METADYNE, Un Hypermédia Adaptatif Dynamique pour l'Enseignement

Nicolas Delestre

Université de Rouen

Thèse

pour l'obtention du grade de Docteur de l'Université de Rouen

Spécialité : Informatique

présentée par **Nicolas DELESTRE**

METADYNE

Un Hypermédia Adaptatif Dynamique pour l'Enseignement

Soutenue le 20 Janvier 2000, devant le jury composé de :

Catherine BARRY-GREBOVAL	*INSA de Rouen, Examinateur*
Françoise GUEGOT	*INSA de Rouen, Examinateur*
Claude FRASSON	*Université de Montréal, Rapporteur*
Jean-Pierre PECUCHET	*INSA de Rouen, Directeur de Thèse*
Pierre TCHOUNIKINE	*Université du Maine, Rapporteur*
Philippe TRIGANO	*Université de Technologie de Compiègne, Président*

Thèse préparée au sein du Laboratoire PSI - INSA de Rouen

Je tiens très sincèrement à remercier :

M. Jean-Pierre Pécuchet, professeur à l'INSA de Rouen, pour m'avoir accueilli au sein de son équipe de recherche et pour m'avoir encadré dans ce travail.

Mme Catherine Barry-Gréboval, maître de conférence à l'INSA de Rouen, pour m'avoir co-encadré durant cette thèse et pour avoir patiemment relu plusieurs fois chaque chapitre de ce mémoire.

M. Frasson, professeur à l'Université de Montréal, pour avoir pris le temps de rapporter sur ce mémoire, et avoir pris la peine de venir du Canada pour ma soutenance.

M. Pierre Tchounikine, professeur à l'Université du Maine, rapporteur de cette thèse, pour avoir bien voulu examiner ce mémoire, et pour toutes les recommandations qu'il m'a prodiguées afin d'améliorer la qualité de ce rapport.

M. Philippe Trigano, professeur à l'Université de Technologie de Compiègne, pour avoir bien voulu examiner mon travail et pour l'honneur qu'il a bien voulu m'accorder en acceptant de présider ce jury.

Mme Françoise Guéguot, professeur associée à temps partiel à l'INSA de Rouen, pour avoir examiné mon travail et pour avoir bien voulu faire partie de mon jury.

Je tiens aussi à remercier tous les membres du laboratoire PSI, Nathalie, Mahmed, Habib et les autres permanents, ainsi que les thésards, tout particulièrement Gaëtan et Marc, pour m'avoir épaulé à de nombreux moments.

Pour finir, ces remerciements ne seraient pas complets si je n'avais pas une pensé pour les êtres qui me sont les plus chers, c'est-à-dire ma famille :

Je dédie donc ce mémoire à mes parents, Françoise et Jean, qui durant toute ma scolarité m'ont toujours donné la possibilité de faire ce que je voulais, et qui ont toujours cru en moi.

Je dédie également ce document à mon frère Benoît et sa femme Nathalie, ainsi qu'à mes quatre nièces, pour le bonheur qu'ils me procurent tout au long de l'année.

J'embrasse très fort mon amie Véronique.

Je remercie chaleureusement ma cousine Caroline pour avoir relu ce manuscrit et avoir enlevé la grande majorité des fautes d'orthographe que j'avais pu laisser, et vous pouvez me croire, elle avait du travail !

Table des matières.

Table des figures.

Table des tableaux.

Introduction.

Depuis Jules Ferry, les outils pour dispenser des cours n'ont évolué que très lentement. Bon nombre d'enseignants utilisent encore le tableau noir, surtout dans l'enseignement secondaire. On a bien vu dans les années quatre-vingt une intrusion timide d'outils multimédia, tels que les laboratoires de langue, ou encore l'utilisation des vidéos. Mais ce type d'enseignement reste très minoritaire.

A côté de cela, de nos jours, l'informatique prend une place prépondérante dans la vie de tous les jours. On la retrouve aussi bien sur notre lieu de travail, que dans nos loisirs. Outre son utilisation dans un cadre professionnel, depuis quelques années l'aspect multimédia de ces appareils attire de plus en plus de personnes.

Mais qu'en est-il de son utilisation dans un cadre éducatif ?

Aujourd'hui l'enseignement assisté par ordinateur, qui se veut intelligent et interactif, prend un nouvel envol. En effet, les capacités multimédia des ordinateurs depuis quelques années ont fait d'énormes progrès, tant au niveau matériel (carte sonore, carte de décompression vidéo MPEG, carte vidéo 2D et 3D, etc.) qu'au niveau logiciel (différents algorithmes de codage, de compression, tel que le MP3, etc.). De plus, depuis environ cinq ans, les ordinateurs peuvent être facilement interconnectés grâce au réseau mondial Internet, et ceci avec une évolution constante de la qualité de la bande passante (il suffit de comparer les vidéos en streaming d'aujourd'hui et d'il y a seulement un an). Cela permet donc la production de logiciels éducatifs plus vivants, plus démonstratifs, et pouvant être utilisés de façon délocalisée.

Cet engouement n'est pas dénué de tout fondement car il peut profiter à tous les types d'apprentissage. En effet :

- L'enseignement traditionnel peut utiliser ce type d'apprentissage pour aider les étudiants en difficulté. Difficultés qui peuvent être dues à quelques échecs scolaires que l'on peut malheureusement définir comme classiques, mais aussi à des problèmes physiques (par exemple il est important que les enfants ou adolescents malades, puissent tout de même pouvoir suivre un cursus scolaire approprié à l'hôpital).
- L'enseignement à distance peut utiliser ce type d'apprentissage pour améliorer la qualité de l'enseignement et éviter le classique cours par correspondance.

- La formation continue peut utiliser ce type d'apprentissage afin par exemple de réduire les coûts de la formation des personnels d'une entreprise, à longue échéance.

Mais qu'est ce que l'enseignement assisté par ordinateur (EAO) ?

Il est bien évident que l'objectif n'est pas d'éliminer les enseignants mais plutôt d'apporter une aide à l'apprenant. On est loin des rêves de certaines personnes, telles que Simon Ramo et son projet C.M.I.[1], dont l'objectif était de ne plus insérer l'être humain dans les cycles d'apprentissage.

Cependant, pour apporter une aide réelle à l'apprenant il faut identifier les connaissances qui doivent lui être transmises. Tout le monde s'accorde alors à penser qu'un enseignement est la transmission de deux types de connaissance, le savoir et le savoir-faire, ce que l'on nomme aussi pour garder le terme "connaissance", les connaissances déclaratives et les connaissances procédurales [Mendelsohn 95]. Par exemple l'enseignement en mathématique du calcul de dérivés de fonctions dans le secondaire est obtenu par l'enseignement simultané ou séquentiel des formules de calcul, tel que $(U . V)' = U'V + V'U$, qui est un savoir, et le calcul de $(2x+3)'$, qui est l'utilisation du savoir, c'est-à-dire un savoir-faire.

Jusqu'à présent, les logiciels d'EAO se sont principalement attachés à améliorer le savoir-faire des apprenants, en leur demandant de résoudre certains problèmes. De plus en plus perfectionnés, ils arrivent même a être qualifiés d'intelligent, car sachant raisonner sur le domaine enseigné et sachant s'adapter à chaque apprenant. On parle alors de système d'enseignement intelligemment assisté par ordinateur (EIAO). A ce niveau, plusieurs écoles de pensées sont présentes, qui se différencient principalement au niveau de la liberté laissé à l'apprenant.

Tout d'abord, le mouvement béhaviorisme, initié par John B. Watson en 1913, dont Skinner fut l'un des premiers représentants dans le monde de l'EAO. Il considère qu'il est possible d'enseigner n'importe quelle notion à un élève si on utilise la technique dite de l'enseignement programmé. Le principe de cet enseignement repose sur la décomposition d'un concept en éléments plus simples, qu'il nomme unité ou *frame* en anglais, et de les présenter à l'apprenant qui assimile à son propre rythme, via un système de questions-réponses.

Ensuite, le mouvement constructivisme, dont J. Piaget est l'initiateur, s'attache à laisser plus de liberté à l'apprenant. L'apprentissage est alors issu des interactions entre ce dernier et la machine. Les systèmes issus de ce courant de pensée sont alors qualifiés d'environnements interactifs d'apprentissage par ordinateur (EIAO).

Mais qu'en est-il du savoir ?

En effet, avant de mieux maîtriser le savoir-faire, il est indispensable de maîtriser ou du moins d'avoir un minimum de savoir (car le savoir-faire peut aussi améliorer le savoir). Or aujourd'hui on constate que la transmission de ce savoir est le parent pauvre de l'EIAO[2].

Les seules études actuelles sur le domaine s'appuient essentiellement sur l'utilisation des NTIC[3] et donc sur l'aspect interactif des cours et sur l'impact du multimédia sur l'apprentissage.

En effet, l'explosion des possibilités multimédia des ordinateurs, ainsi que la possibilité d'accéder rapidement à de gros volumes d'informations (qu'ils soient *off-line*, avec par exemple l'utilisation des CD-ROM et des DVD-ROM, ou *on-line*, avec Internet) ont incité des enseignants, des établissements, des académies ou même des consortiums à produire des cours multimédia accessibles depuis un simple navigateur Web. Par exemple :

---

1. Computer Managed Instruction.
2. Ici, EIAO signifie Enseignement Intelligent et/ou Interactif Assisté par Ordinateur
3. Nouvelles Technologies de l'Information et de la Communication

- M. Xavier Hubaut propose un site des plus complets sur les mathématiques du secondaire[4],
- le site Web AZURNET de l'académie de Nice[5] propose des cours pour les élèves du primaire et du secondaire,
- le projet national Premier Cycle Sur Mesure[6] propose plus de 600 heures de cours pour les étudiants des premiers cycles universitaires scientifiques,

Cependant la production de ce type de serveur coûte énormément cher (en temps pour les particuliers et en argent pour les institutions), et impose des compétences en informatique, surtout pour le développement d'éléments multimédia didactiques tels que des *applets*[7]. De plus une fois en ligne, ces cours semblent à notre avis difficilement utilisables car il n'existe pas une seule et unique façon d'enseigner une notion.

Dès lors, à côté de ces cours construits et statiques, divers projets tentent plutôt de fournir des outils permettant de récupérer des éléments didactiques multimédia, et permettant à des enseignants de construire leur propre cours en les organisant. On peut citer entre autres les projets SEMUSDI, ARIADNE, CDE, etc., que nous allons voir par la suite. Ces projets, qui pour la plupart sont en cours d'élaboration, vont alors permettre à chaque enseignant qui le désire de pouvoir mettre à la disposition des élèves ou des étudiants des cours multimédia en adéquation avec leurs cours dits classiques. On obtient alors des cours adaptés aux caractéristiques de chaque enseignant, on parle alors de cours adaptable.

C'est dans ce prolongement que s'inscrit notre travail, en étendant cette adaptabilité aux apprenants, et en tentant de rendre le système adaptatif (terminologie qui sera plus amplement définie par la suite).

Le projet METADYNE, pour hyperMEdia adaTAtif DYnamique pour l'Enseignement, tente donc :

- de définir l'ensemble des connaissances qui doivent être mises en oeuvre par le système et par les formateurs pour construire des cours hypermédia "intelligents", c'est-à-dire adaptés aux caractéristiques des apprenants. Ces caractéristiques pouvant être la connaissance de l'apprenant sur tel ou tel domaine, aussi bien ses envies que ses objectifs.
- de déterminer une architecture logicielle qui permette aux utilisateurs d'utiliser le système depuis un simple navigateur Web, tout en ayant à disposition des outils dont l'interface utilisateur n'est pas limitée aux possibilités intrinsèques de ces navigateurs.

Suite à cette introduction, nous allons commencer notre étude par répertorier de façon chronologique les différentes familles de système d'enseignement assisté par ordinateur, qu'ils soient "intelligents" ou non, en nous attardant plus précisément sur les systèmes qui permettent de transmettre un savoir à l'aide des nouvelles technologies.

Ensuite, dans le deuxième chapitre, après avoir défini les termes hypertextes et hypermédia, nous étudierons les différentes catégories de système hypermédia, ainsi que leurs apports potentiels dans un cadre éducatif.

Le troisième chapitre quant à lui, va s'attacher à définir plus amplement les différents objectifs du projet Métadyne.

Ce troisième chapitre sera bien entendu suivi du quatrième chapitre qui présentera le modèle conceptuel de notre système, c'est-à-dire les différentes connaissances mises en jeu (modèle du

[4] http://www.bib.ulb.ac.be/coursmath/
[5] http://www.ac-nice.fr/
[6] http://www.univ-enligne.prd.fr/
[7] Programme écrit en Java qui se trouve sur un serveur Web et qui s'exécute au sein d'un navigateur Web

domaine, modèle de l'apprenant, etc.), ainsi que leur représentation respective.

Dans le cinquième chapitre, nous allons voir comment il est possible de représenter informatiquement, à l'aide d'une modélisation objet, les différentes connaissances que l'on aura extraites dans le chapitre précédent.

Le sixième chapitre s'attachera ensuite à présenter l'architecture logicielle de notre système, en nous attardant sur chaque maillon de notre chaîne de systèmes clients-serveur.

Le septième chapitre quant à lui, à travers un "cas d'école", présentera les différentes fonctionnalités de notre serveur pour les différents utilisateurs du système, c'est-à-dire les enseignants et les apprenants.

Enfin, après la conclusion générale de ce travail, en annexe, nous étudierons plus en détail les fonctionnalités et l'architecture du serveur SEMUSDI, qui est un exemple d'outil indispensable au bon fonctionnement du système Métadyne.

Chapitre I - Les systèmes d'enseignement assisté par ordinateur.

Avant l'apparition et l'utilisation de l'informatique, les années cinquante ont vu apparaître les premières machines à enseigner (Cf. figure 1, [Bordeleau 99]). On peut en retenir deux principales :

La première date de 1958, créée par Porter, s'appuie sur les théories béhavioristes, nommées aussi comportementalistes, de Skinner. Cette théorie de l'apprentissage a donné naissance à une ingénierie pédagogique, nommée enseignement programmé, dont l'objectif est d'automatiser le processus d'apprentissage sous la forme stimulus-réponse-renforcement. Cet enseignement est donc un enseignement linéaire : les connaissances sont présentées séquentiellement à l'élève. Ce dernier ne peut passer à la connaissance suivante que lorsqu'il a bien assimilé la connaissance courante.

La seconde date de 1959 et fut construite par Crowder, un instructeur de la U.S. Air Force. Contrairement à Skinner, Crowder considère que les erreurs commises par un apprenant doivent être prises en considération et ne doivent en aucun cas l'empêcher de progresser : il préconise un enseignement par branchement et non plus un enseignement linéaire.

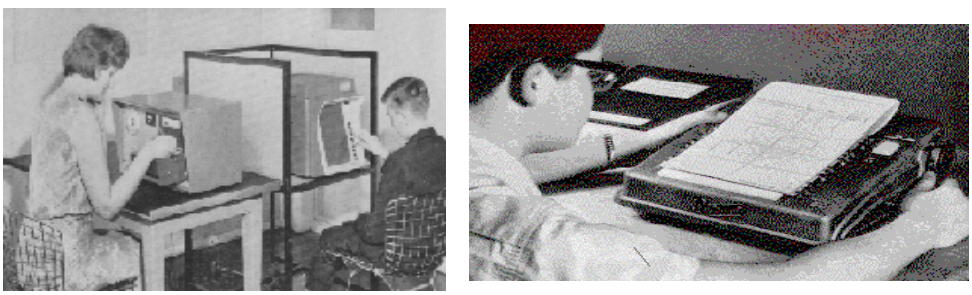


Figure 1 - Exemples de machines à enseigner

Cependant, la construction et l'utilisation de telle machine était très compliquée. Ainsi Bordeleau écrit au sujet de la machine de Crowder :

> *"C'est une machine très sophistiquée qui contient des rouleaux de films sur lesquels sont fixées des séquences d'instructions multiples. Des con-*

> *soles à boutons y sont reliées qui permettent aux élèves de répondre aux questions. Mais à chaque nouveau cours, il faut recharger la machine, opération complexe, s'il en est une, ce qui en limite sérieusement la facilité d'utilisation."*

L'informatique, les années aidant, va permettre de faciliter le développement des systèmes, et non plus des machines, d'enseignement. Ainsi, comme l'indique [Charlot 98], dans les années soixante apparaissent les premiers systèmes d'enseignement informatique. Ils consistaient tout simplement à présenter à l'apprenant une notion particulière et ensuite à l'interroger en lui posant différentes questions, soit sous forme de QCM[8], soit dans l'attente d'un résultat déterminé à l'avance, par exemple un résultat mathématique numérique. On nommait ces systèmes les systèmes "si..alors..sinon" (Cf. [Lelouche 87]), car leur algorithme était basé sur ce principe : Si l'apprenant répondait bien, le système faisait ceci, sinon il faisait cela. Les programmes les plus perfectionnés tiraient leurs informations de bases de données qui stockaient les cours et les exercices avec les solutions attendues. Certains pouvaient même sauvegarder le parcours de l'apprenant, c'est-à-dire conserver les différents résultats de ce dernier. L'enseignement assisté par ordinateur était né.

Dans ce chapitre, nous allons donc faire un tour d'horizon des différents types de systèmes d'enseignement assisté par ordinateur. Nous allons commencer par bien définir les sigles EAO et EAIO, afin qu'il n'y ait pas de confusion par la suite. Ensuite nous allons caractériser rapidement les systèmes qu'a produits la communauté scientifique jusqu'au milieu des années quatre-vingt-dix. Puis nous allons nous attarder plus longuement sur les différents systèmes qui sont apparus depuis quatre à cinq ans, ce qui nous permettra de mieux identifier les objectifs que nous nous sommes fixés.

1. Les sigles EAO et EIAO.

Avant de commencer notre étude, il est indispensable de bien définir les mots que nous allons rencontrer tout au long de ce document : les sigles EAO et EIAO.

En effet, la langue française aidant (c'est un peu plus clair en anglais), il est très facile de s'y perdre. Le E peut évoquer aussi bien le mot "enseignement" que le mot "environnement". Le A quant à lui peut signifier aussi bien "assister" que "apprentissage". Enfin le I peut exprimer aussi bien "l'intelligence" que "l'interactivité" du système.

En fait, nous pensons qu'il est important de ne pas confondre domaines de recherche, et catégories de système qui s'inscrivent dans chaque domaine de recherche.

Dès lors nous avons décidé de définir une taxonomie hiérarchisant les domaines de recherche et pour chaque domaine de recherche les catégories de système qui ont été produits (Cf. la figure 2).

Nous avons ainsi tenté d'énumérer tous les domaines de recherche en informatique qui ont comme problématique principale l'EAO pour Enseignement Assisté par Ordinateur. Nous en avons alors déterminé deux. Tout d'abord l'EIAO, pour Enseignement Intelligemment Assisté par Ordinateur. Ensuite l'EIAO, pour Environnement Interactif d'Apprentissage par ordinateur. Dès lors, nous avons estimé que ces deux domaines de recherche pouvaient être considérés comme des sous-domaines de l'EAO, car leur problématique principale est bien l'enseignement assisté par ordinateur, sachant que le premier s'appuie sur un comportement intelligent des systèmes, alors que le second s'intéresse plus particulièrement à l'apprentissage via l'interaction.

Ensuite pour chaque domaine de recherche, nous avons tenté de catégoriser les systèmes existants. Ce qui nous a par exemple conduit à considérer les Systèmes Tutoriels Intelligents comme s'intégrant parfaitement dans le domaine de recherche de l'Enseignement Intelligemment As-

[8] Questions à Choix Multiples

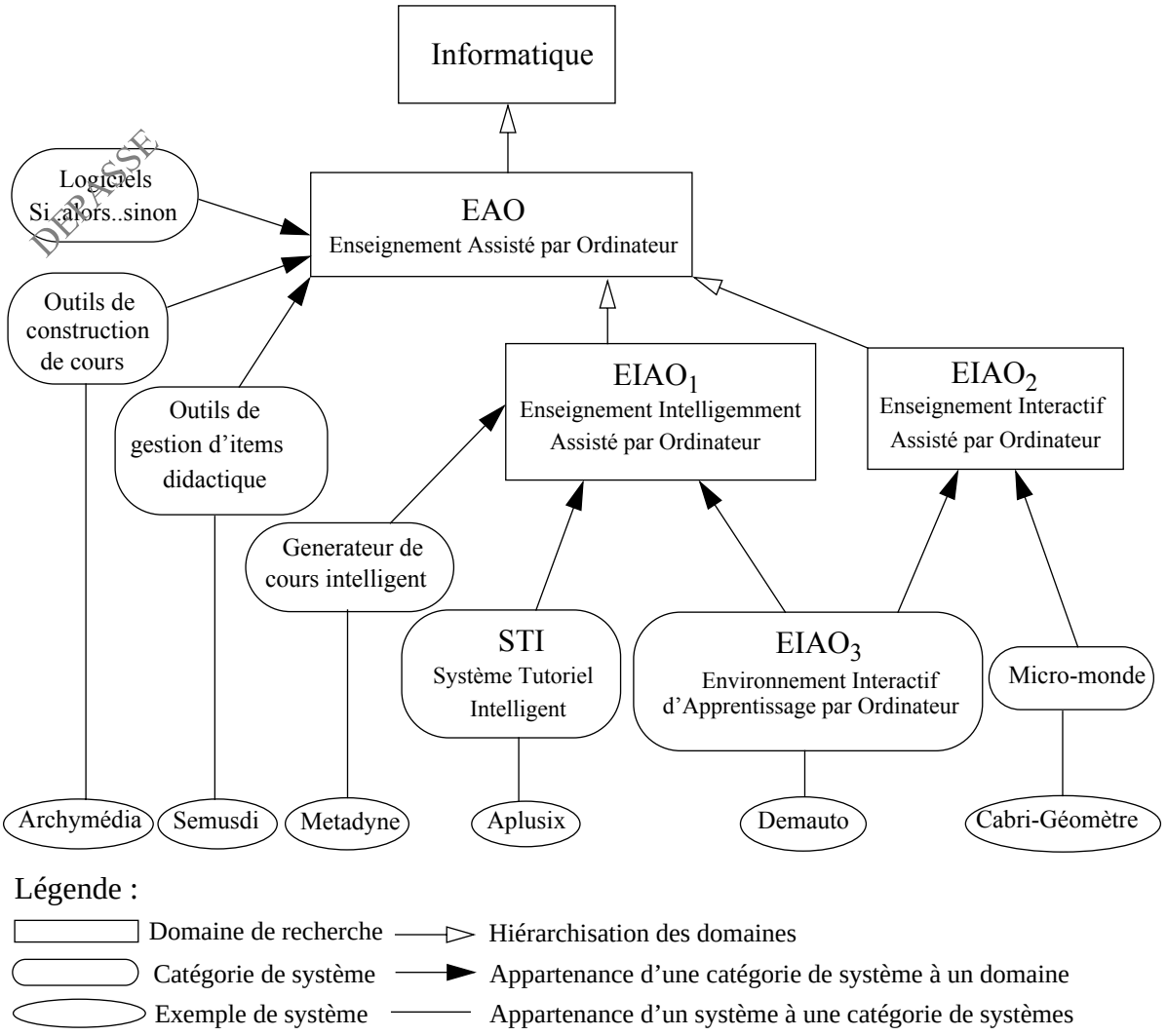


Figure 2 - Distinction entre domaines de recherche, catégories de système et systèmes

sisté par Ordinateur.

Remarques :

- La taxonomie présentée ici se limite volontairement aux problématiques informatiques. Or l'EAO dépasse largement le cadre de l'informatique, c'est un domaine de recherche pluridisciplinaire. On doit donc avoir à l'esprit que bon nombre de ces systèmes utilisent des résultats de recherche d'autres disciplines, telles que la didactique ou bien la psychologie.
- Afin de ne pas confondre les différentes significations du sigle EIAO, nous utiliserons les indices définis dans la figure 2 :
  $EIAO_1$ pour Enseignement Intelligemment Assisté par Ordinateur,
  $EIAO_2$ pour Enseignement Interactif Assisté par Ordinateur,
  $EIAO_3$ pour Environnement Interactif d'Apprentissage par Ordinateur.

Nous allons donc étudier ces différentes catégories de système en les présentant par ordre chronologique. Nous allons tout d'abord rappeler rapidement les caractéristiques des systèmes qui

sont apparus dans les années soixante-dix et les années quatre-vingt, c'est-à-dire les Systèmes Tutoriels Intelligents et les Micro-mondes. Ensuite nous allons caractériser les systèmes qui sont apparus aux débuts des années quatre-vingt-dix, c'est-à-dire les Environnements Interactifs d'Apprentissage. Enfin nous nous attarderons plus longuement sur les systèmes qui sont apparus depuis quelques années.

2. Les systèmes caractéristiques des années 70-80.

Durant ces deux décennies, deux types de systèmes d'enseignement ont vu le jours. Tout d'abord les systèmes tutoriels intelligents, puis les micro-mondes.

2.1. Les systèmes tutoriels intelligents (STI).

L'évolution de l'intelligence artificielle dans le domaine de la résolution de problèmes et de la représentation des connaissances a permis d'ajouter une certaine dose d'intelligence dans les systèmes d'enseignement. On a tenté alors de produire des systèmes qui pouvaient simuler un enseignant humain, en ajoutant au système des capacités de résolution (d'où l'adjectif "intelligent"), lui permettant ainsi de conseiller l'apprenant lorsque ce dernier commettait une erreur dans la résolution d'exercice (d'où l'adjectif tutoriel). Dès lors, comme l'indique [Baron 94], on peut définir un STI comme étant un système d'enseignement dont l'objectif pédagogique est de transmettre un savoir mais surtout un savoir-faire.

Il est généralement constitué des quatre modules suivants :

- Un modèle du domaine, qui permet au système de "raisonner", lui permettant alors de répondre aux questions de l'étudiant et de surveiller ce qu'il fait afin de le conseiller lorsqu'il commet des erreurs.
- Un modèle de l'apprenant, qui permet d'établir à un instant *t* l'état de ses connaissances.
- Un module pédagogique, qui, suivant le comportement de l'apprenant et le modèle de ce dernier, peut effectuer des choix d'enseignement.
- Un module d'interface qui transmet et décode les informations du système vers l'utilisateur et inversement.

Le logiciel SCHOLAR, présenté dans [Carbonell 70] est considéré comme l'ancêtre de tous ces systèmes. C'est en effet le premier à intégrer une représentation des connaissances à enseigner (connaissances liées à la géographie de l'Amérique du Sud) grâce à l'utilisation d'un réseau sémantique hiérarchisé, ce que l'on nomme aujourd'hui le modèle du domaine. Les noeuds de ce réseau représentent soit des concepts, par exemple un pays, soit des objets géographiques[9], par exemple l'Argentine (Cf. figure 3, [Wenger 87]). Ainsi à l'aide d'un ensemble de règles permettant d'exploiter ce réseau, le système est capable de répondre aux diverses questions de l'apprenant. La communication entre l'apprenant et le système s'effectue en langue naturelle (Cf. [Wenger 87]).

A côté du modèle du domaine, SCHOLAR intègre les résultats des étudiants. Cette composante, que l'on nomme aujourd'hui modèle de l'utilisateur, ou plus précisément modèle de l'apprenant dans le cas des systèmes éducatifs, utilise la technique dite de l'*overlay*, qui sera explicité dans le chapitre IV.

Par la suite des dizaines de STI ont été développés améliorant successivement chaque module. On peut citer entre autres Guidon I et Guidon II (Cf. [Clancey 79] et [Wenger 87]), et pour ce qui est de la France le système Aplusix (Cf. [Nicaud & al 99]).

[9] Ici, une instance géographique est considérée comme étant une instance d'un concept.

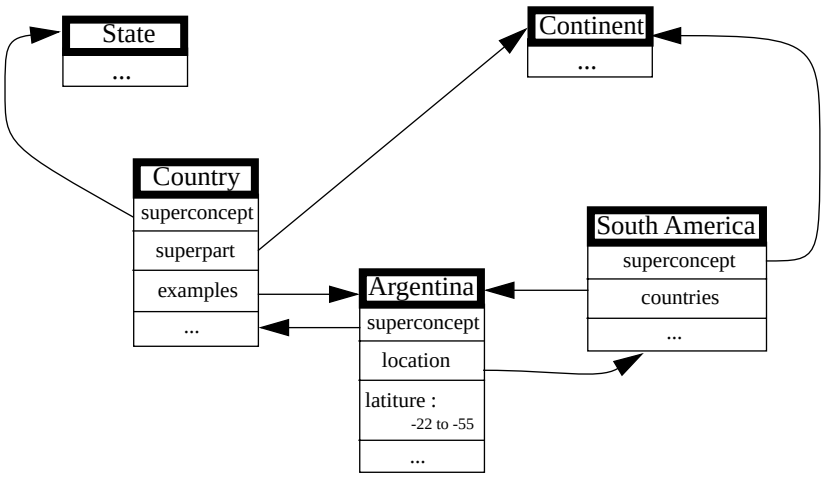


Figure 3 - Partie du réseau sémantique de SCHOLAR

2.2. Les micro-mondes.

A côté des théories béhavioristes de Skinner, un autre courant de pensée se développe après la seconde guerre mondiale : la théorie constructiviste. On doit cette théorie à Jean Piaget qui après un doctorat de biologie s'est intéressé au développement psychologique de l'enfant. Pour Piaget, l'intelligence permet à l'être humain de s'adapter continuellement à son environnement. Cette adaptation, que l'on appelle aussi apprentissage, est induite par les actions de l'homme sur le monde extérieur. Ainsi il ne peut y avoir apprentissage sans action, ou plus exactement sans interaction.

Issus de cette théorie, plusieurs systèmes informatiques ont vu le jour: les micro-mondes. On peut caractériser un micro-monde comme étant un logiciel éducatif dénué de toute connaissance, mais qui de par l'interaction qu'il offre avec l'utilisateur, permet à ce dernier d'assimiler et de comprendre plus facilement des connaissances.

On considère que le langage LOGO est le premier micro-monde qui ait été développé. La première version de ce langage date de 1966, mais ce n'est qu'en 1970 qu'apparaît la première version graphique, où la tortue (élément central du langage) est représentée à l'aide d'un petit triangle. En déplaçant cette tortue, l'enfant peut alors dessiner des figures géométriques. Par la suite, une véritable tortue robot sera utilisée.

Très prisé dans les écoles du monde entier pendant les années quatre-vingt, le langage Logo a par la suite été à l'origine de quelques polémiques (Cf. [Bruillard & al 87]). Aujourd'hui son utilisation est beaucoup plus confidentielle. Il est utilisé principalement comme langage d'initiation à la programmation ou comme support pour l'initiation à la robotique.

Remarque : Il existe un site Web et une liste de discussion permettant aux enseignants de s'échanger de programmes écrits en Logo (http://www.edu-web.be/res_logo/logo.htm)

Cabri-Géomètre, pour Cahier de brouillon informatique pour la géométrie, est aussi un micro-monde qui a connu, et connaît encore un franc succès. C'est Jean-Marie Laborde qui est à l'origine de la première version date de 1985. L'objectif initial de ce système est d'aider les élèves du collège à mieux appréhender les concepts enseignés en géométrie.

Son principe de fonctionnement est assez simple : l'enseignant seul, ou avec ses élèves, construit une figure géométrique en explicitant certaines propriétés (par exemple, un point I au mi-

lieu d'un segment [AB]). Une fois la construction achevée, les élèves peuvent manipuler cette figure, c'est-à-dire modifier la position des points, agrandir ou réduire la figure, etc. et ainsi remarquer que les propriétés mathématiques sont conservées. La figure 4 montre par exemple que quelque soit la forme du triangle, les droites particulières (médiane, médiatrices et bissectrices) conservent leurs propriétés.

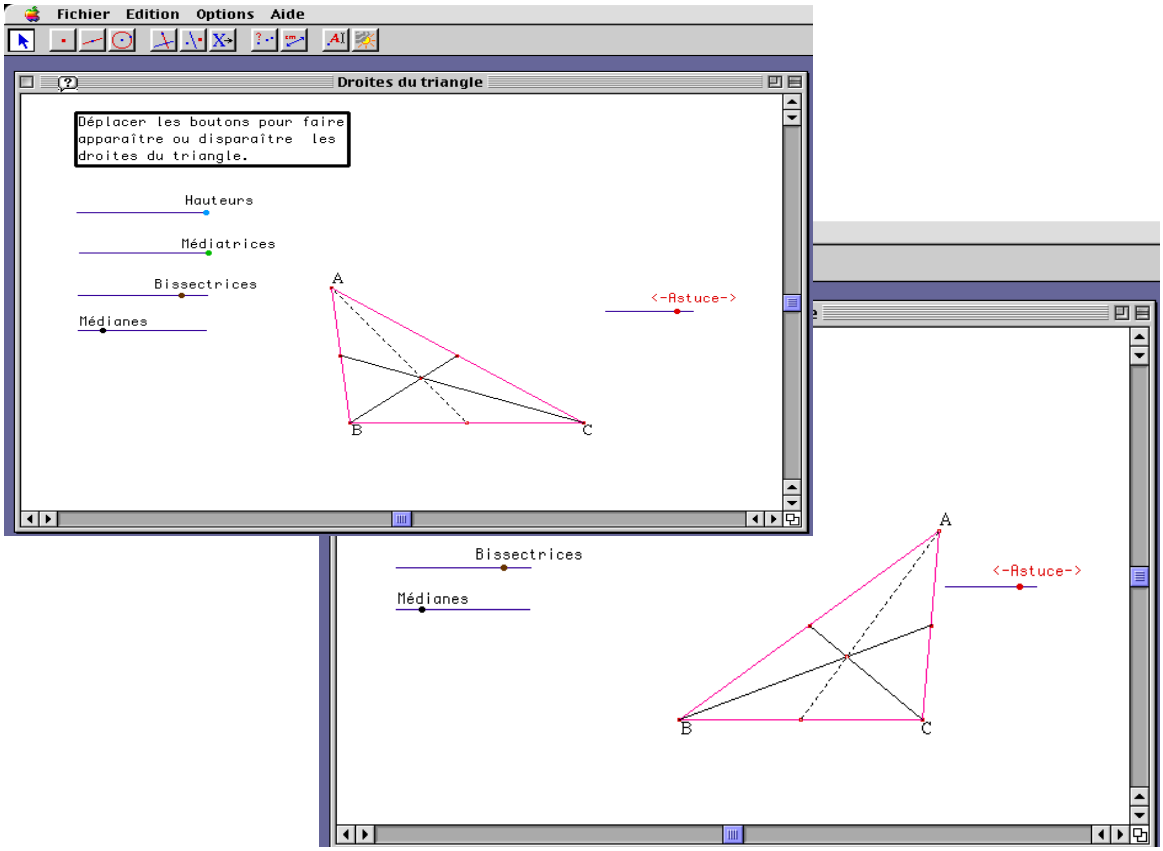


Figure 4 - Exemple de manipulation d'un triangle avec Cabri-Géomètre

En 1988, Apple attribue à l'équipe de J.M. Laborde son trophée éducation, ce qui les incitera à le commercialiser. Aujourd'hui, avec le concours de la société Texas Instrument, on peut trouver une deuxième version de ce système sur différentes plates-formes (Macintosh, Windows et même sur des calculatrices)[10]. De plus cette dernière version, de par ces nombreuses améliorations, ne s'attache plus seulement au domaine de la géométrie, mais à divers domaines scientifiques tels que par exemple la mécanique et l'optique. Enfin, une version en Java sous forme d'*applet* est à l'étude.

Remarques : • De par son aspect programmable, puisqu'il est en effet possible d'écrire des scripts, plusieurs logiciels éducatifs ont été développés au dessus de Cabri-Géomètre. On peut citer entre autres, Cabri-Euclide [Luengo 97] ou encore Cabri-Graphs [Carbonneaux & al 97] . Une conférence internationale est même prévue cette année sur Cabri-géomètre et ses applications (Octobre 1999)[11].

[10] Des versions d'évaluation peuvent être téléchargées sur : http://www-cabri.imag.fr/
[11] Pour plus d'informations vous pouvez consulter : http://www.cabri.com.br/

- Ce logiciel a fait un certain nombre d'émules, fonctionnant tous sur le même principe. On peut citer entre autres le *freeware* Declic disponible sur http://home.nordnet.fr/~eostenne/declic.htm.

3. Début des années 90, l'apogée des environnements interactifs d'apprentissage.

Comme l'indique la figure 2, les $EIAO_3$ sont des systèmes d'enseignement dont l'objectif est de prendre ce qu'il y a de meilleur dans les STI, c'est-à-dire les capacités du système à pouvoir raisonner sur un domaine spécifique, et de prendre aussi ce qu'il y a de meilleur dans les micro-mondes, c'est-à-dire l'apprentissage par l'interaction. Pierre Mendelsohn dans [Mendelsohn 91] définit entre autre les EIAO de la façon suivante :

> *Un ILE[12] est un système qui réalise la synthèse entre, d'une part, les avantages de l'exploration libre et de la construction progressive des objets de connaissance (comme dans les micro-mondes classiques) et, d'autre part, l'intérêt du guidage propre aux systèmes tutoriels. L'idée centrale est de permettre à l'apprenant de transformer rapidement et efficacement ses expériences en connaissances organisées.*

Le mot clef de cette définition est le mot synthèse, car pour qu'un système d'enseignement soit qualifié de $EIAO_3$, il ne suffit pas de proposer d'un côté un mode libre ou l'apprenant peut faire ce qu'il veut, et d'un autre côté un mode tutoré et dans ce cas très strict. Il faut en effet proposer un système dont le "comportement pédagogique" varie en fonction des actions et réactions de l'apprenant.

C'est par exemple ce qu'a essayé de réaliser Claude Moulin avec son $EIAO_3$ basé sur le système expert Demauto (Cf. [Moulin 98]). Ce système a été réalisé à l'aide d'un système multi-agents. Les quatre composantes que nous avons dans la définition s'un STI sont ici représentées par un ou plusieurs agents. Par exemple, le module pédagogique est constitué de deux catégories d'agents. Tout d'abord, les agents d'intentionnalité, qui au début de la session d'utilisation sont initialisés avec une certaine intention, et qui durant la phase d'exercice peuvent être modifiés. Ensuite, les agents d'interprétation globale des actions de l'élève, qui déterminent à un instant *t* l'état de l'apprenant (il avance vers la solution, il piétine, ses actions sont logiques avec les buts qu'il s'est fixé, etc.), il y autant d'agents que de situations. Ainsi par négociation, une interprétation est choisie, ce qui détermine le comportement pédagogique du système.

4. Les systèmes de la fin des années 90.

A partir de 1995, le développement des nouvelles technologies parallèlement aux capacités grandissantes des micro-ordinateurs permet de créer des systèmes d'enseignement ou le maître mot est "multimédia". Ce mot qui nous semble aujourd'hui naturel n'a pu exister que part :

- l'augmentation constante de la puissance de calcul des microprocesseurs dans les micro-ordinateurs : la démocratisation des processeurs pentium, ou compatible, pour les compatibles PC et le remplacement des processeurs motorola (série 68000) par les tous nouveaux PowerPC sur les ordinateur Macintosh.
- la taille de la mémoire vive qui commence à s'envoler (son coût de fabrication baisse constamment)
- les "extensions" multimédia, c'est-à-dire carte vidéo puissante, carte audio, lecteur de CD-Rom ne sont plus considérées justement comme des extensions, mais font

[12] ILE, pour Interactive Learning Environment, équivalent anglais des $EIAO_3$.

maintenant partis intégrantes d'un ordinateur (même de bas de gamme).

A cela s'accompagne le développement d'Internet, que nous étudierons plus en détail dans le chapitre VI.

Un virage s'amorce donc vers la volonté d'une dispense du savoir à grande échelle. Fini le petit système qui fonctionne sur lui-même (mono-poste) et qui scrute les faits et gestes de l'apprenant, les systèmes se veulent maintenant clients-serveur(s) avec dans leurs disques des centaines voire des milliers de cours, plus attractifs les uns que les autres (on voit apparaître une des caractéristiques de l'université, ou plus généralement de l'école, virtuelle).

On a alors créé des outils qui permettent de développer des hypermédia (mot qui sera définie dans le chapitre suivant) éducatifs. Ces outils peuvent être catégorisés de la façon suivante :

- Tout d'abord il faut des outils de construction de documents multimédia, permettant d'intégrer et de synchroniser au sein d'une même entité différents types de média (image, son, etc.)
- Ensuite il faut gérer ces documents, c'est-à-dire les indexer correctement afin de pouvoir les retrouver et les réutiliser par la suite.
- Enfin il faut pouvoir agencer ces documents, les relier les uns aux autres afin de construire des cours cohérents.

Ce sont ces différents types de système que nous allons maintenant examiner.

4.1. Les outils de production d'items didactiques.

Qu'est ce qu'un item didactique ?

Nous entendons par item didactique un document multimédia ou non (mais qui, dans notre problématique, devrait l'être dans la plus part des cas) qui possède intrinsèquement une qualité pédagogique, c'est-à-dire qu'il peut être utilisé dans le cadre de la transmission d'un savoir.

Dès lors, à la lecture de cette description, on peut penser que tout logiciel qui produit un document peut être utilisé pour produire des items didactiques. Si cela est vrai dans l'absolu, certains sont plus propices à ce genre de travail, tout particulièrement les logiciels auteurs, ainsi que les logiciels qui produisent des documents dans un des formats dits standards tel que le HTML.

Ainsi, par exemple, le logiciel Flash de la société Macromédia devient l'une des clefs de voûte incontournable pour la production d'items didactiques. En effet, outre le fait que l'on puisse maintenant trouver des *plugs-in* gratuits pour pratiquement toutes les plates-formes et pour les navigateurs les plus en vogue, c'est surtout les partenariats qui se montent entre les "différents grands de ce monde" telles que les sociétés Microsoft, Sun ou alors ici en l'occurrence Oracle, qui dictent les lois.

Quoiqu'il en soit, la difficulté de mise en oeuvre est l'obstacle principal de la production d'items didactiques. Tout d'abord du point de vue technique, il n'est pas donné à tout le monde de bien maîtriser les cinq ou six logiciels leaders. Ensuite leurs productions requièrent des compétences pluridisciplinaires qui sont pratiquement impossible de retrouver chez une unique et même personne, d'où l'intervention par exemple des informaticiens pour la partie programmation, des graphistes, et ne l'oublions des pédagogues.

Il paraît dès lors opportun de mettre en commun ces items didactiques et de les indexer proprement afin qu'ils puissent être retrouvés et réutilisés.

4.2. Les outils de gestions d'items didactiques.

Nous allons étudier trois systèmes de gestion d'items didactiques en commençant par le projet ARIADNE, puis le projet CDE et enfin le projet SEMUSDI. Pour chaque projet, nous allons

mettre en exergue les choix retenus pour :

- indexer ces items;
- pour stocker ou référencer ces items.

4.2.1. Le projet ARIADNE.

ARIADNE[13], pour *Alliance of Remote Instructional Authoring and Distribution Networks for Europe*, est un projet européen, regroupant principalement la Belgique, l'Espagne, la Finlande, la France, la Grande-Bretagne et la Suisse, et dont l'objectif est de définir des méthodologies et de produire des outils d'échange de données pédagogiques numériques pour les différents types d'enseignement (classique, continu, à distance, etc.).

Les outils créés dans ce consortium se veulent transversaux, en effet :

- Il y a tout d'abord ceux dont l'objectif est la création d'items didactiques, que l'on nomme les *Authoring tools*. Mais comme nous le remarquions précédemment ce travail est à notre avis du temps de perdu puisque les outils disponibles aujourd'hui sur le marché sont des plus puissants, et ont l'avantage d'être déjà très populaires.
- Il y a ensuite des outils de caractérisation des items didactiques.
- Il y a enfin des outils d'administration et de création de cours à partir de ces items didactiques, que nous allons voir brièvement par la suite.

4.2.1.1. La caractérisation des items didactiques.

Bien que peu spectaculaire à priori, la caractérisation des items didactiques est l'un des résultats les plus importants des travaux effectués au sein de ce consortium. Il en a résulté une norme, nommé *Learning Object Metadata*, qui permet d'indexer un document pédagogique à l'aide de huit catégories d'attributs :

1. Catégorie **Générale**, qui permet de spécifier les caractéristiques générales du document, tel que le nom, l'identifiant, la langue, etc.
2. Catégorie **Cycle de vie**, qui permet de spécifier les caractéristiques du cycle de vie, tel que le numéro de version, la date de création, la date d'expiration, etc.
3. Catégorie **Méta-données**, qui détermine la signification de tous ces attributs.
4. Catégorie **Technique**, qui regroupe l'ensemble des caractéristiques physiques du document, tel que le format, la taille, les *plug-ins* indispensables, etc.
5. Catégorie **Pédagogique**, qui détermine les spécificités pédagogiques du document tel que son type, son approche, ou encore sa granularité.
6. Catégorie **Gestion des droits**, qui détermine toutes les particularités juridiques du document.
7. Catégorie **Relation**, qui permet d'associer des documents entre eux.
8. Catégorie **Annotation**, qui permet d'ajouter des commentaires.

Ainsi chaque item didactique peut être précisément référencé, c'est-à-dire aussi bien au niveau de ses caractéristiques physiques, de ses caractéristiques pédagogiques, de son état courant (gestion du cycle de vie) qu'au niveau de sa diffusion (gestion des problèmes juridiques).

4.2.1.2. Le stockage des items didactiques.

Outre la caractérisation des items, c'est aussi grâce à son architecture véritablement répartie que ce projet est innovant. En effet, de par le caractère européen de ce projet, les ressources ne sont

[13] Vous pourrez trouver de plus amples informations sur le site Web officiel : http://ariadne.unil.ch/

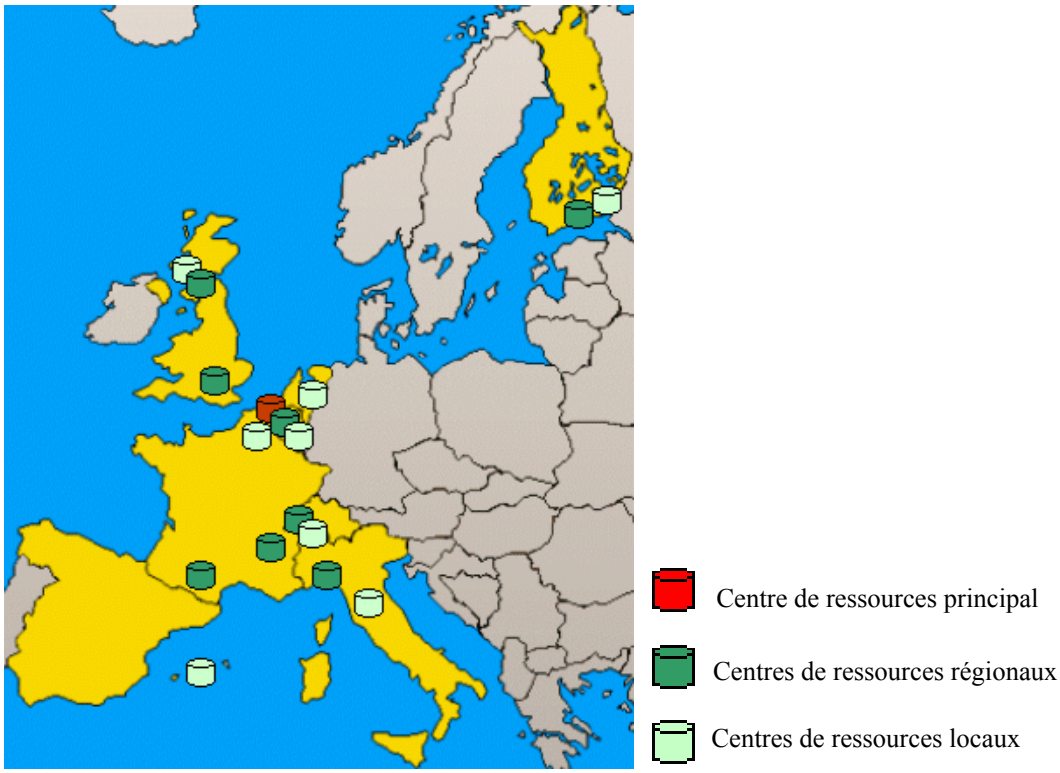


Figure 5 - Répartition des centres de ressources du projet ARIADNE

pas centralisées, mais au contraire réparties sur plusieurs sites, nommés "centre de ressources" (Cf. figure 5). Ces centres sont organisés hiérarchiquement: il y a un centre de ressources principal, qui est relié à des centres de ressources régionaux, eux-mêmes reliés à des centres de ressources locaux.

Avec au départ de grandes ambitions, ce projet est resté et reste à notre connaissance un projet, et n'est donc pas très utilisé pour l'échange de documents pédagogiques. De plus, à l'heure actuelle, le financement européen terminé, on ne sait pas si ce projet va être renouvelé.

4.2.2. Le projet CDE.

Le projet CDE, pour *Course Designer Environment*, de l'Ecole Polytechnique Fédérale de Lausanne, s'inscrit dans le grand projet de l'EPFL nommé *ClassRoom2000*[14] dont le but est de développer des environnements auteurs pour la production de cours interactifs avec un haut niveau de partage et de réutilisation d'items didactiques.

CDE est, comme ARIDANE, un projet transversal puisque outre la problématique de partage d'items didactiques, il propose aussi un outils nommé *CoDes Authoring Tool* pour *Course Designer Authoring Tool* permettant aux enseignants de créer, structurer et visualiser des cours interactifs (Cf. [Rekik & al 99]), outils que l'on ne traitera pas ici. Par contre l'architecture de cet outil de gestion et de partage d'items didactiques est déjà définie et en plein développement.

Cet outil se distingue des autres systèmes par le fait qu'il ne stocke pas ces items, il les référence seulement. Cette caractérisation s'effectue à l'aide du langage XML[15]. La DTD[16] choisie permet de caractériser un item suivant quatre axes :

[14] Voir http://www.ndit.ch/5000/
[15] eXtend Markup Language
[16] Déclaration du Type de Document

- Les attributs physiques, tels que le type *mime*, la taille, etc.
- Les attributs logiques, tels que le *copyright*, la version, l'auteur, etc.
- Les attributs structurels, tels que le contexte, ou les liens.
- Les attributs sémantiques, tels que la description, les mots-clefs, le titre, etc.

L'obtention ou l'ajout d'items peuvent alors être effectués suivant deux protocoles. Soit le client est un logiciel propriétaire, dans ce cas il doit inclure un composant logiciel lui permettant de communiquer avec le serveur. Soit le client est un enseignant, qui a par exemple besoin d'items pour illustrer un cours, et dans ce cas il effectue sa requête à l'aide de son navigateur Internet.

4.2.3. Le système OLA.

Le système OLA, pour *Online Learning Application*, était, car il semble aux dernières nouvelles que le projet ait été abandonné, un système propriétaire de la société Oracle.

Toutefois, il est intéressant de ne pas passer ce système sous silence car OLA avait la particularité d'utiliser la norme RCOS[17], pour *Reusable Content Object Strategy*. Comme le montre la figure 6 extraite de [Hataway 97], la norme RCOS définit quatre niveaux hiérarchiques de document. Le niveau le plus bas est composé de document mono-média, dont les caractéristiques didactiques ne sont pas spécifiées. Ensuite, il y a le niveau des items didactiques simples (nommés RCO), c'est, comme nous allons le voir, ce que l'on nomme une brique élémentaire. Enfin il y a deux niveaux supérieurs, dont les documents pédagogiques sont la résultante d'une agrégation spécifique de RCO.

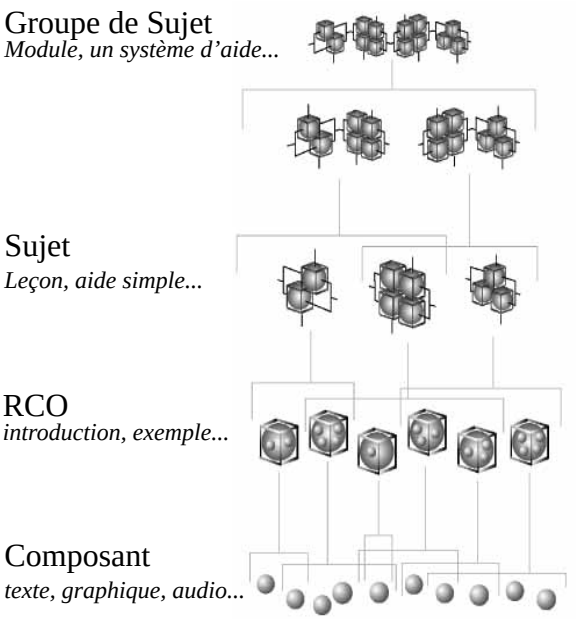


Figure 6 - Décomposition hiérarchique du modèle RCO

Tous les éléments de la norme RCOS possèdent des attributs communs, dont les objectifs sont l'aide à la recherche (tel que le nom, la liste de mots clefs, etc.) et l'aide à la gestion (la version, la date de création et d'expiration, etc.). Ensuite chaque catégorie de documents possède ses propres attributs. Par exemple, comme le montre le tableau 1, une hiérarchie pour la catégorie "composant" a été élaborée permettant ainsi de caractériser pleinement le document; par exemple les vidéos posséderont un attribut durée qui n'a pas lieu d'être pour les textes. De même les

[17] Voir http://www.luminare.com/RCO/RCO/home/bumper1.htm

éléments de la catégorie RCO sont caractérisés par trois catégories d'attributs (les attributs de stratégie de sélection, les attributs de lien, et les attributs d'utilisation).

Media Visuel	Texte	Mots, phrases, paragraphes, listes, tableaux
	Images fixes	Haute définition (photo, capture d'écran) Basse définition (graphique, diagramme)
	Images animées	Haute définition (vidéo) Basse définition (animation)
Média auditif	Langue	Anglais, Français, etc.
	Son	Musique Tonalité

Tableau 1 - Caractérisation des composants dans le modèle RCO

4.2.4. Le projet SEMUSDI.

Ce dernier projet est un projet de l'INSA de Rouen (Cf. [Aziz & al 99a]). Il a débuté en 1996 suite aux constats suivants :

- il existe dans nos écoles d'ingénieurs ou universités un volume important d'informations souvent peu exploitées, parce que peu connues ou difficilement accessibles,
- la notion de document brut, élémentaire mais bien ciblé constitue un support de base qui intéresse fortement les enseignants, les étudiants, et les producteurs de systèmes d'enseignement assisté par ordinateur.

Ces constats nous ont conduits à définir un système capable de regrouper ces informations en les organisant pour les rendre facilement accessibles. De plus nous avons profité de l'apport des nouvelles technologies pour définir un serveur de documents multimédia accessible par le réseau Internet. La notion de documents multimédia est en effet essentielle dans un contexte de formation, elle permet de travailler sur des images, d'effectuer des manipulations sur des objets réels ou virtuels, de gérer l'interactivité.

Le serveur Semusdi est donc conçu pour collecter, gérer, stocker, rechercher, consulter et diffuser des items didactiques ciblés, pour des utilisateurs répartis sur plusieurs sites géographiques distants. Pour bien insister sur l'aspect réutilisable et réorganisable de ces items, nous les nommons briques élémentaires.

Ces briques élémentaires peuvent être de différentes formes :

- texte et hypertexte (HTML),
- images fixes ou animées,
- son,
- logiciel d'animation et de simulation,
- exercices d'application courts,
- exemples de cas concrets.

L'ensemble de ces briques est géré au sein d'une base de données accessible via le Web, la figure 7 présente les fonctionnalités de ce serveur. Des fonctions d'administration du système sont également disponibles, mais elles ne sont accessibles que par un nombre très restreint de personnes identifiées. Les fonctionalités du serveur, ainsi que son architecture actuelle et future sont étudiées en annexe.

La fiche d'identification d'une brique élémentaire dans SEMUSDI est beaucoup moins complè-

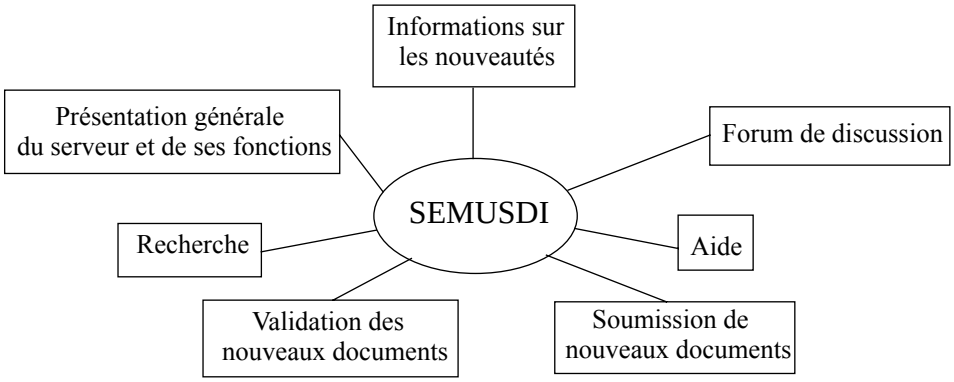


Figure 7 - Principales fonctions de SEMUSDI

te que les différentes normalisations que l'on vient de voir. Tout d'abord de par son aspect francophone, il n'y pas d'information au sujet de la langue. Ensuite, la gestion des problèmes juridiques a été écartée, car on considère que les personnes qui ont accès au site peuvent faire ce qu'ils veulent des documents une fois récupérés (ils sont libres de droit). Enfin, on considère que deux versions d'une même brique correspondent à deux briques distinctes, ce qui résout les problèmes de cycles de vie.

En somme, les briques dans SEMUSDI possèdent quatre types informations :

- Informations générales, tel que son nom, un texte de présentation et des mots clefs.
- Informations au sujet de l'auteur, tel que son nom et prénom et ses coordonnées.
- Informations pédagogiques, tel que le thème et le type (présentation, application, etc.).
- Informations techniques, tel que le format de la brique, et s'il y a besoin, le *plug-in* indispensable pour la visualiser.

On le voit bien, les briques dans SEMUSDI sont moins caractérisées, mais le travail de validation et d'indexation en est grandement facilité, voire beaucoup plus réaliste que dans ARIADNE.

4.3. Les outils de réutilisation d'items didactiques.

Enfin, ce dernier maillon a pour objectif la réorganisation de ces items didactiques afin de construire ces fameux cours multimédia.

De nombreux projets universitaires et industriels se sont intéressés à cette problématique, nous allons donc en voir quelques-uns qui sont à notre avis représentatif de ce qui existe.

4.3.1. Le projet Archymédia.

Le projet Archymédia, pour architecture d'un système hypermédia d'aide à la formation, est la concrétisation du travail de thèse de Christophe Nécaille au sein de notre laboratoire (Thèse soutenue en 1998). Archymédia est un système qui permet à des enseignants non-informaticiens de construire des cours hypermédia. Ce système se veut être à la fois un système ouvert et un système fermé, ce qui peut sembler contradictoire au premier abord, mais qui s'explique très simplement.

En effet Archymédia est un système ouvert car il utilise Internet comme support de communication et le langage HTML comme outils de présentation des items didactiques, tous les acteurs du système n'ont besoin que d'un simple navigateur pour l'utiliser.

Mais, il est par contre fermé car tous les outils d'échange d'information entre les acteurs du sys-

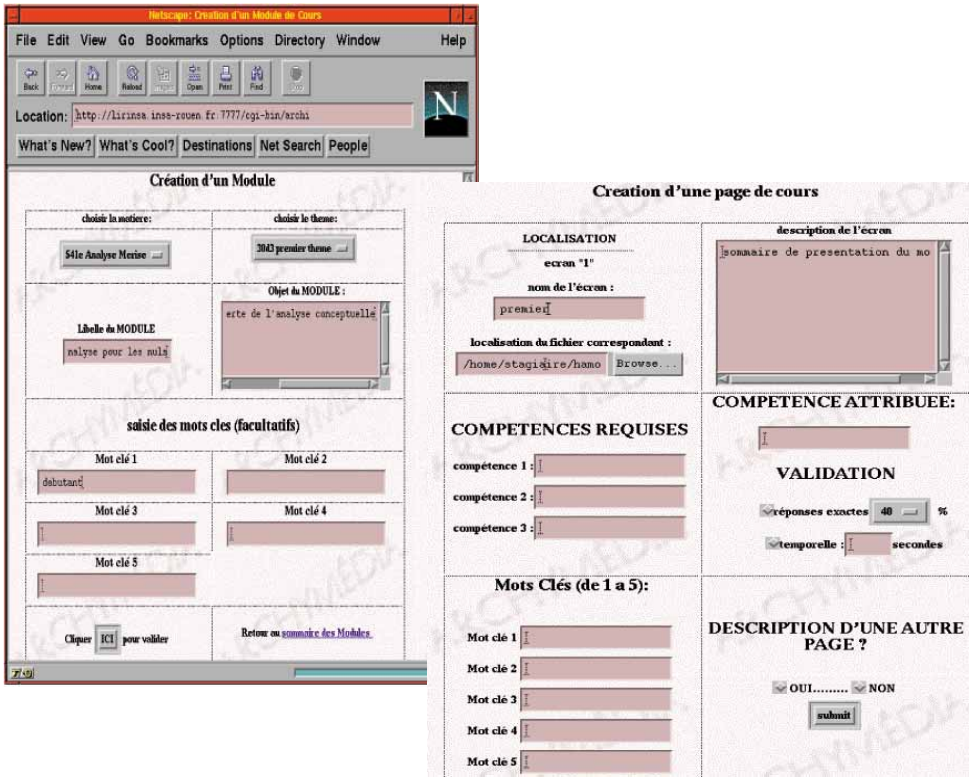


Figure 8 - Edition des modules et des pages hypermédia sous Archymédia

tème tel que les forums de discussion et le courrier électronique, pour améliorer la faciliter d'utilisation, sont internes au système, et donc non standards.

Les enseignants dans ce système sont en charge de créer les cours, nommé dans le système module. Pour ce faire, à l'aide des items didactiques précédemment stockés au sein de la base, ils construisent les différentes pages hypermédia qui vont constituer ce module, qu'ils vont alors pourvoir relier les unes aux autres.

A chaque page est associée une compétence que l'apprenant acquière lorsqu'il la parcourt. Ces compétences sont ensuite utilisées pour savoir si les liens définis par l'enseignant, entre les pages, doivent être affichées au sein du navigateur de l'apprenant.

Les apprenants de leur côté peuvent visualiser les cours construits par les enseignants, peuvent communiquer entre eux, ou bien avec les enseignants.

4.3.2. Le projet ARIADNE.

Nous n'allons pas répéter ce que l'on déjà vu précédemment. Mais il ne faut pas oublier que le projet ARIADNE propose des outils de construction de cours pour les enseignants ainsi que des outils de visualisation de ces dits cours (Cf. figure 9).

4.3.3. Le système Learning Space.

Le système *Learning Space* est un produit vendu par IBM (appartenant précédemment à la société Lotus) qui est basé sur l'utilisation de la base de document Lotus Notes ainsi que sur le serveur Web Domino. Il tire donc tous ces avantages de ces deux programmes, le premier permettant de stocker, rechercher et visualiser à peu près tous les types de documents (son éventail de type de fichier reconnu est très vaste), et le second permet de publier ces documents sur le Web (il produit alors des pages HTML à la volée avec pas ou peu d'*applet* Java, donc très légère). le tout bien évidemment sans aucune ligne de programmation, ce qui le destine à un large

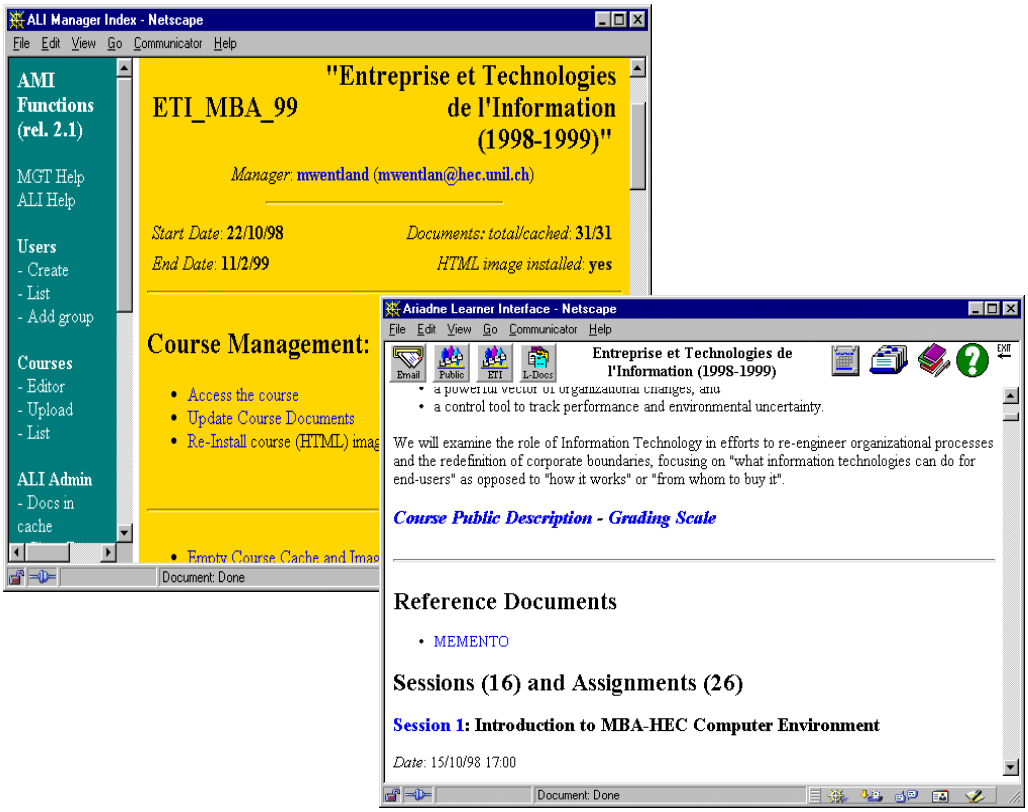


Figure 9 - Création de cours et visualisation de ces cours avec ARIADNE

public.

4.3.4. Le projet MEDIT.

Le projet MEDIT[18] pour *Multimedia Environment for Distributed Interactive Teaching* de l'EPFL est l'un des plus avancé dans le domaine de la distribution de cours interactifs sur le Web.

Comme d'habitude, il existe trois types d'acteurs dans MEDIT: les enseignants, les étudiants et les administrateurs du système.

Les enseignants ont la possibilité de créer des cours. Cette création a toutefois la particularité de pouvoir s'effectuer suivant plusieurs modes de présentation:

- Tout d'abord le mode le plus conventionnel, c'est-à-dire organisé sous forme linéaire. Dans ce cas, l'auteur définie les chapitres et sous-chapitres de son cours, en insérant alors l'item didactique pour chaque partie du cours. On peut remarquer dans ce cas que la taille de l'item didactique peut être assez grande, puisqu'un item didactique peut correspondre à un chapitre de cours.
- Ensuite, l'enseignant peut choisir un mode chronologique, c'est-à-dire que dans ce cas le cours n'est pas entièrement accessible. Le nombre de sessions visualisables dépend alors de la date d'utilisation.
- Enfin l'enseignant peut choisir d'offrir à l'étudiant une découverte du cours sous forme thématique. Dans ce cas l'enseignant doit structurer son cours sous la forme d'un arbre n-aire.

[18] Pour plus d'information vous pouvez aller voir l'excellent site Web dédié à MEDIT : http://medit.epfl.ch:4444

A côté de cette construction de cours, l'enseignant peut construire des exercices, qu'il soit autonome, comme c'est le cas par exemple avec les questionnaires à choix multiples, ou pas et dans ce dernier cas, l'apprenant devra envoyer à l'enseignant un document électronique.

L'apprenant quant à lui n'est pas un apprenant obligatoirement passif, puisque différents outils lui sont proposés. Il a entre autres la possibilité de poser des questions, de prendre des notes ou bien d'échanger des informations avec les autres étudiants. Il a de plus la possibilité de construire ses propres cours, en personnalisant les cours qui lui sont proposés (Cf. la figure 10 issue de [Abou Khaled & al 98]).

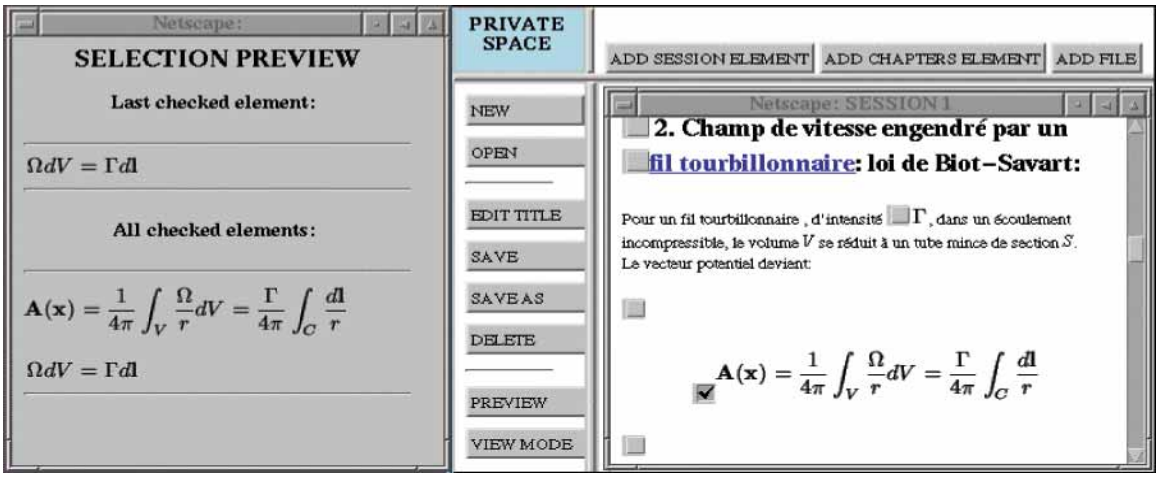


Figure 10 - MEDIT - Un apprenant qui construit son propre cours.

4.3.5. Le système WebCT.

Enfin le projet Web-CT[19] a au départ été développé par le département d'informatique de l'université British Columbia à Vancouver et est maintenant un produit à part entière. C'est un outil qui facilite la création d'environnements éducatifs sur le WWW. Il est utilisé pour créer des cours complets en ligne ou simplement pour publier des matériaux qui complètent des cours existants (Cf. [Murray & al 96] et [Murray & al 97]).

WebCT doit sa grande popularité au fait qu'il soit très simple à utiliser via un classique navigateur Web. A contrario, les outils informatiques mis en oeuvre sont des plus basiques ce qui lui confère tout d'abord une interface d'utilisation assez austère et une redondance de l'information très importance (il n'y a pas de base de données).

5. Conclusion.

Dans ce chapitre nous avons rapidement étudié l'évolution des systèmes d'enseignement assisté par ordinateur durant les trente dernières années. Alors qu'au début on s'est efforcé à produire des systèmes les plus intelligents possible, depuis quelques années c'est la production de cours multimédia en-ligne à grande échelle (donc standard) qui est le centre d'intérêt de beaucoup de recherches.

Cette dualité s'accentue en plus par le fait que pendant près de vingt ans on a essayé de produire des systèmes dont l'objectif pédagogique était d'améliorer le savoir-faire des apprenants, en les mettant dans des situations de résolution de problème, alors que la majorité des cours dispensés sur le web ont pour objectif surtout la transmission d'un savoir, ou sont du moins dans l'incapacité de vérifier les acquis en matière de savoir faire de l'apprenant (si l'on fait exception des

[19] Voir http://www.webct.com/

STI qui ont une ouverture sur le Web, comme par exemple les STI qui peuvent être écrit à l'aide du système SAFARI[20][Frasson & al 96]).

Et c'est sur cette transmission du savoir que le bât blesse car cette dernière est lourde à concevoir et surtout très figée:

- Lourde, car la construction d'un cours à l'aide des systèmes que l'on vient de voir, fait directement intervenir ce qui va être utilisé pour le présenter. Les enseignants avec ces systèmes construisent leur cours directement à l'aide des items didactiques, il n'y a pas de distinction entre le fond, c'est-à-dire les notions qui sont mises en jeu dans cet apprentissage, et la forme, c'est-à-dire les items qui vont être utilisés pour présenter ces notions.
- Figé car quelque soit l'apprenant qui se trouve devant l'écran, le cours est identique.

Il serait par conséquent intéressant d'étudier l'apport que pourrait avoir les techniques d'intelligence artificielle pour des systèmes prioritairement axés sur le transfert du savoir.

Or aujourd'hui, le Web, l'un des services du réseau Internet comme nous le verrons par la suite, est l'une des clefs de voûte des Nouvelles Technologies de l'Information et de la Communication, dont l'acquisition de l'information est basée sur le principe de l'hypertexte.

Mais, qu'est ce que l'hypertexte ?

Cette catégorie de système peut-il tout d'abord transmettre un savoir ?

Et si, oui, peut-il le faire de façon intelligente ?

C'est ce que nous allons aborder maintenant.

[20] Pour Système d'Aide à la Formation par l'Analyse de Raisonnement Interactif, une présentation de ce projet peut être trouvé sur : http://www.iro.umontreal.ca/labs/HERON/SafariWeb/welcomesafari.htm

Chapitre II - Les hypermédia.

Dans ce chapitre nous allons étudier les hypermédia au sens large. Nous allons commencer par préciser ce qu'est un hypermédia et quelles sont ses origines.

Puis nous allons étudier successivement deux types d'hypermédia particuliers : les hypermédia adaptatifs et les hypermédia adaptatifs dynamiques.

Enfin nous terminerons ce chapitre, en étudiant les avantages et inconvénients de l'utilisation de chaque type d'hypermédia dans un cadre éducatif.

1. Qu'est-ce qu'un hypermédia ?

1.1. Un peu d'histoire...

Tout le monde s'accorde à dire que le "père" des hypermédia est Vannevar Bush. Il a en effet écrit un article en juillet 1945, nommé "As We May Think" [Bush 45] où il décrit un système futuriste d'archivage de l'information. En fait ces études à l'issue de la seconde guerre mondiale ont pour objectif de réduire la perte d'informations et/ou de connaissances. Ainsi écrit-il :

> *"Mendel's concept of the laws of genetics was lost to the world for a generation because his publications did not reach the few who were capable of grasping and extending it."*

Il essaye alors de définir les fonctionnalités d'une machine-bureau, nommé *Memex*, qui permettrait à quiconque de sauvegarder des milliers d'informations sous forme de microfilm (Cf. figure 11[21]) :

> *"A memex is a device in which individual stores all his books, records and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory."*

Il remarque toutefois, qu'avoir des tonnes d'informations, c'est bien, mais savoir rechercher et retrouver l'information désirée, c'est mieux. Il définit alors un mécanisme permettant d'indexer

[21] Cette illustration est issue d'une présentation multimédia de Memex, que l'on peut trouver sur le site : http://sloan.stanford.edu/mousesite/Secondary.html

Figure 11 - Memex

les documents de manière assez classique (par mots-clefs), mais surtout il invente un nouveau mécanisme qui permet d'associer les documents, permettant à l'utilisateur de naviguer de document en document :

> *"It affords an immediate step, however, to associative indexing, the basic idea of which is a provision wehereby any item may be caused at will to select immediatly and automatically another. This is the essential feature of the memex."*

Le principe des systèmes hypertextes était né. Malheureusement, Bush n'a jamais eu la possibilité ni les moyens de construire un seul *Memex*.

Toutefois, on ne retrouve pas dans cet article le terme hypertexte, puisque ce mot à été inventé un peu plus tard, en 1965, par Ted Nelson dans [Nelson 65]. Il définit alors un hypertexte comme étant :

> *"a body of written or pictorial material interconnected in a complex way that it could not be conveniently represented on a paper. It may contain summaries or map of its contents and their interrelations; it may contain annotations, additions and footnotes from scholars who have examined it."*

Ce n'est cependant que trois ans plus tard qu'apparaît le premier système hypertexte. Ce système nommé NLS, pour *oN Line System*, fût conçu par Douglas Englebart[22]. L'objectif de ce système était de permettre à l'utilisateur de pouvoir archiver toutes sortes de documents et de les relier par des liens associatifs (on retrouve les idées de Bush). Ce système est l'aboutissement d'une dizaine d'années de recherche autour du projet *Augmentation System*, qui devait permettre aux utilisateurs d'accroître leurs capacités cognitives et intellectuelles [Englebart 62].

Autour de ce projet des dizaines de concepts ont été inventés permettant d'accroître la facilité

[22] Un site Web est consacré aux travaux de D. Englebart : http://www.bootstrap.org/history.htm

d'utilisation de ce système. On peut citer entre autres l'invention de la souris, de l'interface graphique fenêtrée (concept de fenêtre et d'environnement multi-fenêtré) ou encore du courrier électronique.

Dès lors, plusieurs systèmes utilisant les techniques hypertextes vont apparaître sur le marché. On peut citer entre autres :

- le logiciel Dynabook, conçu et développé par Alan Kay de la société Xerox à Palo Alto. Ce logiciel, basé sur une interface graphique qui préfigurera l'interface du Macintosh, devait permettre aux utilisateurs de pouvoir sauvegarder tout type d'information (texte, son, image, etc.). On peut noter que c'est grâce à ce projet que fût inventé le langage Smalltalk.
- Bill Atkinson, figure emblématique de la société Apple, a conçu en 1986 le premier logiciel permettant à "monsieur tout le monde" de pouvoir créer des hypermédia : Hypercard était né. Comme le souligne [Rhéaume 93] :

  "[..] ce logiciel n'était pas concu spécifiquement pour construire des hypertextes. Pourtant sa distribution gratuite et sa facilité d'utilisation tend à populariser les hypertextes."

Enfin, aujourd'hui nous avons à notre disposition le plus grand hypermédia jamais conçu : le World Wide Web, la composante d'Internet la plus connue, que nous verrons plus en détail dans le chapitre VI.

1.2. Comment définir les termes hypertexte et hypermédia ?

Les termes hypertextes et hypermédia[23] peuvent être définis suivant trois points de vue. On peut en effet les définir du point de vue de la structure (ce que nous appelons la définition structurelle), du point de vue de l'interaction entre l'utilisateur et le système (ce que nous appelons la définition fonctionnelle), ou bien du point de la sémantique du terme. Nous allons donc voir ces trois points de vue, puis nous ferons une présentation des quelques modèles formels définis.

1.2.1. Définition structurelle.

[Balasubramanian 94] définit un hypertexte comme étant un système composé de noeuds et de liens.

Les noeuds peuvent être composés d'informations textuelles, on parle alors d'hypertexte, ou multimédia, tels que des images, des graphiques, des animations des vidéos ou bien des programmes informatiques, on parle alors d'hypermédia. Les noeuds sont reliés les uns aux autres par des liens. On distingue les noeuds qui sont à l'origine du lien (on parle de référence) et les noeuds qui sont les destinations des liens (on parle de référent).

Les liens peuvent être plus ou moins complexes : ils peuvent être unidirectionnels, permettant d'aller d'une page à une autre page, ou bidirectionnels, afin de faciliter le retour au point de départ. Ils peuvent aussi être typés afin de spécifier la sémantique du lien. Enfin les liens peuvent être disposer n'importe où dans une page. Toutefois leurs rôles peuvent de temps en temps être définis d'une part de par leur position dans le document, d'autre part de par la sémantique de la page (par exemple si la page est une page d'index, les liens seront des index).

Ce sont ces liens qui définissent l'architecture du système, que l'on nomme hyper-espace.

1.2.2. Définition fonctionnelle.

L'hypertexte peut être considéré comme étant un procédé informatique permettant d'associer une entité (souvent minimale, c'est-à-dire un mot, une - portion d'- image ou une icône) à une

[23] A l'issue de ces définitions, nous utiliserons indifféremment les termes hypertextes et hypermédia.

autre entité (souvent plus étendue comme un paragraphe, une image ou une page).

Ce mécanisme permet donc à l'utilisateur de se diriger librement dans l'hypertexte. En activant, à l'aide d'un pointeur (une souris, un stylet ou une tablette sensitive), une zone du document qui est l'origine d'une association, l'utilisateur peut immédiatement atteindre une autre partie du document. Il n'est donc plus obligé de suivre le cheminement prévu par l'auteur, il définit son parcours en fonction de ses envies et de ses centres d'intérêt. Ainsi [Rhénaume 93] écrit:

> *"L'hypertexte est par conséquent un document virtuel - qui n'est jamais globalement perceptible - dont l'actualisation d'une des potentialités est conditionnée par l'effectivité de la lecture. Cette propriété de l'hypertexte en fait un document "interactif" dans lequel le lecteur tient une place prépondérante."*

1.2.3. Définition sémantique.

Comme l'indique [Nanard 95], épistologiquement le mot hypertexte signifie "plus que texte". Le mot "plus" ne signifie pas plusieurs textes interconnectés, mais une entité qui est composée de deux entités :

- un ensemble de documents,
- une connaissance.

La représentation la plus pauvre de cette deuxième entité est le lien inscrit à l'intérieur même de la première entité, c'est ce que M. Nanard nomme le lien "en dur", qui relie deux documents. A contrario, la forme la plus élaborée de cette connaissance peut être générée par un système complexe se basant sur une modélisation du domaine et sur une modélisation de l'utilisateur[24].

1.2.4. Modèles formels.

Dans les années quatre-vingt-dix, la multiplication des programmes implémentant des caractéristiques hypertextuelles incita quelques chercheurs à formaliser ces caractéristiques afin de pouvoir améliorer la portabilité des systèmes, et donc obtenir une standardisation. Parmi ces modèles formels ont peut retenir entre autres [Langue 90], qui définit toutes les composantes d'un hypertexte à l'aide d'une grammaire. Cette analyse l'a ammené à définir un modèle objet implémentable qu'il a nommé Hyperbase, ainsi qu'une méthodologie de développement d'hypermédia [Langue 96]. Il n'est toutefois pas le seul à avoir suivi cette démarche, il y a aussi par exemple le modèle présenté dans [Maurer & al 93] et [Maurer & al 94], qui se démarque des autres modèles par l'introduction d'un nouveau concept nommé S-collection.

Ayant maintenant précisé les mots hypertexte et hypermédia. Nous allons maintenant nous interesser à deux types d'hypermédia particuliers: les hypermédia adaptatifs et les hypermédia adaptatifs dynamiques.

2. Les hypermédia adaptatifs.

Les hypermédia sont devenus depuis un peu moins de dix ans des systèmes très utilisés pour obtenir de l'information. Comme nous l'avons vu dans le premier chapitre, on peut les utiliser dans un cadre éducatif, mais on peut aussi les utiliser dans des systèmes d'information en ligne (par exemple le système de gestion de documents Lotus Notes de la société IBM permet de naviguer de document en document en activant des liens) ou encore dans les systèmes d'aide (tous nos logiciels aujourd'hui possèdent une aide en ligne basée sur ce principe).

Dès lors il peut être intérssant de posséder un système qui s'adapte à l'utilisateur. Cette adaptation peut être très intéressante, et ce quelque soit le type d'hypermédia. Par exemple, dans un hypermédia éducatif, ceci va permettre de mieux guider l'apprenant. Il peut en être de même

[24] Les hypermédia adaptatifs dynamiques, que l'on va voir par la suite, sont basés sur ce principe.

dans un système d'aide. Mais, cela peut aussi être très utile dans un système d'information, pour par exemple limiter l'accès de certains documents à certaines personnes.

2.1. L'adaptation.

Nous avons vu qu'un hypermédia est composé de pages et de liens. Dès lors, un hypermédia adaptatif doit pouvoir adapter le contenu de ces pages et de ces liens pour mieux guider l'utilisateur dans son cheminement.

2.1.1. L'adaptation du contenu.

L'objectif est d'adapter le contenu des pages de l'hypermédia en fonction des caractéristiques, des volontés et des buts de l'utilisateur. Ainsi, les utilisateurs qui accèdent à une "même page", mais en ayant des profils différents, doivent visualiser en fait des pages différentes. Pour l'instant, très peu de systèmes effectuent une adaptation du contenu, et lorsqu'ils le font, l'adaptation n'a souvent lieu qu'au niveau des données textuelles, comme dans [Brusilovsky 92] ou [Cox & al 99].

2.1.2. L'adaptation pour faciliter la navigation.

L'objectif est d'aider l'utilisateur à se repérer dans l'hyper-espace ou à l'inciter, voire l'obliger à utiliser certains liens plutôt que d'autres. Différentes techniques ont été développées au fil des années, entre autres le guidage direct, l'ordonnancement des liens, la masquage des liens, l'annotation des liens ou encore les cartes adaptatives [Brusilovsky 98].

2.1.2.1. Le guidage direct.

Cette technique a été la première à être utilisée régulièrement, car très simple et très facilement implémentable. Elle est basée sur l'ajout d'un lien hypertexte, nommé souvent "suivant" (ou *next* en anglais), qui permet d'accéder à la meilleure page, c'est-à-dire celle qui est en adéquation avec les objectifs et/ou capacités de compréhension de l'utilisateur.

On peut utiliser cette technique en laissant les autres hyperliens existants au préalable ou en les supprimant. Dans ce dernier cas, l'hypertexte perd beaucoup de ses capacités d'exploration, puisque le système devient totalement linéaire (il conserve toutefois son aspect dynamique).

En fait, pour être réellement efficace, cette technique doit être utilisée conjointement avec au moins une des techniques suivantes.

2.1.2.2. L'ordonnancement des liens.

L'ordonnancement des liens, comme son nom l'indique, est une technique qui propose d'afficher les liens hypertextes suivant un ordre définissant l'intérêt ou l'importance des pages cibles.

Cette technique ne peut pas être utilisée dans tous les cas. En effet, on ne peut pas l'utiliser avec des liens contextuels, c'est-à-dire des liens qui se trouvent au sein de phrases. En fait on ne peut l'appliquer que sur des liens qui appartiennent à un index, ou alors à une carte décrivant l'hyper-espace du système (comme nous le verrons par la suite).

L'intérêt de cette technique est très controversée. Certaines études ont en effet montré que la non-stabilité d'une liste de liens pour une page donnée pouvait désorienter l'apprenant. Et a contrario d'autres études ont montré qu'elle pouvait réduire les temps de navigation des hypermédia surtout ceux axés sur la recherche d'informations [Brusilovsky 94a].

2.1.2.3. Le masquage des liens.

La technique dite du masquage de lien consiste à supprimer les liens hypertextes dont les pages cibles sont soit en inadéquation avec le modèle de l'utilisateur (trop simples ou trop compliquées), soit en inadéquation avec les objectifs de l'utilisateur.

Facile à mettre en oeuvre, puisqu'il suffit, avant d'envoyer la page à l'utilisateur, d'enlever les

liens non désirés, cette technique permet de réduire la taille de l'hyper-espace pour l'utilisateur. Elle s'applique de plus sur tous les types de liens, contextuels ou non, avec des activateurs très divers (texte, bouton, icône, image, etc.).

Tout comme les techniques que nous avons vues précédemment, cette technique a aussi un défaut. La suppression de liens hypertextes peut en effet entraîner chez l'utilisateur, et surtout chez l'apprenant une mauvaise représentation mentale des tenants et aboutissants de chaque page (et donc de chaque notion si dans un système on assimile une notion et une page).

2.1.2.4. L'annotation des liens.

L'annotation des liens part du principe que l'utilisateur doit savoir où il va avant d'activer un lien. Il faut donc adjoindre à chaque lien des explications sur la page cible ou alors définir une syntaxe ou un codage particulier (par exemple telle icône pour dire que c'est une aide, telle couleur pour dire qu'il s'agit d'un exemple, etc.) [Brusilovsky & al 95].

A la différence des commentaires que l'on peut ajouter à nos liens et images de pages html (qui apparaissent sous forme de bulle ou en bas de nos navigateurs), les annotations de liens, pour être efficaces, doivent être fonction de l'utilisateur.

Cette technique, assez simple à mettre en oeuvre, peut être utilisée pour tous les types de liens, et ne semble pas avoir d'effets de bord néfastes.

2.1.2.5. Les cartes adaptatives.

Les cartes, ou *map* en anglais, permettent de présenter à l'utilisateur, l'organisation de l'hyper-espace, à l'aide de liens, soit sous forme textuelle (dans ce cas nous avons souvent une présentation hiérarchique de l'hyper-espace), soit sous forme graphique (Cf. [Pilgrim & al 99]). Dès lors, il est possible de présenter à l'utilisateur une organisation plus ou moins simplifiée en fonction de son profil.

Bien entendu, l'ensemble de ces méthodes peuvent et même doivent être combinées. Par exemple dans [Brusilovsky & al 94b], le système ISIS-Tutor utilise les techniques de guidage directe, de masquage de liens et d'annotation de liens.

2.2. L'interconnexion entre le modèle du domaine et les pages de l'hypermédia.

Nous avons vu dans le premier chapitre, que pour qu'un logiciel éducatif soit "intelligent", il faut qu'il soit composé de quatre composants, que sont le modèle du domaine, le modèle de l'apprenant, le module pédagogique et le module d'interface. En fait ceci s'intègre dans une définition plus large. En effet, pour qu'un système s'adapte à l'utilisateur, il faut absolument au moins deux composantes : le modèle du domaine et le modèle de l'utilisateur. Le modèle du domaine sert à organiser la connaissance, et le modèle de l'utilisateur permet d'adapter le contenu et les liens hypertextuels que l'on va présenter à l'utilisateur.

Se pose alors la question de l'interconnexion entre ce modèle du domaine qui organise la connaissance[25], et les pages de l'hypermédia qui la présentent. [Brusilovsky 94a] a dénombré trois méthodes permettant d'interconnecter ces deux composants : l'indexation par page, l'indexation fragmentée et la relation directe.

2.2.1. La méthode dite d'indexation par page.

L'objectif de cette méthode est d'indexer tout le contenu des pages de l'hypermédia avec les concepts du modèle du domaine. Ainsi chaque concept est associé à une page d'index qui permet d'accéder à toutes les pages présentant ce concept (Cf. figure 12).

[25] Nous verrons dans le chapitre IV de ce document les interconnexions possibles entre le modèle du domaine et le modèle de l'apprenant.

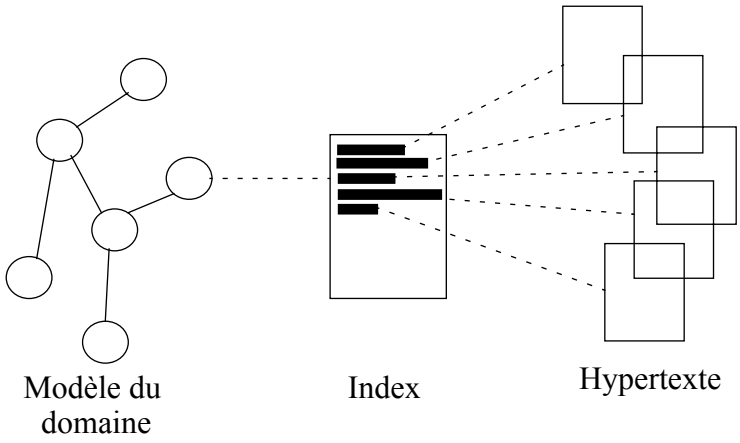


Figure 12 - Indexation par page

Il est aussi possible de créer des index relatifs aux relations définies dans le modèle du domaine. Ainsi le système peut créer un index par concept et par relation. Cet index réfère l'ensemble des pages de l'hypertexte qui contiennent des références à des concepts qui sont cibles de la relation spécifiée. Ainsi si le modèle du domaine contient n concepts et possède m types de relations, cela peut produire n*m index différents.

Cette méthode a été très utilisée dans les hypermédia éducatifs, par exemple dans [Brusilovsky & al 94b], car le ou les index que l'on peut présenter à l'utilisateur peuvent être modifiés à l'aide des techniques d'adaptation vues précédemment, et ainsi l'obliger, ou du moins l'inciter, à suivre telle ou telle direction.

Enfin, l'avantage de cette méthode est surtout sa facilité de mise en oeuvre, mais son système d'indexation est un peu grossier. En effet, si une page de l'hypertexte présente ou explicite plusieurs concepts, elle sera indexée plusieurs fois de la même façon.

2.2.2. La méthode dite d'indexation fragmentée.

Cette deuxième méthode est en fait une évolution de la méthode précédente. Cette fois les pages de l'hypertexte ne sont plus indexées dans leur globalité, mais indexées par fragment (Cf. figure 13). Cela permet d'avoir une indexation plus fine. Deux index peuvent alors référencer une même page de l'hypertexte mais pas au même endroit.

Outre un guidage de l'utilisateur, cette méthode permet aussi d'obtenir une adaptation au niveau du contenu, puisque les éléments référencés au niveau des index peuvent être très spécifiques et le système peut dans certains cas avoir des informations décrivant chaque élément (Cf. [Boyle & al 98]). On voit en fait apparaître par ce biais l'adjonction de sémantiques au sein des documents hypertextuels, ce qui de nos jours semble assez normal, grâce à l'utilisation du langage XML.

2.2.3. La relation directe.

Cette dernière méthode se distingue des deux méthodes précédentes par l'absence de page d'index. Ici, la structure de l'hyper-espace est totalement calquée sur la structure du modèle du domaine : chaque concept du modèle du domaine est représenté par une page de l'hypertexte (bien que certains systèmes relient de temps à autre un concept à un hyperdocument, c'est-à-dire une partie de l'hyper-espace), et à chaque relation on associe un lien hypertexte (Cf. figure 14).

Vis-à-vis des deux méthodes précédentes, la relation directe exige d'avoir une représentation du modèle du domaine sous forme de réseau sémantique (ou tout autre technique apparentée) afin de pouvoir déterminer les hyperliens, alors qu'auparavant il suffisait d'identifier les concepts

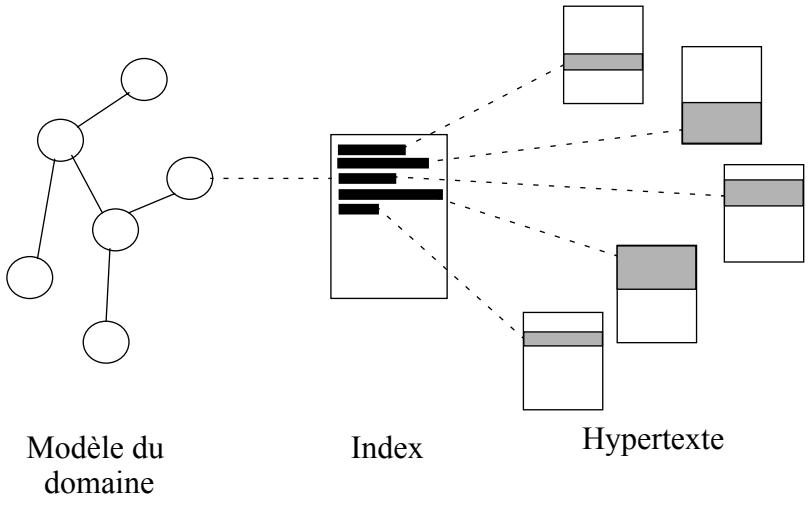


Figure 13 - Indexation fragmentée

sans être obligé de les associer.

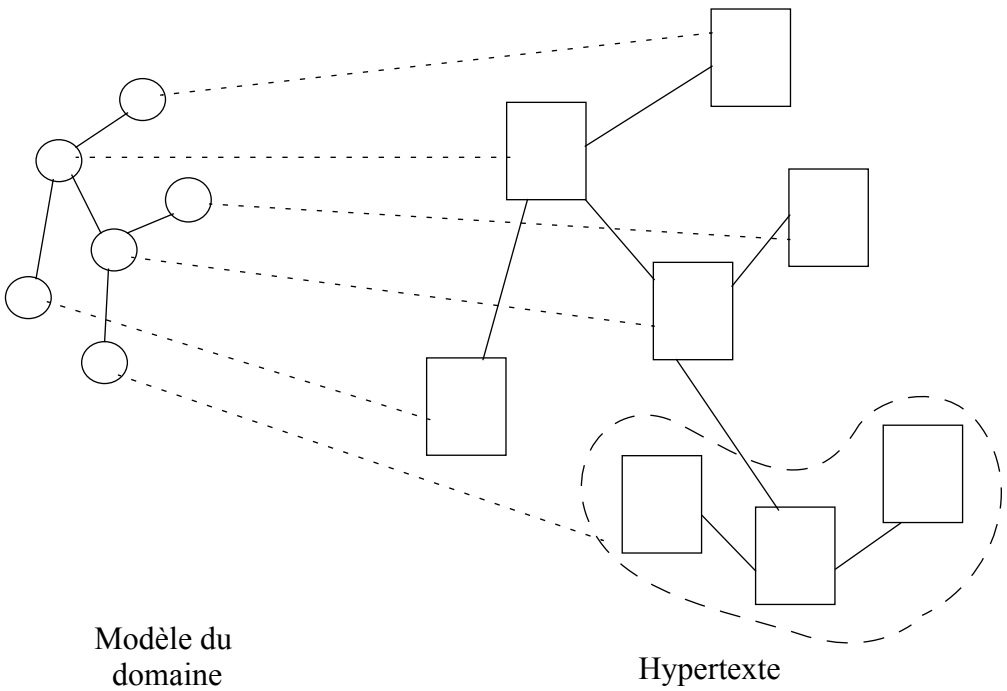


Figure 14 - Relation directe

3. Les hypermédia adaptatifs dynamiques.

Afin d'améliorer la qualité de l'adaptation et de prendre en compte instantanément de nouvelles données, depuis quelques années, les recherches se sont orientées également vers les hypermédia adaptatifs dynamiques.

La principale caractéristique de ces systèmes est d'offrir un hypermédia virtuel [Vassileva 95]. Le système n'est plus constitué de pages et de liens prédéfinis : ils sont construits dynamiquement (Cf. figure 15). L'architecture de ces systèmes repose sur quatre composantes principales que sont : le modèle du domaine, le modèle de l'utilisateur, une base de documents et un géné-

rateur de pages. Le modèle du domaine, comme pour la dernière génération des hypermédia adaptatifs, permet de définir l'architecture globale du système. Il y a par conséquent adéquation entre les nœuds du modèle du domaine et les pages de l'hypermédia virtuel, ainsi qu'entre les relations du modèle du domaine et les liens de l'hypermédia virtuel.

L'utilisation d'un tel système présente plusieurs avantages :

- Tout d'abord l'adjonction d'un nouveau support peut être immédiatement prise en compte, puisque encore une fois, les pages du système sont construites dynamiquement.
- Ensuite, les concepteurs de l'hypermédia ne sont pas obligés de penser à la façon d'agencer les différents documents, ils doivent juste définir l'architecture générale du système (le modèle du domaine) et déterminer, récupérer ou créer les documents qui vont servir à présenter chaque concept.

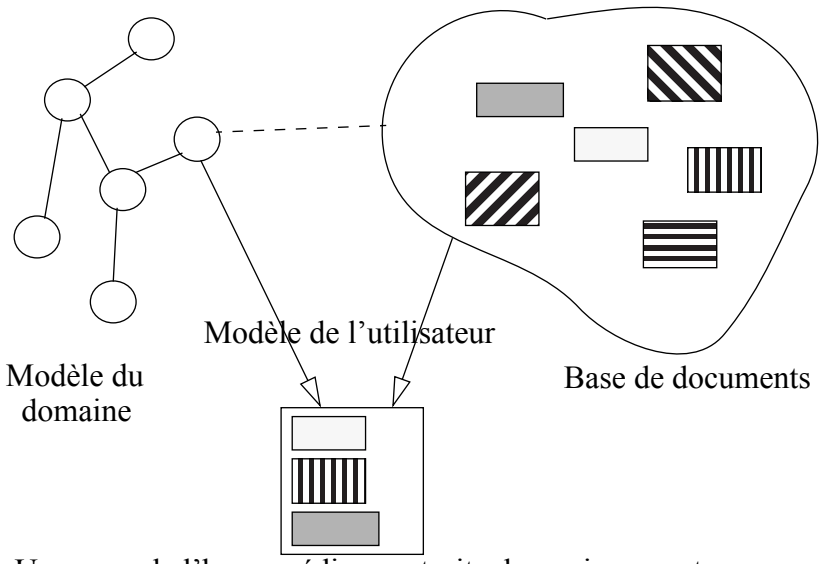


Figure 15 - Principe des hypermédia adaptatifs dynamiques

4. Pour ou contre l'utilisation d'un hypermédia dans un cadre éducatif.

4.1. Les hypermédia dits classiques.

Dès sa création, les hypermédia de par leur structure ont semblé être un nouvel outil pour la transmission du savoir, et donc utiles dans un cadre éducatif. Aujourd'hui, cet intérêt est encore de mise, voire plus, car renouvelé grâce au développement exponentiel du réseau planétaire Internet, avec toutes les nouveautés qui l'accompagnent, tant au niveau technique (applications distribuées, vidéo en temps réel - ou streaming -, visio-conférence à bas pris, etc.) qu'au niveau conceptuel, comme par exemple le concept de classe virtuelle.

Nous allons maintenant énumérer les avantages et inconvénients de l'utilisation des hypermédia dans un cadre éducatif.

4.1.1. Avantages.

Deux grands atouts, issus de la structure intrinsèque des hypermédia, émergent de leur utilisation dans un cadre éducatif : la composante multimédia et la composante hypertexte.

4.1.1.1. Avantages issus de l'aspect multimédia.

L'apport du multimédia dans l'éducation est très controversé : le multimédia apporte-t-il de réels bénéfices au transfert de la connaissance dans un cycle d'apprentissage, ou le multimédia est il un phénomène de mode ?

A ses débuts, pour l'ensemble de la communauté scientifique, il semblait logique que l'utilisation de données multimédia dans des logiciels éducatifs, et plus généralement dans les systèmes d'information, apportât obligatoirement un plus. On se basait alors sur des hypothèses, telles que:

- plus on stimule nos sens, plus l'information est compréhensible,
- le multimédia permet de capter plus longtemps l'attention de l'utilisateur,
- l'aspect ludique du multimédia est bénéfique,
- etc.

Quelques études ont alors essayé de vérifier ces hypothèses. Ainsi [Hoogeveen 95] a mis en évidence quelques attributs, ou critères, définissant la qualité d'une donnée multimédia. Il a ensuite étudié l'impact cognitif de chacun de ces critères. Par exemple:

- l'attribut *Level of Multimediality.* On part du principe que l'on détermine une relation d'ordre sur les différents types de média (on considère par exemple qu'une animation est supérieure à une image fixe).

  Malheureusement, dans la plupart des études, on n'a pas pu montré l'importance de cet attribut. Ainsi [Hoogeveen 95] conclut que :

  "[..] it can be observed that a higher level of multimediality alone is not sufficient for a better task performance, only for some learning tasks is an effective information transfer noted. [..]"

- l'attribut *Level of Man-machine Interactivity*. Cet attribut permet de déterminer l'importance de l'interactivité d'un média. Cela va de la passivité totale, jusqu'à la réalité virtuelle, en passant par l'émission de requêtes (dans une certaine syntaxe ou en langage naturel).

  L'importance de cet attribut a été démontrée. Dans la plupart des cas, plus les média sont intéractifs, plus l'information est correctement assimilée.

- l'attribut *Level of Congruence,* qui peut se traduire par le nombre de média utilisés de façon redondante pour expliciter une même idée. Comme l'indique [Nemetz & al 98], il est maintenant démontré qu'utiliser deux de nos sens (par exemple la vue et l'audition) simultanément permet de mieux appréhender une donnée complexe.

  De plus l'utilisation simultanée de plusieurs sens permet d'éviter à l'apprenant de mal comprendre un concept. Par exemple l'analyse d'une courbe seule, peut engendrer des incompréhensions qui peuvent être évitées si en plus de ce schéma, l'apprenant entend une voix qui le décrit.

  Toutefois cette juxtaposition de média est difficile à mettre en oeuvre pour qu'elle soit efficace. Pour l'instant aucune méthodologie n'a été élaborée pour obtenir de bon résultat. Mais [Recker 95] a tout de même classifié les types de média (texte, animation, image, son) à utiliser pour chaque type d'action pédagogique (présentation, explication, exemple, etc.). Par exemple, elle considère que le son est particulièrement bien adapté lorsque le système doit alerter l'apprenant, ou lui présenter un résumé du cours qu'il vient de voir.

[Hoogeveen 97] finit par conclure que l'utilisation d'éléments multimédia si elle est judicieuse, c'est à dire en adéquation avec l'apprenant, et de qualité, peut améliorer le transfert de connais-

sances dans certains domaines. Il ajoute aussi qu'une information multimédia est souvent plus facile à mémoriser qu'une information monomédia. Enfin, et sur ce point toutes les tendances se rejoignent, l'aspect ludique d'une information multimédia ne peut être que bénéfique.

4.1.1.2. Avantages issus de l'aspect hypertextuel.

Outre la composante multimédia des hypermédia, la composante hypertexte peut aussi améliorer la qualité de l'enseignement.

En effet, les hypertextes, par leur structure, aident l'apprenant à mieux se représenter la connaissance, à mieux appréhender les tenants et les aboutissants de chaque concept. La non-linéarité de la progression de l'apprenant l'oblige à se construire sa connaissance en créant des connections entre les concepts. En effet, comme l'indique F. Nadeau dans [Nadeau 97] :

> *L'apprentissage comme la pensée ne se font pas par des idées isolées mais par des relations significatives ou associatives entre idées. [...] Donc l'hypermédia devient un outil de structuration de la pensée.*

On retrouve en fait les fondements de la théorie constructiviste, où l'apprenant apprend en interagissant avec le système. Dans le cas des hypertextes, l'apprenant apprend en activant les hyperliens du système, comme il le fait dans les micro-mondes en interagissant avec le système.

Pour conclure, [Nadeau 97] déclare que les hypertextes ont les avantages suivants :

- Les hypertextes favorisent la pensée associative, puisqu'ils permettent de présenter les tenants et les aboutissants de chaque concept.
- Les hypertextes favorisent l'initiative de l'apprenant, puisque l'apprenant interagit avec le système, il ne peut pas rester passif.
- Les hypertextes sont un support à l'apprentissage collaboratif, car contrairement aux autres supports pédagogiques tels que les livres, plusieurs apprenants peuvent l'utiliser simultanément. Les hypermédia sont alors un outils propices à la résolution de problèmes en groupe, ce qui peut amener des discussions, des négociations entre les apprenants.
- Les hypertextes facilitent l'apprentissage interdisciplinaire. Il est en effet tout à fait envisageable de construire des ponts entre différents hypermédia. On peut imaginer par exemple que la présentation d'une notion de science physique par un hypertexte, fasse référence à des notions mathématiques dans un autre hypertexte, et fasse aussi référence au découvreur de cette notion ou théorie dans un troisième hypertexte historique.

4.1.2. Inconvénients.

Malheureusement ces avantages peuvent devenir préjudiciables. [Rhéaume 93] souligne en effet que plusieurs problèmes peuvent apparaître lorsque l'on utilise les hypermédia à des fins éducatives. L'apprenant peut rencontrer deux problèmes que tout utilisateur d'Internet a déjà rencontré, c'est-à-dire la désorientation et la surcharge cognitive.

4.1.2.1. La désorientation.

La désorientation est issue de la facilité qu'a l'apprenant à se déplacer de nœud en nœud dans le système. Ainsi cette liberté de déplacement peut finir par troubler l'apprenant. Il risque de se poser des questions du type :

- "Où suis-je ?",
- "Pourquoi suis-je là ?",
- ou encore "Que dois-je faire ?".

[Rhéaume 93] explique que ceci est principalement dû à notre mémoire à court terme, puisque

comme l'a montré [Miller 56], les êtres humains ne sont capables de mémoriser sur le moment qu'un nombre limité d'informations (sept items à plus ou moins deux près).

4.1.2.2. La surcharge cognitive.

La surcharge cognitive, quant à elle, est provoquée par "l'avalanche d'informations" que risque de "déverser" le système. En effet, la redondance, pour être bénéfique, doit être construite de façon intelligente. En aucun cas, il ne faut présenter la même information à l'aide de différents média ne nécessitant pas tous le même niveau de connaissance.

4.2. Les hypermédia adaptatifs.

4.2.1. Avantages.

Les hypermédia adaptatifs représentent une avancée non négligeable vis-à-vis des hypermédia classiques. Ils sont un atout pour les utilisateurs du système : les enseignants et les apprenants.

Tout d'abord, les différentes techniques utilisées permettent à l'étudiant d'être guidé dans son apprentissage. Ainsi, sans toutefois annihiler la liberté de navigation intrinsèque aux hypermédia, l'étudiant est constamment guidé dans son cheminement. Plusieurs études ont montré l'intérêt des hypermédia dynamiques vis-à-vis des hypermédia dits classiques ou statiques. Les hypermédia dynamiques peuvent améliorer l'assimilation des connaissances, ils peuvent réduire de façon considérable le parcours de l'utilisateur dans l'hyper-espace. Par exemple [Bodmer & al 97] ont montré à travers une étude, réunissant divers types d'utilisateurs dont le but est de répondre à huit questions à l'aide d'un système hypertexte, que les plus novices étaient grandement aidés. De même [Cox & al 99] ont montré que les chemins parcourus par les utilisateurs d'un hypermédia dynamique étaient beaucoup plus clairs que ceux qui parcouraient un hypermédia classique.

Ensuite, les hypermédia adaptatifs (surtout pour ceux qui appartiennent à la dernière catégorie, c'est-à-dire qui effectuent une relation directe entre le modèle du domaine et les pages de l'hypermédia) permettent aux enseignants de mieux structurer leur travail. En effet, le fait de distinguer la connaissance des outils qui permettent de la présenter éclaircit le travail de l'enseignant. Ce dernier peut alors mieux structurer son travail, en pensant tout d'abord à l'organisation des connaissances, et ensuite à la façon de les exposer.

4.2.2. Inconvénients.

Cependant, quelques problèmes persistent :

- Tout d'abord, l'accent a surtout été mis sur l'adaptation des liens, afin de guider l'apprenant dans son cheminement. Or la deuxième composante de l'adaptation, c'est-à-dire l'adaptation du contenu, a souvent été mise de côté.

  Pourquoi ?

  Tout simplement parce que la méthodologie de développement et l'architecture de ces systèmes ne s'y sont pas réellement prêtées. En effet, bon nombre de systèmes hypermédia adaptatifs sont issus de systèmes hypermédia classiques déjà définis, auxquels les chercheurs ont ajouté des techniques d'adaptation. Or, alors qu'il est assez aisé de cacher des liens, ou bien de les annoter, il est beaucoup plus difficile de remplacer un item d'une page, ou bien de modifier la structure d'une page.
- Ensuite, on sait qu'un bon système éducatif doit être un système ergonomiquement uniforme (par exemple dans les livres scolaires, les différents cours ont à peu près la même architecture, le même enchaînement logique). Or rien n'oblige les hypermédia à suivre cette démarche, ce qui peut être dommageable pour l'apprenant.
- Enfin, tout comme un enseignant, il faut que le système puisse utiliser immédiate-

ment toute nouvelle connaissance, ou tout nouveau média pour présenter une nouvelle connaissance. C'est une des caractéristiques d'un bon enseignant, il doit par exemple utiliser au maximum l'actualité pour agrémenter son cours. Ainsi, si une personne trouve ou construit un nouveau média en rapport avec un des concepts enseignés, le fait de l'ajouter doit permettre au système d'enrichir instantanément les cours sur ce concept, ce qui pour l'instant n'est pas très aisé à réaliser.

4.3. Les hypermédia adaptatifs dynamiques.

A priori, il est difficile d'évaluer les avantages et inconvénients de l'utilisation de ce type de système dans un cadre éducatif, car avant notre étude, à notre connaissance, seule J. Vassileva avait développé un tel système [Vassileva 97]. Depuis, outre notre étude, divers systèmes sont en développement, tel que le système CAMELEON de Melle Laroussi [Laroussi 98], ou encore le système de Melle Kavcic [Kavcic 98].

Toutefois, d'après ce que l'on a vu précédemment, l'architecture des hypermédia adaptatifs dynamiques doit reposer sur quatre composantes principales :

- le modèle du domaine,
- le modèle de l'utilisateur,
- une base de documents multimédia didactiques (ou *teaching materials*)
- et un générateur de cours.

Dès lors, l'utilisation d'un tel système doit apporter les avantages suivants :

- Tout d'abord l'adjonction d'un nouveau support peut être immédiatement pris en compte, puisque encore une fois, les pages du système sont construites dynamiquement.
- Ensuite, les enseignants ne sont pas obligés de penser à la façon d'agencer les différents média, ils doivent juste définir l'architecture générale du système (le modèle du domaine) et déterminer, récupérer ou créer les matériaux pédagogiques qui vont être utilisés pour présenter les notions introduites dans les dits cours.

5. Conclusion.

Les hypermédia représentent une "nouvelle" méthode de transmission de l'information. Leur utilisation dans un cadre éducatif, qui pour les premiers types d'hypermédia posait quelques problèmes (la désorientation, la surcharge cognitive), est aujourd'hui un fait incontournable. En effet, bien que certaines études ont tenté, ou tentent encore de minimiser l'impact cognitif des hypermédia (ce qui devient de plus en plus difficile avec l'apparition des hypermédia adaptatifs), la position prédominante des hypermédia dans les nouvelles technologies de l'information et de la communication, les rend pratiquement incontournables.

Cependant, le dernier paragraphe de ce chapitre sur l'utilisation des hypermédia adaptatifs dynamiques pour l'enseignement ne nous satisfait pas totalement. En effet, on peut légitimement se poser les questions suivantes :

- Quelles sont les techniques IA utilisées dans ces systèmes ?
- Quelles connaissances doit intégrer le système ?
- Comment caractériser ces fameux *teaching materials* ?
- Comment construire dynamiquement les pages de l'hypermédia ?
- Comment développer ce type de projet pour qu'il soit accessible depuis un simple navigateur Web ?

C’est entre autres à partir de ce type de questions que nous avons débuté notre réflexion, ce qui nous a amené à définir et un concevoir un prototype : le projet METADYNE.

Chapitre III - Objectifs.

1. Limites des systèmes existants.

Suite à ce que nous venons de voir, on peut affirmer que la dispense d'un savoir à l'aide des NTIC semble des plus prometteuses, mais pour l'instant les systèmes qui permettent de construire des cours en ligne ont les limites suivantes :

Tout d'abord, ils ont souvent la faiblesse de ne fournir aux enseignants que des outils de construction de cours par agencement de média sans leur fournir des outils de modélisation de ces cours, les incitants alors à ne suivre aucune méthodologie. C'est un peu comme si on ne donnait à un informaticien qu'un simple compilateur, ou même un environnement de développement, sans lui fournir de logiciel de modélisation. Les "bons" informaticiens vont tout de même modéliser leur projet, suivre les différentes étapes définies par le génie logiciel, mais les mauvais "élèves" seront tentés de directement programmé, ce qui par expérience est catastrophique

Ensuite, ces logiciels se veulent ouverts, mais chaque enseignant travaille un peu dans son coin. Ils peuvent s'échanger des items didactiques, mais il n'y a pas d'échanges de point de vue sur l'utilisation de ces derniers. Les seuls échanges d'informations s'effectuent via les classiques courriers électroniques, et forums de discussion, qui ne sont pas les plus aptes à échanger des points de vues, surtout si l'un des acteurs n'utilise pas souvent le système.

Puis, les cours construits par ces enseignants sont relativement statiques. Une fois construits, il y a peu de chance pour qu'ils évoluent régulièrement (aussi bien au niveau des items didactiques utilisés, qu'au niveau des liens qui unissent les différentes pages du cours). Ensuite, quelque soit l'apprenant qui se trouve devant la machine, les cours proposés sont les mêmes. Il n'y a donc caractérisation des cours, ce que l'on nomme l'adaptabilité, que de la par de l'enseignant. Ce sont ses choix qui déterminent les caractéristiques des cours qu'il construit, en aucun cas les choix de l'apprenant n'ont d'influence sur ce dernier.

Enfin, du point de vue informatique, dans la plupart des cas, l'architecture logicielle mise en oeuvre est souvent dépassée. Dès lors l'enseignant utilise un système dont l'ergonomie est des plus douteuses, et en tout cas, pas des plus intuitives. Les interfaces sont en effet décrites à l'aide du langage HTML qui n'est pas très adapté à la description d'interfaces utilisateur interactives. De plus le code HTML fourni par ces serveurs est du code HTML standard, qui n'utilise pas ou peu le Javascript et encore moins le HTML dynamique, ce qui limite encore plus les capacités d'interaction entre les utilisateurs et le système.

2. Notre approche.

Notre objectif est donc de définir un système de production de cours basé sur le concept de réutilisation d'items didactiques, qui ne possèdent pas, ou du moins qui minimise, les limites définies ci-dessus.

2.1. Distinguer le fond de la forme.

Afin d'inciter les enseignants à suivre une méthodologie de construction, le système ne doit pas leur permettre d'agencer les items didactiques pour construire leurs cours. Il doit plutôt tout d'abord leur permettre d'identifier les notions qui sont introduites dans les cours, en les reliant par différents types de relation. Puis il doit leur permettre d'identifier les différents items didactiques qui vont permettre par la suite d'introduire ces différentes notions.

Cette distinction entre le fond, c'est-à-dire les notions induites par les différents cours ainsi que leurs relations inter-notions, et la forme, c'est-à-dire la présentation de ces notions à l'aide d'items didactiques et la représentation des relations inter-notions à l'aide entre autres de liens hypertextes, outre l'incitation à l'utilisation d'une démarche méthodologique, va aussi avoir des répercutions sur les capacités du système.

En effet, les cours ne sont plus alors statiques, ils sont la résultante entre autres (car nous allons voir que d'autres paramètres vont aussi intervenir) des notions, des relations entre notions et des items didactiques qu'auront référencés les enseignants. Dès lors, par exemple, l'ajout d'un nouvel item didactique sera pris en compte immédiatement. Les cours sont donc dynamiques, ils sont construits à la volée par une composante de notre système que l'on nomme le générateur de cours.

2.2. Permettre aux enseignants de partager leurs points de vues.

De plus, comme le suggère [Marquesuzaà 98], cela va permettre aux enseignants utilisant Métadyne de pouvoir échanger leurs points de vues, de pouvoir récupérer le travail de leurs collègues. On dépasse alors le partage d'items didactiques, on peut en plus partager les notions et les relations inter-notions présentées par ces items didactiques.

2.3. Améliorer l'adaptabilité.

Ensuite, cette distinction entre le fond et la forme va permettre d'étendre l'adaptatibilité du système. En effet alors qu'auparavant, seules les volontés de l'enseignant transparaissaient dans les cours, avec Métadyne les apprenants vont pouvoir spécifier certaines caractéristiques, qui seront utilisées par le générateur de cours. Un cours est donc dans ce cas adaptable par l'enseignant et l'apprenant, il est fonction des informations référencées par l'enseignant et des volontés pédagogiques de l'apprenant.

2.4. Vers un système adaptatif.

De plus, si l'on insère dans Métadyne le profil de l'apprenant (comme par exemple ses connaissances), le générateur pourra en tenir compte, ce qui modifira les cours produits. Cette fois, Métadyne dépasse le cadre de l'adaptabilité, pour devenir un système adaptatif. Dès lors tout cours produit par le système sera fonction des informations fournies par l'enseignant, des caractéristiques fournis par l'apprenant et du profil de l'apprenant.

2.5. Fournir des outils intuitifs.

Enfin, les possibilités actuelles des navigateurs nous permettent de concevoir un système dont les outils sont de véritables applications qui peuvent être activées comme une page Web et s'exécutées au sein du navigateur, d'où des outils beaucoup plus simples à utiliser et beaucoup

plus intuitifs.

2.6. Conclusion.

Pour conclure sur ces objectifs, on peut dire que Métadyne tente de s'inscrire dans une évolution qui semble inexorable. Comme l'indique la figure 16, des cours en ligne formant une seule et même entité tels que les sites Web d'Azurnet et PCSM, on est passé à des systèmes permettant de construire des cours multimédia en ligne à l'aide d'agencement d'items didactiques comme par exemple le projet ARIADNE, pour arriver enfin à un système qui tente de construire les cours automatiquement.

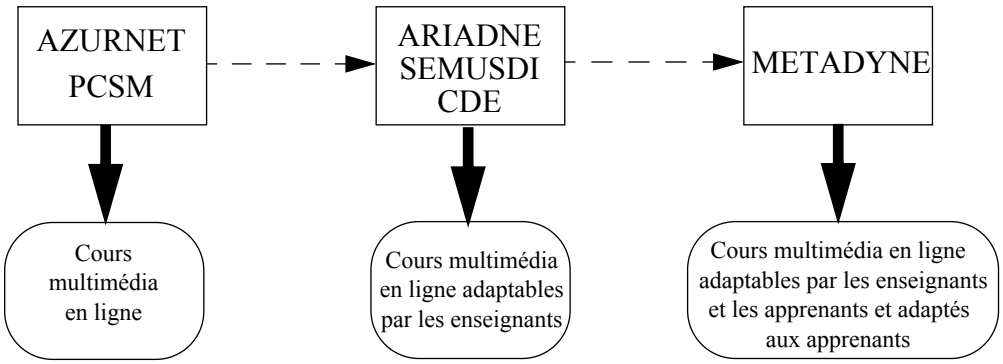


Figure 16 - Evolution des systèmes de transmission du savoir à l'aide des NTIC

3. Notre démarche.

Nous sommes avant tout des informaticiens, nous allons donc concevoir et réaliser une architecture informatique du système Métadyne, à l'aide des méthodologies issues du génie logiciel et du domaine de l'intelligence artificielle.

Tout d'abord, nous allons tenter d'identifier les différentes connaissances qui sont indispensables à la construction automatique de cours. Nous allons donc concevoir un modèle conceptuel (objectif du chapitre IV).

Ensuite, à partir de ce modèle conceptuel, nous allons définir un modèle objet implémentable (objectif du chapitre V).

Enfin, nous allons expliciter dans le chapitre VI, l'architecture logicielle de notre système.

Chapitre IV - Le modèle conceptuel[26].

Suite aux objectifs que nous nous sommes fixés dans le chapitre précédent, nous allons maintenant définir conceptuellement les différents éléments de notre système. Après une petite expertise définissant ce qu'est un cours, nous allons commencer par analyser les différentes connaissances liées aux enseignants, c'est-à-dire le modèle du domaine. Puis nous nous intéresserons aux deux facettes de l'image de l'apprenant dans le système, c'est-à-dire les deux sous-modèles du modèle de l'apprenant que sont le modèle comportemental et le modèle épistémique. Ensuite nous nous intéresserons à la caractérisation des composants (ou briques élémentaires) qui vont permettre de construire les cours. Enfin, nous finirons par déterminer le comportement du générateur de cours, en explicitant les différentes heuristiques que l'on utilise.

1. Expertise.

Nous avons vu dans le premier chapitre que différents outils étaient mis à la disposition des enseignants pour leur permettre de construire des cours multimédia et même hypermédia. Encore faut-il avoir une méthodologie cohérente pour bien les utiliser. En fait, en simplifiant il est vrai un peu, la création d'un cours, comme le montre l'interview d'un enseignant dans [Marquesuzaà 98], s'effectue en deux étapes :

- Tout d'abord déterminer les tenants et les aboutissants de chaque notion présentée.
- Ensuite créer physiquement le cours en agençant différents média.

Etudions donc ces deux phases.

1.1. Structuration d'un cours.

Un cours n'est jamais une entité isolée. Il s'inscrit toujours dans une démarche pédagogique précise, qui suivant le niveau d'étude (enseignement primaire, secondaire et supérieur) peut être plus ou moins spécifiée par les textes de lois du ministère de l'éducation nationale, mais c'est toujours finalement l'enseignant qui de par ses acquis et de par sa personnalité spécifie précisément cette démarche. Cette démarche s'inscrit alors dans un cursus complet plus ou moins long (généralement une année scolaire pour les établissements primaires et secondaires, et le plus souvent un semestre pour l'enseignement supérieur), et elle est rattachée à un champs d'enseignement, appelé aussi matière.

[26] Dans ce chapitre, toutes les représentations des modèles utilisent la notation UML.

Remarque : Ces champs d'enseignement sont organisés hiérarchiquement, sous forme d'arbre, c'est-à-dire qu'un champs peut être un sous-ensemble d'un champ père (on parle alors de sur-champ) et peut lui-même être décomposé en plusieurs champs (on parle alors de sous-champs): La classification de Dewey en est un bon exemple[27].

De plus, cette démarche pédagogique définit aussi la structure interne de ce cours. Là encore, bien que guidé par des documents pédagogiques tels que les manuels scolaires, c'est l'enseignant qui finalement détermine cette démarche.

Ainsi on peut appréhender l'organisation d'un cours suivant deux points de vues. Tout d'abord l'aspect macroscopique qui définit les relations pouvant exister entre les cours. Ensuite, l'aspect microscopique qui définit les relations pouvant exister entre les notions introduites dans un cours.

Prenons un exemple simple, l'organisation d'un cours physique sur les oscillations électriques libres (OEL), en classe de terminale scientifique :

- Au niveau macroscopique, ce cours s'inscrit dans une démarche d'enseignement des vibrations dans le domaine de l'électricité.
  Ce cours apparaît dans un cursus classique qui est à l'origine d'une succession de cours, tels que le cours sur les oscillations électriques forcées (OEF). On voit ainsi apparaître la relation de prérequis entre les cours.
  De plus, ce cours peut apparaître après un cours sur les oscillations mécaniques libres (OML), mais ceci n'est pas obligatoire. En effet en aucun cas le cours sur les OML est un prérequis pour les cours sur les OEL. Mais le fait d'avoir assimilé au préalable les notions introduites dans le cours sur les OML, peut aider à comprendre le cours sur les OEL, car les deux notions introduites dans ces cours sont proches.
- Au niveau microscopique, ce cours présente plusieurs notions, telles que la notion de pulsation propre d'un circuit oscillant ($\omega_0 = \langle\sqrt{LC}\rangle^{-1}$), ou encore la formule, dite de Thomson ($T_0 = (2\pi)/\omega_0 = 2\pi\sqrt{LC}$), permettant de calculer la période de l'oscillation. A ce niveau, un cours peut donc être décomposé en une succession de notions.

Mais qu'est ce qui différencie une notion d'un cours ?

A première vue, on serait tenté de répondre que cela dépend du point de vue que l'on choisit. Au niveau macroscopique, on a tendance à référencer les cours, alors qu'au niveau microscopique on a plutôt tendance à référencer les notions.

Cependant, rien n'empêche au niveau macroscopique, de relier un cours et une notion, et au niveau microscopique de relier une notion et un cours. Par exemple, un cours sur les OEF introduit la notion d'impédance. Or cette notion a entre autres pour prérequis le cours sur les nombres complexes. Ce qui signifie que l'on peut voir apparaître la notion "nombres complexes" dans un cours sur les nombres complexes, mais aussi dans un cours sur les OEF.

De ce fait, on peut considérer que les cours sont des notions particulières qui possèdent au niveau macroscopique une visibilité que les notions ne possèdent pas : les cours se comportent alors comme des points d'entrée dans le graphe des notions. C'est typiquement ce que l'on retrouve dans tout document pédagogique comme par exemple les livres scolaires. En effet, bien que présentant un ensemble de notions, un livre scolaire met tout d'abord en avant certaines d'entre elles dans la table des matières, ce sont les cours, alors qu'au contraire, les pages d'index présentent toutes les notions sans distinction.

[27] Vous pouvez trouver plus amples informations sur la classification décimale de Dewey sur http://www.oclc.org/oclc/fp/index.htm

1.2. Présentation d'un cours ?

Une fois que l'enseignant a bien déterminé les notions présentes dans son cours, ainsi que les relations qui les unissent, il faut construire physiquement ce cours, c'est-à-dire pour chaque notion, organiser séquentiellement différents média, qui la plupart du temps se résument à des média textuels (paragraphe, tableau, équations, etc.) ou picturales (images, graphes, histogrammes, etc.). Apparaît alors deux concepts importants que sont le public visé, et la réutilisation de matériaux didactiques.

Tout d'abord le public visé va déterminer l'approche pédagogique de l'enseignant (par exemple une approche plutôt pragmatique ou plutôt théorique), ce qui va avoir une influence d'une part sur la structure de présentation des notions, que l'on nomme le canevas, et d'autre part sur le choix des média présentant ces notions. Ainsi, si le public visé est néophyte en la matière, une approche pragmatique, expérimentale est le plus souvent souhaitable, ce qui suppose une présentation des notions axée sur l'exemple et la pratique. Alors que si le public visé est déjà compétent, ou si l'aptitude des étudiants le permet, une présentation théorique est directement envisageable. Ainsi dans notre exemple d'un cours sur les OEL, une présentation pragmatique de la formule de Thomson passera plutôt par l'utilisation de média ressemblant à la manipulation illustrée par la figure 17 (manipulation que l'on peut retrouver sur le site Web AZUR-NET[28]).

Ensuite la réutilisation de matériaux didactiques est un concept important en éducation. En effet, si un média présente particulièrement bien une notion, ou si un exercice permet de savoir précisément si l'apprenant a bien assimilé une notion, il peut être profitable de réutiliser ce média (et c'est ce qui se passe dans la pratique).

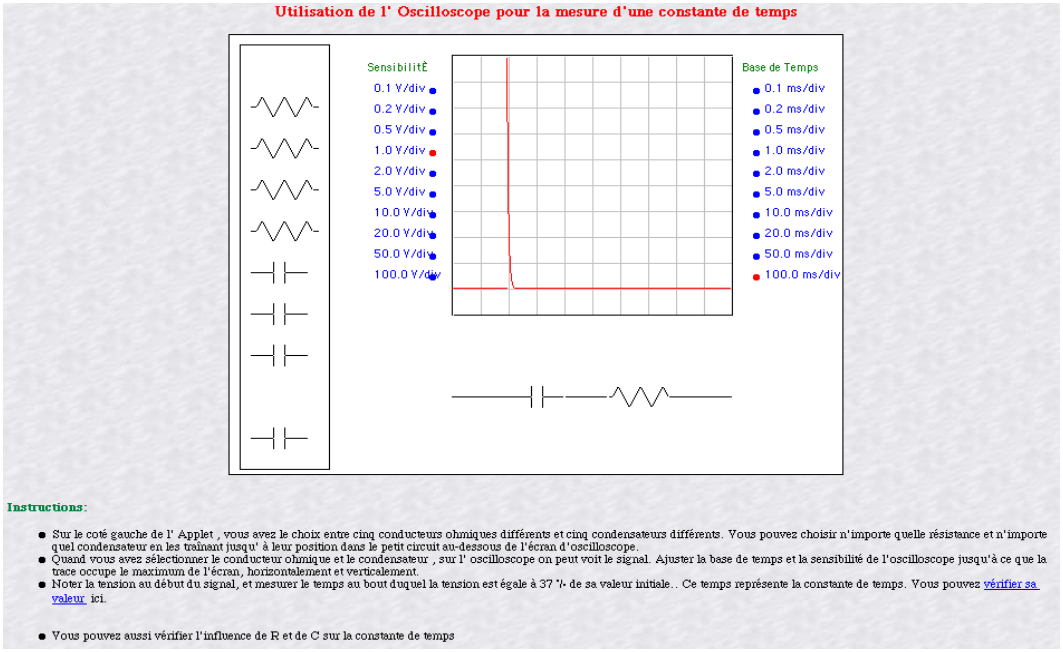


Figure 17 - Introduction de la formule de Thomson de façon expérimentale

En résumé, la présentation classique d'un cours passe par la présentation des notions qui le composent, et la présentation d'une notion s'effectue par l'introduction séquentielle, qui est fonction du public visé, de média créés ou réutilisés, qui encore une fois sont aussi fonction du public visé.

[28] http://www2.ac-nice.fr/second/discip/physique/Rc/RC.htm

1.3. L'apport d'un système d'enseignement informatisé.

Sans répéter ce que l'on a déjà vu précédemment, on peut toutefois mettre en exergue les points suivants :

- Tout d'abord, l'informatique permet d'accéder rapidement à l'information. L'accès aux cours, puis aux notions, à l'aide d'un hypermédia adaptatif, peut donc se faire instantanément en guidant l'apprenant.
- Ensuite, les types des média disponibles sont beaucoup plus variés, en plus des classiques média textuels et picturales, on peut afficher des animations, des vidéos, des applications interactives, etc.
- Enfin la personnalisation d'un cours, de par la définition de sa structure et de par les média qui sont choisis pour le présenter, qui pour un cours classique est déterminée par l'enseignement, peut aussi être caractérisé par l'apprenant (en choisissant un certain canevas, en spécifiant les types de média préférés, en choisissant le point de vue de tel ou tel enseignant, etc.).

A partir de ces différents constats, nous allons maintenant déterminer quelles sont les informations qui dépendent des enseignants, des apprenants, et des média.

2. Le modèle du domaine.

Nous avons vu dans le chapitre sur les systèmes d'enseignement assisté par ordinateur, que le modèle du domaine est la composante d'un système d'enseignement qui permet à l'ordinateur de "connaître" ce qui va être enseigné à l'apprenant. Ce modèle est dès lors défini par les experts du domaine à enseigner, c'est-à-dire dans notre cas les enseignants.

Nous allons donc tout d'abord expliciter l'ensemble des connaissances que devront spécifier les enseignants, ainsi que leur organisation. Nous allons voir ensuite les différents formalismes qui permettent de représenter ce type de modèle. Ceci nous permettra alors de spécifier notre façon de représenter le modèle du domaine.

2.1. Description.

D'après ce que l'on vient de voir, l'enseignant est tout d'abord en charge de décrire la structure générale d'un cours, aussi bien au niveau macroscopique, qu'au niveau microscopique. Ainsi, si on définit une notion[29] comme étant une unité d'information pédagogique minimale, par exemple la pulsation propre d'un circuit oscillant, ou comme un ensemble de notions chapeauté par une notion, par exemple le cours sur les OEL, on peut déterminer quatre types de relation entre ces notions[30]:

1. La relation **de prérequis** qui indique que l'apprentissage d'une notion est assujetti à la maîtrise de la notion A.
   Par exemple, l'apprentissage de la notion "Oscillations électriques forcées" doit être précédé de l'apprentissage de la notion "Oscillations électriques libres".
2. La relation **d'analogie** qui indique que la maîtrise d'une notion A peut aider l'apprentissage d'une nouvelle notion B.
   Par exemple, l'apprentissage de la notion "Oscillations électriques libres" peut être facilité par la connaissance de la notion "Oscillations mécaniques libres".
3. La relation **de conjonction** qui indique que l'apprentissage d'une notion A s'effectue via l'apprentissage séquentiel d'une succession de notions A_i.

[29] Dans ce document, nous considérons que les termes "notion" et "concept" sont équivalents. De ce fait on les utilisera indifféremment.
[30] Nous avons déterminé ces quatre types de relation de façon pragmatique.

Par exemple l'apprentissage de la notion "Oscillations électriques libres" passe par l'apprentissage successif des notions "équation de décharge" et "énergie électrique totale d'un circuit LC".

4. La relation **de disjonction forte** qui indique que l'apprentissage d'une notion peut s'effectuer via l'apprentissage de telle ou telle notion.
   Par exemple la notion "Théorie de la lumière" peut être présenté à l'aide de la notion "Théorie ondulatoire de la lumière" ou (exclusif) à l'aide de la notion "Théorie corpusculaire de la lumière".

On peut toutefois améliorer ce modèle. En effet, l'analogie entre deux notions peut être plus ou moins importante. Par exemple, dans le domaine de la physique, l'analogie entre les OML et les OEL est assez flagrante, alors que dans le domaine de l'informatique, l'analogie entre les notions "Langage Pascal" et "Langage C" l'est beaucoup moins. Ainsi il peut être intéressant de pondérer ce type de relation entre deux notions afin d'indiquer l'importance de cette dernière.

Nous pouvons suivre le même type de raisonnement au sujet de la relation de prérequis. Dès lors, deux des quatre relations que nous venons de voir sont pondérées, pondération qui est proportionnelle à l'importance que donne l'enseignant créateur à sa relation.

Enfin, nous avons vu au début de ce sous-chapitre que la démarche pédagogique qui consiste à organiser les notions est propre à chaque enseignant. Sachant que notre système se veut interprofessorale et même inter-disciplinaire, il faut que le modèle du domaine intègre l'ensemble des démarches pédagogiques de chaque enseignant. Par conséquent, il faut que la création d'une notion ou d'une relation par un enseignant implique que l'item créé soit étiqueté comme appartenant à ce dernier. Cela va permettre aux différents acteurs du système, c'est-à-dire les enseignants, mais aussi les apprenants, de "savoir" qui pense quoi. Ainsi :

- Les enseignants pourront visualiser leurs démarches pédagogiques ainsi que celles de leurs collègues.
- Les apprenants pourront choisir de suivre le parcours pédagogique de tel ou tel enseignant, et pourquoi pas de suivre le parcours pédagogique d'un ensemble d'enseignants.

En résumé, le modèle du domaine de notre système est composé d'un ensemble de notions (incluant les cours) appartenant à des domaines d'enseignement organisés hiérarchiquement. Ces notions peuvent être reliées par quatre types de relation, dont deux sont pondérées. Chaque élément du modèle du domaine, c'est-à-dire les notions et les relations, est étiqueté par l'enseignant créateur. Ceci afin de permettre aux différents utilisateurs du système, enseignant et apprenant, l'accès aux points de vue d'un enseignant ou d'un ensemble d'enseignants.

Remarque : Sachant que la désignation d'une notion à l'aide d'un mot (ou même d'un ensemble restreint de mots) peut quelquefois porter à confusion, chaque enseignant (créateur ou associé) est invité à préciser sa pensé en ajoutant une définition.
Ainsi, chaque concept possède au moins une définition. La première définie par l'enseignant créateur, et les autres par les enseignants qui s'associent à l'existence de cette dernière.

2.2. Comment représenter un modèle du domaine ?

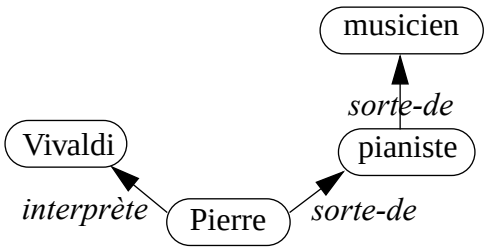
En fait, les méthodes, issues des recherches en intelligence artificielle, qui permettent de représenter la connaissance, diffèrent considérablement suivant le type de connaissance. En effet, on n'utilise pas les mêmes formalismes pour représenter d'un côté les connaissances liées au sa-

voir, et d'un autre coté les connaissances liées au savoir-faire.

Avant de déterminer le formalisme que l'on va utiliser, faisons un petit tour d'horizon de ce qui existe déjà en spécifiant s'ils sont plus ou moins aptes à représenter le savoir et le savoir-faire.

2.2.1. Formalismes basés sur l'utilisation de graphes.

Le principe est simple : réunir sous la forme d'un graphe, les notions représentant la connaissance, et les interconnecter par des liens. Le réseau sémantique est la première méthode basée sur ce principe. Elle fut imaginée M.R. Quillian dans [Quillian 68] en 1968 (Cf. figure 18). En 1979, G.G. Hendrix propose d'améliorer cette méthode de représentation en partitionnant le réseau en sous-réseaux organisés hiérarchiquement. La même année, S.E. Fahlman propose une deuxième variante : les réseaux à propagation de marqueurs. Enfin, en 1984, J.F. Sowa dans [Sowa 84] propose le modèle des graphes conceptuels, encore largement utilisé aujourd'hui.



Représentation de "Pierre est pianiste et interprète du Vivaldi"

Figure 18 - Représentation à l'aide d'un réseau sémantique

Dans ce formalisme le savoir est représenté par le réseau sémantique et le savoir-faire est représenté par un mécanisme qui permet d'inférer sur le réseau.

2.2.2. Formalismes basés sur la logique.

La logique est un système formel composé d'un langage, d'axiomes, de règles de production ou de réécriture et d'une fonction d'interprétation (qui permet de déterminer la validité d'une formule).

Il existe plusieurs logiques, tel que la logique des propositions, la logiques du premier ordre, les logiques modales et la logique floue. Elles se différencient de par leur alphabet (par exemple la logique des prédicats du premier ordre ajoute à la logique des propositions les symboles de fonctions et de quantifications), mais surtout de par leur fonction d'interprétation, permettant de représenter outre la véracité, l'incertain, la possibilité, la temporalité, etc.

La logique permet donc de représenter le savoir par l'intermédiaire de son langage et le savoir-faire par l'intermédiaire des règles de production, qui permettent de faire des inférences valides.

2.2.3. Formalismes basés sur l'utilisation des bases de règles.

Ce formalisme est basé sur de règle de production et une base de fait. Les règle sont du type "Si C1 et C2 et ... et Cn alors A" ou les C_i sont des conditions et A est une action qui agit sur le contenu de la base de fait (en aucun cas elle ne modifie la base de règle).

Un peu comme les formalismes basés sur la logique que nous venons de voir, il existe différents formalismes utilisant des bases de règles, par exemple le formalisme propositionnel (qualifié d'ordre 0), le formalisme Attributs-Valeur (qualifié d'ordre 0+), le formalisme Objets-Attributs-Valeurs, ou O-A-V (qualifié d'ordre +) et avec variables locales (qualifié d'ordre 1). Par exemple le tableau 2 présente pour chaque formalisme, la règle permettant de représenter : "le

fait que M. Dupond soit né à Paris, lui permet d'obtenir la nationalité française".

Formalisme	Type de règles
Propositionnel	SI Dupond-né-à-Paris ALORS Dupond_est_francais
Attributs-Valeurs	SI lieu_naissance = Paris ALORS nationalité=France
Objets-Attributs-Valeurs	SI Dupond lieu_naissance Paris ALORS Dupond nationnalité France
Avec variables locales	SI $pers lieu_naissance Paris ALORS $pers natinalité France ou par généralisation SI $pers lieu_naissance $ville ET $ville pays France ALORS $pers nationalité France

Tableau 2 - Représentation à l'aide de règles

Dans ce formalisme le savoir repose sur la base de règle et sur la base de fait. Le savoir-faire quant à lui repose sur les règles d'inférence qui permettent d'utiliser la base de règle.

2.2.4. Formalismes basés sur l'utilisation des langages de *Frames*.

Le principe de ces langages est de regrouper sous une même entité, nommée *frame*, des connaissances déclaratives et des connaissances procédurales. Très inspiré des langages informatiques objets, ce type de représentation est basé avant tout sur le principe de l'héritage.

Ce formalisme part donc du principe qu'à chaque savoir est associé un savoir-faire qui lui est propre (Cf. [Mendelsohn & al 93]).

2.3. Comment représenter notre modèle du domaine ?

Notre modèle du domaine représente le savoir des enseignants, qui va par la suite être utilisé pour construire des cours. Nous n'allons donc pas effectuer des inférences sophistiquées sur ce modèle, nous allons juste extraire de l'information suivant différents critères (par exemple prendre le point de vue d'un enseignant). Par conséquent, au vue de ce que l'on vient de voir, nous avons décidé d'utiliser un formalisme basé sur les graphes.

Ainsi dans ce formalisme, lorsque l'on prend en considération la description que l'on a faite du modèle du domaine, les noeuds de ce graphe peuvent être de quatre types :

- une notion à enseigner (qui peut être un cours),
- un champ d'enseignement qui permet de regrouper un ensemble de notions;
- un acteur du modèle du domaine, c'est-à-dire un enseignant,
- une information sur une relation, par exemple pour une pondération.

et les arcs de ce graphe représentent :

- les relations entre champs d'enseignement, pour introduire la notion de sur-champ et de sous-champ d'enseignement,
- ou les relations entre notions, telles que les relations de prérequis, d'analogie, etc.,

- ou bien les relations entre champs d'enseignement et notions.

Ainsi comme le montre la figure 19, nous venons de définir les primitives de construction de notre modèle du domaine. Ce dernier est alors une instanciation par les enseignants de ces primitives. Par exemple la figure 20 représente un fragment d'un modèle du domaine présentant le cours "Oscillation Electrique Libre" par l'enseignant Monsieur X.

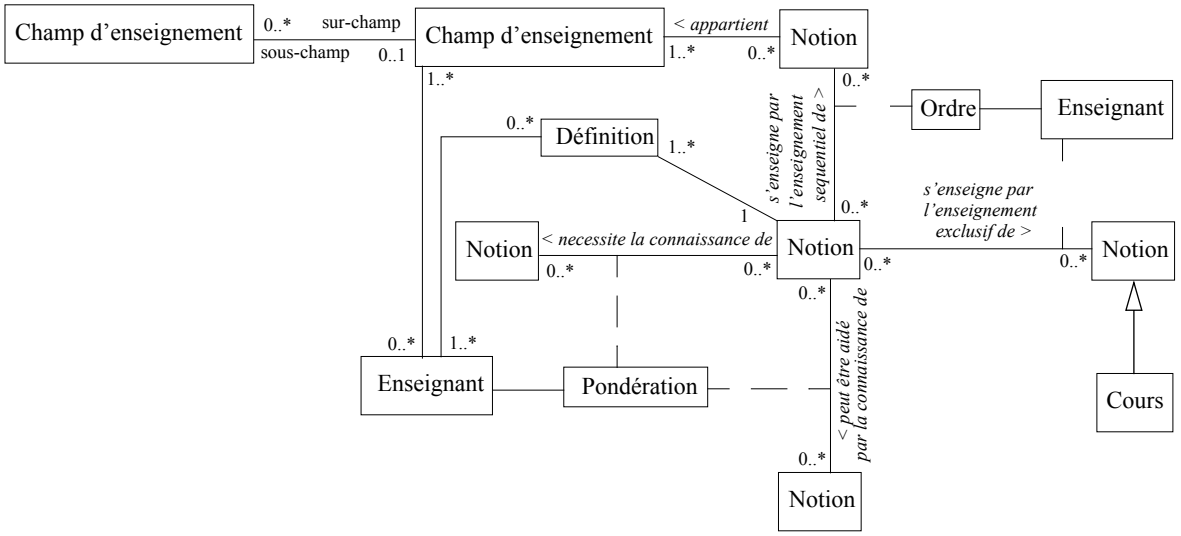


Figure 19 - ¨Primitives de construction du modèle du domaine.

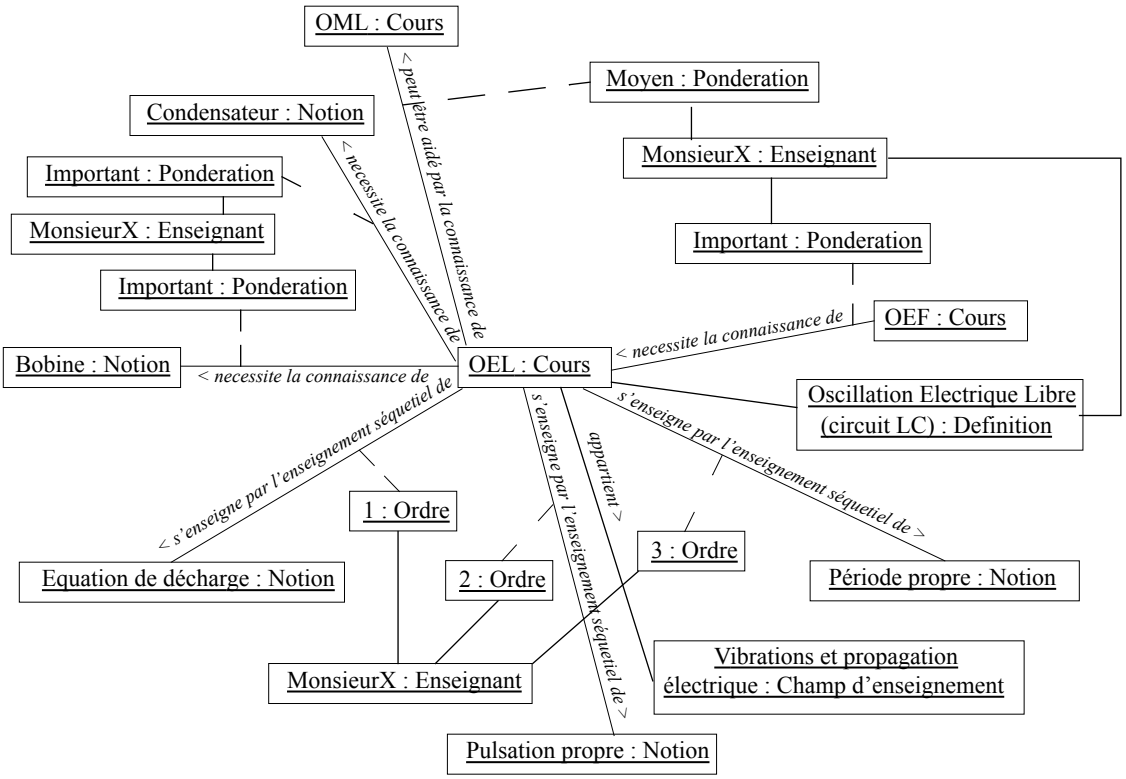


Figure 20 - Exemple de représentation du modèle du domaine

3. Le modèle de l'apprenant.

Pour qu'un système éducatif, qui transmet un savoir, soit "intelligent", il faut aussi que ce dernier soit capable de s'adapter à l'apprenant qui se trouve devant l'écran de la machine.

Qu'entendons nous par "s'adapter" ?

Dans notre cas, nous considérons qu'un système s'adapte à l'apprenant lorsque son "comportement pédagogique" ressemble à celui d'un enseignant, c'est-à-dire qu'il présente des connaissances compréhensibles, qu'il utilise ce que sait déjà l'apprenant ou encore qu'il utilise des média favorisant l'apprentissage de nouveaux concepts.

Ceci ne peut être atteint que par la connaissance du profil de l'apprenant. Ce profil doit intégrer aussi bien les connaissances de l'apprenant sur le modèle du domaine, ce que l'on nomme le modèle épistémique (ce que tente d'intégrer tous les modèles de l'apprenant), mais aussi ses particularités non-épistémiques, c'est-à-dire par exemple ses préférences ou ses objectifs pédagogiques. Ce deuxième sous-modèle, très rarement utilisé, se nomme le modèle comportemental.

Nous allons donc maintenant définir l'ensemble des connaissances caractérisant le modèle de l'apprenant. Puis après un rappel sur les représentations habituellement utilisées, nous déterminerons notre mode de représentation.

3.1. Description.

3.1.1. Le modèle épistémique.

Comme nous l'avons vu, l'objectif de ce sous-modèle est d'apprécier à sa juste valeur l'état des connaissances de l'apprenant pour les notions présentes dans le modèle du domaine.

L'expérience montre que lors du premier contact qu'il est à priori très difficile d'acquérir ces informations. Cela ne peut donc être obtenue que par une phase d'initiatisation, induite par l'une, ou la combinaison, des trois méthodes suivantes :

- Une méthode associée au cursus : à chaque niveau scolaire, on assigne un certain niveau de connaissance. Ainsi les apprenants ont tous le même modèle lors de la première utilisation.
- Une méthode associée aux résultats scolaires : on initialise le modèle de l'apprenant en fonction de ses résultats scolaires.
- Une méthode d'auto-évaluation: c'est l'étudiant qui détermine les caractéristiques de son modèle.

Cependant, quelque soit la ou les méthodes utilisées, il peut y avoir des manques (surtout lorsque c'est la méthode d'auto-évaluation qui est utilisée), il est alors très important de ne pas confondre une mauvaise connaissance du système au sujet de l'apprenant avec une mauvaise connaissance de l'apprenant pour une notion précise.

Enfin, comme tout le monde a pu le remarquer, notre mémoire n'est pas infaillible. Qui ne s'est jamais trouvé dans l'impossibilité de se remémorer une notion acquise il y a quelques temps? C'est à partir de ce constat que nous avons décidé que le modèle épistémique devait prendre en compte l'aspect temporel de la mémoire, et ceci non pas d'une manière uniforme, mais en fonction des caractéristiques du modèle comportemental que nous allons voir maintenant.

3.1.2. Le modèle comportemental.

Alors que le modèle épistémique est toujours présent dans les systèmes d'enseignement, le modèle comportemental est, le plus souvent, très limité voire absent. Or si l'on reprend la métaphore du choix d'un cours ou d'un ouvrage éducatif, ce choix est fonction :

- de la présentation du cours : son organisation, les média utilisés,
- du contexte d'apprentissage courant, si c'est pour présenter un exposer ou préparer un examen (dans ce dernier cas, l'expérience montre que les étudiants tentent de trouver les ouvrages écrits par l'examinateur),
- du sujet d'étude.

Nous avons donc décidé d'organiser les connaissances intrinsèques à ce sous-modèle en trois catégories.

La première, que l'on nomme les préférences, va permettre à l'apprenant de spécifier l'organisation des cours ainsi que les types physiques[31] de média préférés. Ainsi il a la possibilité :

- de choisir un canevas, c'est-à-dire une structure pour les cours qui vont lui être proposés.
- de définir un classement sur les types physiques de média.

La deuxième catégorie définie les objectifs pédagogiques de la séance courante. Ces objectifs vont avoir une influence sur le comportement du système. En effet, que l'utilisateur veuille réviser pour un examen, ou qu'il veuille approfondir de façon informelle sa connaissance, le système devra être plus ou moins souple et adopter un point de vue sur l'organisation de la connaissance plus ou moins large. De même, l'apprenant doit avoir la possibilité de choisir le point de vue de tel ou tel enseignant, voire pourquoi pas, de choisir le point de vue d'un ensemble d'enseignants.

Enfin, la troisième catégorie doit prendre en charge les capacités de l'étudiant, non pas de façon globale, mais par champs d'enseignement. Ainsi, comme l'indique la figure 21 qui sera plus amplement explicitée par la suite, les modifications temporelles du modèle épistémique ne seront pas toutes équivalentes, elle varieront suivant le champ d'enseignement de la notion courante.

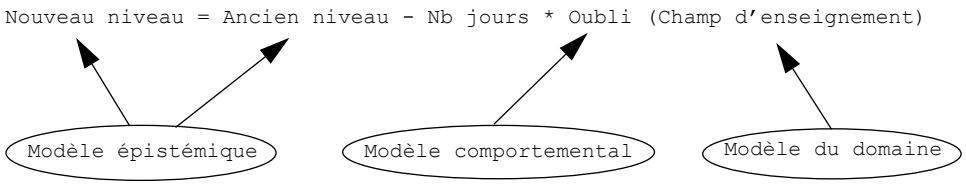


Figure 21 - Evolution temporelle du modèle épistémique

3.2. Comment représenter un modèle de l'apprenant ?

La représentation du modèle de l'apprenant, ou plus exactement la représentation du sous-modèle épistémique du modèle de l'apprenant, est un domaine de recherche à part entière. Toutefois les méthodes proposées peuvent être regroupées sous la coupole de deux méthodes principales, que sont la méthode dite de l'*overlay* et la méthode dite du *buggy model* (Cf. [Avgoustos & al 99]).

3.2.1. La méthode dite de "l'overlay".

Cette méthode, nommée aussi méthode de représentation par recouvrement, apparue avec le premier système d'enseignement, considère le modèle de l'apprenant comme un sous-ensemble du modèle du domaine, auquel il manque des informations (Cf. figure 22). Dans ce cas le but de l'apprentissage est d'obtenir finalement une superposition parfaite du modèle de l'apprenant

[31] Nous verrons plus loin ce que l'on entend plus précisément par "type physique"

sur le modèle du domaine.

3.2.2. La méthode dite du "buggy model".

Cette deuxième méthode est une extension de la précédente. Introduite par J.S. Brown et R.R. Burton dans [Brown & al 78], elle consiste à adjoindre aux données issues de la méthode par recouvrement une liste d'erreurs qu'a commises l'apprenant. Le but est alors d'examiner ces "bugs" et d'essayer de comprendre leurs origines (en les comparant à une liste d'erreurs typiques déjà observées), permettant alors d'améliorer les stratégies pédagogiques du système. Par la suite l'avènement de l'intelligence artificielle distribuée, et principalement l'avènement des systèmes multi-agents, à permis de remplacer cette liste statique d'erreur par la négociation entre agents. Différents agents adoptent chacun un point de vue sur l'état de connaissance et stratégie de résolution de l'apprenant. Ensuite, de par les interactions entre l'apprenant et le système, les agents, par échanges de "points de vue", tentent alors de faire émerger un profil cohérent (Cf. [Moulin 98],[Giroux & al 96] ou [Leman 96]).

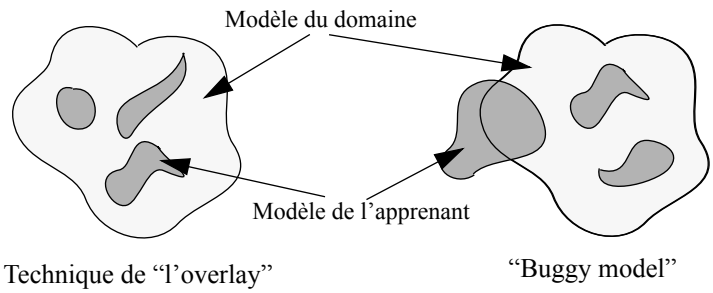


Figure 22 - Modèle de l'apprenant : technique de "l'overlay" et du "buggy model"

3.3. Comment représenter notre modèle de l'apprenant ?

Notre objectif initial n'était pas de trouver une nouvelle méthode de représentation du modèle de l'apprenant. De ce fait nous avons utilisé des méthodes classiques, qui peuvent présenter des défauts, mais qui ont l'avantage d'être facilement compréhensibles et implémentables.

3.3.1. Pour le modèle épistémique.

Dès lors, pour le modèle épistémique nous avons décidé d'utiliser la technique de recouvrement, qui bien que pouvant fonctionner sur des connaissances procédurales, est tout particulièrement bien adaptée aux savoirs, ce qui est notre cas. De ce fait, le modèle épistémique est représenté via une relation entre l'acteur du système qu'est l'apprenant et les différentes notions qu'il a pu étudier. De plus, ce lien possède une valeur, comprise entre 0 et 1, proportionnelle à la connaissance qu'a l'apprenant sur la notion associée. Enfin, si l'on veut prendre en compte le temps, il faut dater chacune de ces relations.

Dès lors comme le montre la figure 23, les primitives du modèle épistémique se résument à un ensemble de liens pondérés datés, associant un apprenant à un ensemble de notions du modèle du domaine.

3.3.2. Pour le modèle comportemental.

Nous avons vu que le modèle comportemental est constitué de trois catégories de connaissances. Parmi ces trois catégories, seules la première et la troisième sont représentées dans le modèle comportemental. La seconde catégorie quant à elle n'existe que par les informations qui sont envoyées au générateur de cours pour construire les pages du système. De ce fait la représentation du modèle épistémique se résume à représenter :

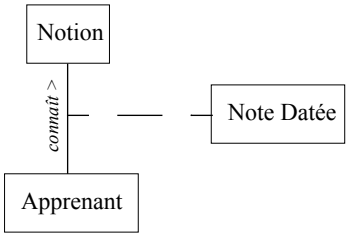


Figure 23 - Primitives de construction du modèle épistémique.

- le canevas qu'aura choisi l'apprenant,
- l'ordonnancement des différents types physiques de média,
- les capacités cognitives de l'apprenant pour chaque champ d'enseignement.

Comme nous l'avons vu, le canevas est la structure de présentation d'une notion, elle définit séquentiellement les éléments pédagogiques (encore appelé type cognitif). Par exemple, la liste suivante est un canevas plausible :

1. Introduction.
2. Définition.
3. Exemple.
4. Exercice.
5. Rappels.

Remarque : Afin que la structure soit pédagogiquement bonne, l'apprenant pourra choisir un canevas, mais il ne pourra en aucun cas en définir un lui même. Ces canevas ne sont pas fixés dans le système, chaque enseignant pourra en créer autant qu'il le désire.

Le classement des types physiques de média définit un ordre d'importance entre les différentes catégories de média (texte, vidéo, images, sons, etc).

Les capacités cognitives de l'apprenant pour chaque champ d'enseignement peuvent être assimilées à un oubli plus ou moins important des notions de ce champ, présentées utltérieurement. Elles peuvent donc être assimilées à une fonction qui pour un champ d'enseignement donnée retournera un coefficient de perte d'informations journalière, ce qui peut être représentée par une relation pondérée entre l'apprenant et chaque champ d'enseignement.

Remarque : Lorsque l'on a besoin de connaître les capacités cognitives de l'apprenant pour un champ d'enseignement donné, et que ces dernières ne sont pas connues, on recherche ses capacités pour le champ d'enseignement supérieur (dans l'organisation des champs d'enseignement), et ce jusqu'à ce que l'on obtienne un résultat.
Si l'on remonte jusqu'à la racine des champs d'enseignement sans obtenir l'information désirée, on considère que le coefficient de perte d'information est égal à zéro (ce qui signifie, que le temps n'est plus pris en compte dans le modèle de l'apprenant).

En somme, la figure 24 montre comment il est possible de représenter le modèle comportemental, et finalement la figure 25 montre un exemple complet d'un modèle de l'apprenant en restreignant le modèle épistémique autour de la notion d'OEL.

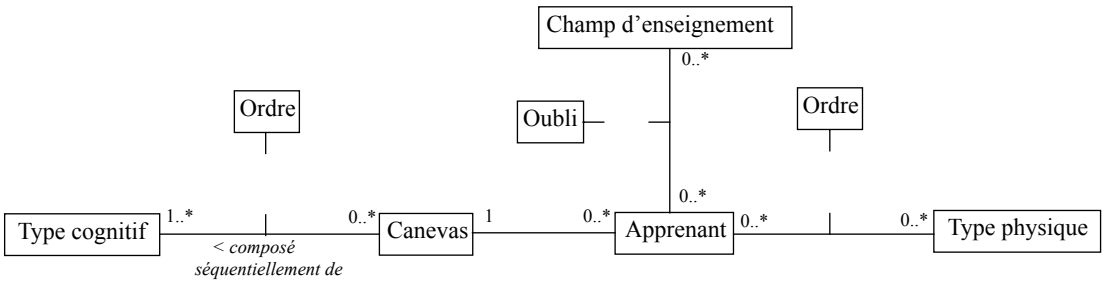


Figure 24 - Primitives de construction du modèle comportemental

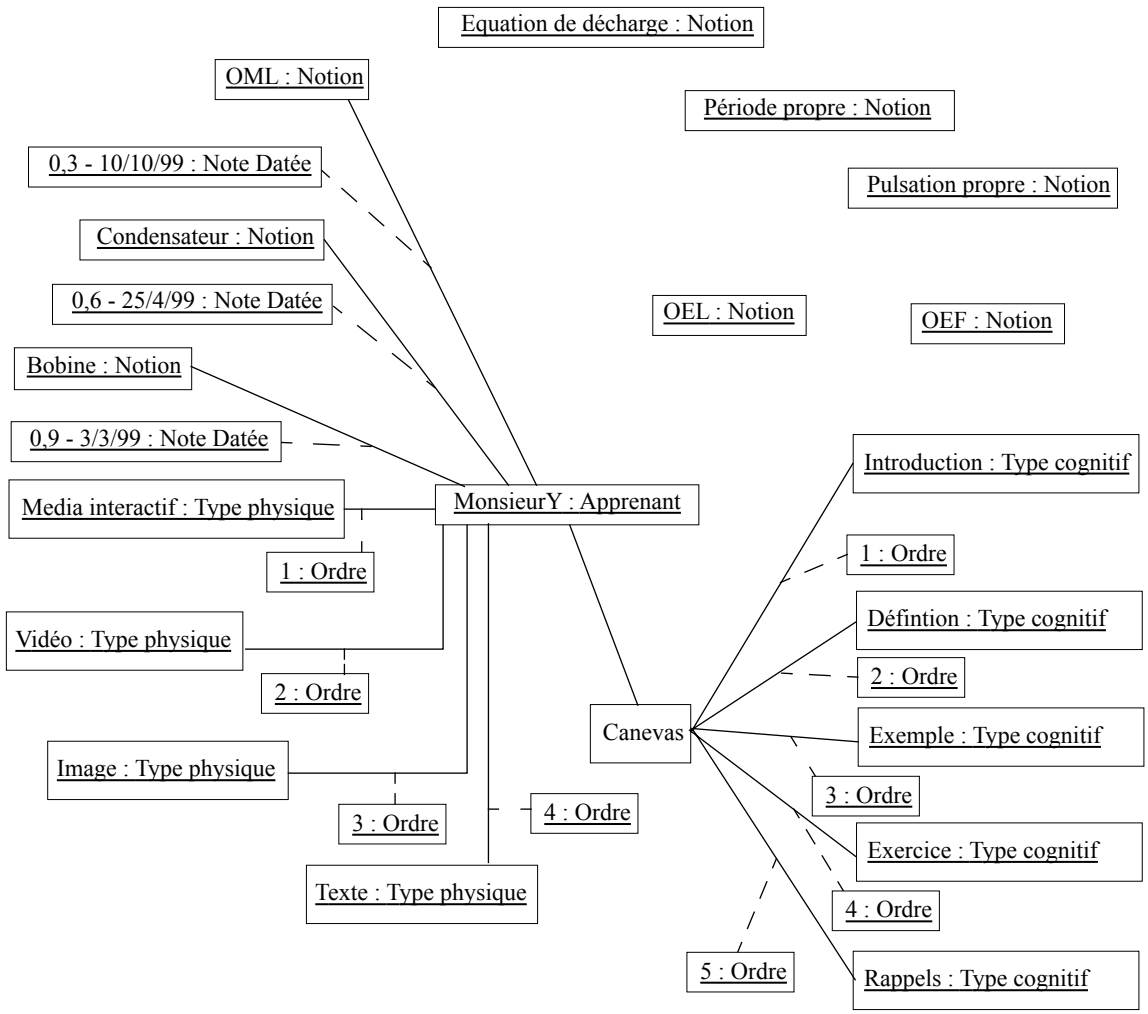


Figure 25 - Exemple de représentation du modèle de l'apprenant

4. Le concept de brique élémentaire.

Maintenant que l'on sait organiser les notions, et que l'on sait comment on va organiser pédagogiquement la présentation de chacune d'elles, il faut posséder des données multimédia pour pouvoir les présenter. Apparaît alors le concept de brique élémentaire (entrevu dans la présentation de SEMUSDI), c'est-à-dire des items didactiques multimédia minimaux :

- **items multimédia**, car ces données peuvent être aussi bien du texte, des images, que des animations, des vidéos, etc.
- **didactiques**, car ces données peuvent être utilisées dans un cursus d'apprentissage, pour présenter une notion (une introduction, un exemple, etc.), évaluer l'apprenant, etc.
- **minimaux**, car ces données doivent être assez ciblées pédagogiquement afin qu'elles puissent être réutilisées dans différents contextes.

Avant de définir les connaissances associées à chaque brique élémentaire, nous allons définir les objectifs que nous voulons atteindre.

4.1. Objectifs.

En fait, l'idée de réutilisation de données pédagogiques n'est pas nouvelle. Tous les enseignants ont un jour repris un polycopié d'un collègue, ou bien un ou plusieurs chapitres d'un livre pour préparer un cours. Aujourd'hui, rien n'a changé si ce n'est que l'on peut maintenant outre des données textuelles ou picturales utiliser et réutiliser des données multimédia. Cependant il faut pouvoir caractériser ces composants afin de pouvoir les rechercher efficacement et les utiliser et réutiliser correctement.

Nous avons vu dans le premier chapitre que différentes normes ont été définies. Certaines sont très complètes comme c'est le cas dans le projet ARIADNE, d'autres un peu moins comme c'est le cas dans le projet SEMUSDI. Ce semblant "d'anarchie" est du à l'aspect novateur du concept d'items didactiques, et disparaîtra nous le pensons lorsqu'un consortium établira une norme commune qui satisfera tout le monde. En attendant, il faut savoir que des filtres sont toujours envisageables pour passer d'une norme à une autre (c'est une des études menées dans le projet SEMUSDI). En attendant, pour ne déroger à la règle, nous allons établir nous aussi notre propre norme.

Dans notre projet, nous considérons que la sauvegarde, l'indexation et les problèmes juridiques, en somme tout ce qui concerne la gestion des documents, ne sont pas inclus dans notre problématique. L'enseignant qui veut ajouter une brique élémentaire dans le système doit seulement être capable de spécifier ses qualités physiques et didactiques.

En fait, pour qu'une brique soit convenablement référencée dans notre système; il faut pouvoir répondre aux questions suivantes :

1. Où se trouve la brique ?
2. Quel est son aspect physique ?
3. Comment s'insère-t-elle dans un cursus pédagogique ?
4. Quel niveau de connaissance faut-il avoir pour l'appréhender correctement ?
5. A quelle notion est associée la brique ?

Remarque : Les réponses aux trois premières questions sont intrinsèques à la brique, puisque en théorie, elles doivent être invariables quelques soit l'enseignants qui la référence (ce qui peut toutefois se discuter pour la troisième question).
Par contre les réponses aux deux dernières sont propres à chaque enseignant.

4.2. Notre choix.

Pour répondre à la première question, il faut être capable de sauvegarder la position de la brique sur le réseau Internet. Cela ne pose aucun problème, il suffit de sauver son adresse, c'est-à-dire son URL.

Pour répondre à la deuxième question, il faut associer à chaque brique un mot-clef issu d'une taxinomie entre les différents genres de média que l'on peut rencontrer. Il faut bien comprendre que l'objectif n'est pas de sauvegarder le format physique exact de la brique (par exemple un fichier au format JPEG) mais bien d'extraire le genre de média utilisé (par exemple une image). En fait, pour établir cette taxinomie, nous nous sommes inspirés de la hiérarchisation présentée dans le modèle RCOS pour les composants (Cf. tableau 1). Ainsi l'aspect physique d'une brique est identifié par un des éléments de la liste suivante :

- **Texte**, pour les média qui sont majoritairement composés de texte, de tableaux, ou de listes.
- **Vidéo**, pour les média qui sont majoritairement composés de film incluant ou non du son.
- **Animation**, pour les média qui sont majoritairement composés "d'images animées", sans son.
- **Photo**, pour les média qui sont majoritairement composés d'images fixes haute résolution telles que des photos ou des *snapshots*.
- **Graphique**, pour les média qui sont majoritairement composés de graphiques, de schémas ou d'histogrammes.
- **Son**.
- **Applications interactives**;

Tout comme pour la deuxième question, pour répondre à la troisième question il faut associer à chaque brique un mot clef issue de la liste suivante :

- **Introduction**, pour tout ce qui est présentation, introduction d'un concept.
- **Définition**, pour tout ce qui se rapporte à une présentation théorique, tel que les définitions, les axiomes, les couples théorème-démonstration.
- **Exemple**, pour représenter des exemples théoriques, applications pratiques ou des exercices types avec la ou les solutions.
- **Simulation**, pour faire manipuler l'apprenant comme par exemple des travaux pratiques.
- **Exercice**.
- **Rappel**.

Enfin, comme nous l'avons remarqué précédemment, les réponses aux deux dernières questions dépendent de l'enseignant qui insère la brique dans le système. Par exemple, une animation pré-

sentant la lune en train de tourner autour de la terre pourrait, pour un premier enseignant, être utilisée pour illustrer un cours sur la gravité. Alors que pour un deuxième enseignant elle pourrait illustrer un cours sur le mouvement des marées. Ainsi une brique peut être associé à différentes notions par une relation pondérée (indiquant le niveau minimum requis pour bien la comprendre) : relation qui est associée à l'enseignant créateur.

En résumé les briques élémentaires sont caractérisées par quatre attributs :

1. **La position**, qui permet de répondre à la question "Où se trouve la brique ?".
2. **Le type physique**, qui permet de répondre à la question "Quel est l'aspect physique de la brique ?".
3. **Le type cognitif**, qui permet de répondre à la question "Comment s'insère la brique dans un cursus pédagogique ?".
4. **Le niveau cognitif**, qui permet de répondre aux questions "Quel doit être le niveau de connaissance de l'apprenant pour appréhender correctement la brique ?", et "A quelle notion est-elle associée ?".

Dès lors, comme l'indique la figure 26, il est possible d'associer cet ensemble de connaissances à notre représentation du modèle du domaine. La figure 27 quant à elle, présente un exemple d'instanciation de ce modèle: l'enseignant "Monsieur X" a référencé une brique élémentaire qu'il a nommée "Déphasage entre U et I dans un circuit RLC". Cette brique est une simulation, avec une difficulté de compréhension "moyenne". Elle est composée majoritairement d'une application interactive et elle peut être retrouvée sur Internet grâce à l'URL *position*.

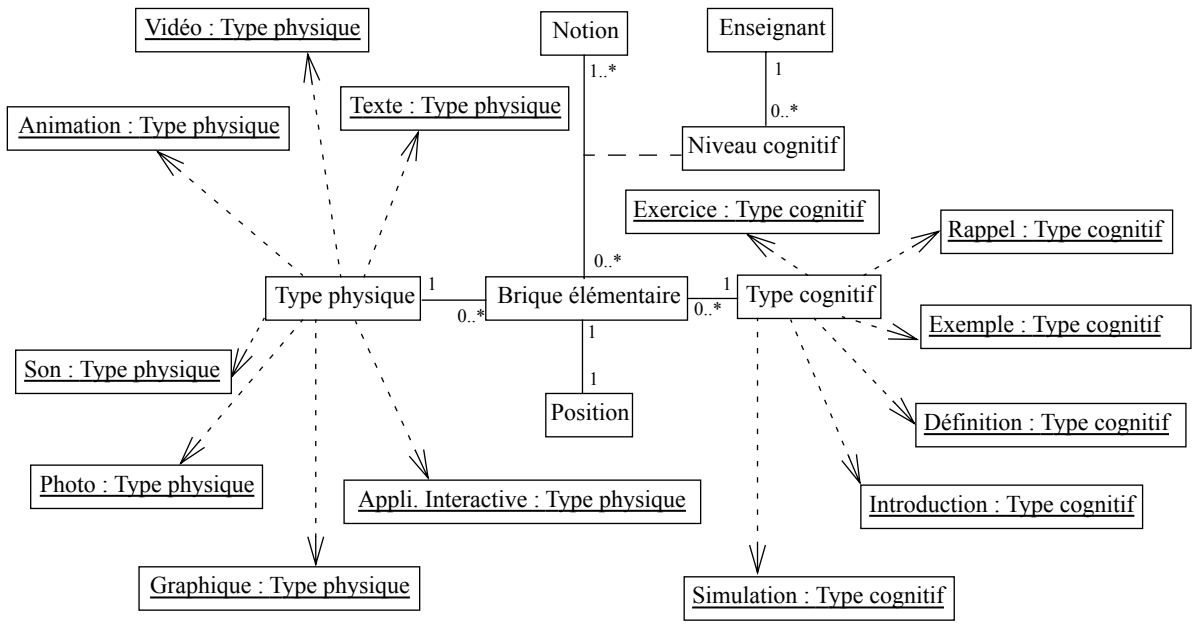


Figure 26 - Primitives de représentation du concept de brique élémentaire.

5. Le générateur de cours.

Nous allons maintenant étudier la dernière composante de notre modèle conceptuel, c'est-à-dire le générateur de cours (GC).

5.1. Objectifs.

Les objectifs de cette dernière composante, que l'on aurait pu aussi appeler module d'interface,

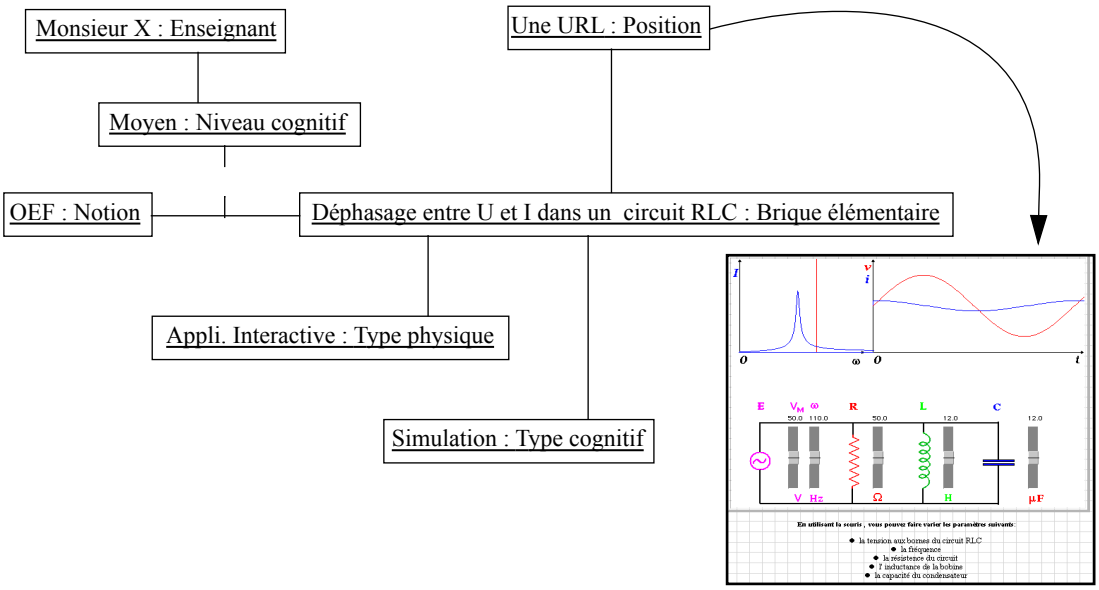


Figure 27 - Exemple de représentation du concept brique élémentaire.

sont simples, puisqu'elle est en charge, pour une notion choisie par l'apprenant, de la construction dynamiquement une page de l'hypermédia, en fonction de l'état courant du modèle du domaine, du modèle de l'apprenant et de la base de briques élémentaires.

Remarque : Le GC se distingue des modules que nous avons vus jusqu'à présent par le fait que ce dernier n'intègre pas de connaissance au sens propre du terme, il s'agit plutôt d'un système à part entière qui infère sur le modèle du domaine, le modèle de l'apprenant et sur la base de briques élémentaires.

Nous avons vu qu'un hypermédia est composé de pages et de liens. Le GC va donc être en charge de créer à un instant *t* une page et les liens qui lui sont associés.

5.2. La construction de la page.

La construction d'une page de l'hypermédia est principalement fonction du modèle de l'apprenant dans sa globalité, c'est-à-dire fonction du modèle épistémique et du modèle comportemental, ainsi que de la base de briques élémentaires. A ceci s'ajoute l'utilisation de trois filtres qui permettent d'extraire un ensemble de briques élémentaires en fonction tout d'abord de leur type cognitif, ensuite en fonction de leur niveau cognitif, et enfin en fonction de leur type physique.

En fait la construction d'une page débute lorsque l'apprenant décide d'activer un cours (donc une notion), ou lorsqu'il clique sur un lien hypertexte qui l'amène sur une nouvelle notion. Le GC récupère alors le canevas de l'apprenant, ainsi que le niveau de l'apprenant pour le concept courant. Ce niveau est automatiquement réévalué en fonction de la date de la dernière mise à jour, du champ d'enseignement de la notion courante et du modèle comportemental de l'apprenant, comme l'a montré la figure 21.

Ensuite le GC récupère le canevas de l'apprenant afin de déterminer la structure du cours. Il va donc essayer pour chaque élément de ce canevas de trouver le meilleur média. Pour cela, il va récupérer l'ensemble des briques élémentaires associées à la notion courante en se restreignant pour l'instant au point de vue choisi par l'apprenant. En effet, comme nous l'avons vu dans le

chapitre sur les objectifs du système, l'apprenant peut choisir de suivre le point de vue de tel ou tel enseignant, voire d'un ensemble d'enseignants.

Pour choisir la meilleure brique élémentaire, le GC va alors appliquer trois filtres. Le premier permet d'extraire un sous-ensemble de briques pour un type cognitif déterminé. Le second permet d'effectuer la même opération mais pour un ensemble de niveau cognitif déterminé. Enfin le troisième effectue le même genre d'extraction mais pour un type physique donné.

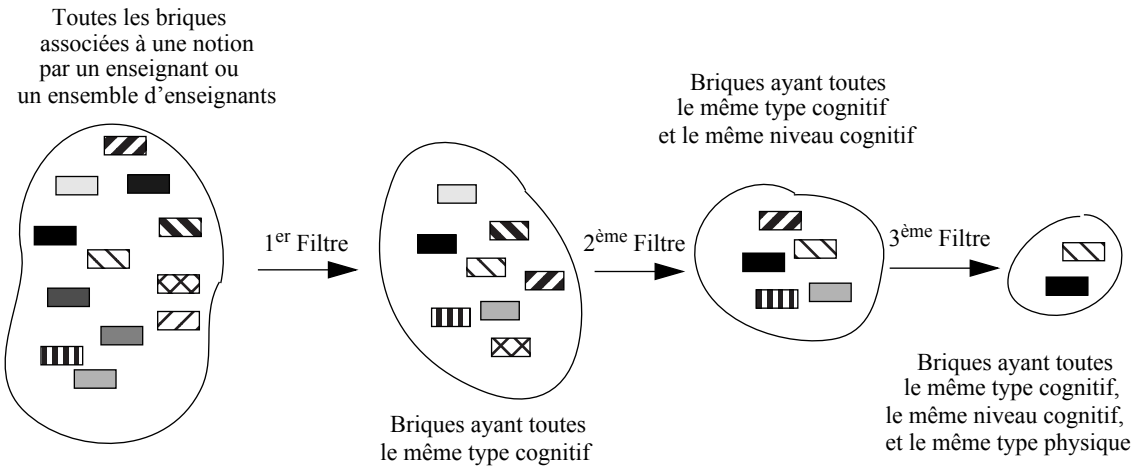


Figure 28 - Extraction de briques élémentaires à l'aide des filtres

Ainsi comme le montre la figure 28, l'utilisation successive de ces trois filtres permet d'extraire "la meilleure brique élémentaire" (si finalement il y en a plusieurs ont choisi une brique qui n'a pas encore été vue par l'apprenant, et s'il les a tous vues au moins une fois, on prend la première). En effet, le canevas détermine le type cognitif que doit utiliser le premier filtre. Ensuite, le modèle épistémique de l'apprenant permet de fixer le niveau cognitif souhaité pour le deuxième filtre. Enfin, l'ordonnancement des types physiques défini dans le modèle comportemental permet de déterminer le type de média que le troisième filtre doit prendre en compte.

Toutefois, il se peut que par manque de briques élémentaires l'utilisation de tel ou tel filtre induise l'obtention d'un ensemble vide en sortie. Dans ce cas, les règles suivantes sont appliquées:

- Si à l'issue du troisième filtre, nous n'obtenons aucune brique, nous élargissons la recherche en prenant en considération l'ordonnancement des types physiques définis dans le modèle comportemental de l'apprenant.
- Si à l'issue du deuxième filtre, nous n'obtenons aucune brique dont le niveau cognitif est équivalent à la connaissance de l'apprenant, nous réitérons l'opération en recherchant des briques dont le niveau cognitif est inférieur au niveau de l'apprenant.
- Si à l'issue du deuxième filtre, nous n'obtenons aucune brique dont le niveau cognitif est inférieur ou égal à la connaissance de l'apprenant, nous réitérons l'opération en recherchant des briques dont le niveau cognitif est supérieur au niveau de l'apprenant.
- Si à l'issue du premier filtre, nous n'obtenons aucune brique, nous réitérons la recherche en prenant un point de vue plus large, c'est-à-dire en ajoutant successivement le point de vue des enseignants du même domaine, puis ceux des champs d'enseignement "supérieur" dans la hiérarchie des domaines d'enseignement.

Ce qui peut se résumer par l'algorithme suivant :

```
Soit notion la notion courante
Soit apprenant l'apprenant courant
Soit note <- Mis à jour de la note de apprenant pour notion
Soit type_prefere <- La liste ordonnee des type physique prefere de apprenant
Pour chaque élément c du canevas de apprenant faire
   Soit enseignants un point de vue du modèle du domaine choisi par apprenant
   Soit trouve <- Faux
   Tant que non Trouve et qu'il reste des enseignants non incluant dans enseignants faire
      Soit m1 <- Recupérer les média pour enseignants de notion de type cognitif c
      Si m1 n'est pas vide alors
         Soit m2_1 <- les média de m1 ayant un niveau cognitif equivalent à note
         Soit m2_2 <- les média de m1 ayant un niveau cognitif inférieur à note
         Soit m2_3 <- les média de m1 ayant un niveau cognitif supérieur à note
         Soit i <- 1
         Tant que i <= 3  et non trouve faire
            Si m2_i n'est pas vide alors
               Soit t <- le premier élément de type_prefere
               Soit encore <- Vrai
               Tant que non encore faire
                  Pour chaque élément m de m2_i faire
                     Si m et t sont de même type physique alors
                           media_choisi <- m
                           trouve <- Vrai
                     Fin si
                  Fin pour
                  Si non trouve alors
                     Si t n'est pas le dernier type préféré de type_prefere alors
                           t <- le type préféré suivant dans type_prefere
                     sinon
                           encore <- Faux
                     Fin si
                  Fin si
               Fin tant que
            Fin si
            Incrémenter i
         Fin tant que
      Fin si
      Si non trouve alors
         enseignants <- agrandire le nombre d'enseignant
      Fin si
   Fin tant que
   Si trouve alors
      Afficher media_choisi
   sinon
      Produire une erreur "Pas assez de media"
   Fin si
Fin pour
```

5.3. La sélection des liens hypertextuels.

Une fois que l'on a construit le contenu d'une page, il faut que le système détermine les liens hypertextes permettant à l'utilisateur d'accéder à d'autres notions. Ici pas de problème, il suffit de prendre en compte :

- les relations du modèle du domaine,
- le modèle épistémique,
- type de cours que désire l'apprenant:
  - en vue d'un examen : dans ce cas on prend le point de vue d'un seul enseignant et on guide fortement l'apprenant en réduisant au maximum le nombre de liens hypertextes : non prise en compte des relations d'analogie et des relations de prérequis dont la pondération est inférieure au niveau de connaissance de l'apprenant.
  - en vue d'un parcours plus libre : dans ce cas l'apprenant peut choisir de suivre le point d'un ou plusieurs enseignants et le système ne guide pas du tout l'apprenant, il affiche tous les liens hypertextes, en spécifiant toutefois leur importance.

Enfin, il faut déterminer le type de page ou de lien hypertexte pour chaque type de relation pour le modèle du domaine. Nous avons décidé de suivre les représentations hypertextuelles suivantes :

Relation du modèle du domaine	Type de lien hypertexte
Prérequis	Page d'index
Analogie	Page d'index
Conjonction	Liens précédent et suivant
Disjonction forte	Page de choix

Tableau 3 - Représentation hypertextuelle des relations du modèle du domaine

Nous verrons dans le chapitre VII comment tout ceci s'organise sur l'écran.

6. Conclusion.

En fait, la figure 29 résume parfaitement le modèle conceptuel de notre système.

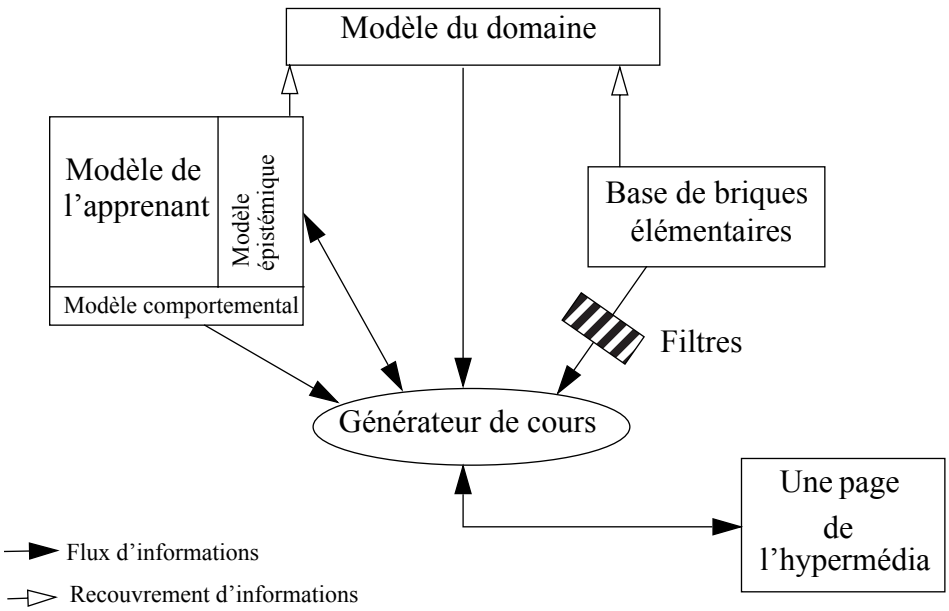


Figure 29 - Modèle conceptuel de METADYNE

Le système est composé de quatre modules :

- un modèle du domaine qui permet de sauver la connaissance pédagogique des enseignants,
- un modèle de l'apprenant, subdivisé un deux sous-modèles (le modèle épistémique et le modèle comportemental), qui permet de connaître les caractéristiques de l'apprenant,
- une base de briques élémentaires qui permet de caractériser les documents didactiques qui vont introduire les notions du modèle du domaine,
- un générateur de cours qui est en charge de construire dynamiquement les pages de l'hypermédia.

Chapitre V - Le modèle objet[32].

Dans le chapitre précédent, nous avons vu que le système s'organise autour de quatre composants principaux que sont le modèle du domaine, le modèle de l'apprenant, la base de briques élémentaires et le générateur de cours. Nous avons également vu comment nous avons décidé de représenter ces composants.

On va maintenant s'intéresser à la représentation informatique de ces modèles à l'aide d'une modélisation objet. Nous allons tout d'abord voir comment on peut modéliser un réseau sémantique extensible et modulable, en nous intéressant premièrement au concept d'étiquette et deuxièmement au concept de relation. Nous verrons ensuite comment le modèle du domaine peut utiliser ce que nous aurons défini. Par la suite, nous appréhenderons le modèle informatique du modèle de l'apprenant. Enfin nous finirons par nous intéresser à la modélisation objet des briques élémentaires.

1. Étiquettes et relations.

Lorsque l'on a choisi de représenter le modèle du domaine à l'aide d'un réseau sémantique et une partie du modèle de l'apprenant (le modèle épistémique) grâce à la technique dite de l'*overlay*, on a pu constater que la grande majorité de l'information, conséquence du choix de ces représentations, n'était pas intrinsèque aux concepts, mais induite par les relations qui les associaient (et aussi les différents acteurs du système, c'est-à-dire les apprenants et les enseignants). Par exemple, le modèle épistémique est représenté à l'aide d'étiquettes pondérées datées associant un apprenant à différents concepts. De même, lorsque l'enseignant décide d'associer deux concepts par une relation de prérequis, on crée en fait une relation pondérée entre ces deux concepts mais qui est associée à l'enseignant créateur.

Dès lors, on s'aperçoit rapidement que tout ce qui est créé est associé à un des acteurs du système, un enseignant ou un apprenant. Lorsque l'acteur est associé à une entité, nous utiliserons le terme d'étiquette (ou *label* en anglais), et lorsqu'il associe deux entités nous parlerons de relation. Nous allons donc maintenant voir comment on peut modéliser tous les types d'étiquette et de relation à l'aide d'une modélisation objet ouverte et facilement implémentable (comme nous le verrons dans le chapitre suivant).

---

[32] Dans ce chapitre, tous les modèles objets utilisent la notation UML.

1.1. Les étiquettes.

L'étiquette est une association entre un acteur du système (représenté à l'aide de la classe *Person*) et une ou plusieurs entités (représentées à l'aide de la classe *Entity*). Nous allons donc représenter ces deux types d'étiquettes à l'aide de deux classes. La première, nommée *UnaryLabel*, qui va associer une personne à une entité à l'aide d'une relation un-aire, et la seconde, nommée *NaryLabel* qui va associer une personne à plusieurs entités grâce à l'utilisation d'une relation n-aire.

Ces classes vont gérer ces relations à l'aide de méthodes. Par exemple la classe *UnaryLabel* va gérer sa relation avec la classe *Entity* à l'aide des méthodes *getEntity*, *setEntity* et *delEntity*. De même, la classe *NaryLabel* va gérer sa relation n-aire à l'aide des méthodes *appendEntity*, *pushEntity*, *getEntities*, *getEntityCardinality*, *delEntity* et *clearEntities*. De plus ces deux classes doivent gérer la relation un-aire avec la classe *Person*, elles vont donc posséder aussi les méthodes *getPerson*, *setPerson* et *delPerson*.

Le problème qui se pose maintenant est la manière d'associer ces différentes classes. Nous allons donc voir les trois façons d'associer ces classes, en écartant tout d'abord les deux premières solutions et en retenant la troisième.

1.1.1. Relation par héritage.

La première solution consiste à considérer la classe *UnaryLabel* comme étant une classe *NaryLabel* spécialisée (Cf. figure 30). Cette solution n'est pas acceptable car dans ce cas les méthodes de la classe *NaryLabel* qui permettent de gérer les relations entre cette classe et la classe *Entity* n'ont pas lieu d'être dans la classe *UnaryLabel*. En effet que signifie par exemple la méthode d'ajout d'une relation (méthode *appendEntity*) lorsque l'on considère que la relation qui associe la classe *UnaryLabel* à la classe *Entity* est une relation un-aire.

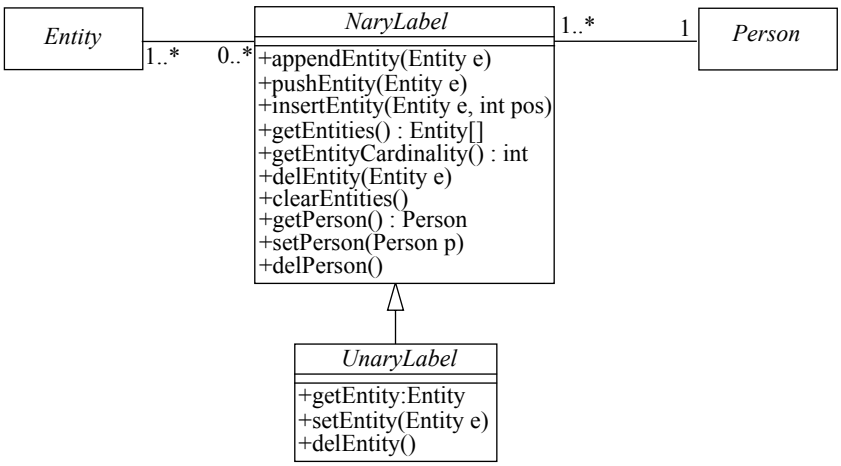


Figure 30 - Relation *UnaryLabel* - *NaryLabel* par héritage

1.1.2. Relation par agrégation.

Cette deuxième solution consiste à considérer que la classe *NaryLabel* est en fait un ensemble de classes *UnaryLabel* (Cf. figure 31).Cette solution bien que supérieure à la précédente pose toutefois le problème de redondance de l'information. En effet, dans ce cas de figure on associe la classe *Person* à la classe *UnaryLabel*. Dès lors pour la classe *NaryLabel* il y aura autant de relations de créées entre les objets de la classe *UnaryLabel* et l'auteur de cette étiquette n-aire, c'est-à-dire instance de la classe *Person*, que la cardinalité de l'instance de la classe *NaryLabel*.

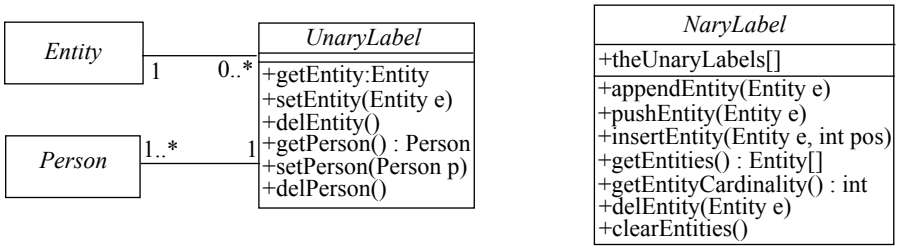


Figure 31 - Relation *UnaryLabel* - *NaryLabel* par agrégation

1.1.3. Relation par classe mère.

Cette dernière solution, que l'on a choisie, consiste à déplacer la relation qui associe une personne à une étiquette sur une classe mère, nommée *Label*, et de définir deux sous-classes distinctes *UnaryLabel* et *NaryLabel* où chacune de ces classes est en charge de la relation un-aire ou n-aire avec la classe *Entity*. Ce qui nous donne le schéma suivant :

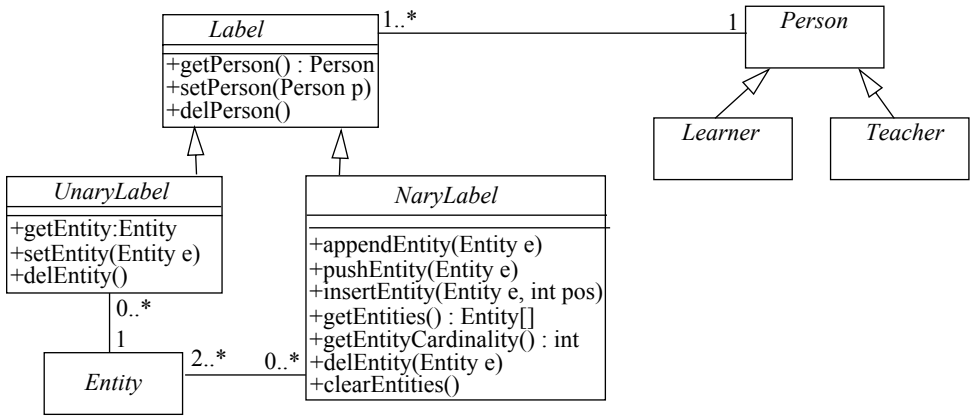


Figure 32 - Relation *UnaryLabel* - *NaryLabel* retenue

Toutefois, le modèle des étiquettes ne peut pas se résumer aux classes précédentes, car on a besoin d'étiquettes particulières. Par exemple on a besoin d'étiquettes qui peuvent être pondérées, datées ou les deux. Ainsi on peut améliorer le modèle de la façon suivante :

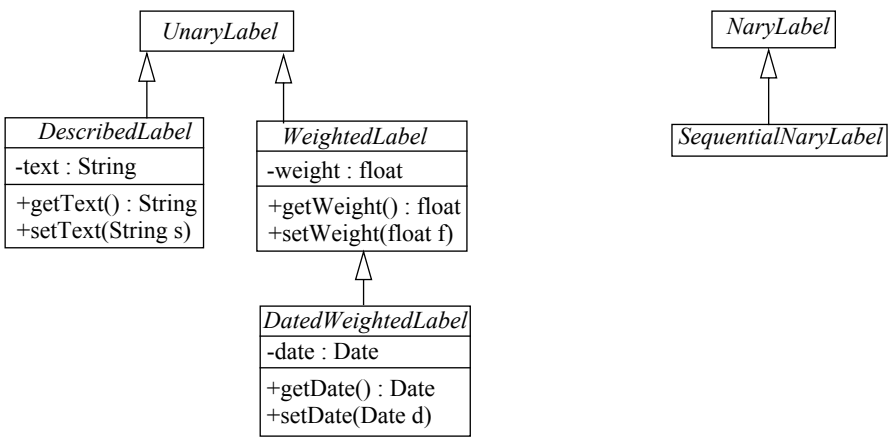


Figure 33 - Modélisation objet des étiquettes

Dès lors, comme nous allons le voir dans les modèles objets de nos représentations du modèle du domaine et du modèle de l'apprenant, pour créer une étiquette spécifique, il suffira de créer une sous-classe des classes présentées dans le modèle ci-dessus, et d'ajouter dans les classes *Learner* ou *Teacher*, et *Entity* les méthodes de gestion de cette étiquette.

1.2. Les relations.

Outre les étiquettes, nous avons vu que les relations ont une importance primordiale dans notre système.

Mais qu'est ce qu'une relation ?

Une relation est une liaison qui associe une entité source à une ou plusieurs entités destinations et qui est crée par un acteur du système. Par conséquent, une relation peut être considérée comme une spécialisation d'une étiquette un-aire (sous-classe de la classe *UnaryLabel*). En effet, sachant qu'une étiquette un-aire associe une personne à une entité, on peut considérer cette dernière comme étant la source d'une relation. Il suffit donc de définir dans une sous-classe de la classe *UnaryLabel*, le ou les entités destination.

De ce fait, tout comme pour les étiquettes, nous allons distinguer les relations un-aires (*UnaryRelation*) qui ont une seule entité destination, et les relations n-aires (*NaryRelation*) qui ont plusieurs destinations. De plus, on a vu dans les différents modèles du modèle conceptuel que l'on a besoin de relations pondérées (*WeightedUnaryRelation*), de relations ordonnées (*SequentialNaryRelation*), et de relations pondérées datées (*DatedWeightedUnaryRelation*).

Remarque : La relation entre les classes *UnaryRelation* et *NaryRelation* est identique à celle choisie pour les classes *UnaryLabel* et *NaryLabel*. Par conséquent nous ne la détaillerons pas dans ce paragraphe.

Ce qui nous donne le modèle objet suivant :

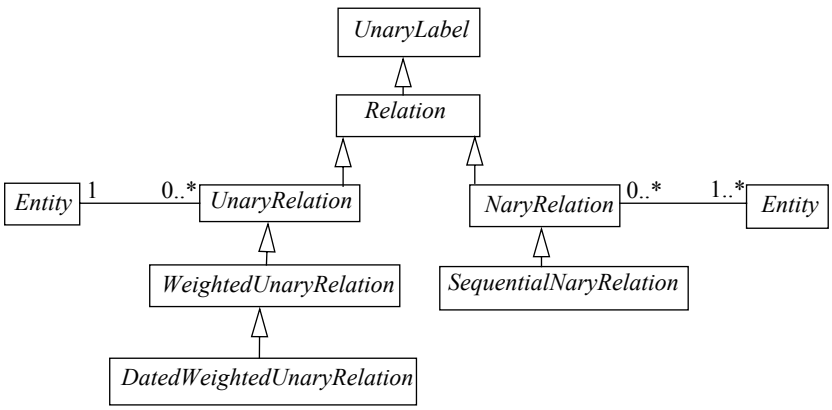


Figure 34 - Modélisation objet des relations

Ainsi, pour créer une relation, il suffit de créer une sous-classe d'une des classes présentées ci-dessus et d'ajouter à la classe *Person* et/ou à la classe *Entity*, ou plus exactement à une de ses sous-classes, les méthodes adéquates d'accès à la relation (création, ajout, suppression, etc.).

Nous allons maintenant voir comment à partir de ces classes nous avons spécifié les modèles objets du modèle du domaine, du modèle de l'apprenant et des briques élémentaires.

2. Le modèle du domaine.

Reprenons la représentation conceptuelle de notre modèle du domaine :

Le modèle du domaine est un réseau sémantique où les noeuds de ce réseau sont des concepts. Ces concepts sont reliés entre eux par quatre types de relations (les relations d'aide à la compréhension, de prérequis, de composition conjonctive et de composition disjonctive forte).

Chaque concept est identifié par un nom (ou plus exactement par une chaîne de caractères). Cet attribut est unique et universel, tous les acteurs du système le "voient". Par contre, on peut adjoindre différentes définitions à un concept (une définition au maximum par enseignant). Ensuite ces concepts peuvent être reliés par des relations propriétaires (propriété d'un enseignant). Enfin chaque concept est associé à un ou plusieurs champs d'enseignement (en fait un champ d'enseignement par enseignant associé à ce concept).

Ainsi, un concept doit être défini comme étant une sous-classe de la classe *Entity*, avec comme attribut : son nom. On peut lui associer des définitions (une par enseignant), qui sont alors les instances d'une classe sous-classe de la classe *DescribedLabel* : la classe *Definition*. Ensuite les champs d'enseignement (instance de la classe *Field*), organisés en arbre (un champ d'enseignement peut avoir un sur-champ, et peut avoir des sous-champs), sont aussi source de relations, donc sous-classe de la classe *Entity*. Enfin les relations sont définies comme suit :

- Les relations d'aide à la compréhension (*HelpRelation*) et de prérequis (*PrerequisiteRelation*) sont des relations un-aires pondérées (suivant l'importance que donne l'enseignant à ces relations). Ce sont donc des sous-classes de la classe *WeightedUnaryRelation*.
- La relation de composition disjonctive forte (*DerivationRelation*) associe un concept à plusieurs concepts, c'est donc une relation n-aire. Par conséquent cette relation est représentée par une sous-classe de la classe *NaryRelation*.
- La relation de composition conjonctive (*CompositionRelation*) associe un concept à plusieurs concepts en respectant un ordre de priorité défini par l'enseignant propriétaire de cette relation. Il s'agit donc d'une relation n-aire séquentielle, donc on peut la représenter à l'aide d'une classe sous-classe de *SequentialNaryRelation*.
- La relation qui associe un concept à un champ d'enseignement par enseignant, associé au concept étudié (*FieldRelation*), peut être représentée à l'aide d'une relation n-aire, et donc instance d'une sous-classe de la classe *UnaryRelation*.

Ce qui peut se résumer par le modèle objet suivant :

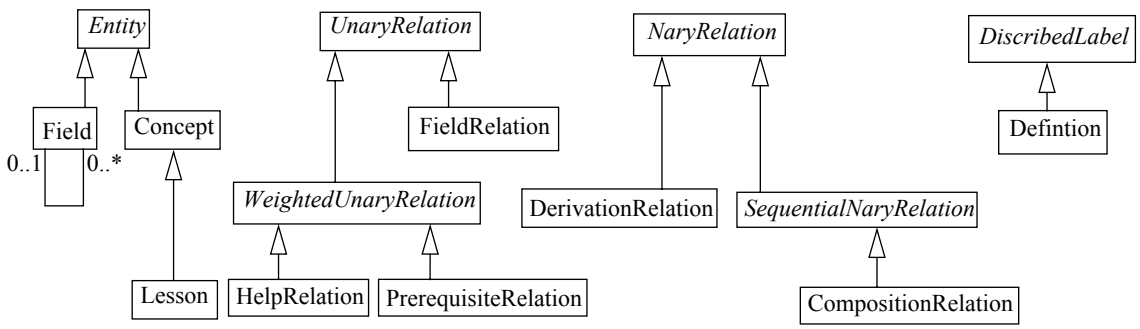


Figure 35 - Modélisation objet du modèle du domaine

3. Le modèle de l'apprenant.

Nous avons vu dans le chapitre précédent que le modèle de l'apprenant est en fait composé de deux sous-modèles, le modèle épistémique et le modèle comportemental. Etudions donc maintenant le modèle objet de la représentation de ces deux sous-modèles.

3.1. Le modèle épistémique.

Nous avons vu que nous avions décidé de représenter le modèle épistémique par la technique dite de l'*overlay*, c'est à dire en associant à chaque concept du modèle du domaine une pondération. Ces pondérations peuvent prendre deux formes distinctes, soit une valeur particulière indiquant que le système n'a pas connaissance du niveau de l'apprenant sur le concept correspondant, soit une valeur proportionnelle au niveau de connaissance de l'élève (qu'il aura déterminée, ou qui sera issue des résultats précédents). Enfin, cette dernière valeur est associée à une date, afin de prendre en compte les phénomènes d'oubli.

Le modèle objet du modèle conceptuel est dès lors implicite. Il suffit en effet de créer une nouvelle classe (que l'on a nommé *Mark*) sous-classe de la classe *DatedWeightedLabel* comme le montre le modèle suivant :

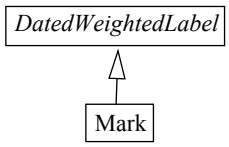


Figure 36 - Modèle objet du modèle épistémique

Remarque : La valeur particulière indiquant que le système n'a pas de connaissance du niveau de l'apprenant sur un concept sera ici représentée par l'absence de cette relation entre l'apprenant et le concept.

3.2. Le modèle comportemental.

Ce deuxième sous-modèle de l'apprenant va permettre de mettre en forme les cours qui vont être présentés à l'apprenant. C'est ce modèle qui donne l'aspect adaptable du système puisque c'est l'étudiant qui détermine les caractéristiques de ce modèle. Ces caractéristiques sont :

- la structure générale d'un cours, que l'on a nommé dans le chapitre précédent le canevas. Cette composante va donc relier de façon ordonnée un apprenant à différents types cognitifs. Il faut donc tout d'abord que cette composante soit représentée à l'aide d'une étiquette n-aire ordonnée, c'est à dire une classe nommée *Caneva*, sous-classe de la classe *SequentialNaryRelation*. Ensuite, les types cognitifs, pour être destination d'une étiquette, doivent être représentés par une sous-classe de la classe *Entity* que l'on a nommée *CognitiveType*.
- l'importance du type physique des média. L'apprenant a la possibilité de déterminer un ordre de préférence concernant l'aspect physique des éléments multimédia qui vont lui être présentés. Ainsi nous pouvons représenter cette composante via une étiquette n-aire ordonnée, c'est-à-dire par une classe nommée *PreferedPhysicalMediaType*, sous-classe de la classe *SequentialNaryRelation*. Pour que cette étiquette puisse relier un apprenant aux différents types physiques du système, il faut enfin que ces derniers soient des instances d'une classe, que l'on a nommée *PhysicalType*,

sous-classe de la classe *Entity*.

- l'importance des champs d'enseignement. On a en effet vu dans le chapitre précédent que l'actualisation du modèle épistémique de l'apprenant doit dépendre des capacités de l'apprenant dans le domaine concerné. Il faut donc que l'apprenant puisse spécifier dans quel domaine il se sent plutôt bon, dans quel domaine il se sent moyen et dans quel domaine il se sent mauvais. Nous pouvons obtenir cette information en associant l'apprenant et les champs d'enseignement par des étiquettes pondérées, donc par des instances de la classe que l'on a nommées *PreferedField* sous-classe de la classe *WeigthedLabel*.
  Sachant qu'il n'est pas réalisable de demander à chaque apprenant d'associer une valeur à chaque champ d'enseignement, il faut absolument conserver la propriété de hiérarchisation des champs que nous avons spécifiés lors de la présentation du modèle objet du modèle du domaine. Ainsi, si on a besoin de connaître la valeur associant un apprenant A à un champ d'enseignement C, et que cette dernière n'existe pas, on prendra la valeur associant A au sur-champ d'enseignement de C. Si toutefois on remonte ainsi jusqu'à la racine, on admet alors que l'apprenant est moyen dans ce champ.

Ce qui peut être résumé par le modèle objet suivant :

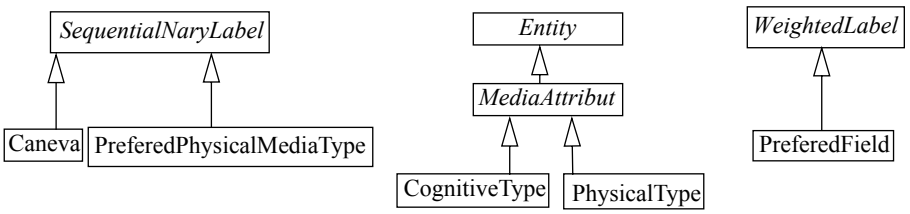


Figure 37 - Modèle objet du modèle comportemental

Remarque : Sachant que les classes *CognitiveType* et *PhysicalType* sont des attributs des briques élémentaires (que nous allons voir tout de suite), nous les avons regroupées afin qu'elles soient sous-classes d'une classe abstraite *MediaAttribut*.

4. Les briques élémentaires.

La dernière composante du modèle conceptuel que l'on doit modéliser est tout ce qui se rapporte aux briques élémentaires. Nous avons bien précisé dans le chapitre précédent que le stockage de ces briques n'est pas la problématique de notre système, mais il faut pouvoir les référencer.

Toutefois, lors de l'élaboration de Metadyne nous avons décidé d'inclure les QCME[33], au sein du système en considérant que ces QCME sont des briques élémentaires particulières.

Pourquoi ce choix ?

En fait, les QCME sont les seuls outils permettant d'évaluer les apprenants et de ce fait, de mettre à jour le modèle épistémique de ces derniers. Il faut donc qu'il y ait une connection entre le

[33] Questionnaire à choix multiple étendu.

serveur Metadyne et ces QCME, ce qui n'est envisageable que s'ils font partie intégrante du système.

Nous allons donc maintenant voir comment nous faisons référence aux briques élémentaires de façon générale, puis nous étudierons plus particulièrement la modélisation des QCME.

4.1. Cas général.

Les briques élémentaires, comme nous l'avons vu dans le chapitre précédent, sont caractérisées par trois attributs : leur type cognitif, leur niveau cognitif et leur type physique.

Le type cognitif et le type physique sont intrinsèques au média qui est référencé par la brique (une définition restera toujours une définition, et une image restera toujours une image). Par contre, le niveau cognitif dépend de l'enseignant qui a voulu associer cette brique à un certain concept. De ce fait comme nous l'avons vu précédemment, le type cognitif et le type physique sont représentés par deux classes nommées *CognitiveType* et *PhysicalType*, alors que le niveau cognitif est représenté à l'aide d'une relation pondérée associant un concept à une brique élémentaire, nommée *CognitiveLevelRelation*.

Ces caractéristiques étant communes à toutes les briques élémentaires (donc aussi au QCME), on les retrouve au sein de la sur-classe, nommée *Media*, de la classe des briques élémentaires à proprement parlé, la classe *ElementaryBrick*, et de la classe des QCME, la classe *ECMQ*.

Ceci peut être résumé par le schéma suivant :

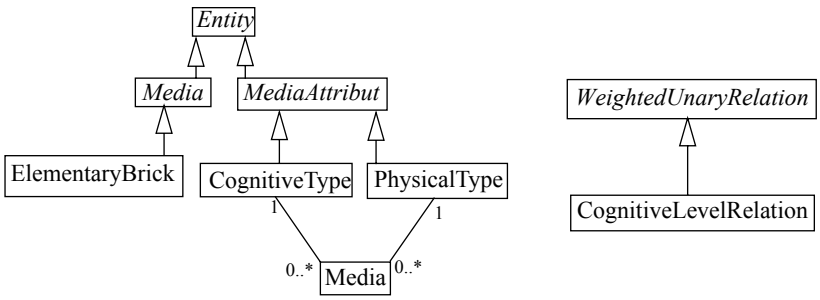


Figure 38 - Modèle objet des briques élémentaires

4.2. Le cas des QCME.

Les QCME se distinguent des QCM classiques par le fait qu'une mauvaise réponse de la part de l'apprenant peut entraîner une question supplémentaire. Ceci permet alors de préciser le type d'erreur qu'a effectué l'apprenant et ainsi de mieux l'évaluer.

Un QCME peut alors être vu comme un arbre, comme par exemple celui présenté par la figure 39).

On remarque alors que :

- un QCME est composé de plusieurs questions principales ordonnées,
- chaque question admet différentes réponses qui sont soit vraies, soit fausses.
- une réponse fausse peut induire une nouvelle question.

Dès lors un QCME peut être représenté par un objet de classe *ECMQ*. Une relation ordonnée doit alors relier cette instance aux différentes questions (principales) instances de la classe *Question*. Une question est alors reliée simplement à au moins deux réponses. Une réponse fausse pouvant entraîner une nouvelle question, chaque réponse fausse peut être reliée à une nou-

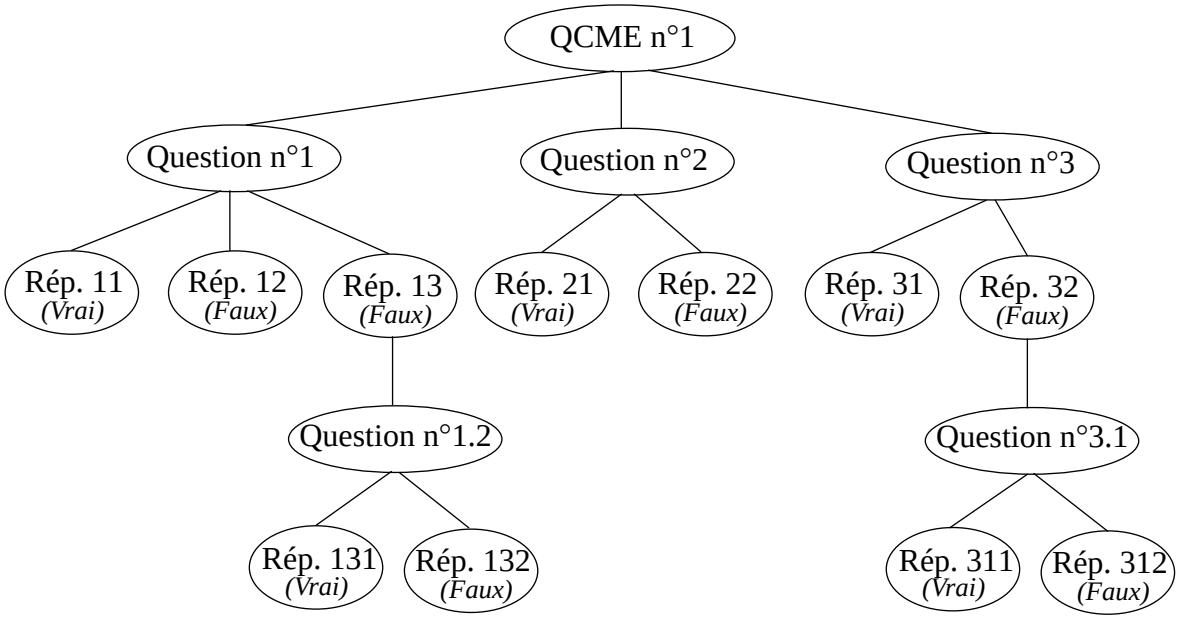


Figure 39 - Exemple théorique de QCME

velle question. Enfin, les questions pouvant être multimédia, chaque question peut être reliée à une brique élémentaire.

Ainsi, on obtient le modèle objet suivant :

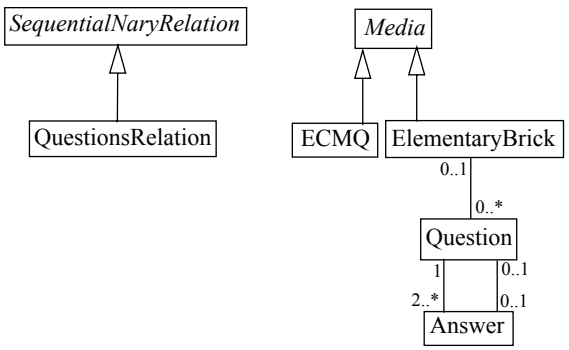


Figure 40 - Modèle objet des ECMQ

5. Conclusion.

Ce chapitre a défini la représentation informatique, à l'aide d'un modèle objet, des différents modèles conceptuels que nous avons vus au chapitre précédent. Ce modèle objet est basé sur une définition précise et une utilisation intensive de deux classes principales. Tout d'abord la classe *Label* qui permet d'associer un utilisateur du système à une entité. Puis la classe *Relation*, sous-classe de la classe *Label*, qui permet à un utilisateur du système de relier deux entités.

Le figure 41 représente le modèle objet dans son ensemble. Les classes terminales, non abstraites, de par leur couleur, peuvent être associées à chaque modèle conceptuel.

Dans le chapitre suivant, nous allons voir comment nous pouvons implémenter cette modélisation à l'aide d'une architecture client-serveur totalement objet.

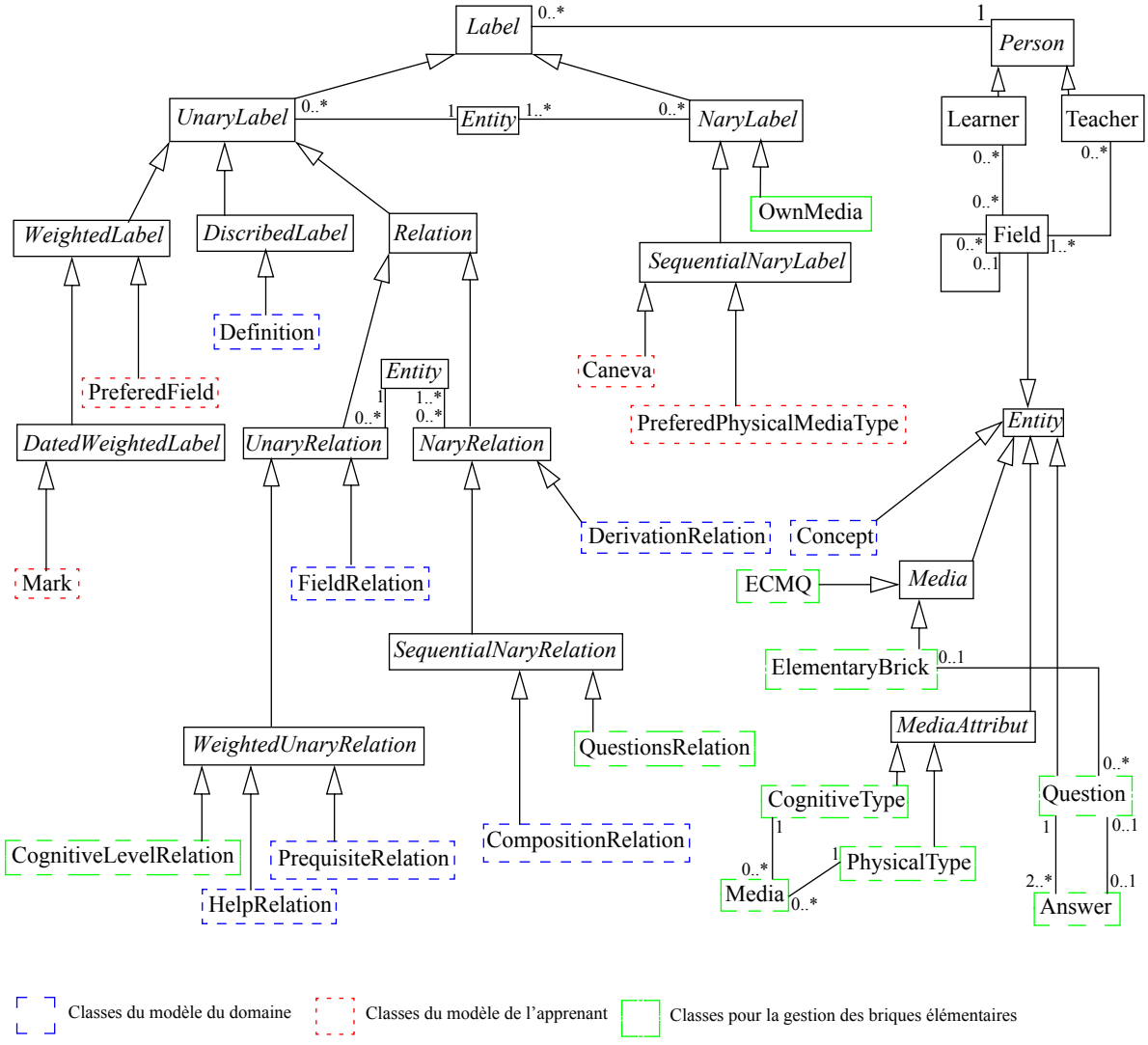


Figure 41 - Modèle objet de Metadyne

Chapitre VI - L'architecture Client-Serveur[34].

L'objectif de ce chapitre est de présenter les outils et méthodes mis en œuvre pour réaliser le prototype Metadyne.

Après un rappel sur les différentes architectures pouvant être utilisées pour développer des sites Internet, nous allons étudier l'architecture que nous avons retenue. Nous analyserons cette architecture en nous focalisant tout d'abord sur l'utilisation d'une base de données et plus particulièrement sur l'utilisation de la base de données orientée objet Matisse. Nous verrons ensuite comment un langage objet peut transformer des objets transients en objets persistants grâce à l'utilisation d'une base de données. Pour finir, nous nous intéresserons alors à la communication entre le serveur Metadyne et les clients grâce aux technologies d'objets distribués.

1. L'architecture logicielle.

1.1. Quelques rappels au sujet d'Internet.

Internet est né en 1969 sous l'impulsion du département américain de la défense (DOD). Le réseau, qui s'appelait alors ARPANET, devait assurer les échanges d'informations électroniques entre les centres névralgiques américains dans le contexte de la guerre froide. Le cahier des charges, établi par le DOD, imposait que le réseau puisse poursuivre ses activités en cas d'attaque nucléaire soviétique. Si l'un ou plusieurs des sites et lignes de connexion venaient à être détruits, les messages parviendraient à leur destinataire par des itinéraires alternatifs. Un grand nombre de centres de recherche, militaires, publics et privés prirent part à ce projet. Il était normal que leurs réseaux internes furent les premiers reliés à Internet. C'est pourquoi, dès sa création, Internet fut un méta-réseau, un réseau de réseaux, qui relia peu à peu la communauté scientifique et universitaire mondiale.

Internet arriva en Europe en 1982. L'année 1984 fut une année charnière : Internet perd son caractère militaire. Son financement n'était plus assuré par le DARPA mais par un organisme scientifique civil créé deux ans plus tard : La *National Science Foundation* (NSF). Le réseau fut alors scindé en deux parties:

- MILnet, réseau strictement militaire
- NSFnet, le *backbone* ou épine dorsale d'Internet.

[34] Dans ce chapitre, sauf contre-indication, tous les modèles objets utilisent la notation UML

En 1989 est apparu le World Wide Web (WWW). Ce projet fut dévéloppé au CERN (Centre d'Etude et de Recherche Nucléaire) à l'instigation de Tim Berners-Lee et Robert Cailleau. Basé sur les techniques hypertextes, de par sa simplicité d'utilisation, le WWW contribua grandement à populariser les autoroutes de l'information.

1.2. Architecture classique client/serveur Web.

L'architecture classique client/serveur Web peut être décomposée en trois parties distinctes :

- Les fichiers HTML[35] contenant les informations qui vont être présentées à l'utilisateur.
- Le serveur HTTP[36], qui réceptionne les requêtes des clients, sélectionne le bon fichier (HTML, image, etc.) et le renvoie au client en le préfixant d'une information appelée "type mime".
- Le client qui est en fait un programme spécifique, appelé navigateur. A l'aide de ce navigateur, l'utilisateur active des liens hypertextes. Ces actions sont alors interprétées comme des requêtes et par conséquent transmises au serveur HTTP destinataire.

Cette architecture peut être résumée par la figure suivante :

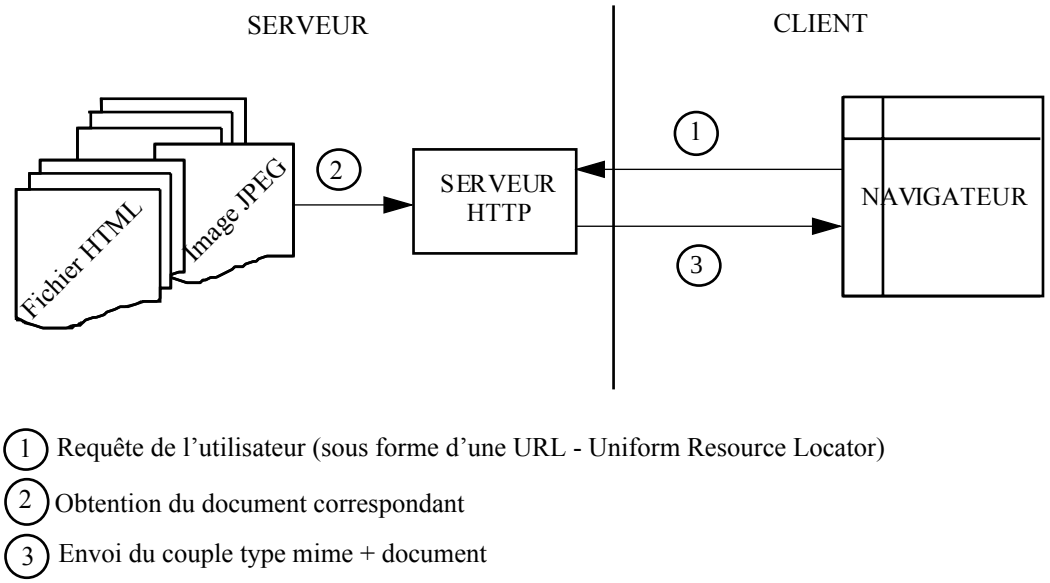


Figure 42 - Architecture client/serveur web

Bien entendu ceci n'est qu'une vision simpliste du fonctionnement du World Wide Web, si vous le désirez, vous pourrez trouver de plus amples informations dans [Vigne 99].

1.3. Utilisation des programmes CGI.

L'architecture que nous venons de voir brille par sa simplicité de fonctionnement et de mise en œuvre, surtout avec les outils graphiques et WYSIWYG (*What You See Is What You Get*) d'aujourd'hui, mais elle a le gros défaut d'être très statique, très figée. En effet, tout d'abord les informations qui sont présentées à l'utilisateur sont stockées sous forme de fichier (avec tous les inconvénients que cela entraîne : problème d'indexation, de maintenance, de sécurité, d'harmonie ergonomique, etc.). Ensuite l'utilisateur n'a que les liens hypertextes (ou URL[37] saisie

35. Hypertext Markup Language.
36. Hypertext Transfert Protocol.

manuellement) pour effectuer des requêtes auprès des serveurs.

Les programmes CGI[38] sont alors apparus au début des années 90. Ce sont en fait des programmes qui récupèrent la requête émise par l'utilisateur, effectuent un traitement et retournent une information au client (cette information est le plus souvent une page HTML, qui est créée ou sélectionnée par le programme CGI). De plus, la requête transmise au serveur peut être assez complexe, ce n'est plus seulement une URL classique, mais des informations peuvent l'accompagner. Ces informations peuvent provenir des champs d'un formulaire (que l'utilisateur aura au préalablement remplis), des *cookies* (données sauvegardées par certains sites) ou bien de fonctions Javascript[39].

Ainsi comme le montre la figure suivante, lorsqu'un serveur HTTP reçoit une requête, il détermine la nature de cette dernière :

- soit cette requête est classique (elle correspond à un fichier ordinaire). Dans ce cas on se trouve dans la situation du paragraphe 1.2,
- soit cette requête correspond à l'exécution d'un programme CGI. Dans ce cas, le serveur exécute ce dernier, qui doit retourner une information équivalente à une requête classique.

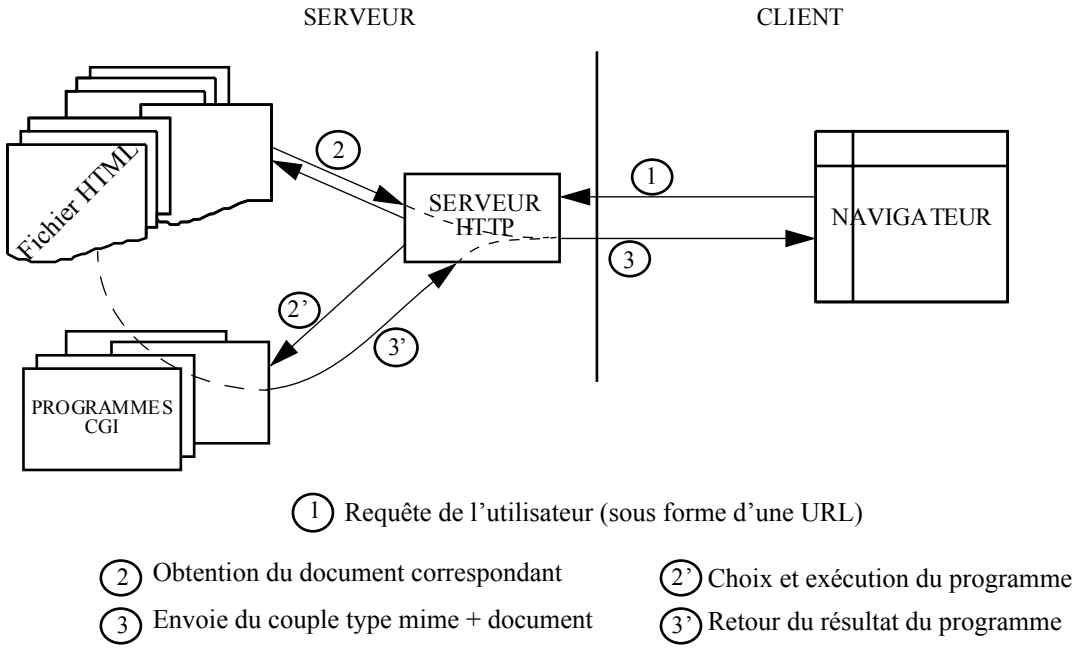


Figure 43 - Architecture client/serveur Web utilisant des programmes CGI

1.4. Utilisation des "servlets".

Les *servlets* peuvent être vues comme une remise au goût du jour de l'architecture incluant des programmes CGI. En effet, les services rendus par les *servlets* sont équivalents à ceux rendus par les programmes CGI, mais les *servlets* tirent parti du langage qui leur a donné le jour, c'est-à-dire le langage JAVA. Les *servlets* ont entre autres les trois avantages suivants sur les programmes CGI :

37. Uniform Ressource Language
38. Common Interface Gateway
39. Langage intialement proposé par Netscape qui permet d'inclure des fonctions au sein de pages HTML qui sont interprétées par le Navigateur

- Tout d'abord, alors qu'un programme CGI est un processus qui est lancé à chaque requête reçue, une *servlet* est un processus qui est persistant au niveau du serveur (il peut être lancé lors du lancement du serveur HTTP, ou lors de la réception de la première requête).
- Ensuite les *servlets* lancent un *thread* (processus fils) par requête, alors que les programmes CGI lancent des processus indépendants. Cela permet par exemple de pouvoir créer une mémoire commune à toutes les requêtes, ou encore permettre aux *threads* de communiquer entre eux.
- Enfin, le langage JAVA est un langage orienté-objet, alors que les programmes CGI sont le plus souvent programmés en C ou perl. De plus JAVA est un langage multi-plateforme. Par exemple, une *servlet* écrite sous un environnement Windows NT, fonctionnera parfaitement sous un environnement UNIX.

L'architecture d'un serveur Web utilisant les *servlets* peut être résumée par le schéma suivant :

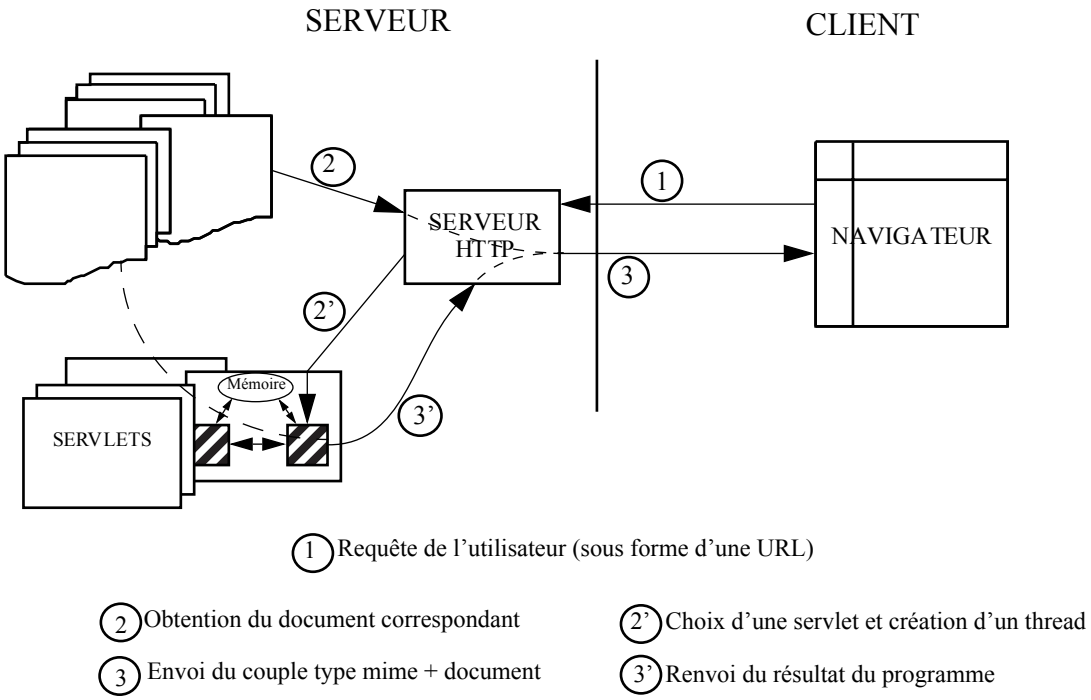


Figure 44 - Architecture client/serveur Web utilisant des *servlets*

1.5. L'architecture logicielle de METADYNE.

Malheureusement, les différentes architectures que nous venons de voir ne répondent pas aux impératifs de Metadyne. En effet, au niveau fonctionnel, les exigences de notre prototype sont les suivantes :

- Tout d'abord nous devons fournir aux apprenants des cours hypermédia, ce qui ne peut convenablement être obtenu que par l'utilisation des navigateurs Web.
- Ensuite, nous devons fournir des outils pour la gestion du modèle du domaine et pour l'évaluation des apprenants. Dans ce cas les capacités IHM (Interface Homme Machine) du HTML (même combiné à du Javascript) sont insuffisantes. De plus, il faut dans ce cas que l'outil ait une certaine autonomie (chaque action de l'utilisateur ne doit pas engendrer une requête au niveau du serveur). Dès lors, seule l'utilisation de véritables applications est envisageable.

- Enfin, nous avons besoin de sauvegarder les différents modèles (modèle du domaine et modèle de l'apprenant). Nous allons donc utiliser une base de données. Or l'accès à cette base de données doit être possible aussi bien depuis le serveur HTTP, que depuis les applications qui vont s'exécuter au niveau des clients.

Tout ceci implique donc :

- au niveau du client, d'avoir une cohabitation entre un navigateur et quelques applications,
- au niveau du serveur, d'avoir outre le serveur Web, un outil de communication et de persistance adapté.

Ce qui nous a amené à concevoir l'architecture suivante :

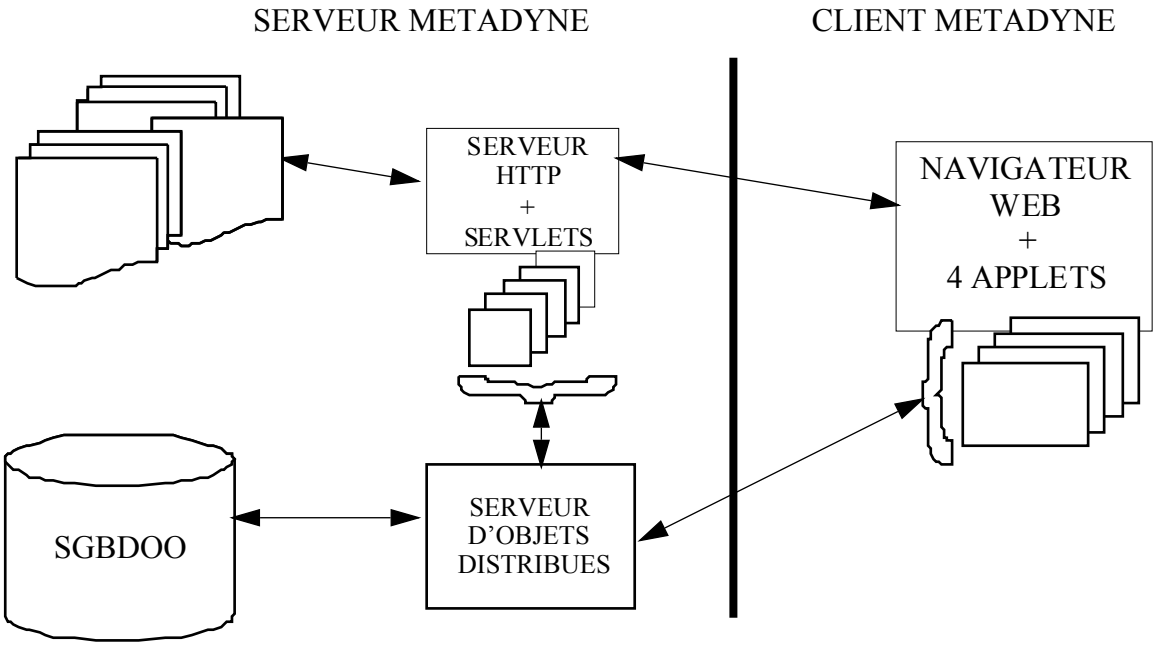


Figure 45 - Architecture logicielle de Métadyne

Etudions cette architecture plus en détail :

- Au niveau du serveur Métadyne, nous trouvons trois composants majeurs :
  1. Un système de gestion de base de données (SGBD) qui permet de stocker toutes les informations que nous voulons persistantes. Nous allons voir dans la deuxième partie de ce chapitre comment et pourquoi nous avons choisi la base de données Matisse.
  2. Un serveur HTTP associé à des *servlets*. Ce serveur est en charge de construire les cours qui sont présentés à l'apprenant. Pour se faire il peut communiquer avec la base de données (via le serveur d'objets distribués) et avec l'*applet* fonctionnant sur le poste client.
  3. Un serveur d'objets distribués qui permet aux *servlets* et aux *applets* tournant sur les différents clients de pouvoir accéder à la base de données. Nous verrons dans la troisième partie de ce chapitre quels sont les protocoles et outils de distribution d'objets existants sur le marché et pourquoi notre choix s'est arrêté sur HORB.
- Au niveau du client Métadyne, les navigateurs Internet peuvent exécuter quatre *applets* :

1. Une *applet* pour la gestion du modèle du domaine, permettant à chaque enseignant de pouvoir modifier à loisir sa vision du modèle du domaine.
2. Une *applet* qui, en collaboration avec le serveur HTTP, est en charge de construire et d'afficher les cours.
3. Deux *applets* pour la construction (par les enseignants) et l'utilisation (par les étudiants) des QCME[40], qui sont en charge d'évaluer les étudiants.

Les avantages de cette architecture sont multiples, entre autres :

- Nous sommes dans un environnement "tout objet", tant au niveau du client (grâce à l'utilisation des *applets*), qu'au niveau serveur, de par les *servlets* du serveur HTTP, de par le serveur d'objets distribués, et par la base de données. Ce choix permet d'implémenter simplement le modèle objet que nous avons vu dans le chapitre V de ce document, et les extensions futures de Métadyne en seront grandement facilitées.
- L'utilisation des *applets* au niveau du client permet de proposer aux différents utilisateurs du système de manipuler les données de façon souple, ce qui est impensable avec les pages HTML, et très difficile avec l'utilisation des scripts au sein du navigateur comme nous le verrons dans l'implémentation du serveur SEMUSDI.

Remarque : Il est important de comprendre que le serveur Métadyne est composé d'une chaîne de client/serveur à trois maillons.

- Le premier maillon est le SGBD. Comme il se doit, le SGBD que nous avons choisi dispose d'une architecture client/serveur.
- Le deuxième maillon est le serveur d'objets distribués, qui est le client du premier maillon, c'est à dire le SGBD.
- Le troisième maillon est le serveur HTTP, dont une de ses composantes (les *servlets*) est un des clients du serveur d'objets distribués (les autres clients étant les *applets* qui s'exécutent à un instant *t* sur les postes clients du serveur Metadyne).

Nous allons maintenant nous intéresser tout particulièrement aux deux premiers maillons, le SGBD et le serveur d'objets distribués.

2. La base de données.

Dans l'élaboration du prototype Metadyne, nous avons besoin de sauvegarder, de rendre persistant, les objets créés par les différents utilisateurs du système. On a en effet besoin de sauvegarder les différentes vues du modèle du domaine, les caractéristiques et résultats des étudiants et les informations sur les briques élémentaires. L'utilisation d'une base de données semble alors naturelle, puisque c'est sur cet outil que l'on va déporter tous les problèmes de concurrence d'accès, de réplication de l'information et de recherche efficace d'informations précédemment sauvegardées.

Les bases de données[41] les plus répandues sont les bases de données relationnelles. Apparues dans les années 80, elles reposent sur le modèle relationnel [Codd 70]. Le modèle relationnel lui-même repose sur la théorie des ensembles et fournit un langage ensembliste pour manipuler les relations ou les tables de la base de données : SQL est le plus connu de ces langages.

[40] Questionnaire à Choix Multiples Etendus

[41] Dans cette partie, nous allons utiliser indifféremment les termes "base de données" et "système de gestion de bases de données". Toutefois, nous ne nous sommes arrêtés qu'à l'étude de véritables SGBD, c'est à dire les SGBD disposant d'une architecture client/serveur.

Toutefois les bases de données relationnelles répondent mal à certains besoins. En effet, le plus souvent, elles ne savent gérer que des types simples (entier, réel, chaîne de caractères, etc .). De plus lorsqu'elles ont les capacités de gérer des types plus complexes, elles le font par l'intermédiaire de tableau d'octets : les Blobs (Binary Large OBject). Enfin, alors qu'aujourd'hui les modélisations sont orientées objets (OMT, UML, etc.), ainsi que les langages d'implémentation tels que C++, Smalltalk, CLOS, ou encore Java (avec il est vrai des degrès divers), il est dommage de ne pas utiliser des outils de persistance objets.

Nous avons donc décidé d'utiliser une base de données objets, mais laquelle choisir ?

2.1. Les bases de données objets.

Reprenons la définition d'une base de données dans [Bouzeghoub & al 97] :

> *"Une base de données est une organisation cohérente de données permanentes et accessibles par des utilisateurs concurrents."*

Donc une base de données objets est une organisation cohérente d'objets persistants et partagés par des utilisateurs concurrents. Dès lors une base de données objets s'appuie sur deux technologies : la technologie des bases de données et la technologie objet. Par conséquent deux méthodes permettent d'obtenir le résultat voulu :

- Partir d'une base de données relationnelle et insérer de la technologie objet, c'est ce que l'on nomme les bases de données relationnelles-objets.
- Partir des concepts objets et insérer des mécanismes de persistance efficaces, c'est ce que l'on nomme les bases de données orientées-objet.

Examinons ces deux points de vues.

2.1.1. Les bases de données relationelles-objets.

Cette démarche est à la base de la spécification du langage SQL 3, et c'est la démarche suivie par les grands constructeurs de base de données relationnelles, tel qu'ORACLE (à partir de la version 8). Dans les SGBDRO, le concept de classe et d'objet reposent sur les types abstraits de données (les TAD) et sur les tables objet-relationnelles.

On peut définir les TAD comme étant un nouveau type de données, ou plus exactement une structure, qui est explicitée par l'utilisateur. Cette structure peut alors être réutilisée dans différentes tables. En plus de cette structure, les TAD intègrent le concept d'encapsulation (le corps des méthodes sont alors écrites en PL/SQL), et le concept d'héritage (Cf. figure 46, [Soutou 99]).

Les tables objet-relationnelles, quant à elles, sont des tables définies exclusivement à l'aide de TAD. C'est à travers ces tables que l'utilisateur du SGBDRO a la possibilité de créer des instances de TAD. Et dans un SGBDRO seuls ces éléments ont OID (un identifiant d'objet), ce qui leur permet d'être référencés via un simple pointeur [Soutou 99]).

```
CREATE TYPE Personne AS OBJECT
(nom VARCHAR(10),
prenom VARCHAR(5),
pere REF Personne,
mere REF Personne
)

CREATE TABLE personnes of Personne;
```

Figure 46 - Exemple de création d'un TAD et d'une table objet-relationnelle sous Oracle 8

En conclusion, bien que le concept d'objet apparaisse effectivement dans les SGBDRO, l'utili-

sation de table est obligatoire. Bien que cela permette d'effectuer une migration souple des bases de données relationnelles vers le concept d'objet, on ne voit pas très bien l'intérêt de les utiliser lorsque toute la conception d'un projet est objet.

2.1.2. Les bases de données orientées-objets.

Il y a deux façons de créer une base de données orientée-objet :

- Partir d'un langage objet et implémenter des paquetages de gestion de persistance. C'est la solution utilisée par ObjectStore avec le langage C++ ou par GemStone avec le Smalltalk.
- Partir d'un modèle objet pur, indépendamment de tout langage, et définir des APIs [42] permettant d'attaquer ce modèle. C'est la solution reprise par exemple par la base de données O2 [ODeux 91]

A la différence des bases de données relationnelles-objets, la plupart des bases orientées-objets tentent d'implémenter un nouveau langage de requête nommé OQL. Bien que très proche du langage SQL par sa syntaxe, le langage OQL peut s'intégrer dans un langage de programmation orienté objet. Par exemple l'OQL permet d'invoquer des méthodes au sein des requêtes.

Ainsi la requête SQL suivante :

```
SELECT nom FROM Enseignent WHERE age>20
```

est remplacé par la requête OQL suivante :

```
SELECT p.getNom() FROM Enseignant p WHERE p.getAge()>20
```

Bien que cet exemple puisse sembler anodin (puisque les méthodes vont certainement récupérer les valeurs d'un attribut) cela permet toutefois de masquer les attributs (ce qui est généralement le cas dans une bonne implémentation objet) car on ne sait pas à priori ou ont été définis ces attributs et donc ces méthodes. Par exemple la méthode *getAge()* a pu être définie dans une super-classe de la classe *Enseignant*, par exemple la super-classe *Personne*.

Remarque : Philosophiquement, l'implémentation de l'OQL pose le problème de la répartition des méthodes entre le client et le serveur.
En effet lorsqu'un SGBD possède une architecture client-serveur, c'est le serveur qui exécute les requêtes, et les méthodes d'objets (implémentées dans un langage particulier) sont exécutées chez le client. Par conséquent, comment fait le serveur pour exécuter les méthodes inclues aux requêtes ?
Nous n'avons pas étudié toutes les bases de données orientées-objets, mais parmi celles que nous avons étudiées, celles qui implémentaient l'OQL ne le faisaient pas d'une façon satisfaisante. Par exemple pour SGBD O2 les méthodes utilisées dans les requêtes OQL doivent être implémentées au niveau du serveur dans le langage O2C, alors que les méthodes s'exécutant sur le poste client peuvent être implémentées en C ou Java.

2.2. La base de données Matisse.

Notre choix s'est arrêté sur le SGBD orienté-objet client-serveur Matisse (Multilingual Advanced Technology for Information Systems Semantic Engineering) de la société Euriware. Vous pourrez trouver une description précise de la genèse de cette base de données à l'Université Technologique de Compiègne dans [Necaille 98].

---

[42] Application Program Interface

La puissance de Matisse réside principalement dans sa gestion optimisée de son architecture client-serveur et par la puissance de son méta-schéma.

2.2.1. Architecture client serveur.

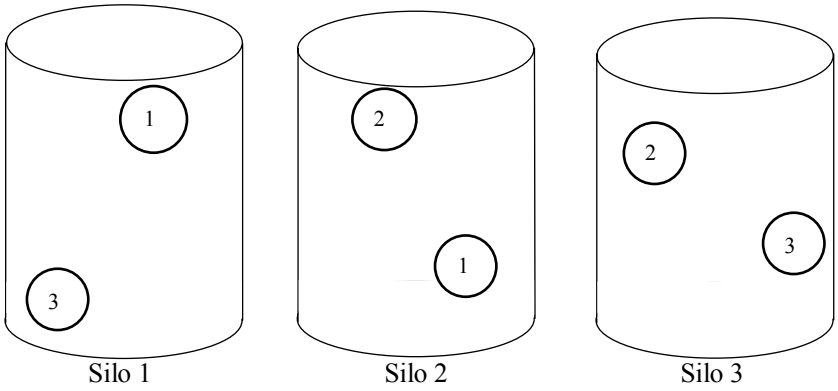
2.2.1.1. Le serveur.

Le serveur Matisse se distingue des autres serveurs de bases de données par les points suivants :

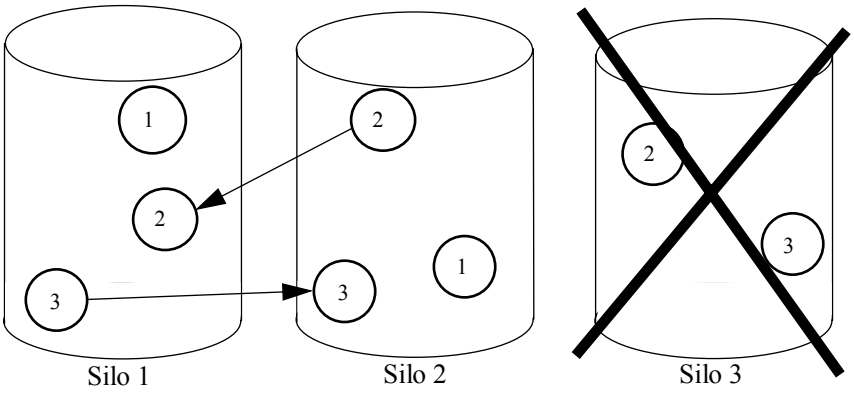
- Il intègre un mécanisme de versionisme qui permet d'éviter la création de fichiers *log* (fichiers qui sauvegardent l'ensemble des transactions) pour assurer l'intégrité de la base. Ceci permet d'améliorer la vitesse des transactions concurrentes. De plus, grâce à ce mécanisme, l'administrateur a le pouvoir de modifier le schéma ou d'effectuer une copie de la base sans pour autant déconnecter les utilisateurs présents.
- Il intègre un mécanisme de réplication d'objets sur plusieurs silos[43], ce qu'il lui procure une grande tolérance vis-à-vis des problèmes de sauvegarde.

Pour bien comprendre, analysons le cas d'école suivant :

Supposons qu'une base soit composée de trois silos et de trois objets, Matisse répartira les objets de la façon suivante :



Imaginons maintenant que le disque où se trouve le troisième silo tombe en panne. Matisse reconstruira alors une nouvelle version homogène de la base sur les deux premiers silos, comme le montre la figure suivante :



On peut remarquer que Matisse n'effectue pas un simple "mirroring" pour avoir une bonne tolérance aux pannes, la copie s'effectue en effet au niveau des objets et non au niveau de la base.

[43] Le silo est l'unité de base de stockage du SGBD Matisse. L'administrateur, pour chaque base Matisse, détermine le nombre de silos, leur taille (de quelques kilos à plusieurs gigas octets) et leur position respective sur les disques du système.

2.2.1.2. Les clients.

Il existe trois catégories de programme client qui peuvent interagir avec le serveur :

- L'application d'administration (*Mt_Dba* pour *Matisse Database Administrator*) qui permet de gérer les bases accessibles depuis un serveur, c'est-à-dire créer, détruire, lancer et stopper des bases. De plus, pour chaque base, *Mt_Dba* permet de gérer les silos, effectuer des réplications et gérer les comptes utilisateurs (spécifiques à chaque base de données). Il ne peut y avoir qu'une application d'administration par serveur, et cette dernière est obligatoirement exécutée sur la machine hôte du serveur Matisse.
- L'application d'édition (*Mt_Editor* pour *Matisse Editor*) qui permet de gérer les objets d'une base. Comme nous allons le voir, Matisse est une base totalement objet (avec un méta-schéma). De ce fait la gestion d'un modèle s'effectue par la création et la destruction d'objets. Les applications d'édition peuvent tourner sur une petite dizaine de plates-formes et comme pour tout client Matisse (à l'exception de l'outil d'administration) il peut y avoir plusieurs *Mt_Editor* qui attaquent la même base à un instant t.
- Les applications propriétaires, c'est-à-dire celles qui sont programmées pour un modèle déterminé de la base de données (on peut les nommer aussi applications terminales). Elles sont à la charge des développeurs, qui disposent d'API pour les programmer. Il existe des API pour les langages C, C++, Eiffel, Smalltalk et Java. De plus pour certaines plates-formes, Matisse est accompagné de *driver ODBC* [44] permettant d'étendre encore le nombre de langages utilisables.

Remarque : Quelque soit le SGBD, dont Matisse, les API sont dépendantes du langage et du système d'exploitation du programme client. De ce fait, si un éditeur de SGBD veut fournir des API pour n langages et m systèmes d'exploitation, il doit développer *n*m* API.

2.2.2. Le méta-schéma.

La figure 47, extraite de [Necaille 98], présente le méta-schéma de Matisse. Ce schéma, qui peut sembler à première vue un peu compliqué, reprend le principe du méta-protocole introduit par les langages ObjVLisp [Demphlous 98] et CLOS [Steel 90] (Vous pourrez trouver une analyse du méta-protocole de CLOS dans [Delestre 96]).

Pour bien comprendre, il faut savoir que dans Matisse tout est objet : les classes (instances de la classe *Mt Class*), les relations (instances de la classe *Mt Relationship*), les index (instances de la classe Mt Index), les attributs (intances de la classe *Mt Attribute*) etc.

La méta-classe (la classe de toutes les classes) de ce schéma est la classe *Mt Class*. Elle définit ce qu'est une classe Matisse. En effet, *Mt Class* possède :

- deux attributs *Mt Name* (qui va permettre de nommer les classes) et *Mt Instance Check* (qui va permettre d'associer une fonction constructeur aux classes).
- sept relations qui permettent d'associer à une classe une liste d'attributs (*Mt Attributes*), une liste de relations (*Mt Relationships*), une liste de super-classes (*Mt Superclasses*), une liste de sous-classes (*Mt Subclasses*), une liste de relations dont *Mt Class* est la destination (*Mt Successor Of*), une liste de methodes (*Mt Own Method*) et une liste d'index (*Mt Index*).

---

[44] Object Database Connectivity

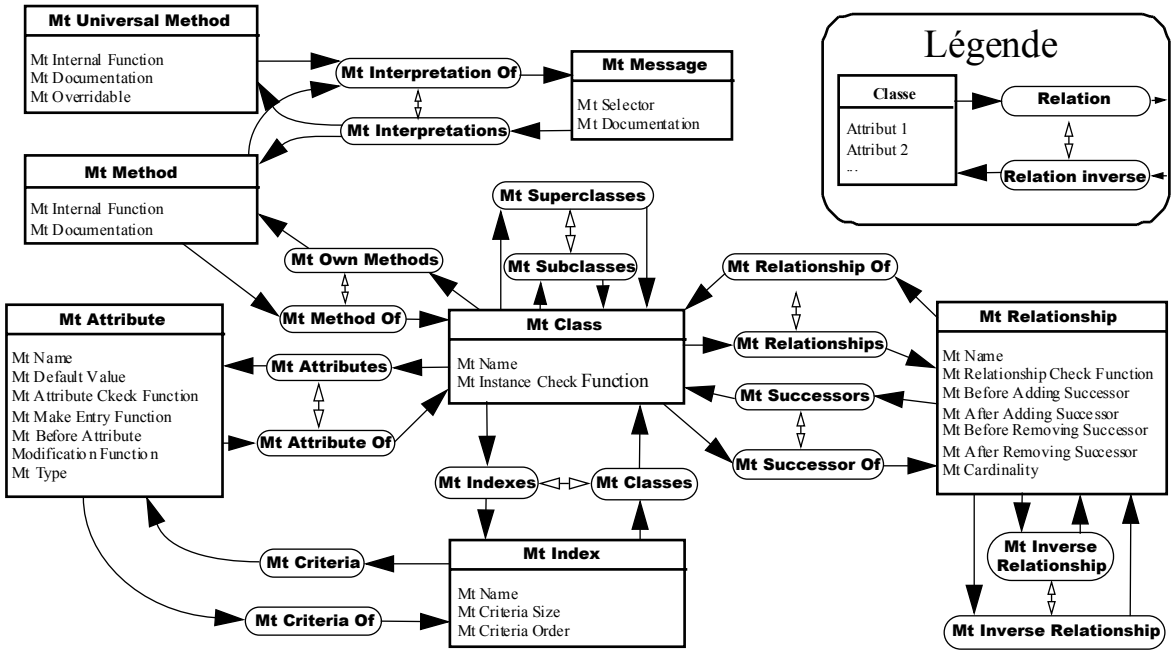


Figure 47 - Méta-schéma de Matisse (notation non standard)

La méta-classe *Mt Class* est aussi une classe (c'est sa propre instance), elle respecte donc cette description, comme l'indique le tableau suivant :

Attributs		
	Mt Name	Mt Class
	Mt Instance Check Function	nil
Relations		
	Mt Attributes	Mt Name, Mt Instance Check Function
	Mt Relationship	Mt Attributes, Mt Relationship, Mt Superclasses, Mt Subclasses, Mt Index, Mt Own Method, Mt Successor Of
	Mt Superclasses	nil
	Mt Subclasses	nil
	Mt Index	nil
	Mt Own Method	nil
	Mt Successor Of	Mt Successors, Mt Relationship Of, Mt Superclasses, Mt Subclasses, Mt Method Of, Mt Attribute Of, Mt Classes

Tableau 4 - Auto-définition de la méta-classe Mt Class

A la différence du méta-protocole de CLOS, il n'y a pas de super-classe par défaut (l'équivalent de la classe *standard-objet* de CLOS). Ainsi, si on désire créer une classe, il suffit de créer une nouvelle instance de la méta-classe *Mt Class* en ne spécifiant aucune super-classe, et si l'on veut

créer une nouvelle méta-classe il suffit de créer une classe, instance de *Mt Class* et sous-classe de *Mt Class*.

Les autres classes de ce méta-schéma sont des classes standards. Il y a entre autres :

- *Mt Relationship* qui permet de créer des relations entre les classes. Il y a par défaut plusieurs instances de cette classe (*Mt Subclasses*, *Mt Superclasses*, *Mt Own Method*, etc.). On notera que toutes les relations sous Matisse possèdent une relation inverse. Par exemple la relation *Mt Own Method* possède sa relation duale, qui est *Mt Method Of*.
- *Mt Attributes*, qui permet de définir des attributs. Il y a par défaut plusieurs instances de cette classe dans le méta-schéma (*Mt Name*, *Mt Selector*, etc.).
- *Mt Index*, qui permet de définir des index sur les attributs de certaines classes.
- *Mt Universal Method*, *Mt Method* et *Mt Message*, qui permet de simuler de la programmation objet (invocation de message) sur les objets persistants via des langages procéduraux tel que le C. Ces classes ne sont pas utilisées avec les langages objets.

2.3. Implémentation du modèle de Metadyne dans Matisse.

Grâce à l'homogénéité du méta-schéma de Matisse l'implémentation du modèle objet de Metadyne que nous avons vu dans le chapitre III ne pose aucun problème. Nous avons suivi la démarche suivante :

- Déterminer l'ensemble des attributs des différentes classes, en précisant leurs types (car sous Matisse on peut associer plusieurs types à un attribut : possibilité que nous n'avons pas utilisée dans notre implémentation) et leur valeur par défaut.
- Créer pour chaque attribut une instance de la classe *Mt Attribute*.
- Pour chaque classe du modèle du domaine créer une instance de la classe *Mt Class* en associant les attributs précédemment créés. A ce niveau, nous avons ajouté le préfixe *Cl* à chaque nom de classe afin de les distinguer aisément des classes Matisse.

Nous venons d'étudier succinctement la base de données Matisse, mais une base a beau être esthétiquement belle et transactionnellemment puissante, si il n'y a aucun outils pour l'utiliser, elle ne sert à rien. Or dans notre cas, les clients de Matisse vont être différents programmes écrits en Java. Nous allons donc maintenant nous intéresser à l'inter-connection entre Java et les bases de données.

3. La persistance des objets en JAVA via un SGBD.

Nous allons commencer cette étude par l'analyse de la persistance des objets Java à l'aide d'une base de données relationnelle. Puis nous effectuerons la même analyse avec les bases de données objets. Nous finirons par étudier l'interfaçage proposé par Matisse.

Avant toute chose, il faut bien comprendre que la persistance d'objet Java n'est possible que si le programmeur dispose d'un outil permettant d'interconnecter son programme avec la base de données cibles : on parle alors de passerelle. Ensuite, une fois que l'interconnection est établie, il existe différentes façons de rendre certains objets persistants. Ce sont ces deux facettes que nous allons étudier maintenant.

3.1. Interfaçage avec une base de données relationnelle.

3.1.1. JDBC.

Afin de fournir une passerelle entre les bases de données et Java, la société Sun à créé JDBC (pour Java DataBase Connectivity). Cette API, fournie par les éditeurs de bases de données, per-

met aux programmes Java de s'interfacer avec des bases de données relationelles via le langage SQL. Comme l'indique [Poulain 98], cet outil n'est adapté à la persistance d'objets Java que si le nombre de classes à sauvegarder est faible (deux à trois) et que les attributs de ces classes sont de type simple (entier, réel, chaîne de caractère, etc.). Dans le cas contraire, l'informaticien est obligé d'effectuer à la main le "mapping" du modèle objet en table, ce qui est très coûteux en temps et très contraignant : le moindre petit changement sur le modèle objet peut avoir des conséquences catastrophiques sur le modèle relationel. De plus, toute navigation entre objets se traduit par une requête nécessitant au moins une jointure qui s'effectue au détriment de la clarté du code et des performances.

Donc, l'utilisation du JDBC impose :

- au développeur de connaître le schéma de la base de données correspondant au schéma objet de son programme.
- l'utilisation du SQL au sein des programmes, les résultats obtenus doivent alors être retranscrits en objets Java (par exemple si on fait une requête sur une table personne, nous allons obtenir son nom, prénom,… c'est-à-dire les attributs de la personne, qu'on va utiliser pour construire un nouvel objet personne).

3.1.2. Java Blend.

Le produit Java Blend de la société Sun (qui a seulement quelques mois d'existance aujourd'hui) propose une solution beaucoup plus souple basée sur le *mapping* automatique Objet/Relationnel. Le programmeur commence par écrire entièrement son application en Java, et l'outil Java Blend élabore alors un modèle relationnel correspondant au modèle objet initial. Basé sur l'utilisation du JDBC, il est donc utilisable avec un grand nombre de bases de données. Parmi toutes ses caractéristiques, nous pouvons noter que :

- Le *mapping* peut être totalement automatique ou guidé par l'informaticien.
- Il est conforme au standard ODMG[45] pour le *mapping* Objet/Relationnel .
- Il adopte un mécanisme de concurrence très sophistiqué.

3.2. Interfaçage avec une base de données objet.

La persistance manuelle, qui est obtenue par l'exécution d'une méthode d'un objet "base de données" sur un objet transient (en mémoire temporaire), est la première technique qui est apparue. Cependant elle enlaidit fortement le code : il est préférable d'encapsuler plus fortement les propriétés de persistance. Les langages objets (et donc Java) proposent donc deux techniques de persistance : la persistance par héritage et la persistance par référence.

3.2.1. La persistance par héritage.

La persistance par héritage considère que la propriété de persistance d'un objet doit être définie par une classe-mère de persistance. Cette classe-mère définit alors au sein de ses constructeurs et destructeurs les opérations de connexion à la base de données, de création de l'objet correspondant, de modification d'attribut, etc. Ainsi le fait de créer un objet d'une classe héritant de cette classe, crée un objet identique dans la base de données.

L'inconvénient majeur de cette technique est qu'il n'est alors plus possible d'avoir une classe qui pourrait avoir des objets transients et des objets persistants : les instances sont, soit toutes transientes, soit toutes persistantes.

45. Object Database Management Group

3.2.2. La persistance par référence.

Pour palier le problème que nous venons de soulever, il faut déplacer la propriété de persistance de la classe vers les objets. Toutefois afin de ne pas tomber dans les travers de la persistance manuelle, la technique de persistance par référence consiste à :

- soit définir une racine de persistance, c'est-à-dire spécifier à l'objet sa qualité de persistance lors de sa création (en utilisant un constructeur particulier),
- soit associer un objet transient à une racine de persistance (l'objet transient deviendra alors persistant).

Par exemple (tiré de [Bouzeghoub 97]) :

```
paris : Site
paris = new persistent ("Paris")
martin : Employé
martin = new ("Martin")// martin est un objet temporaire
paris.affecter(martin)// martin devient persistant
```

3.3. Le cas de MATISSE.

La politique suivie par la société ADB pour la base Matisse est celle de la persistance par héritage. Ce choix qui peut sembler restrictif au premier abord, est la conséquence du processus d'obtention des classes Java. En effet, alors que certaines bases de données spécifient leur schéma en fonction des classes Java qu'on leur soumet [O2], Matisse fabrique les classes Java correspondant à son propre modèle. Comme le montre la figure suivante, le processus de construction des classes Java s'effectue en deux étapes :

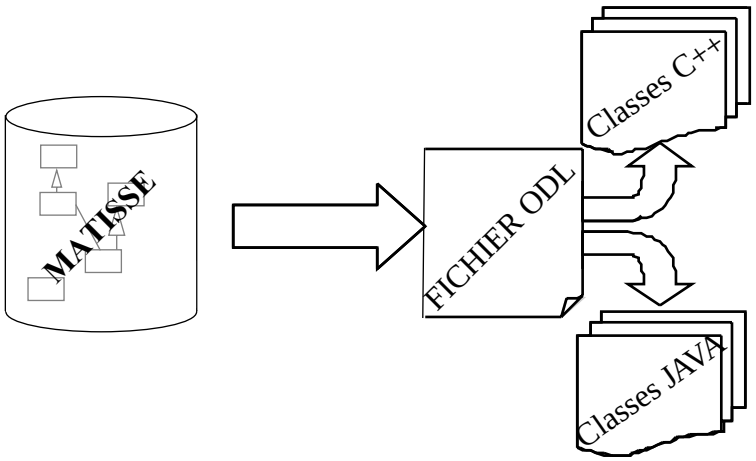


Figure 48 - Obtention des classes Java depuis Matisse

- La première étape consiste à produire un fichier de description du schéma instancié dans la base. Cette description est effectuée grâce à l'utilisation du langage ODL[46] de l'ODMG.
- A partir de ce fichier il est alors possible d'obtenir des classes,équivalentes aux classes de la base, soit en langage C++, soit en langage Java.

3.3.1. Le cas des classes.

Les classes Java obtenues respectent alors les règles suivantes :

46. Object Definition Language

- Si la classe Matisse correspondante n'admet aucune super-classe, la classe Java admet comme super-classe la classe *Mt Object* (qui est la classe de persistance).
- Si la classe Matisse correspondante possède des attributs, la classe Java possédera trois méthodes de gestion pour chaque attribut. Par exemple si une classe Matisse possède un attribut *name*, la classe Java correspondante possédera les méthodes *getName* (pour obtenir la valeur de l'attribut), la méthode *setName* (pour fixer la valeur de l'attribut) et *clearName* (pour initialiser la valeur de l'attribut, c'est-à-dire mettre la valeur par défaut définie dans Matisse ou null sinon).
- Si la classe Matisse correspondante possède des relations :
  - Si la relation est une relation unaire, alors on gère cette relation comme un attribut (on a donc les méthodes *getXX*, *setXX*, et *clearXX*, où XX est le nom de la relation)
  - Si la relation est une relation naire, alors on gère cette relation via les méthodes *getXX* (pour obtenir toutes les destinations), *pushXX* (pour ajouter une destination en début de liste), *appendXX* (pour ajouter une destination en fin de liste), *insertXX* (pour insérer une destination), *removeXX* (pour détruire une relation) et *clearXX* (pour détruire toutes les relations).

Remarque : L'obtention du fichier ODL et des classes (en C++ ou en Java) s'effectue à l'aide d'un seul et unique utilitaire *mt_odl*. En fait cet utilitaire permet d'interagir totalement avec Matisse via le langage ODL. Il peut, comme nous l'avons déjà vu, obtenir le schéma de la base, construire les classes correspondantes, mais aussi implémenter un schéma dans une base vierge. Dès lors un chaînage avec un atelier de génie logiciel qui produit des fichiers ODL est alors possible (c'est ce que nous avons étudié dans [Delestre & al 97a] avec l'atelier de génie logiciel DOM, pour Delphia Object Modeler, de la société ATOS). On peut donc obtenir le chaînage présenté par la figure 49.

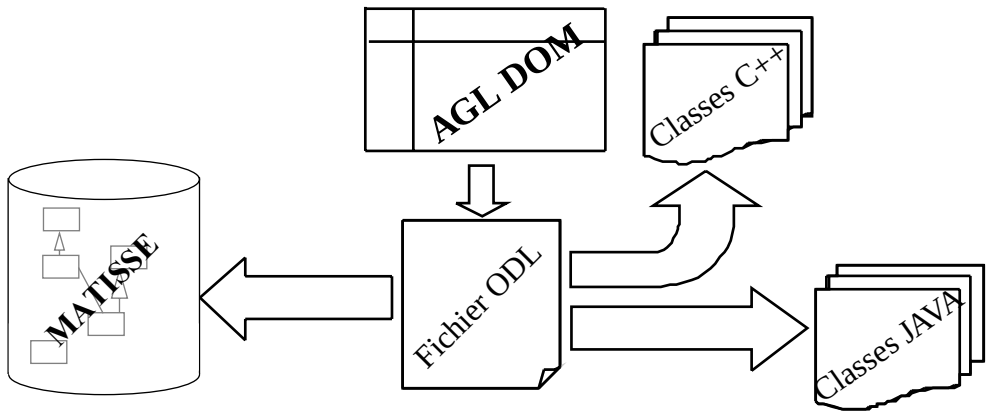


Figure 49 - Chaîne AGL - Matisse - Code

3.3.2. Le cas des méta-classes.

Ce que nous venons de voir limite le champ d'action du programmeur, car les programmes clients n'ont accès qu'aux objets et classes (non méta-classes) déjà créés. Avec ce mécanisme de persistance par héritage il est impossible :

- de parcourir le méta-schéma d'un modèle d'une base,
- de modifier le schéma du modèle d'une base, c'est-à-dire de créer des nouvelles classes ou méta-classes.

Les API Java et C++ grâce à la classe *Dynamic Object Interface* (DOI) permettent d'éliminer ce problème. En effet, la DOI permet de manipuler des objets Matisse sans connaître aupréalable leur structure. De plus la DOI permet de créer de nouveaux objets (et donc de nouvelles classes et méta-classes) en spécifiant grâce à des méthodes les super-classes, les attributs, les relations, etc.

Remarque : En Java et C++, il est possible d'ajouter aux classes des attributs de classe et des méthodes de classe (nommés attributs et méthodes statiques). Dans un schéma objet admettant des méta-classes, ces attributs et méthodes, sont des attributs et méthodes de méta-classe de niveau 1 (noté méta^1-classe), c'est à dire que leurs instances sont des classes (et non pas des méta-classes). Or justement, le schéma objet de Matisse n'admet que des méta-classes de niveau 1, ce qui signifie que l'on peut en Java ou C++ "simuler" l'existence de telles méta-classes grâce à l'adjonction des cas attributs et méthodes statiques.

Nous savons maintenant comment sauvegarder dans Matisse des objets à partir de programmes Java. Nous allons donc maintenant nous intéresser au deuxième maillon de notre chaîne, c'est-à-dire le serveur d'objets distribués.

4. Les objets distribués.

Le concept "objet distribué" est une des conséquences logiques de l'évolution du génie logiciel. Il y a une vingtaine d'années furent créés les RPC[47], car on avait besoin d'exécuter des procédures ou fonctions sur des machines distantes (par exemple exécuter des calculs matriciels sur des machines parallèles). Aujourd'hui, les besoins sont toujours les mêmes, certaines machines ont des capacités que d'autres n'ont pas (capacité de persistance, capacité de calcul, etc.). Or la programmation procédurale, et donc l'utilisation des RPC, n'est plus à l'ordre du jour : il a alors fallu mettre en œuvre un mécanisme qui permette aux programmes dits clients de faire référence à des objets distants, permettant ainsi l'invocation de méthodes distantes.

Ainsi, un système d'objets distribués doit permettre :

- de retrouver un objet distant,
- d'invoquer des méthodes sur cet objet distant,
- de passer en paramètre à des méthodes locales ou distantes, des objets locaux ou distants,
- de créer de nouveaux objets distants.

Dans notre cas, le serveur d'objets distribués doit permettre à des clients, quelque soit le type de machine et le système d'exploitation utilisés, de sauvegarder des objets dans notre base de données.

Avant d'analyser les particularités du serveur d'objets distribués de Metadyne, nous allons faire un tour d'horizon des principes et outils disponibles sur le marché, en commençant par les plus connus CORBA et en finissant par le moins connu HORB.

---

[47] Remote Procedure Call

4.1. CORBA.

4.1.1. Généralités.

Avant de décrire les fonctionnalités et les particularités de CORBA[48], il est bon de rappeler que CORBA n'est pas un outil mais une spécification de l'OMG[49]. Charge alors aux différents éditeurs de développer des logiciels qui adoptent cette norme : par exemple Sun a développé JOE (Java Object Environement) qui est compatible CORBA.

Les principes de CORBA reposent principalement sur une séparation stricte entre l'interface et l'implémentation d'une classe, ainsi que sur la transparence d'accès aux objets (lorsqu'un client utilise un objet distribué, il n'a pas à savoir où ce dernier se trouve). Dès lors CORBA peut être en grande partie identifié par :

- l'utilisation de l'IDL (Interface Definition Language) qui comme son nom l'indique, est un langage universel de définition d'interface,
- l'ORB[50] qui est l'élément clé de la communication entre les clients et les serveurs. En fait comme l'indique la figure suivante (extrait de [Geib & al 98]), on peut diviser ORB en deux parties : l'interface et le noyau (repère n°1).

Il existe alors deux manières d'invoquer une méthode d'un objet distribué :

- L'appel statique, qui est utilisé dans la plupart des cas, n'est possible que lorsque le développeur du client connaît déjà l'ensemble des objets proposés par le serveur. Il utilise alors les interfaces d'invocation statique (repère n°2).
- L'appel dynamique permet de récupérer un objet et de pouvoir y invoquer des méthodes sans connaître au préalable son interface. Ce type d'appel est très rarement utilisé, on peut toutefois le retrouver dans des applications de développement ou des navigateurs d'objets (repère n°3).

Lorsqu'un client veut utiliser un objet distribué, il ne reçoit pas l'objet en tant que tel, mais une référence sur cet objet. Cette référence est créée au niveau de l'adaptateur d'objet (OA, repère n°4). Cet OA a aussi en charge l'enregistrement des classes serveurs dans le référentiel d'interface (annuaire des objets distribués, repère n°5), l'instanciation de nouveaux objets à la demande des clients, la gestion des appels des clients (récupération des paramètres d'une méthode, exécution de la méthode, retour du résultat au niveau du client) et l'authentification éventuelle des clients en cas de protection.

Enfin, pour finir cette présentation générale de CORBA, l'OMG a défini différents services au dessus du bus. Il y a entre autres les services :

- de nommage, qui permet de retrouver un objet en fonction d'un nom symbolique qu'on lui a attribué,
- de cycle de vie, qui permet de faire une copie d'un objet, de créer un nouvel objet ou de le détruire,
- de sécurité, qui permet d'identifier les clients et de crypter l'ORB,
- de persistance, qui permet de stocker des objets dans de simples fichiers ou dans des bases de données (relationelles ou objets),
- etc.

En fait, l'OMG a défini en tout 12 services [Geib 97]. On peut toutefois remarquer que les différentes distributions CORBA disponibles sur le marché n'intègrent jamais (en tout cas celles

48. Common Object Request Broker Architecture.
49. Object Management Group
50. Object Request Broker

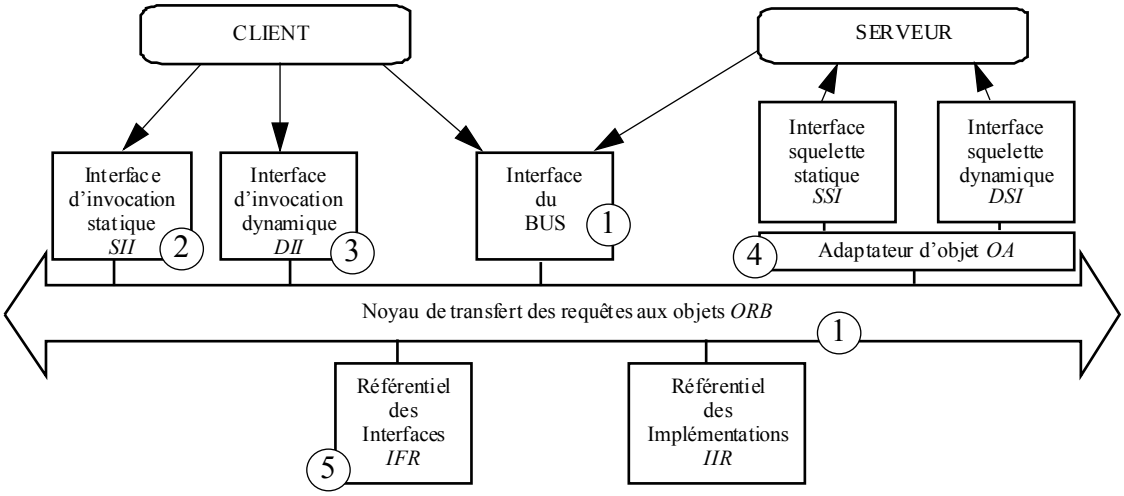


Figure 50 - CORBA

que l'on a pu étudier) tous ces services.

4.1.2. Les différentes étapes pour l'obtention d'un serveur CORBA en Java.

Dans cette partie, nous allons prendre Java IDL de la société Sun comme outil de développement de serveur d'objets distribués. Il est composé de trois éléments :

- un ORB conforme à CORBA,
- un compilateur IDL, nommé *idltojava*,
- deux paquetages Java (nommés *org.omg.Corba* et *org.omg.CosNaming*) qui contiennent des classes utilisées par le compilateur et l'ORB.

Tout commence par la création d'un fichier IDL[51] définissant la ou les classes des objets que l'on veut distribuer. Après la compilation de ce fichier à l'aide de idltojava, on obtient cinq fichiers, deux classes pour la partie cliente, une classe pour la partie serveur et une interface et une classe commune aux deux parties.

Imaginons que nous voulions distribuer un objet de classe *XX*. Nous définissons l'interface de cette classe à l'aide du langage IDL. En compilant ce fichier, on obtient l'interface *XX* et la classe *XXHolder*, qui est commune au client et au serveur. Cette classe permet de simuler le passage par adresse. En effet, le langage IDL permet de définir le type de passage de paramètre (in, out ou inout) alors qu'en Java, les types simples sont passés par valeur et les objets sont passés par adresse. Dès lors si une méthode de l'objet distribué a un de ses paramètres de type simple qui est passé par adresse, dans le code Java correspondant ce sera son équivalent encapsulé qui sera utilisé. Par exemple, comme l'indique [Vogel & al 98], pour un paramètre de type *short*, ce sera la classe de type *ShortHolder* qui sera utilisée.

De plus, côté client nous obtenons :

- La classe *_XXStub* qui est la partie cliente de l'objet distribué. Elle implémente l'interface *XX* et est une sous-classe de la classe *org.omg.CORBA.portable.ObjectImpl*.
- La classe *XXHelper* qui permet de créer un référent sur l'objet distribué (grâce entre autre à la méthode statique *narrow()*)

et côté serveur nous obtenons :

- La classe abstraite *_XXImplBase* qui est l'équivalent de XXStub pour le serveur.

[51] Ne pas confondre l'ODL qui est un langage de description de classe et l'IDL qui est un langage de description d'interface.

Elle implémente donc aussi l'interface *XX* et est une sous-classe de la classe *org.omg.CORBA.DynamicImplementation.*

À l'issue de cette génération de fichiers, il faut encore développer à la main deux classes :

- La classe de notre objet distribué, qui va se nommer *XXServant* et qui hérite de la classe *_XXImplBase*. C'est dans cette classe qu'est décrit le corps de l'ensemble des méthodes de l'objet que l'on veut distribuer.
- La classe *XXServer* qui est l'adaptateur d'objet qui permet d'instancier l'objet qui va être distribué.

On le voit bien la mise en oeuvre d'un serveur d'objets distribués compatible CORBA n'est pas une mince affaire : les manipulations sont lourdes et coûteuses en temps de développement. On peut même penser que pour un objet simpliste, il est plus coûteux de le distribuer que de le développer.

4.2. RMI.

4.2.1. Généralités.

Pour les personnes qui programment exclusivement en Java, et qui ne veulent pas entrer dans les méandres de CORBA, la société Sun a créé un produit gratuit permettant de créer des serveurs d'objets distribués : RMI[52] (Cf. [Nicolas & al 97]). Ce produit est composé de deux parties :

- Un ensemble d'interfaces et de classes que l'on va pouvoir utiliser pour créer les composantes client et serveur du système.
- Un programme d'enregistrement des objets que l'on veut distribuer

Les objectifs de RMI sont les suivants :

- Il est possible d'invoquer une méthode d'un objet distribué, où que soit l'objet (sur une machine virtuelle différente, sur une machine physique différente, sur un autre système d'exploitation).
- La syntaxe d'invocation d'une méthode d'un objet distant est strictement identique à l'invocation d'une méthode d'un objet local.
- Un objet distribué peut être déclaré par plusieurs interfaces distantes qui définissent les méthodes de l'objet.

4.2.2. Mise en œuvre.

La mise en œuvre est beaucoup plus simple que CORBA. Tout d'abord, il faut définir l'interface de l'objet que l'on va distribuer. Ensuite on doit implémenter cette interface côté serveur et la rattacher à une URL. Puis il faut générer les stubs (partie visible par le client) et les skeletons (partie côté serveur) grâce l'utilitaire *rmic*. Enfin les clients n'ont plus qu'à récupérer l'objet distribué pour pouvoir invoquer les méthodes désirées.

On le voit bien, avec RMI, la création d'objets distribués est beaucoup plus simple. Le temps nécessaire à la création d'objets distribués reste raisonnable. Toutefois, dans le monde Java, RMI a un concurrent des plus sérieux : HORB.

[52] Remote Method Invocation

4.3. HORB.

4.3.1. Généralités.

HORB est un ORB Java développé par M. Satoshi HIRANO dont la première version date de Novembre 1995, soit quatre mois avant la première annonce de la société Sun au sujet de RMI (première version qio n'est apparue que début 1997). A la différence de RMI, qui utilise quelques méthodes natives (entre autres lors de la sérialisation des objets), HORB est entièrement écrit en Java (les sources sont publiques). Ceci a permis à HORB :

- d'être dès le début compatible avec tous les navigateurs internet (il fonctionne parfaitement avec le JDK 1.02, alors qu'il faut au minimum le JDK 1.1 pour RMI)
- d'évoluer constamment, grâce à une petite communauté très active (il existe entre autres un serveur de newsgroup et une liste de diffusion). Par exemple, il n'a fallu environ que deux mois pour que la version 1.3 de HORB devienne compatible avec la dernière version du JDK (la version 1.2). De plus, il est important de noter que la future version de HORB (Version 2.0) sera inter-opérable avec des bus CORBA.

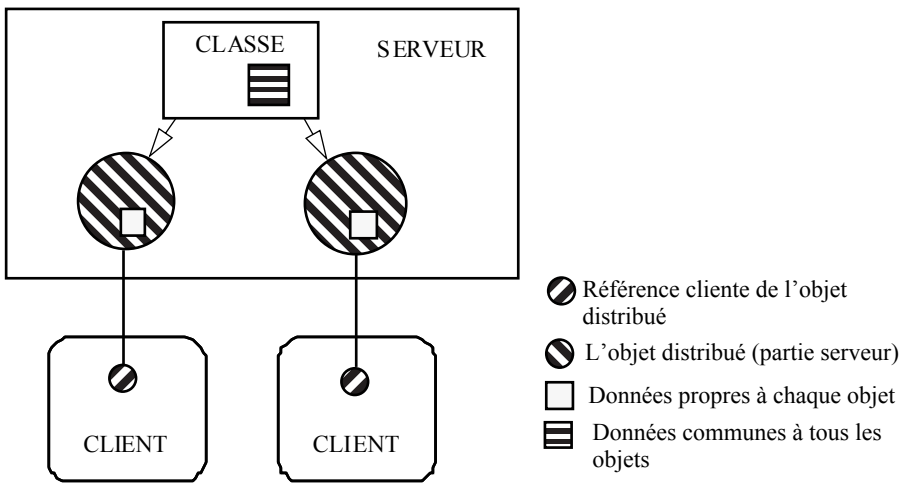


Figure 51 - Création dynamique d'objets distribués

Les caractéristiques principales d'HORB sont (Cf. [Hirano 97]) :

- La création dynamique d'objets distribués :
  un client peut demander à un serveur de lui créer un objet distribué propre (cette propriété peut être vue comme un partage de classe, ou une réelle distribution d'objets, cf. figure 51). HORB est le seul serveur d'objet distribué à permettre cette manoeuvre (si l'on exclut les distributions CORBA qui intègrent le service de cycle de vie). Ainsi deux clients peuvent avoir chacun un objet distribué qui leur est propre, avec par conséquent des données propres et pourquoi pas aussi des données partagées au sein de la classe.
- La connexion à un objet distribué :
  un client peut faire référence à un objet préalablement créé par le serveur. C'est le principe qu'applique l'ensemble des serveurs d'objets distribués (HORB, CORBA, DCOM, RMI, etc.). Nous pensons qu'il est alors plus honnête de parler d'objet partagé que d'objet distribué, car comme le montre la figure 52, les modifications sur des données de l'objet faites par un client ont une répercussion sur les autres clients.
- La possibilité d'avoir des méthodes asynchrones : le client peut invoquer une méthode, puis faire autre chose, et enfin récupèrer le résultat.

- La possibilité de positionner des droits d'accès.

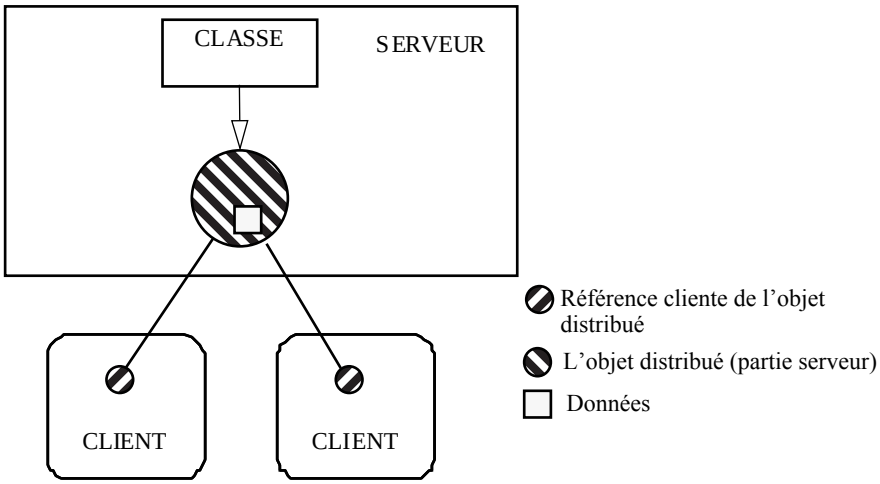


Figure 52 - Connection à un objet distribué

4.3.2. Mise en œuvre.

La mise en œuvre d'un serveur d'objet avec HORB est d'une grande simplicité. Oubliez tout ce que l'on vient de voir, plus de fichiers IDL, seulement deux exécutables à utiliser.

- Le compilateur nommé *horbc*, qui à partir d'une classe, crée automatiquement la classe qui sera utilisée par le client (le *proxy*) et la classe qui se trouvera sur le serveur (le *skeleton*).

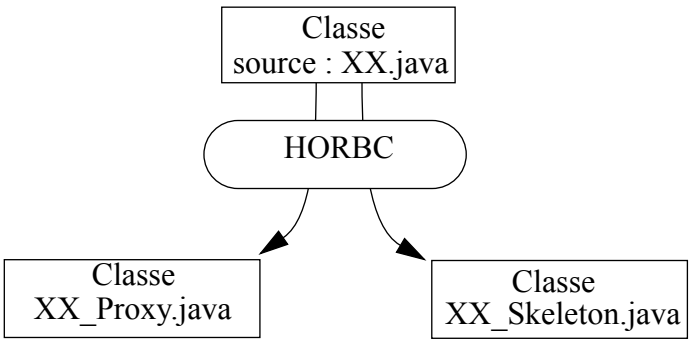


Figure 53 - Utilisation de horbc

  Bien entendu, ces deux classes générées sont écrites en java, et les sources sont aussi consultables. Ainsi il est possible de créer instantanément une classe distribuée.

- Le serveur à proprement parlé qui s'exécute dès le lancement du programme *horb*.

Ainsi, lorsque l'on veut distribuer une classe :

- On implémente la classe *XX*.
- On infère le compilateur HORB (*horbc*) sur cette classe : on obtient donc deux nouvelles classes, la classe *XX_Proxy* et la classe *XX_Skeleton*.
    - La classe *XX_Proxy* est la classe qui peut être instanciée au niveau du client. Le constructeur de cette classe admet au moins un paramètre, qui est l'URL du serveur d'objet. On retrouve, de plus, l'ensemble des méthodes **publi-**

ques définies dans la classe originale.

- La classe *XX_skeleton* est la classe qui se trouve sur le serveur d'objet. C'est au sein de cette classe que l'on retrouve notre classe de départ (relation d'agrégation).

- L'instanciation d'un objet de la classe *XX_Proxy* entraîne l'instanciation d'un objet de la classe *XX_Skeleton*, et donc un objet de la classe originale.
- L'invocation d'une méthode de *XX_Proxy*, envoie un message à l'objet de la classe *XX_Skeleton* correspondant, qui exécute la bonne méthode de l'objet de la classe originale. Si cette dernière retourne un résultat, ce dernier est transféré à l'objet de la classe *XX_Proxy*.

Enfin, des tests de performance ont été effectués, entre autres dans [Hirano & al 98], montrant que HORB est toujours plus rapide que les outils compatibles CORBA et que RMI. Seul DCOM est plus rapide dans certains cas (par exemple lorsque le serveur crée un objet à partager). De plus, de notre côté, nous avons effectué quelques tests afin de mesurer différents temps de réaction (pour la création dynamique d'objets distribués, la création par le serveur d'objets distribués, et l'invocation de méthodes) ainsi que le comportement du serveur lorsque le nombre d'objets distribués grandit. Les résultats de ces tests ont été concluants, puisque nous avons pu créer plus de trois milles objets distribués sur le serveur et les temps d'accès observés sont les suivant :

	Client et serveur	
	sur la même machine	sur des machines différentes
Création dynamique de 1000 objets	< 21 s	< 28 s
Création de 1000 objets	< 16 s	< 19 s
Invocation d'une méthode (1000 fois)	< 2 s	< 2 s

Remarque : Les machines que nous avons utilisées sont tout ce qu'il y a de plus banal puisqu'aussi bien pour le serveur que pour le client, nous avons utilisé des compatibles PC 200 MMX avec 64 Mo de RAM sous Windows NT 4.0. Ces deux machines étaient reliées par un réseau à 100 Mbits.

4.4. Les autres.

Outre CORBA, RMI et HORB, il existe d'autres outils pour invoquer des méthodes sur des objets distants.

4.4.1. DCOM.

Comme à son habitude, la société *Microsoft* a décidé de construire un système d'objets distribués propriétaire. Ce système est propriétaire car il est ancré au coeur même de Windows NT 4.0, et par conséquent ne fonctionne que sous cet environnement et de ce fait que sur les plates-formes à base de processeur Intel ou Alpha.

DCOM, pour Distributed COM est, comme son nom l'indique, une extension de COM (Component Object Model) car il permet aux composants (ou objets) COM d'inter-opérer sur des ma-

chines différentes. Lorsque le client et le composant serveur sont situés sur des machines distantes, DCOM remplace tout simplement le processus de communication inter-processus, par un système de communication réseau.

4.4.2. E.

E[53] est un outil de distribution sécurisé d'objets produit par *Electric Communities* sous licence *Mozilla* (c'est-à-dire que les sources de cet outil sont libres). E définit un modèle objet spécifique permettant de distribuer des objets avec des transmissions d'information sécurisées et avec des capacités de persistance. Cet outil est composé de deux éléments distincts, tout d'abord la librairie *Elib* et la langage de script *E language*.

- La librairie *ELib*, écrite en Java, permet d'implémenter le modèle Objet E, ce qui permet aux applications finales d'activer des méthodes sur des objets distants.
- Le langage de script E est un shell (un interpréteur de commandes) qui permet de programmer des fonctions et d'accéder à des objets Java.

4.5. Notre serveur d'objets distribués.

Reprenons l'architecture que nous avons choisie précédemment our Metadyne (figure 45). Le serveur d'objets dans notre cas permet aux *servlets* et *applets* de pouvoir stocker de l'information dans la base de données (car il ne faut pas oublier que pour nous le serveur d'objets distribués est un client du serveur MATISSE, et que les *applets* et *servlets* sont des clients pour le serveur d'objets distribués).

4.5.1. Objectifs et choix de l'outil.

Pour concevoir notre serveur et choisir l'outil que nous allions utiliser, nous sommes partis de l'objectif suivant :

> *Il faut que la programmation des clients soit la plus naturelle possible, la création et l'utilisation d'un objet distribué persistant doivent être équivalentes à la création et l'utilisation d'un objet local.*

Ce qui peut être résumé par la figure suivante :

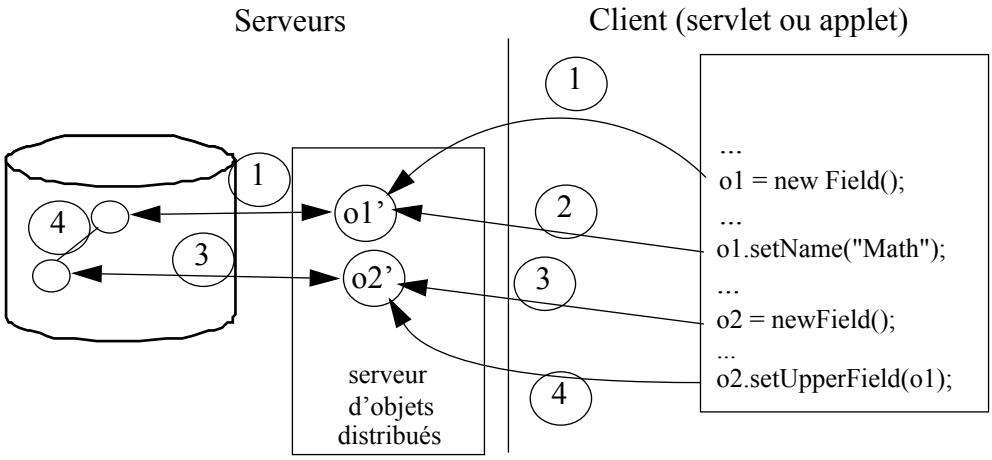


Figure 54 - Fonctionnement idéal de notre chaîne de persistance

[53] Vous pouvez trouver plus amples informations sur le serveur officiel : http://www.erights.org.

Etudions les quatre étapes types :

1. Le client (une *servlet* ou une *applet*) veut créer un nouveau champ d'enseignement, il crée donc une nouvelle instance de la classe *Field* (l'objet o1). Cette instanciation crée automatiquement l'objet distribué correspondant au niveau du serveur (l'objet o1'). Cette dernière instanciation crée dans la base de données l'objet correspondant.
2. L'invocation d'une méthode sur l'objet o1, entraîne l'exécution de la méthode correspondante sur l'objet distribué correspondant (l'objet o1'), qui modifie (ou récupère) de l'information au sein de la base de données.
3. Idem 1).
4. L'invocation d'une méthode sur l'objet o2, avec comme paramètre l'objet o1, entraîne l'exécution de la méthode équivalente au niveau du serveur sur l'objet o2' avec comme paramètre l'objet o1'. Ceci finalement modifie l'état de la base de données (par exemple dans notre cas, ceci crée une relation hiérarchique entre les deux champs).

Cet objectif de fonctionnement peut être en partie atteint en utilisant le mécanisme de distribution d'objets offert par HORB. En effet, HORB, outre sa simplicité d'utilisation, est le seul serveur permettant la création dynamique d'objets : dans notre exemple, la création de l'objet o1 entraîne automatiquement la création du véritable objet o1'.

4.5.2. La structure générale du serveur.

Afin de bien comprendre la structure générale de notre serveur, prenons l'exemple de distribution d'une classe, la classe *Field* (champ d'enseignement).

Nous avons tout d'abord la classe Java *ClField* produite par la base de données MATISSE. L'intanciation de cette classe provoque la création d'un objet correspondant dans la base. Cependant les classes Java de Matisse ne sont pas sérialisables. Cela signifie entre autres qu'elles ne peuvent pas être transférées via le réseau, et par conséquent ce ne sont pas ces classes que l'on peut distribuer. Il faut donc les encapsuler dans d'autres classes qui vont pouvoir être distribuées. Ainsi la classe *ClField* peut être encapsulée de la manière suivante :

```
public class Field {
   private ClField thePersistentObject;
   ...
   public void setName(String name) {
   ...
   }
   ...
   public Field getUpperField(){
   ...
   }
   public void setUpperField(Field theUpperField){
   ...

}
```

Toutefois nous ne pouvons toujours pas distribuer cette classe, c'est à dire lui appliquer le compilateur *horbc* afin d'obtenir les classes *Field_Proxy* et *Field_Skeleton*. En effet, au niveau du client, lorsque l'on va créer une instance de *Field_Proxy* et que l'on va vouloir invoquer la méthode *setUpperField*, on va passer en paramètre un objet de type *Field_Proxy* et non pas un objet de type *Field*. Pour résoudre ce problème, il faut créer une interface *Field* qui sera implémentée aussi bien par la classe source que l'on va nommer maintenant *Field_Impl* que par son équivalent distribué, c'est-à-dire la classe *Field_Impl_Proxy*. Soit :

pour l'interface :

```
public interface Field {
```

```
   ...
   public void setName(String name);
   public Field getUpperField();
   public void setUpperField(Field theUpperField){
   ...
}
```

et pour la classe *Field_Impl* :

```
public class Field_Impl implement Field {
   private ClField persistentObject;
   ...
   public void setName(String name) {
   ...
   }
   ...
   public Field getUpperField(){
   ...
   }
   public void setUpperField(Field theUpperField){
   ...

}
```

Malheureusement un nouveau problème apparaît. En effet lorsqu'un client crée une instance de la classe *Field_Impl_Proxy* et invoque la méthode *setUpperField*, au niveau du serveur c'est la méthode *setUpperField* de l'objet de classe *Field_Impl* correspondant qui est exécutée, avec comme paramêtre un objet de type *Field_Impl_Proxy*. Or cet objet ne connaît pas son équivalent serveur (l'objet de classe *Field_Impl*), et par conséquent ne peut donner accès à l'objet persistent correspondant (l'objet de classe *ClField*).

Pour être plus clair, revenons sur le fonctionnement de HORB grâce au schéma suivant :

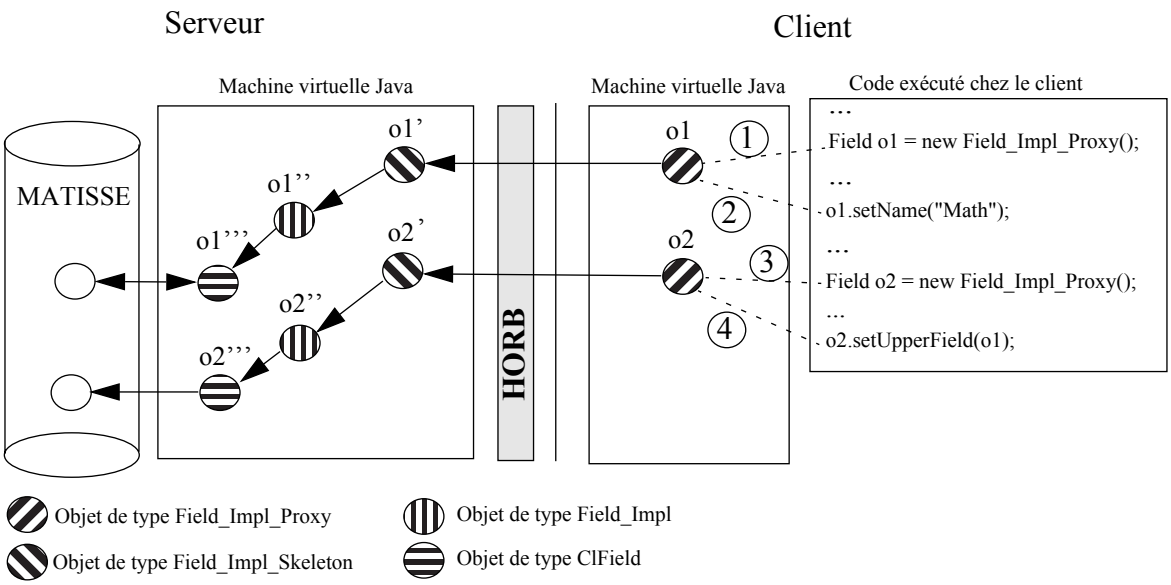


Figure 55 - Fonctionnement réel du serveur d'objets distribués

1. La création d'un objet de type *Field_Impl_Proxy* (o1) au niveau du client, provoque l'instanciation d'un objet de classe *Field_Impl_Skeleton* (o1'), qui provoque l'instanciation d'un objet de classe *Field_Impl* (o1'') et qui provoque enfin la création d'un objet persistant de classe *ClField* (o1'''). A ce niveau o1 "connaît" de façon transparente o1' (grâce à HORB) mais "ne connaît pas" o1'' et encore moins o1'''.
2. L'invocation de la méthode *setName* sur l'objet o1 entraîne l'invocation de la mé-

thode *setName* de l'objet o1'' (toujours grâce à HORB). Ici il n'y a pas de problème, puisque le paramètre de la méthode est une chaîne de caractères (type simple connu aussi bien du côté du client que du serveur).

3. Idem que le 1) mais pour l'objet o2.
4. C'est à ce niveau que le problème intervient. En effet, l'invocation de la méthode *setUpperField* sur l'objet o2 avec comme paramètre l'objet o1, provoque l'invocation de la méthode *setUpperField* de l'objet o2'' avec comme paramètre l'objet o1. Cette méthode va donc invoquer la méthode *setUpperField* de l'objet o2''' (ce qui ne pose pas de problème, puisque l'objet o2'' "connaît" parfaitement l'objet o2'''), mais il faudrait qu'en paramètre de cette méthode nous passions l'objet o1''', ce qui n'est pas possible, puisque l'objet o1, "ne connaît pas" l'objet o1'' et encore moins l'objet o1'''. Il faudrait donc au niveau de la classe *Field_Impl* et *Field_Impl_Proxy*, avoir une méthode qui retourne l'objet persistant associé (de la classe *ClField*), ce qui n'est pas possible puisque les classes Matisse ne sont pas sérialisables.

Nous avons résolu ce problème par l'utilisation d'une mémoire partagée au niveau des classes *XX_Impl*. Ainsi on ne fait plus référence aux objets persistants via un attribut privé local (comme dans notre exemple l'attribut *persistentObjet* de type *ClField*), mais on utilise un tableau dynamique, commun à toutes les classes *XX_Impl*. Ainsi l'instanciation d'un objet sur cette dernière alloue un objet persistant, que l'on positionne dans ce tableau et qui est référencé par son rang.

Dès lors, les classes *XX_Impl* possèdent entre autres deux nouvelles méthodes. L'une publique (que l'on retrouve donc aussi au sein des objets *XX_Impl_Proxy*) nommée *getPersistentObject-Position* qui retourne le rang de l'objet persistant dans la mémoire partagée. L'autre privée (qui n'est donc pas une méthode de *XX_Impl_Proxy*) nommée *getPersistentObjectAt*, qui retourne l'objet persistant correspondant à un rang passé en paramètre.

Ainsi la méthode *setUpperField* de la classe *Field_Impl* se résume aux lignes suivantes :

```
public void setUpperField(Field theUpperField) {
      try{
         openReadWriteAccess();
         ClField thisPersistentObject = (ClField)getPersistentObject();
         ClField upperPersistentObject = (ClField)getPersistentObjec-
tAt(theUperField.getPersistentObjectPosition());
         thisPersistentObject.setUpperField(upperPersistentObject);
         commitReadWriteAccess();
      }
      catch (Exception e){
         System.out.println("Field_Impl.setUpperField() : "+e);
      }
   }
```

Analysons rapidement ce code en reprenant les objets de l'exemple ci-dessus:

1. Le paramètre est de type *Field*, on peut donc appeler cette méthode aussi bien avec un objet de classe *Field_Impl* qu'avec une objet de type *Field_Impl_Proxy*, dans notre cas, c'est un objet de type *Field_Impl_Proxy*, c'est à dire l'objet o1.
2. Après l'ouverture d'un accès en lecture-écriture sur la base (méthode *openReadWriteAccess* que nous allons voir plus loin), nous récupérons l'objet persistant lié à l'objet de la méthode que nous exécutons, c'est à dire que *thisPersistentObject* pointe sur o2'''.
3. On fait de même pour le paramètre, c'est-à-dire que *upperPersistentField* pointe sur o1'''. Remarquez toutefois que la méthode *getPersistentObjectPosition* est la méthode de l'objet o1, alors que *getPersistentObjectAt* est la méthode de o2''.
4. Il ne reste plus alors qu'à invoquer la méthode *setUpperField* au niveau des objets

persistants, c'est-à-dire invoquer cette méthode au niveau de l'objet o2''' avec comme paramètre l'objet o1'''.

Afin d'intégrer ces différentes actions, nous avons opté pour le modèle objet suivant :

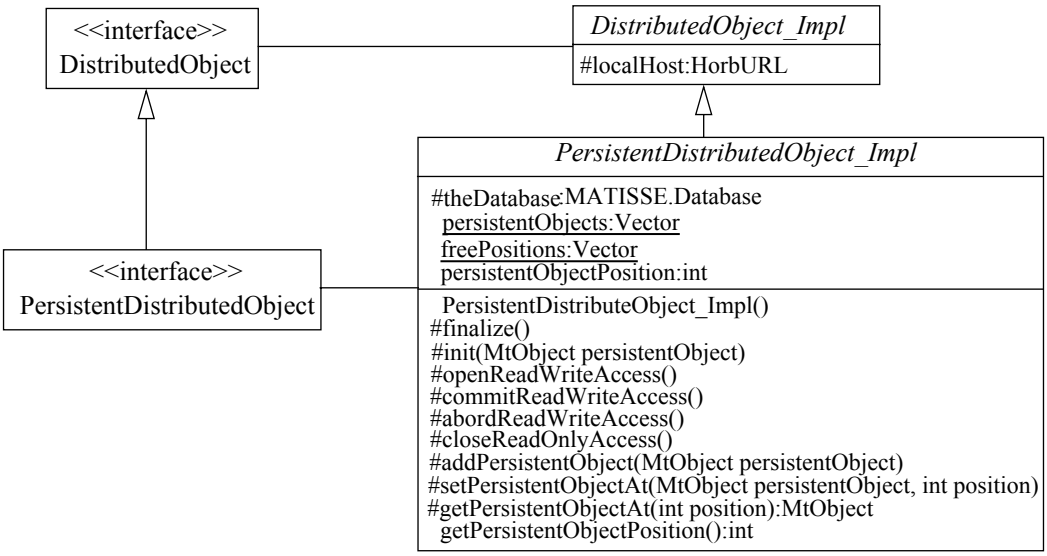


Figure 56 - Classes mères du serveur d'objets distribués

La classe *persistentDistributedObject* a donc deux fonctions principales : la gestion de la mémoire partagée et l'accès à la base de donnée :

- La gestion de la mémoire partagée est réalisée grâce à trois attributs (*persistentObjects*, *freePositions*, *persistentObjectPosition*) et six méthodes (*finalize*, *init*, *addPersistentObject*, *setPersistentObjectAt*, *getPersistentObjectAt*, *getPersistentObjectPosition*). Ainsi lorsque l'on crée un objet distribué, il y a exécution de la méthode *init* qui stocke l'objet persistant dans *persistentObjects* (attribut de classe, donc commun à tous les objets distribués du système) et qui affecte la position de cet objet à l'attribut *persistentObjectPosition* (attribut propre à chaque objet). L'affectation de la position de l'objet distribué est fonction de l'attribut *freePositions* qui permet de savoir s'il y a "des trous" dans *persistentObjects*.
  Ainsi lorsqu'un objet de la partie serveur (objet de classe *XX_Impl*) veut récupérer un objet persistant (le sien ou celui d'un autre objet distribué), il appelle successivement les méthodes *getPersistentObjectPosition,* qui est une méthode publique (donc qui fonctionne sur les classes *XX_Impl* et *XX_Impl_Proxy*), puis la méthode *getPersistentObject,* qui est une méthode protégée, donc qui est connue et qui ne fonctionne que du côté du serveur (pour les objets de type *XX_Impl*).
  Lorsqu'un objet n'est plus référencé, avant d'être supprimé par le garbage collecteur, le destructeur est exécuté (méthode *finalize*), qui supprime l'objet dans l'attribut *persistentObjects* et ajoute la position de cet objet dans l'attribut de *freePosition* pour spécifier que l'emplacement est disponible pour un futur objet distribué.
- L'accès de la base de données est facilité par les méthodes protégées d'accès en lecture et en lecture-écriture.

Ainsi les classes que nous allons distribuer seront toutes des sous-classes de la classe *PersistentDistributedObject*, et les interfaces de ces classes seront des "sous-interfaces" de *PersistentDistributedObject*.

5. Conclusion.

Ce chapitre après quelques rappels, a décrit l'implémentation du prototype Metadyne. Ce dernier se distingue des autres hypermédia adaptatifs par une utilisation constante des dernières technologies (base de données orientée-objet, objet distribués, etc.) avec comme objectif l'obtention d'un système tout objet.

Dans le chapitre qui suit nous allons voir comment ceci prend forme pour les différents acteurs du système, et nous allons vérifier que l'adaptation des cours est réelle.

Chapitre VII -Expérimentation.

Dans ce chapitre, nous allons tout d'abord étudier les différents outils qui sont mis à disposition des acteurs de notre système. Nous commencerons donc par nous intéresser aux deux outils proposés aux enseignants, c'est-à-dire le gestionnaire du modèle du domaine et le créateur de QCME. Puis nous étudierons celui qui est dédié à l'apprenant, c'est-à-dire le gestionnaire de cours. Cette présentation sera alors suivie d'un exemple de production de cours. Afin de bien cerner cet exemple, nous commencerons par définir le sujet du cours, et le modèle du domaine qui en découle. Ensuite nous analyserons le profil de notre apprenant. Nous étudierons alors le cours généré. Enfin avant la conclusion qui mettra en avant les atouts et faiblesses de notre réalisation, nous étudierons les conditions d'utilisation du système.

1. Le système Métadyne.

Le système Métadyne se présente sous la forme d'un site Web classique dont la première page demande aux utilisateurs de se déclarer au niveau du système (grâce à la saisie d'un identifiant et d'un mot de passe). Le système est alors capable de déterminer le type d'utilisateur courant, c'est-à-dire soit un enseignant, soit un apprenant.

Etudions donc ces deux cas de figure.

1.1. Pour les enseignants.

Nous avons vu dans le chapitre IV, que les enseignants sont en charge :

- de construire le modèle du domaine, ou du moins une partie de ce dernier correspondant à leur vision des choses,
- d'identifier les briques élémentaires qui vont permettre de présenter les notions de ce modèle,
- pour le cas particulier des QCME, de définir les questions, réponses et éventuelles sous-questions de chaque questionnaire.

De ce fait lorsqu'un utilisateur est identifié par le système en tant qu'enseignant, le système lui permet d'utiliser deux applications, une pour la gestion du modèle du domaine et l'identification des briques élémentaires, et une autre pour la gestion des QCME.

1.1.1. La gestion du modèle du domaine.

La gestion du modèle du domaine s'effectue via le programme dont l'interface utilisateur est présentée par la figure 57. L'interface utilisateur peut être décomposée en quatre parties.

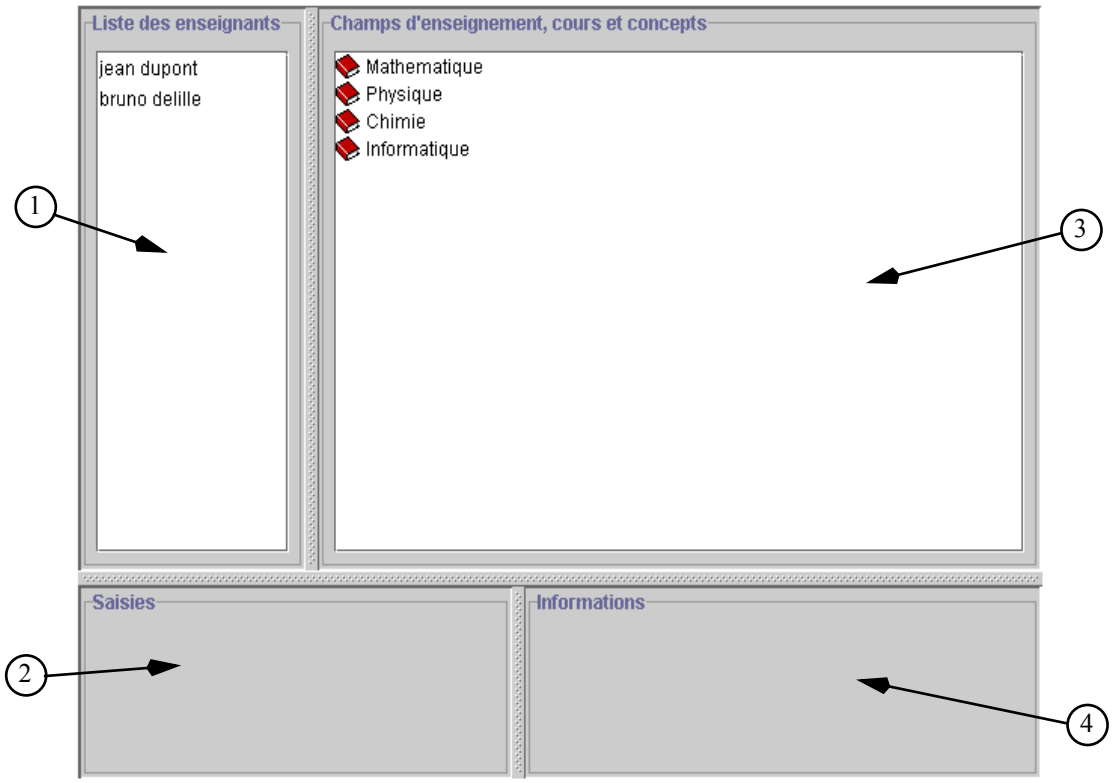


Figure 57 - Le gestionnaire du modèle du domaine

Tout d'abord la partie principale (numéro 3 sur la figue 57) est un arbre hiérarchique qui présente par défaut le point de vue de l'enseignant sur le modèle du domaine. Les noeuds supérieurs de cet arbre contiennent les champs d'enseignement. Par exemple, la figure 58 montre que la "mécanique" et "l'électro-magnétisme" sont des sous-champs d'enseignement de "la physique". Ensuite pour chaque champ d'enseignement, l'enseignement peut visualiser les cours et notions qu'il a précédemment ajoutés (on différencie les cours et les notions par l'icone qui précède son nom, pour les cours et pour les notions). Puis, pour chaque cours et pour chaque notion, l'enseignant peut visualiser les relations (que l'on a définies dans notre modèle conceptuel) qu'il a pu ajouter, ainsi que les briques élémentaires pertinentes qui permettront de présenter cette notion ou ce cours. Les notions ou les briques élémentaires cibles sont alors des sous-feuilles d'un noeud relation, qui visuellement est représenté à l'aide de l'icone suivi du nom de relation. Par exemple sur la figure 58, on peut voir que les notions "Oscillations électriques forcées" et "Oscillations électriques libres" sont reliées par une relation de prérequis ("Nécessite la connaissance de"). De plus, si la relation est pondérée, ou si un ordre est défini dans la présentation des concepts cibles, comme c'est le cas par exemple avec la relation de conjonction (nommée "Se décompose en"), un nombre est ajouté au niveau du concept cible. Par exemple dans la relation que nous venons de voir le concept OEL est suivi du nombre 0,5 (Cf. figure 68).

L'enseignant a dès lors la possibilité de modifier sa vision à l'aide d'un menu contextuel. L'ensemble des commandes proposées par ce menu est alors fonction du noeud sélectionné. Par exemple, la figure 58 présente un menu contextuel correspondant à la sélection d'un cours ou d'une notion. On peut remarquer que la commande "S'associer" est inactive, car cette dernière

n'a d'intérêt que lorsque l'enseignement visualise, comme nous allons le voir par la suite, le point de vue d'un autre enseignant.

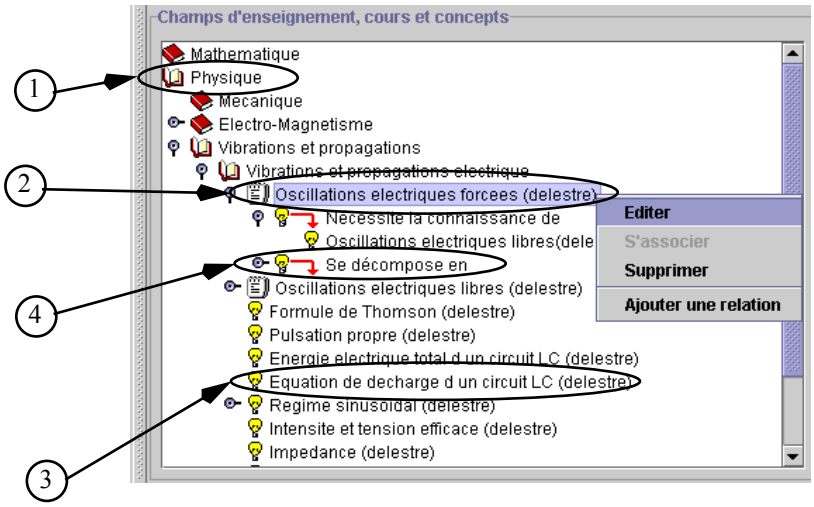


Figure 58 - Modèle du domaine: champs d'enseignement, cours et concepts

Lorsque l'enseignant active une de ces commandes, c'est la zone de saisie numéro 2 de la figure 57, qui devient active (comme le montre la figure 59). Les informations, que doit alors saisir l'enseignant, dépendent bien évidemment de la commande qui a été sélectionnée. Par exemple la figure 59 montre la zone de saisie correspondant à l'action "Ajouter une relation".

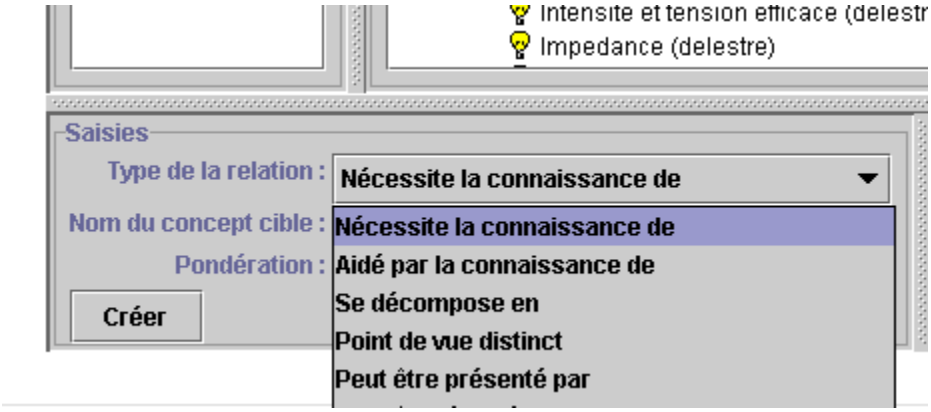


Figure 59 - Modèle du domaine: exemple de saisie

Enfin, la dernière fonctionnalité de cette application, est la possibilité pour l'enseignant de visualiser en plus de sa vision, la vision de ses collègues. Il lui suffit pour ce faire, de sélectionner les enseignants désirés dans la zone 1 de la figure 57. La sélection, ou la désélection, d'un enseignant entraîne alors une modification de l'arbre des concepts. Comme l'indique le figure 60, lorsqu'au moins deux enseignants partagent un même point de vue sur un concept, le nom de ce dernier est suivi du nom de ces enseignants. Ensuite lorsqu'un concept est une notion pour un enseignant, alors que c'est un cours pour un autre, le nom de ce concept est précédé de l'icône  .

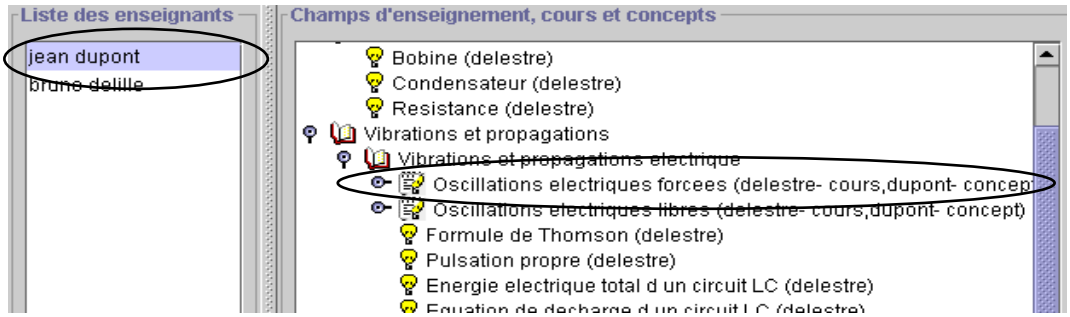


Figure 60 - Modèle du domaine: visualisation du point de vue des autres enseignants

1.1.2. La production de QCME.

Le deuxième outil du projet Métadyne, accessible pour les enseignants, est le constructeur de QCME.

Comme le montre la figure 61, cette application est constituée de deux zones. Tout d'abord une zone de sélection, sur le côté gauche, qui permet à l'enseignant de gérer ces propres QCME, c'est-à-dire :

- de créer ou supprimer des QCME,
- de créer, d'éditer ou de supprimer des questions d'un QCME,
- de créer, d'éditer ou de supprimer des réponses à une question d'un QCME,
- de créer, d'éditer ou de supprimer des sous-questions aux réponses d'une question d'un QCME.

Mais l'enseignant a aussi la possibilité de visualiser les QCME des autres enseignants. Si l'un de ces derniers l'intéresse, il peut alors le dupliquer. Dans ce cas, ce nouveau QCME est ajouté à sa liste, ce qui lui permet de le modifier si le besoin s'en fait sentir. Comme dans l'application précédente, toutes ces actions s'effectuent à l'aide de menu contextuel au niveau de cet arbre.

L'édition d'un QCME ou de l'un de ses composants (question, réponse et/ou sous-question) s'effectue dans la zone de droite. Le contenu de cette zone dépend alors de l'élément édité. Par exemple dans la figure 61, l'enseignant est en train d'éditer la première question d'un QCME sur les RLC. Il a alors la possibilité de spécifier le nom de cette question, le temps imparti pour répondre, l'URL de la brique élémentaire qui va permettre d'illustrer cette question (il visualise alors cette brique directement dans l'application) ainsi que l'intitulé de la question. L'ensemble de ces informations sera utilisé par la suite pour évaluer l'apprenant (Cf. figure 62). On retrouve en effet dans cette autre application, la brique élémentaire, la question à proprement parler, les réponses prévues ainsi qu'une barre de progression qui permet à l'apprenant de savoir combien de temps il lui reste pour répondre.

1.2. Pour les apprenants.

Lorsque l'utilisateur est un apprenant, le système est beaucoup plus simple à utiliser, puisqu'il ne lui propose qu'une seule et unique application. Grâce à cette dernière, l'apprenant peut alors choisir de visualiser un cours ou bien spécifier son modèle de l'apprenant ou plus exactement son modèle comportemental tel que l'on a pu le voir dans le paragraphe 3.3.2 du chapitre IV.

Ainsi, comme le montre la figure 63, cette application est principalement constituée d'une série d'onglets qui permettent à l'apprenant de spécifier ses choix. Cela passe par le choix du cours à visualiser, le choix du point de vue d'un ou de plusieurs enseignants, de la structure des cours qu'il va visualiser, de ses préférences au sujet du type physique des média, enfin de la liberté de navigation que doit lui laisser le système. Etudions donc une à une ces possibilités.

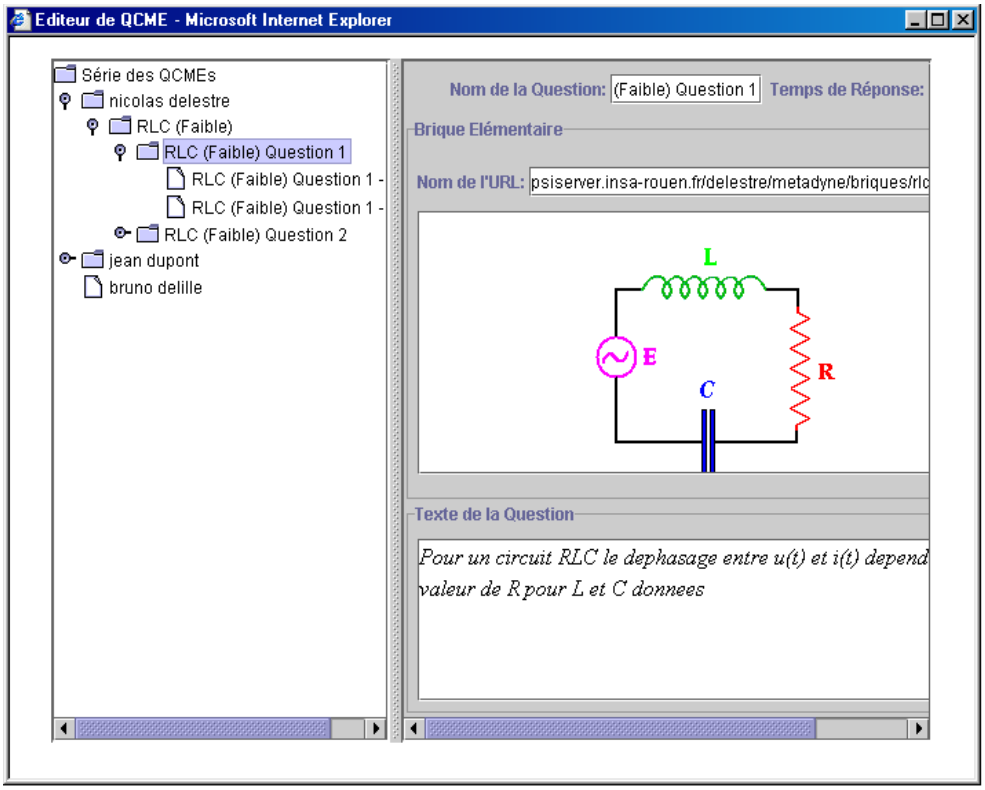


Figure 61 - Editeur de QCME

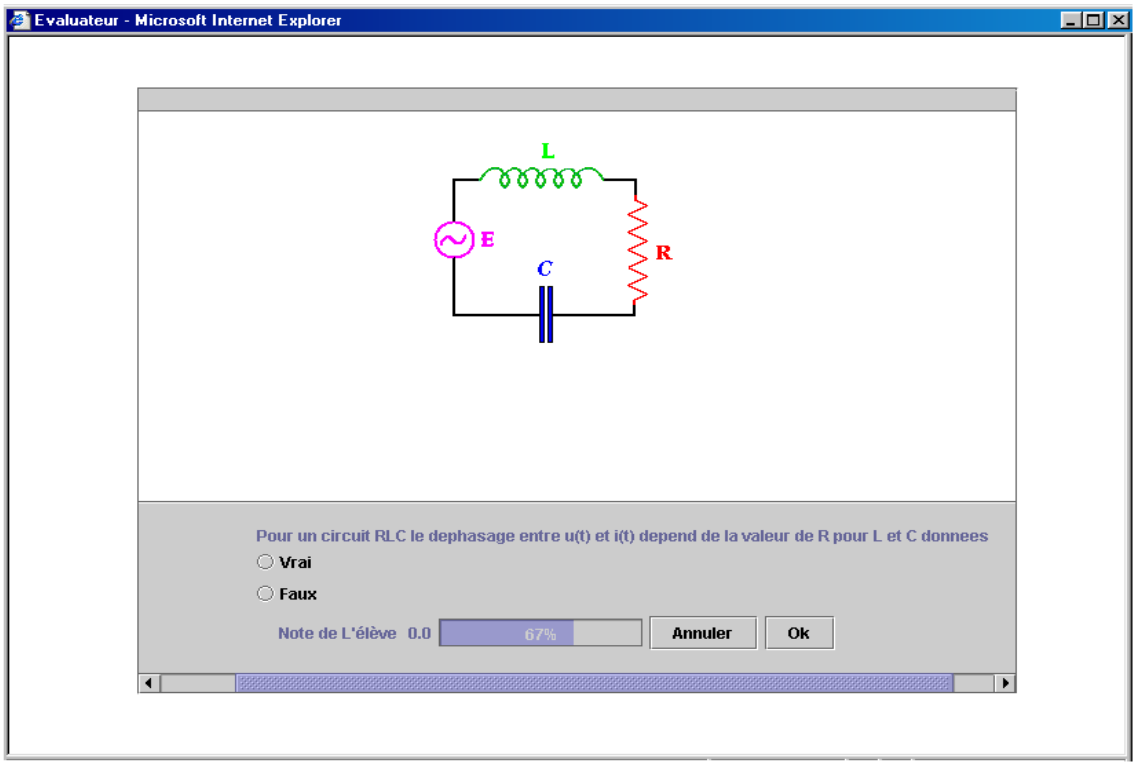


Figure 62 - Evaluation des apprenants

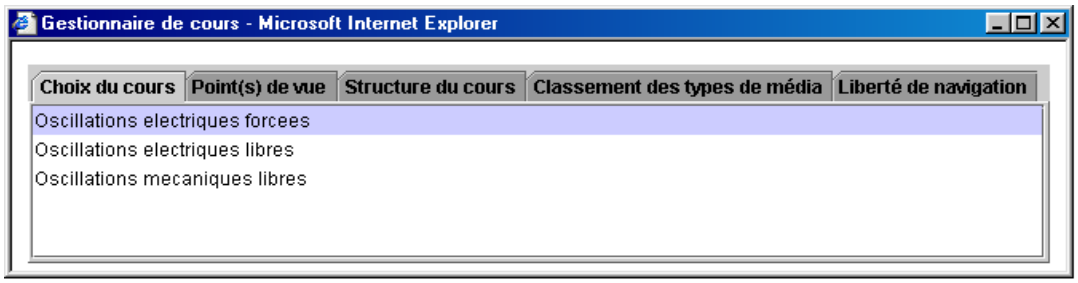


Figure 63 - Gestionnaire de cours: choix d'un cours

1.2.1. L'onglet "choix du cours".

Ce premier onglet permet à l'apprenant de visualiser l'ensemble des cours disponibles sur le serveur (liste de cours qui est bien entendue fonction des choix effectués dans l'onglet "Point(s) de vue"). Il a alors la possibilité d'en sélectionner un, ce qui a pour conséquence, comme nous le verrons plus en détail par la suite, d'ouvrir une nouvelle fenêtre au sein de son navigateur lui permettant ainsi de visualiser le cours désiré. Bien entendu, pendant la visualisation de ce dit cours, l'application "Gestionnaire de cours" est toujours active, ce qui lui permet a tout moment de modifier les caractéristiques du cours courant, ou encore de pouvoir visualiser dans une autre fenêtre un autre cours.

1.2.2. L'onglet "Point(s) de vue".

A l'aide de ce deuxième onglet (Cf. figure 64), l'apprenant a la possibilité de choisir le ou les points de vue qui sont utilisés pour construire un cours. Il visualise donc l'ensemble des enseignants inscrits dans le système, qu'il peut alors sélectionner ou désélectionner.

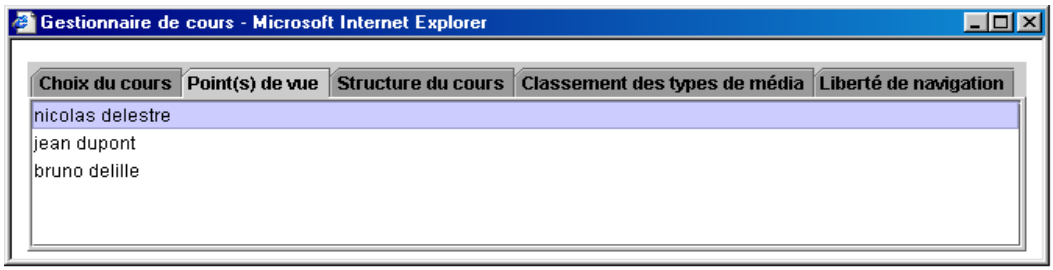


Figure 64 - Gestionnaire de cours: choix d'un ou plusieurs enseignants

1.2.3. L'onglet "structure du cours".

L'onglet "structure du cours" quant à lui permet à l'apprenant de choisir ce que l'on a nommé précédemment un canevas, c'est-à-dire un ensemble trié de types cognitifs qui va permettre au système de déterminer l'organisation de la présentation de chaque notion. Comme nous l'avons vu auparavant, l'apprenant n'a pas la possibilité de créer des canevas, il peut juste sélectionner les canevas qui ont été définis au préalable par les enseignants. Ainsi comme le montre la figure 65, l'apprenant visualise à gauche la liste des canevas disponibles, et lorsqu'il en sélectionne un, il peut visualiser à droite l'organisation générale des cours qui vont lui être soumis.

1.2.4. L'onglet "Classement des types de média".

Cet onglet permet à l'apprenant de spécifier ses préférences au niveau du type physique des média qui peuvent être utilisés pour introduire chaque notion. Comme le montre la figure 66, la

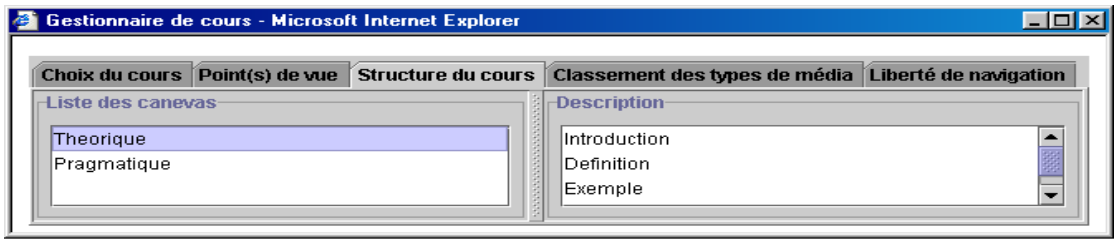


Figure 65 - Gestionnaire de cours: choix d'un canevas

liste de gauche énumère l'ensemble des différents types physiques de média déclarés au sein du système. La liste de droite quant à elle représente les préférences de l'apprenant: le premier type de média référencé étant considéré comme celui qui doit être prioritairement utilisé. L'apprenant n'est pas obligé de référencer tous les types de média. Pour en ajouter un, il le sélectionne dans la liste de gauche et clique sur le bouton "Ajouter". Pour en supprimer un, il le sélectionne dans la liste de droite, puis clique sur le bouton "Enlever". Lorsque l'apprenant décide d'ajouter un type physique non encore référencé, celui-ci est ajouté en tête de liste des préférences. S'il décide de diminuer l'importance de ce dernier, il le sélectionne (dans la liste de droite) et clique sur le bouton "Descendre", ce qui le décale dans la liste. Au contraire, si l'apprenant veut donner plus d'importance à un type physique, il le sélectionne, et clique sur le bouton "Monter".

Figure 66 - Gestionnaire de cours: classement des types physiques de média

1.2.5. L'onglet "liberté de navigation".

Enfin, ce dernier onglet (Cf. figure 67) permet de paramétrer l'importance du modèle épistémique de l'apprenant lors de la création des cours, et par voie de conséquence, la liberté de navigation de l'apprenant. Si l'apprenant décide d'utiliser un système très strict, l'application l'obligera alors à bien connaître les concepts qui sont prérequis du concept courant (connaître signifie dans ce cas que sa note sur le concept cible est au moins supérieure à la pondération associée à la relation de prérequis correspondante). Au contraire, plus l'apprenant choisira une liberté de navigation importante, moins le système tiendra compter du modèle épistémique.

Maintenant que l'on a vu les différents outils, nous allons étudier la création automatique de cours à partir d'un cas d'école.

2. Un exemple d'adaptation.

Dans ce sous-chapitre, nous allons voir un exemple de production automatique de cours. Nous allons tout d'abord définir de façon globale notre cours. Ensuite nous allons étudier le modèle du domaine correspondant en le restreignant au point de vue d'un enseignant. Ensuite nous allons examiner le profil de notre apprenant. Enfin nous verrons l'organisation et le contenu du

Figure 67 - Gestionnaire de cours: Souplesse du système

cours à proprement parler.

2.1. Le sujet du cours.

Depuis le début de ce mémoire, quand nous voulions nous référer à un exemple de cours concret, nous nous sommes intéressés aux notions d'oscillations, et tout particulièrement aux oscillations électriques libres et forcées. Encore une fois, nous allons rester dans ce champ d'enseignement, et ce pour plusieurs raisons :

- Nous pensons que la science physique est l'un des domaines d'enseignement le plus propice à tirer parti des avantages que peut apporter l'enseignement assisté par ordinateur, et tout particulièrement les systèmes de production de cours multimédia [Mallet & al 99].
- Les expériences sur les oscillations de par leurs aspects théoriques parfaitement maîtrisés sont facilement simulables (il suffit d'effectuer une simple recherche sur Internet pour trouver des dizaines d'*applets* simulant aussi bien les oscillations mécaniques, qu'électriques).
- Ce sont des notions qui sont souvent très difficilement assimilables par les élèves du secondaire, en l'occurrence des élèves de terminale scientifique. Elles doivent donc être prioritairement traitées.

Le sujet de ce cours va donc porter sur les oscillations électriques, mais contrairement à ce que l'on a vu dans le chapitre IV, on va s'intéresser plus particulièrement aux oscillations électriques forcées, nommées OEF ou études des circuits RLC, qui est un cours qui se trouve en aval par rapport au cours sur les OEL (existence d'une notion de prérequis entre les OEL et OEF). Ceci introduit donc le modèle du domaine que nous allons maintenant expliciter.

2.2. Le modèle du domaine.

La première difficulté consiste à déterminer à quel champ d'enseignement se rapporte la notion d'OEF. Nous avons décidé de nous appuyer sur [Bramand & al 83], qui considère que ces problèmes d'oscillation font partie du champ d'enseignement "Vibrations et propagations". Nous lui avons alors ajouté deux sous-champs d'enseignement: "Vibrations et propagations mécaniques" et "Vibration et propagations électriques". Le premier champ d'enseignement regroupe toutes les notions qui sont en rapport avec les OML, et le second tout ce qui touche aux notions d'OEL et d'OEF.

Ensuite, nous nous sommes posés des questions au sujet des tenants et aboutissants de la notion d'OEF. Comme nous venons de le voir, la notion d'OEF a pour prérequis la notion OEL. Ensuite, toujours d'après notre référence, la notion d'OEF passe par la présentation de la notion de régime sinusoïdal, puis par la notion d'impédance enfin par la notion de résonance d'intensité. De ce fait la notion OEF est reliée à ces trois notions par une relation de conjonction.

Nous avons alors effectué ce même travail d'analyse pour chacune de ces trois notions, ce qui en fin de compte donne le modèle du domaine présenté par la figure 68.

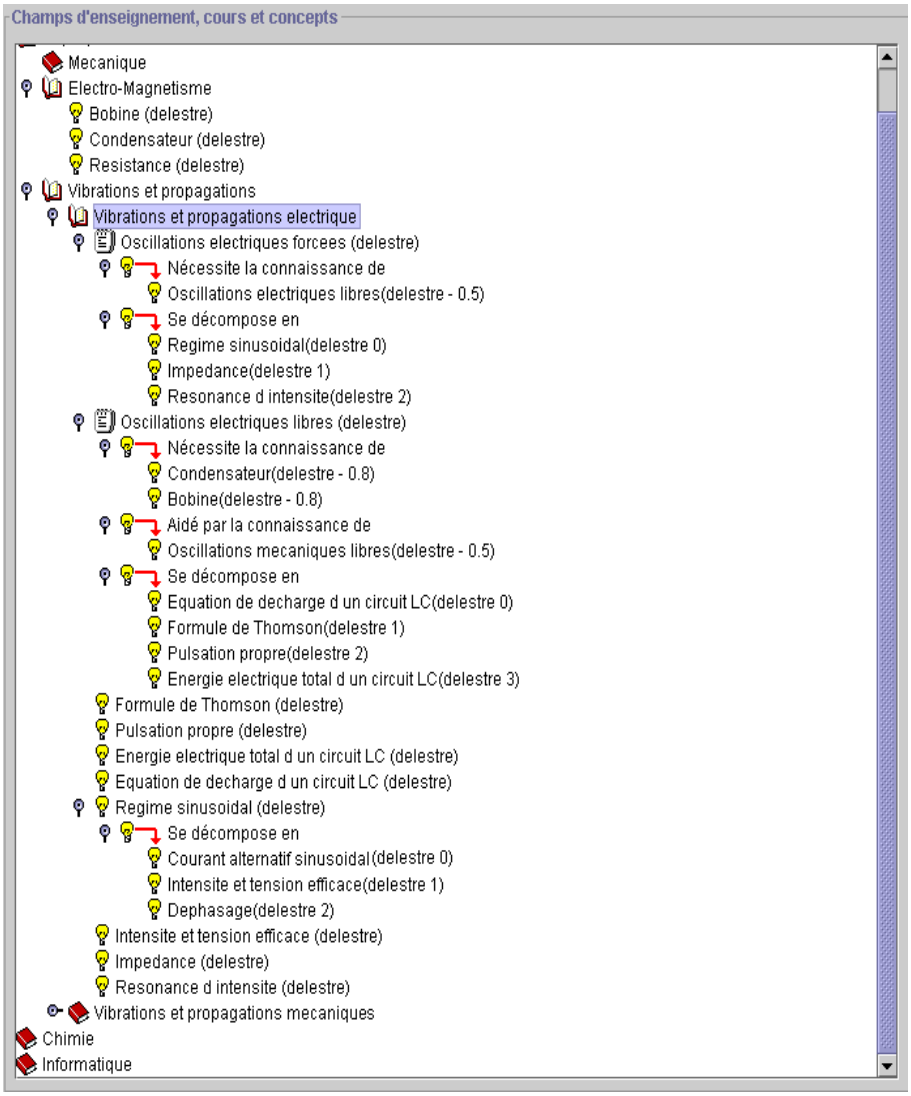


Figure 68 - Modèle du domaine: zoom sur le cours sur les OEF

2.3. Le profil de l'apprenant.

Définir le profil de l'apprenant revient à définir l'état courant de son modèle épistémique et de son modèle comportemental. Pour ce qui est du modèle épistémique, nous avons décidé qu'à la première connexion, le modèle est totalement vierge, c'est-à-dire qu'il ne référence aucune notion du modèle du domaine. Par contre le modèle comportemental est convenablement référencé, puisque c'est l'apprenant qui le détermine à l'aide du gestionnaire de cours, c'est-à-dire :

- Le point de vue choisi est celui de l'enseignant "nicolas delestre",
- Le canevas choisi est le canevas théorique, qui propose d'introduire un cours de la façon suivante: introduction, définition, exemple, exercice,
- Les types physiques sont classés comme suit: texte, vidéo, animation, graphique,
- Enfin dans un premier temps la liberté de navigation est définie comme "très stricte".

2.4. Le cours généré.

Lorsque l'apprenant clique sur le cours "Oscillations électriques forcées" de l'onglet "Choix du cours", une nouvelle fenêtre s'ouvre au sein de son navigateur identique à celle présentée par la figure 69.

Figure 69 - Exemple de cours sur les OEF (système très strict)

Que remarque-t-on ?

Tout d'abord, le nom de cette fenêtre correspond au nom du cours demandé. Ensuite, la page est divisée en deux parties. La partie de gauche présente le plan du cours de façon hiérarchique, ainsi que la notion qui est active (celle qui est en bleu, c'est-à-dire "Courant alternatif sinusoïdal"). Ce plan est automatiquement créé à partir de la vision sur le modèle du domaine des enseignants sélectionnés par l'apprenant. Sur la partie de droite on peut visualiser en théorie une présentation de la notion active. Ce n'est pas le cas ici, car le modèle épistémique de l'apprenant est totalement vide. Or comme la liberté de navigation est très stricte, le système considère qu'il est indispensable à l'apprenant de voir en priorité deux notions. Ces notions sont des notions de prérequis au cours. La première est un prérequis aux cours sur les OEF, et la seconde est un prérequis de la notion courante. Le pourcentage qui est associé à ces notions représente l'importance pour les enseignants de la connaissance de ces notions dans l'apprentissage du cours. Si l'apprenant clique sur le premier lien, une nouvelle fenêtre apparaît, correspondant au cours sur les oscillations électriques libres, comme l'indique la figure 70. Et le même processus recommence.

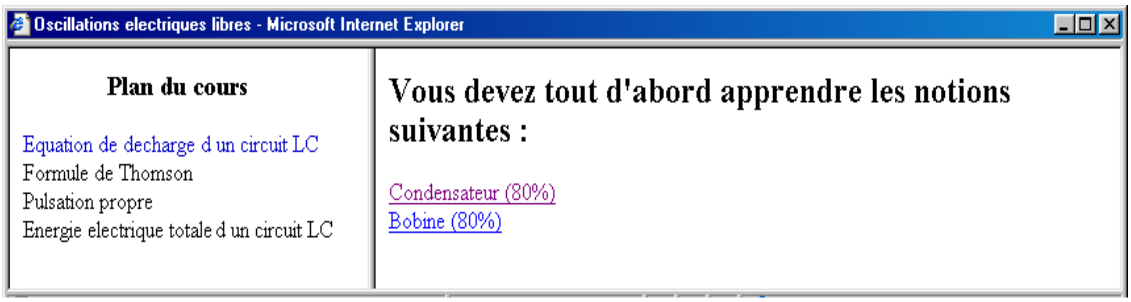


Figure 70 - Exemple de cours sur les OEL (système très strict)

Imaginons maintenant que l'apprenant décide d'avoir un système un peu moins strict. S'il décide de visualiser de nouveau le cours sur les OEF, comme le montre la figure 71, le système ne l'oblige plus qu'à apprendre les notions liées aux OEL, car il considère alors que la relation de prérequis entre la notion de courant alternatif sinusoïdal et le cours sur les fonctions trigonométriques n'est pas si primordiale.

Enfin dans le cas où l'apprenant désire avoir un système plus libre (ou lorsque les pondérations

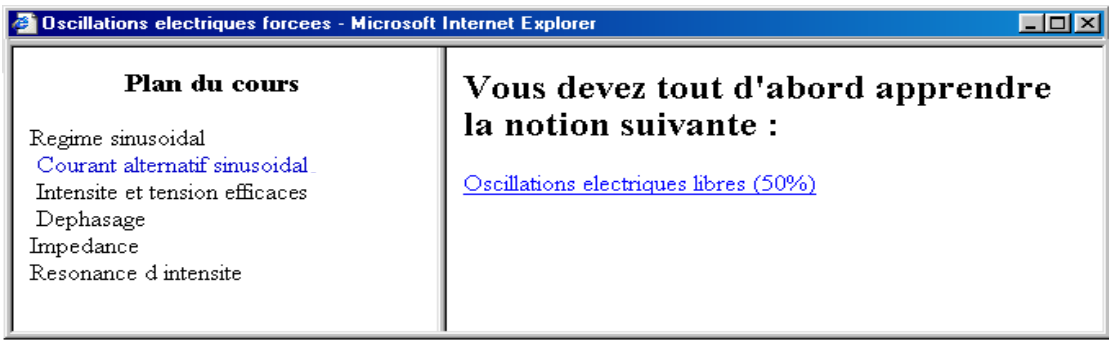


Figure 71 - Exemple de cours sur les OEF (système moins strict)

de son modèle épistémique sont en adéquation avec les valeurs associées aux relations de prérequis), le système construit réellement le cours, comme le montre la figure 72.

Cette fois les notions de prérequis ne sont plus contraignantes, mais elles restent activables au cas ou l'apprenant voudrait visualiser les cours correspondants. Sous ces liens hypertextes apparaît alors le cours à proprement parler. On retrouve l'organisation choisie par l'apprenant, c'est-à-dire le canevas, et pour chaque partie de ce canevas une brique élémentaire est sélectionnée par le système comme nous l'avons vu dans le paragraphe 5.2. du chapitre IV. A la fin de cette page, l'apprenant pourra trouver un lien hypertexte nommé "suivant", qui s'il est activé, permettra à l'apprenant de visualiser la notion suivante du cours, c'est-à-dire dans notre cas la notion "Intensité et tension efficaces". Bien entendu pour les autres pages, exceptée la dernière, les pages possèdent les liens "suivant" et "précédent".

3. Quelques aspects techniques.

Sans revenir dans les détails, définissons maintenant les conditions d'utilisation de notre système et explicitons plus précisément la nature des applications que nous venons de voir.

3.1. Les conditions d'utilisation du système.

Nous avons vu dans le chapitre précédent (Cf. figure 45) que les applications que nous avons développées étaient accessibles depuis un simple navigateur Web. En fait, ceci n'est pas tout fait exact. En effet, notre prototype a été développé entièrement en Java 2 (JDK 1.2). Or cette version du Java n'est pour l'instant pas inclus au sein des navigateurs standards (dans l'ensemble ils sont tous pour l'instant compatibles avec la version 1.1.6 ou 1.1.7 du langage de Sun). Toutefois, la société Sun a développé des *plugs-in* pour tout de même pouvoir exécuter des *applets* écrites avec la version 2 de Java. Cependant ces *plugs-in* ne fonctionnent que sous un environnement Windows (95, 98 et NT) et Sun Solaris, pour les navigateurs Internet Explorer et Netscape sous Windows et Netscape sous Sun Solaris. Mais, ceci n'est qu'une question de temps, et il est fort à parier que d'ici peu, l'ensemble des navigateurs du marché sera compatible avec cette nouvelle version du langage Java.

De ce fait, la première page du serveur vérifie que le *plug-in* adéquat est bien présent, et si tel n'est pas le cas, il propose à l'utilisateur de le télécharger depuis le site Internet de Sun et de l'installer. Une fois cette opération effectuée, l'utilisateur doit alors modifier le fichier de configuration de sécurité de la machine virtuelle Java (par défaut le fichier *policy.security*) afin que les différentes applications de notre prototype puissent communiquer sans problème avec le serveur d'objets distribués. Bien entendu, cette phase de paramétrage n'est effectuée qu'une seule et unique fois.

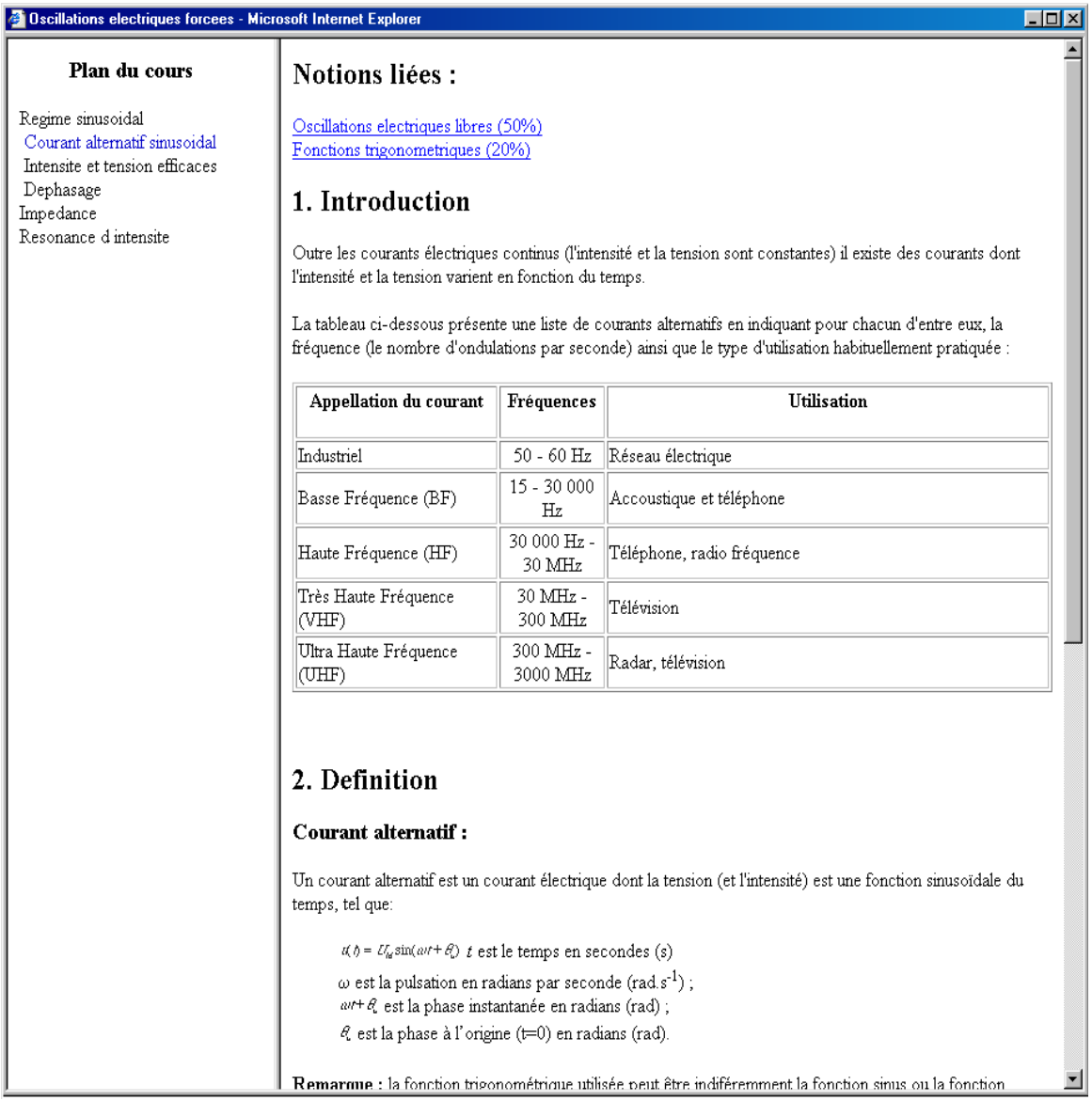


Figure 72 - Exemple de cours sur les OEF (système libre)

3.2. La nature des différentes applications.

Il faut distinguer les applications, dites finales, que nous venons de voir, qui sont directement utilisées par les acteurs du système, des applications qui sont implicitement utilisées par ces derniers.

3.2.1. Les applications dites “finales”.

Parmis ces applications, il faut distinguer celles qui sont destinées aux enseignants et celles qui sont destinées aux apprenants. Les premières peuvent être utilisées au sein d’un navigateur, mais ce n’est pas obligatoire, alors que les secondes le sont obligatoirement.

Ainsi le gestionnaire du modèle du domaine, et le créateur de QCME ont la particularité d’être à la fois des *applets* et des programmes indépendants Java, permettant alors aux enseignants réguliers d’éviter le quelque fois fastidieux téléchargement de l’*applet*.

Par contre le gestionnaire de cours ainsi que l’évaluateur de QCME ne sont disponibles que sous forme d’*applet* car ces applications doivent être constamment en liaison avec un navigateur

Web.

3.2.2. Les applications "cachées".

Les applications qui sont implicitements utilisées par les utilisateurs sont celles qui ont pour objectifs de déterminer le type d'utilisateur courant et de construire les cours à proprement parler. Dans le premier cas, l'application est invoquée à chaque connexion d'un utilisateur, et dans le deuxième cas, elles (car il y en a plusieurs) sont invoquées soit par le gestionnaire de cours lorsque l'apprenant désire visualiser un cours particulier, soit par les liens hypertextes intrinsèques aux cours que visualise l'apprenant (liens de prérequis, ou liens "précédent" et "suivant" que nous avons vu précédemment).

Ces applications, toujours écrites en Java sont des *servlets*. Elles sont pour l'instant aux nombres de quatre.

- La première qui se nomme *Login* détermine le type d'utilisateur et crée la page d'accès aux fonctions du système.
- La seconde qui se nomme *CoursesMaker* est en charge de produire le code HTML permettant d'afficher les deux frames qui composent un cours, c'est-à-dire le cadre de gauche qui va contenir le plan du cours, et le cadre de droite qui va contenir le cours à proprement parler (Cf. figure 73).
- La troisième nommée *CoursesMap* crée justement le code HTML permettant à l'apprenant de visualiser le plan du cours.
- Enfin, la quatrième, nommée *CoursesContent*, crée le code HTML permettant à l'apprenant de visualiser le cours à proprement parler.

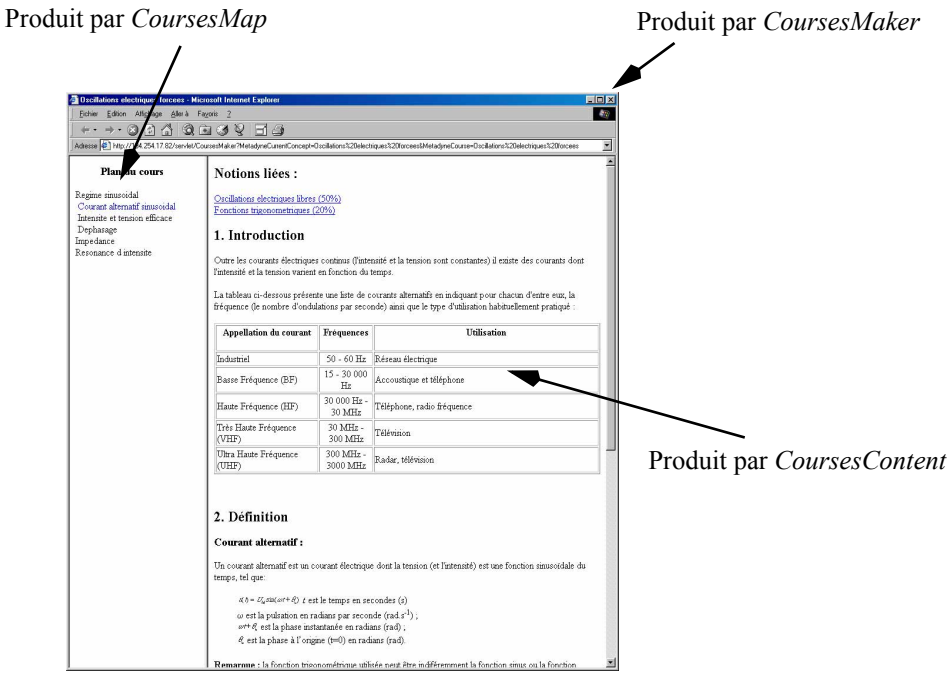


Figure 73 - Les applications qui construisent les cours.

Ainsi si on reprend la figure 42, on peut remplacer les mots génériques *servlets* et *applets* par le nom des applications, telle que le présente la figure 74.

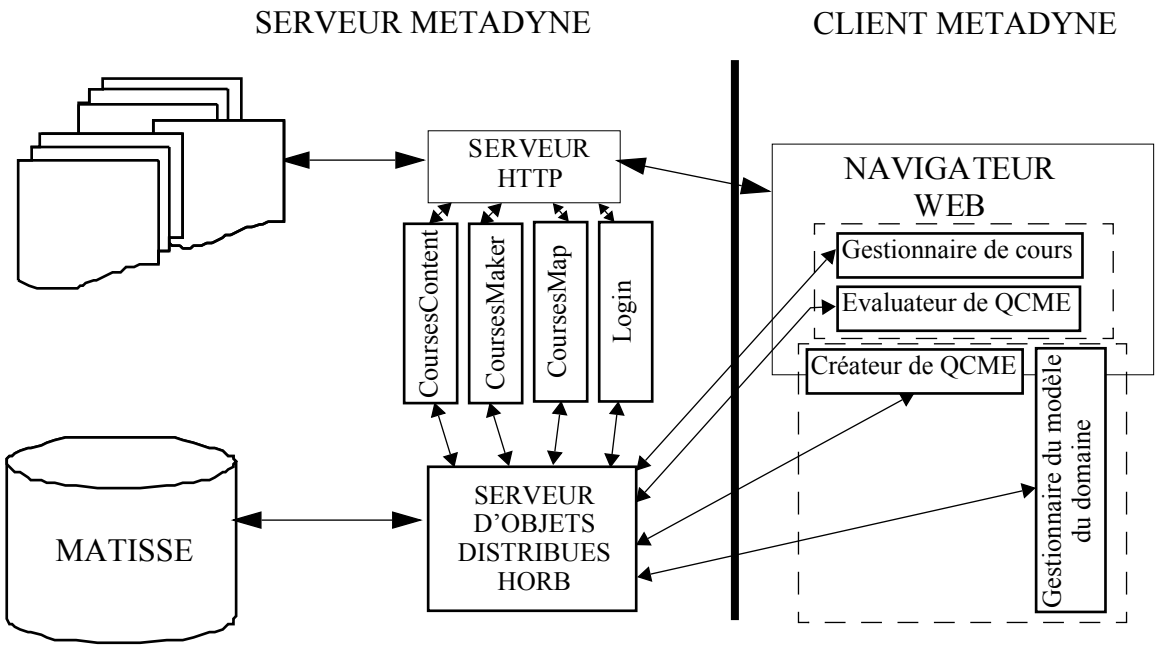


Figure 74 - Architecture logicielle de METADYNE: Qui fait quoi ?

4. Conclusion.

Nous venons d'étudier les différents outils mis à disposition des enseignants et des apprenants, ainsi qu'un petit exemple de production de cours adapté aux caractéristiques de l'apprenant.

Cet exemple, qui nous en sommes conscients ne peut valider complètement les qualités de notre système, nous a tout de même permis de faire quelques constations.

Commençons tout d'abord par les points positifs, c'est-à-dire principalement la validation de la modélisation et de l'architecture logicielle que nous avons décidée d'adopter. En effet, à notre connaissance, il n'existe aucun serveur d'objets distribués qui distribue réellement les objets comme nous le faisons. Dans notre système, les clients de ce serveur (dans notre cas les *servlets* et les *applets*) utilisent des objets distribués comme ils les utiliseraient en local. Si un client à besoin à un instant *t* d'un objet distant, il l'instancie naturellement. Cela permet donc de concevoir des programmes clients de manière transparente, ce qui facilite grandement leur modélisation et surtout leur implémentation. Cette facilité permettra d'implémenter rapidement les quelques applications qui manquent encore, c'est-à-dire un outil d'administration (pour créer et supprimer les comptes utilisateur) ainsi qu'un outil de gestion de canevas (qui sera certainement intégré dans l'application de gestion du modèle du domaine).

Revers de la médaille, le serveur d'objets peut être amené, si le nombre de clients qui l'utilisent augmente, à créer énormément d'objets distribués. A priori on ne sait pas trop comment ce dernier va se comporter si plusieurs dizaines voire centaines de clients se connectent en même temps. Durant nos tests, nous avons pu constater, qu'en moyenne, nos programmes avaient constamment environ une cinquantaine d'objets actifs. Or auparavant, nous avions validé la charge du serveur HORB (nous avions créé jusqu'à trois milles objets "simples"). On peut donc raisonnablement penser que notre système devrait convenablement se comporter pour au moins une vingtaine d'utilisateurs connectées au même moment.

Ensuite, cette expérimentation a confirmer une intuition que nous avions eu dès le début, c'est-à-dire l'importance du nombre de briques différentes pour pouvoir prétendre avoir un système qui s'adapte réellement à l'utilisateur: une dizaine de briques par notion ou cours et par type co-

gnitif semble être un minimum. Or la production de brique élémentaire n'est pas des plus triviale, et coûte énormément cher en temps et/ou en argent. D'où l'intérêt et l'importance de serveurs tels que le serveur SEMUSDI que nous avons déjà vu, et dont nous allons étudier l'architecture en annexe.

Bilan.

1. Conclusions.

A l'aube du vingt et unième siècle, les progrès technologiques sont en train de bouleverser totalement les données de l'enseignement traditionnel. On se dirige doucement vers un apprentissage personnalisé et délocalisé, et nous pensons fortement que nous nous orientons progressivement mais irrémédiablement vers une production de documents pédagogiques électroniques qui va supplanter d'ici quelques années les documents papiers, qui restent aujourd'hui le support de prédilection de la transmission de l'information en général et de l'éducation en particulier.

En effet les avantages intrinsèques aux documents électroniques pédagogiques sont nombreux, on peut citer entre autre :

- l'accès immédiat à l'information, quelque soit l'heure ou la position géographique de l'usager. Cela ne va aller qu'en s'accentuant, puisque d'ici peu de temps il sera possible de naviguer sur le Web avec des appareils très simples d'utilisation et totalement indépendants.
- les liens hypertextes qui permettent d'associer différentes notions.
- la diversité des média utilisables pour introduire une notion.
- la possibilité de personnaliser à souhait les informations qui sont transmises à l'utilisateur.

Et les quelques critiques existantes encore aujourd'hui qui plaident encore pour l'utilisation du papier n'auront d'ici quelques années plus lieu d'exister. Par exemple :

- la fatigue engendrée par la lecture d'information sur écran aura bientôt disparue, puisque l'on nous annonce d'ici deux ou trois ans des écrans dont la résolution graphique sera équivalente voire supérieure aux documents papiers (des écrans d'une résolution supérieure à 300 points par pouce existent déjà dans les laboratoires de recherche). Il existe même des projets de papiers électroniques, c'est-à-dire des livres dont le contenu des pages peut être modifié à souhait.

- la possibilité d'annoter les documents, de pouvoir surligner des informations, ne sont plus des utopies sur les documents électroniques.

Toutefois, en attendant ces progrès technologiques, il est indispensable de préparer les solutions qui permettront dans un avenir proche de délivrer des cours multimédia adaptés aux caractéristiques de chaque apprenant.

C'est donc dans cette problématique que s'inscrit notre mémoire, puisque nous avons tenté de décrire le plus clairement possible un système de production de cours hypermédia "intelligent" basé sur la réutilisation d'items didactiques multimédia ciblés que l'on a nommé brique élémentaire. Cette étude a dès lors débouché sur l'implémentation d'un prototype.

Pour ce faire, nous avons explicité et catégorisé les connaissances qui d'une part sont du ressort des créateurs de documents pédagogiques (dans notre cas les enseignants) et d'autre part celles qui interviennent dans la personnalisation d'un cours, c'est-à-dire les préférences générales de l'apprenant, mais aussi ses objectifs pédagogiques ainsi que ses connaissances. Nous avons dès lors essayé de formaliser ces connaissances, ce qui nous a amené à définir un modèle conceptuel décomposable en quatre parties, c'est-à-dire le modèle du domaine, le modèle de l'apprenant, le module de gestions des briques élémentaires et le générateur de cours.

Nous avons par la suite défini un modèle objet basé sur le définition du concept d'étiquette et de relation permettant d'implémenter dans une base de données objet, et donc de rendre persistant, le modèle du domaine, le modèle de l'apprenant et les informations sur les briques élémentaires.

Enfin nous avons développé un prototype basé sur une chaîne de trois systèmes client-serveur (la base de données objet, le serveur d'objets distribuées et le serveur Web) permettant aux différents acteurs du système depuis un simple navigateur Web de spécifier les connaissances du modèle du domaine, du modèle de l'apprenant, et celles qui caractérisent les briques élémentaires.

Notre travail se distingue des autres travaux de la communauté scientifique sur les hypermédia adaptatifs pour l'enseignement tel que [Carver & al 96], [Laroussi & al 98] ou encore [Carro & al 99] de par :

- les fonctionnalités offertes aux différentes acteurs du système,
- l'architecture logicielle que nous avons adoptés.

Tout d'abord, Métadyne oblige les enseignants à suivre une certaine méthodologie pour construire leurs cours. Il ne leurs suffit pas d'agancer simplement des briques élémentaires, mais ils doivent au contraire suivre une démarche pédagogique. Cette démarche consiste tout d'abord à définir les notions présentées dans le cours, puis à définir les tenants et aboutissants pour chaque notion, et finalement à choisir les items didactiques qui vont permettre d'introduire les différentes notions. Cela leurs permet donc de concevoir des cours mieux construits, mais en plus, cela leurs permet de pouvoir s'échanger leur point de vue, et donc finalement d'améliorer la structure des cours.

Ensuite, Métadyne offre aux apprenants la possiblité d'être actifs dans la phase d'apprentissage en pouvant "paramétrer" les cours qui vont leurs être proposés. Ainsi cela permet d'avoir un système adaptatif mais aussi adaptable.

Enfin l'architecture logicielle que nous avons adoptée, outre son évolutivité intrinsèque à la modélisation objet effectuée ainsi que l'utilisation d'un serveur d'objets disitribués, de proposer aux utilisateurs de véritables applications (avec tout ce que cela implique en terme de facilitée d'utilisation) totalement intégrées au sein du navigateur Web.

2. Perspectives.

Toutefois, nous sommes conscients que notre travail n'est pas exempt de tout défaut, et qu'il pourra évoluer dans un futur plus ou moins proche.

Tout d'abord nous prévoyons d'expérimenter plus amplement notre prototype en coordination avec l'équipe de M. Tribollet du laboratoire de didactique LIRDHIST - UCBL à Lyon. D'après les discussions que nous avons eu jusqu'à présent, des critiques sont apparues particulièrement au sujet des relations que nous avons définis au niveau du modèle du domaine. M. Tribollet pense en effet qu'à priori les quatre relations que nous avons définies s'accordent particulièrement bien à une utilisation d'organisation de savoir scientifique, mais risquent de ne pas être suffisantes pour d'autres champs d'enseignement. Nous envisageons dès lors de laisser la possibilité aux enseignants d'utiliser les relations déjà existantes ou d'en définir de nouvelles, en spécifiant dans ce dernier cas les caractéristiques principales de ces nouvelles relations (relation pondérée, ordonnée, datée, etc.) et en spécifiant le type de lien hypertexte qui sera utilisé pour les représenter. Au niveau implémentation, l'esprit ouvert de notre modélisation objet, ainsi que les capacités du méta-protocole de la base de données Matisse, ne devrait pas poser de problème.

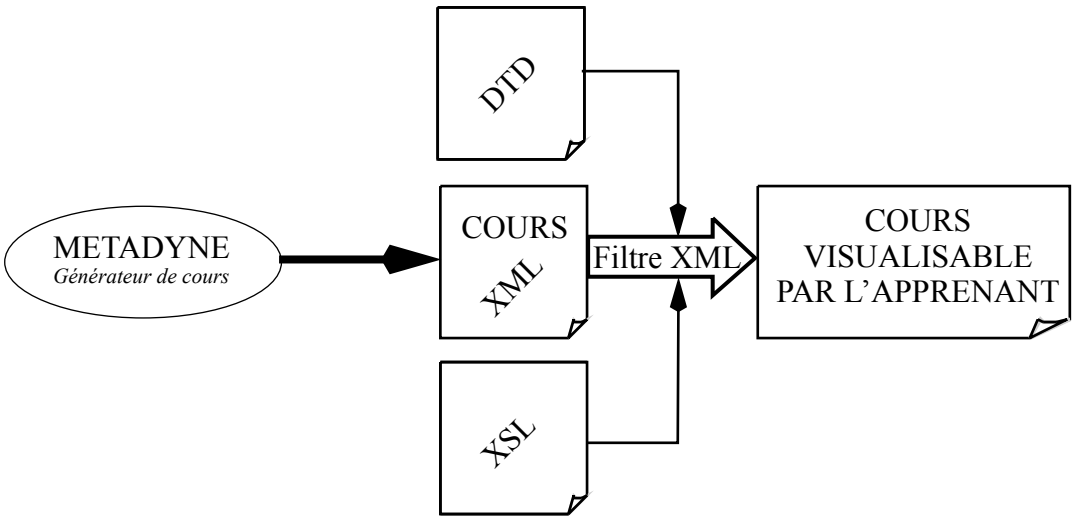


Figure 75 - Utilisation du langage XML pour la construction de cours

Ensuite, parallèlement, il est envisagé d'utiliser le langage XML. En effet, ce langage dans un avenir proche (d'ici deux à trois ans) devrait devenir le langage de référence du Web remplaçant alors le langage HTML. Cette transition permettra d'ajouter une définition sémantique aux documents qui est pour l'instant absente. Nous pensons donc utiliser ce langage pour définir la structure des pages de l'hypermédia créées à la volé. Comme l'indique la figure 75, cela passera par la création d'une DTD qui définira sémantiquement ce qu'est un cours ainsi que les éléments qui le compose, et cela passera aussi par la création d'une feuille de style XSL qui définira la manière de présenter chaque élément composant ce cours.

Annexe : Le projet SEMUSDI.

Nous avons vu brièvement les objectifs du projet SEMUSDI dans le premier chapitre, ainsi que la norme adoptée pour indexer les briques élémentaires.

Dans cette annexe, nous allons nous attarder sur les fonctionnalités du serveur, la modélisation objet qui permet de stocker les briques, ainsi que sur l'architecture mise en oeuvre pour disposer d'un serveur rapide et convivial.

1. Les fonctions du serveur.

Le synoptique de la figure 7 (page 29) correspond à la première page présentée par le serveur sur le Web (Cf. figure 76).

Selon la complexité de la fonction sélectionnée, on est automatiquement dirigé vers d'autres écrans plus ou moins imbriqués.

Présentation de quelques fonctions :

1.1. La recherche de briques puis la consultation.

La fonction de recherche permet de retrouver des briques en spécifiant des mots clefs. La syntaxe de ce moteur s'apparente à celle des moteurs de recherche du Web, comme par exemple altavista: on y peut inclure des opérateurs booléens tels que AND et OR.

Cette recherche n'est pour l'instant effectuée que sur les attributs des briques, c'est-à-dire leur(s) auteur(s), leurs mots-clefs, leur titre ainsi que leur texte de présentation. Une version totalement plein texte est à l'étude, ainsi que l'ajout d'opérateur unaire tel que peut le proposer altavista (comme par exemple NOT, url:, image: etc.).

Une fois la recherche effectuée, le serveur affiche l'ensemble des briques correspondantes (maintenant sous forme d'arbre), permettant ainsi à l'utilisateur de pouvoir les visualiser une à une. Si une des briques l'intéresse, l'utilisateur peut l'ajouter à son panier. Ce dernier contiendra alors toutes les briques qu'aura choisies l'utilisateur, qu'il pourra alors télécharger par la suite.

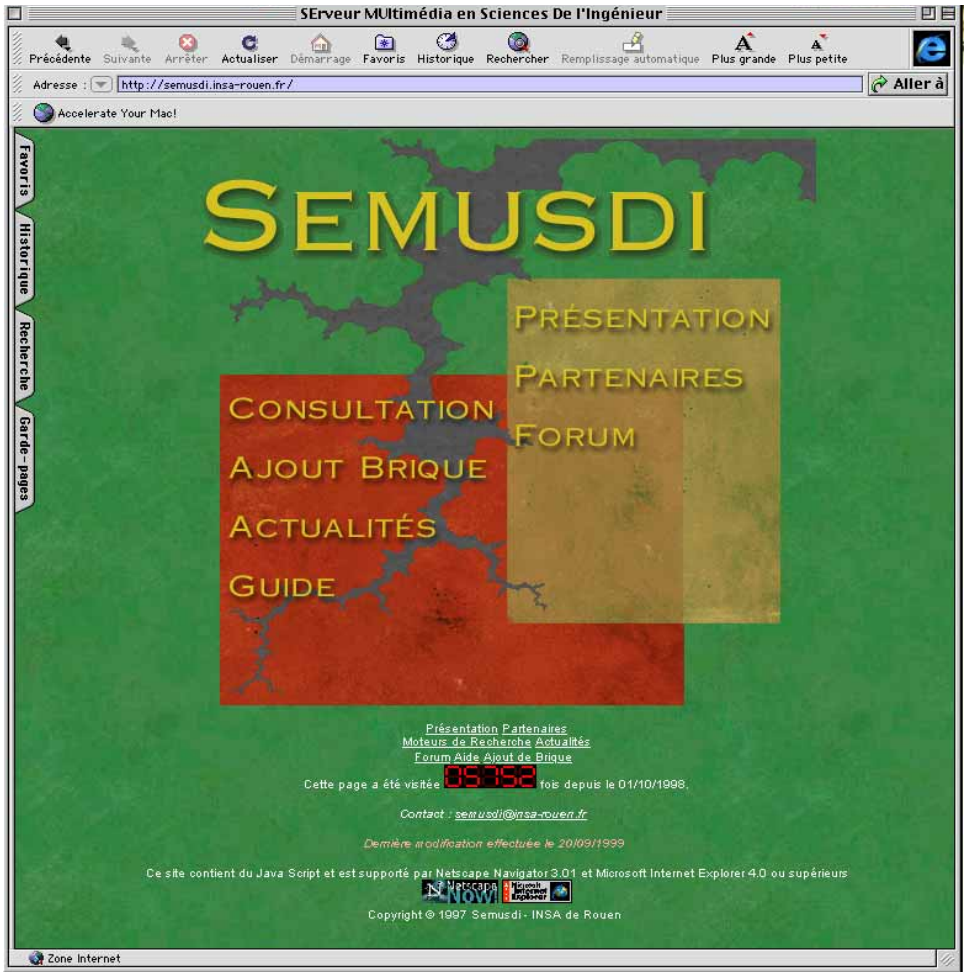


Figure 76 - Page d'accueil du serveur Web SEMUSDI.

1.2. L'ajout de nouvelles briques.

L'ajout de nouvelles briques est une fonction réservée aux utilisateurs identifiés, un mot de passe leur est alors demandé. L'utilisateur, une fois l'identification effectuée, peut alors soumettre des briques en remplissant un formulaire qui correspondant à la carte d'identité de la brique. Une fois cette saisie effectuée et la brique envoyée, cette dernière est analysée pour vérifier qu'il ne s'agit pas d'un document composite (comme par exemple un fichier HTML qui est contiendrait des images). Si tel est le cas, un nouveau formulaire est présenté à l'utilisateur, lui demandant d'envoyer les fichiers annexes.

Une fois la brique envoyée, elle n'est pas immédiatement visualisable, elle doit tout d'abord être validée, ce que nous allons voir maintenant.

1.3. La validation des nouvelles.

Le cycle de validation des briques (Cf. figure 77) permet d'effectuer un tri parmi toutes les briques qui sont soumises, assurant ainsi la qualité du contenu proposé. Pour accéder à ce type de fonction dite fonction de "Back office", il faut être référencé comme relecteur, indexeur, responsable de thème ou administrateur du serveur. L'indexeur et le responsable de thème sont en charge de valider le contenu scientifique, pédagogique et l'aspect visuel de la brique. L'indexeur, comme son nom l'indique est en charge de vérifier les mots clefs de la brique, et si besoin de les changer. Enfin le ou les administrateurs SEMUSDI sont les seules personnes à permettre la mise en libre accès d'une brique.

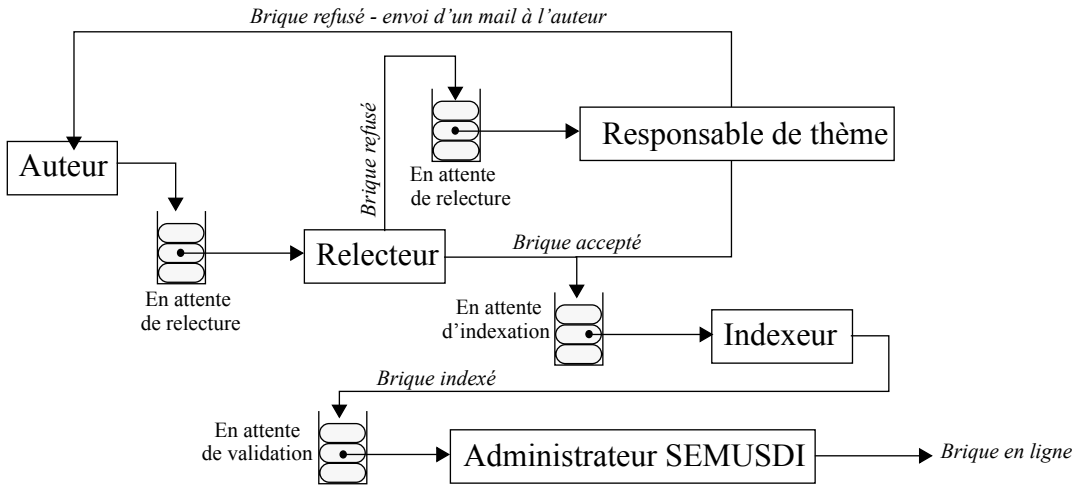


Figure 77 - Cycle de validation des briques élémentaires.

Etudions maintenant le cycle de validation.

Comme nous l'avons vu, toute nouvelle brique est automatiquement mise en attente de validation.

Lorsqu'un relecteur se connecte au serveur (et se déclare en tant que relecteur, puisqu'une même personne peut avoir plusieurs statuts), il peut visualiser toutes les briques en attente de validation qui correspondent à son ou ses domaines de compétence (mais seulement celles dont il n'est pas l'auteur). Il peut alors en choisir une, la visualiser, et l'accepter ou la refuser. S'il l'accepte, elle est directement mise dans la file d'attente de l'indexeur. S'il la refuse, elle est déposée dans la file d'attente des briques provisoirement refusées qui doivent être étudiées par le responsable de thème.

Lorsqu'un responsable de thème se connecte, il peut visualiser toutes les briques qui on été refusé par les relecteurs. Il peut alors les accepter, dans ce cas elles sont envoyées dans la file d'attente de l'indexeur, ou les refuser et dans ce cas il est en charge de prévenir l'auteur de la brique tendancieuse, en lui spécifiant les raisons de ce refus.

Lorsqu'un indexeur se connecte, il visualise les briques en attente de relecture. Il doit alors vérifier les mots-clefs décrivant chaque brique, et si le besoin s'en fait sentir, les modifier, les supprimer ou en ajouter. Les briques indexées sont alors envoyées dans la file d'attente de l'administrateur qui pourra alors les mettre en ligne.

1.4. Les autres fonctions.

D'autres fonctions sont accessibles depuis cette première page, entre autres :

- L'accès aux dernières nouveautés du système permet d'obtenir rapidement les n dernières briques validées par le comité de lecture.
- Les FAQ (*Frenquently Ask Question*) du site.
- Un didacticiel qui permet d'apprendre à utiliser ce serveur.

2. Le modèle objet.

Dans cette partie, nous allons étudier le modèle objet implémenté dans la base de données Matisse (cf. figure 78), car comme nous allons le voir dans le chapitre suivant, nous avons de nouveau utilisé la SGDBDOO Matisse pour stocker les informations persistantes du serveur Semusdi.

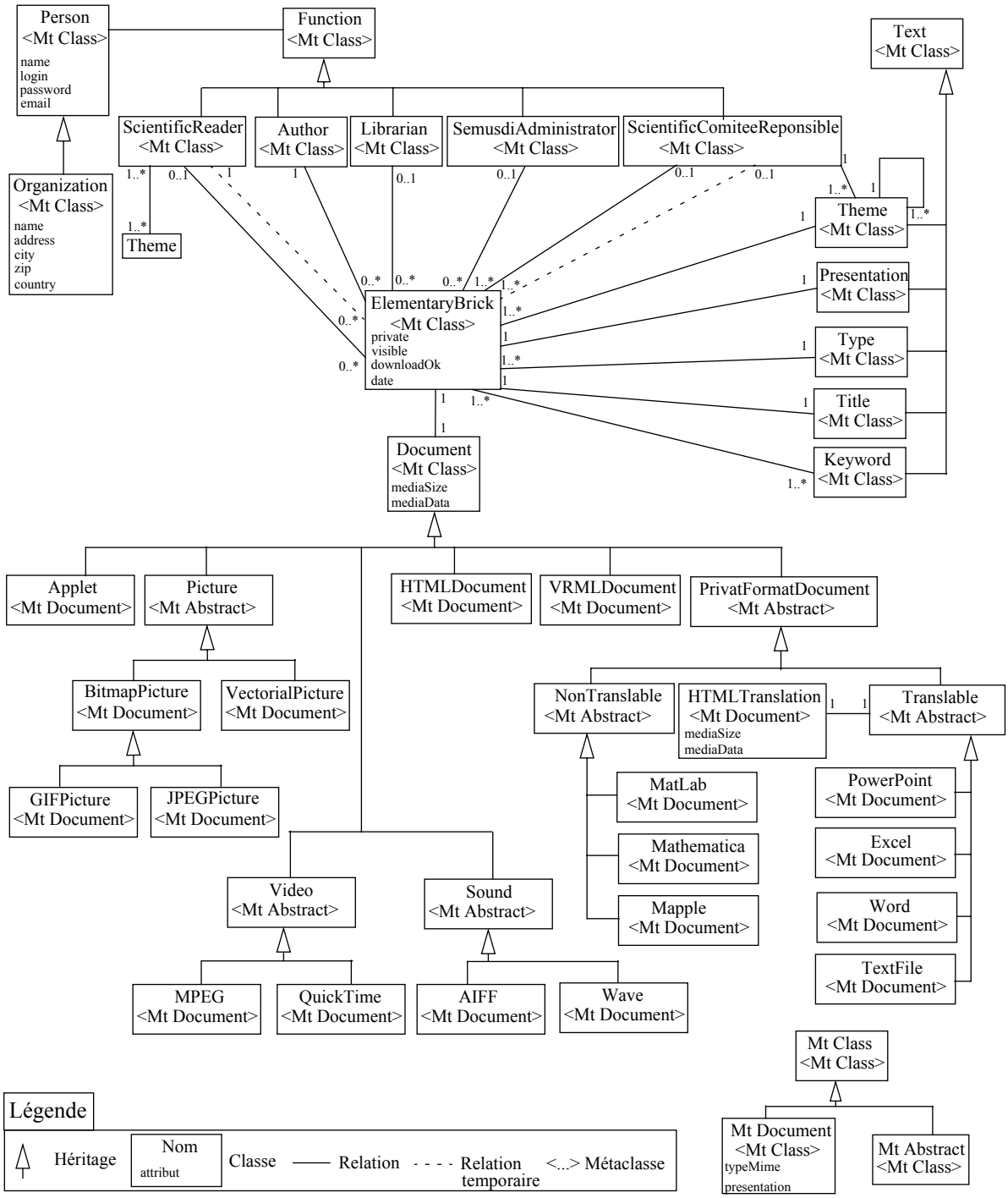


Figure 78 - Modèle objet de SEMUSDI (notation non standard)

Nous pouvons décomposer ce modèle en trois parties distinctes :

- Tout d'abord les classes qui caractérisent une brique élémentaire, instance de la classe *ElementaryBrick*. On retrouve ce que nous avons vu dans le paragraphe précédent, c'est-à-dire le thème de la brique (instance de la classe *Theme*), une présentation succincte (instance de la classe *Presentation*), une rubrique (instance de la classe Type), un titre (instance de la classe *Title*) et des mots-clefs (instances de la classe *Keyword*). Ces classes étant sous-classes de la classe *Text*, pour retrouver une brique via une liste de mots, il suffit d'effectuer une requête au niveau de la classe *Text*.

- Ensuite les sous-classes de la classe *Function* (les classes *Author*, *ScientificReader*, *Librarian*, *SemusdiAdministrator* et *ScientificComitteeResponsible*) sont associées au *Back-office*, c'est-à-dire à l'ajout et la validation de nouvelles briques élémentaires. Chaque personne identifiée par le serveur (instance de la classe *Person*) peut avoir plusieurs "casquettes". Par exemple il peut être auteur (instance de la classe *Author*) et relecteur (instance de la classe *ScientificReader*). Ainsi lorsqu'un auteur ajoute une brique, il y a une relation de créer entre l'objet représentant cette brique, et l'objet de la classe *Author* associé à l'auteur de la brique. Il en est de même après la relecture et la validation de la brique par le relecteur, etc. Le cycle de validation d'une brique est alors le suivant :

  1. Un utilisateur référencé se connecte à Semusdi, et se déclare comme auteur durant cette session. Il ajoute alors une nouvelle brique. Cette brique est créée dans la base mais marquée comme non relu (aucune relation entre la brique et un objet de classe *ScientificReader*).
  2. Un utilisateur référencé se connecte à Semusdi, et se déclare comme relecteur durant cette session. Il a possibilité de visualiser l'ensemble des briques non validées appartenant à son domaine de compétence (mais ne lui appartenant pas en tant qu'auteur). Il peut donc relire la brique (on crée alors une relation temporaire entre lui et la brique car cette relecture peut prendre un certain temps).
     S'il la valide, elle est considérée comme validée. On crée alors une relation entre cette brique et l'administrateur du serveur : instance de la classe *SemusdiAdministrator*, et une relation définitive entre lui et la brique.
     Si il ne la valide pas, on demande alors au responsable de thème de la valider ou pas (en créant une relation temporaire entre le responsable du thème et la brique).
  3. Un utilisateur référencé se connecte à Semusdi, et se déclare comme responsable de thème durant cette session. Il a la possibilité de visualiser les briques non validées par les relecteurs. Il peut choisir de relire une brique. Comme pour le relecteur, on crée une relation temporaire entre cette brique et l'instance de la classe *ScientificComittee* le représentant (cette relation est en fait un verrou).
     S'il la valide, elle est considérée comme validée. On crée alors une relation entre cette brique et l'administrateur du serveur : instance de la classe *SemusdiAdministrator*, et une relation définitive entre lui et la brique.
     Si il ne la valide pas, Il indique à l'auteur, via un courrier électronique, que sa brique n'a pas été acceptée, en précisant les raisons de ce refus.

- Enfin les sous-classes de la classe *Document* permettent de sauvegarder le document multimédia à proprement parler. On retrouve une classe par type de donnée multimédia accepté par le serveur SEMUSDI. Cela permet tout d'abord de pouvoir créer des actions particulières pour chaque type de données. Ensuite, SEMUSDI est capable de déterminer très facilement les types de document multimédia qui sont acceptés par le serveur. En effet, sachant que toutes ces classes sont des instances de la méta-classe *Mt Document*, sous-classe de la méta-classe standard de la base Matisse, il suffit, pour obtenir cette fameuse liste, de demander à la base qu'elle nous fournisse l'ensemble des instances de la classe *Mt Document*.
  De plus l'adjonction de l'attribut *presentation* à la méta-classe *Mt Document* permet de présenter en français les types de données multimédia acceptés. Par exemple pour la classe *JPEGPicture*, on a affecté la chaîne de caractères "Image au format JPEG" à l'attribut *presentation*.
  Dans le même esprit l'attribut *typeMime* permet de fixer le type mime de chaque document multimédia. Par exemple, on a attribué la chaîne de caractères "<image/jpg>" à l'attribut *typeMime* de la classe *JPEGPicture*.

Nous pouvons conclure cette partie en affirmant que l'utilisation d'une base de données orientée

objet est judicieuse, car elle permet d'avoir un modèle souple, non redondant (grâce à l'utilisation des méta-classes), dynamique (toujours grâce à l'utilisation des méta-classes) et ouvert.

Nous allons maintenant élargir notre vision, en nous intéressant à l'architecture client-serveur de Semusdi.

3. L'architecture.

Nous allons voir dans cette partie comment nous avons architecturé ce serveur et comment nous pensons la faire évoluer.

3.1. Architecture des versions 1 et 2.

La construction d'un serveur, qui doit permettre aux usagers d'échanger des données multimédia pose trois problèmes majeurs :

- Comment stocker les données au niveau du serveur, sachant que les données mutimédia sont hétéroclites, aussi bien de par leur format que de par leur taille.
- Comment accélérer les transactions entre le serveur et les clients, et donc la réactivité de l'interface au niveau des clients.
- Comment interfacer des outils de transmission afin que tout utilisateur (informaticien ou non) puisse les utiliser facilement. Dès lors il faut tout intégrer au sein de notre serveur et éviter d'utiliser des outils annexes (logiciels FTP, lecteur de courier électronique, lecteur de forum de discussion, etc.).

Nous avons en partie atteint ces trois objectifs en adoptant l'architecture présentée dans la figure 79 (Cf. [Delestre & al 98a]).

Le stockage des données multimédia est réalisé à l'aide de la base de données Matisse. Sans revenir sur les caractéristiques de ce SGBD, on peut toutefois noter que le modèle objet présenté dans le chapitre précédent a été implémenté tel quel (nous avons utilisé le système de classe et méta-classe).

La grande originalité de cette architecture est l'utilisation conjointe de programmes CGI qui permettent d'attaquer la base de données, et de librairies Javascript. Dans cette optique les programmes CGI ne retournent plus des données formatées, mises en forme (avec le code HTML entier), mais un code HTML épuré, réduit à sa plus simple expression, avec toutefois les particularités suivantes :

- Chargement des librairies Javascript adéquates

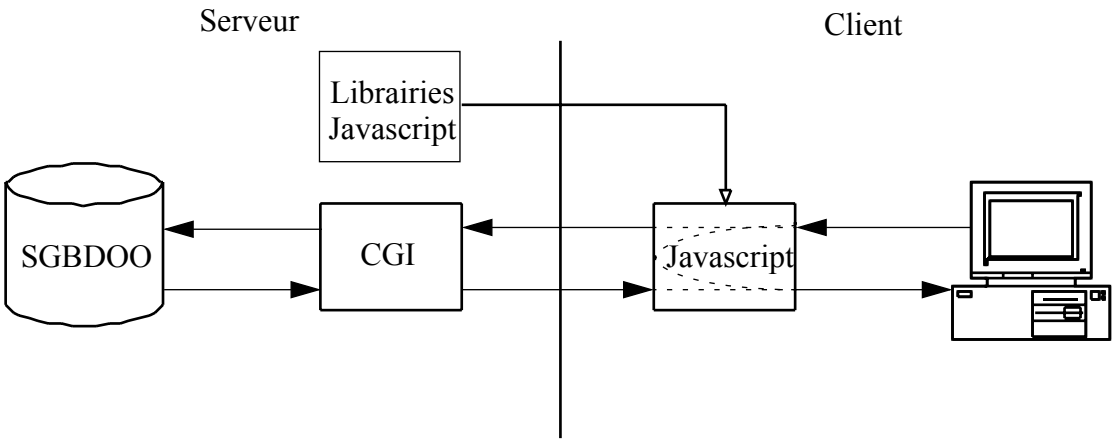


Figure 79 - Architecture actuelle du serveur SEMUSDI

- Insertion de données brutes au sein de tableaux Javascript
- Exécution d'une fonction Javascript particulière au chargement de la page

Ainsi, grâce à ces trois actions, le navigateur a la possibilité de construire l'interface qui va être présentée à l'apprenant. Par exemple, la figure 80 montre une page qui a été entièrement construite chez le client, seules les données textuelles et la brique proprement dite sont transférées chez le client.

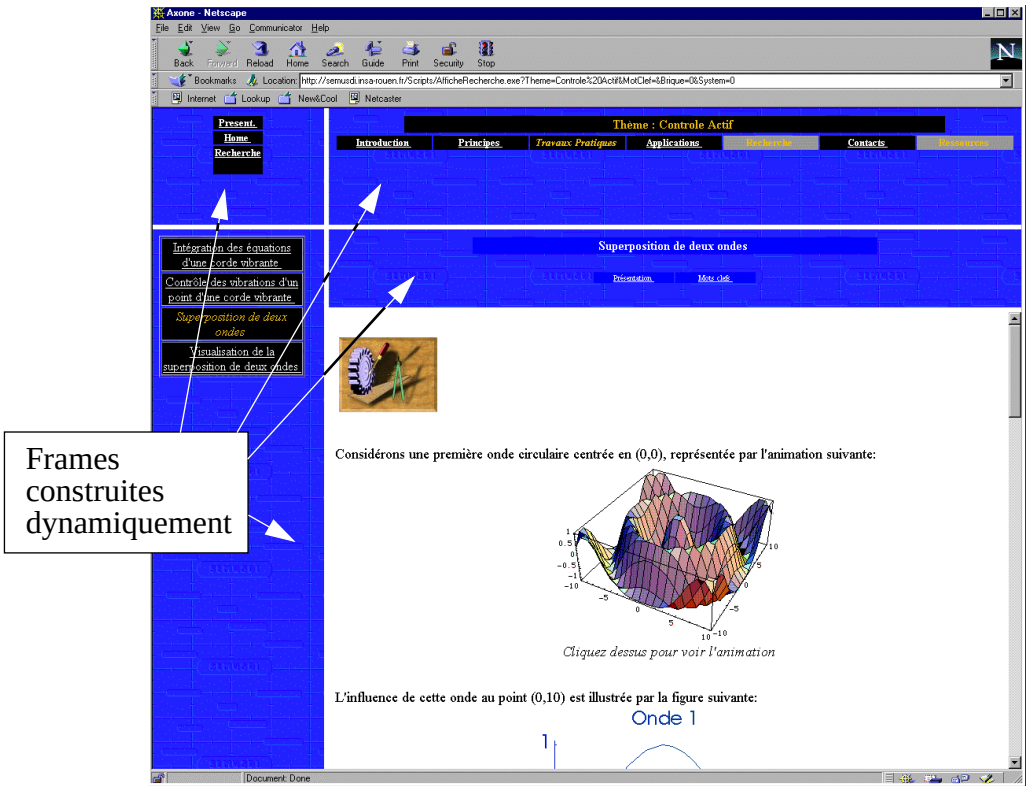


Figure 80 - Construction dynamique de l'interface de SEMUSDI (Version 1)

Ce principe de fonctionnement permet tout d'abord d'améliorer la réactivité au niveau de l'interface. En effet certaines actions sont traitées directement par le navigateur sans utiliser le serveur (par exemple l'obtention de la liste des mots clés de la brique courante). De plus le nombre de données transférées du serveur vers le client est réduit, puisque seules les données brutes et la brique à visualiser sont tranférés (les librairies Javascript une fois transférées restent dans le cache du navigateur). Ensuite cela permet de dissocier le fond de la forme, et donc de pouvoir modifier assez simplement cette dernière si le besoin s'en fait sentir, il n'y a en aucun cas besoin de modifier les programmes CGI. Ainsi la version actuelle (version 2) du serveur, dont l'habillage est totalement différent, a été obtenu principalement en réécrivant les librairies Javascript.

Ainsi la figure 81 montre l'évolution de l'interface, lorsque l'on visualise la même brique.

Toutefois, cette architecture n'est pas exempte de tout défaut. Tout d'abord les programmes CGI ont été écrits en C, langage non objet et de par nature peu évolutif. Sachant que l'on utilise une base de données objets, il est dommage de perdre cet avantage (pour notre défense, en 1996, les API Java pour Matisse n'étaient pas disponible). Ensuite, les programmes sont liés à la base de données Matisse. Dans l'état actuel, il est impossible de changer de base de données sans reprendre tous les programmes. Enfin, le langage Javascript n'est pas standard. L'interprétation du code dépend de la machine utilisée (compatible PC, Mac, etc.), de l'éditeur du navigateur (Netscape et Microsoft) et de la version du navigateur. Pour résoudre ce problème, les librairies Javascript prennent en considération ces trois paramètres pour exécuter tel ou tel code.

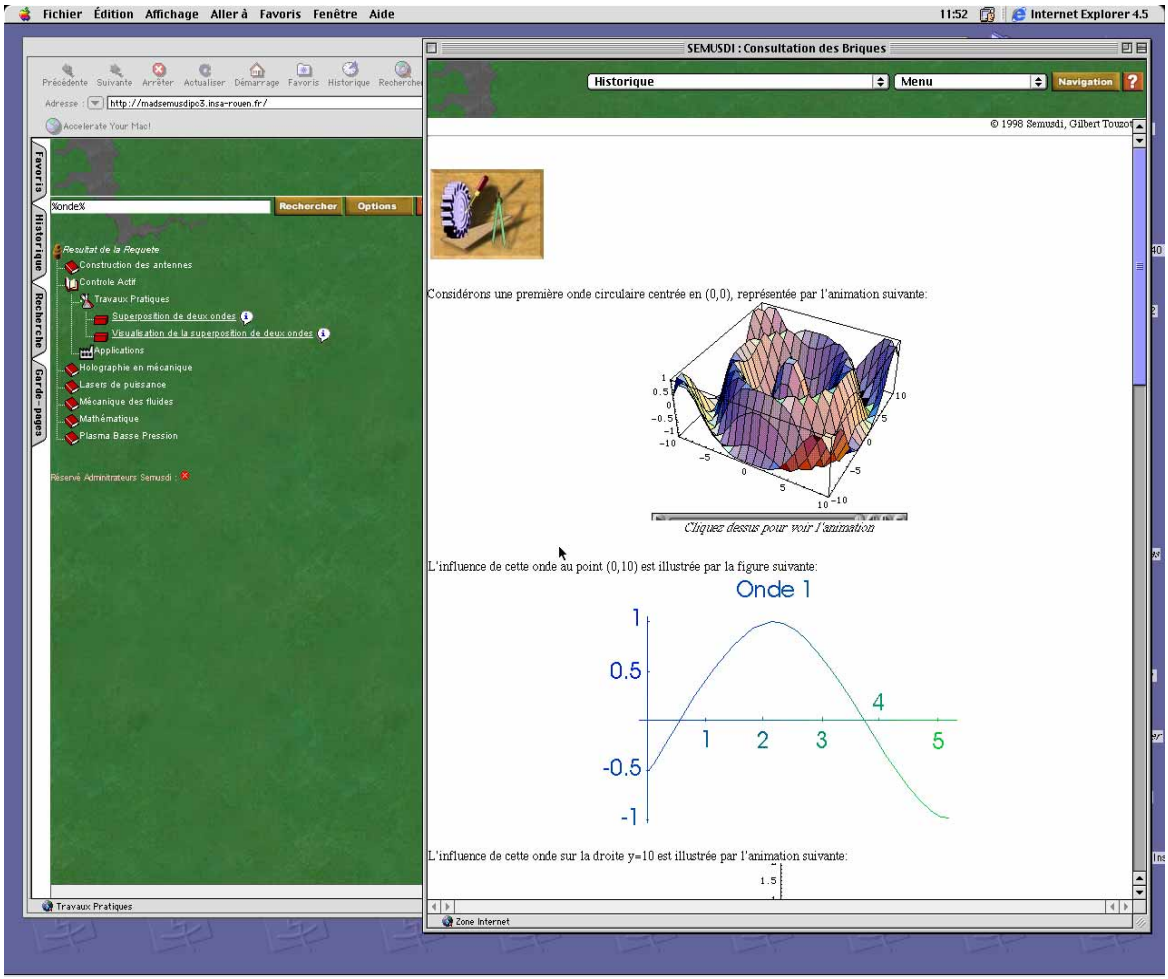


Figure 81 - Interface de la version 2

3.2. Architecture de la future version 3.

Au moment où je rédige ce mémoire, la modélisation de la version 3 du serveur SEMUSDI est terminée et le développement débute. Cette nouvelle architecture tente d'annihiler les défauts que nous venons de voir en renforçant la distinction entre les différents services :

Analysons cette nouvelle architecture.

Il y a tout d'abord un serveur HTTP, qui possède un ensemble de *servlets*. Ces *servlets* vont peu à peu remplacer les CGI de la version 2 (ainsi la transition entre les deux versions va se faire en douceur). Ce *servlets* vont émettre des requêtes au niveau du serveur de service SEMUSDI (le SSS). Ces requêtes sont du type donne moi telle brique, recherche l'ensemble des briques spécifier par la requête suivante, etc. Dès lors le SSS émet des requêtes au serveur de services de base de données (le SSBD). Ce dernier effectue alors ces actions sur la base de données sélectionnée en se basant toutefois sur le modèle présenté au chapitre précédent.

Les avantages d'une telle architecture sont :

- L'interface standard du client est toujours construite dynamiquement (comme dans la version 2).
- On peut ajouter des *applets* (entre autre pour la gestion du *back-office*) qui seront directement reliées au SSS.
- Nous avons une chaîne au niveau du serveur totalement objet (jusqu'au niveau de la base de données si on choisit une base objet) écrite en Java.

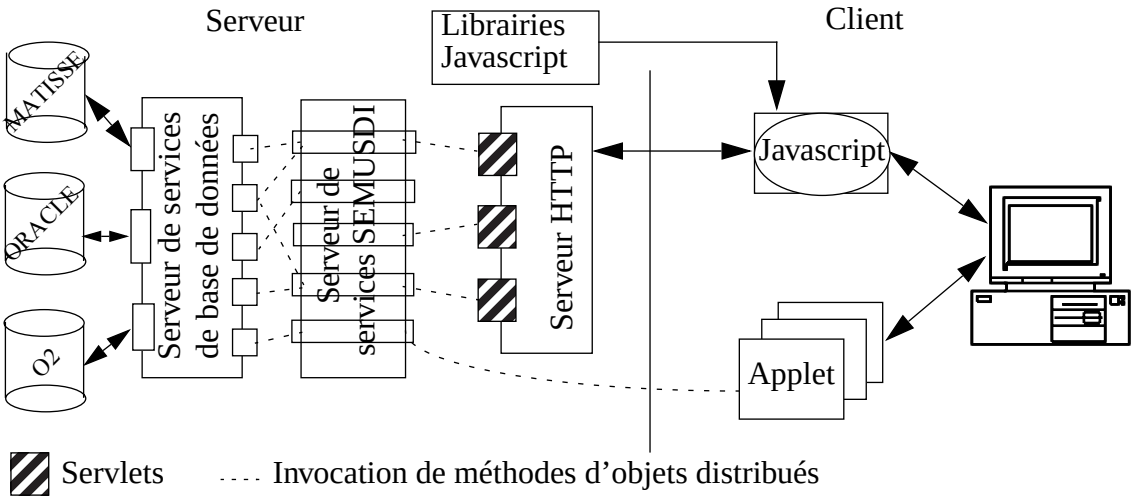


Figure 82 - Architecture de la version 3 de SEMUSDI

- La séparation en service permet de répartir les différentes actions sur différentes machines, permettant par la suite d'avoir pourquoi pas plusieurs serveur HTTP Semusdi, un serveur de service central et des serveurs de bases répartis géographiquement.

Si dans le futur, la bande passante du réseau Internet augmente convenablement, l'interface utilisateur pourra reposer totalement sur des *applets*.

4. Conclusion.

Cette annexe a présenté la mise en oeuvre d'un site Internet permettant à différentes personnes (étudiants, enseignants, et ingénieurs en activité) d'échanger des briques élémentaires sur le domaine des sciences de l'ingénieur. L'intérêt majeur de ce type de serveur est simple : il permet à des non-informaticiens de trouver facilement des documents didactiques multimédia, sans passer des heures à les rechercher sur Internet ou à les construire à l'aide d'outils souvent complexes.

Afin d'obtenir un produit performant et facile à manipuler, une architecture client-serveur innovante (utilisation conjointe de programmes CGI et librairies Javascript) et une base de données orientée objet ont été utilisées.

A l'heure actuelle, la version 2 vient d'être mise en ligne (l'ergonomie et le moteur de recherche ont été améliorés) et la version 3 avec une architecture encore plus innovante est à l'étude.

Références bibliographiques.

[Abou Khaled & al 98] O. Abou Khaled, M.C. Pettenati, C. Vanoirbeek, G. Coray, "MEDIT : a Distance Education Prototype for Teaching and Learning", Actes de WebNet'98, Orlando (Etats-Unis), p7-12, 1998.

[Avgoustos & al 99] A.T. Avgoustos, G.M. Kostantinos, "Student Modelling Review-What can be learned for Distance Education ?", WebNet'99, Hawai, Etas-Unis, p277-282, 1999.

[Aziz & al 99a] W. Aziz, S. Barthélémy, N. Delestre, "How to share multimedia data about engineering sciences ?", CAEE'99, Sofia, Bulgarie, p72-78, 1999.

[Balasubramanian 94] V. Balasubramanian, "State of the Art Review on Hypermedia Issues And Applications", Graduate School of Management, Rutgers University, Newark, NJ, 1994.

[Baron 94] M. Baron, "EIAO : quelques repères.", Revue Terminal N°65, 67-84, 1994.

[Bodner & al 97] R. Bodner, M. Chignell, J. Tam, "Website Authoring using Dynamic Hypertext", Proceeding of WebNet'97, Toronto, p59-64, 1997.

[Bordeleau 99] P. Bordeleau, "L'histoire des technologies informatiques et quelques-unes de leurs applications en éducation", http://www.scedu.umontreal.ca/histoiredestec/, 1999.

[Bouzeghoub & al 97] M. Bouzeghoub, G. Gardarin, P. Valduriez, "Les objets", Edition Eyrolles, 1997.

[Boyle & al 98] C. Boyle, A. O. Encarnacion, "Metadoc : An Adaptive Hypertexte Reading System", Adaptive Hypertext and Hypermedia, Kluwer Academic Publishers, p71-89, 1998.

[Bramand & al 83] P. Bramand, P. Faye, G. Thomassier, "Physique, Terminal C et E", Collection Eurin-gié, Hachette, 1983.

[Brown & al 78] J.S. Brown, R. R. Burton, “Diagnostic models for procedural bugs” dans basic mathematical skills, Cognitive Science, Vol. 2, p155-192, 1978.

[Bruillard & al 87] E. Bruillard, I. Péreira, “Quelques problèmes d’apprentissage avec Logo et Prolog”, Actes du Congrès Francophone sur l’Enseignement Assisté par Ordinateur, pp267-276, 1987.

[Brusilovsky 92] P. Brusilovsky, “Intelligent Tutor, Environment and Manual for Introductory Programming”, Educational and Training Technology International, 29(1), p26-34, 1992.

[Brusilovsky 94a] P. Brusilovsky, “Adaptative Hypermedia : An attempt to Analyze and Generalize”, Multimedia, Hypermedia and Virtual Reality, First International Conference, MHVR’94, p288-304, 1994.

[Brusilovsky & al 94b] P. Brusilovsky, L. Pesin, “ISIS-Tutor : An Intelligent Learning Environment for CDS/ISIS Users”, CLCE’94, 1994.

[Brusilovsky & al 95] P. Brusilovsky, L. Pesin, “Visual annotation of links in adaptative hypermedia”, CHI’95, Denver, Etats-Unis, 1995.

[Brusilovsky 98] P. Brusilovsky, “Methods and techniques of Adaptive Hypermedia”, Adaptive Hypertexte and Hypermedia, Kluwer Academic Publishers, p1-43,1998.

[Bush 45] V. Bush, “As We May Think”, The atlantic monthly, 1945.

[Carbonell 70] J.R. Carbonell, “AI in CAI: An Artificial Intelligence Approach to Computer Assisted Instruction”, IEEE Transactions on Man-Machine Systems, Vol. 11, N. 4, p190-202, 1970.

[Carbonneaux & al 97] Y. Carbonneaux, J-M. Laborde, M.R. Madani, “Une interface sur les graphes dans CABRI-Graphs”, EIAO’97, Edition Hermes, p302-303, 1997.

[Carro & al 99] R.M. Carro, E. Pulido, P. Rodriguez, “An Adptive Driving Course Based on HTML Dynamic Generation”, WebNet’99, Hawaï, Etats-Unis, p171-176, 1999.

[Carver & al 96] C. A. Carver, C. Ray, “Automating Hypermedia Course Creation and Maintenance”, WebNet’96, San Francisco, Etats-Unis, p131-136, 1996.

[Charlot 98] J-M. Charlot, “Formalisation et comparaison cognitives de modèles mentaux de novices et d'experts en situation de résolution de problèmes”, Thèse de l’université de Sherbrooke, Québec, Canada, http://www.gme.usherb.ca/~charlot/These/, 1998.

[Clancey 79] W.J. Clancey, “Tutoring rules for guiding a case method dialogue.”, Internationnal Journal of Man-Machine Studies, N. 11, p25-49, 1979.

[Codd 70] E.F. Codd, “A Relational Model for Large Shared Data Banks”, Communications de l’ACM, Vol. 13, N°6, Octobre 1970.

[Cox & al 99] R. Cox, M. O’Donnell, J. Oberlander, “Dynamic versus static hypermedia in museum education: an evaluation of ILEX, the intelligent labelling explorer”, Artificial Intelligence in Education - AIED’99, p181-188, 1999.

[Delestre 96] N. Delestre, “Analyse des modèles objets de New Flavors et CLOS : Stratégies de traduction”, Mémoire de DEA, INSA de Rouen, 1996.

[Delestre & al 97a] N. Delestre, J-P. Pécuchet, Didier Viard, "Etude ergonomique de l'interfaçage de l'AGL DOM avec le SGBDO Matisse", rapport pour le projet Européen Eureka-Matisse, 1997.

[Delestre & al 97b] N. Delestre, C. Gréboval, J-P. Pécuchet, "METADYNE, a Dynamic Adaptive Hypermedia System for Teaching", 3rd ERCIM Workshop, Obernai, France, p143-149, 1997.

[Delestre & al 98a] N. Delestre, B. Rumpler, "Architecture d'un Serveur Multimédia pour les Sciences de l'Ingénieur", NTICF'98, Rouen, France, p39-46, 1998.

[Delestre & al 98b] N. Delestre, J-P. Pécuchet, C. Gréboval, "Architecture d'un hypermédia adaptatif dynamique pour l'enseignement", NTICF'98, Rouen, France, p383-391, 1998.

[Delestre & al 99a] N. Delestre, J-P. Pécuchet, C. Gréboval, "How to design a dynamic adaptive hypermedia for teaching", AIED'99, Le Mans, France, p654-656,1999.

[Delestre & al 99b] N. Delestre, J-P. Pécuchet, C. Gréboval, "Why to use a dynamic adaptive hypermedia for teaching, and how to design it", WebNet'99, Hawai, Etats-Unis, p277-282, 1999.

[Demphlous 98] S.Demphlous, "Gestion de la persistance au sein de systèmes réflexifs à objets", Thèse de l'université de Nice-Sophia Antipolis, 1998.

[Duffy 91] T. Duffy, R. Knuth, "Hypermedia and Instruction : Where is the match", dans Designing Hypermedia for Learning, D. Jonassen, H. Mandl (Eds.), Springer-Verlag, 1991.

[Englebart 62] D. Englebart, "Augmenting human intellect: A conceptual framework", Summary Report Contract, SRI Project, Stanford Research Institute Menlo Park, http://www.histech.rwth-aachen.de/www/quellen/engelbart/ahi62index.html, 1962.

[Frasson & al 96] C. Frasson, E. Aimeur, "SAFARI, a University-Industry Cooperative Project", International Conference on Successand Pitfall of Knowledge-Based Systems in Real-World Application, Bangkok, 1996.

[Gaillard 94] F. Gaillard, "La modélisation des connaissances et l'utilisation de base de données objet en productique", Thèse de l'Université Technologique de Compiègne, 1994.

[Geib & al 98] J-M. Geib, C. Gransart, P. Merle, "CORBA : des concepts à la pratique", 1998.

[Giroux & al 97] S. Giroux, S. Leman, P. Marcenac, "Representing Organisational Student Models : A Generic Concurrent Coordination", Workshop on Architectures and Methods for Designing Cost-Effective and Reusable ITSs, ITS'96, Montréal, Canada, 1996.

[Hathaway & al 97] J. Hathaway, S. Kirshbaum, "The Reusable Content Object Strategy", White Paper v1.6, Oracle, 1997.

[Hirano 97] S. Hirano, "HORB : Distributed Execution of Java Programs", Worldwide computing and its applications, springer lecture notes in computer science pp29-42, 1997.

[Hirano & al 98] S. Hirano, Y. Yasu, H. Igarashi, "Performance Evaluation of Popular Distributed Object Technologies for Java", Workshop on Java for High-Performance Network Computing, 1998.

[Hoogeveen 95] M. Hoogeveen, “Toward a New Multimedia Paradigm : is Multimedia Assisted Instruction Really Effective ?”, ED-MEDIA 95 World Conference on Educational Multimedia and Hypermedia, p348-353,1995.

[Hoogeveen 97] M. Hoogeveen, “Towards a Theory of the Effectivness of Multimédia Systems”, International Journal of Human Computer Interaction, 9(2), p151-168, 1997.

[Kavcic 98] A. Kavcic, “A Model of An Adaptive Hypermedia Educational System”, NTICF’98, Rouen, France, p345-350, 1998.

[Langue 90] D. Langue, “A formal model of hypertext”, Hypertext standardization Workshop, Gaithersburg, 1990.

[Langue 96] D. Langue , “An object-oriented design approach for Developing Hypermedia Information Systems”, Journal of Organizational Computing and Electronic Commerce 6(3), p269-293, 1996.

[Lapujade & al 96] A. Lapujade, C. Ernst, “Un panorama des systèmes d’aide à la conception de logiciels éducatifs”, Revue Sciences et Techniques Educatives, Volume 3, n°3, p297-334, 1996.

[Laroussi & al 98] M. Laroussi, M. Ben Ahmed, “CAMELEON : Computer Aided MEdium for LEarning On Networks,” ED-MEDIA & ED-TELECOM, Feibourg, Allemagne, 1998

[Lelouche 87] R. Lelouche, “Apports de l’EIAO à l’EAO”, Actes du Congrès Francophone sur l’Enseignement Assisté par Ordinateur, p173-181, 1987.

[Leman 96] S. Leman, S. Giroux, P. Marcenac, “Auto-similarité d'un modèle d'agent. Application à la modélisation d'un utilisateur.”, Journées Françaises IAD & SMA, 1996.

[Luengo 97] V. Luengo, “Un micro-monde de preuve intégrant la réfutation : Cabri-Euclide”, EIAO’97, Edition Hermes, p85-97, 1997.

[Mallet & al 99] F. Mallet, G. Mallet, “Some examples of diophanentos functionalities”, CAEE’99, Sofia, Bulgarie, p238-243, 1999.

[Marquesuzaà 98] C. Marquesuzaà, “OMAGE : Outils et méthode pour la spécification des connaissances au sein d’un atelier de génier éducatif”, Thèse de l’Université de Pau et des pays de l’Adour, 1998.

[Maurer & al 93] H. Maurer, N. Scherbakov, P. Srinivasan, “A new hypermedia data model”, Database and Expert Systems Applications - DEXA'93, p685-696, Prague, 1993.

[Maurer & al 94] H. Maurer, N. Scherbakov, K. Andrews, P. Srinivasan, “Object-Oriented Modeling of Hyperstructure: Overcoming the static link deficiency”, Information and Software Technology, Vol.36 (6) p315 - 322, 1994.

[Mendelsohn 91] P. Mendelsohn, “Les environnements intelligents d’apprentissage”, Cahiers de la Fondation Archives Jean Piaget, Université de Genève, p75-86, 1991.

[Mendelsohn & al 93] P. Mendelsohn, P. Dillenbourg, “Le développement de l’enseignement intelligement assisté par ordinateur”, dans J.F. Le Ny (Ed), intelligence naturelle et intelligence artificielle. Paris: PUF, 1993.

[Menselsohn 95] P. Mendelsohn, "Peut-on opposer Savoirs et Savoir-faire quand on parle d'apprentissage?" , dans A. Bentolila (Ed) - Savoirs et savoir-faire. Les Entretiens Nathan (Actes V). Paris: Nathan, 1995.

[Miller 56] G. A. Miller, "The magical number seven plus or minus two: some limits on our capacity for processing information", Psychological Review, p81-97, 1956.

[Moulin 98] C. Moulin, "Adaptation dynamique d'un système d'aide à l'apprentissage de la géométrie : Modélisation par un système multiagent", Thèse de l'Université de Rouen, 1998.

[Murray & al 96] W.G. Murray, S. Salari, P. Swoboda, "World Wide Web Course Tool: An Environment for Building WWW-Based Courses", Actes de Fifty International World Wide Web Conference, Paris (France), 1996.

[Murray & al 97] W.G. Murray, S. Salari, "An update on WebCT (World Wide Web Course Tools, a Tool for the Creation of Sophisticated Web Based Learning Environments", Acted de NAUWeb'97, Flagstaff (Etas-Unis), 1997.

[Nadeau 97] F. Nadeau, "Applications et impacts de l'hypermédia constructif sur l'apprentissage", http://www.fse.ulaval.ca/fac/ten/64448/nado/semi.html, 1997.

[Nanard 95] M. Nanard, "Les hypertextes : au-delà des liens, la connaissance", Sciences et techniques éducatives, Vol 2, n°1, Edition Hermes,1995.

[Necaille 98] C. Nécaille, "ARCHYMEDIA, architecture d'un système hypermédia d'aide à la formation", Thèse de l'université de Rouen, 1998.

[Nelson 65] T. Nelson, "A File Structure for the Complex, The Changing and The Indeterminate", ACM 20th National Conference, 1965.

[Nemetz & al 98] F. Nemetz, P. Johnson, "Developing Multimedia Principles from Design Features", dans A. Sutcliffe, J. Ziegler & P. Johnson (eds.), Designing Effective and Usable Multimedia Systems, Kluwer Academic Publishers, pp.57-71, 1998.

[Nicaud & al 99] J-F. Nicaud, D. Bouhineau, C. Varlet, A. Nguyen-Xuan, "Towards a product for teaching formal algebra", Actes d'AIED'99, (Artificial Intelligence in Education), Le Mans (France), p207-214, 1999.

[Nicolas & al 97] C. Nicolas, C. Avare, F. Najman, "Java, Client-Serveur", Collection Fi System, Eyrolles, 1997.

[Nkambou 96] R. Nkambou, "Modélisation des connaissances de la matière dans un système tutoriel intelligent: modèle, outils et applications", Thèse de la Faculté des études supérieures de l'Université de Montréal, 1996.

[O2] "O2 Database system technical overview", société Ardent.

[Pilgrim & al 99] C.J. Pilgrim, Y.K. Leung, "Site Maps - Where are we now ?", WebNet'99, Hawaï, Etats-Unis, p883-888, 1999.

[Quillian 68] M.R. Quillian, "Semantic Memory", Semantic Information Processing, MINSKY M. ed., Cambridge, Mass. M.I.T. Press, p227-70, 1968.

[Poulain 98] L. Poulain, "Les objets JAVA Persistant", Journal DATABASES, the european technical reference, N°15, 1998.

[Recker 95] M. Recker, “Cognitive Media Types for Multimedia Information Access”, Journal of Education Multimedia and Hypermedia, 1995

[Rhéaume 93] J. Rhéaume, “Les hypertextes et les hypermédia”, Revue EducaTechnologie, Volume 1, numéro 2, Décembre 1993.

[Rekik & al 99] Y.A. Rekik, C. Vanoirbeek, “Course designer Environment : Authoring Environment for nNteractive Courses with a Shared Course Components Library”, CAEE’99, Sofia, p296-303, 1999.

[Self 88] J. Self, “By passing the intractable problem of student modeling”, Proceedings d’ITS 88, Montréal, p18-24, 1988.

[Soutou 99] C. Soutoun, “Objet-relationnel sous Oracle 8”, Edition Eyrolles, 1999.

[Sowa 84] J.F. Sowa, “Conceptual Structures Information processing in mind and machine”, Addison-Wesley, 1984.

[Steel 90] G.L. Steel JR, “Common LISP, The Language, Second Edition”, Digital, 1990.

[Vassileva 95] J. Vassileva, “Dynamic Courseware Generation : at the Cross of CAL, ITS and Autoring”, the International Conference on Computers in Education, ICCE’95, Singapore, p290-297, 1995.

[Vassileva 97] J. Vassileva, “Dynamic Courseware Generation on the WWW. Proceedings of the workshop : Adaptive Systems and User Modeling on the World Wide Web”, Sixth International Conference on User Modeling, 1997.

[Vigne 99] A. Vigne, “Applications temps réel réparties pour Internet”, Mémoire CNAM de l’INSA de Rouen, 1999.

[Vogel & al 98] Andreas Vogel and Keith Duddy, “Java programming with CORBA (2nd Edition)”, Wiley Computer Publishing, 1998

[Wenger 87] E. Venger, “Artificial Intelligence and Tutoring Systems, Computational and Cognitive Approaches to the Communication of Knowledge”, Morgan Kaufman Publisher, 1987.

Glossaire.

API — *Application Program Interface.*
Ensemble de conventions définissant de quelle manière un service peut être joint et utilisé par un logiciel

Applet — Programme écrit en Java qui se trouve sur un serveur Web et qui s'exécute au sein d'un navigateur Web.

Blob — *Binary Large Object.*
Type de base de données permettant de stocker une grande quantité d'informations non formatées (considéré comme une suite d'octets).

Brique élémentaire — Document multimédia didactique très ciblé (terme retenu dans le projet SEMUSDI).

CGI — *Common Gateway Interface.*
Programme qui s'exécute sur un serveur Web, qui permet d'effectuer des traitements suite à la requête d'un navigateur.

COM — *Component Object Model.*
Composants logiciels réutilisables spécifiques à Microsoft et à son environnement Windows.

CORBA — *Common Object Request Broker Architecture.*
Spécification proposée par l'OMG permettant l'interopération entre objets clients et objets serveurs distribués.

CSS — *Cascading Style Sheets.*
Ensemble de règles définissant des styles de mise en page (police de caractères, alignement, couleur, etc.)

DCOM — *Distributed Component Object Model*.
Sur-couche de COM permettant d'utiliser des composants situés sur des machines distantes (Mécanisme ancré au coeur du système Windows NT).

DTD	Déclaration du Type de Document. Ensemble de règles qui définit et qui gouverne un document XML.
EAO	Enseignement Assisté par Ordinateur. Domaine de recherche en informatique dont l'objectif est de produire des systèmes d'enseignement.
EIAO	Enseignement Intelligement Assisté par Ordinateur. Domaine de recherche en informatique, sous domaine de l'EAO, dont l'objectif est de produire des logiciels éducatifs intelligents. Cette intelligence peut-être mesurée par son adaptation vis à vis de l'utilisateur.
EIAO	Enseignement Interactif Assisté par Ordinateur. Domaine de recherche en informatique, sous domaine de l'EAO, dont l'objectif est de produire des logiciels qui permettent d'enseigner des notions à l'utilisateur grâce à l'interaction entre ces deux acteurs.
EIAO	Environnement Interactif d'Apprentissage par Ordinateur. Catégorie de logiciels issue de la recherche dans les domaines de recherche des deux EIAO (avec I qui signie Intelligent et Interactif). Ce sigle correspond au sigle anglophone ILE, pour *Interactive Learning Environment*.
HORB	*Hirano Object Request Broker.* Ensemble d'outils développés à l'origine par Hirano Satoshi permettant de distribuer des objets Java (http://openlab.etl.go.jp/horb/).
HTML	*HyperText Markup Language.* Langage permettant de construire des documents visualisables à l'aide de navigateur Web.
HTTP	*HyperText Transfert Protocol*. Protocole de transfert de fichiers HTML à travers un réseau TCP/IP. Un serveur HTTP gére les requêtes / reponses entre client et serveur.
Hyper-espace	Graphe formé par les pages et liens d'un hypertexte.
Hypermédia	Document électronique multimédia, composé de pages et de liens entre ces pages, permettant à l'utilisateur d'avoir une lecture non linéaire de l'information.
Hypermédia adaptatif	Hypermédia assez "intelligent" pour pouvoir modifier ce qu'il présente à l'utilisateur en tenant compte des caractéristiques de ce dernier.
Hypertexte	Document électronique textuel, composé de pages et de liens entre ces pages, permettant à l'utilisateur d'avoir une lecture non linéaire de l'information.
IDL	*Interface Definition Language.* Langage de définition d'interface permettant de spécifier des opérations de certains objets (proposer comme un standard par l'OMG).
IHM	Interface Homme-Machine. Domaine de recherche s'interessant à la problématique de la communication entre l'homme et la machine.
Javascript	Langage intialement proposé par Netscape qui permet d'inclure des fonctions au sein de pages HTML qui sont interprétées par le Navigateur.
JDBC	*Java DataBase Connectivity.*

	Spécification comparable à ODBC mais connue pour interroger des bases de données depuis un programme Java
JDK	*Java Development Kit.* Ensemble d'outils fournis principalement par la société Sun, permettant de compiler et d'exécuter des programmes Java.
ODBC	*Open DataBase Connectivity.* Interface d'accès aux bases de données promue par Microsoft.
ODL	*Object Definition Language.* Langage de définition de déclaration de schémas (extension de l'IDL de l'OMG).
ODMG	*Object Database Management Group.* Groupe de constructeurs de bases de données objets proposant une recommandation en matière de langage de définition, d'interrogation et de manipulation de base de données objets (http://www.odmg.org/).
OMG	*Object Management Group.* Organisation qui définit des standards en matière d'orientation objet (http://www.omg.org).
OMT	*Object Modelling Technique.* Méthode d'analyse et de conception objet.
OQL	*Object Query Language.* Langage d'interrogation de bases de données objets basé sur une extension de SQL proposé par l'ODMG.
ORB	*Object Request Broker.* Sous-système distribué permettant l'envoi de message depuis un objet client vers un objet serveur distant et permettant la récupération des réponses.
QCME	Questionnaire à choix multiples étendus.
RMI	*Remote Method Invocation.* Ensemble d'outils et de classes fournis par Sun permettant de distribuer des objets Java.
RPC	*Remote Procedure Call.* Mécanisme permettant à une application d'appeler une procédure distante et de récuprer le résultat de cette dernière.
SEMUSDI	SErveur MULtimédia pour les Sciences de l'Ingénieur. Serveur Web mis en place à l'INSA de Rouen permettant à une communauté (étudiants, enseignants, chercheurs et ingénieurs en activités) de partager des briques multimédia didactiques dans le domaine des sciences de l'ingénieur (http://semusdi.insa-rouen.fr).
Servlet	Programme écrit en Java qui s'exécute sur un serveur Web, qui permet d'effectuer des traitements suite à la requête d'un navigateur.
SGBD	Système de Gestion de Bases de Données. Ensemble de programmes permettant la création et l'utilisation de bases de données.
SGBDO	SGBD Objets. SGBD supportant les concepts d'objet et de classe, d'héritage et de polymorphisme.

SGBDOO SGBD Orientés-Objets.
Extension d'un modèle à objets pour la définition et la manipulation d'objets persistants.

SGBDR SGBD Relationnelles.
SGBD qui repose sur le modèle relationel (lui même repose sur la théorie des ensembles).

SGBDRO SGBDR Relationnelles-Objets.
Extension d'un SGBD relationnelles et de son langage d'interface avec les concepts objets.

SQL *Simple Query Language.*
Langage de requête pour les SGBD Relationnelles.

STI Système Tutoriel Intelligent.
Système d'enseignement dont l'objectif pédagogique est de transmettre un savoir mais surtout un savoir-faire. Il est généralement constitué de quatre modules : modèle du domaine, modèle de l'apprenant, module pédagogique et module d'interface.

UML *Unified Modeling Language.*
Langage graphique d'analyse, de modélisation et de conception objet (successeur d'OMT).

URL *Uniform Ressource Locator.*
Adresse, en caractères alphanumériques, déterminant le chemin d'accès d'un document disponible sur l'Internet (ou plus généralement d'un site, d'un objet ou d'un service).

WWW *World Wide Web.*
Sous-ensemble d'Internet qui regroupe des milliers de sites Web reliés entre eux par des liens hypertextuels

WYSIWYG *What You See Is What You Get.*
Acronyme qui signifie que ce que l'on voit à l'écran est identique à ce qui va être imprimé ou utilisé.

XML *eXtend Markup Language.*
Langage extensible de balisage de documents, élaboré par le groupe de travail ERB (*Editorial Review Board*) du W3C (*World Wide Web Consortium*); version simplifiée de SGML destinée aux applications Internet.

XSL *eXtend Stylesheet Language*.
Language qui définie à l'aide des CSS un modèle de représentation visuelle des données d'un fichier XML.

METADYNE, un Hypermédia Adaptatif Dynamique pour l'Enseignement

Résumé

L'enseignement assisté par ordinateur (EAO) est depuis plus de vingt ans un domaine de recherche pluridisciplinaire des plus actifs. La problématique principale de ces recherches a été la production de systèmes d'enseignement axés sur la transmission du savoir-faire: systèmes de plus en plus intelligents avec de fortes capacités pédagogiques. Mais ces recherches se sont totalement désintéressées de la problématique liée à une bonne transmission du savoir. Or, l'apparition des Nouvelles Technologies de l'Information et de la Communication a révolutionné cette façon de penser, car elles sont propices à la diffusion du savoir à grande échelle. Des systèmes de construction de cours hypermédia ont donc été développés, dont la plupart étaient, et sont, basés sur le concept de réutilisation d'items didactiques. Cependant, ces derniers incitent l'enseignant à ne suivre aucune méthodologie pour construire leurs cours (on leur propose seulement d'agencer les items didactiques), et les cours ainsi créés ne profitent pas des avancées issues des recherches dans les domaines de l'EAO et des hypermédia adaptatifs.
Ceci définit donc notre problématique de recherche : Définir et prototyper un système qui permette à des enseignants de construire des cours hypermédia qui pourront par la suite s'adapter aux caractéristiques et volontés de chaque apprenant. A travers une démarche mixte, modélisation des connaissances et modélisation objet, nous avons déterminé un modèle conceptuel, c'est-à-dire l'ensemble des connaissances qui sont du ressort des enseignants, des apprenants ainsi que celles qui sont intrinsèques aux items didactiques. Ceci nous a permis de définir un modèle objet, qui finalement a donné naissance à un prototype basé sur une architecture clients-serveurs, composé d'une base de données objets, d'un serveur d'objets distribués et enfin d'un serveur Web.

Mots-clés :

Enseignement assisté par ordinateur | Hypermédia adaptatif | Ingénierie de connaissances
Génie logiciel | Architecture client-serveur

METADYNE, a Dynamic Adaptive Hypermedia System for Teaching

Abstract

For more than twenty years, the computer aided teaching (CAT) has been a very active research field. The major problem of this research was the production of educational systems centred on the transmission of the know-how. But this research did not take into account the problems related to a good learning transmission. However, the New Technologies of Information and Communication revolutionised this way of thinking, since they are favourable to the diffusion of the learning on a large scale. Hypermedia courses building systems have been developed, and most of them were, and still are based on the concept of didactic item re-use. However, they encourage teachers to follow no methodology to build their courses (they only offer to arrange didactic items for teachers), and achieved courses do not use solutions that were found in the research fields of CAT or adaptive hypermedia.
Then, our research problem consists in defining and to prototype a system that helps teachers to build methodically hypermedia courses that will ready to be adapted to the characteristics and the wills of the student. We have followed a mixed approach to design the systems, knowledge design and object design. First we have determined the domain model, i.e. the teacher knowledge, the student knowledge and how to characterise a didactic item. Then we have made an object-oriented design of those different kinds of knowledge. It helps us to make a prototype based on a clients-servers architecture, composed of an object database, a distributed object server, and a Web server.

Keywords :

Compter-aided teaching | Adaptive hypermedia | Knowledge enginnering
Software enginnering | Client-server architecture

Laboratoire PSI
INSA de Rouen, BP08, Place Emile Blondel, 76131 Mont Saint Aignan Cedex