description>
  <Content>www.iro.umontreal.ca/~boukhere/XML/XSC0001.htm</Content>
  <FailThreshold>20</FailThreshold>
  <SuccessThreshold>65</SuccessThreshold>
  <Time>20</Time>
</Exercise>
</Base>
```

Figure 13 : Fichier XML généré à partir de la base de données des exercices.

Le document type definition correspondant au fichier XML précédemment mentionné est la suivante :

```
<!ELEMENT Base (Exercise)* >
<!ELEMENT IDExercise (#PCDATA) >
<!ELEMENT Title (#PCDATA) >
<!ELEMENT Description (#PCDATA) >
<!ELEMENT Content (#PCDATA) >
<!ELEMENT FailThreshold (#PCDATA) >
<!ELEMENT SuccessThreshold (#PCDATA) >
<!ELEMENT Time (#PCDATA) >
<!ELEMENT Exercise (IDExercise, Title, Description,Content,
FailThreshold, SuccessThreshold, Time) >
```

Figure 14 : Un fichier DTD correspondant au fichier XML de la figure 13.

4.2.1 Visualisation des exercices générés

Au premier regard sur les deux fichiers précédemment mentionnés (figures 13 et 14) ne constituent pas un moyen convivial de visualiser les données. L'utilisateur doit avoir un moyen esthétique et présentable pour visualiser ces informations. La solution est le développement d'un « viewer » XML qui permet de rendre les informations contenues dans un fichier XML lisibles et exploitables. Pour cela, il faut utiliser un « parser XML » qui, en utilisant une API (Application Programming Interface), permet d'explorer le contenu des fichiers XML d'une manière standard. Dans le standard mis en place par le consortium W3C, il existe deux manières distinctes pour « parser » un document XML : SAX (Simple API XML) et DOM (Document Object Model).

Après génération, le concepteur peut visualiser l'ensemble de ses exercices grâce à un utilitaire spécialisé « viewer XML » représenté à la figure 15. Ce qui est intéressant dans cet utilitaire, c'est qu'il permet la visualisation de tous les éléments qui composent un document XML. On y trouve les parties suivantes :

- *Haut à gauche* : La structure du document sous la forme d'un arbre (DOM), les nœuds et les fils des nœuds et ainsi de suite.

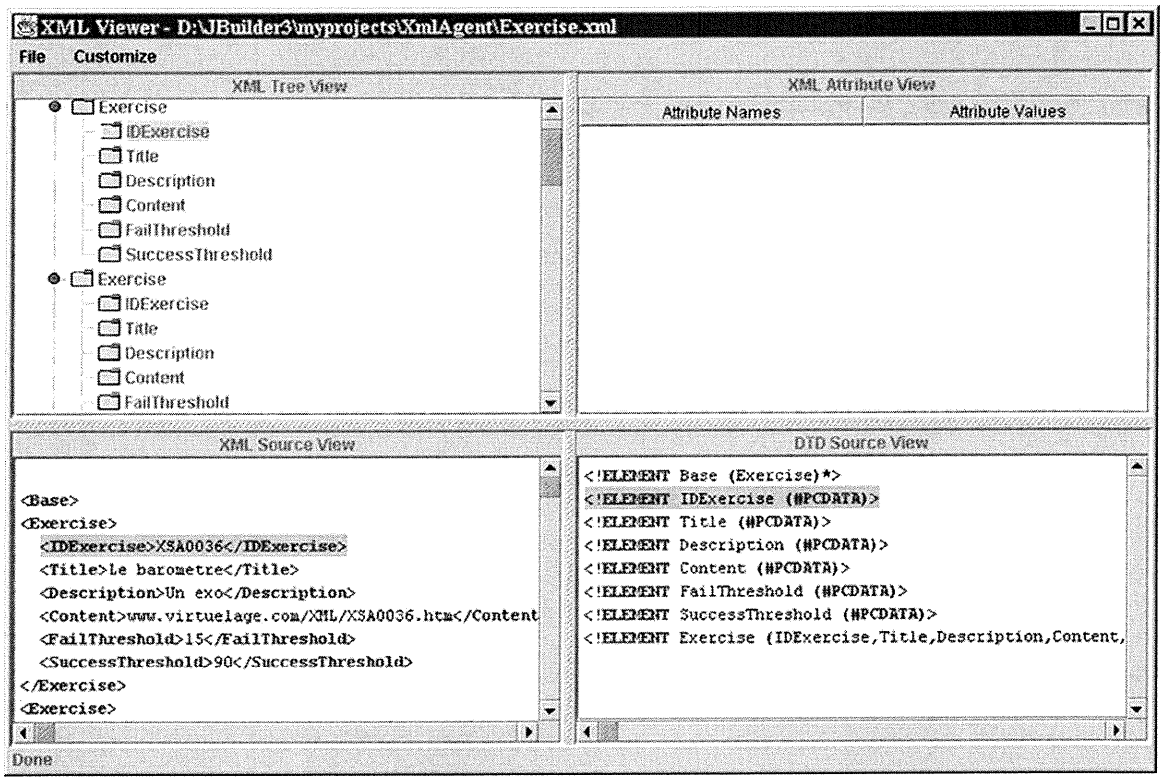


Figure 15 : Un viewer de fichiers XML.

- *Haut à droite* : Les attributs définis éventuellement dans chaque nœud. Ici nous n'avons pas d'attribut défini.

- *Bas à gauche* : Le document XML lui même, si l'utilisateur est familier avec le standard XML, il pourra visualiser le contenu.

- *Bas à droite* : Le document type definition (DTD) qui accompagne le document XML présentement affiché.

Nous tenons à préciser que l'utilisateur pourra afficher ou cacher chaque partie du viewer selon ses besoins.

4.3 Architecture multi-agents

4.3.1 Théorie des agents

La question de savoir ce qui constitue réellement un agent et comment il diffère d'un programme normal, a été débattue pendant plusieurs années. Sans perte de généralité, nous pouvons définir les agents comme étant «des entités informatiques qui effectuent une série d'opérations au nom d'un usager ou d'un autre programme et qui, pour cela, utilisent soit des connaissances, des représentations, des buts ou des désirs de l'usager. » [Gilbert et al., 95].

Plusieurs types d'agents ont été définis dans la littérature. De manière générale, la classification des agents est faite en fonction de la nature de leur environnement. Cette sous-section a pour but de présenter quelques catégories reconnues dans le domaine de la recherche des agents.

- *Agents sociaux*

Cette classe représente tous les agents qui sont aptes à communiquer avec d'autres agents. Lorsque plusieurs agents se rassemblent afin d'interagir ensemble, nous parlons alors d'une société d'agents. La performance d'une communauté d'agents peut être évaluée par sa cohérence et par la coordination entre les divers agents. Il existe des théories de coordination à utiliser lorsque nous nous intéressons à la coordination d'agents. En voici deux : la théorie de jeux et la théorie économique. Le travail de Joakim Eriksson et celui de Nicolas Finne portant sur les marchés électroniques [Eriksonn et al., 1997] sont des exemples d'un travail basé sur la coordination d'agents. Des modèles d'information et d'interaction permettant aux agents de localiser les participants de marchés pertinents, d'échanger leur intérêt, et de négocier des transactions ont été définies dans cet ouvrage. Nos agents se caractérisent par la sociabilité puisqu'ils collaborent entre eux pour mieux servir l'apprenant.

- *Agents mobiles*

Les agents mobiles ont l'habilité de se déplacer entre différents lieux situés sur différents ordinateurs. En termes plus formels, ces agents sont définis comme des objets ayant un comportement, un état et une location. L'exemple d'agent mobile est celui de l'agent Sulla qui est un agent de recherche et de filtrage sur le Web [Eichmann et al., 1994]. Avec l'explosion de l'information disponible sur le réseau Internet, la recherche d'informations précises équivaut généralement à la lecture de nombreux documents ayant peu d'informations pertinentes. Le filtrage de l'information non pertinente peut être un processus long et frustrant. L'agent Sulla permet de visiter plusieurs sites, de chercher l'information disponible, et de construire un index de liens pointant sur des morceaux d'informations satisfaisant les critères de recherche de l'usager. Nos agents sont mobiles puisqu'ils agissent sur les machines locales des différents apprenants utilisant le système.

- *Agents intelligents*

Les agents intelligents sont des agents qui utilisent une représentation symbolique. Cette représentation peut porter sur leur environnement, sur un usager ou sur tout autre aspect. Ces connaissances sont enregistrées dans une base de connaissances leur permettant d'effectuer leurs propres réflexions sur une situation précise. Une classe d'agent encore plus complexe est celle des agents intelligents intégrant en plus d'un système d'inférence, un système d'amélioration de leur base de connaissances. Ces agents ont la capacité d'apprendre. Un exemple de ce type d'agent intelligent est celui de l'agent Instructo-Soar [Huffman, 1994]. Cet agent est un robot qui apprend à manipuler des blocs dans un environnement virtuel. Ce robot réalise des actions concrètes par l'entremise d'un dialogue avec son instructeur. Lorsque celui-ci est incapable d'effectuer une instruction, le robot demande des précisions sous forme d'instructions à son instructeur. C'est à partir de ces instructions que le robot enrichit sa base de connaissances et que l'on peut parler d'apprentissage.

4.3.2 Introduction à notre approche de système multi-agents

La problématique est de concevoir un système qui permet de mettre à la disposition des concepteurs tous les exercices développés dans les STI reconnus à travers le monde. Parmi les caractéristiques que doit présenter le système nous pouvons citer :

- prendre en charge le maximum d'exercices différents,
- faciliter l'extension du système pour l'ajout d'autres modules,
- permettre l'expansion du système afin de pouvoir l'installer sur plusieurs plates-formes,
- permettre l'interopérabilité avec d'autres systèmes,

Des décisions de conception ont été prises pour mettre au point l'architecture du système. En effet, nous avons fait en sorte que les caractéristiques citées précédemment soient assurées. Pour ce faire, nous avons :

- utilisé un système multi-agents permettant de fournir plusieurs services,
- choisi le standard XML pour l'interopérabilité et le stockage de données.

L'architecture multi-agents est basée sur plusieurs entités, permettant une gestion distribuée. Cette distribution des responsabilités permet d'augmenter la fiabilité, la robustesse et la performance de tout le système. Elle permet aussi de réduire les coûts en termes de communications dans le réseau et de répartir les traitements.

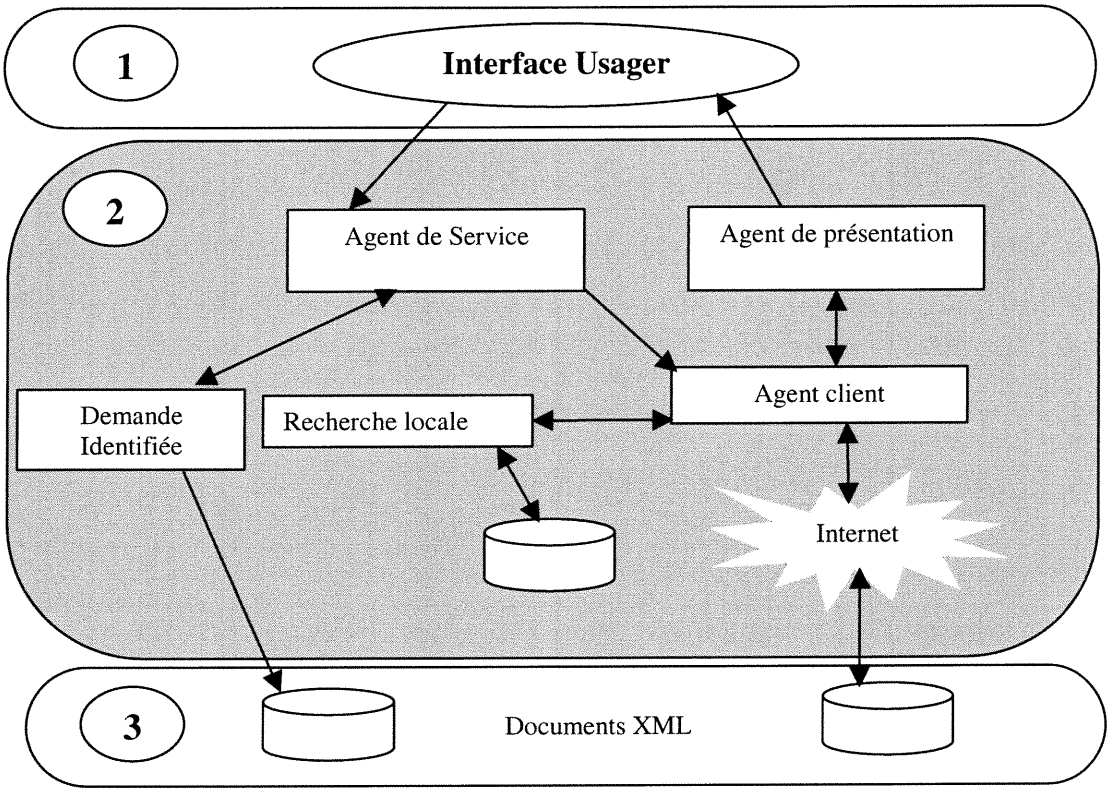


Figure 16 : Architecture globale du système. Trois parties importantes, l'interface usager (1), le système multi-agents (2) et le système de stockage (3).

La figure 16 donne un aperçu de l'architecture globale du système. Nous pouvons y remarquer les différents types d'agents impliqués. Nous pouvons aussi distinguer la partie gauche qui traite les demandes provenant de l'extérieur et la partie droite qui s'occupe de l'envoi des demandes concernant un besoin précis de l'usager.

Une première décomposition de cette architecture nous amène aux trois composantes principales comme illustrés dans la figure 16 précédente :

- l'interface graphique du système,
- le système de traitement des demandes,
- le système de stockage.

Dans les sections suivantes, nous allons décrire les trois composantes dans le détail.

4.3.3 Le système de traitement des demandes

- *Aperçu général du système*

Du point de vue logique, le système est organisé en une architecture hiérarchique. La figure 17 montre les composantes principales. Nous y observons les agents suivants :

L'agent d'administration (AA) : la gestion et la coordination des agents qui composent le système doivent être assumées par un agent unique, le c*entralisateur*. C'est par lui que se font connaître les autres agents. Il dispose d'un certain nombre de services qui vont assurer le bon fonctionnement du système, tels que la recherche, la vérification et le suivi.

L'agent de service (SA) : C'est un agent qui travaille en étroite collaboration avec l'agent D'administration pour répondre aux requêtes provenant de l'extérieur. Il peut y avoir autant de d'agents que de requêtes en cours de traitement. Il permet également de vérifier l'identité de la requête, si elle est connue auprès du AA. Notre système fonctionne, d'une part comme client pour effectuer des recherches sur le réseau, d'autre part comme serveur pour répondre aux différentes demandes parvenues d'autres STI. C'est cette dernière fonctionnalité qui impose la conception de l'agent de service pour mettre à la disposition des STI partenaires les exercices conçus localement.

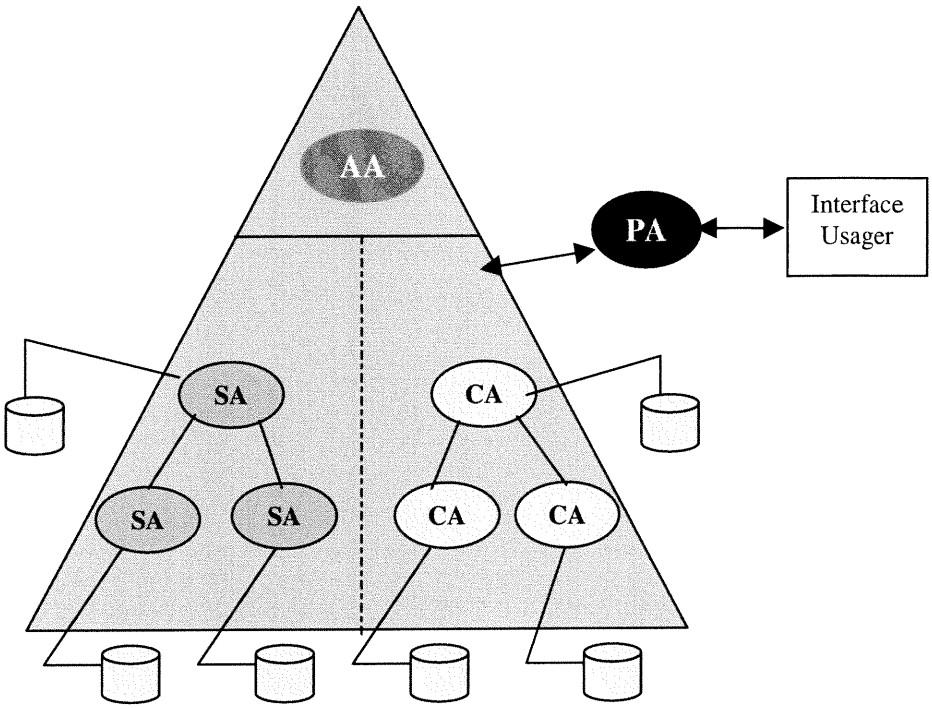


Figure 17 : Organisation pyramidale du système. Le AA est au niveau le plus élevé, puis à un niveau inférieur viennent SA et CA.

L'agent de présentation (PA) : cet agent permet de formater les fichiers reçus selon les exigences de l'usager. Par exemple, si l'usager veut voir le résultat de sa recherche sous format HTML ou envoyé directement à son adresse électronique, alors les services de ce type d'agents seront utilisés avant l'envoi de la requête.

L'agent client (CA) : Cet agent est chargé de répondre aux requêtes locales faites par les concepteurs. En effet, quand un concepteur local désire effectuer une recherche auprès des autres STI partenaires sur Internet, il fait appel aux services fournis par cet agent. Il s'agit d'une recherche sur un ou plusieurs exercices avec des caractéristiques précises et bien spécifiées.

- *Description des agents du système*

L'agent d'administration (AA)

Cet agent est unique et est le premier à être mis en service dans le système. Après sa création et son initialisation, il enregistre les informations nécessaires à son identification dans le réseau.

Plusieurs tâches sont assignées au AA. En effet, il a comme première tâche la gestion de l'intégrité du système, c'est à dire de vérifier que les autres agents sont effectivement actifs. Dans ce cas, toute anomalie ou dysfonctionnement détectée implique la mise hors service de l'agent en question et son remplacement par un autre.

La deuxième tâche qui incombe à l'agent d'administration concerne la répartition de la charge que les agents ont à absorber. Selon le traitement en cours, AA crée une instance de l'agent correspondant et lui affecte la tâche à réaliser. Prenons comme exemple un concepteur qui désire effectuer une recherche sur le réseau, le AA crée une instance de l'agent CA en lui fournissant toutes les caractéristiques des exercices de la requête. Le déroulement de chaque tâche est supervisé par AA. En effet, l'agent d'administration crée une pile de toutes les tâches qui sont en cours d'exécution. Une fois la tâche terminée, l'agent responsable rend compte de son état à l'agent d'administration. Dans le cas d'une procédure de recherche réalisé par un CA, les états possibles sont illustrés dans la figure 18 suivante.

État de la tâche	Interprétation
Terminée avec succès	L'agent client a pu terminer sa tâche selon les recommandations. En plus, il a pu trouver un STI qui a répondu favorablement aux exigences spécifiques aux exercices demandés.
Terminée sans résultat	L'agent client a pu terminer sa tâche selon les recommandations. Mais la requête reste insatisfaite par manque de résultats correspondant aux spécifications demandées.
En attente	MA attend toujours la fin de l'exécution de cette tâche. Un timing, bien sûr, est mis en marche pour éviter les attentes éternelles.
Erreur	Pour une raison quelconque, l'agent client n'a pu terminer sa tâche. Problème technique survenu en cours de route.
Pas de réponse	Les STI à qui est adressée la requête ne répond pas. Soit le serveur est en panne ou une mise à jour est en train de faire…

Figure 18 : États possibles suite à une procédure faite par un agent CA.

L'agent de service (SA)

Il y a autant d'agents de service que de requêtes à traiter à un instant donné. C'est un agent qui travaille sous tutelle de l'agent d'administration en répondant aux requêtes provenant d'autres STI. La première et importante tâche qui incombe au SA est de s'assurer que les requêtes provenant d'un autre site soient reconnues et fassent partie du réseau des STI. Ce travail se fait en collaboration avec l'agent d'administration car c'est lui qui tient la mise à jour de la base des STI partenaires avec leurs signatures de reconnaissance. Une fois cette phase de reconnaissance terminée, le SA entre en négociation avec l'agent client, comme le montre la figure 19 ci-dessous, qui a sollicité ses services pour satisfaire la requête. Cette négociation se fait par échange de messages faisant référence au contenu de la requête en cours de traitement.

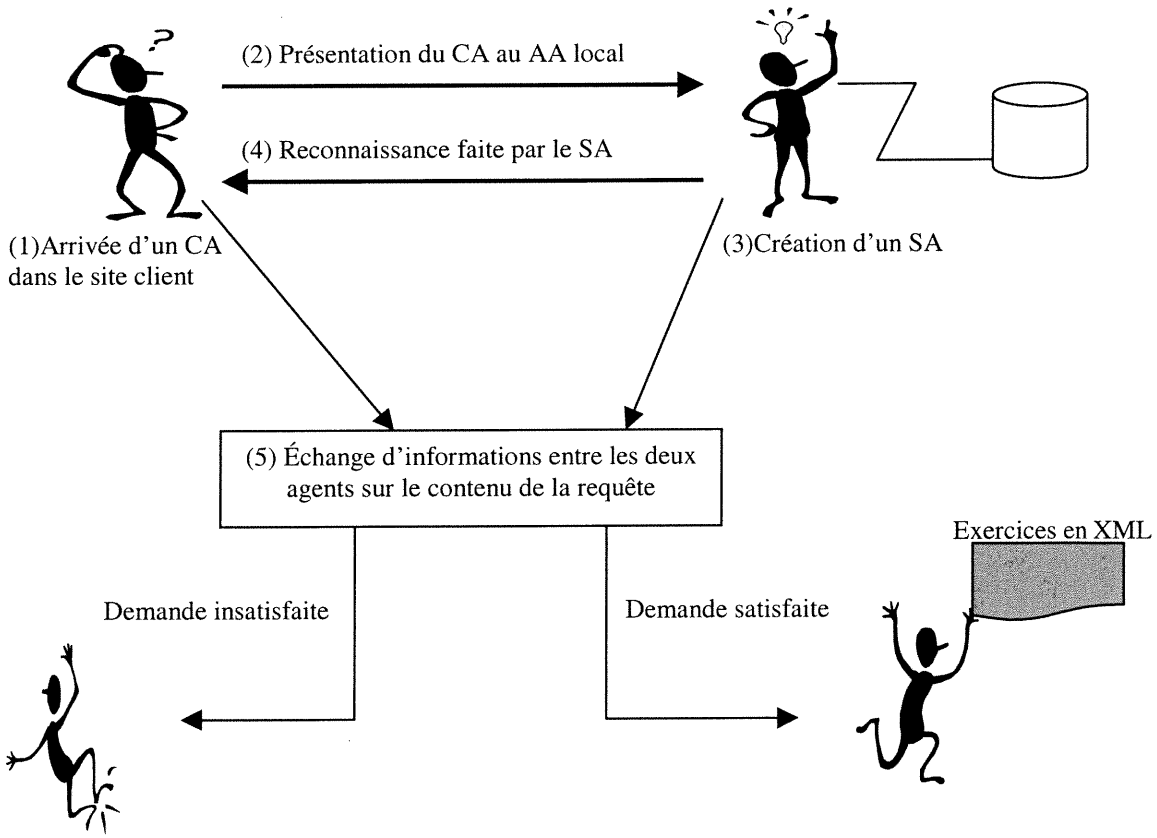


Figure 19 : Les étapes de traitement d'une demande.

L'agent de présentation (PA)

Cet agent offre le service de formatage et présentation de documents avant leur restitution aux usagers. Avant de faire une demande à l'agent CA, l'usager doit préciser de quelle manière il souhaite obtenir le résultat de la recherche. Les choix qui s'offrent présentement à l'utilisateur sont, soit le fichier des exercices récupéré est envoyé directement à son adresse électronique, soit le fichier résultat est formaté dans une page HTML et visualisé au Browser. Dans ce dernier cas, le PA intervient pour combiner le standard XML et HTML et produire une feuille sous format XSL (eXtended Style Language).

De part sa conception même, XML sépare un document proprement dit de sa grammaire DTD, de son format d'affichage et de ses mécanismes d'hyperliens, un document XML ne permettant pas, tel que, de décrire le format d'affichage ou d'impression d'un document. Pour cela nous utilisons une description XSL (que l'on écrit en XML) et qui s'applique à un document comme le fait son DTD, mais pour définir l'affichage des divers éléments. XSL est un complément « feuille de style » de XML comme CSS est un complément « feuille de style » de HTML. En effet, PA se sert de la description XSL pour décrire la mise en page de l'exercice reçu sous format XML.

L'agent client (CA)

C'est l'agent qui est directement en relation avec l'interface graphique utilisée par l'usager pour transmettre des requêtes de recherche. Une fois tous les paramètres de la recherche saisis (par exemple le cours concerné, les concepts importants et les STI cibles), un agent client est crée par le AA. Il est détruit à la fin de sa mission, qui se traduit soit par un échec ou un succès comme l'illustre la figure 19. En fait, cet agent se déplace vers le site destination, après une présentation à l'agent AA local, il entre en négociation avec le SA du site en question. Après avoir présenté sa requête, une réponse lui est rendue. Dans le cas d'une réponse positive au premier site rencontré, cet agent revient à la machine source, remet le résultat de la requête et s'autodétruit

automatiquement. Le résultat de la requête, qui représente un ensemble de fichiers, est inséré dans le contexte de l'agent pour pouvoir le rapatrier localement.

Nous avons vu précédemment que pour certains types d'exercices, notament « Steps Illustration » la taille des fichiers vidéos à transmettre pourrait être trop grosse. Pour l'optimisation de ce transfert, nous avons doté cet agent d'une fonctionnalité supplémentaire qui est la compression des données avant la transmission. La machine source ne peut effectuer cette opération car différents agents clients peuvent utiliser différents outils de compression. Donc, une fois arrivé à destination, l'agent client interroge l'agent de service pour des exercices spécifiques, si requête satisfaite, compresse les documents obtenus, si nécessaire, et revient sur son site d'origine avec les données compressées. Sur son site, l'agent va décompresser les données et les livrer à l'agent de présentation (PA). Pour étudier le coût de ce transfert, considérons deux machines dont le débit de transfert entre elles est dt. Nous supposons que le facteur de compression du document est fc (si T est la taille initiale du document, le document compressé a une taille de fc * T) et que les temps respectifs de compression et décompression sont tc et td. Nous cherchons à identifier les conditions à partir desquelles la compression du document est rentable.

Le temps de transfert du document avec compression est :

$tc + fc * T / dt + td$

respectivement le temps de compression du document, le temps de transfert du document compressé et le temps de décompression du document.

En conséquence, la compression est rentable si :

$tc + fc * T / dt + td < T / dt$

Ceci implique que l'agent client gagne dans les cas où le débit du réseau entre sa machine et la machine destination est tel que :

$dt < T(1 - fc) / (tc + td)$

Cette formule indique à partir de quand la compression est rentable. Nous ne prenons pas en compte ici le coût du transfert de l'agent lui même.

Cas pratique :

Pour réaliser notre évaluation, nous avons utilisé un fichier de données de taille 1,2 Mo. Nous avons utilisé l'outil de compression gzip et avons obtenu avec ce fichier un facteur de compression de 30%. Les temps de compression et décompression mesurés sont 2,8 secondes et 0,2 secondes. En reportant ces valeurs dans la formule ci-dessus, nous obtenons que la compression devient rentable lorsque dt < 280 Ko/s. En général, quand on essai de télécharger des documents à travers l'Internet, à l'université, le débit est de l'ordre de 180 à 250 ko/s, en fonction des moments et de la surcharge du réseau...Dans ce cas la compression est rentable. Par contre à l'intérieur du réseau local, nous obtenons un débit de l'ordre de 700 ko/s pour lequel la compression n'est pas rentable.

En conclusion, cette fonctionnalité de l'agent client sera éventuellement utilisée dans le cas d'un transfert de documents de grande taille.

Antonio et al.[Carzaniga 97] ont étudié les trafics réseaux générés par des applications réparties construites en utilisant les paradigmes client-serveur et agents mobiles. Cette étude s'appuie sur un modèle de coût prenant essentiellement en compte les volumes d'information transférés.

4.4 Le système d'Interface Graphique

L'interface graphique est le moyen que les usagers utilisent pour effectuer leur recherches à travers le réseau ou satisfaire les demandes qui proviennent de l'extérieur. Elle permet à un concepteur de rechercher des exercices à travers un réseau d'STI dans le monde avant de construire d'autres ou de mettre ses exercices à la portée des autres usagers. L'interface possède les caractéristiques suivantes :

- convivialité et facilité d'utilisation,
- facilité d'installation et d'initialisation,
- affichage des résultats de recherche sous un format familier

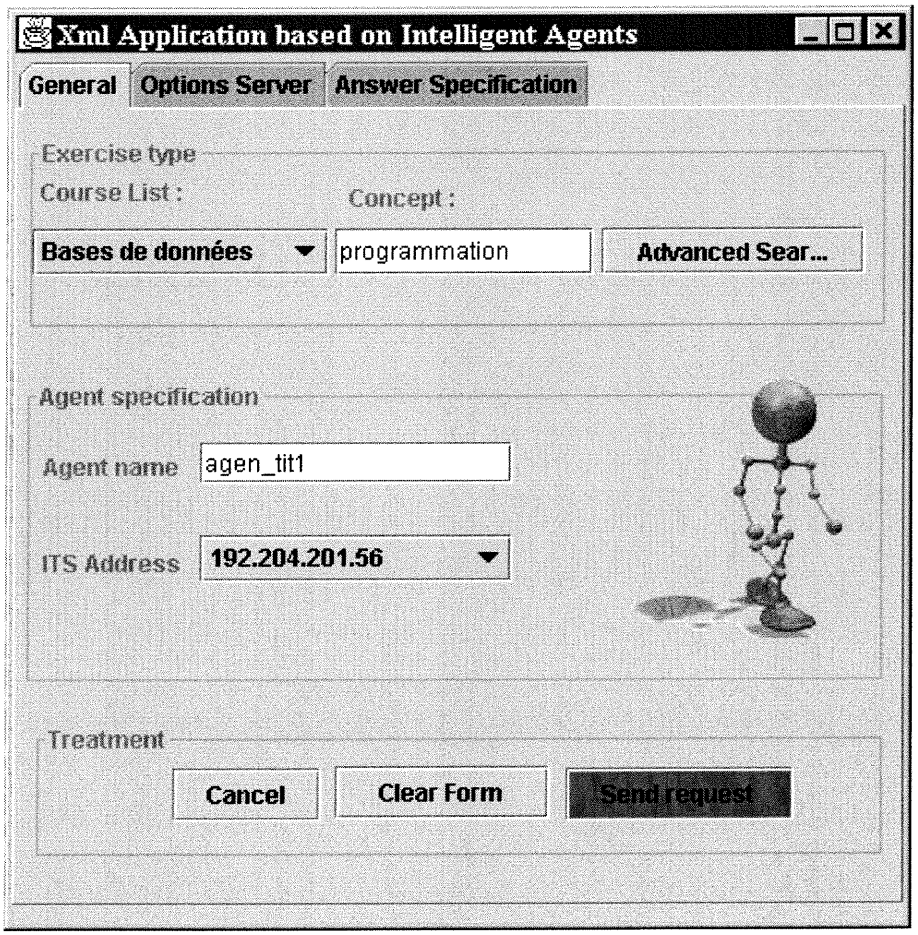


Figure 20 : Interface graphique utilisée pour effectuer une recherche d'exercices.

Pour effectuer une recherche, l'utilisateur doit spécifier une certains nombre de paramètres utiles pour réussir sa recherche. Il s'agit du titre du cours, le concept ou l'unité cognitive et la liste des STI où doit se passer la recherche. L'usager peut spécifier aussi s'il veut que les explications pour les cas d'erreurs de ces exercices soient rapatriées. Comme mentionné précédemment, pour chaque exercice, il existe une bonne réponse et un ensemble de mauvaises réponses. Pour chaque cas d'erreur, le concepteur peut éventuellement y associer un groupe d'explications. Sur demande d'un utilisateur, ces explications pourront être fournies sous format XML avec le contenu de l'exercice correspondant. La figure 20 ci-dessus représente une requête qu'un utilisateur désire envoyer à un STI bien précis sur le réseau. Cette requête concernant des exercices sur le cours bases de données avec comme concept la programmation.

4.5 Conclusion

Dans ce chapitre, nous avons abordé la seconde partie du prototype qui consiste en la mise en place d'un système distribué pour les exercices pédagogiques. Les partenaires concernés sont les spécialistes de STI à travers le monde.

Trois sous-parties ont été étudiées : l'interface graphique, le système multi-agents et le système de stockage et traitement des données.

Nous avons axé notre description sur les deux dernières parties car elles constituent la base du prototype. Nous avons décrit un système réduit d'agents coopératifs pour assurer des tâches bien précises. Un autre nouvel aspect a été traité qui est la hiérarchisation des agents.

Dans la dernière partie, nous avons montré comment sont traitées les données, l'apport de XML dans l'échange d'information entre des applications développées sous diverses plates-formes. Enfin, nous avons décrit le rôle des analyseurs (parsers) XML pour supporter les structures variées et complexes des documents.

Plusieurs moyens techniques s'offrent à nous pour la réalisation de cette solution technique. Dans le chapitre suivant, nous faisons le point sur les choix effectués et comment ils ont été implantés.

CHAPITRE 5

IMPLANTATION

5.1 Introduction

Nous décrivons dans ce chapitre les techniques et outils utilisés pour implanter le prototype du système multi-agents dont la conception a été détaillée précédemment. Nous allons détailler les trois sous-systèmes identifiés à savoir l'interface graphique, le système multi-agents et le système de stockage des exercices.

Les points important pris en considération lors de l'implantation sont :

- les plates-formes cibles,
- l'environnement d'exécution,
- les langages utilisés,
- les outils et techniques utilisés pour implanter l'aspect distribué,
- le standard utilisé pour assurer le stockage, l'accès et l'interopérabilité des applications.

L'environnement pour lequel est destiné le prototype est un environnement distribué, en l'occurrence Internet. L'application peut être téléchargée à travers le web et installée dans une machine locale.

5.2 Les langages utilisés

Le principal langage utilisé pour développer les différentes composantes du système est le langage Java de Sun Microsystems.

5.2.1 L'interface graphique

Elle représente la clé du succès pour le développement d'un système informatique. En effet, l'interface graphique représente la porte d'entrée au système, une fois cette porte l'accueille agréablement, l'usager trouve un plaisir à utiliser le système. Pour cela, nous avons développé cette partie importante du système en Java, en utilisant l'API SWING de SUN. Cette solution présente les avantages suivants :

- interface graphique conviviale et riche en fonctionnalités,
- système moins alourdi car l'interface de l'application s'exécute sur la machine cliente. Car si notre interface comporte plusieurs objets graphiques, cela peut engendrer un temps de chargement assez lent.
- intégration facile d'objets Java distribués.

5.2.2 Le système Multi-agents

Les agents du système ont été développés en Java au-dessus de l'environnement de programmation distribué Voyager (objectspace). Java a la réputation d'être un langage lent, mais les avantages offerts restent toujours plus importants par rapport à cet inconvénient. Quant aux applications distribuées, Java reste un outil convenable pour le développement car il possède des APIs intéressantes pour tout ce qui concerne les objets distribués et le réseautage.

5.2.3 Le système de stockage et d'accès

Les caractéristiques de tous les exercices conçus dans un STI sont stockées dans une base de données ACCESS de Microsoft. Le choix de ce SGBD n'est pas obligatoire, tout autre système pourrait être mis en place. Mais ce choix a été essentiellement motivé par l'aspect coût. La présentation de l'exercice, quant à elle, est stockée dans un fichier HTML pour les raisons évoquées précédemment à savoir la portabilité du texte, images, son, vidéo, etc. Mais une fois que le concepteur décide de mettre des exercices à la disposition du public, tous ces exercices sont générés sous le format standard XML.

5.2.4 XML et Java

Le langage Java offre la portabilité du code, XML la portabilité des données : les deux sont donc complémentaires. On accédera donc à des objets XML (documents et leurs éléments) portables à l'aide de classes Java portables elles aussi. Le slogan Java « Write Once, Run Everywhere » peut s'étendre à XML sous l'une des deux formes « Write Once, Store Everywhere » ou « Write Once, Publish Everywhere ».

Sun et des partenaires ont défini un ensemble de classes Java d'accès à des objets DOM/XML constituant une extension de la bibliothèque standard du langage. Également, Sun et Oracle ont offert une API appelée « SAX » (Simple API to XML) qui permet l'accès à des documents et messages XML.

En concevant XML, le consortium W3C a cherché à surmonter les limitations du Html avec une syntaxe à balises qui pourrait être étendue indéfiniment. En utilisant l'arbre issu du Document Object Model (DOM), les données provenant de n'importe quelle source peuvent être englobées dans une structure de données et peuvent par la suite être utilisées par n'importe quelle application pour l'analyse et l'exploitation de ces données. Puisque la structure et les données sont regroupées dans une instance de l'objet document, donc le document est la *base de données*. Cette « base données » peut être traitée par toute application qui manipule XML, même si l'origine de ces données est méconnue pour cette application.

- ***Qu'est ce que le DOM(Document Object Model) ?***

Le DOM est une interface de programmation d'application (API) pour les documents HTML ou XML. Il définit la structure logique du document et la façon d'accéder et de manipuler ce même document. Dans la spécification du « DOM », le terme 'document' est utilisé dans le sens le plus large, XML est défini pour représenter différents types d'informations qui pourraient être stockées dans diverses plates-formes, traditionnellement ces informations sont considérées comme données plutôt que documents. Néanmoins, XML présente ces données comme documents et DOM pourrait être utilisé pour le traitement de ces données.

À l'aide de DOM, nous pouvons établir des documents, naviguer dans leurs structures, et ajouter, modifier ou détruire des éléments de ce document. Toute information présente dans un document XML peut être changée, détruite ou ajoutée en utilisant le DOM.

Prenons un exemple d'un fichier XML et sa représentation avec DOM :

```
<?xml version="1.0" ?>
<!DOCTYPE Base SYSTEM "Exercise.dtd">
<document>
<Exercise>
  <IDExercise>XMA0001</IDExercise>
  <Title>Commandes</Title>
</Exercise>

<Exercise>
  <IDExercise>ORA0001</IDExercise>
  <Title>PLSQL</Title>
</Exercise>
</document>
```

Figure 21 : Un petit fichier XML pour illustrer le DOM.

Le DOM représentant la table ci-dessus est le suivant :

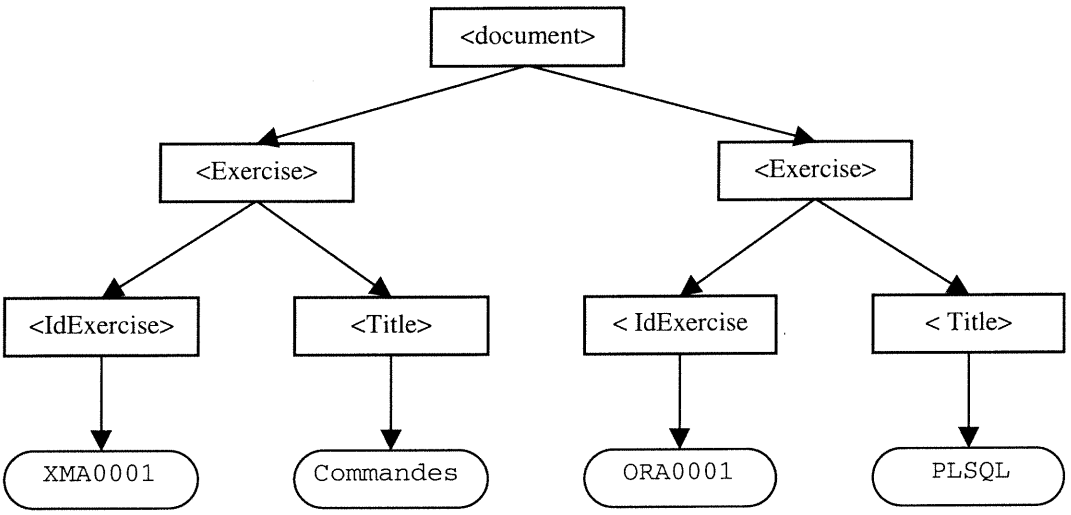


Figure 22 : Représentation DOM correspondant à l'exemple ci-dessus.

Grâce à son appartenance à la famille des langages orientés objet, Java a été un choix judicieux pour supporter les structures DOM. Créer son propre algorithme d'analyse (parser) de ces documents serait inutile, puisqu'il existe sur Internet des programmes standards comme celui d'IBM conçu pour le traitement de documents XML (XML4J) pour Java [sites XML $_{(3)}$].

5.2.5 Le système à objets distribués Voyager

Voyager est un système en langage Java qui offre une API aux usagers pour la programmation d'applications distribuées portables. Grâce à ce système, nous faisons abstraction de l'aspect distribué d'une application en utilisant des objets distants. Il supporte le développement d'applets, les architectures clients/serveurs, les agents mobiles et présente certaines caractéristiques de mise en œuvre comme la persistance d'objets, le stockage d'objets distants, la sécurité et la programmation Web.

5.2.6 Java RMI vs. CORBA

Le choix entre l'utilisation de la technologie RMI (Remote Method Invocation) ou la technologie CORBA (Common Object Request Broker Architecture) reste un grand débat au sein de la communauté scientifique qui s'intéresse aux applications distribuées. En comparant les avantages et les inconvénients [sites: Java RMI & CORBA] des deux technologies, nous ne pouvons dire si l'une des deux technologies est meilleure que l'autre, tout dépend de l'usage qu'on souhaite en faire. Dans notre cas, le choix s'est porté sur Java RMI pour les raisons suivantes :

- Expérience déjà acquise de cette technologie car elle était déjà introduite depuis la version 1.02.

- CORBA ne permet pas l'envoi ou la transmission de code exécutable (agents) ou d'objet en général, sur une machine distante.

- Java est un langage jeune et récent, encore en développement, donc sa capacité à se conformer à la technologie CORBA laisse les scientifiques perplexes. En outre, toutes les caractéristiques de CORBA ne sont pas présentes dans les divers produits Java/CORBA sur le marché.

Les conséquences de notre choix résident dans le fait que nous ne pouvons communiquer avec d'autres objets développées à l'aide d'autres langages autres que Java (C, C++, Ada,...) et nous restons loin du standard CORBA.

5.3 Les plates-formes utilisées

Le développement de tous les modules est fait sous Windows-NT. Mais comme le langage de programmation utilisé est Java « Write once, Run Everywhere », il devrait pas y avoir de problèmes quant à un changement de plate-forme.

Notre implémentation pour le transport et le traitement des données XML utilise un simple système d'agents qui peuvent opérer sur n'importe quelle plate-forme capable de supporter une machine virtuelle Java et une communication TCP/IP, incluant la majorité

des systèmes d'exploitation comme Linux, NT, Macintosh OS, Unix et Solaris. Ces agents sont implémentés en utilisant des systèmes ouverts, comme Java, Java RMI, plate-forme Voyager (Java) et W3C XML DOM. Nous avons aussi utilisé l'analyseur standard de fichiers XML d'IBM (XML4J).

5.4 État d'avancement de notre prototype

Comme nous l'avons mentionné précédemment, notre travail a été subdivisé en deux parties. La première qui concerne la génération de plusieurs types d'exercices pédagogiques est achevée et testée. Le module gestionnaire d'exercices (MGE) est fonctionnel et permet ce qui suit :

- conception de plusieurs types d'exercices spécifiques et génériques,
- interaction avec l'agent pédagogique pour le suivi des performances des apprenants,
- sauvegarde des caractéristiques des exercices la base de données,
- génération de la présentation html des exercices.

Notre prototype a permis la conception de deux cours présentement :

- Oracle avec des exercices spécifiques et génériques,
- PowerPoint avec des exercices génériques (en cours de développement).

La seconde partie a été divisée en sous parties comme suit :

- génération d'exercices conçus en format XML, cette sous partie est finie, testée et opérationnelle,
- mise en place de l'architecture multi-agents pour l'échange de ces exercices avec d'autres STI. Cette partie est finie mais testée uniquement sur le réseau local entre deux machines. La fonctionnalité de l'agent client (CA), qui sert à compresser les gros fichiers pour pouvoir les transférer sur le réseau, présente encore des problèmes.

Nous avons constaté, lors de la phase du test, que certaines requêtes sont restées sans réponse. En outre, nous nous sommes confrontés à un problème de performances car nous utilisons beaucoup de 'threads : processus parallèles pour les agents'. Une fois le nombre de requêtes en cours devient important, le système s'arrête.

La première tâche à réaliser pour la continuité de ce travail serait d'optimiser les programmes pour rendre le système plus stable. Ensuite, apporter quelques améliorations à l'interface graphique, comme indiqué dans la figure 20, au lieu de donner l'adresse IP du destinataire, l'usager donne juste un nom de site qui sera facile à retenir.

CHAPITRE 6
CONCLUSION

Ce rapport présente les résultats de recherche d'une partie du projet mené au sein du laboratoire Héron pour l'étude et la mise en place d'un Système Tutoriel Intelligent complet. Ce projet a consisté en la réalisation d'un prototype offrant la possibilité de suivre un cours intelligent à travers l'Internet.

Les points importants abordés lors de cette étude sont :

- la définition de l'architecture globale de notre STI,
- le rôle et la définition de plusieurs types d'exercices dans notre STI,
- l'architecture distribuée pour l'interopérabilité avec d'autres STI.

L'étude des systèmes existants nous a permis d'identifier les principales caractéristiques que devrait présenter notre prototype. Nous avons passé en revue plusieurs architectures développées auparavant afin de profiter de leurs expériences, combler leurs lacunes si nécessaire et utiliser de nouvelles technologies pour améliorer la performance et la fiabilité d'un nouveau système. Basé sur les résultats de la synthèse des systèmes étudiés, nous nous sommes fixés un certain nombre d'objectifs à atteindre :

- offrir des systèmes de formation et d'évaluation évolutifs et conviviaux,

- assister intelligemment l'apprenant dans sa formation,
- communiquer avec d'autres systèmes sans entraves de langues,
- assurer l'évolution du système.

Pour atteindre nos objectifs, nous avons mis en œuvre un système basé sur une architecture moderne à savoir une architecture multi-agents et utilisé un standard inévitable de nos jour pour le stockage et l'interopérabilité à savoir XML.

Différents types d'agents ont été utilisés pour mettre en place le système. Certains d'entre eux sont indispensables au fonctionnement du système et constituent les éléments de base, comme par exemple, l'agent d'administration, l'agent client et l'agent de service. Pour chaque nouveau service introduit dans le système, nous utilisons un ou plusieurs agents pour l'implémenter. Cette architecture multi-agents nous a permis de faire face aux problèmes de traitement rapide de recherche d'exercices ou de satisfaction d'une demande venue de l'extérieur. Les agents que nous avons utilisés sont simples et faciles à implémenter. Néanmoins, des services plus complexes et plus riches, comme la recherche multilingue, peuvent être conçus et intégrés au système.

Quant à l'interopérabilité entre les différents STI partenaires, elle a été assurée par l'utilisation du standard XML. En effet, XML nous a permis de définir, tous les types d'exercices réalisés, dans un format compréhensible par n'importe quel STI développé avec n'importe quel langage.

Cette étude a été sanctionnée par la publication d'un article [Boukherouaa et al., 2000] dans lequel nous avons fait part des avantages qu'offre la technologie des agents dans le domaine du travail coopératif et de l'échange mutuel d'expertise. Cet article a été accepté et sera publié dans la conférence International (TICE 2000) qui se tiendra à Troyes en France du 18 au 20 Octobre prochain.

Travaux futurs :

Le développement des différentes parties de notre prototype a été faite d'une façon modulaire. Ceci dans le but de faciliter l'extension du système. En ce qui concerne la première partie du projet à savoir l'édition des exercices, nous pensons que de nouveaux types d'exercices peuvent être rajoutés. Prenons comme exemple, des exercices où l'apprenant doit saisir une réponse sous la forme d'une phrase ou d'un paragraphe. Le système doit être capable d'analyser syntaxiquement les mots contenus dans la phrase, faire une corrélation avec la phrase juste et déterminer si la réponse pourrait être acceptée ou pas. Nous disons corrélation car l'apprenant peut se tromper dans l'écriture de certains mots ou mal finir sa phrase.

En ce qui concerne la seconde partie, l'interopérabilité entre les systèmes, nous pensons à une stratégie d'encouragement des concepteurs. Dans le chapitre quatre, nous avons montré que la coopération entre les STI évite la redondance, permet la réutilisation et améliore la qualité des exercices. Cependant, plusieurs aspects restent indéfinis, entre autres, « Comment inciter les concepteurs à mettre le fruit de leur travail à la disposition des autres? » et « Comment s'assurer que cette contribution est de qualité? ». Dans [Boukherouaa et al., 2000], une stratégie a été définie pour encourager les experts à contribuer et contrôler leur travail. En effet, pour encourager les concepteurs à mettre leurs exercices pédagogiques à la disposition d'autres personnes, nous avons pensé à un système de récompense basé sur le meilleur concepteur d'exercices dans un cours donné. En fonction des performances de ces exercices, le système élira un expert pour ce cours. Les performances sont calculées en fonction d'une évaluation faite par tous les utilisateurs de ces exercices. Le rôle que va assurer cet expert dans le futur est de contrôler tous les exercices, pour le cours en question, qui vont être échangés entre les utilisateurs. Ceci va nécessiter le passage, des données échangées, par un serveur central pour vérification avant acheminement.

Ce module pourrait intégrer le système afin d'améliorer la qualité des exercices qui sont échangés par les différents STI faisant partie du réseau.

Nous pensons aussi à l'extension de l'agent client (CA) afin d'inclure des fonctionnalités supplémentaires telle que la traduction. En effet, un concepteur bilingue ou trilingue pourrait utiliser les services de cet agent pour effectuer une recherche en plusieurs langues. Le concepteur spécifie les caractéristiques des exercices recherchés et l'agent s'occupe de la traduction des termes et peut apporter des résultats avec les différentes langues désirées. Comme il s'agit de la traduction simple de termes, cette extension serait un atout pour notre prototype.

BIBLIOGRAPHIE

[Aïmeur et Frasson, 1996] E. Aïmeur, C. Frasson. "Analyzing a new learning strategy according to different knowledge levels", Computer and Education, An International Journal, vol. 27, no. 2, pp 115-127, 1996.

[Aïmeur et Fahmi, 1998] E. Aïmeur, M. Fahmi. "Pedagogical Agents in DTL: the Double Test Learning Strategy", Workshop 2 on Pedagogical Agents in ITS'98, International Conference on Intelligent Tutoring Systems, San Antonio, Texas, pages 14-19,1998.

[Alessi et Trollip, 1985] Anderson, J.R. « Computer-Based Instruction: Methods and Development « , Prentice-Hall, Inc., Englewood Cliffs, NJ, 1985.

[Arsac, 1987] J. Arsac "Les machines à penser. Des ordinateurs et des hommes », Éditions du Seuil, 1987.

[Barbuceanu et Fox, 1997] M. Barbuceanu, M. Fox « Integrating Communicative Action, Conversations and Decision Theory to Coordinate Agents», Proceedings of the First International Conference on Autonomous Agents, Marina del Bay, California, February 5-8, 1997.

[Barr et Feigenbaum, 1982] A. Barr, E. A. Feigenbaum « The handbook of Artificial Intelligence », vol. 2, HeurisTech Press, William Kaufmann 1982.

[Bestougeff et Fargette, 1982] H. Bestougeff, J. P. Fargette « Enseignement et ordinateur », Presse universitaire de France, Paris, 1982.

[Boukherouaa et al., 2000] E. Boukherouaa, E. Gardouh, C. Frasson « Système de gestion des intérêts d'une communauté au moyen d'un agent Intelligent appelé 'DOC' », Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie (TICE 2000). 18-20 Octobre 2000 – Troyes, France.

[Brown et al., 1982] J. S. Brown, R. R. Burton, J. Kleer « Pedagogical, natural language and knowledge engineering in Sophie I, II and III, in D. », In Sleeman and

J.S. Brown (Eds.), « Intelligent tutoring systems », pp 227-282, Academic Press, New York, 1982.

[Burns et Capps, 1988] H. L. Burns, C. G. Capps « Foundations of Intelligent tutoring system : an introduction », In Polson, M.C. and Richardson, J. J. (Eds.) Foundation of Intelligent Tutoring Systems, pp 21-53, Lawrence Erlbaum Associates Publishers, 1988.

[Chan et Baskin, 1990] T. W. Chan, A. B. Baskin « Learning companion systems », In Frasson, C. and Gauthier, G. (Eds.), Intelligent Tutoring Systems : At the Crossroads of Artificial Intelligence and Education, pp 6-33, New Jersey : Ablex Publishing Corporation, 1990.

[Finin et Labrou, 1998] T. Finin, Y. Labrou « Agent Communication Languages», Proceedings of the Second International Conference on Autonomous Agents, St. Paul, Minneapolis, May 9-13, 1998.

[Frasson et Gauthier, 1989] C. Frasson, G. Gauthier « Intelligent Tutoring Systems : At the Crossroads of Artificial Intelligence and Education », Norwood, NJ : Ablex, 1989.

[Frasson et Gauthier, 1994] C. Frasson, G. Gauthier « A gradual software environment for developing tutoring systems », Advances in Artificial Intelligence, Theory and Application II, vol. 2, pp 73-79, 1994.

[Frasson et al., 1996a] C. Frasson et al. « Rapport semestriel d'activités » Projet SAFARI, Université de Montréal, Laboratoire Héron, 1996.

[Frasson et al., 1997] C. Frasson, T. Mengelle, E. Aïmeur « Using Pedagogical Agents In a Multistrategic Intelligent Tutoring System », Workshop on Pedagogical agents in AI-ED 97, World Conference on Artificial Intelligence and Education, Japon, pp 40-47, 1997.

[Gilbert et al., 1995] D. Gilbert, M. Aparicio, B. Atkinson « Intelligent Agent Strategy », Rapport technique, Research Triangle Park, IBM Corporation, 1995.

[Guttman et Maes, 1998] R. Guttman, P. Maes « Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce. », Proceedings of the Second International Workshop on Cooperative Information Agents, Paris, France, July 3-8, 1998.

[Herzog, 1992] Ch. Herzog « From elementary knowledge schemes towards heuristic expertise-Designing an ITS in the field of parallel programming », In Frasson, C., Gauthier, G. and G. I. McCalla (Eds.) : Intelligent Tutoring Systems, Second International Conference, ITS'92 Proceedings, pp 96-105, Springer-Verlag Berlin Heidelberg, 1992.

[Hollan et al., 1987] J. D. Hollan, E. L. Hutchins, L. Weitzman « STEAMER : an interactive inspectable simulation-based training system », In G. Kearsley (Ed.),

Artificial intelligence and Instruction, Addison Wesley, pp 113-134, 1987.

[Holt et Wood, 1990] P. Holt, P. Wood « Intelligent tutoring systems : a review for beginners », Canadian Journal of Educational Communication, vol. 19, no. 2, pp 107-123, 1990.

[Lefevre, 1988] J. M. Lefevre « Guide pratique de l'EAO », Édition Cedic Nathan, 1988.

[Mengelle et Frasson, 1996] T. Mengelle, C. Frasson « A multi-Agent architecture for an ITS with multiple strategies », In A. Díaz de Ilarraza Sanchez and I. fernandez de Castro (Eds) : Computer Aided Learning and Instruction in Science and Engineering, Third International Conference « CALISCE'96 » Proceeding, San Sebastián, Spain, pp 96-105, Springer-Verlag Berlin Heidelberg, 1996.

[Miller, 1988] J. R. Miller « The role of human-computer interaction in intelligent tutoring systems », In Polson, M.C. and Richardson, J. J. (Eds.) Foundation of Intelligent Tutoring Systems, pp 21-53, Lawrence Erlbaum Associates Publishers, 1988.

[Moore, 1999] D. Moore « Creating Social Agent Desktop Applications : A practitioner's Guide », Proceedings of the Third International Conference on Autonomous Agents, Seattle, Washington, May 1-5, 1999.

[Morris et Maes, 2000] J. Morris, P. Maes « Sardine : An Agent-facilitated Airline Ticket Bidding System », Proceedings of the Fourth International Conference on Autonomous Agents, Barcelona, Spain, June 3-7, 2000.

[Péninou, 1993] A. Péninou « M. A. C. T. : Un modèle d'Agents Centrés Tâches pour la production des systèmes tuteurs intelligents par l'Atelier de Génie Didacticiel Intégré », Doctorat de l'Université de Toulouse III, laboratoire A. P. I., 1993.

[Polson et Richardson, 1988] M. C. Polson, J. J. Richardson « Foundation of Intelligent Tutoring Systems », Lawrence Erlbaum Associates Publishers, Hillsdale, New Jersey, 1988.

[Shortliffe, 1976] E. H. Shortliffe « Computer-based medical Consultations : MYCIN » New York : Elsevier, 1976.

[Silverman, 1992] B. G. Silverman « Building expert critiquing systems : A situated tutoring alternative », In Frasson, C., Gauthier, G. and G. I. McCalla (Eds.) : Intelligent Tutoring Systems, Second International Conference, ITS'92 Proceedings, Tutoriel Proceedings, Montréal, 1992.

[Sleeman et Brown, 1982] D. Sleeman, J. S. Brown « Intelligent tutoring systems », Academic Press, New York, 1982.

[Stevens et Collins, 1977] A. L. Stevens, A. Collins »The goal structure of a Socratic tutor », In Proceedings of the National ACM Conference, Seattle, Washington, pp 256-263, Association for Computing Machinery, New York, 1977.

[W3C 98] Le site officiel du consortium chargé d'établir les normes sur le web. http://www.w3c.org/XML.

[sites: XML] http://www.ebXML.org

http://www.oasis-open.org/cover/sgml-xml

http://www.ibm.com/developer/xml/

http://www.xml.org

http://www.xml.com/xml/pub

http://www.ucc.ie/xml

http://www.xmlx.com

http://www.goxml.com

[sites: les agents] http://www.agent.org

http://agents.www.media.mit.edu/groups/agents

[sites: applications distribuées] http://www.netapp.com/tech_library/3071.html

http://dast.nlanr.net

[sites: Java RMI & CORBA] http://www.omg.org

http://www.javacoffeebreak.com/articles/rmi_corba/index.html