Un environnement de développement pour TECFA

Réflexions et propositions pour la mise en place d'un environnement commun de développement d'application pédagogique au sein de l'unité TECFA.

Olivier Clavel

Sous la direction du Dr Daniel Schneider.

Mémoire présenté en vues de l'obtention du diplôme d'étude supérieure STAF
(Sciences et Tehcnologies de l'Apprentissage et de la Formation)

Soutenance le 30.10.2002 à l'université de Genève.

Jury:
Barbara Class, Assistante de recherche, TECFA, Université de Genève
Samir Fitouri, Chargé d'enseignement, TECFA, Université de Genève
Dr Daniel Schneider, Chargé de cours, TECFA, Université de Genève

Table des matières

1 Introduction

TECFA (Technologie de Formation et Apprentissage) est une petite unité académique de la Faculté de Psychologie et des Sciences de l'Education (FPSE) de l'Université de Genève. L'unité TECFA est active dans le domaine des technologie éducatives. Le personnel de TECFA est impliqué dans différents enseignements au sein de la FPSE. Plus particulièrement, TECFA propose un diplôme d'études supérieures en Sciences et Technologies de la Formation et de l'Apprentissage (STAF). Le plan d'étude s'étend sur deux années et donne une large place à l'enseignement par projets organisé en périodes. Chaque période comporte une semaine de cours présentiels suivit par un temps de travail à distance avec des interactions régulières entre les enseignants et les étudiants. L'accent est également mis sur la collaboration entre les étudiants pour effectuer leur travaux.

Ce diplôme constitue un très bon terrain d'essai et d'expérimentation pour la petite équipe de TECFA qui cherche constamment à se tenir à la pointe en matière de technologies éducatives. Toujours à la recherche de l'innovation, cette équipe ne peut se satisfaire d'utiliser seulement des produits déjà présent sur le marché. Des développement internes sont menés pour satisfaire des besoins spécifiques et mettre à disposition des activités pédagogiques pour les étudiants.

Cependant, les ressources humaines ne sont pas suffisamment abondantes et stables pour permettre de développer, de maintenir et de faire évoluer constamment un produit de qualité professionnelle. Les développements se situent donc toujours au niveau du prototype évolué. En même temps, ces prototypes sont tout de même utilisés, soit par les étudiants du diplôme STAF, soit comme vitrine des savoirs-faire de la maison. La multiplication des développements pour les différents enseignements fini par pousser l'équipe à construire un outil qui permette de regrouper cet ensemble de ressources pédagogiques au sein d'un espace commun: le *campus virtuel*

Porteur d'espoir et objet de beaucoup d'enthousiasme à ses débuts, le campus virtuel ne survivra malheureusement pas vraiment à son vieillissement prématuré et tombera petit à petit dans l'oubli, principalement à cause d'un manque de maintenance et d'une trop grande rigidité. Faut-il pour autant abandonner l'idée d'un espace commun de développement pour les applications pédagogiques au sein de cette unité et laisser chaque projet évoluer dans son coin de façon indépendante ?

Bien qu'il soit formateur pour les étudiants et les assistants de mener des développements depuis le point zéro, il parait cependant aberrant que les membres de cette équipe "ré-inventent la roue" à chaque fois qu'un nouveau prototype est mis en chantier. Il paraîtrait donc judicieux de pouvoir proposer un ensemble stable de "fonctions" ou "services" de base qui composeraient une fondation solide pour le départ de chaque nouveau développement. Pour ne donner qu'un exemple simple, pourquoi ne pas utiliser une base d'utilisateurs unique et des fonctions d'authentification efficaces plutôt que de laisser chaque développeur perdre du temps à reconstruire une solution complète pour chaque nouveau projet d'application.

Le travail que nous présentons ici essayera de poser les bases pour la mise en place d'un nouveau dispositif qui puisse remplacer le projet de campus virtuel maintenant totalement sur le déclin. Nous essaierons de proposer des solutions pour maintenir un environnement léger, ouvert, évolutif, adaptable et stable qui puisse constituer une base

commune de développement d'applications pédagogiques pour l'ensemble de l'équipe TECFA

Pour cela, nous commencerons tout d'abord par présenter la philosophie générale du campus virtuel et mettre en évidence les principales causes de son échec au travers de son historique. Puis nous nous intéresserons aux différentes catégories de philosophies de l'apprentissage et aux différents modèles des processus de la transmission, de l'acquisition et de la construction des connaissances qu'elles ont générés. Nous poserons ensuite un cadre de référence qui nous permettra d'appréhender toute la complexité des processus à supporter dans un dispositif pédagogique selon le modèle d'enseignement choisi.

Grâce à ce cadre de référence, nous étudierons les différentes catégories de dispositifs pédagogiques qui existent actuellement sur le marché. Nous passerons en revue les outils et les fonctionnalités qu'ils proposent et les différents modèles d'enseignement qu'ils permettent de supporter.

A la lumière des ces informations, nous ébaucherons ensuite le cahier des charges du dispositif à mettre en place et passerons en revue les différentes solutions techniques qui existent pour le mettre en oeuvre. Nous proposerons ensuite une architecture de base et des directives pour son évolution, son utilisation, sa maintenance et nous présenterons les premiers essais de développement pour illustrer la mise en pratique de nos proposition.

Ce travail n'a pas la prétention de donner une recette universelle. Il doit plutôt être vu comme une première approche exploratoire qui pourrait être utilisé par la suite comme piste de réflexion pour la mise en place d'un véritable environnement de développement léger et ouvert qui permette le partage et la réutilisation des ressources produites à TECFA. Durant le développement de ce travail, nous avons garder à l'esprit les interrogations suivantes:

- Quelles doivent être les composantes du dispositif pour qu'elles soient à la fois suffisantes et non contraignantes ?
- Quels outils minimaux de bas niveau doivent être disponibles pour les développeurs ?
- Comment faut-il concevoir l'architecture technique du dispositif pour qu'elle soit adaptable, maintenable avec un minimum de ressources humaines et évolutive ?
- Comment faciliter l'utilisation du dispositif et quels processus mettre en place pour qu'il soit utilisé et maintenu de façon durable ?

Bien que ce travail soit focalisé sur les problèmes rencontrés spécifiquement à TECFA, les réflexions qui y sont menés pourront s'avérer utiles à d'autre structures similaires. Nous pensons notamment à des structures dont l'activité principale n'est pas l'informatique mais qui développe cependant des applications Web légères. Au niveau théorique tout d'abord, l'étude que nous avons menés des différents dispositifs pédagogiques existants pourra servir de base de réflexion pour choisir un système adapté à une situation d'enseignement particulière. Par ailleurs, les différents points techniques abordés dans ce document tels que l'authentification centralisée, les solutions pour le stockage de données, les différents formats de stockage et de descriptions de ressources, l'organisation et la gestion d'un projet de développement, la réutilisation des ressources produites... sont des problèmes récurrents dans le monde de l'informatique.

2 Le campus TECFA jusqu'à aujourd'hui.

2.1 Les prémices

Nous allons remonter jusqu'en 1993, date de la naissance du serveur Web TECFA qui sera historiquement le premier serveur web pédagogique en suisse (premier serveur installé par une unité faisant de l'enseignement). Le serveur est principalement utilisé comme système d'information interne, pour la publication des recherches, pour donner de l'information sur TECFA et pour proposer quelques ressources.

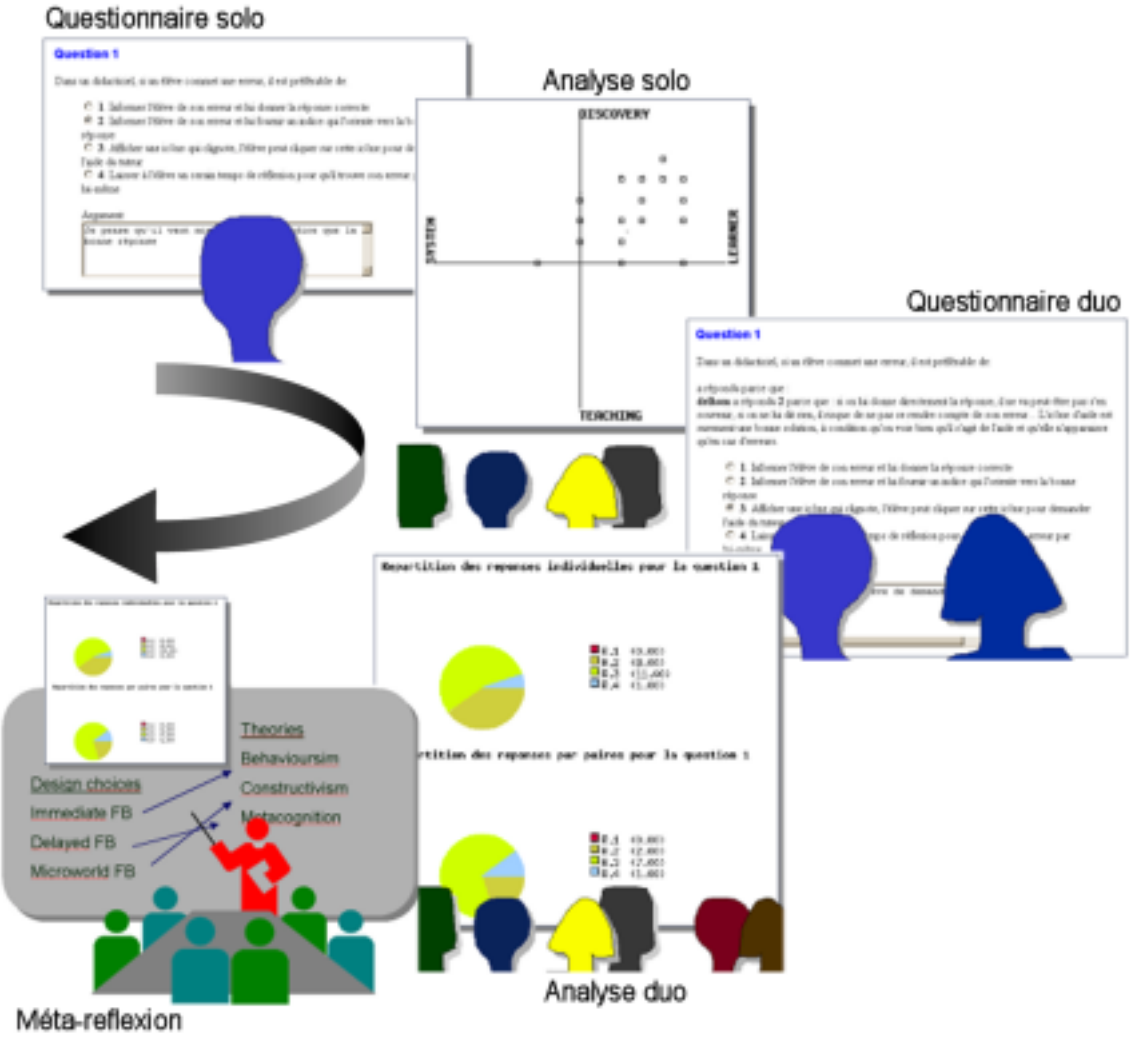
En 1994 le diplôme d'études supérieures STAF (Sciences et Technologies de l'Apprentissage et de la Formation) est créé et accueille sa première promotion. C'est le début de l'utilisation du serveur Web pour l'enseignement. On commence alors à y trouver les descriptifs des cours et des travaux, les ressources pour chaque enseignement (bibliographie, fiches concepts, liens vers d'autres sites) et les home pages des étudiants. C'est l'époque des pages statiques. Parallèlement au serveur web, on pourra aussi noter l'installation du serveur MOO (MUD [Multi User Dungeon] Object Oriented), un environnement de réalité virtuelle textuel.

Dans les années qui suivent, le serveur web s'étoffe et de nouvelles technologies sont intégrées ou utilisées de façon complémentaire. Parmi celles qui apportent le plus de changement, il faut noter l'introduction des bases de données à partir de 1995 qui conduiront aux pages interactives en 1996. On entre alors dans la période des pages dynamiques qui vont être de plus en plus nombreuses sur le serveur.

Puis apparaissent en 1997 les premières activités pédagogiques. Elles se développent sous l'impulsion de quelques enseignants pour quelques uns de leurs enseignements. Les premières à être utilisées sont "l'Argue Graph" et le "Studio" que nous allons présenter plus en détails.

"L'Argue Graph" a été dessiné comme une activité permettant de fournir un support de cours sur les théories de l'apprentissage utiles dans la conception de didacticiels. La figure 1 schématise le déroulement de l'activité. Les apprenants répondent d'abord séparément à un questionnaire sur les choix de design pour un didacticiel. Le questionnaire est sous forme de QCM avec un champ texte permettant de fournir une argumentation courte. Il est important de noter que les réponses proposées ne sont ni bonnes ni mauvaises mais correspondent à des philosophies assez différentes en matière de design et de pédagogie. Les données de chaque questionnaire sont récupérées en temps réel et reportées sur un graphique qui va permettre d'analyser les réponses de façon globale pour tout le groupe. L'enseignant forme ensuite des paires. Chaque paire répond au même questionnaire. La seule différence avec le questionnaire solo, c'est que les réponses et l'argumentation de chaque participant pour chacune des réponses apparaît à l'écran. De la même façon, les réponses sont enregistrées. Une fois que toute les paires ont répondu, on peut accéder à un graphique synthétique présentant l'évolution des résultats par rapport au premier questionnaire. Ce graphique sert alors de base à une discussion entre l'enseignant et les apprenants.

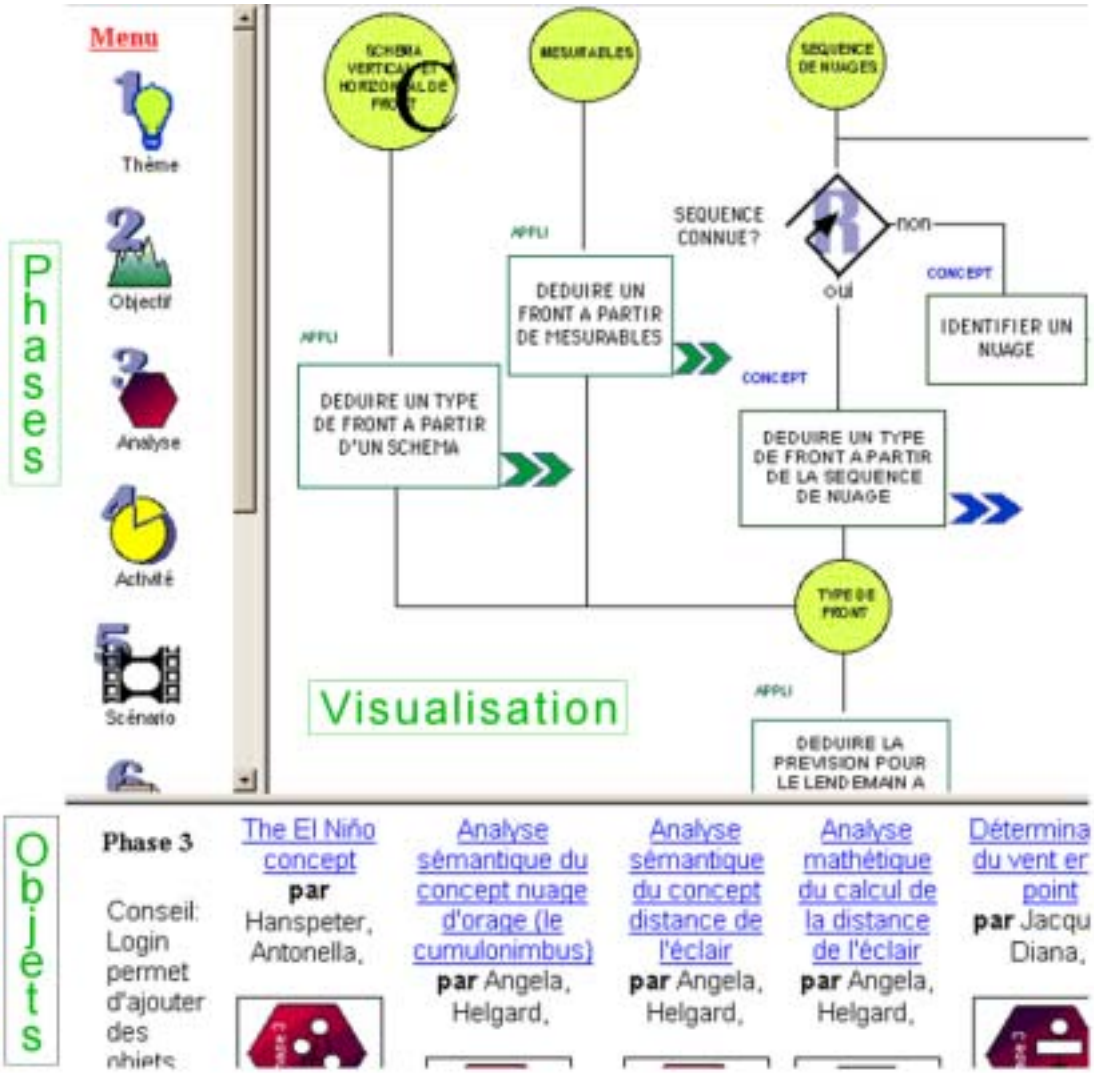
Fig 1.L'activité "Argue Graph"



Pour le studio, l'usage est un peu différent. Il s'agit ici de fournir un environnement de travail facilitant la création de tutoriels de façon collaborative. Les étudiants sont ainsi guidés dans les différentes phases d'analyse et de création du programme et peuvent échanger des concepts ou des morceaux de programmes. La figure 2 montre un écran de cet environnement. Dans ce cas, l'activité constitue plus une aide et une motivation externe pour créer un produit fini qu'un réel support d'enseignement en salle.

C'est suite à la création de ces activités phares qu'à commencé la création du campus virtuel qui voit le jour et commence à être utilisé par les étudiants en Octobre 1998.

Fig 2.L'activité "Studio"



2.2 Le concept du campus virtuel TECFA et son design.

Le campus virtuel TECFA est conçu pour offrir un support aux étudiants qui suivent les cours du diplôme STAF sur un mode mixte présence/distance. Ce dispositif doit pouvoir supporter différents types d'interactions, d'activités et d'enseignements. Le principe de base est de permettre à une communauté d'utilisateurs d'accéder à un espace structuré et de collaborer entre eux. Les buts de ce dispositif sont multiples:

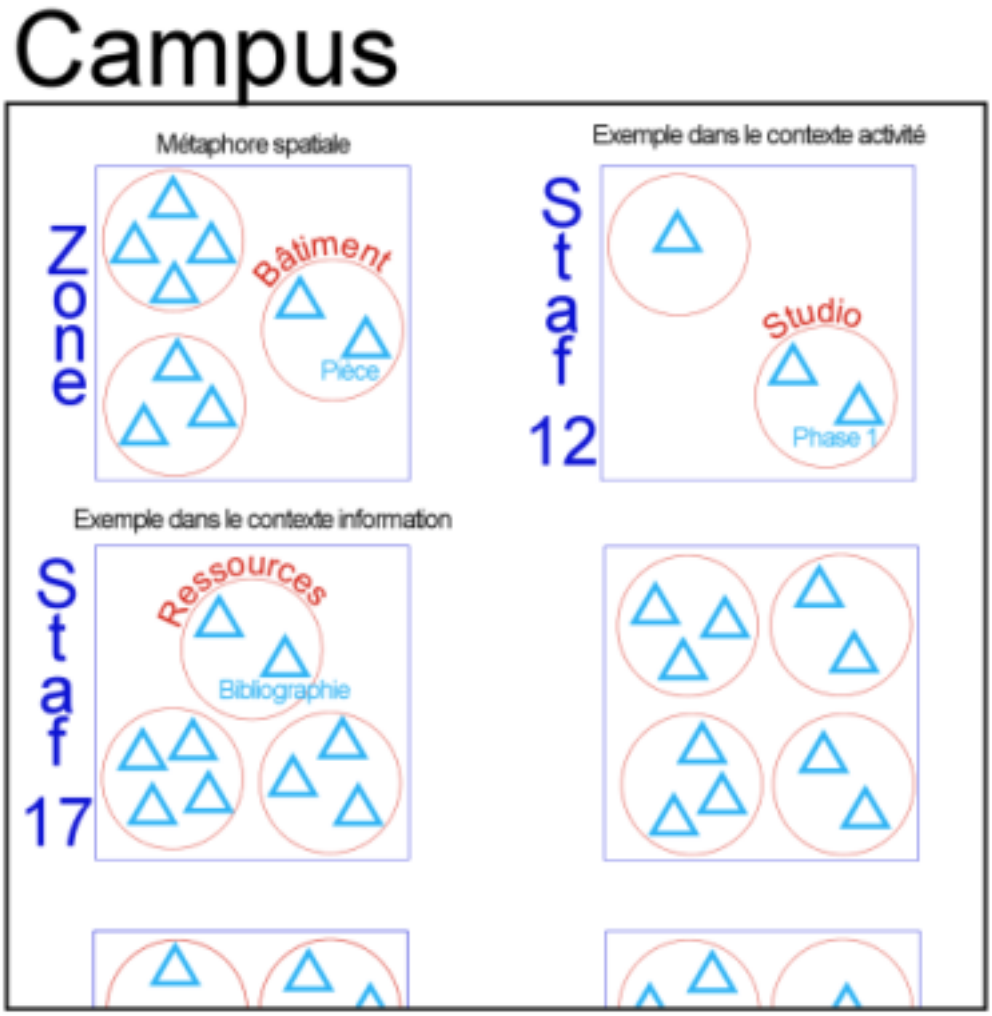
- Fournir un support de cours pendant les périodes présentielles, que ce soit pour des activités (comme dans le cas de l'activité "l'argue graphe") ou de la diffusion de document sur écran.
- Proposer des ressources à la communauté des étudiants.
- Encourager la collaboration entre les étudiants notamment pendant les périodes à distance (comme dans le cas de l'activité "studio").
- Encourager et faciliter la création d'un sentiment de communauté chez les utilisateurs.

Pour essayer de faciliter la navigation, l'architecture générale se base sur une métaphore spatiale. Chaque cours du diplôme correspond à une "Zone". A l'intérieur des cette zone, l'enseignant responsable possède encore deux niveaux pour organiser l'espace: le "bâtiment" et la "pièce". Selon les enseignements et les enseignants responsables, le bâtiment et les pièces qui le composent jouent des rôles assez divers. Toutefois on peut schématiser leurs usages en deux catégories:

- dans le cas des activités, le bâtiment constitue un atelier subdivisé en différentes étapes ou sous-activités. C'est par exemple le cas du studio présenté plus haut qui a été repris dans le campus et où chaque phase constitue une pièce de ce même atelier.
- dans le cas de la présentation brute d'informations, le bâtiment sert à regrouper celles-ci en catégories, selon leur utilisation. On peut trouver par exemple un "centre de ressources" regroupant des pièces "bibliographie", "fiches d'expertises", Fiches concepts"... ou encore des bâtiments.

La figure 3 synthétise la correspondance entre la métaphore spatiale du campus et les concepts qui lui sont rattachés.

Fig 3.La métaphore spatiale du campus virtuel



Un certain nombre d'outils d'information sont proposés et présents quel que soit l'endroit où l'utilisateur se trouve. On peut par exemple:

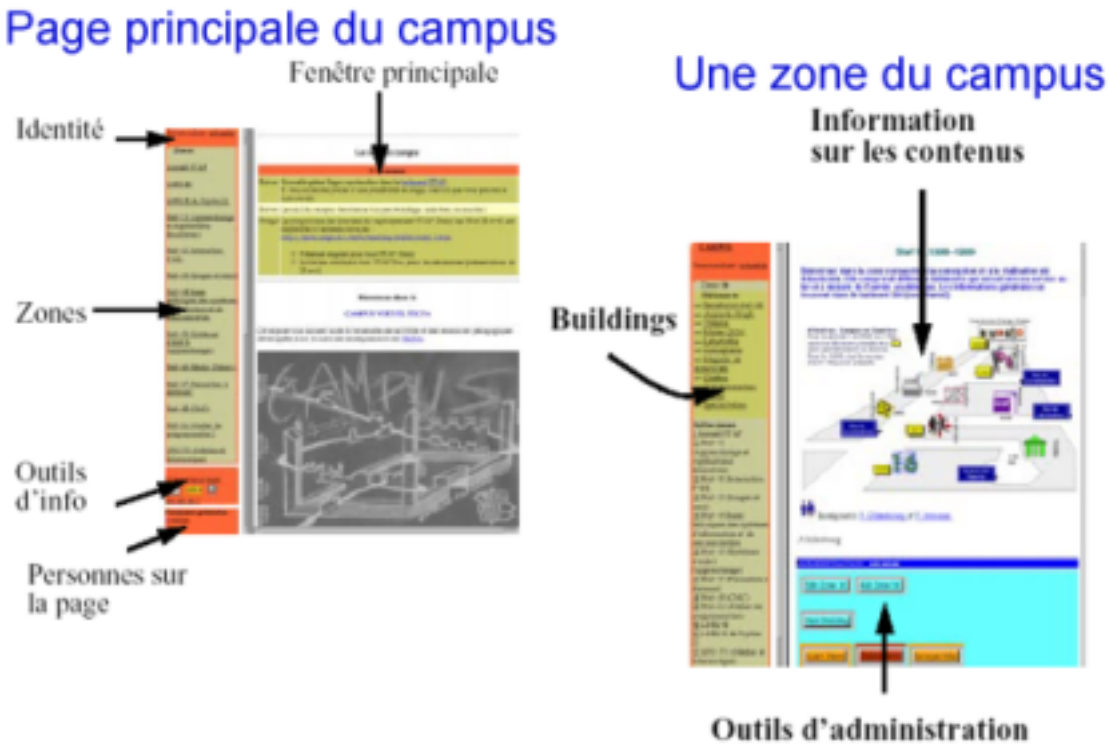
- accéder à sa fiche personnelle qui récapitule les informations fournies (email...) et les groupes auxquels il appartient.
- voir les nouvelles qui ont été déposées par les administrateurs
- accéder à une carte présentant la fréquentation des différents lieux du campus
- utiliser une vue du campus en deux dimensions permettant de naviguer à l'intérieur.
- prendre connaissance de la liste des utilisateurs et consulter pour chacun d'eux sa fiche personnelle, la date et l'heure de sa dernière connection et l'endroit où il se trouvait dans le campus à ce moment là.

Enfin, pour renforcer le sentiment de communauté, une liste des personnes présentes sur la page que l'utilisateur est en train de visualiser est systématiquement présentée.

La figure 4 présente une vue de la page d'accueil du campus et la vue d'une zone. Sur la vue de la zone, on peut noter la présence d'un outil d'administration. Ce dernier ne s'affiche que quand l'utilisateur enregistré fait partie du groupe *ad-hoc* et permet d'entreprendre différentes actions:

- ajouter/ supprimer/ modifier des zones/ bâtiments/ pièces
- ajouter des nouvelles
- ajouter ou éditer des utilisateurs

Fig 4.Vue de la page d'accueil et d'une zone du campus virtuel TECFA



Les technologies retenues pour l'implémentation du campus sont le langage de script

PHP couplé à la base de donnée MySQL, technologies enseignées à TECFA, installées sur différents serveurs, bien connues de la plupart des assistants et d'un abord plutôt facile pour les non-spécialistes. Les différents développeurs travaillent sur une base de données commune dont la structure permet de stocker tous les objets du campus (utilisateurs, groupes, zones, bâtiments, pièces) ainsi que leurs interdépendances (utilisateurs<->groupes, zones<->bâtiments, bâtiments<->pièces). Quelques fonctions php communes sont implémentées (authentification, manipulation de données, gestion des utilisateurs...)

2.3 La vie du campus, les succès et les échecs

Comme nous l'avons déjà signalé plus haut, le campus a commencé à être utilisé par les étudiants à partir du mois d'octobre 1998 soit à la rentrée universitaire. Pour ce début, les étudiants sont plutôt enthousiastes. Les différents lieux du campus se remplissent et s'étoffent petit à petit. On notera par exemple le développement de plusieurs autres activités tels que "l'iconomètre" permettant de tester l'indice monosémique de collections d'icônes ou "les bureaux virtuels" pour la gestion de projet de création de systèmes d'aide à l'apprentissage.

Mais ce début marque également l'apparition des premiers problèmes. Ils seront d'abord d'ordre conceptuel et pratique. En effet, bien que le projet ait été décidé en groupe, il y a un manque de consensus au sein de l'équipe.

- La métaphore spatiale ne fait pas l'unanimité de tous les acteurs. Pour certains, elle essaie trop de calquer la réalité alors qu'ils auraient préféré une conceptualisation plus abstraite de cet espace virtuel. Ironiquement, ils qualifieront le campus de "projet blanche-neige".
- Certains enseignants utilisent des supports de cours qui sont déjà faits depuis des années avec d'autres technologies et qui fonctionnent très bien. Ils ne veulent pas passer du temps à adapter ces contenus au campus car rien n'a été prévu pour les intégrer facilement s'ils n'ont pas été conçu de façon spécifique. Le recours à des formulaires en ligne pour intégrer du contenu au campus rebutera plusieurs personnes et beaucoup céderont à la tentation, de façon temporaire ou définitive, de ne mettre dans le campus que des liens vers la version antérieure de leurs supports de cours.

A la fin de la première année d'utilisation, le bilan est tout de même positif. Le campus a bien été utilisé par les étudiants. Ils comprennent apparemment la métaphore spatiale qui régit son espace et les différentes activités remportent un certains succès. Il est néanmoins intéressant de noter les premières critiques. La plus flagrante est qu'il est assez difficile de retrouver des informations spécifiques. Cette difficulté est renforcée par le fait que certaines informations sont dupliquées entre le campus et le site web *sensu-stricto* de TECFA. Par exemple, personne ne sait jamais exactement à quel endroit trouver le programme des cours de la semaine présentielle, celui-ci changeant de place à chaque période en fonction du professeur responsable de la semaine présentielle en question.

Bien que cette première année soit également émaillée de petits problèmes techniques, les plus importants apparaissent lors de la deuxième et de la troisième année d'utilisation. Une mise à jour du campus s'impose et certaines activités ont besoin d'être remises à zéro.

La base de donnée des utilisateurs n'a pas été prévue pour prendre en compte les

promotions d'étudiants, ou ceci a été fait tellement dans l'urgence que des erreurs de design difficilement rattrapables ont été commises. Par exemple, beaucoup d'activités reposent sur des groupes conceptualisant à la fois l'année d'étude et les cours suivis. Ces groupes ont été créés de manière spécifique alors qu'ils pouvaient être inférés à partir d'autres informations se trouvant ailleurs dans la base de données. Ceci oblige à une création incessante de nouveaux groupes et au nettoyage des anciens pour ne pas surcharger l'affichage de certains formulaires au risque de casser certaines associations dans la base de données et par la même des application entières.

La maintenance technique du campus s'avère également difficile. Les données et la logique des applications sont éparpillées entre diverses bases de données, des fichiers HTML inclus, des librairies de fonctions PHP. La documentation fait cruellement défaut. Certaines bases de données ont été développées de façon concurrente alors que la finalité était la même. Par exemple, certains développeurs ont créé leur propre base d'utilisateurs alors que le campus à déjà la sienne. On en arrive à des situations cocasses où l'utilisateur est obligé de s'authentifier sur le campus pour atteindre une page d'authentification pour une application.

La difficulté de maintenance est accrue par la fait que la plupart des développeurs qui ont contribué à ce projet arrivent en fin de parcours à l'université et quittent TECFA les uns après les autres. Les suivants doivent alors percer le secret assez obscur de milliers de lignes de code, souvent sans commentaires internes et sans documentation comme nous l'avons déjà dit plus haut.

Petit à petit, des morceaux entiers du campus se dégradent, les pannes deviennent de plus en plus fréquentes, les bugs de plus en plus difficiles à réparer.

Dans le courant de la troisième année d'utilisation, lors d'une évaluation interne de l'utilisation du campus par les étudiants, il apparaît que ceux-ci ne l'utilisent plus que par obligation. D'autre part, la métaphore spatiale, bien que comprise dans son ensemble est très peu appréciée et semble poser des problèmes de navigation à l'intérieur du dispositif.

Suite à une série de réunion de l'équipe, il est décidé d'arrêter la maintenance coûteuse de ce dispositif et de repartir sur de nouvelles bases. Nous allons essayer de définir, dans les chapitres qui suivent, quelles doivent être ces nouvelles bases. Nous commencerons, comme nous l'avons annoncé dans l'introduction, par essayer de caractériser les fonctions des dispositifs pédagogique en nous intéressant aux différentes philosophies pédagogique et aux évolutions quelles ont apporté au monde de l'éducation, plus particulièrement en matière de technologie, à travers le temps.

3 Les grandes catégories de philosophies de l'enseignement et de l'apprentissage.

La conception théorique de l'enseignement et de l'apprentissage à évolué dans le temps. Cette évolution a donné naissance à différentes philosophies qui ont chacune apporté des idées nouvelles et proposé des modèles différents des processus de la transmission, l'acquisition et la construction des connaissances. Pour être utilisé dans des environnements informatisés, ces différents concepts demande la mobilisation de plus ou moins de ressources et sont plus ou moins compliqué à mettre en oeuvre.

En schématisant et sans trop entrer dans les détails, on peut ramener les types de modèles pédagogiques (ou philosophies d'enseignements) à quatre grandes classes:

- le béhaviorisme
- le constructivisme
- le socio-constructivisme
- la cognition située et distribuée

Le but n'est pas ici de discuter quelle philosophie est supérieure à l'autre, ce qui à notre sens est une discussion stérile. Il convient plutôt de savoir laquelle est la mieux adaptée à ou la plus performante pour la matière ou le sujet que l'on veut enseigner.

Voyons donc pour chacune de ces classes quels sont ses fondements théoriques et les implications de son application à l'enseignement.

3.1 Le béhaviorisme

Le Behaviourisme au sens pédagogique trouve son origine dans les théories béhavioristes de la psychologie comportementale et plus particulièrement dans la théorie du conditionnement pavlovien qui en signe le début. On considère ici l'apprenant comme une "boite noire" que l'on va conditionner pour acquérir un comportement par un jeu de stimulus - réponse - renforcement.

La technologie de l'enseignement programmé (ou programme linéaire) qui a connu ses lettres de noblesse à partir de 1954 sous l'impulsion du psychologue de l'apprentissage B.F. Skinner est le premier exemple d'application des théories béhavioristes. Elle se fonde sur des principes simples empruntés aux études expérimentales et plus particulièrement au conditionnement opérant selon lequel *"si l'occurrence d'un opérant est suivie par la présentation d'un stimulus de renforcement, sa force en est augmentée"* (Skinner, 1938). Un "opérant" est une unité de comportement qui n'est pas obtenue de manière consistante avec un stimulus particulier, par opposition à un "répondant" qui lui l'est. L'enseignement devient donc *"simplement l'arrangement des évènements de renforcement"* (Skinner, 1968)

L'évènement important étant ici le renforcement qui suit une bonne réponse, il en résulte qu'il faut:

- utiliser des questions-réponses nombreuses avec renforcement immédiat

- favoriser les renforcements positifs pour les bonnes réponses aux dépens des renforcements négatifs pour les mauvaises en rendant ces dernières hautement improbables

L'application de ces principes a des implications sur la préparation du matériel d'enseignement:

- la matière à enseigner est fragmentée en une série d'éléments courts afin que le renforcement soit le plus immédiat possible. Le renforcement correspond ici aux constat de l'exactitude de la réponse donnée.
- les éléments sont présentés sous forme de séquence linéaire. Cette séquence est la même pour tous les sujets. Cependant, chaque sujet peut parcourir la séquence à sa propre vitesse.
- la séquence des éléments part des notions les plus simples et générales pour conduire aux notions les plus complexes et précises
- le matériel d'enseignement est testé pour éliminer au maximum les éléments conduisant à des réponses erronées afin de minimiser les risques de fixation des mauvaises réponses et le recours au feedback négatif est considéré comme peu efficace. Si une question conduit à une mauvaise réponse, la solution consiste généralement à introduire des éléments supplémentaires dans la séquence afin de diminuer la difficulté dans la progression de l'apprentissage entre chaque étape.

L'objectif immédiat de ce type d'enseignement et de mettre le sujet dans la capacité de fournir la bonne réponse. Idéalement, tous les sujets devraient fournir la bonne réponse à chaque question sans jamais se tromper. Bien que l'apprentissage puisse conduire par la suite à la construction d'une représentation d'ensemble qui apparaisse comme nécessaire, cette notion n'est pas utilisée dans l'enseignement programmé.

La contribution principale de l'enseignement programmé aux théories globales de l'apprentissage est d'avoir mis en avant l'importance du feedback et de l'individualisation de l'enseignement (la vitesse d'acquisition du matériel est libre d'un individu à l'autre). Mais le feedback n'est considéré dans ce cas comme important qu'après une bonne réponse alors qu'il peut jouer un rôle bien plus constructif (O'Shea & Self, 1983).

C'est Crowder, un contemporain de Skinner, qui va, le premier, commencer à donner au feedback un rôle plus important avec la programmation ramifiée (ou programmation intrinsèque). *"Le problème essentiel est celui du contrôle d'un processus de communication par l'utilisation du feedback. La réponse de l'étudiant est avant tout un moyen de déterminer si le processus de communication a été efficace et permet en même temps de mener une action réparatrice appropriée"* (Crowder, 1959). Ici, toute réponse donnée par l'apprenant est considérée comme importante et doit mener à un feedback approprié. Le matériel d'enseignement est sensiblement différent:

- Les éléments présentés aux sujets sont beaucoup plus longs (sous forme de petits exposés).
- A la fin de chaque élément, une question est posée avec plusieurs réponses possibles.
- Les réponses peuvent être vraies ou fausses mais aussi plus ou moins acceptables. On voit ici une incompatibilité avec l'enseignement programmé puisque le sujet est constamment mis en présence de réponses erronées.

- La séquence des éléments dépend de la réponse de l'étudiant. S'il s'est trompé, il recommence avec le même élément ou peut également être amené vers une autre série d'exercices destinés à combler sa lacune.

La programmation ramifiée apporte donc une dimension supplémentaire à l'individualisation et à l'importance du feedback:

- non seulement le temps reste libre pour parcourir le matériel mais celui-ci sera différent d'un apprenant à l'autre puisque dépendant des réponses données.
- le feedback est considéré comme important même dans le cas ou une mauvaise réponse est donnée.

Toutefois, l'enseignement programmé et la programmation linéaire gardent en commun la croyance que l'élève apprend en "étant enseigné" (O'Shea & Self, 1983). C'est notamment ce point particulier qui a donné lieu à beaucoup d'oppositions et de controverses à l'application des théories béhavioristes dans l'enseignement, dont la célèbre Théorie Hydraulique de l'Education de Paul M. Davies (1969). *"Il y a une théorie de l'éducation très répandue que l'on pourrait appeler la Théorie Hydraulique [...] D'après [celle-ci], le savoir est une sorte de liquide que l'on trouve de façon abondante dans les enseignants et les livres, comme dans de grands vaisseaux, et difficilement où que ce soit ailleurs. En particulier ce liquide est rare dans le petit vaisseau connu sous le nom d'étudiant. Le but de l'éducation, donc, est de transférer ce liquide du grand vaisseau vers les plus petits [...]".*

Bien que le béhaviorisme se base sur des concepts très simples et constitue certainement la forme la moins évoluée des théories de l'apprentissage, il est encore largement utilisé, notamment dans les logiciels dit "d'enseignement assisté par ordinateur" (EAO). Certains concepts sont aujourd'hui réutilisés comme, par exemple, l'attention particulière donnée à la fixation de l'erreur dans l'enseignement de connaissances syntaxiques telles que l'orthographe.

3.2 Le constructivisme

La théorie du constructivisme trouve son origine dans les travaux de Jean Piaget, psychologue du développement. La théorie de Piaget, connue sous le nom de constructionnisme comporte deux composantes majeures: "âges et étapes" qui prédit ce que les enfants peuvent et ne peuvent pas comprendre à un âge donné, et une théorie du développement qui décrit comment les enfants développent leurs capacités cognitives.

La théorie piagétienne du développement fait la proposition qu'on ne peut pas "donner" de l'information à l'homme qu'il puisse immédiatement comprendre et utiliser. Au lieu de cela, l'homme doit "construire" son propre savoir. Il construit son savoir à travers l'expérience. Les expériences lui permettent de créer des schémas (des modèles mentaux dans sa tête). Ces schémas sont changés, élargis, et deviennent plus sophistiqués au travers de deux processus complémentaires:

- l'assimilation (l'incorporation des informations extérieures en schémas internes)
- l'accommodation (la modification de ses propres schémas pour les faire cadrer avec la réalité extérieure observée)

Une des généralisations importantes de la théorie de Piaget concerne le rôle de l'enseignant. Dans une salle de classe piagétienne, un des rôles de l'enseignant est de

fournir un environnement d'apprentissage riche qui permette l'exploration spontanée de l'enfant. Une salle de classe remplie de choses intéressantes à observer encourage les apprenants à devenir des constructeurs actifs de leur propre savoir (et de leurs propres schémas) à travers des expériences qui encouragent l'assimilation et l'accommodation.

En se basant en grande partie sur les travaux de Piaget, Jerome Bruner à, lui aussi, proposé une théorie du constructivisme. Un des thèmes majeurs de cette théorie et que l'apprentissage est un processus actif dans lequel l'apprenant construit de nouvelles idées et de nouveaux concepts basés sur ses connaissance actuelles et antérieures. L'apprenant, en s'appuyant sur sa structure cognitive, sélectionne et transforme l'information, construit des hypothèses, et prend des décisions. La structure cognitive (i.e. schémas, modèles mentaux) apporte une signification et une organisation aux expériences de l'individu et lui permet "d'aller plus loin que l'information qui lui a été donnée".

En ce qui concerne l'instruction, l'enseignant doit encourager les apprenants à découvrir les principes par eux mêmes et les aider dans cette tâche. Les apprenants et l'enseignant doivent s'engager dans un dialogue actif (i.e apprentissage socratique). La tâche de l'enseignant et de traduire l'information qui doit être apprise dans un format adapté à la capacité de compréhension de l'apprenant. Le programme d'étude doit être organisé "en spirale" afin que l'apprenant puisse constamment construire de nouveaux savoirs sur ceux déjà acquis.

Bruner (1966) déclare qu'une théorie de l'instruction devrait aborder quatre points essentiels:

1 la disposition à apprendre (motivation),

2 les façons de structurer un ensemble de savoirs pour que l'apprenant puisse facilement le comprendre,

3 les séquences de présentation du matériel les plus efficaces,

4 la nature et la fréquence des récompenses et des punitions.

L'application d'une bonne méthode de structuration des connaissances devrait avoir pour résultat de les simplifier, de générer de nouveaux points de vue et d'augmenter la manipulation d'informations et de données pour l'apprenant.

Voyons maintenant quelles sont les implications pratiques de cette théorie dans le monde de l'enseignement. Cette liste s'inspire d'un article de Hoover (1996).

- En premier lieu, l'enseignement ne peut plus être vu comme une transmission de savoir depuis une personne éclairée à une personne non éclairée. Un enseignant constructiviste ne prend pas le rôle de "sage on the stage" (sage sur la scène). Il doit plutôt agir comme "guide on the side" (guide sur le côté) qui donne aux étudiants la possibilité de tester l'adéquation de leur compréhension du moment.
- Si l'apprentissage est basé sur les connaissances antérieures, l'enseignant doit alors prendre ces connaissances en compte et proposer un environnement d'apprentissage qui exploite les contradictions entre les compréhensions du moment de l'apprenant et les nouvelles expériences qui s'offrent à lui. Ceci est un défi pour l'enseignant puisqu'il ne peut pas supposer que deux apprenants comprennent la même chose de la même façon. De plus, des apprenants différents peuvent avoir besoin d'expériences différentes pour évoluer vers un autre niveau de compréhension.

- Si les apprenants doivent appliquer leurs savoirs dans de nouvelles situations pour pouvoir construire de nouvelles connaissances, alors les enseignants doivent engager les apprenants dans l'apprentissage en mettant en avant les connaissances du moment des étudiants. Les enseignants doivent s'assurer que les expériences d'apprentissage incorporent des problèmes qui soient importants pour les apprenants, et pas des problèmes importants pour les enseignants eux-mêmes ou l'institution en général.
- Enfin, si la connaissance est construite de façon active, alors il faut du temps pour la construire. De larges plages de temps accordées aux apprenants facilitent la réflexion sur les nouvelles expériences vécues, l'alignement de ces nouvelles expériences avec les connaissances de l'étudiant et le développement d'une meilleure vue du monde par l'acquisition d'une compréhension différente.

Un des apports principaux de la théorie constructiviste aux théories générales de l'enseignement est la prise en compte plus grande de l'apprenant lui même dans le processus de l'apprentissage. Elle a également contribué à l'émergence d'environnements d'apprentissage informatique centrés sur l'apprenant ou sur l'expérience tel que les micro-mondes

3.3 le socio-constructivisme

Avant d'entrer dans le détail de la présentation de cette théorie, il nous semble important de faire une précision. Nous avons regroupé sous le terme *socio-constructivisme* deux théories qui se trouvent parfois présentée de façon totalement distincte dans la littérature: le socio-constructivisme *sensus stricto* et l'approche socio-culturelle. Ces deux théories ont en fait énormément de points communs. Leur différence majeure réside dans l'unité de base utilisé pour l'étude des phénomènes. L'école socio-constructiviste *sensus stricto* (qui provient de "l'école genevoise", un groupe de psychologue de Genève particulièrement influencé par les travaux de Piaget) se concentre sur le développement individuel et prend comme élément de base l'action individuelle. L'interaction sociale est vue comme un catalyseur permettant les changements individuels. L'école socio-culturelle (fondée sur les travaux de Vygotsky) quant à elle se concentre sur les relations causales entre les interactions sociales et les changements cognitifs individuels. L'unité d'analyse de base est l'activité sociale, qui constitue la base du développement des fonctions mentales. Les processus inter-psychologiques sont eux-mêmes internalisés par les individus concernés.

Cette distinction étant faite, nous regrouperons maintenant sous le terme *socio-constructivisme* l'ensemble de ces deux approches.

Selon la théorie socio-constructiviste, l'apprenant maîtrise de nouvelles habitudes d'apprentissage à travers l'interaction avec les autres. Le socio-constructivisme peut être vu comme une extension des travaux de Piaget (qui s'était concentré sur les raisons du développement cognitif au niveau individuel) en renforçant la partie faible de sa théorie: l'aspect social. Dans le socio-constructivisme, l'accent est mis sur l'***inter***-action plutôt que sur l'action elle même. A un niveau donné de développement intellectuel, l'individu est capable de participer à certaines interactions sociales qui produisent de nouveaux états individuels qui, à leur tour, permettent des interactions sociales plus sophistiquées, etc. (Dillenbourg et al, 1996). Ces interactions sont alors internalisées par le sujet: c'est le concept d'appropriation, qui peut être vu comme l'équivalent social de l'assimilation de Piaget dans la théorie constructiviste.

Le socio-constructivisme, plus particulièrement à la lumière des travaux de Vygotsky,

distingue les fonctions qu'un apprenant peut maîtriser tout seul et celles qu'il ne peut maîtriser qu'à l'aide d'une autre personne. C'est le concept de *zone de développement proximal* définie comme *"la distance entre le niveau de développement du moment déterminé par la résolution de problèmes de façon indépendante et le niveau de développement potentiel déterminé par la résolution de problèmes sous la direction d'un adulte ou en collaboration avec un pair plus compétent"* (Vygotsky, 1978).

Les interactions sociales qu'un individu expérimente modèlent donc une partie importante de son apprentissage et contribue à son développement par un effet d'échafaudage. Cependant, le bénéfice tiré de l'interaction par chaque membre n'est pas forcément équivalent, de même que la relation n'est pas forcément symétrique.

Cette théorie à poussé les chercheurs à savoir si l'apprentissage collaboratif était plus efficace que l'apprentissage solitaire. Il convenait aussi de savoir dans quelles conditions ce type d'apprentissage était plus efficace. Voici quelques résultats de ces recherches, synthèses très succinctes de ceux présentés par Dillenbourg et al (1996):

- Bien que des effets positifs puissent être trouvés à la collaboration, il peut y en avoir des négatifs. En particulier, il a été montré que les individus qui ont tendance à prendre peu d'initiatives deviennent totalement passif quand ils collaborent avec quelqu'un qui en prend beaucoup.
- Pour que la collaboration puisse prendre place, il faut un certain degré de différence entre les membres d'un même groupe afin que les conditions soient réunies pour que certains conflits émergent et que la relation sociale puisse être internalisée. Ces différences peuvent se trouver au niveau des points de vue des participants sur un sujet ou au niveau des aptitudes pour accomplir une certaine tâche. Si cette différence est trop petite, l'interaction aura du mal à s'initier. Au contraire, si cette différence est trop grande, il peut n'y avoir aucune interaction. Il faut donc trouver un niveau de différence optimum.
- Pour pouvoir collaborer, les individus en présence ont besoin de certaines compétences. Ils semble que l'interaction n'apporte aucun bénéfice à un individu s'il ou elle se trouve en dessous d'un certain niveau de développement. Il s'agit ici du niveau absolu de développement individuel, pas celui relatif aux autres membres du groupe. In semble donc que certaines capacités à comprendre les états mentaux d'autres personnes soit requis chez les différents individus pour que la collaboration puisse prendre place.
- Enfin, il est assez évident que la capacité à collaborer et à apprendre quelque chose de nouveau de cette interaction dépend également du type de tâche à accomplir. Un changement conceptuel sera, par exemple, difficilement observable chez deux individus collaborant sur un travail purement procédural et demandant peu de compréhension. De même, il sera difficile d'observer une amélioration des capacités de régulation si la tâche ne demande aucune planification. Certaines tâches sont tout simplement moins partageables que d'autres.

Si cette théorie a le mérite de faire émerger l'importance des relations sociales de façon générale dans l'éducation, l'analyse des interactions se situe toujours sur le plan inter-individuel (et non pas sur le plan social). Par exemple, s'il a bien été observé que le fait d'expliquer quelque chose à quelqu'un amenait à améliorer sa propre connaissance (Webb, 1991), ce résultat est attribué à l'effet d'auto-explication. En s'obligeant à expliquer sa vision des choses à son partenaire, l'apprenant verbalise sa pensée et la re-

assimile, ce qui fait évoluer son schéma mental personnel. Nous verrons plus loin (voir section 3. 4 page 17) que l'école de la cognition distribuée a une approche sensiblement différente de ce problème et que l'explication n'est plus considérée comme une simple transmission d'information entre celui qui explique et celui qui écoute.

Dans la pratique, il découle de cette théorie que l'interaction entre les apprenants doit être favorisée autant que celle entre les apprenants et le formateur. L'enseignant doit encourager les interactions de groupe où les échanges entre les participants aident les apprenants à expliciter leur propre compréhension en la comparant à celle de leurs pairs. Mais un des concepts qui peut être le plus difficile à accepter est que, si le constructivisme nous a montré que l'enseignant n'est pas un vecteur du savoir mais un facilitateur de son acquisition, le socio-constructivisme, quant à lui, postule que l'enseignant apprend également en interagissant avec ses étudiants.

3.4 La cognition située et distribuée

La théorie de la cognition située et distribuée passe encore une étape supplémentaire dans l'analyse des interactions sociales entre les acteurs de la formation. Elle s'inspire de travaux menés en sociologie et en ethnographie. Elle est différente de la théorie du socio-constructivisme dans le sens qu'elle se concentre sur l'environnement dans lequel l'apprentissage intervient plutôt que sur les processus cognitifs indépendants de l'environnement d'apprentissage.

L'environnement est constitué par deux contextes: le contexte social et le contexte physique. Le socio-constructivisme se concentrait seulement sur le contexte physique (la présence simultanée de plusieurs membres pour collaborer) pour étudier l'apprentissage. L'approche de la cognition située et distribuée donne une plus grande importance au contexte social: elle ne se concentre pas seulement sur le groupe temporaire de collaborateurs mais également sur la communauté sociale dans laquelle ces collaborateurs interagissent.

En fait, plutôt que de regarder le groupe comme un ensemble d'individus formant des systèmes cognitifs distincts, c'est maintenant le groupe lui même qui est vu comme un seul système cognitif complexe. Ce système englobe les apprenants et l'enseignant en les plaçant dans les contextes (social et physique) où les interactions se développent. A partir de ce moment, les propriétés du groupe ne peuvent plus être expliquées par une addition des propriétés distinctes des individus qui le composent.

Faisons ici une parenthèse pour noter comment l'histoire peut se répéter. Piaget avait développé sa théorie de l'épistémologie génétique et, par là, du constructivisme car il n'était pas satisfait des travaux de la psychologie expérimentale qui essayaient d'expliquer le comportement humain par une somme d'observations ponctuelles de parties individuelles de la totalité du phénomène. Dans son ouvrage "*Six Etudes de Psychologie"* (1964), il explique cela par une métaphore. Si l'on veut étudier les caractéristiques physiques de l'eau, on peut être tenté de les expliquer par les caractéristiques physiques de ses composants respectifs, l'hydrogène et l'oxygène. Or, si l'on applique ce principe, on sera surpris d'observer que l'eau à tendance à éteindre le feu alors que l'oxygène et l'hydrogène l'attisent. Il est troublant de voir que l'école du socio-constructivisme est en retour victime des mêmes critiques de la part des partisans de la cognition située et distribuée.

Si le groupe ne peut plus être regardé comme un simple ensemble d'individus, alors la collaboration ne peut plus être expliquée comme une simple interaction (au niveau inter-

individuel) entre les membres de ce groupe. La collaboration est maintenant vue comme le processus de construction et de maintenance d'une conception partagée d'un problème. Les concepts émergents sont analysés comme un produit du groupe, pas comme des schémas individuels.

Avec la cognition située et distribuée, une des questions soulevées par l'école socio-constructiviste se transforme. On ne cherche plus à savoir dans quelles conditions l'apprentissage collaboratif est efficace. La question se retrouve scindée en deux:

- quelles interactions interviennent dans quelles conditions?
- quels sont les effets de ces interactions?

La clef du problème est maintenant de trouver des variables décrivant les interactions pouvant être reliées de façon empirique et théorique avec les conditions d'apprentissage et les résultats de l'apprentissage. On ne cherche plus les effets généraux de la collaboration sur le développement individuel mais on se concentre sur des effets plus spécifiques avec une attention particulière aux structures micro-génétiques de l'interaction.

Prenons l'exemple de l'explication que nous avons citée plus haut (voir chapitre 3.3 page 18). Dans le cas d'un individu donnant une explication à un autre, il a été observé que l'individu "expliquant" bénéficiait de cette explication en terme d'apprentissage (Bargh & Schul, 1980). Un effet similaire a été observé quand un apprenant est forcé de s'expliquer un exemple à lui même (Chi et al, 1989). Il peut être tentant, dans une perspective socio-constructiviste, d'attribuer les mêmes effets à la même cause: l'auto-explication. Cependant, cette interprétation est un non sens dans la perspective de la cognition située et distribuée. En effet, cela sous-estime gravement le rôle joué par celui qui écoute l'explication donnée dans l'élaboration de l'explication elle-même. Dans ce cas, l'explication est vue comme une construction conjointe des deux partenaires qui essaient de se comprendre.

La cognition et l'apprentissage se situent donc toujours par rapport à un environnement physique et social concret. Autrement dit, les connaissances sont contextuelles et apprendre signifie s'insérer dans une communauté de pratique en exerçant des activités réelles. Parmi les avantages de cette approche appliquée de façon concrète à l'enseignement, on peut citer les suivants:

- En liant ensemble un contexte spécifique et un savoir à apprendre, les partenaires apprennent dans quelles conditions ces connaissances devraient être appliquées.
- Ces situations nourrissent la pensée créative. Les partenaires apprennent souvent comment les connaissances acquises peuvent être appliquées à de nouvelles situations.
- La mise en situation des savoirs conduit à une acquisition des connaissances de façon plus pratique.

4 Cadre de référence pour l'étude des dispositifs pédagogiques.

L'études des différentes philosophies de l'apprentissage que nous venons de présenter nous permet de faire émerger différents processus qui peuvent être mis en place pour faciliter la transmission, l'acquisition et la construction du savoir. En combinant ces différents processus, nous pouvons arriver à des scénarios d'apprentissage relativement complexe et divers. Pour aller plus loin, nous allons essayer de proposer un cadre de référence qui nous permette d'analyser le fonctionnement et les fondement de différents dispositifs utilisé à des fins pédagogiques.

Pour définir un cadre de référence, nous allons nous appuyer sur deux modèles présentés dans la littérature. Nous présenterons tout d'abord l'échelle développée par Reeves & Reeves (1997) pour l'analyse de formation en ligne. Nous passerons ensuite en revue les différentes familles de modèles d'enseignement présentées par Joyce et al. (2000) et les modèles qu'ils en tirent. Nous replacerons ensuite les différents modèles de Joyce et al. sur l'échelle de Reeves & Reeves. Cette analyse s'inspire très largement de celle présentée par Class (2001)

Enfin, pour élargir un peu ce cadre et en guise de référence, nous présenterons les situations naturelle d'enseignement définies par Paquette (2000)

4.1 Le modèle de Reeves & Reeves (1997): dix dimensions de l'enseignement interactif sur le Web

Suite à son invention, le Web a attiré l'attention de beaucoup de gens à travers le monde dont, entre autres, celle des enseignants, formateurs et pédagogues. Mais malgré cet intérêt, il y avait à l'époque peu de recherches permettant de connaître l'efficacité de l'instruction basée sur le Web (Web-based instruction - WBI). Reeves & Reeves (1997) ont développé un modèle dans le but de fournir un guide au développement, à l'implémentation et à l'évaluation des programmes pédagogiques interactifs distribués par le Web.

Les auteurs mettent en avant l'importance du scénario pédagogique dans la formation et propose un modèle comprenant dix dimensions de l'apprentissage interactif sur le Web:

1 la philosophie pédagogique,

2 la théorie d'apprentissage,

3 l'objectif d'apprentissage,

4 l'orientation de l'activité,

5 la source de motivation,

6 le rôle de l'enseignant,

7 le support méta-cognitif,

8 l'apprentissage collaboratif,

9 la sensibilité culturelle,

10 la flexibilité spatio-temporelle.

Cet ensemble de dix dimensions n'est pas exhaustif et peut, bien entendu, faire l'objet d'améliorations. Il a malgré tout le mérite de s'attaquer à une incompréhension fondamentale à savoir que ce en quoi la WBI est unique n'est pas la richesse et la diversité des médias sur lesquels elle peut s'appuyer (texte, graphiques, son, animations, vidéo...) ni la connection possible à des ressources sur la planète entière, mais les dimensions pédagogiques qu'elle peut être amenée à véhiculer.

Chacune des dimensions de ce modèle est présentée comme une quantité abstraite oscillant entre deux valeurs extrêmes très contrastées. Cette simplification dichotomique ne peut en aucun cas rendre compte de la complexité globale de l'enseignement ou de l'apprentissage. Pour ne donner qu'un exemple, nous avons déjà vu plus haut (voir chapitre 3 page 14) que les philosophies pédagogiques peuvent difficilement se résumer à deux catégories. Cependant ce ne sont pas les valeurs de chaque dimension individuelle qui sont importantes ici mais plutôt l'ensemble qu'elles forment quand on applique ce modèle à tel ou tel site Web ou activité pédagogique.

Pour chacune des dimensions proposées par Reeves & Reeves nous allons ci-dessous donner les valeurs extrêmes entre lesquelles elles varient et discuter le contraste existant entre chaque extrémité de l'échelle.

4.1.1 Philosophie pédagogique

Au risque de simplifier un peu trop le problème, l'échelle de la philosophie pédagogique oscille entre le strict instructivisme d'une part et le constructivisme d'autre part.

Chez les instructivistes, on insiste sur l'importance des objectifs d'apprentissage qui existent à part de l'apprenant. Une fois les objectifs identifiés, ils sont séquencés selon une hiérarchie de progression et chaque partie de la séquence est acquise par une instruction directe. L'apprenant est vu comme un réceptacle passif que l'on doit remplir de savoir. Le savoir (*sensu stricto*) est défini de telle sorte qu'il est séparé de la connaissance ("savoir que l'on sait"). La réalité existe en elle même. Les humains acquièrent le savoir sur cette réalité par les sens, de manière objective. Cela consiste à acquérir la vérité, et ce savoir peut être mesuré de façon précise par des tests.

Chez les constructivistes, l'accent est mis sur l'importance des intentions, de l'expérience et des stratégies cognitives de l'apprenant. Celui-ci construit des structures cognitives différentes basées sur ses connaissances antérieures et son expérience dans différents environnements d'apprentissage. Il est donc primordial que ces environnements soient les plus riches et divers que possible. L'apprenant est vu comme un individu rempli de connaissances préexistantes, d'aptitudes et de motivations. Le "savoir" n'existe nulle part hors de la pensée humaine et ce que nous savons de la réalité est construit de façon individuelle et sociale en se basant sur des expériences préalables.

Beaucoup de sites pédagogiques sur le Web sont basés sur une philosophie instructiviste (tutoriel) plutôt que constructiviste (outils). Mais il en existe de plus en plus qui fonctionnent comme des centres de ressources pour des apprenants engagés dans un processus de construction de leur propre représentation du savoir.

4.1.2 Théorie d'apprentissage.

Le design instructionel d'une activité ou d'un site devrait être basé sur de solides théories de l'apprentissage. Bien qu'elles soient nombreuses, on résume ici l'échelle à deux extrémités qui sont souvent mises en opposition: la psychologie béhavioriste et la psychologie cognitive.

La psychologie behavioriste sous-tend encore de nos jours la plupart des systèmes d'apprentissage interactif et ceux présents sur le Web ne font pas exception. Sans développer à nouveau ce que nous avons déjà présenté plus haut ((voir chapitre 3.1 page 14), l'enseignement se borne ici à faire acquérir au sujet un comportement par l'arrangement de stimulus, réponses, feedbacks et renforcements.

En contraste, la psychologie cognitive porte une plus grande attention aux états mentaux internes du sujet qu'à son comportement. Les cognitivistes affirment que pour qu'un environnement d'apprentissage soit efficace il doit mettre en oeuvre une variété de stratégies d'apprentissage telles que la mémorisation, l'instruction directe, la déduction, le "drill and practice" (exercice et entraînement), l'induction... Les stratégies à mettre en oeuvre dépendent de manière spécifique du type de connaissance qui doit être construit.

4.1.3 Objectif d'apprentissage

Les objectifs d'un apprentissage peuvent varier en précision. On peut ainsi se trouver face à un objectif très précis (par exemple savoir régler le diaphragme d'un appareil photo), alors qu'un autre peut être beaucoup plus général et de plus haut niveau (savoir composer une photographie de façon artistique).

Dans le cas d'un savoir très précis, une instruction directe par le biais d'un simple tutoriel accessible par le Web peut éventuellement suffire. Mais d'autre savoir sont si ténus, créatifs, ou de si haut niveau, qu'une acquisition inductive des connaissances sera beaucoup plus appropriée. Certaines écoles de médecine progressistes, par exemple, placent les étudiants en clinique au contact des patients tout en leur fournissant un support pédagogique pour apprendre les connaissances et aptitudes de bases, au fur et à mesure de leur besoins, cela par le biais du réseau (Brandau & Chi, 1996)

4.1.4 Orientation de l'activité

Dans les théories cognitives contemporaines ainsi que dans les théories de l'apprentissage pour les adultes, une grande place est donnée au contexte de l'apprentissage. L'orientation de l'activité pourra varier des tâches "académiques" aux

tâches “appliquées”.

Beaucoup de sites de ressources en pédagogie proposent, et se basent, sur des activités de type académique. Mais si nous considérons l’exemple d’un site destiné à l’apprentissage de la grammaire pour des adultes en difficulté par exemple, il sera peut-être plus judicieux de proposer une activité pratique appliquée se basant sur la rédaction d’une lettre de candidature pour un emploi plutôt que de proposer une activité plus académique consistant à souligner et identifier des parties de phrases dans un texte déjà écrit.

Il faut cependant garder à l’idée qu’une connaissance acquise dans un contexte particulier sera plus facilement réutilisée dans le même contexte. Il est donc important que les sites et activités soient développés en essayant de maximiser les capacités de transfert de l’apprenant.

4.1.5 Source de motivation

La motivation est un facteur important dans toute théorie ou modèle de l’apprentissage. Toute nouvelle technologie promet d’être intrinsèquement motivante, et le World Wide Web ne fait pas exception (Perelman, 1992). Mais une fois que la technologie n’est plus “nouvelle” cela reste-t-il suffisant? Peut-on simplement compter sur l’inclusion de quelques vidéos, d’animations, ou d’un gentil petit personnage dans un didacticiel pour motiver l’apprenant dans la durée?

La dimension “source de motivation” peut donc varier de extrinsèque (hors de la tâche ou de l’environnement d’apprentissage) à intrinsèque (faisant partie intégrante de la tâche ou de l’environnement). Elle doit être prise avec une aussi grande considération que les autres dimensions lors du développement d’une activité ou d’un site.

4.1.6 Rôle de l’enseignant

Les systèmes de WBI peuvent être conçus pour autoriser différents rôle de l’enseignant. On trouve à un bout de l’échelle le rôle didactique traditionnel de l’instructeur vecteur du savoir (sage on the stage). A l’autre bout de l’échelle on trouve le rôle de facilitateur de la construction des connaissances (guide on the side).

4.1.7 Support méta-cognitif

La métacognition dans l’apprentissage peut se résumer en trois point:

- la conscience qu’un apprenant a de ses objectifs d’apprentissage,

- la possibilité qu'il a de prévoir et d'évaluer ses stratégies d'apprentissage,
- sa capacité à évaluer ses progrès et à ajuster ses comportements d'apprentissage à ses besoins.

On peut donc, en bref, définir les capacités métacognitive d'un sujet comme ses capacités "d'apprendre à apprendre". La dimension du support méta-cognitif se réfère donc à la possibilité que l'on va donner de développer cette capacité (awareness). Elle peut varier de l'absence pure et simple à une intégration parfaite dans l'environnement.

On trouve aujourd'hui de plus en plus de site proposant des outils "d'awareness" permettant de savoir, par exemple, le nombre de fois que l'on a fait appel à l'aide du dispositif ou de connaître sa position relative sur un cheminement prévu de formation

4.1.8 Apprentissage collaboratif

La dimension de l'apprentissage collaboratif se réfère à l'importance donnée aux méthodes d'apprentissage dans lesquelles les apprenants travaillent ensemble en paire ou en petits groupes pour accomplir des objectifs partagés. Le support des applications peut aller d'une absence totale (comme dans un didacticiel pur par exemple) à l'inclusion de cette dimension comme partie intégrante du processus d'apprentissage (exclusif)

On peut rappeler ici l'importance de l'interaction sociale dans les théories générales de l'éducation déjà mentionnées plus haut (voir chapitre 3.3 page 18) (voir chapitre 3.4 page 20). L'intégration du support collaboratif doit donc être considérée avec une attention particulière.

4.1.9 Sensibilité culturelle

Tous les systèmes d'apprentissage ont des implications culturelles. Bien qu'il soit souvent difficile de concevoir un système qui puisse s'adapter à toute culture (en prenant en compte les différents tabous, la signification de telle ou telle couleur, etc.), il semble une évidence qu'un site, une activité, une application, devraient au minimum prendre en compte les différences ethniques et culturelles des divers apprenants qui seront amenés à s'en servir.

4.1.10 Flexibilité spacio-temporelle

La dimension de la flexibilité spacio-temporelle se réfère à la possibilité ou non de participer aux évènements d'apprentissage quel que soit le lieu et le moment. Cette

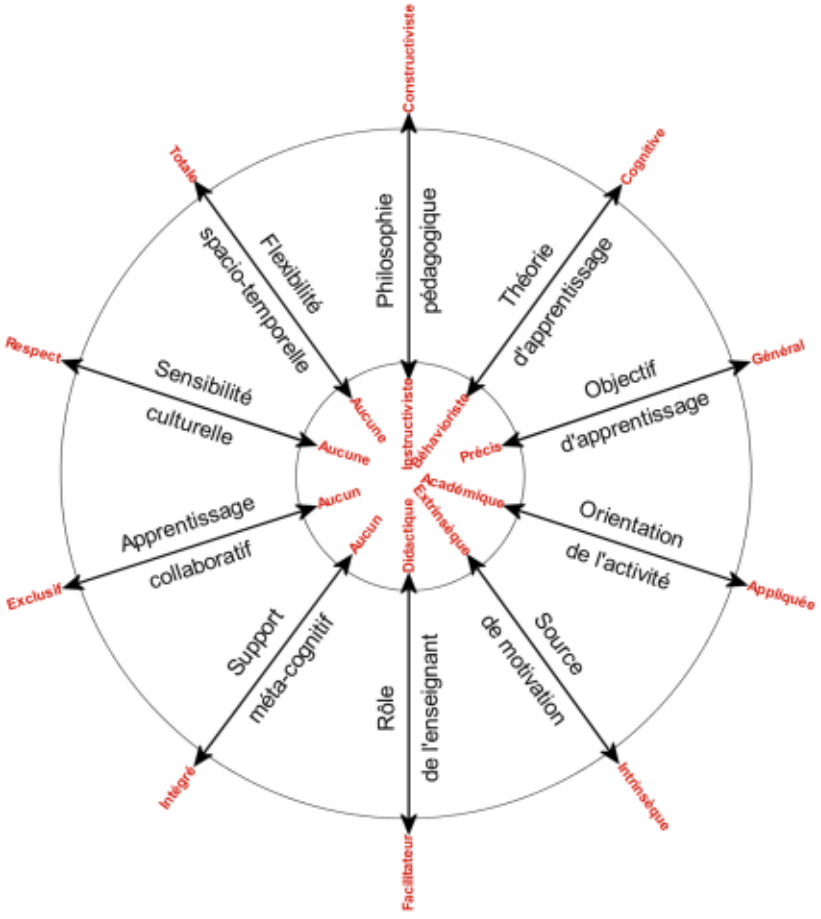
flexibilité peut varier d'aucune (les participants doivent accomplir la même tâche au même moment dans le même lieu) à totale (les activités et les documents peuvent être consultés depuis n'importe quel endroit et à n'importe quel moment).

Le Web est bien évidemment un vecteur de choix pour permettre de sortir du contexte fixe traditionnel de l'instruction académique.

4.1.11 Mise en oeuvre du modèle

Les auteurs mettent le modèle en oeuvre dans leur présentation en superposant sur plusieurs couches les axes représentant les différentes dimensions et en pointant sur chacun d'eux la valeur qui y correspond. Nous avons décidé de modifier légèrement cette représentation. Dans la suite de ce document nous utiliserons chaque fois qu'il est fait référence à ce modèle une représentation en "roue" (qui ne doit en aucun cas être confondue avec un diagramme polaire). La figure 5 montre cette représentation.

Fig 5.Représentation en "roue" de l'échelle de Reeves & Reeves.



Nous avons introduit cette modification car il nous semble que la représentation qui en résulte est plus claire, qu'elle permet de voir toutes les dimensions d'un seul coup d'oeil et surtout qu'elle ne privilégie pas une dimension par rapport à une autre (comme pourrait le faire supposer la fausse hiérarchie introduite par la superposition dans la représentation utilisée par les auteurs).

4.2 Présentation des modèles de l'enseignement selon Joyce et al. et discussion par l'application de l'échelle de Reeves & Reeves.

Joyce et al. (2000) distinguent treize modèles génériques de l'enseignement qu'ils classent en quatre grandes familles distinctes.

- La famille ***Socialisation***
- La famille ***Traitement de l'Information***
- La famille ***Individualité*** (ou ***Personnalité***)
- La famille ***Systèmes Béhavioristes***

Nous allons commencer par présenter chacune des familles de manière générale. Nous exposerons ensuite brièvement chacun des modèles qui composent la famille en nous basant sur la fiche récapitulative fournie par les auteurs dans leur ouvrage. Enfin, nous tenterons de placer ces modèles sur l'échelle de Reeves & Reeves et d'interpréter le résultat pour chacune de ces familles.

La présence d'une fiche de description pour chaque modèle de chaque famille pourra paraître un peu lourde et rébarbative au lecteur averti. Nous avons cependant souhaité les laisser à l'intérieur de ce texte car elle donne à nos yeux de bons exemples et permettent de d'appréhender la richesse et la complexité de certaines situations d'enseignement. Nous ne saurions trop conseiller aux personnes qui trouveraient cette lecture ennuyeuse de se concentrer dans un premier temps sur l'analyse qui est faite de la famille générale à la fin de chaque chapitre et de revenir sur les fiches descriptives en cas de besoin.

4.2.1 La famille Socialisation

L'objectif global des modèles de la famille socialisation est de construire des communautés d'apprenants en exploitant la synergie produite par l'interaction entre les apprenants.

La philosophie repose sur le fait que le développement d'une culture d'apprentissage passe d'une part par l'établissement d'une interaction intégrative et productive entre les apprenants ainsi que par l'élaboration de normes supportant des activités d'apprentissage sérieuses.

Cette famille comprend trois modèles:

- Partenariat d'apprentissage
- Jeu de rôle
- Enquête jurisprudentielle

Le modèle "partenariat d'apprentissage"

- ***Objectif spécifique:***

  Le but de ce modèle d'enseignement est d'organiser le groupe de façon sociale pour permettre la collaboration entre ses membres.

- ***Les différentes phases (syntaxe):***

1 Se trouver dans une situation (prévue ou non) stimulant fortement la curiosité et la motivation des membres du groupes.

2 Explorer les réactions des apprenants face à cette situation.

3 Formuler la tâche à accomplir et l'organiser: définition du problème, définition des rôles des apprenants, des différentes actions à entreprendre, etc.

4 Apprentissage individuel et en groupe.

5 Analyser l'avancement et les stratégies des apprenants.

6 Conclusion et réflexions sur l'activité d'apprentissage.

- ***Système social:***

  Le système est basé sur le processus démocratique et les décisions de groupe, avec peu de structuration externe. Il doit y avoir une véritable motivation, celle-ci ne peut pas être imposée. Les échanges spontanés et authentiques sont essentiels. L'atmosphère dominante est soit le raisonnement, soit la négociation.

- ***Principe de réaction***

  L'enseignant joue un rôle de facilitateur en portant une attention particulière aux phénomènes de groupes et aux exigences de l'enquête: il aide les apprenants à formuler leur plan, à agir, il gère le groupe. Les apprenants choisissent le type d'information nécessaire à l'approche de leur problème, effectuent leurs recherches, formulent des hypothèses, les testent, continuent leur enquête ou en commence une nouvelle. Le groupe variant constamment, l'enseignant ne peut pas réagir de façon mécanique et doit constamment "lire" le comportement social et académique des apprenants afin de pouvoir garder une dynamique.

- ***Support matériel:***

  L'environnement doit pouvoir répondre à une grande variété de demande de la part des apprenants. L'enseignant et les étudiants doivent pourvoir assembler ce dont ils ont besoin au moment où ils en ont besoin.

- ***Exemples d'utilisation:***

  Ce modèle est utilisable dans toutes les disciplines. Il permet plus particulièrement d'accentuer la formulation et la résolution de problèmes plutôt que l'acquisition d'une information préalablement structurée.

Le modèle "jeu de rôle"

- ***Objectif spécifique:***

  Le jeu de rôle permet à ses acteurs de prendre conscience de leur rôle social dans l'interaction avec leurs pairs. Il permet d'analyser ses propres valeurs et comportements, de développer une stratégie pour régler des problèmes interpersonnels (et personnels) et de développer l'empathie envers les autres. Les changements de rôles à l'intérieur d'un même scénario sont particulièrement efficaces.

- ***Les différentes phases (syntaxe):***

  1 Mettre le groupe dans l'ambiance (présenter et expliciter le problème, expliquer en quoi consiste le jeu de rôle).

2 Sélectionner les participants (analyse des rôles, sélection des acteurs).

3 Préparer la scène (donner les lignes d'action, ré-expliquer les rôles, entrer dans la situation du problème).

4 Préparer les observateurs (décider de ce qu'il y a à observer et attribuer les tâches d'observation).

5 Activation du jeu (commencer, maintenir et terminer le jeu).

6 Discussion et évaluation (analyser les différents rôles joués, discuter du problème central, préparer la prochaine mise en scène).

7 Re-activation du jeu (redistribuer les rôles et suggestions des étapes suivantes ou des attitudes à avoir).

8 Discussion et évaluation (comme à la phase 6).

9 Partage de l'expérience et élargissement (mettre en relation le problème avec une situation réelle et plus générale, explorer les principes généraux du comportement).

- ***Système social:***

  Ce modèle est assez peu structuré. L'enseignant est chargé d'initier les phases et de guider les apprenants dans les différentes activités de chaque phase. Le contenu particulier de la discussion est décidé en grande partie par les acteurs du jeu de rôle.

- ***Principe de réaction***

  ➠ Accepter toutes les réactions sans évaluation préalable.

  ➠ Aider les apprenants à explorer les différentes facettes du problème et à comparer des points de vues alternatifs.

  ➠ Faire prendre conscience aux apprenants de leurs propres points de vues et leurs émotions en leur retournant leurs réponses, en paraphrasant et en résumant leurs discussions.

  ➠ Utiliser le concept de rôle et insister sur le fait qu'il existe différentes manières de jouer un rôle.

  ➠ Mettre en avant qu'il existe plusieurs façons de résoudre un problème particulier.

- ***Support matériel:***

  Rien de particulier si ce n'est l'objet du jeu de rôle initial

- ***Exemples d'utilisation:***

  Ce modèle est particulièrement adapté aux thématiques sociales (développement, éthique, communauté, etc.). A utiliser pour le développement de:

  1 l'analyse de valeurs et de comportements personnels.

  2 les stratégies propres à la résolution de problème.

  3 l'empathie

Le modèle "enquête jurisprudentielle"

- ***Objectif spécifique:***

Spécialement adaptée dans les études sociales, ce modèle est basé sur l'étude de cas et a pour objectif de poser des questions sur les sujets soulevés par l'étude.

- ***Les différentes phases (syntaxe):***

  1 Présentation du cas à étudier (présentation du matériel et passage en revue des différents faits).

  2 Identification des questions (synthétiser les faits dans une perspective de politiques publiques, sélectionner un des axes politiques, identifier les valeurs et les conflits de valeurs, reconnaître les question factuelles et les problèmes de définitions qui sous-tendent le problème).

  3 Prise de position (argumenter la prise de position en fonction de valeurs sociales ou de conséquences de la décision)

  4 Schéma d'argumentation (établir le point de rupture à partir duquel le système de valeurs n'est plus respecté, montrer les conséquences désirables et indésirables d'une position, clarifier les conflits de valeurs avec des analogies, donner les priorités d'une valeur sur les autres).

  5 Parachèvement et qualification des positions (expliciter les positions et les raisons qui ont poussé à les prendre, examiner certaines situations similaires, qualification des positions).

  6 Tester les hypothèses factuelles qui découlent des positions adoptées (identifier les hypothèses factuelles et déterminer leur adéquation, déterminer des conséquences prévues et examiner leur validité factuelle).

- ***Système social:***

  Ce modèle possède une structure modérée à forte, l'enseignant dirigeant et contrôlant la discussion. Cependant, c'est une atmosphère d'équité intellectuelle et d'ouverture qui doit dominer.

- ***Principe de réaction***

  Maintien d'un climat intellectuel vigoureux et respectant les différents points de vue. Il faut éviter l'évaluation directe des opinions des étudiants. Maintien de la dialectique (dialogue de confrontation, interrogation des suppositions des apprenants et raisonnement par analogie).

- ***Support matériel:***

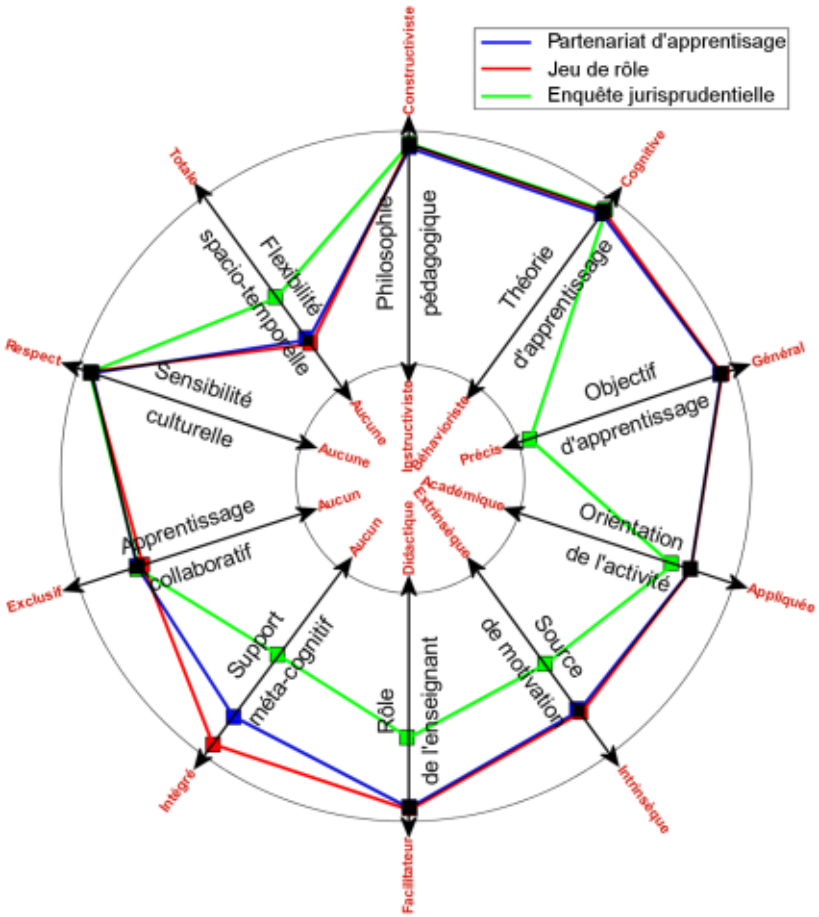
  Documents de base centrés sur la situation du problème à étudier.

- ***Exemples d'utilisation:***

  Modèle utilisable dans tous les domaines. Utilisation préconisée: maîtrise d'une structure d'analyse de cas (identifications des points clés, application de valeurs, utilisation analogique, proposition de solutions)

Application du modèle de Reeves & Reeves

Fig 6.Echelle de Reeves & Reeves pour les modèles de la famille "Socialisation"



Comme on peut s'en douter de par le nom de cette famille, une importance particulière est donnée à l'interaction sociale. C'est donc cette caractéristique qui va influencer la plupart des dimensions de l'échelle de Reeves & Reeves (figure 6).

- L'apprentissage collaboratif joue un rôle assez important, les participants étant amenés à échanger leurs points de vue.
- La réflexion ultérieure sur l'activité elle même et l'exploration des réaction offrent un bon support méta-cognitif.
- La motivation tend vers l'intrinsèque car elle est surtout suscitée par le climat intellectuel régnant au sein du groupe.
- La sensibilité culturelle est totalement respectée puisque chaque personne doit pourvoir trouver sa place au sein de l'équipe d'apprenants.
- L'orientation de l'activité est plutôt appliquée en se basant sur le concret de la situation rencontrée.
- La flexibilité spacio-temporelle est plutôt faible car les apprenants ont besoin de se retrouver ensemble pour interagir. On pourrait cependant imaginer de tels scénarios utilisant des technologies de communications asynchrone. Une certaine coordination est malgré tout indispensable.

- La philosophie pédagogique est constructiviste et la théorie d'apprentissage est cognitive car on laisse les points de vue se développer au sein du groupe. Il n'y a à priori rien de vrai ou de faux, ce qui importe c'est la participation des apprenants et leur réflexion.

Cependant, le modèle "enquête jurisprudentielle" se distingue des deux autres: son objectif d'apprentissage est très précis et l'enseignant y joue un rôle plutôt didactique en menant la discussion.

4.2.2 Famille traitement de l'information

L'objectif global des modèles de cette famille est de développer l'intellect, que ce soit par l'apprentissage de la recherche d'information, l'apprentissage de la conceptualisation, la démarche qui vise à poser des hypothèses et à les tester, ou la pensée créative.

La philosophie qui sous-tend cette famille est l'étude du monde et de la société par le développement de l'argumentation et de la pensée critique.

Cette famille comprend six modèles:

- Pensée inductive
- Acquisition de concepts
- Enquête scientifique
- Mémorisation
- Synectics (cf. racine grecques "continuité")
- Apprentissage par présentation

Modèle "pensée inductive"

- ***Objectif spécifique:***

  Apprendre à trouver et à organiser l'information d'une part et apprendre à créer et à tester des hypothèses reflétant les relations entre les données d'autre part.

- ***Les différentes phases (syntaxe):***

  - Stratégie 1: "formation de concepts"

    1 énumération: différenciation- identification d'items différents

    2 groupement: identification de propriétés communes- abstraction

    3 labélisation/ catégorisation: super et sub-catégorisation

  - Stratégie 2: "interprétation de données"

    1 identification des relations critiques: différenciation

    2 exploration de ces relations: mettre en relation des catégories- détermination de relations de cause à effet

    3 inférences: aller au delà de ce qui est donné- extrapoler

  - Stratégie 3: "application de principes"

1 prédiction des conséquences, explication de phénomènes non familiers, hypothèses: analyse de la nature du problème, retrouver l'information pertinente

2 explication et/ou renforcement des prédictions ou des hypothèses: détermination des liens de causalité amenant aux hypothèses

3 vérification des prédictions: utilisation de principes logiques pour déterminer des conditions suffisantes

- ***Système social:***

  Ce modèle a une structure élevée à modérée. Il se base sur le mode coopératif mais l'enseignant initie et contrôle les activités.

- ***Principe de réaction***

  L'enseignant ajuste les tâches en fonction du niveau cognitif des apprenants.

- ***Support matériel:***

  Fournir aux apprenants des données brutes à analyser.

- ***Exemples d'utilisation:***

  Modèle utilisable dans tous les domaines mais plus particulièrement dans l'apprentissage de la formation de concepts et dans l'enseignement des concepts. Effets metacognitifs: attention donnée au langage, à la nature du savoir et à la logique.

Modèle "acquisition de concept"

- ***Objectif spécifique:***

  Aide à la conceptualisation par la présentation d'une information organisée à différents stades de développement.

- ***Les différentes phases (syntaxe):***

1 Présentation de données et identification de concepts

a l'enseignant présente des catégories exemples labélisés, les apprenants comparent les attributs des différents exemples positifs et négatifs.

b Les apprenants génèrent et testent les hypothèses.

c Les apprenants établissent une définition en fonction des attributs principaux.

2 Test de l'acquisition de concept

a les apprenants testent d'autres exemples non labélisés en répondant par oui ou non.

b l'enseignant confirme les hypothèses, nomme les concepts et réoriente les définitions si nécessaire.

c les apprenants génèrent des exemples.

3 Analyse des stratégies cognitives

a les apprenants décrivent les pensées qu'ils ont eues.

b Les apprenants discutent le rôle des hypothèses et des attributs.

c Les apprenants discutent le type et le nombre d'hypothèses.

- ***Système social:***

  Ce modèle a une structure modérée. L'enseignant contrôle la séquence mais un dialogue ouvert s'installe dans la dernière phase. L'interaction entre les apprenants est encouragée et ceux-ci sont poussés à prendre des initiatives au fur et à mesure que leur expérience s'accroit.

- ***Principe de réaction***

  1 Aider les apprenants et mettre l'accent sur le fait que la discussion est basée sur des hypothèses

  2 Aider les apprenants à comparer les hypothèses

  3 Mettre l'accent sur les traits spécifiques de certains exemples

  4 Assister les apprenants dans la discussion et l'évaluation de leurs stratégies cognitives

- ***Support matériel:***

  Le matériel de base doit être organisé et choisi avec précaution, notamment les exemples

- ***Exemples d'utilisation:***

  Ce modèle est particulièrement préconisée dans l'élaboration de stratégies pour la construction de concepts et pour apprendre à tolérer un seuil d'ambiguïté.

Modèle "enquête scientifique"

- ***Objectif spécifique:***

  Par immersion dans le processus scientifique, l'apprenant doit trouver l'information, vérifier les hypothèses et théories et réfléchir sur les mécanismes de construction du savoir.

- ***Les différentes phases (syntaxe):***

  1 Confrontation avec le problème (explication des procédures d'enquête, présentation des événements déroutants)

  2 Collecte de données - Vérification (vérifier la nature des objets et des conditions, vérifier l'occurrence de la situation du problème)

  3 Collecte de données - Expérimentation (isolement de variables faisant du sens, faire des hypothèses et tester les relations causales)

  4 Organisation, formulation et explication (formulation de règles ou d'explications)

  5 Analyse du processus d'enquête (analyse de la stratégie employée et en développer de plus efficaces ou mieux adaptées)

- ***Système social:***

  L'enseignant peut donner les différentes étapes à suivre pour mener l'enquête mais il est préférable de les guider: laisser les apprenants échanger et découvrir les étapes par eux-mêmes.

- ***Principe de réaction***

  S'assurer que les apprenants formulent bien leurs questions (de manière à pouvoir y répondre par oui ou non) et encourager l'interaction entre étudiants sans évaluer les différentes théories émergentes.

- ***Support matériel:***

  Un set de matériel servant à la confrontation. Un enseignant comprenant les processus intellectuels et les stratégies d'enquête.

- ***Exemples d'utilisation:***

  Modèle utilisable dans les domaines scientifiques.

  Utilisation préconisée: capacité d'observation, collecte et organisation de données, identification et contrôle des variables, élaboration et test d'hypothèses, formulation d'explication et établissement d'inférences.

Modèle "mémorisation"

- ***Objectif spécifique:***

  Stratégies d'aide à la mémorisation et à l'assimilation d'information.

- ***Les différentes phases (syntaxe):***

  1 S'occuper du matériel (utilisation de techniques visant à souligner, réfléchir et lister les points clés du matériel)

  2 Développer des relations/connexions (rendre le matériel familier et développer des relations par l'utilisation de mots clés, de paraphrases)

  3 Association d'images sensorielles (utilisation de techniques d'exagération ou de ridiculisation puis recentrage sur l'image à faire passer)

  4 mise en pratique de la mémorisation jusqu'à l'acquisition du matériel

- ***Système social:***

  Système coopératif entre l'enseignant et l'apprenant.

- ***Principe de réaction***

  L'enseignant aide l'apprenant à identifier les mots clés, les images, etc. Il offre des suggestions basées sur le cadre de référence de l'apprenant.

- ***Support matériel:***

  Tout matériel audio-visuel permettant de faciliter la mémorisation par association.

- ***Exemples d'utilisation:***

  Modèle utilisable dans tous les domaines.

  Utilisation préconisée: développement de la capacité à intégrer et à retrouver de l'information. Développement de la capacité à penser par associationnisme (images, mots, son, etc.).

Modèle "synectics"

- ***Objectif spécifique:***

Un des objectifs est d'aider à jeter un nouveau regard sur quelque chose de familier. L'autre objectif est d'aider à intégrer et à rendre familier quelque chose de nouveau.

- ***Les différentes phases (syntaxe):***

  ➠ Les différentes phases de la stratégie qui privilégie l'objectif 1:

  1 Description de la condition présente (les apprenants décrivent la situation telle qu'ils la perçoivent)

  2 Analogie directe (les apprenants suggèrent des analogies, en choisissent une et la décrivent de manière plus approfondie)

  3 Analogie personnelle (les apprenants se ré-approprient l'analogie choisie précédemment)

  4 Conflits implicites (les apprenants reprennent les descriptions des phases 2 et 3, suggèrent plusieurs conflits implicites et en choisissent un.

  5 Analogie directe (les apprenants choisissent une autre analogie basée sur le conflit implicite)

  6 Réexamen de la tâche originale (l'enseignant ramène les apprenants à la tâche du début et utilise tout ou partie du processus de continuité)

  ➠ Les différentes phases de la stratégie qui privilégie l'objectif 2:

  1 Introduction de la matière nouvelle (l'enseignant apporte l'information sur le nouveau thème)

  2 Analogie directe (l'enseignant suggère des analogies et demande aux apprenants de les décrire)

  3 Personnalisation de l'analogie (les apprenants “deviennent” l'analogie)

  4 Comparaison des analogies (les apprenants identifient et expliquent les points de similarité entre la matière nouvelle et l'objet de leur analogie)

  5 Explication des différences (les apprenants expliquent dans quels cas l'analogie employée ne peut fonctionner)

  6 Exploration (les apprenants réexaminent la matière nouvelle pour elle-même en se débarrassant des analogies)

- ***Système social:***

  L'enseignant a pour rôle d'initier les phases

- ***Principe de réaction***

  Encourager l'ouverture d'esprit, l'irrationnel et l'expression créative.

- ***Support matériel:***

  Matériel de réflexion.

- ***Exemples d’utilisation:***

  Modèle utilisable dans tous les domaines.

  Utilisation préconisée: stimuler la créativité individuelle et celle du groupe.

Modèle “apprentissage par présentation”

- ***Objectif spécifique:***

  Apporter à l'apprenant une structure cognitive qui l'aidera à comprendre le matériel donné par l'enseignant.

- ***Les différentes phases (syntaxe):***

  1 Présentation du méta-modèle (clarification des buts de la leçon, présentation de la matière avec identification des attributs, passage à des exemples dans les cas appropriés, contextualisation, répétition, accentuation de l'awareness des apprenants à partir de leurs connaissances et expériences)

  2 Présentation du matériel à apprendre (présentation du matériel, explicitation de l'ordre de présentation de la matière, établissement du lien entre la matière et le méta-modèle)

  3 Renforcement de l'organisation cognitive (utiliser des principes à réconciliation intégrative, provoquer l'approche critique face au thème, clarifier les idées, appliquer activement les idées)

- ***Système social:***

  Modèle hautement structuré et guidé par l'enseignant. Il faut néanmoins de la collaboration entre apprenants et enseignant.

- ***Principe de réaction***

  Négociation du sens, établir les rapports entre le méta-modèle et la matière.

- ***Support matériel:***

  Il faut beaucoup de données et un matériel très bien organisé.

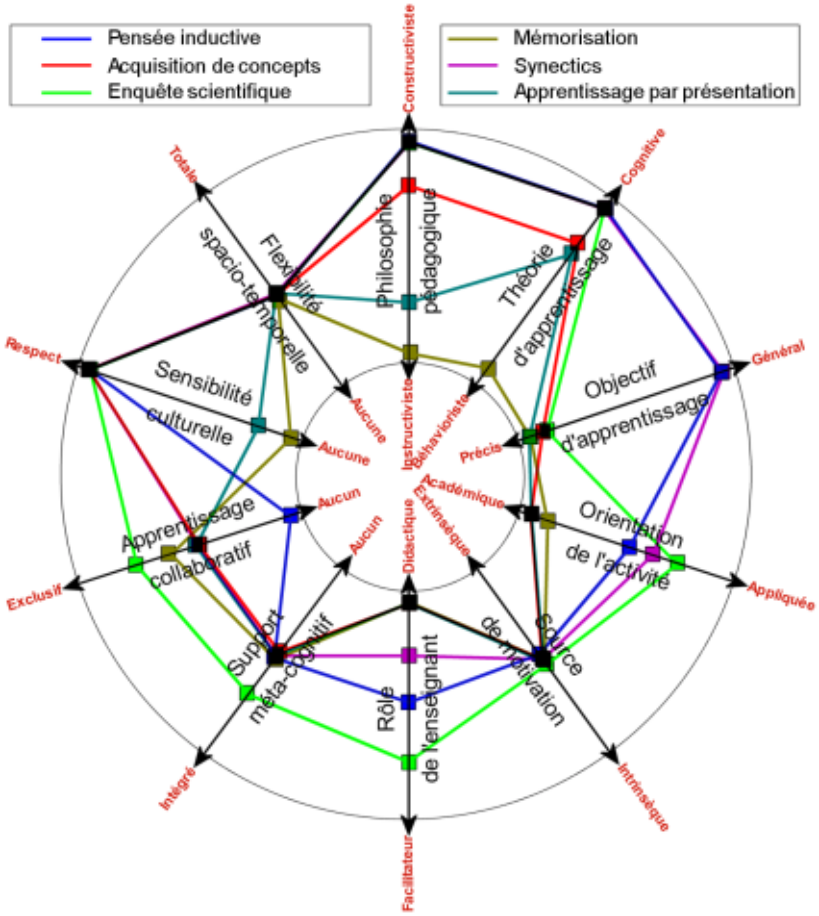
- ***Exemples d’utilisation:***

  Modèle utilisable dans tous les domaines.

  Utilisation préconisée: valeur ajoutée de la dimension instructionnaliste entre méta-cognition et information à intégrer.

Application du modèle de Reeves & Reeves

Fig 7.Echelle de Reeves & Reeves pour les modèles de la famille "traitement de l'information"



Les modèles de cette famille sont très dispersés sur l'échelle de Reeves & Reeves (figure 7) ce qui rend la mise en évidence de caractéristiques communes assez difficile. On peut cependant faire quelques remarques:

- La philosophie d'enseignement dominante est le constructivisme sauf pour les deux modèles "apprentissage par présentation" et "mémorisation" ou elle tend vers l'instructivisme. Ces deux modèles se distinguent encore des autres en étant moins respectueux des différences culturelles.
- Tous les modèles tendent vers une théorie d'apprentissage cognitive à l'exception du modèle "mémorisation" qui, lui, se base sur une théorie totalement béhavioriste.
- Bien que d'une importance relativement faible, l'apprentissage collaboratif est utilisé.
- L'objectif d'apprentissage varie d'un extrême à l'autre (précis - général) ainsi que l'orientation de l'activité (académique - appliquée).
- L'enseignant peut être amené à jouer des rôles très différents

- Les dimensions “source de motivation”, “support méta-cognitif” et “flexibilité spacio-temporelle” ne sont pas significatives et peuvent se situer n’importe où sur leurs échelles respectives selon la matière étudiée et l’implémentation concrète de l’enseignement.

4.2.3 Famille individualité (ou personnalité)

Dans la famille individualité, l'objectif global est de développer, à partir des particularités de l'individu, une méta-analyse pour qu’il puisse mieux comprendre ses propres modes de fonctionnement, et de modeler l'enseignement-apprentissage en fonction de ceux-ci.

La philosophie est de mieux se connaître pour apprendre plus efficacement et mieux gérer ses responsabilités.

Un seul modèle est issu de cette famille:

➠ Apprentissage non directif

Modèle “apprentissage non directif”

- ***Objectif spécifique:***

  L'enseignant est au service de l'apprenant. Ce modèle est souvent utilisé en combinaison avec d'autres pour s'assurer que l'objectif d'apprentissage est atteint.

- ***Les différentes phases (syntaxe):***

  1 Définition de la situation (l'enseignant encourage l'expression libre des sentiments)

  2 Exploration du problème (l'apprenant est encouragé à définir le problème, l'enseignant clarifie les sentiments)

  3 Développement de la phase auto-réflexive (l'apprenant discute le problème, l'enseignant l'aide)

  4 Planification et prise de décision (l'apprenant planifie une prise de décision, l'enseignant clarifie les décisions possibles)

  5 Intégration (l'apprenant augmente sa vision réflexive et développe davantage d'actions positives, l'enseignant l'aide et l’entraîne)

- ***Système social:***

  L'apprenant initie et l'enseignant facilite autour de la discussion du problème.

- ***Principe de réaction***

  L'enseignant adopte un comportement empathique face à l'apprenant tout en prenant de la distance.

- ***Support matériel:***

  Nécessité de contact individualisé entre enseignant et apprenant.

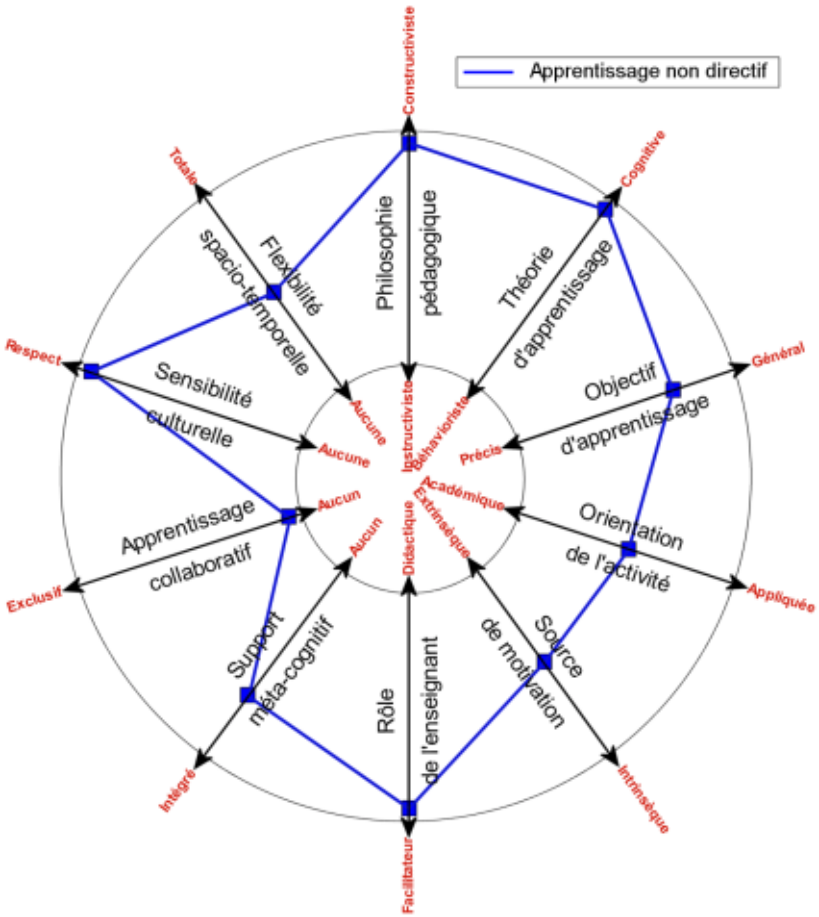
- ***Exemples d’utilisation:***

  Modèle utilisable dans tous les domaines.

  Utilisation préconisée: aide à l'approfondissement d'un thème.

Application du modèle de Reeves & Reeves

Fig 8.Echelle de Reeves & Reeves pour le modèle de la famille "individualité"



Il est difficile de faire une discussion sur une famille comprenant un seul modèle. On peut cependant mettre en évidence une caractéristique qui amène à faire une remarque importante sur l'échelle elle même.

L'apprentissage collaboratif est marqué comme inexistant (figure 8). Bien qu'à la lumière de la cognition distribuée (voir chapitre 3.4 page 20) on puisse objecter qu'un dialogue enseignant-apprenant constitue une certaine forme de collaboration, nous nous tenons à la définition stricte de cette dimension donnée par Reeves & Reeves (voir chapitre 4.1.8 page 26) qui ne concerne que le travail entre les apprenants eux mêmes.

L'importance donnée à la collaboration enseignant-apprenant se mesure d'après nous sur cette échelle plutôt par la dimension "rôle de l'enseignant": plus il joue un rôle de facilitateur, plus un dialogue ouvert peu s'installer et plus les deux partenaires apprenants-enseignants peuvent mutuellement bénéficier de cette interaction.

4.2.4 Famille systèmes béhavioristes

Dans la famille systèmes béhavioristes, l'objectif est de travailler sur la réaction de l'être humain face à une tâche.

La philosophie de cette famille se base sur le fait que l'être humain est un système de communication auto-correcteur qui modifie son comportement en fonction de

l'information qu'il reçoit lors de la réalisation d'une tâche.

Cette famille comprend trois modèles

- Appropriation des connaissances (mastery learning et instruction programmée)
- Instruction directe
- Apprentissage par simulation

Modèle “appropriation des connaissances”

- ***Objectif spécifique:***

  la matière est présentée de façon modulaire et les apprenants, travaillant individuellement, doivent l'acquérir.

- ***Les différentes phases (syntaxe):***

  1 Faire en sorte que chaque apprenant puisse traverser à son rythme les modules organisés en séquence d'apprentissage

  2 Développer dans chaque apprenant un degré de maîtrise de la matière

  3 Développer l'initiative et la régulation de l'auto-apprentissage

  4 Encourager et ancrer le développement de la résolution de problème par l'intermédiaire de processus

  5 Encourager l'auto-évaluation et la motivation à apprendre

- ***Système social:***

  L'enseignant a pour rôle d'accompagner l'apprenant et de développer les différentes caractéristiques.

- ***Principe de réaction***

  Tutorat, coaching, relation individualisée entre enseignant et apprenant pour le mettre en confiance.

- ***Support matériel:***

  Matériel d'enseignement très structuré.

- ***Exemples d’utilisation:***

  Modèle utilisable dans tous les domaines.

  Utilisation préconisée: matériel d'auto-apprentissage à compléter par des activités présentielles, des séances de Q&A, etc.

Modèle “instruction directe”

- ***Objectif spécifique:***

  L'enseignant est là pour donner des feedbacks à l'apprenant dans sa progression d'apprentissage et pour le réguler.

- ***Les différentes phases (syntaxe):***

  1 Orientation (établissement d'un cadre de travail pour le cours:

- présentation des objectifs et niveaux de compétence attendus
- description du contenu du cours et établissement de relations avec des connaissances antérieures
- explication des différentes parties du cours et différents rôles/responsabilités des apprenants

2 Présentation (explication des nouveaux concepts et/ou compétences)

3 Pratique structurée (tous les apprenants participent, l'enseignant montre des exemples et suscite la participation des apprenants)

4 Pratique guidée (les apprenants travaillent sans l'aide de l'enseignant mais peuvent recourir à lui en cas de nécessité)

5 Pratique indépendante (les apprenants travaillent sans l'aide de l'enseignant, ils reçoivent un feed-back ultérieurement, le but est de développer la maîtrise parfaite du cours)

Il est important de vérifier que les apprenants disposent des prérequis nécessaires afin que les différents niveaux de pratiques puissent atteindre leurs objectifs.

- ***Système social:***

  L'enseignant accompagne l'apprentissage et contribue au renforcement des connaissances. La motivation de l'apprenant est générée par l'avancée pas à pas.

- ***Principe de réaction***

  Rôle de stimulateur et d'instructeur.

- ***Support matériel:***

  Matériel structuré

- ***Exemples d'utilisation:***

  Modèle utilisable dans tous les domaines.

  Utilisation préconisée: dans l'acquisition d'information et de compétences de base.

Modèle "apprentissage par simulation"

- ***Objectif spécifique:***

  Deux pratiques sont à noter. La première issue de la tendance cybernétique, vise un apprentissage par cheminement de la théorie à la pratique. La deuxième constitue une vraie simulation basée sur une situation de la vie réelle.

- ***Les différentes phases (syntaxe):***

1 Orientation (présentation du thème général de la simulation et des concepts abordés, explication de la simulation et du jeu, donner un aperçu de la simulation)

2 Entraînement des participants (mise en place du scénario avec spécification des règles, des rôles, des procédures, des performances et de l'évaluation, du type de décision attendu et des objectifs, distribution des rôles, tenue d'une séance abrégée de répétition)

3 Simulation des opérations (conduire l'activité, feed-back et évaluation des performances et des décisions prises, clarification des incompris, poursuite de la simulation)

4 Debriefing (résumé des événements et des perceptions, résumé des difficultés et de la réflexivité, analyse des processus, comparaison de la simulation avec le monde réel, mise en rapport de la simulation avec le contenu de cours, évaluation et redesign de la simulation)

- ***Système social:***

  L'enseignant dirige la simulation.

- ***Principe de réaction***

  Rôle de support de la part de l'enseignant et d'accompagnateur.

- ***Support matériel:***

  Matériel de base très structuré.

- ***Exemples d'utilisation:***

  En ce qui concerne la première pratique, ce modèle est utilisable dans tous les domaines. On le préconise en particulier pour le développement de la pensée critique, de la prise de décision, de l'efficacité, de l'entrée en action.

  En ce qui concerne la deuxième pratique, ce modèle n'est utilisable que dans un domaine très précis: les activités à hauts risques dans la vie réelle comme, par exemple, piloter un avion.

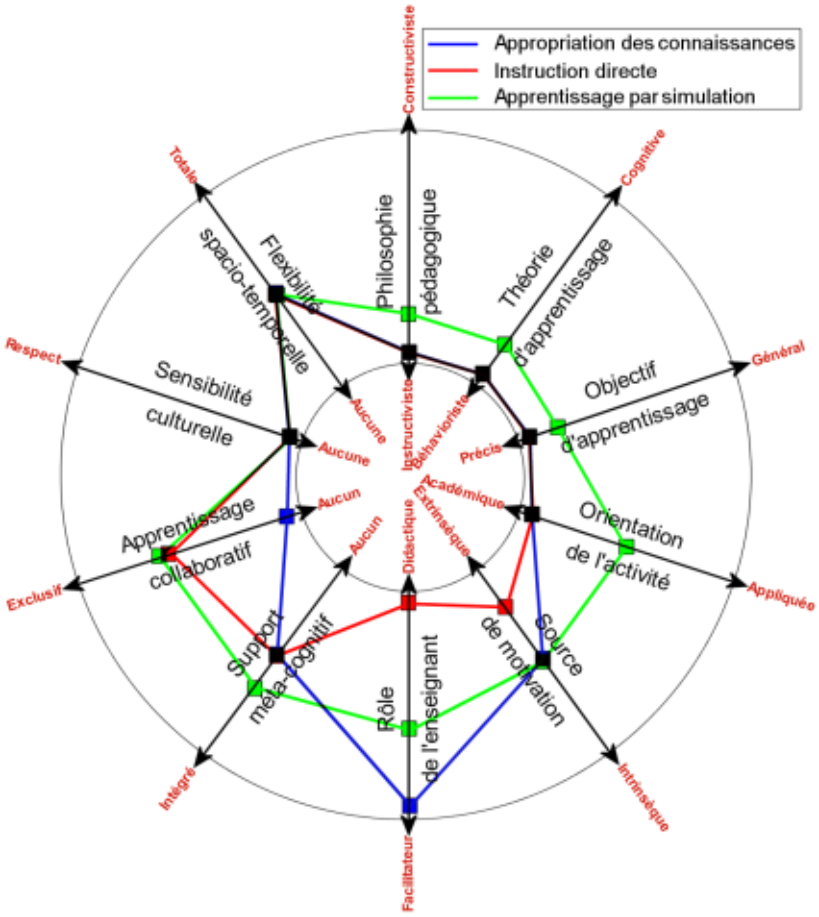
Application du modèle de Reeves & Reeves

Comme dans le cas de la famille socialisation, on ne sera pas étonné ici que la dénomination "béhavioriste" influence un grand nombre de dimensions de l'échelle et cela pour chacun des modèles de la famille. En effet (figure 9) tous les modèles se basent sur une philosophie instructiviste, une théorie d'apprentissage béhavioriste. L'objectif d'apprentissage est relativement précis avec une tâche orientée plutôt vers l'académique.

La sensibilité culturelle est peu respecté car le matériel est le même pour tout le monde et la progression très linéaire. Les dimensions "support méta-cognitif", "apprentissage collaboratif" et "flexibilité spacio-temporelle" ont peu d'influence et ne sont pas significatives.

En revanche, on peut être surpris de la dispersion du rôle de l'enseignant dans des modèles autant basés sur la transmission du savoir. En effet, si la transmission s'opère par le biais d'un matériel particulièrement bien préparé ou par l'intermédiaire d'un ordinateur, comme dans le modèle "appropriation des connaissances", l'enseignant peut alors prendre du recul et se trouver dans un rôle très facilitateur en aidant l'apprenant à clarifier les points qui posent problème.

Fig 9.Echelle de Reeves & Reeves pour les modèles de la famille "systèmes béhavioristes"



4.3 Les situations "naturelles" d'enseignement de Paquette.

Les situations naturelles d'enseignement présentées par G. Paquette (2000) dans l'étude qu'il a faite pour la construction de portails pédagogiques à l'aide de l'éditeur Explor@ sont issues d'une observation du réel. Cette classification "naturelle" ne constitue pas une bonne base. Il faudrait en tirer un modèle qui permette la création de nouveaux types. Cependant, nous avons souhaité la présenter pour mémoire car elle donne, sous un point de vu encore différents, des détails qui permettent d'appréhender encore mieux la richesse et la complexité des différentes situations d'enseignements.

4.3.1 Classe technologique

La classe technologique est tout simplement une classe au sens traditionnel du terme où on peut trouver un ensemble de technologies installées et utilisées de façon permanente. On pourra trouver:

- Des équipements multimedias (écran, vidéo, sonorisation...)
- Des ordinateurs en réseau local (intranet) ou étendu (internet) avec un projecteur pour les présentation ex-cathedra

- Un système de vidéo conférence installé sur les postes de travail permettant d'assister à des présentations de personnes situées à distance.

La classe technologique est ouverte aux informations de l'extérieur, mais elle ne se déroule que dans un seul lieu physique (sauf exception liée par exemple à un manque de poste de travail obligeant l'enseignant à répartir les élèves dans plusieurs salles).

4.3.2 Classe distribuée

La classe distribuée est une classe technologique virtuelle répartie en plusieurs lieux physiques distants. On peut classer ces lieux de distribution schématiquement en 3 catégories:

- Salle de conférence équipée d'un système de diffusion de masse: les apprenants suivent l'évènement sur un grand écran
- Salle informatique comportant plusieurs postes de travail équipés d'outils de vidéo-conférence: les apprenants peuvent alors suivre à plusieurs dans une même pièce mais chacun devant son poste
- Equipement personnel sur un lieu de travail ou à domicile

Ces différents lieux peuvent être utilisés de manière exclusive ou conjointe.

On pourra trouver, associés aux moyens de vidéo-diffusion, une variété plus ou moins grande d'équipements périphériques tels que:

- caméras (webcam par exemple) permettant à l'enseignant de voir le(s) apprenant(s)
- microphones sensibles à la voix permettant un dialogue entre le(s) apprenant(s) et l'enseignant.
- caméra document pour la visualisation de documents à distance
- magnétoscope, lecteur CD ou DVD...

Dans ce cas, comme dans le cas précédent de la classe technologique, les évènements d'apprentissage se déroulent en direct, animés par un professeur utilisant une variété d'instruments de présentation de l'information. La présence simultanée des étudiants dans l'une ou l'autre des salles ou devant leurs postes personnels reliés par télécommunications avec la salle principale où se trouve le professeur est obligatoire.

4.3.3 Hypermedia distribué

Depuis l'arrivée massive de l'Internet, on compte déjà des milliers de cours Hypermedias sur le web et le nombre augmente chaque jour. L'Hypermedia distribué mise sur l'apprentissage individualisé pour un apprenant. L'apprentissage est autonome, généralement sans intervention d'un formateur, et sans nécessairement de collaboration entre les apprenants. Il n'y a pas donc pas de contrainte de lieu ou de temps imposé par le modèle de formation. Chaque apprenant accède à des contenus préfabriqués multi-médiatisés. Le matériel pédagogique peut être entièrement local (sur CD ou téléversé sur le poste de travail), en ligne via l'Internet par audio ou vidéo streaming, ou disponible de façon hybride en mode local ou en ligne.

4.3.4 Formation “en ligne”

La formation en ligne utilise aussi l’Internet, mais d’une façon fort différente. Elle est gérée par un formateur effectuant des présentations et coordonnant des interactions en différé (mode asynchrone) avec un groupe d’apprenants. Ceux-ci peuvent donc progresser à leur rythme, interagir entre eux et avec les matériels pédagogiques, entre les étapes définies par le professeur. Lors de ces étapes, le rythme des activités et, en bonne partie, le contenu des échanges, est géré par le formateur. Celui-ci allouera par exemple trois semaines pour un module, lancera une discussion, proposera des consignes de travaux, puis agira comme conseiller et expert de contenu jusqu’au module suivant. Dans ce modèle, les outils technologiques principaux sont asynchrones: forums de discussion, email pour l’échange privé de messages entre apprenants et le tutorat, transfert de fichiers pour l’échange et l’évaluation des travaux. Ce modèle est utilisé depuis au moins une quinzaine d’années dans les universités totalement à distance, comme la Télé-université du Québec ou l’Open University de Grande Bretagne, et, de plus en plus, dans des universités campus partout dans le monde où il devient une alternative à la formation en classe.

4.3.5 Communauté de pratique

La communauté de pratique peut utiliser les mêmes outils de communication asynchrone que la formation en ligne, mais aussi, parfois, des outils de discussion en temps réel tels que l’audio ou la vidéoconférence sur le poste de travail ou en salle. La principale caractéristique de ce modèle est l’échange d’informations et la discussion entre un groupe de spécialistes autour d’une tâche de travail. Il n’y a pas de formateur proprement dit, mais plutôt un animateur de réseau jouant un rôle similaire à l’animateur de groupe en présence. Règle générale, contrairement à un professeur ou un formateur, il possède moins d’informations que les participants, mais il dispose de techniques lui permettant de faciliter un échange fructueux entre les participants. Ceux-ci apprennent en échangeant des informations qu’ils n’ont pas nécessairement au départ et en comparant des pratiques à partir d’études de cas. Un serveur de document leur permet d’enrichir la base de connaissances commune. À partir de là, il peuvent résoudre des problèmes en équipe ou réaliser des projets à travers lesquels ils apprendront de nouvelles connaissances ou de nouvelles aptitudes.

Les échanges sont centrés sur l’exercice d’une tâche professionnelle comme apprendre à utiliser les technologies pour la formation. Ce modèle est particulièrement bien adapté en formation professionnelle continue et il a été utilisé à cette fin par des enseignants, des médecins spécialistes, des ingénieurs, pour parfaire leurs connaissances et confronter leurs informations et leurs pratiques. Le modèle a aussi été utilisé dans plusieurs cours universitaires à distance dans lesquels, par exemple, un ou plusieurs modules du cours prennent la forme d’ateliers autour d’une tâche à exercer ou l’approfondissement d’une pratique.

4.3.6 Support à la performance

Tout comme le modèle précédent, les systèmes informatisés de support à la performance (“Electronic performance support systems (EPSS)”) sont axés sur une tâche de travail, mais d’une façon différente. Ici la formation est surtout individuelle. Elle se déroule en liaison étroite avec les activités de travail, soit pendant cette activité lorsque l’apprenant a besoin de formation pour avancer dans la tâche, soit après l’activité, parce que

l’apprenant veut approfondir des questions qu’il s’est posé dans l’exercice de la tâche, soit avant l’activité, parce qu’il prévoit d’avoir besoin d’un supplément de formation pour la réaliser.

Le système de support à la performance (EPSS) s’intègre étroitement avec les systèmes informatisés de l’organisation servant au travail, notamment les bases de données institutionnelles. Il s’y ajoute divers modules de formation formelle, des aides à la tâche, des foires aux questions maintenues par un gestionnaire ou un expert de contenu, des agents conseillers intelligents ou en ligne. L’usager obtient ainsi des informations “juste à temps”, en fonction des tâches à résoudre. L’apprentissage est vu comme un processus de traitement de l’information.

5 Etude typologique des systèmes existants pour l'enseignement et mise en situation dans le cadre de référence théorique.

Afin de pouvoir appréhender les besoins et les caractéristiques du nouveau dispositif à mettre en oeuvre pour TECFA, il est important de faire une étude des logiciels et des outils présents sur le marché à l'heure actuelle. Nous nous attacherons à savoir quels usages peuvent en être fait. Nous allons présenter une typologie des différents systèmes existants, en soulignant les différents outils qui en sont caractéristiques, et en les situant dans le cadre de référence défini plus haut (voir chapitre 4 page 22)

Il existe aujourd'hui sur le marché une variété de produits qui permettent d'offrir des services de type pédagogique. Certains sont gratuits, d'autres commerciaux. Ils ne proposent pas tous le même degré d'interaction. Certains sont orientés vers la diffusion de supports de cours, d'autres sur la gestion des étudiants... Ils n'offrent pas tous le même degré de liberté par rapport aux types de contenus que l'on peut diffuser. Sans être exhaustifs, nous allons essayer de donner une typologie des différents systèmes existants.

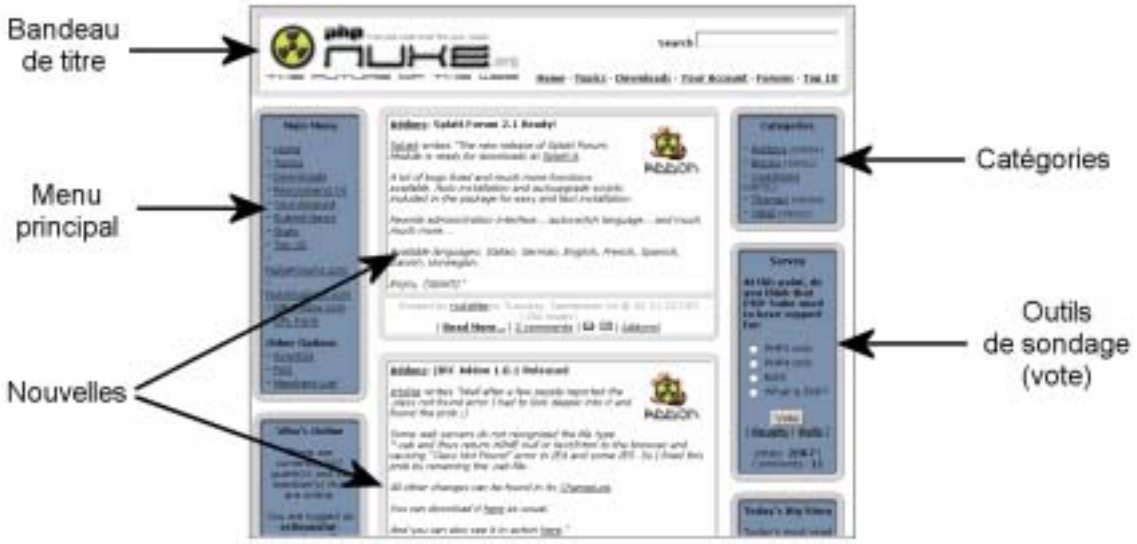
5.1 Utilisation pédagogique de portails d'information

5.1.1 Présentation

Le terme "portail" étant, à l'heure où nous écrivons ces lignes, utilisé à tort et à travers dans la littérature, les médias, les ressources en lignes, etc., il convient tout d'abord de définir avec précision l'objet de notre propos. On peut assimiler, de façon simple, le portail d'information à un centre de ressource. Dans sa forme la plus dépouillée, il peut s'agir d'une simple page HTML mise à jour de façon régulière qui liste des ressources en rapport avec un thème particulier comme par exemple une liste de sites Web consacrés à l'enseignement du français. Dans sa forme la plus évoluée, le portail peut être un dispositif complet doté de nombreux outils pour l'administration et l'utilisation, traitant de plusieurs sujets organisés par catégories généralement liées autour d'un thème central.

Il existe aujourd'hui de nombreux produits clef-en-main permettant de mettre en place des portails utilisant des technologies variées parmi lesquelles php, asp, jsp couplés à des bases de données MySQL, postgres... Il y a notamment eu ces dernières années une explosion de l'offre open-source avec des dispositifs tels que Slash, PHPweblog, Thatware, Sips, PHP-Nuke... Mais il existe également des produits commerciaux (comme IPlanet de Sun par exemple). Ces produits sont à des stades de développement plus ou moins avancés et les fonctionnalités qu'ils proposent varient en nombre et en complexité. Nous allons cependant essayer de dresser ici une liste des caractéristiques et des fonctionnalités les plus courantes et les plus intéressantes.

Fig 10.Vue de la page de garde du portail PHP-nuke (http://phpnuke.org)



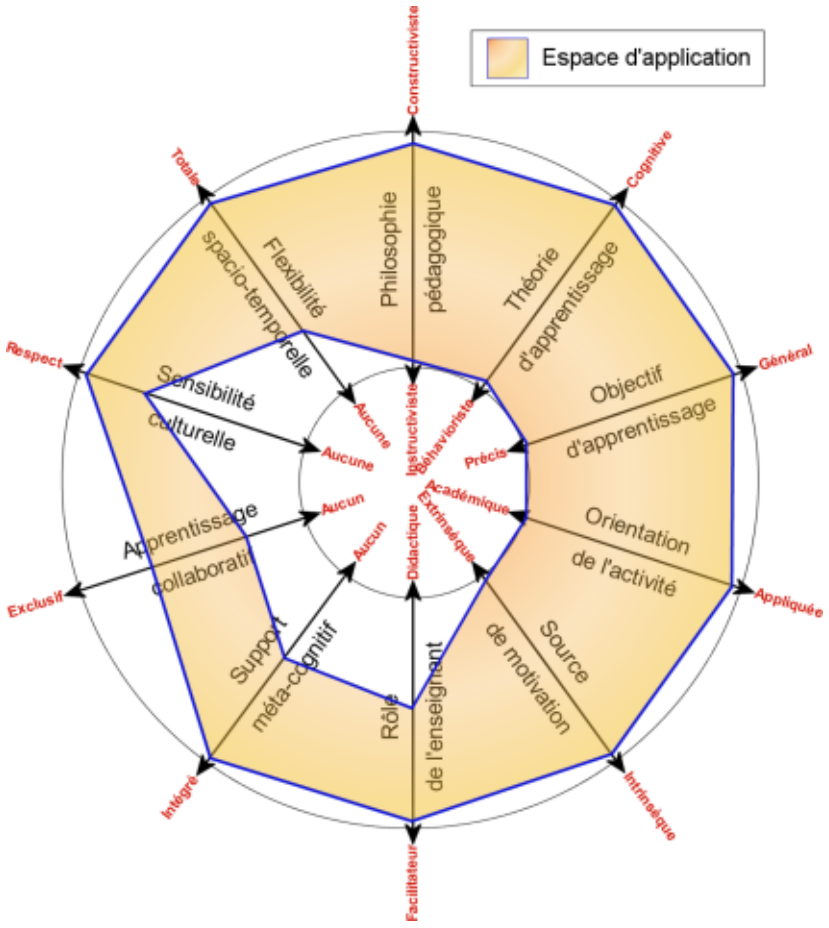
- La page d'entrée d'un portail est généralement très riche en informations. Pour s'y retrouver, les informations sont arrangées dans des boites (box) qui les regroupent par thème ou par fonctionnalité. La figure 10 montre une telle mise en page pour le portail PHP-nuke (http://phpnuke.org).
- Pour les dispositifs les plus évolués, il est possible d'intégrer des boites présentant des informations en provenance d'autres sites (la bourse, la météo, les dernières publication d'un éditeur... pour ne donner que quelques exemples). On parle alors d'informations syndiquées. Ces "échanges de boites" peuvent donner lieu à des formations de groupes d'intérêt entre différents sites. Il faut noter ici que le développement des technologies XML qui permettent le transport facile d'informations structurées permet un nouvel essor de ce genre de pratiques.
- La plupart de ces systèmes permettent à l'utilisateur de s'identifier et de fournir des préférences afin de paramétrer son environnement. Nous verrons dans les points suivants les aspects du portail qui peuvent être paramétrés. Cependant, il est important de noter que la quasi totalité des produits actuels permettent une certaine forme d'individualisation.
- Les utilisateurs du portail peuvent généralement contribuer à sa mise à jour en fournissant des ressources par le biais de nouvelles (stories - voir figure 10). Dans les cas où le système permet de s'authentifier, une trace de l'utilisateur qui a fourni la ressource est gardée.
- Les nouvelles fournies par les utilisateurs peuvent donner lieu à des discussions sous forme de commentaires organisés en forum. L'utilisateur peut, selon les dispositifs, déterminer la façon dont s'affichent les commentaires. Il peut par exemple choisir de visionner les plus vieux en premier, ou ceux qui ont remporté le plus de succès (*i.e.* ce qui ont été les plus lus).

- L'utilisateur peut également modifier l'affichage de la page d'entrée du portail. Ceci peut se faire à deux niveaux. Au niveau de l'aspect tout d'abord (look and feel), il peut généralement choisir dans une liste de thèmes pour en trouver un à sa convenance. Deuxièmement, il peut décider des boites qui seront présentées et de leur position sur la page selon ses intérêts particuliers, et même, dans certains cas, se fabriquer un boite personnalisée pour son propre usage.
- Des outils d'administration permettent au(x) responsable(s) du site de gérer facilement les utilisateurs, de créer de nouvelles sections, de supprimer/ modifier/ archiver des nouvelles ou des commentaires, de rajouter de nouvelles boites d'informations
- On peut encore trouver de nombreux outils tels que sondages avec consultation instantanée des résultats (voir figure 10), indicateurs de présence (nombres d'utilisateurs connectés), téléchargement, moteur de recherche intégré, calendrier... Les produits les plus aboutis permettent d'intégrer des modules et proposent une interface de programmation pour les développer.
- Les portails, du moins pour ceux diffuser gratuitement que nous avons pu tester, sont relativement facile à mettre en place. Leur installation est possible même par des débutants, sous réserve qu'ils disposent de l'environnement adéquat (serveur web, langage de script, base de données) ainsi que des droits et des outils pour y accéder (telnet, ftp, front end pour les bases de données).

5.1.2 Analyse

Vu l'importance de l'offre de produits portails, et les fonctionnalités qu'ils contiennent, il n'est pas étonnant de voir, ici et là, des tentatives pour les utiliser dans la pédagogie et l'apprentissage.

Fig 11.Espace d'application des portails sur l'échelle de Reeves & Reeves



Les portails sont par définition des environnements très ouverts puisqu'ils permettent de référencer n'importe quelle ressource en ligne. Nous essayons sur la figure 11 de tracer une zone d'application pour l'utilisation pédagogique des portails d'information sur l'échelle de Reeves & Reeves. Les sujets que l'on pourra traiter et la forme des documents à analyser pourra donc être très variée. La philosophie pédagogique et la théorie d'apprentissage pourront largement changer selon la mise en oeuvre choisie. Il en va de même pour la source de motivation et l'orientation de l'activité.

Cependant, quelques dimensions de l'échelle méritent une attention particulière.

- Il y aura toujours une certaine forme de support métacognitif qui pourra être donné par le biais des commentaires pouvant être laissés sur chaque ressource par les utilisateurs. Celui-ci pourra être plus ou moins intégré à l'activité mais pourra difficilement être absent.
- L'apprentissage collaboratif pourra également être plus ou moins utilisé. Il est difficile de concevoir qu'il soit totalement inexistant car les discussions pourront toujours y jouer un rôle. On peut également difficilement voir comment il pourrait être exclusif, le simple forum asynchrone n'étant pas le dispositif le mieux adapté pour cela.
- La possibilité de personnaliser le portail apporte un respect de la sensibilité culturelle plutôt grand. Attention toutefois aux possibilités qui sont laissées pour personnaliser cette interface.

- Enfin, sauf cas exceptionnel dicté par un enseignant par exemple, la flexibilité spacio-temporelle est totale puisque le dispositif est accessible à tout moment et depuis n'importe quel poste équipé d'un navigateur.

L'utilisation pédagogique de portails d'information est particulièrement bien adapté pour l'apprentissage à partir de ressources (ressource based learning).

5.2 Plateformes pédagogiques

5.2.1 Présentation

Les plateformes pédagogiques[1] sont des produits permettant d'assister la conduite des enseignements à distance. Ces produits peuvent être commerciaux, comme WebCT (http://www.webct.com) ou LearningSpace (http://www.lotus.com). Cependant, l'offre de produit open source s'est sensiblement développée ces dernières années. On peut citer comme exemple la plateforme gratuite Ganesha (http://www.anemalab.org).

Ce type de logiciel regroupe les outils nécessaires aux trois principaux utilisateurs[2] d'un dispositif qui a pour finalité la consultation à distance de contenus pédagogiques, l'individualisation de l'apprentissage et le télétutorat.

- Dans ce système, l'enseignant crée des parcours pédagogiques types et individualisés de son enseignement, incorpore des ressources pédagogiques multimedias et effectue un suivi des activités des étudiants.
- L'étudiant consulte en ligne ou télécharge les contenus pédagogiques qui lui sont recommandés, organise et a une vue de l'évolution de son travail, effectue des exercices, s'auto-évalue, transmet des devoirs à corriger et effectue des tests notés (type QCM généralement).
- Enseignants et étudiants communiquent individuellement ou en groupe, créent des thèmes de discussion et collaborent à des documents communs.
- L'administrateur installe et assure la maintenance du système, gère les accès et les droits des uns et des autres, crée des liens avec les systèmes d'information externes (scolarité, catalogues, ressources pédagogiques, etc.). On entend donc par administrateur un rôle spécifique à la plate-forme et non un rôle administratif habituel de l'établissement.

Autour de ces premières finalités, peuvent s'ajouter d'autres fonctionnalités et d'autres rôles. Une plate-forme pourra ainsi comporter des fonctionnalités relatives aux référentiels de formation et à la gestion de compétences, aux catalogues de produits de formation, au commerce électronique, à la gestion administrative, à la gestion des ressources pédagogiques, à la gestion de la qualité de la formation.

Dans le cadre de l'évolution des techniques, des infrastructures de réseau et des normes, une plate-forme pourra utiliser des médias et des modes de communication plus diversifiés et enrichir les procédures d'échanges de données avec des ressources

1.La définition et les explications qui suivent sont très largement inspirées et adaptées de "*Etude Comparative Technique et Pédagogique des Plates-Formes pour la Formation Ouverte et à Distance"* (OVAREP, 2000)

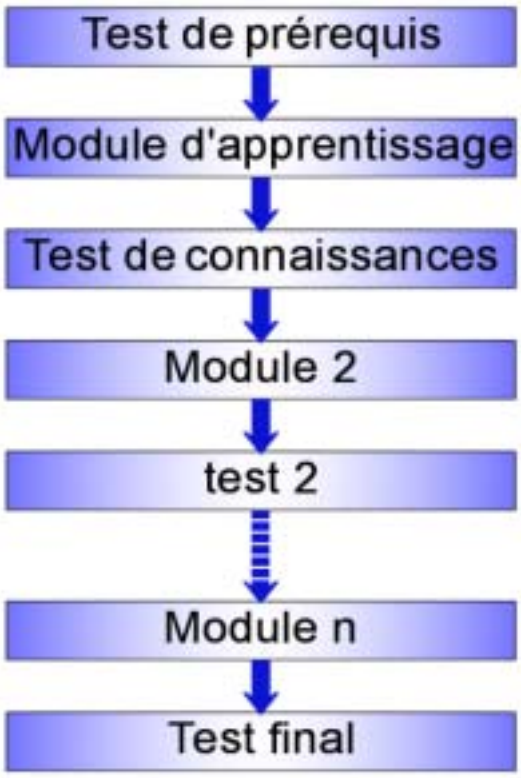
2.Enseignant/créateur de cours, étudiant et administrateur

pédagogiques d'apprentissage ou d'autres systèmes d'information.

Comme pour les portails présentés plus haut (voir chapitre 5.1 page 49), les fonctionnalités offertes par les logiciels de cette catégorie varient en nombre et en complexité. Mais nous allons également tenter de dresser une liste de celles qui sont les plus communes.

- Les rôles de base présentés plus haut (administrateur, formateur/créateur de cours, étudiant) sont les trois rôles minima supportés par toutes les plateformes. Selon les différents produits ils peuvent être modulables. Certaines plateformes proposent des rôles supplémentaires avec par exemple des rôles d'administration des matériaux pédagogiques, des rôles d'administration de la scolarité ou de la formation, la dissociation des rôles de formateur et créateur de cours, ou encore l'apparition de rôles tels que assistant d'enseignement (pour corriger des tests) ou helpdesk (pour l'aide aux utilisateurs de la plateforme). Sur quelques rares systèmes, il est possible de cumuler plusieurs rôles.
- La création et la modification de contenu pour la plateforme peut se faire de deux manières différentes. Soit directement dans la plateforme avec des outils d'édition fournis, soit en important des matériaux créés de façon externe. Les matériaux créés à l'extérieur de la plateforme peuvent avoir différents formats selon les produits: html, format de produits existants (word, excel...). Certaines plateformes offrent une compatibilité avec des éditeurs tiers (front page, dreamweaver...). Il est cependant important de noter que, quel que soit le produit, on arrive finalement à une certaine forme de stockage sous un format propriétaire. Il devient alors difficile de transporter la totalité d'un cours créé dans une plateforme particulière vers une autre. Enfin, l'édition ou la création du contenu directement dans la plateforme oblige très souvent à travailler en ligne, ce qui peut revenir cher en communication téléphonique pour les personnes connectées par modem.
- Tous les produits proposent au minimum un éditeur en ligne de quizz simples de type QCM. Certaines plateformes offrent également la possibilité d'importer les tests. Pour cela, le support est assez différent. Il peut s'agir du support d'un logiciel tiers (support d'un module de création de test de dreamweaver pour la plateforme Ganesha par exemple) ou du support pour importer des fichiers textes contenant la description du test (avec un format de type XML ou de fichier plat avec séparateurs). Dans ce dernier cas, la création du fichier demande une certaine connaissance technique.
- La progression de l'apprenant dans ces environnements se fait sur la base de parcours types créés par l'enseignant. Ces scénarios restent bien souvent assez linéaires, les possibilités pour définir la disponibilité d'un module de l'enseignement se basant sur des résultats de tests ou sur une date de disponibilité. La figure 12 schématise une progression assez commune dans une plateforme classique. Il faut cependant noter que de nombreux progrès ont été accomplis ces dernières années pour augmenter les possibilités des parcours proposés. Ainsi, à condition d'y consacrer le temps nécessaire, il est aujourd'hui possible d'implémenter des parcours assez complexes basés sur le contrôle par l'apprenant plutôt que par le contrôle du système.

Fig 12.Progression classique dans une plateforme pédagogique



- Les compétences requises pour l'utilisation d'un produit peuvent être regardées à trois niveaux différents qui correspondent aux trois rôles de base: utilisation simple (étudiant), création de cours (enseignant) et installation/maintenance (administrateur).

  1 Pour l'utilisation, aucune compétence particulière n'est requise. Pour une utilisation de base, il suffit de savoir se servir d'un navigateur.

  2 Pour la création de cours, les compétences varient largement selon les produits et peuvent être de deux types. Il peut être nécessaire de maîtriser des langages ou des formats de fichier externes (HTML pour la création de documents, format XML pour la création de test). Cependant, même si cela parait être une évidence, il faut maîtriser la plateforme elle même, comprendre son fonctionnement et se familiariser avec la palette d'outils proposés.

  3 Pour l'administration, l'installation de la plateforme elle même est très rarement automatisé. Il faut donc intervenir sur le serveur. Il peut également être nécessaire d'installer d'autres produits nécessaires au fonctionnement de celle-ci (serveur de messagerie, langage de script, serveur web spécifique, base de donnée...)

- Les plateformes pédagogiques incluent, au minimum, des outils de communication asynchrone de type email interne et forum. Les forums se distinguent, entre les différents produits, par leur capacité à être contextualisés pour chacun des enseignements proposés sur la plateforme, voire sur une activité spécifique d'un enseignement. Certaines plateformes proposent également des outils de communication synchrone comme les "chats" (discussion synchrone textuelle en ligne) ou la vidéoconférence. Cependant, ces outils sont assez communément

présentés et implémentés comme des fonctionnalités de télétutorat, permettant le suivi des travaux des étudiants par l'enseignant, au même titre que les outils permettant de visualiser la progression des étudiants dans le parcours et de connaître les scores des tests. Ces fonctions sont rarement présentées comme une fonctionnalité de collaboration entre apprenants.

- Enfin, un certain nombre d'outils supplémentaires peuvent être proposés aux étudiants et aux enseignants tels que:

  - Outils avancés de gestion des groupes d'étudiants
  - Outils avancés de gestion de contenu
  - Aide à la création et modèles de parcours types
  - Visualisation de parcours, dictionnaires, glossaires

5.2.2 Analyse

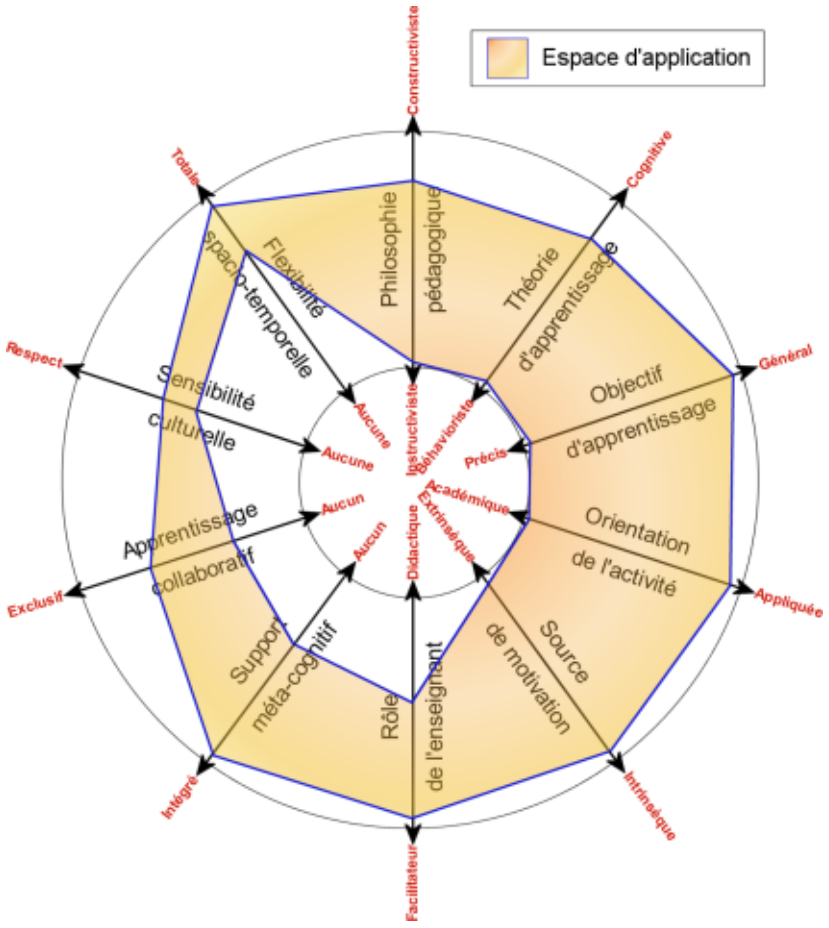
Les plateformes sont des environnements fermés. Bien qu'elle aient beaucoup évoluées ces dernières années, elles restent assez contraignantes dans leur utilisation et ne permettent pas de sortir du schéma imposé par leur design. De plus, comme nous l'avons déjà dit plus haut, le format de stockage est toujours propriétaire et ne permet pas de transférer le contenu d'une plateforme à une autre d'un type différent. En d'autres termes, il n'est pas du tout sûr que l'effort investi pour fournir du contenu dans une plateforme puisse être rentabilisé en cas de changement de système.

Cependant, la grande diversité des outils que les plateformes offrent, l'intégration de ceux-ci dans un ensemble cohérent ainsi que la relative facilité de prise en main et de mise en oeuvre du produit ont fait leurs succès auprès de beaucoup d'institutions d'enseignement. Elles se prêtent particulièrement bien à des apprentissage de type instruction programmée ou programmation ramifiée. Pour ne pas induire le lecteur en erreur, nous tenons à préciser qu'il est tout-à-fait possible d'utiliser d'autres styles de pédagogies dans ces plateformes, mais elle ne sont pas forcement les mieux adaptées pour créer un environnement collaboratif par exemple. L'implémentation d'un dispositif particulier sera parfois plus aisée et plus rapide avec d'autres catégories de produits.

En bref, nous tenons à souligner que, malgré le fait que ce type de produit domine actuellement le marché (le produit WebCT se taillant une part de lion dans le gâteau), ils ne constituent pas la panacée, ***la*** solution.

Voyons maintenant sur quelles dimensions de l'échelle de Reeves & Reeves ces plateformes nous permettent de jouer (figure 13).

Fig 13.Espace d'application des plateformes sur l'échelle de Reeves & Reeves



- L'objectif d'apprentissage, la source de motivation et l'orientation de l'activité pourront varier entre leurs valeurs extrêmes selon le bon vouloir du concepteur du cours.
- La philosophie pédagogique et la théorie d'apprentissage peuvent en théorie varier également d'un extrême à l'autre. Dans la pratique cependant, il est beaucoup plus aisé de faire des cours basés sur le béhaviorisme et l'instructivisme à l'aide de ces systèmes. Se placer de l'autre côté de l'échelle demande, la plupart du temps, un gros effort, souvent rédhibitoire.
- Un certain nombre d'outils sont proposés pour le support méta-cognitif: auto-évaluation, visualisation de parcours dans le module... Celui-ci peut être amélioré par des activités spécifiques.
- Une certaine forme d'apprentissage collaboratif pourra être menée par le biais des quelques outils de communication: forum, mail interne... Toutefois, même si certains produits proposent des "black board" distribués, et la possibilité de partager des URL dans des "chats", la collaboration ne constitue pas le coeur de la philosophie de ces plateformes. Leur rôle est plus tourné vers la distribution à un grand nombre d'apprenants.
- La sensibilité culturelle pourra être respectée. L'interface de chaque étudiant peut être personnalisée dans une certaine mesure. Cependant, ces possibilités restent limitées. D'autre part, il n'existe pas d'outils permettant l'adaptation du matériel pédagogique.

- La flexibilité spacio-temporelle est, quant à elle, quasiment totale. Mis à part les rendez-vous pour des communications synchrones (chat, vidéo-conférence...) ou bien des contraintes spécifiques de temps pour accéder à un document précis (test disponible de 14h00 à 16h00 le lundi par exemple), le système reste généralement accessible tout le temps depuis n'importe quel ordinateur disposant d'un navigateur.

6 Cahier des charges du nouveau dispositif

Maintenant que nous avons posé un cadre théorique de référence qui nous a permis d'appréhender la complexité des interactions dans l'enseignement et que nous avons fait le tour des systèmes existants qui permettent de supporter ces interactions, il convient d'esquisser le cahier des charges du nouveau dispositif pour TECFA. Quels types d'interactions et de situations théoriques d'apprentissage devront être supportées ? Quelles fonctionnalités et services devront être proposés au différents types d'utilisateurs ? Comment éviter les problèmes qui ont amené au délaissement et à la mort de la plateforme actuelle ?

Nous ne nous trouvons pas ici dans un cadre institutionnel classique. L'unité TECFA ne se situe pas comme un simple consommateur de "produits éducatifs standards". TECFA enseigne les technologies de l'apprentissage et de la formation en utilisant ces mêmes technologies. Comme nous l'avons brièvement expliqué dans l'historique que nous avons tracé du campus (voir chapitre 2 page 7), chaque enseignant a des points de vue différents sur les technologies à utiliser pour ses propres enseignements et sur la manière de les utiliser. Ceci implique comme première contrainte que le nouveau dispositif devra pouvoir s'adapter au plus grand nombre possible de situations d'apprentissage théorique. Idéalement, si l'on se réfère au cadre théorique utilisé (voir chapitre 4 page 22), il faudrait pouvoir supporter toute les dimensions de l'échelle introduite par Reeves & Reeves, toutes les familles de modèles d'enseignement décrites par Joyce et al., toutes les situations d'enseignement présentées par Paquette. Nous pouvons donc dès à présent poser l'hypothèse que l'utilisation d'un produit unique existant sera difficile, aucune de ces catégories de produits ne supportant un spectre aussi large.

D'autre part, il faut pouvoir intégrer, avec un effort minimum, des ressources préexistantes sans avoir à les insérer à l'intérieur d'une structure rigide et contraignante. C'est un des points cruciaux qui a conduit certains enseignants à ne pas utiliser le campus virtuel. Pour cette raison encore, il parait difficile d'utiliser un seul produit clef en main, chacun d'eux faisant généralement appel à un format propriétaire, que ce soit au niveau des activités et des interactions proposées que de la gestion du contenu. Charger le contenu d'un cours dans un nouveau dispositif en passant par des formulaires n'est pas plus acceptable que d'avoir à réécrire la totalité des documents produits dans un autre format, en suivant de nouvelles règles.

De même, il doit être possible d'intégrer des outils existants, produits sur place ou empruntés à d'autres projets. Il est par exemple beaucoup plus coûteux de programmer soi même un calendrier avec toutes les fonctions qu'il doit comporter alors que de tels outils existent déjà, libres de droits, modifiables et adaptables à souhait. Si des outils extérieurs sont adaptés au nouveau dispositif, il faut pourvoir, dans la mesure du possible, tirer partie des améliorations qui seront faites après leur intégration. Il est donc nécessaire de fournir des outils permettant de gérer les versions des différents outils utilisés et d'intégrer le plus facilement possible les modifications apportées par un ou plusieurs développeurs qui peuvent, de plus, travailler en même temps.

Bien que l'ouverture de ce nouveau dispositif doivent être la plus large possible afin de laisser la porte ouverte à tous types d'activités et de contenus, il est tout de même indispensable qu'il propose un certain nombre de fonctions et d'outils constituant une base solide sur laquelle tous les développeurs de TECFA puissent s'appuyer. Cette base

doit être évolutive. On peut donner comme exemple l'accès à la base de données des utilisateurs et l'authentification. De plus, cette base de services et d'outil sera amenée à s'élargir au fur et à mesure que de nouveaux développements seront réalisés. En effet, dans un soucis de ne pas "re-inventer la roue" à chaque fois, il serait judicieux de pouvoir intégrer à cet ensemble de base des librairies de fonctions ou des services développés dans le cadre d'une activité par exemple et qui s'avéreraient bien adaptés et utiles dans un ensemble de situations récurrentes. Pour cette raison encore, il faut un outil de gestion des versions qui permette de modifier les librairies de base, d'y intégrer de nouveaux fichiers et de travailler à plusieurs sur le même code source.

Tous les services et fonctions de base de l'environnement doivent être documentés de sorte qu'il soit possible pour un développeur nouveau venu dans un projet de pouvoir les utiliser sans avoir à lire directement le code source de chacun des fichiers qu'il va utiliser. Pour cela, outre le fait que les developpeurs devront s'astreindre à une hygiène de programmation, il est indispensable d'utiliser des outils permettant de générer la documentation de façon automatique, lisible et ergonomique. C'est encore un des points qui a mené à la mort de l'ancien campus.

Les grandes lignes de ce cahier des charges étant tracées, nous allons maintenant rentrer plus en détails dans quelques points particuliers.

6.1 Les différents types d'utilisateurs et les outils associés.

Tout dispositif ou plateforme a des types d'utilisateurs différents qui ont des besoins différents. On parle plus volontiers de rôle, chaque utilisateur pouvant au besoin passer d'un rôle à l'autre ou en remplir plusieurs à la fois. Le dispositif dont nous sommes en train de tracer les grandes lignes sera utilisé par des personnes pouvant se répartir dans les cinq grands rôles suivants:

- Les visiteurs anonymes
- Les étudiants
- Les enseignants
- Les administrateurs
- Les développeurs

On peut immédiatement mettre à part le rôle de développeur qui est différent des quatre autres. En effet, il n'est pas attaché à l'utilisation directe du dispositif mais plutôt à son élaboration et à son évolution dans le temps. Nous traiterons donc celui-ci en dernier.

6.1.1 Les visiteurs anonymes

Le but premier de ce dispositif n'est pas sa visibilité à l'extérieur. Cependant, TECFA a une tradition d'ouverture vers l'extérieur et souhaite la conserver. La plus grande partie des documents, des activités, des bases de données doit pouvoir être visible sans contrôle d'accès. On pourrait résumer de façon simple la philosophie qui sous-tend ce libre accès aux informations contenues dans le système par la formule: *"voir sans participer"*

Les visiteurs n'ont donc besoin d'aucun outil particulier. Il faut seulement prévoir dans le dispositif, à chaque fois que cela est possible, un accès anonyme sans possibilité de modification de l'information présentée. Des exceptions peuvent être faites selon le type d'information présentée. L'évaluation d'un exercice par exemple ne devrait être

accessible que par le professeur responsable et les étudiants concernés. A l'opposé, il peut être intéressant de laisser la possibilité aux visiteurs de participer à certains forums pour augmenter le nombre de point de vue exposés et stimuler l'échange d'information.

6.1.2 Les étudiants

Les étudiants sont en définitive le coeur de cible de ce dispositif, comme ils l'étaient dans l'ancien campus virtuel. C'est à eux que l'on s'adresse en priorité. S'ils n'étaient pas là, tout le dispositif n'aurait plus de raison d'être.

Les étudiants doivent pouvoir se servir de ce dispositif pour différentes utilisations qui sont à nos yeux d'égale importance:

- La recherche d'informations directement liées aux enseignements (heures des cours, dates, travaux à rendre....). Pour cela, des outils doivent être fournis tels qu'un agenda, des pages de descriptions des cours... Il est crucial que ces informations puissent être retrouvées de façon efficace à un emplacement stable dans le temps, du moins à l'échelle de l'année scolaire.
- La recherche d'information à des fins personnelles ou pour réaliser un travail (documentation de logiciel, de langage de programmation, recherche de références bibliographiques ou de documents...). Bien entendu, TECFA ne doit pas être la seule source d'informations de ce type. Cependant, un grand nombre d'informations sont déjà présentes dans le site Web ou dans des bases de données. Toutes ces informations doivent pouvoir être intégrées ou référencées et facilement accessibles.
- La participation à des activités en ligne, que ce soit en support à un cours présentiel ou à distance, telle que l'ArgueGraph ou le Studio (voir chapitre 2.1 page 7). Beaucoup de ces activités nécessitent une authentification afin de pouvoir suivre le parcours de l'étudiant, enregistrer ses résultats, former des groupes de travail avec des pairs. Il faut donc prévoir un authentification efficace et centralisée afin d'éviter la multiplication des logins et des mots de passe pour une même personne, situation amenant souvent la confusion. Pour faciliter le travail en groupe, un certain nombre d'outils standards peuvent être fournis tels que des systèmes de gestion et d'annotation de documents, des outils de co-rédaction (co-authoring).
- L'entraide, la valorisation et la dissémination des connaissances acquises, la participation à une communauté de pratique. Les étudiants doivent pouvoir disposer d'espaces d'échanges communautaires et de moyens de communication synchrones et asynchrones qui puissent stimuler et faciliter le fonctionnement social de la communauté: mailing listes, forums, système de nouvelles type portail, chat...

6.1.3 Les enseignants

Les enseignants, comme nous l'avons déjà dit, ont des besoins variés qui dépendent principalement du sujet qu'ils enseignent, du type d'enseignement qu'ils souhaitent mettre en place et de leurs points de vue par rapport à l'utilisation des technologies. Les interactions mises en jeu seront par exemple très différentes selon que l'on demande aux étudiants de trouver de l'information par eux mêmes pour rédiger un papier ou qu'on leur demande de participer en ligne à une activité collaborative. D'autre part, ils doivent pouvoir tirer parti du travail qu'ils ont déjà réalisés en mettant au point des pages de présentation de leurs enseignements, des activités pédagogiques ou des bases documentaires pré-existantes.

C’est pourquoi il est très important que le système soit le plus ouvert possible quant aux types d’interactions supportés, à l’intégration ou au référencement de différents formats de media ou d’activités et aux différentes technologies supportées.

Cependant, tous les enseignants ont des besoins génériques relatifs notamment à la gestion des étudiants et de la communauté:

- Ils doivent pouvoir créer des groupes de travail, gérer les étudiants participants à leurs cours, donner des nouvelles concernant les délais pour rendre des travaux ou des informations spécifiques à leur enseignement. Ils gèrent leurs différentes activités pédagogiques (mise à disposition d’un test...), corrigent des travaux, donnent des notes aux étudiants. Nous voyons là encore le besoin pour une base de données utilisateurs centralisée assortie d’outils pour l’interroger et la modifier.
- Ils doivent pourvoir communiquer avec leurs étudiants de façon synchrone et/ou asynchrone, envoyer des mailings, mettre à jour les informations correspondant à leurs enseignements (horaires, deadlines pour les travaux à rendre...). Pour cela, ils doivent pouvoir accéder à des outils de mailing list, des forums, fournir leurs propres pages de contenu et les référencer ou utiliser un outil clef en main
- Ils essayent, dans la mesure du possible, de motiver la collaboration entre les étudiants de la communauté et des les aider (tutorat). Pour cela, ils doivent avoir les moyens de surveiller les interactions entre les étudiants en utilisant des outils leur facilitant la tâche leur permettant de visualiser l’activité des étudiants (awareness tools): avancement des travaux, activité sur un forum...

6.1.4 Les administrateurs

Les administrateurs s’occupent de la gestion du dispositif et de ses utilisateurs. Ils prennent en charge par exemple l’inscription des nouveaux étudiants à la rentrée universitaire, mettent à jour les données concernant les utilisateurs, créent des groupes globaux, ouvrent de nouveaux forums, font circuler les informations globales concernant le fonctionnement du dispositif (down time pour les maintenances, nouveaux outils disponibles...).

Encore une fois, nous sentons la nécessité de stocker dans un endroit centralisé toute l’information concernant les utilisateurs ainsi que le besoin d’outils appropriés pour utiliser ces données et les modifier.

6.1.5 Les développeurs

Les développeurs forment une catégorie à part et ont des besoins spécifiques. En effet, leur rôle n’est pas directement lié à l’utilisation du dispositif en tant que tel. Ils doivent faire évoluer le dispositif, créer des activités ou des services basés sur le dispositif mis en place. Pour cela, ils doivent disposer d’une interface stable leur permettant d’utiliser le dispositif existant.

Cet interface doit constituer une véritable couche d’abstraction permettant de manipuler les différents objets du système et d’utiliser ses différentes fonctions sans rentrer dans les détails du fonctionnement interne du système. Une fonction d’authentification par exemple doit pouvoir être utilisée sans pour autant connaître le type de base de données contenant les informations utilisateur, le cryptage utilisé pour le mot de passe ou le serveur sur lequel se trouvent les données.

En d'autres mots, le nouveau dispositif doit disposer d'une "interface de programmation d'application" ou API (Application Programming Interface). Dans un premier temps, cette API ne contiendra que des fonctions de bases. Mais elle sera progressivement amenée à évoluer au fur et à mesure des développements pour contenir des fonctions de plus en plus haut niveau (affichage de données, génération d'images, communication avec les bases de données...).

Afin de mener à bien toute ces taches, les developpeurs ont besoin d'un certain nombre d'outils qui répondent aux problématiques suivantes:

- La gestion du code source.
- La documentation de l'API existante
- L'hygiène de programmation et la documentation du code créé

Il est fréquent que plusieurs développeurs travaillent sur un même projet en même temps. D'autre part, il arrive aussi souvent qu'un projet laissé par un développeur soit repris par un autre quelques temps plus tard. Enfin, le nombre de logiciels libres de droits se multipliant, il n'est pas rare que l'on souhaite adapter des logiciels existants à ses propres besoins, sans pour autant perdre le bénéfice des améliorations apportées dans la version originale par la suite

Pour que tout le monde puisse travailler ensemble sans se gêner (écrasement de fichier ouverts de façon concurrente...) et que tout le monde puisse profiter des améliorations apportées par chaque personne (qu'il soit interne au projet ou externe dans le cas de l'adaptation d'un logiciel existant) il est absolument indispensable que l'équipe dispose d'un outil de gestion des versions de code source. Ces outils permettent à chaque développeur de travailler sur sa propre copie du code, d'importer les changements apportés par d'autres developpeurs en cours de travail, d'exporter ses modifications pour les rendre disponibles aux autres developpeurs. Ce type d'outils permet également d'importer la nouvelle version d'un logiciel tout en conservant les modifications apportées sur sa version antérieure. Tous les fichiers de développement sont stockés dans un dépôt central qui contient le code source et toutes ses révisions depuis sa création.

Pour que les développeurs puissent programmer efficacement, il doivent avoir à leur disposition la documentation des fonctions, classes, objets qu'ils doivent utiliser. Cette documentation doit être exhaustive et la plus précise possible.

Afin d'assurer la pérennité du dispositif, les développeurs doivent s'astreindre à quelques règles de base, décidées en commun. Il faut que les différents programmeurs s'entendent sur des conventions à respecter pour le nom des variables, des fonctions. Toute modification du code existant ou toute création de code nouveau doit faire l'objet d'une opération dans le dépôt de code source dûment labélisée pour pouvoir garder la trace des changements opérés. Il est également indispensable que la documentation soit maintenue à jour et générée de façon automatique avec un outil approprié. La plupart des langages de programmation possède de tels outils. Pour qu'il soit utilisable, les commentaires dans le code source doivent, là encore, suivre certaines conventions qu'il faut expliciter.

7 Revue des possibilités techniques

Pour réaliser le cahier des charges que nous venons de présenter, nous allons devoir mettre en oeuvre un certain nombre de technologies et de logiciels existants. Afin de pouvoir faire notre choix en toute connaissance de cause, nous devons tout d'abord identifier ces besoins et passer en revue les différentes solutions existantes. Ces différentes possibilités devront ensuite être mises en perspective avec l'environnement informatique de TECFA, les possibilités de mise en oeuvre et les compétences des différents collaborateurs.

Les besoins au niveau technique se répartissent principalement dans les 4 domaines suivants:

- Stockage d'information / Bases de données
- Gestion de documents / contenu
- Langages de programmation / scripting
- Gestion de codes source (collaboration / versions)

Afin de limiter les contraintes posées sur les développeurs, nous nous intéressons ici aux logiciels et aux technologies qui pourront former la couche de base de bas niveau de notre dispositif. Les solutions choisie doivent permettre aux programmeurs un maximum de flexibilité. Nous nous bornerons donc à l'analyse d'outils générique, de technologies de base et de formats standards pour stocker des données. Les outils commun de plus haut niveau devront être développé ou choisi après concertation de l'ensemble des intéressés.

7.1 Stockage d'informations / Bases de données

La grande majorité des pages web, documents et autres médias que nous pouvons aujourd'hui visualiser sont créés de façon dynamique, au moment où on y accède. De même, les informations concernant un utilisateur particulier sont généralement stockées de manière stable pour pouvoir être interrogées, mises à jour. Toutes ces données sont généralement enregistrées de façon à ce qu'elles soient facilement recherchables, mises en relation avec d'autres données similaires ou complémentaires, organisées selon une logique définie.

Il existe aujourd'hui trois solutions principales de stockages des données qui répondent chacune à des utilisations et des besoins différents:

- Les annuaires type LDAP (Lightweight Directory Access Protocol)
- Les bases de données relationnelles
- XML (fichiers / bases de données)

7.1.1 Les annuaires de type LDAP

L'acronyme LDAP (Lightweight Directory Access Protocol) désigne au départ non pas un type de stockage des données ou de base de données mais un protocole permettant d'accéder de façon simple à des annuaires d'entreprises. Ces annuaires sont basés sur le

standard X.500 de l'International Telecommunications Union (Johner et al., 1998). Ils sont conçus pour stocker de l'information concernant les personnes mais également les machines d'un réseau, les réseaux eux mêmes.

L'information à l'intérieur de l'annuaire est organisée sous forme d'arbre logique, chaque noeud contenant un objet.

Mettre en place un annuaire de ce type demande d'énormes ressources tant au niveau de la qualité du réseau que de la puissance des machines qui le gère (serveur) ou l'interroge (client). En effet, ce modèle est basé sur l'utilisation d'un standard de communication à sept couches baptisé OSI (Open System Interconnect). Ce protocole se développe lentement dans les grosses infrastructures alors que la naissance d'internet consacre le protocole TCP/IP moins coûteux en ressources et plus largement répandu. Germe alors l'idée de pouvoir accéder aux annuaires X.500 par l'intermédiaire d'un protocole moins lourd, basé sur TCP/IP: le Lightweight Directory Access Protocol.

De nos jours, LDAP s'apparente plus à un standard. S'il existe encore des services LDAP constituant des passerelles vers des annuaires X.500, de nombreux produits commerciaux ou open source ont vu le jour et stockent directement les informations. On fait référence indifféremment, dans l'un ou l'autre cas, à un serveur LDAP.

Un serveur LDAP est une base de données qui a des caractéristiques particulières. Pour commencer, les annuaires sont plus souvent consultés que mis à jour. Ils sont optimisés pour la lecture (bien souvent au détriment de l'écriture). Ils sont donc particulièrement efficaces pour stocker des informations stables qui seront distribuées à un grand nombre de client.

Le stockage des données s'effectue sous forme d'objets contenant des attributs qui constituent les noeuds d'un arbre. Le protocole de communication permet la distribution de l'arbre de données sur différents serveurs ainsi que la réplication des données pour la sauvegarde ou la répartition de charge pour les grosses structures.

L'arbre de données peut être parcouru, en entier ou sur une branche spécifique, à la recherche de différents types d'objets et de leurs attributs. Les objets peuvent être polymorphes. Un noeud peut par exemple être en même temps de type "utilisateur" et "imprimante" pourvu qu'il possède les attributs nécessaires. Les types d'objets et leurs attributs sont décrits dans des schémas que l'on déclare dans le fichier de configuration du serveur en fonction de ses besoins. Chaque objet possède un nom distinct (distinguished name ou dn) basé sur la place qu'il occupe dans l'arbre de données. La figure 14 schématise la structure d'un arbre LDAP.

Fig 14.Exemple de structure d'un arbre de données LDAP

Une autre caractéristique des annuaires est qu'ils n'offrent généralement pas de support pour les transactions (il existe cependant des exceptions). Les transactions sont des opérations de type tout-ou-rien qui doivent être réalisées en totalité ou pas du tout. Par exemple, une opération de débit/crédit entre deux comptes en banque: si une somme d'argent est débitée d'un compte donné, elle doit être créditée sur un autre. Les deux opérations doivent être faites de façon simultanée ou pas du tout, ou la banque court le risque de se retrouver avec une erreur de comptabilité.

Dans un annuaire, si deux personnes échangent leurs bureaux, on va d'abord mettre à jour le numéro de bureau pour une personne, puis ensuite pour une autre. Durant un court instant l'annuaire pourra donc montrer le même numéro de bureau pour les deux personnes. Cependant, cette situation et considérée comme acceptable car les données ne sont pas souvent mises à jour.

LDAP est aujourd'hui supporté par un grand nombre d'applications tels que les logiciels de courrier électronique pour la recherche d'adresse email (netscape, eudora, outlook...), les systèmes d'exploitation pour l'authentification des utilisateurs ou des machines (linux, unix, windows, samba...), etc. L'accès à des services LDAP est également supporté par un grand nombre de langages de programmation (C, C++, php, Java, Delphi...). LDAP devient *de facto* une solution bien adaptée pour mettre en place une authentification et un contrôle d'accès centralisé pour les différents systèmes et logiciels utilisés dans les organisations de grande taille.

Parmi les produits proposés actuellement pour mettre en place un annuaire ldap, on peut citer Iplanet Directory (Sun), Active Directory (Microsoft), openldap (open source)

7.1.2 Les bases de données relationnelles

Les bases de données relationnelles constituent de loin aujourd'hui le stockage

d'information le plus populaire. On en trouve partout, sous différentes formes, optimisées pour un plus ou moins grand nombre d'accès concurrents, pour la vitesse de réponse, pour l'intégrité des données. Il existe de nombreux produits commerciaux ou open source répondants aux besoins les plus variés.

Cependant, la philosophie sous-jacente à tous ces produits reste la même. Chaque base de données est composée d'une ou plusieurs tables. Chaque table comporte un ou plusieurs champ de données (numérique, texte, binaire....) qui peut s'apparenter à une colone dans un tableur, chaque enregistrement dans la table correspondant à une ligne de donnée pour laquelle on met des valeurs pour chaque colonne.

Les différentes tables et les différents champs sont ensuite mis en relation les uns avec les autres. On pourra par exemple avoir une table d'utilisateurs, chacun d'eux ayant un identifiant unique. Cet identifiant pourra alors être repris dans une autre table qui servira à référencer les messages envoyés. La base de donnée est ensuite interrogée par l'intermédiaire d'un langage spécifique dont le plus connu et le plus répandu est le langage SQL (Structure Query Language). Les requêtes permettent de combiner plusieurs tables et de poser des conditions afin de soustraire uniquement l'information désirée.

La difficulté de la mise en place d'une base de donnée relationnelle réside dans l'organisation logique des données. Ceci demande une analyse des informations à stocker de sorte que:

- "une donnée se trouve à un endroit et un seul". Le nom d'un client par exemple ne doit apparaître que dans un seul champ d'une seule table de la base de donnée, les autres tables utilisant des références à cet enregistrement. Ceci permet d'éviter les erreurs lors des mises à jour et les incohérences de données par la suite.
- les relations entre les tables soient de type "un enregistrement vers un enregistrement" ou "un enregistrement vers plusieurs enregistrements" mais jamais "plusieurs enregistrements vers plusieurs enregistrements". Si ce dernier cas survient, il est alors nécessaire de décomposer la relation en passant par une table supplémentaire.

Chaque logiciel de base de données propose ensuite des fonctions plus ou moins avancées qui permettent d'automatiser le traitement de certaines taches, de limiter la charge sur le serveur, d'accélérer certaines requêtes souvent utilisées par les clients. On peut citer:

- Les triggers: ce sont des morceaux de programmes qui sont exécutés lorsqu'une opération intervient sur le champ particulier d'une table. On peut par exemple imaginer un trigger chargé de supprimer toutes les références de prix, de stock, de description dans la base de donnée si une opération de suppression a été effectuée sur la table des produits proposés aux clients.
- Les fonctions: ce sont des fonctions que l'on peut programmer à l'intérieur de la base de données et qui peuvent ensuite être utilisées dans les requêtes. Leur rôle est généralement d'automatiser des taches répétitives. On peut cependant noter que, selon les produits, les fonctions ont des performances variés (bien souvent moindre que si l'on programme l'opération dans le programme formulant la requête) et que leur utilisation rend difficile la migration des données d'un produit à un autre. En effet, les fonctions qui ont été programmées dans un logiciel ne seront pas connues d'un autre et devront être re-programmées, peut-être de manière totalement différente.

- Les vues: Ce sont en fait des tables dynamiques qui combinent les données en provenance de plusieurs autres tables. Elles peuvent être interrogées et écrites de la même façon qu'une table ordinaire. Elle permettent bien souvent de simplifier les requêtes émises par le client, surtout lorsque celles-ci mettent en jeu plusieurs tables.

Il existe aujourd'hui un grand nombre de produits adaptés à différents besoins pour faire des bases de données relationnelles. Parmi les produits commerciaux on peut citer Oracle, 4D, Ingres (computer associates), SQL Server (microsoft), Access (microsoft). Parmi les produits open source on peut retenir MySql, PostgreSQL, mSql.

La plupart des langages de programmation (Php, Java, C...) proposent aujourd'hui un support pour accéder à toutes ces bases de données, soit directement, soit par l'intermédiaire d'une couche ODBC (Open DataBase Connectivity)

7.1.3 XML (fichiers / bases de données)

XML est un formalisme qui permet de définir des structures de document, des langages. Un document XML est un document au format texte composé de différents marqueurs (comme les marqueurs HTML) qui peuvent ou non avoir des attributs. Les marqueurs peuvent être vides (pas de données, pas de marqueur de fermeture), ou contenir des données (marqueur de fermeture). Pour illustrer notre propos, voici quelques lignes qui définissent un document XML conforme:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<document>
 <title>ceci est un document XML</title>
 <content>
  <para>ceci est un paragraphe</para>
  <media type=”photo” src=”images.png” />
 </content>
</document>
```

Un document XML peut également être déclaré valide s'il utilise et respecte une certaine grammaire. La grammaire définit la structure logique du document en déclarant les éléments qui peuvent être utilisés et comment ils doivent s'imbriquer les uns par rapport aux autres. Dans ce cas, le document XML déclare la grammaire qu'il utilise (le type de document XML. Le plus souvent, les grammaires sont définies dans des Document Type Definition (DTD). Il existe d'autre formalismes pour définir des grammaires, comme XML Schema par exemple, qui permet d'écrire une grammaire XML en XML.

Le document XML peut ensuite être lu par un analyseur (parser) qui permet de reconnaître sa structure, ses éléments, leur contenu et leurs attributs. On est alors en mesure de retirer l'information spécifique qui se trouve à un endroit donné du fichier. Il est donc tout à fait possible d'utiliser le format XML pour stocker du contenu structuré. La solution la plus simple consiste à stocker les données sous forme de fichiers sur un disque dur. Mais il est également possible aujourd'hui d'utiliser des bases de données spécialisées pour stocker des objets XML, chercher dans leur contenu et en extraire des données.

La majorité des langages de programmation propose en standard des fonctions pour travailler avec XML et incluent un parser. C'est notamment le cas de Java, PHP et ASP. Si le parser n'est pas proposé en standard, il y a fort à parier qu'il existe une implémentation disponible, qu'elle soit commerciale ou open-source.

Il existe également toute une palette d'outils permettant de créer des documents XML,

depuis un simple éditeur de texte ne vérifiant pas la syntaxe jusqu'à des produits évolués qui incluent dès le départ un certain nombre de grammaires très utilisées (docBook, MathML ...) et un validateur.

7.2 Formats de contenu.

Les applications que nous utilisons dans notre vie de tous les jours, notamment les navigateurs web, nous permettent de visualiser du contenu: texte, graphiques, images, vidéo, son... Lorsque le système qui délivre ce contenu devient conséquent et qu'il contient un grand nombre d'informations, il devient important, voire vital, de pouvoir gérer le contenu qu'il propose de façon rationnelle. La gestion de contenu se réfère donc à:

- la mise à disposition de ressources,
- la modification de ressources existantes,
- la réutilisation de ressources existantes pour créer de nouvelles ressources,
- la mise en relation des différentes ressources entre elles,
- la suppression de ressources qui ne sont plus à jour ou devenues inutiles,
- la visualisation des différentes ressources existantes, de leurs relations entre elles.

Pour que tout cela soit possible, le contenu doit être présenté sous une forme structurée qui permette d'en extraire le contenu lui même, sa description, mais également le contexte dans lequel il est mis à disposition. La structuration des données permet alors de faire des recherches sur leur contenu, d'en extraire un élément particulier, d'agencer différentes sources en un ensemble cohérent, voire de former des ensembles créés dynamiquement en fonction de données fournies par un utilisateur.

Il existe pour certains systèmes des formats propriétaires pour encapsuler le contenu. C'est le cas par exemple de la plateforme pédagogique WebCT. Dans cette situation, le contenu n'est alors transférable facilement qu'entre deux systèmes identiques. La réutilisation avec d'autres systèmes n'est pas forcement impossible mais peu demander une somme de travail conséquente.

D'autre part, certains formats sont destinés à transmettre non seulement les données mais également l'application qui va les utiliser. Il devient à ce moment là difficile d'extraire les données pour les utiliser dans un autre contexte.

Pour avoir accès à la description du contenu, deux solutions sont envisageables. La première consiste en l'insertion de méta-données qui sont transmises avec le contenu. Elles contiennent par exemple des mots clefs, le type de media mis à disposition, un résumé... C'est une solution assez simple puisqu'il suffit de récupérer les méta-données pour avoir une description du contenu. Le problème avec cette solution est que les méta-données doivent être générées à la main par l'auteur du contenu ou l'utilisateur qui les inclut dans son système. Si le contenu est modifié ultérieurement, les méta-données doivent être elles aussi mises à jour pour correspondre au contenu. Si cela n'est pas fait, il y a alors une disparité entre la description et le contenu lui même.

La deuxième solution consiste à distribuer le contenu sous une forme extrêmement structurée selon une grammaire stricte qui est distribuée avec le contenu lui même. La structure exacte du contenu étant alors connue, il est possible d'en extraire la description en passant en revue le contenu lui même. Ainsi, la description est automatiquement

synchronisée en cas de modification puisqu'elle est extraite de façon dynamique. Mais la multiplication des grammaires disponibles pose problème. Il est donc important que le contenu s'appuie sur des grammaires standards et largement usitées, dont la pertinence et la signification des éléments ont été publiées. D'autre part, l'obligation de passer en revue le contenu dans son entier peut requérir des ressources informatiques non négligeables. Ce dernier inconvénient peut être contourné en générant les descriptions une seule fois, à l'occasion de l'enregistrement du contenu dans un système ou de sa modification.

Cette deuxième solution peut être mise en oeuvre avec des documents au format XML. Il existe aujourd'hui plusieurs grammaires stables, largement utilisées permettant de structurer du contenu. Chacune permet de répondre à différents besoins. Nous allons essayer de passer en revue les différentes grammaires qui pourraient s'avérer utiles pour TECFA.

7.2.1 XHTML

XHTML (extensible hypertext markup language) est une version revisitée de HTML pour la rendre conforme à la norme XML. Cette spécification du *World Wide Web Consortium (w3c)* a été conçue pour éviter les problèmes qui ont surgi avec l'utilisation massive de HTML lors du grand boum du Web. En effet, une grande majorité des pages HTML utilisent les marqueurs non pas pour structurer les documents de façon logique mais pour en effectuer la mise en forme. Ainsi, la signification d'une balise "titre niveau 1" (<h1>) par exemple se transforme en "écrire en gros caractères". XHTML règle le problème en séparant clairement la structure logique du document de sa mise en forme. Toutes les références à l'aspect visuel du document ont été retirées du langage pour être insérées dans une feuille de style associée. Le fait qu'un paragraphe soit centré par exemple ne figure plus dans le contenu lui même.

XHTML est adapté pour fournir du contenu à des navigateurs Web. La plupart des logiciels de création de page web supporte aujourd'hui XHTML en parallèle avec HTML. XHTML est également supporté par la grande majorité des navigateurs Web récents. Pour les plus vieux, l'affichage, sans rester conforme au résultat souhaité, reste en général satisfaisant pour accéder au contenu.

7.2.2 DocBook

DocBook est un format dédié à l'écriture de livres et d'articles. Ce format à d'abord été développé spécialement pour l'informatique mais il est aujourd'hui utilisé par tous types d'auteurs et s'est hissé au rang de standard industriel pour l'écriture de documentation technique. Pour donner un exemple des plus concrets, ce mémoire que vous êtes en train de lire a été réalisé en utilisant cette grammaire. DocBook est issu du monde SGML, un précurseur de XML, plus élaboré et plus difficile à mettre en oeuvre. Cette grammaire est très largement utilisée dans le monde de l'industrie et son adaptation à la norme XML apporte chaque jour de nouveaux adeptes. La traduction de DocBook en XML a été initiée par Norman Walsh. Aujourd'hui, cette grammaire évolue et de nouvelles versions voient le jour au fur et à mesure des changements. Cependant, l'évolution se fait de telle sorte qu'il y ait une compatibilité ascendante, c'est-à-dire que les documents réalisés avec une version de DocBook soient compatibles avec les versions ultérieures.

La spécification DocBook contient en elle même deux formats: un format simplifié destiné à écrire de simples articles et un format exhaustif et complet pour écrire des

documents plus gros. La spécification de DocBook est disponible en téléchargement (http://docbook.org) et peut être utilisée avec tout éditeur XML sachant valider un document. Elle est également distribuée avec certains traitements de texte compatibles XML (tels que Adobe Framemaker utilisé pour la rédaction de ce mémoire[1])

7.2.3 DITA

Le *Darwin Information Typing Architecture (DITA)* est une architecture basée sur XML pour écrire, produire et diffuser de l'information technique (IBM Corporation, 2002). Cette architecture, basée sur XML, définit un cadre pour créer des modules d'informations réutilisables constituant des thèmes (topic). Un thème est une unité d'information qui décrit une tâche, un concept ou une référence unique. La catégorie d'information (concept, tâche ou référence) constitue le "type" d'information (ou infotype) du thème et en définit la structure. De plus, DITA inclut la notion de vocabulaire spécialisé par domaine. Il est ainsi possible d'utiliser des éléments de description relatif à un domaine particulier et ce quel que soit le type de thème utilisé. La figure 15 fait une synthèse visuelle de la base de cette architecture.

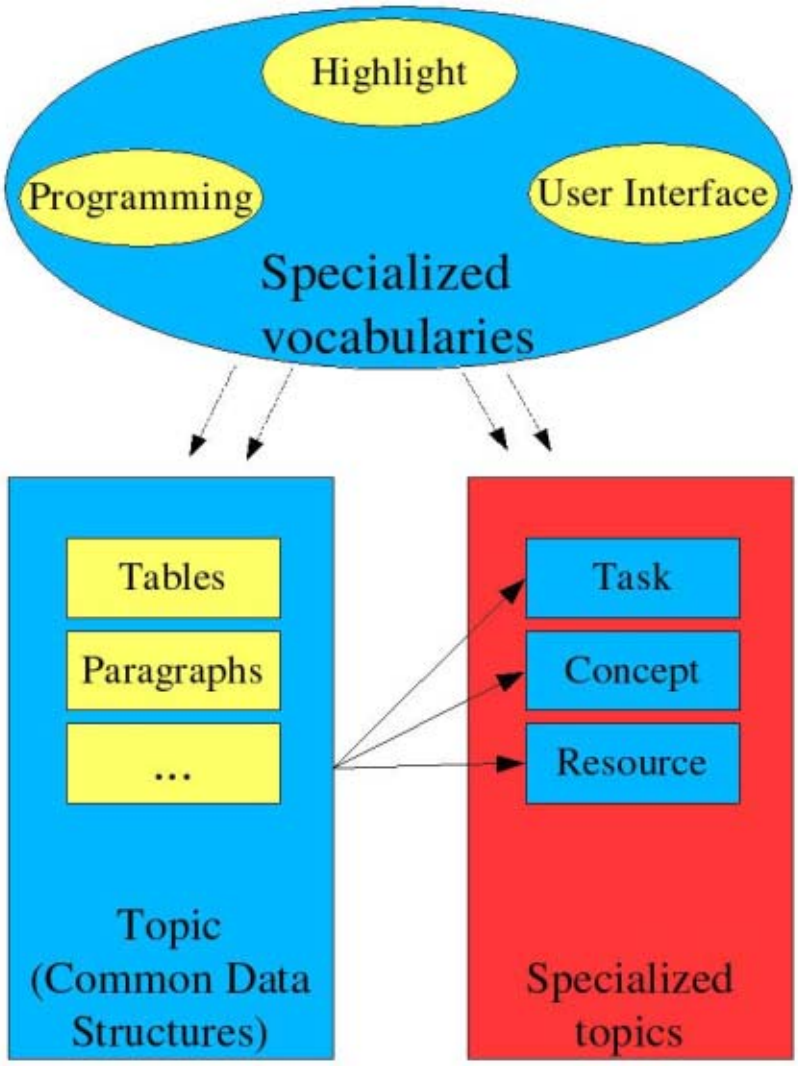
La force de DITA se révèle en fait dans les mécanismes prévus pour la spécialisation. En effet la grammaire de base à été conçue de telle sorte qu'il est possible de l'étendre à souhait pour l'adapter à des besoins particulier. La spécialisation est possible pour les thèmes et pour les vocabulaires de domaine. Ainsi, un auteur pourra décider de créer un thème spécialisé pour l'écriture de recettes de cuisine (qui pourrait être dérivé du thème "tâche" déjà existant ou directement du thème de base). De même, il pourra construire un vocabulaire spécialisé pour la cuisine qu'il pourra ensuite rendre disponible à tous les auteurs pour tous les thèmes spécialisés déjà existants.

La spécialisation se fait en construisant des extensions de grammaires par dessus celles qui sont déjà existantes de sorte que tout objet nouvellement créé est une extension d'un objet existant. Ainsi, une "recette de cuisine" spécialisée à partir d'une "tâche" pourra être mise en forme ou analysée avec les mêmes règles que l'objet parent (tâche) s'il n'existe pas de règles spécifiques pour ce thème particulier.

DITA, contrairement à d'autres grammaires (comme XHTML ou DocBook) permet d'organiser les données au niveau sémantique plutôt que typographique. L'organisation en thème de petite taille permet leur réutilisation dans différents contextes.

1.FrameMaker distribue en fait la version originale de DocBook en SGML. La version utilisée pour ce mémoire est celle écrite en XML, ce qui a demandé certaines adaptations.

Fig 15.Structure de base de l'architecture DITA



7.2.4 SCORM

Le *Sharable Content Object Reference Model (SCORM)* est né d'une coopération entre différentes institutions afin de faire émerger un standard pour les objets d'apprentissage. Ce standard définit pour les environnements d'apprentissage basés sur le Web un "modèle d'agrégation de contenu" et un "environnement d'exécution" (ADL, 2002). Pour résumer, cette initiative cherche à rassembler les producteurs de contenus, les développeurs de produits auteurs et les fournisseurs de plateformes pédagogiques autour d'un même standard qui permette ensuite de pouvoir échanger et réutiliser les différents objets d'apprentissage sans se soucier de la plateforme sur laquelle ils sont déployés.

SCORM s'appuie sur le concept d'actif (asset) qui constituent la forme la plus élémentaire du contenu d'apprentissage, c'est-à-dire de représentations électroniques de médias, de texte, d'images, de séquences sonores, de pages web, d'objets d'évaluation ou d'autres éléments d'information qui peuvent être envoyés à un client Web. Un actif peut être décrit au moyen de méta données d'actif (Asset Meta-data) permettant la recherche et le repérage dans des dépôts de données en ligne, ce qui accroît les possibilités de réutilisation. Les actifs sont ensuite regroupés sous forme d'objets de contenu

partageable (Sharable Content Object, SCO). Le SCO représente le niveau de granularité le plus fin des ressources d'apprentissage pouvant être suivi par un système de gestion de l'apprentissage (Learning Management System, LMS) avec un environnement d'exécution SCORM. Les différents SCO sont ensuite liés entre eux par un mécanisme d'agrégat de contenu qui permet d'ordonner ces modules dans un cheminement pédagogique.

La spécification SCORM est aujourd'hui disponible en version 1.2. Les différents acteurs ont à leur disposition un certain nombre d'outils tels que des environnements d'exécution génériques ou des suites logicielles pour tester la conformité des contenus créés. Certaines vendeurs de plateformes ont adapté leurs produits pour les rendre conformes à la spécification.

Cependant, il semble largement admis aujourd'hui que si SCORM s'adapte particulièrement bien pour des objets d'apprentissage destinés à la formation professionnelle, son architecture ne soit pas suffisamment évolutive pour répondre aux besoins de l'éducation secondaire, notamment les universités (McLean, 2001). C'est pourquoi d'autres projets voit le jour aujourd'hui, tels que la *Open Knowledge Initiative (OKI)* du *Masachuset Intitute of Technology (MIT)*.

7.3 Langages de programmation / scripting

Les langages de programmation permettent de réaliser l'interface utilisateur en accédant aux différentes sources d'information possibles. Ils permettent de définir la logique d'une application en appliquant des algorithmes sur des données. Il existe toute une variété de langages de programmation répondant, encore une fois, à différents besoins.

Les langages peuvent être classés dans différentes catégories, non exclusives, selon l'angle de vue sous lequel on les considère. On pourra ainsi parler de langages fonctionnels ou procéduraux, haut ou bas niveau, orientés objets... Pour notre analyse, nous nous contenterons de deux catégories distinctes qui s'opposent:

- Les langages de script conçus pour être inclus dans des documents HTML (*HTML embedded script*)[1]. Ils sont largement répandus, souvent faciles d'accès pour les débutants et posent peu de problèmes de compatibilité car ils sont pour la plupart exécutés du côté serveur. Les clients sont des navigateurs Web standard, capables d'interpréter des documents conformes aux différentes versions de HTML ou XML. Les langages de script constituent la base de la plupart des applications Web.

1.Le terme général de *langage de script (scripting language)* couvre un vaste domaine qui dépasse de loin les scripts inclus dans du HTML. Perl ou Python par exemple sont largement utilisés encore aujourd'hui dans le monde du Web pour fournir du contenu dynamique. Nous n'aborderons cependant pas ces langages dans cette étude car leur utilisation est plus complexe pour les débutants. En fait, les mécanismes d'utilisation sont les même (le serveur web appelle un programme extérieur pour obtenir le contenu à renvoyer au client), mais ces langages sont plus structurés et permettent d'écrire de véritables applications indépendante. Ils sont utilisés sur un grand nombre d'application mais ils sont moins "populaire" et mis en oeuvre par un public assez professionnel.

- Les langages “stand-alone”, conçus pour programmer des applications indépendantes. Ils permettent de fabriquer de véritables interfaces graphiques, des serveurs spécialisés, des modules pour d’autres applications Web ou locales. Ils sont souvent plus contraignant dans l’écriture du code, plus difficiles d’accès pour les débutants. Les problèmes de compatibilité peuvent être épineux, notamment pour les interfaces graphiques qui peuvent être amenées à fonctionner sous différentes architectures (Windows, Unix, Mac). Ils permettent cependant de dépasser les limites parfois contraignantes du protocole HTTP dans les applications Web, notamment pour les outils collaboratifs nécessitant de la communication en temps réel.[1]

Cependant, certains langages de script *HTML embedded* évoluent aujourd’hui vers des langages stand-alone. En effet, certaines applications Web deviennent tellement complexes que les pages de script ne comportent quasiment plus aucun marquage HTML. D’autre part, certains de ces langages proposent maintenant des librairies pour fabriquer des interfaces graphiques en exécutant le code sur la machine cliente (PHP-Gtk par exemple).

7.3.1 Les langages de script HTML embedded

Les langages de script HTML embedded on tous une particularité commune: ils sont insérés directement dans des pages de données HTML en utilisant des marqueurs spéciaux. Les morceaux de code ainsi ajoutés sont remplacés de façon dynamique par le résultat de leur exécution lorsque la page est servie au client Web.

Le code est exécuté du côté du serveur. Ceci implique la présence d’un serveur Web pour distribuer ces pages dynamiques, l’exécution du code étant déléguée à un programme externe. Ce programme externe peut être installé soit comme un programme CGI (Common Gateway Interface), soit comme module dynamique chargé lors du démarrage du serveur web, ou encore directement compilé dans le serveur Web lui même. Selon les produits, plusieurs types d’installation sont disponibles. Le choix dépend alors des contraintes de sécurités, de performance, d’utilisation de la mémoire du serveur.

Pour les grosses applications supportant un grand nombre d’utilisateurs, certains langages de script proposent, nativement ou sous forme de modules séparés, des systèmes de caches permettant de réduire la charge du serveur. Lors de la première requête pour une page, celle ci est exécutée et le résultat, en plus d’être fourni au client, est stocké dans un cache temporaire. Lorsque cette page est demandée à nouveau, c’est la copie mise en cache qui va être servie ce qui évite de faire travailler le serveur inutilement. Bien entendu, il existe des mécanismes permettant de savoir si la copie mise en cache est encore à jour ou si une nouvelle exécution est nécessaire afin d’actualiser son contenu.

Il existe aujourd’hui plusieurs langages de script HTML embedded qui se disputent le marché des applications Web dynamiques. Nous allons passer les plus populaires en revue en essayant de donner leurs caractéristiques principales.

1.Le protocole HTTP (Hyper Text Transfert Protocol) permet uniquement des connections discontinues. Le navigateur client fait une requête au serveur, le serveur envoie une réponse. La connection ne reste pas ouverte entre deux requêtes ce qui limite les possibilités en terme de communication client-serveur. De plus, le protocole seul ne permet pas d’identifier un client spécifique. Il faut passer par d’autres moyens (cookies, session...) pour conserver l’identité d’une connection dans le temps.

PHP

PHP est un langage open source qui a été conçu au début du Web par Rasmus Lerdorf comme *Personnal Home Page generator*. Aujourd'hui, la signification de cet acronyme a évolué et signifie *Hypertext Processor*. Plusieurs versions du langage se sont succédées. Au moment où nous écrivons ces lignes, nous en sommes à la version 4.2.3. Les évolutions ont touché plusieurs domaines sensibles tels que les performances, la sécurité, l'orientation objet...

Parmi les caractéristiques intéressantes de ce langage, nous pouvons citer les plus intéressantes dans notre contexte:

- PHP est open source. Il est donc gratuit et libre de droit, ce qui permet de l'utiliser dans toutes les circonstances, pour des applications commerciales ou non. Son développement est stable et suivi depuis plusieurs années.
- PHP peut être installé sur un grand nombre de systèmes d'exploitation (Windows, Unix, Linux, MacOS X...) et de serveurs Web (Apache, Microsoft IIS, Netscape...).C'est un grand avantage pour écrire du code portable dans différents environnements logiciels. De plus, son installation et son intégration à un serveur Web sont de plus en plus aisés, notamment sur les plateformes Windows, ce qui permet à des personnes ne possédant pas de connections permanentes au réseau de l'installer localement pour développer leurs applications.
- PHP possède un excellent support pour l'accès aux bases de données (MySQL, mSQL, Oracle, Microsoft SQL Server, Sybase, FilePro, ODBC...). Il est même possible aujourd'hui d'accéder à ces serveurs soit par des fonctions propriétaires d'une base de données particulière, soit par une couche d'abstraction supportant différentes bases de données. Cette dernière solution est encore un avantage pour écrire du code portable à différents environnements informatiques.
- En plus de l'accès aux bases de données, PHP supporte également un grand nombre de librairies: fonctions Web (cookies, authentification, sessions, redirection), LDAP pour l'accès aux annuaires, XML/DOM pour travailler avec des documents structurés, création dynamique de différents types de fichier multimedia (PDF, Gif, Jpeg, PNG, Flash...), compression de données (Bzip, Zip...), accès aux fonctions systèmes (fichier, exécution de programmes...), etc.
- PHP est facile à apprendre pour les débutants qui peuvent assez rapidement créer des pages pour traiter des formulaires, accéder à des bases de données. Cependant, cette simplicité relative tend à disparaître dans les évolutions récentes du langage au profit d'une sécurité accrue et d'une rationalisation des échanges de données entre les pages. Mais PHP est également un langage qui permet aux utilisateurs experts de développer des applications bien structurées, orientées objet et évolutives.
- PHP est aujourd'hui utilisé de façon assez massive dans le monde. Il existe de nombreux tutoriels en ligne. La documentation officielle est très exhaustive et assez bien conçue. La communauté des utilisateurs de PHP reste assez fidèle à la philosophie open source du produit et il existe un grand nombre de sites sur lesquels on peut échanger du code, ainsi que de nombreux développements open sources eux-

mêmes tels que des forums, des portails... Tous ces développements existants peuvent servir d'exemple pour l'enseignement, de point de départ à des futurs développements ou comme modules pour des développements en cours. Un grand nombres d'outils professionnels pour la création de pages Web supporte aujourd'hui la syntaxe PHP pour la création de pages dynamiques (GoLive, DreamWeaver...).

ASP

ASP, acronyme pour *Active Server Pages*, peut être considéré comme l'alternative commerciale de microsoft au langage PHP. La version 1.0 à été officiellement mise sur le marché en décembre 1996, intégrée au serveur Web IIS 3 (Programmers Resource, 2000). Le langage par défaut utilisé par ASP est le VBScript, basé sur le langage Visual Basic, également de la firme Microsoft. Il est cependant possible d'utiliser d'autres langages pour écrire les scripts (comme perl par exemple) sous réserve d'avoir installé les extensions appropriées. A l'heure actuelle, seuls le VBScript et une implémentation microsoft de javascript (JScript) sont disponibles en standard.

ASP est fourni comme module pour les serveur IIS sous Windows. Il n'est pas fourni pour d'autres plateformes ou serveur Web. Cependant, il existe 2 produits qui permettent d'exécuter des pages ASP sur d'autres architectures: *Chili!Soft ASP* de *Sun* (rebaptisé dernièrement *Sun ONE Active Server Page*s) et *Instant ASP* de *Halcyon Software*. Ces deux produits sont payants et relativement onéreux (environs 500 $ par serveur). De plus, il semble qu'il y ait des problèmes de compatibilité et que certaines applications développées avec la version originale d'ASP doivent être modifiées pour fonctionner correctement.

Après l'installation de base, ASP contient très peu de fonctions et s'appuie sur des composants pour en fournir de nouvelles. Il faut par exemple ajouter un composant pour pouvoir envoyer des emails, un autre pour pouvoir se connecter à des bases de données. Il est possible de développer ses propres modules avec différents langages proposés par microsoft (Visual Basic, Visual C++...)

Tout comme pour PHP, on peut trouver des tutoriels en ligne pour apprendre ASP. La documentation existe également. Elle est cependant disséminé sur plusieurs sources et il manque un document officiel qui permette d'avoir une vue globale du processus de développement, des différents packages utilisables etc.

Il existe des sites qui proposent des exemples de code. Les applications complètes en open-source sont beaucoup plus rares. A titre d'exemple, un site renommé pour le développement open source, Source Forge (www.sf.net), recensait le 1 octobre 2002 parmi tous les 48179 projets qu'il hébergeait 281 développements en ASP contre 4907 en PHP.

ASP est aujourd'hui assez largement employé dans le monde avec le gain en place de marché du serveur IIS sur lequel il est installé par défaut. De par son manque de documentation exhaustive et sa dépendance sur de nombreux modules différents pour accéder à des fonctions de haut niveau, il est relativement difficile de maîtriser ASP sans une aide extérieure (voire un cours spécifique à choisir dans le catalogue de formations de microsoft)

JSP

JSP est l'acronyme de *Java Server Pages*. JSP fait parti des outils de la suite *Java 2*

Enterprise Edition (J2EE) pour l'environnement *Java* dans laquelle il a fait son entrée fin 1999 (Bergsten, 2001). L'environnement *Java* et les outils *J2EE* sont disponibles gratuitement sous licence Sun. Le langage Java est né d'un effort de développement de la société Sun pour fournir aux développeurs un environnement de programmation indépendant de la machine sur laquelle il est exécuté. C'est donc par essence un environnement multi-système (Unix/Linux, windows, MacOS X...).

La première condition logicielle pour pouvoir utiliser la technologie JSP est d'installer un environnement d'exécution Java (Java Standard Development Kit). Cela fait, il n'y a plus que l'embarras du choix quant au produit à utiliser. Il en existe un grand nombre, commerciaux ou non, qui ont chacun leurs particularités. On peut cependant distinguer deux types d'installation:

- soit comme module ou add-on pour un serveur Web existant (apache, IIS...). Dans ce cas, le serveur Web s'occupe de la communication avec les navigateurs clients et transmet les demandes à des pages JSP vers le module installé.
- soit comme serveur autonome. Dans ce cas, on obtient un serveur complet, qui peut également distribuer des pages HTML classiques et exécuter des pages JSP.

Certains logiciels permettent les deux types d'installation. C'est le cas du serveur le plus utilisé à l'heure actuelle *Tomcat*, développé au sein du projet *Jakarta* par le groupe *Apache* (http://apache.org).

JSP est un peu différent des deux langages de script que nous avons présenté auparavant. Une différence majeure réside dans la façon dont les pages sont exécutées. En effet, la spécification JSP a été construite par dessus la technologie des servlets Java. Sans entrer trop dans les détails, alors que les pages PHP ou ASP sont interprétées à chaque requête d'un client, les pages JSP sont compilées lors de la première requête en un véritable programme informatique qui va être chargé en mémoire vive dans le serveur. Cette page peut alors rester "vivante" entre les requêtes des différents clients, ce qui diminue le temps de calcul, l'utilisation du processeur, le temps pour servir les pages et permet l'interaction entre les différentes pages de l'application, même lorsqu'il n'y a aucune requête de client en cours.

Au niveau de la programmation, on retrouve les mêmes choses que précédemment: des scripts peuvent être insérés à l'intérieur d'une page HTML. Les script sont écrits en Java. Cependant, JSP va plus loin que le simple scripting et permet de séparer les taches entre les personnes qui fabriquent les pages de contenu et les programmeurs. En effet, il est possible de développer des librairies de code très complexes et de haut niveau (développeurs) qui sont ensuite utilisables dans les pages JSP en faisant appel à de simples tags XML (designers).

Pour illustrer notre propos, imaginons la situation suivante: les designers souhaitent avoir la possibilité d'inclure dans leur page les résultats des étudiants. Ces résultat peuvent être présentés sous forme de listes ou de tableaux, inclure les notes intermédiaires de chaque examens ou donner seulement la moyenne, et enfin être donnés pour tous les cours suivis ou pour un seul cours particulier. Trouver tous ces résultats demande d'accéder à un annuaire pour identifier les étudiants et à une base de données relationnelle pour trouver les différentes notes. En PHP ou en ASP, il n'y a pas d'autre solution que d'écrire le code dans la page qui va traiter ces informations. Avec JSP en revanche, une équipe de développeurs peut travailler sur le problème, écrire un programme prenant les différentes possibilités en compte et le distribuer aux designers

par l'intermédiaire d'une librairie de tags (TagLib). Il ne reste alors plus qu'aux designers à déclarer la taglib et à connaître la syntaxe du tag particulier qu'ils veulent utiliser. L'insertion d'un tableau de résultats, qui aurait demandé l'écriture de plusieurs dizaines de lignes de code, la connaissance d'un grand nombre de fonctions pour accéder aux annuaires et aux bases de données peut devenir aussi simple que:

```
<%@ taglib uri="/customlib" prefix="custom" %>
 <custom:studentResults>
  <custom:param name="studentId" value="13" />
  <custom:param name="display" value="table" />
  <custom:param name="course" value="all" />
 </custom:studentResults>
```

En ce qui concerne l'apprentissage, JSP est relativement complexe à apprendre. Cependant, dans un contexte où les équipes de designers et de développeurs sont clairement séparées, il peut constituer un choix opportun et faciliter le processus de développement.

Enfin, il existe des mécanismes permettant d'encapsuler les applications JSP dans un seul fichier *Web Archive* (.war). Ce fichier contient tous les éléments utilisés par l'application ainsi que des fichiers stipulant comment le package doit être installé. Il suffit alors de déposer ce fichier sur un serveur compatible JSP et l'application sera automatiquement installée et mise à disposition des utilisateurs.

Autres produits

Notre but n'est pas ici de faire une liste exhaustive de tous les produits qui permettent de faire du scripting de pages Web. Nous avons voulu citer les principaux qui pouvaient représenter un certain intérêt dans le cadre de notre étude. Il existe bien d'autres produits, open-source ou commerciaux qui permettent de fournir du contenu dynamique. Parmi ceux que l'on a pas cité, on peut signaler:

- *ColdFusion* de *Macromédia*: une solution intégrée et propriétaire pour fournir du contenu dynamique. Elle contient des composants serveur et un éditeur de code intégré. Compatibilité avec toute la suite logicielle macromédia (Flash, Director, DreamWeaver....). http://www.macromedia.com/software/coldfusion/
- Moto, un langage open source développé par David Hackim. Il s'agit d'un module pour apache. A l'heure actuelle, il est uniquement disponible pour Apache sous linux/unix. Ce langage permet deux modes d'exécution: interprété ou compilé. Il permet également, un peu à l'instar de JSP, de distribuer des applications qui ont été empaquetées dans un seul fichier. Ce fichier se comporte alors comme un module du serveur web, sans besoin d'avoir l'environnement moto installé.

7.3.2 Les langages "stand-alone"

Comme nous l'avons indiqué dans l'introduction de ce chapitre, les langages "stand-alone" permettent de programmer des applications à part entière. Contrairement aux langages de script, le code n'est pas inclus dans des pages HTML et ne s'appuie pas sur un environnement d'exécution intégré à un serveur Web. Les programmes ainsi développés sont exécutés directement sur la machine qui va les recevoir.

Pour toute ces raisons, ils offrent une plus grande liberté de programmation, notamment pour ce qui concerne le développement d'applications client-serveur. En effet, la

communication entre les différentes parties d'une application distribuée n'est plus limitée au protocole HTTP voir note 1. en bas de page 74 (limite imposée par la dépendance au serveur Web pour les langages de script). Il est alors possible de recevoir et de transmettre de l'information en continue, ce qui permet des interaction en temps réel entre les différents composants d'une application. C'est ainsi qu'on pourra, par exemple, développer des environnements en 3 dimensions et permettre l'interaction à distance entre plusieurs utilisateurs, proposer des logiciels de communication synchrone évolués, transmettre du son en temps réel... Ces langages permettent également de s'affranchir du navigateur Web pour le rendu final de l'application, laissant un plus libre choix pour l'interface graphique à proposer à l'utilisateur.

Le gain de liberté au niveau du développement s'accompagne cependant de difficultés techniques.

- La communication entre le client et le serveur devient plus complexe et doit être programmée au sein même de l'application
- La fabrication d'un interface graphique demande plus de connaissances que pour fabriquer une simple page HTML
- Le fait que le programme soit exécuté sur la machine cliente pose des problèmes de compatibilité entre différentes plateformes. Les tests de l'application doivent donc être plus poussés, effectués sur différentes architectures. Le code de l'application doit prendre en compte les différentes possibilités.

Bien que les développements effectués jusqu'ici à TECFA soient majoritairement composés d'application Web reposant sur des langages de script, il est important de ne pas fermer la porte de notre plateforme à ces différents langages. Ils comportent tous des outils permettant d'accéder à des bases de données ou des annuaires, de travailler avec du XML, de communiquer avec d'autres plateformes. Parmi les quelques langages déjà utilisés à TECFA, on peut citer:

- Java
- C/C++
- Delphi
- ...

7.4 Gestion et versions de code source

Les outils de gestions et de versions de code source permettent d'organiser le développement d'une application, de rationaliser les échanges entre différents développeurs travaillant sur le même projet et d'enregistrer l'historique des différents fichiers de code source. Il devient alors plus simple de retrouver les bugs introduits à un moment ou un autre du développement, des les corriger, de revenir à une version antérieure d'un fichier. Egalement, chaque développeur peut travailler sur sa propre copie d'une application, y effectuer des changements, transmettre ces changements aux autres développeurs travaillant sur le projet et mettre à jour sa propre copie avec les changements introduits par des tierces personnes.

Ces outils permettent aussi de modifier des programmes existants pour son propre usage tout en gardant la possibilité de profiter des améliorations effectuées sur le programme original.

Les fichiers de codes sources sont stockés dans un dépôt (repository) sous un format spécial. Les fichiers contiennent la totalité des changements effectués depuis la première version du fichier. On n'accède pas au dépôt de façon directe mais par l'intermédiaire d'un programme permettant de retirer la version souhaitée.

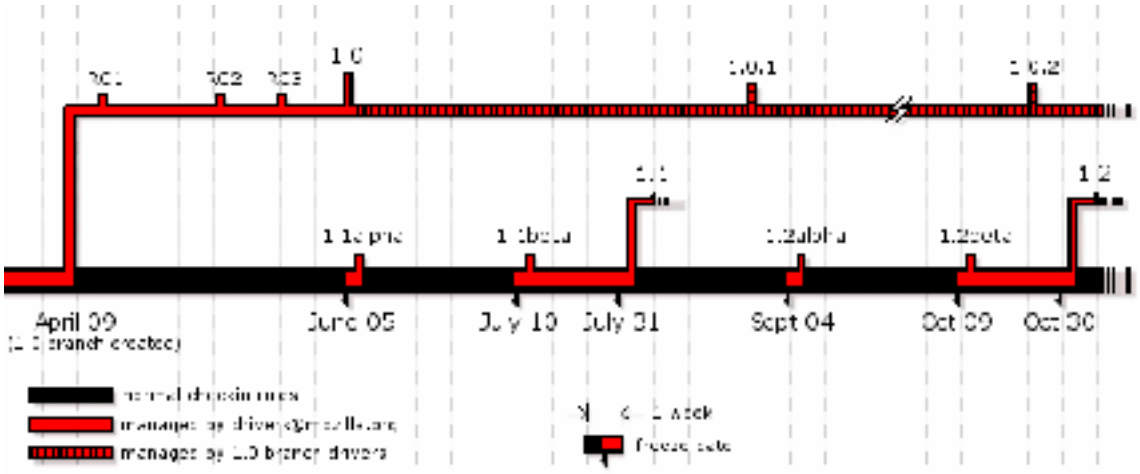
Un séance de travail typique avec ce genre de système se déroule comme suit:

- Le développeur retire une copie du code source depuis le dépôt (check-out)
- Il travaille sur cette copie localement, sur sa propre machine. Il effectue des changements, corrige des bugs, teste son code.
- En cours de travail, si cela s'avère utile ou nécessaire, il met à jour la copie locale de son code avec les changements apportés dans le dépôt par d'autres développeurs (update)
- Une fois son travail achevé, il soumet les modifications réalisées (commit) afin de les rendre disponibles pour les autres développeurs. Cette opération consiste à transmettre les modifications en elles mêmes au dépôt central ainsi qu'une courte description des changement réalisés. Cette description servira par la suite à passer en revu l'historique du fichier modifié.

Pour les administrateurs de projets, les outils de gestions et de version de code sources apportent encore d'autres fonctionnalités. Ils peuvent créer des "branches" de code source afin de pouvoir conserver dans le même dépôt des versions différentes de la même application tout en continuant à apporter des améliorations et corriger des bugs sur chacune d'elle.

Donnons un exemple simple pour illustrer cette possibilité. Une équipe de développeurs commence à programmer une nouvelle application. Au bout de quelques temps, le projet est suffisamment avancé pour qu'on décide de mettre à disposition des utilisateurs une première version du logiciel. L'administrateur crée alors une branche 0.1 qui fixe l'état de développement. Les développeurs continuent à faire évoluer le logiciel sur la branche principale (head). Après quelques temps, des informations commencent à remonter des utilisateurs et pointent quelques bugs dans la version qu'ils ont utilisée. Les bugs sont alors corrigés sur la branche 0.1, et une nouvelle version (0.1.1) est distribuée aux utilisateurs. Cependant, les corrections sont également intégrées à la branche principale ce qui permettra d'en profiter pour la prochaine version (0.2) qui sera proposée une fois que suffisamment de nouvelles fonctionnalités auront été apportées et que la stabilité de l'application sera jugée satisfaisante. La figure 16 illustre cet exemple en présentant les différentes branches du développement du navigateur Mozilla

Fig 16.Feuille de route du projet Mozilla (récupérée sur http://www.mozilla.org/roadmap.html)



Il existe aujourd'hui un grand nombre de logiciels permettant de faire du contrôle de code source. Les différences se situent au niveau de l'optimisation des performances, du système de stockage des fichiers dans le dépôt central, de la capacité à travailler sur des "paquets" de fichiers ou sur des fichiers seuls, du support pour le développement en équipe, de la sécurisation de l'accès. Notre but n'étant pas ici de faire une liste exhaustive de tous ces produit, nous nous attarderons plus particulièrement sur le plus populaire et utilisé d'entre eux: le *Concurent Version System (CVS)*. La grande majorité des projets de développement open-source utilise aujourd'hui ce système et offre en général un accès anonyme aux utilisateurs qui le souhaitent pour retirer ou mettre à jour leur version de l'application.

CVS est lui même un développement open source, disponible gratuitement pour un grand nombre de systèmes d'exploitation. Il est distribué de façon standard avec la plupart des systèmes d'exploitation unix/linux. CVS est un outil en ligne de commande, ce qui peut le rendre difficile d'approche pour les débutants. Cependant, grâce à sa large utilisation, plusieurs clients graphiques ont été développés et permettent d'effectuer les opérations de base de façon simple:

- Retrait, mise à jour et soumission des sources vers le dépôt.
- Ajout et suppression de fichiers.
- Visualisation des différences entre les fichiers locaux et ceux du dépôt.
- Importation de nouvelles version fournies par des tierces parties.

Des modules disponibles pour le Web permettent de visualiser les fichiers d'un dépôt CVS sans disposer du programme sur sa propre machine. Des fonctionnalités supplémentaires permettent par exemple d'envoyer un mail à toutes les personnes qui participent à un projet dès qu'une modification à été soumise.

Le programme CVS de base est disponible sur le site http://cvshome.org. Les différents clients graphiques sont généralement gratuits tels que Cervisia (http://cervisia.sf.net), wincvs, maccvs et gcvs (http://cvsgui.sf.net). Différents clients installables sur un serveur web sont aussi disponibles.

7.5 Autres outils, autres besoins...

Nous avons dans ce chapitre étudié quelques outils de bas niveau qui permettent de répondre aux quatre besoins principaux au niveau technique pour notre environnement en ce qui concerne le stockage des données, le format du stockage des ces données, les langages de programmation et la gestion du code source pour le développement d'applications. Cet inventaire n'est pas exhaustif. De plus, bien d'autre besoin peuvent être identifiés ou apparaître, nécessitant la mise en oeuvre de nouveau outils ou d'autres technologies.

Le but n'était pas d'opposer ici les différentes solutions entre elles pour chaque catégorie de besoin et de choisir la meilleure. Chacune d'entre elles propose des fonctionnalités qui peuvent être intéressantes dans un contexte donné et peuvent donc se révéler bien adaptées pour un enseignement, le développement d'une application particulière. Si TECFA veut rester à jour par rapport aux dernières innovation en matière de technologies d'éducation, l'unité doit garder un esprit d'ouverture. Par conséquent, l'environnement de base qu'elle utilise pour ses développement doit garder dans la mesure du possible la possibilité d'intégrer plusieurs solutions utilisables.

8 Proposition d'architecture et directives de développement

Maintenant que nous avons passé en revue les différents types d'interactions pédagogiques à supporter, les produits existants, que nous avons établie un cahier des charge et exploré de façon succincte les différents moyens techniques à notre disposition pour la mise en oeuvre, nous pouvons faire l'ébauche de l'architecture du nouveau dispositif à mettre en place et des directives pour son développement et son utilisation. Notre proposition s'articule autour de 6 axes qui répondent chacun à des besoins que nous avons mis en évidence en dressant le cahier des charges:

- Faciliter l'authentification des utilisateurs et leur gestion, centraliser l'information
- Fournir un outil de base pour la dissémination de l'information et de la communication
- Fournir une couche logicielle de base pour le développement des applications et l'accès à la base de données utilisateur
- Gérer le développement et les versions des différents projets, permettre le travail en groupe pour le développement d'une application
- Donner des règles de bonne pratique pour l'écriture du code et son commentaire
- Générer et garder à jour la documentation de la couche logicielle de base et des différents projets développés.

8.1 Gestion utilisateurs, authentification: un annuaire centralisé.

Nous avons mis en évidence à plusieurs reprises le besoin d'authentification pour les utilisateurs de notre dispositif ainsi que l'importance d'une source unique pour stocker les informations qui les concerne et y accéder. C'est pourquoi notre première proposition concerne la mise en place d'un annuaire LDAP centralisé pour tous les utilisateurs de la plateforme.

L'utilisation de LDAP offre plusieurs avantages dont certains dépassent le cadre de cette étude. Tout d'abord, comme nous l'avons vu précédemment (voir chapitre 7.1.1 page 64), LDAP de part ces caractéristique constitue une solution parfaitement adaptée au stockage de ce genre d'information stable dans le temps. Le nombre de produit supportant LDAP augmentant de façon constante, ce choix nous permet de garder une large ouverture en ce qui concerne les différentes solutions que l'on pourra utiliser.

Le polymorphisme des objets LDAP nous permet d'utiliser des schémas d'objets standards, déjà connu et utilisés, qui fourniront la base des attributs dont nous avons besoin pour décrire nos utilisateur. Au besoin, ces schémas pourront être étendus par d'autre schémas, récupérés ou développés pour un besoin particulier.

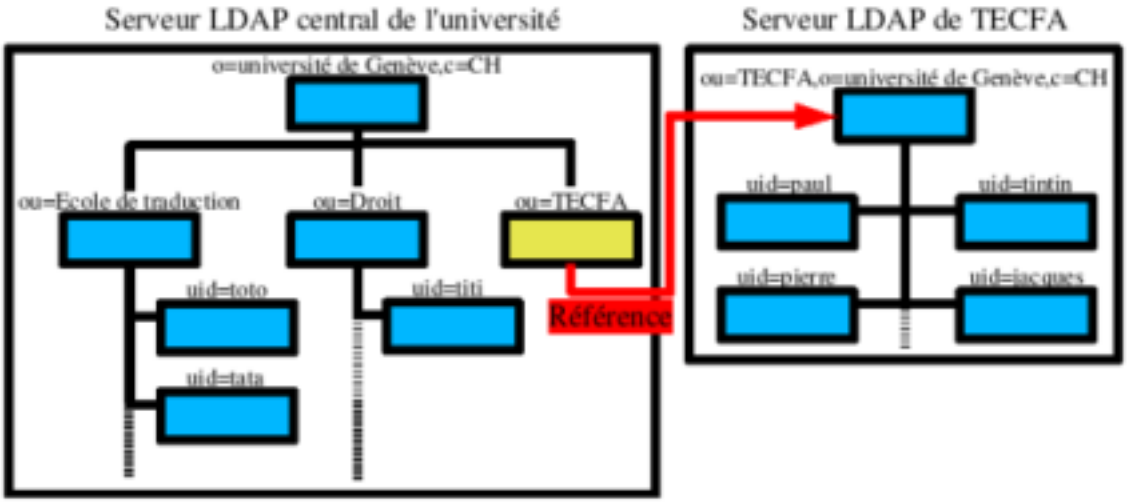
Par exemple, tous les attributs permettant de donner une adresse email, un numéro de bureau, construire des listes de diffusion... existent déjà et sont regroupés au seins de types d'objets qu'il ne risque plus qu'à utiliser. Ces schémas étant largement utilisés et hissés au rang de standard, ils sont donc reconnus par un grand nombre des applications qui savent déjà utiliser LDAP comme les logiciels de courrier électronique qui sauront,

sans aucune modification à apporter, rechercher et récupérer les adresses email des utilisateur pour pouvoir leur adresser des messages. Ceci n'empêche en aucun cas de développer un schémas spécifique à TECFA (type d'objet et éventuellement attributs) pour décrire un étudiant ou un collaborateur, sa page personnelle sur le Web, la promotion de laquelle il est issu, les cours qu'il suit ou qu'il donne...

De façon plus large, LDAP pourrait permettre une intégration et une collaboration avec d'autre services de l'université. Tout d'abord, LDAP pourrait permettre à terme d'authentifier les utilisateurs sur les différentes stations de travail de l'université. En effet, tous les systèmes d'exploitation utilisés sont configurables pour utiliser LDAP comme base de données utilisateur (Unix, Linux, Windows 2000/XP avec Samba, MacOS X) à l'exception de quelques vieux systèmes en voie de disparition (Win9x, MacOS 9 et inférieur).

D'autre part, LDAP offrant la possibilité de répartir un arbre de données sur différents serveurs, il est tout à fait envisageable que TECFA possède son propre serveur qui s'intègre avec celui de l'administration centrale. La figure 17 illustre de façon fictive comment cette intégration pourrait être mise en oeuvre. Cette solution permettrait que tecfa garde le contrôle de ses utilisateurs tout en limitant le nombre d'informations qui sont dupliquées.

Fig 17.Répartition de l'arbre LDAP de l'université de Genève sur plusieurs serveur (exemple fictif)



8.2 *Un portail d'information pour la communauté Etudiants-Enseignants.*

Au cours de l'évolution du dispositif dans le temps, le besoin pourra se faire sentir de développer des outils spécifiques pour la communication, la dissémination d'information, la gestion de document. Cependant, ces fonctionnalités de base peuvent être fournies dans un premier temps par l'installation d'un portail d'information simple (pour une revue des fonctionnalités fournies par les portails voir voir chapitre 5.1 page 49). Vu le nombre de produits disponibles aujourd'hui, il n'y a que l'embarra du choix.

Nous préconisons de choisir ce portail parmi les produits open source, ce qui permettra

éventuellement d'adapter le produit à nos besoins particuliers, notamment à l'utilisation de LDAP comme moyen d'authentification. D'autre part, il serait également judicieux de choisir un produit conçu de telle façon qu'il soit possible de l'étendre.

En effet, un certains nombres de portail propose aujourd'hui une interface de programmation de l'application (*Application Programming Interface, API*) permettant de programmer des modules qui seront insérés. Le choix d'un tel produit permettra d'enrichir ses fonctionnalités de bases avec des développement menés à TECFA.

De plus, certains portails commencent à être aujourd'hui des produits relativement abouti qui se positionne sur le marché comme des moteurs d'affichage (display engine) pour du contenu. Leurs fonctionnalités de base sont réduite à un minimum et toutes les fonctionnalités supplémentaires sont fournies sous forme de modules. Il serait donc tout à fait possible à terme que l'environnement de développement d'un portail devienne la couche de base dont nous avons besoin. Cependant, ces évolutions sont trop récentes pour qu'on envisage de se baser dessus. En effet, les formats et les directives pour développer les modules diffère énormément entre les différents portails, ce qui ne permet pas la réutilisation des modules développées pour un portail dans un autre.

8.3 Couche logicielle de base pour le développement d'applications: une API pour TECFA

Le développement d'applications dans un environnement particulier conduit les programmeurs à utiliser souvent le même genre de routines ou à répéter des opérations spécifiques. Afin d'assurer une pérennité au dispositif que nous allons mettre en place et d'en faciliter la maintenance, il est indispensable qu'un certains nombre d'opérations de haut niveau soient utilisables à partir d'un librairie de fonctions et d'objet constituant une base solide et stable: une API

Si nous nous en tenons simplement à ce qui concerne les utilisateurs, chaque programmeur aura, une fois ou l'autre, besoin d'utiliser une fonction d'authentification, de rechercher une adresse email, de savoir à quel groupe de travail appartient un étudiant.

De plus, nous devons nous assurer que les données stockées dans le serveur LDAP sont utilisées de la même façon par tout le monde et que leur signification reste constante dans le temps et dans les différentes application développées. Il serait par exemple relativement ennuyeux qu'un développeur mal informé utilise l'attribut "promotion" pour stocker les années durant lesquelles un étudiant à été actif (e.g. 1996-1998) alors que cet attribut a été spécialement conçu pour recevoir le nom d'une promotion (e.g. A pour Alice ou H pour Heidi)

La mise à disposition d'une couche logicielle pour les programmeurs permet de constituer une couche d'abstraction pour la base de donnée qui intègre la logique des opérations à effectuer. Cela permet donc de régler les deux problèmes que nous venons de pointer du doigts: fournir des fonctions de haut niveau réutilisables et s'assurer de l'intégrité dans l'utilisation des données.

Fournir une couche d'abstraction apporte aussi un avantage pour l'évolutivité des applications. En effet, la couche est utilisée par l'intermédiaires de fonctions ou de méthodes. Les méthodes/fonctions acceptent des variables comme paramètres et retournent des valeurs. Du point de vue de l'utilisateur de la couche logicielle, le fonctionnement interne n'a aucune importance pourvu qu'il connaisse le but d'une

fonction/méthode, les paramètres qu'il doit lui envoyer et la signification de la valeur qui va être retournée.

Ceci implique que le fonctionnement interne de la couche d'abstraction peut être totalement modifié sans aucune conséquence sur les programmes existant à partir du moment ou l'interface pour y accéder reste stable. Pour illustrer par un exemple, la couche de gestion de utilisateurs programmée pour fonctionner avec LDAP peut très bien être re-écrite pour fonctionner avec une base de donnée relationnelle, voire même un simple fichier texte. A partir du moment ou les fonctions restent les même, acceptent les mêmes paramètres et retourne les même valeurs, les applications qui l'utilisent continueront de fonctionner sans aucun problème.

Pour dépasser le strict cadre de l'accès aux informations utilisateur, cette API doit être vue comme une véritable boite à outils du programmeur adaptée à l'environnement informatique et au champ d'application théorique de TECFA. Elle doit être complétée au fur et à mesure de l'évolution du dispositif pour fournir toujours plus de fonctions de haut niveau qui faciliteront le travail de développeurs. A terme, on peut imaginer que cette couche contienne un certain nombre d'outils génériques répondant à de nombreux besoins

- génération dynamique de diagrammes ou d'illustrations.
- couche d'abstraction réunissant l'accès aux différents types de bases de données relationnelles au sein d'une même interface,
- outils génériques pour la création de scénarios pédagogiques,
- outils génériques pour la collaboration
- outils génériques pour la gestion de documents
- etc.

La modification de l'API au cours de son évolution devra se faire de manière prudente. Les changement apportés ne doivent pas, dans la mesure du possible, modifier l'interface pour les fonctions/méthodes existantes afin que les développements faits à partir d'une version antérieure puisse fonctionner. Les outils supplémentaires qui seront ajoutés doivent être discutés en groupe par les différents développeurs impliqués. Ceci permet d'être sur que le concept de la finalité des outils est partagé par toute l'équipe, que ces outils répondent effectivement aux besoins des programmeurs et qu'il seront utilisés. De manière générale, il faut éviter la multiplication anarchique des outils dont certains tomberont rapidement dans l'oubli ou feront double emploie avec d'autre déjà existants.

8.4 Un dépôt central pour le code source et sa gestion

Nous proposons la mise en place d'un dépôt central de code source pour tout les développement menés à TECFA. Chaque projet devra y être ajouté, à commencer par l'API de base. CVS que nous avons présenté plus haut (voir chapitre 7.4 page 79) étant très largement répandu, gratuit et disposant de différents client graphiques, nous penchons en faveur de cette solution.

L'utilisation systématique d'un dépôt CVS centralisé apporte des avantages substantiels par rapport à l'enregistrement bruts des sources dans un système de fichiers comme cela était fait auparavant.

- Pour commencer, tous les projets de développement se trouvent au même endroit. Cette remarque peut paraître triviale. Cependant, qui n'a jamais perdu quelques minutes (voire quelques heures !) à la recherche d'un document dont on a oublié l'emplacement dans un système de fichier ? Le problème peut devenir très épineux quand il s'agit de reprendre le développement d'une application qui n'a pas été utilisée depuis longtemps et dont les différents fichiers se retrouvent disséminés dans plusieurs répertoires. La centralisation du dépôt nécessité seulement la mémorisation d'un emplacement unique à partir duquel on peut consulter une arborescence de projets.
- De plus, cet endroit centralisé contient non seulement le développement dans son dernier état connu, mais également tout l'historique des différents fichiers depuis leur création. Pour peu que les développeurs s'astreignent à fournir une description des modifications effectuées lors de chaque soumission de modification, il est possible par la suite de retirer un historique complet et intelligible de l'évolution d'un fichier particulier et éventuellement de mettre le doigt sur une erreur commise au cours de son évolution. La version actuelle d'un fichier et toutes ces modifications sont stockées dans un fichier unique au niveau du dépôt, ce qui fait gagner un espace disque considérable par rapport à une copie périodique du projet dans son ensemble avant chaque modification. Par ailleurs, toutes les modifications sont disponibles, alors que le système de la copie de sauvegarde géré à la main peut en masquer certains.
- CVS permet à plusieurs développeurs de travailler sur le même fichier source en même temps, alors que ceci est impossible si les fichiers sont directement stockés dans un répertoire. Dans le meilleur des cas, certains éditeurs vont poser un verrou (lock) sur le fichier qu'ils sont en train de modifier. Ceci permet d'éviter qu'un autre développeurs travaillant avec le même outils ouvre également ce fichier en écriture. Cependant, cette technique a ces limites (accès avec un autre outils ne reconnaissant pas le même mécanisme de lock) et ne permet en aucun cas des modification concurrentes. Avec l'utilisation de CVS, par contre, chaque développeur travaille sur sa propre copie de fichier et ne soumet que les modifications qu'il a effectuées vers le dépôt. Le client CVS qui permet d'accéder au dépôt gère l'application des différentes modifications en provenance de sources différentes. Les conflits sont relativement rares et mis directement en évidence au moment de la soumission. Il est alors possible de les résoudre.
- CVS permet le suivi de code source développé à l'extérieur de TECFA. Pour illustrer ceci, imaginons que nous ayons fait notre choix pour un portail d'information que nous allons modifier. Le code source du portail est téléchargé chez son fournisseur. Il est ensuite importé dans le dépôt CVS puis les développeurs commencent à effectuer leurs modifications. Lorsqu'une nouvelle version du portail est annoncé, il est légitime de vouloir profiter des améliorations effectuées par l'équipe de développement originelle sans pour autant perdre les modifications introduites localement pour des besoins spécifiques. Il suffit alors de télécharger cette nouvelle version et de l'importer à nouveau. Les fichiers qui n'ont pas été modifiés localement sont alors mis à jour avec les nouvelles versions. Pour les autres, les modifications sont intégrées tout en conservant le travail réalisé pour l'adaptation du portail. Encore une fois, il peut y avoir des conflits qui sont signalés lors de l'importation et peuvent ensuite être résolus.

- Enfin, même si l'utilité de cette fonctionnalité ne parait pas évident pour TECFA, CVS permet de maintenir différentes versions du même projet dans le même dépôt. Les programmeurs peuvent alors décider de fournir une version 0.1 d'une application en conservant la possibilité de modifier cette version pour corriger les bugs trouvés par les utilisateurs, de continuer le développement pour la version 0.2 et d'intégrer les changements effectués sur la version 0.1 à la version 0.2.

8.5 Hygiène de programmation et de développement

Si les différents outils à mettre en place constituent bien la base informatique de notre environnement, il faut bien garder à l'esprit qu'ils sont utilisés par des humains. Or, si la machine ne se trompe jamais (pas de sa propre initiative en tous cas...) et fait exactement ce pour quoi elle a été programmée, il n'en vas pas de même pour les hommes. La bonne utilisation et la bonne évolution du dispositif dépend non seulement de la mise en place d'outils robustes et adaptés aux besoins des programmeurs mais également, et en grande partie, de l'utilisation qui en sera faite.

C'est pourquoi il est à nos yeux extrêmement important que tous les développements se base sur des règles de bonnes pratique qui seront respectées par tous. Pour être accepté, ces règles devront être le fruit d'un consensus entre les différents developpeurs. Nous proposons ici un ensembles de conseils qui pourront servir de base à cette discussion.

8.5.1 Utilisation du dépôt CVS

L'ensemble formé par le dépôt CVS et les outils pour y accéder constitue un outil formidable... si l'on s'en sert et qu'on l'utilise correctement. Chaque personne étant amenée à l'utiliser devra se familiariser avec les différentes commandes ou explorer différents client graphiques pour en trouver un à sa convenance.

Chaque nouveau projet doit y être ajouté dès le tout premier stade de sa conception. Quand le travail se fait en groupe, chaque membre de l'équipe doit prendre l'habitude de mettre sa copie de travail à jour le plus régulièrement possible. Pour renforcer cette bonne pratique, il existe des logiciels permettant d'avertir (par mail le plus souvent) les différents membres d'un projet à chaque mise à jour dans le dépôt.

Il n'existe pas de règle absolue pour calculer la fréquence des soumissions vers le dépôt. Si le programmeur soumet trop souvent, l'historique de modification des fichiers peut devenir long et le risque de soumettre du code qui ne fonctionne pas correctement augmente. Ces erreurs peuvent ensuite poser des problèmes à d'autre membre de l'équipe lorsqu'il mettent à jour leur copie de travail.

S'il ne soumet pas assez souvent, les modifications entre deux version peuvent devenir trop importantes pour que, en cas de problème, on puisse facilement retrouver la source de l'erreur qui a été commise. Les amélioration ne sont pas transmise assez vite aux autres membres de l'équipe qui risque de ne pas les intégrer suffisamment tôt.

Il faut donc arriver à trouver un juste milieux, en essayant de garder à l'esprit les règles suivantes lors de chaque soumission:

- Lorsque plusieurs fichiers on été modifiés, les outils CVS permettre de soumettre tous les changement de tous les fichiers en une seule opération. Il est cependant préférable que chaque fichier soit soumis de façon séparé pour que les changement effectués soit décrit pour chaque fichier. On pourra faire des exceptions pour des modifications identiques dans un groupe de fichiers similaires par exemple.
- Il est possible de soumettre un changement sans en indiquer la description. Cependant, cette pratique est à éviter absolument. La description doit être systématique, être simple et concise, de sorte qu'elle soit compréhensible et que l'historique des modifications du fichier soit lisible.
- Enfin, chaque modification devrait être testé le plus soigneusement possible avant d'être envoyé vers le dépôt. Sauf cas extrêmement exceptionnels, les programmeurs devrait s'astreindre de soumettre du code qui ne fonctionne pas du tout.

Pour conclure, CVS est un outil qui permet uniquement de gérer le code source. Ce n'est pas un outil de gestion de projet, ni d'organisation du développement. Il est donc nécessaire, dans le cas de travail en groupe, que l'équipe de développement reste en contact, discute des choix de développement et que le travail soit organisé de façon à se que l'application ne se construise pas de façon anarchique

8.5.2 Choix des identifiants

Tout langage de programmation utilise des identifiant pour nommer ses différentes variables, fonctions, méthodes et objets/classes. L'identifiant est librement choisi par le programmeur (hors des mots réservés par le langage particulier qu'il utilise). Il est souhaitable cependant que les identifiants retenus respectent quelques règles pour que le code soit plus facilement lisible par des personnes extérieures.

- l'identifiant doit avoir une signification représentant la chose qu'il identifie. Pour que les choix restent cohérents à travers les différents fichiers et les différents projets, tous les identifiant devraient être choisi dans la même langue naturelle. Nous recommandons l'emploie de l'anglais.

  ➠ Pour une variable, il doit être significatif du contenu de la variable. Une variable contenant le nom de l'utilisateur pourrait par exemple être nommé *userName* plutôt que *chaîneDeCaractère* ou *variable1*. De cette façon, on peut savoir d'un seul coup d'oeil ce que représente la variable.

  ➠ Pour une fonction ou une méthode, il doit être significatif de l'action qui va être effectuée. Un identifiant *myFunction()* ou *fonction1()* par exemple nous renseigne assez peu sur l'opération que la fonction exécute, alors que *drawLoginBox()* nous laisse entrevoir au premier regard qu'il s'agit d'afficher une boite d'authentification à l'écran. Il existe des convention qu'il serait bon de respecter pour les méthodes chargées d'accéder au propriété d'un objet.

    ❑ Les identifiant des méthodes qui lisent la valeur d'un propriété qui n'est pas de type booléen, on utilise le préfixe *get* suivit du nom de la propriété. *getUserName()* par exemple retourne le nom de l'utilisateur contenu dans la variable *userName*.

- ❑ Pour les méthodes qui lisent des propriétés de type booléen, le préfixe utilisé et *is* suivit du nom de la propriété. *isStafStudent()* par exemple retourne la valeur vrai ou faux selon que l'utilisateur est un étudiant Staf ou non.
- ❑ Pour les méthodes chargées de modifier la valeur d'une propriété, on utilise le préfixe *set* suivit du nom de la variable. *setStafPromotion()* par exemple enregistre la valeur de la promotion d'un étudiant Staf dans la variable *stafPromotion*

➠ Pour un objet, l'identifiant doit donner une bonne idée de ce qu'il représente. *UserManager* par exemple nous renseigne sur un objet contenant des méthodes et des attributs pour la gestion des utilisateurs. *MyClass* ou *ClassA* au contraire sont vide de sens.

- l'utilisation de majuscules/minuscules ou d'un caractère de séparation pour les mots composés doit être la même dans les différents fichiers/projets afin que tous les identifiants soit construits de la même façon. Nous proposons de respecter la convention utilisée dans les programmes Java ou C/C++ largement répandue dans le monde informatique.

  ➠ Les noms des variables et des fonction/méthodes commencent par une minuscule. Chaque mot suivant commence par une majuscule. Exemples: *sort()*, *promotion*, *userName*, *setUserName()*, *addItemToFilter()*

  ➠ Les noms de classes/objets commencent par une majuscule, chaque mot suivants commençant également par une majuscule. Exemples: *User*, *UserManager*, *HtmlCaracterConverter*

8.5.3 Commentaires du code

Si le choix d'identifiants bien appropriés permet de rendre le code plus lisible et plus compréhensible, l'utilisation de commentaire permet encore de le rendre plus intelligible. Les programmeurs devraient s'astreindre à commenter leur code, même de façon succincte.

La plupart des langages de programmation acceptent deux types de commentaire: les commentaire en bloc (block comments) et les commentaire de ligne (inline comments).

- Les block comments sont généralement utilisés juste avant la déclaration d'une fonction, d'une variable ou d'un objet pour le décrire de façon précise. Voici un extrait de code PHP qui déclare une variable et une fonction avec des block comments.

```
/**
 * User name
 *
 * Contains the user name as it has been registered
 * in the database when he first registered
 * @var string
 */
var $userName;

/**
 * subscribes user to a group
 *
 * subscribes user identified by {@link $userName}
 * to a working group identified by {@link $groupId}
 * @param string userName The user's name
 * @param int groupId The group ID
 * @return boolean status TRUE on success
 * @see {@link http://tecfa.unige.ch/group.html group
listing}
 */
function addUserToGroup($userName, $groupId) {
 contactDatabase();
 $userId=findUserId($userName);
 return insertGroupID($userId, $groupId);
}
```

Cet exemple illustre l'utilisation d'un format de commentaire et de tags qui seront utilisés ensuite pour la génération automatique de la documentation de l'application. Nous aborderons ce point plus loin (voir chapitre 8.6 page 92).

L'écriture de block comments avant la déclaration d'un objet, d'une fonction ou d'une variable doit devenir une habitude si l'on veut disposer d'un code source compréhensible, lisible et réutilisable dans de bonnes conditions. En tout état de cause, ces commentaires sont impératifs si le code doit faire partie d'un librairie réutilisée par d'autres programmeurs. Nous recommandons notamment d'écrire ces commentaires avant d'écrire le code lui même, ce qui a en plus l'avantage de forcer à la réflexion pour écrire un programme mieux structuré.

- Les commentaires de ligne décrivent généralement une opération spécifique à l'intérieur d'une fonction ou d'un programme pour donner un peu plus de détails sur son exécution. Voici un exemple en PHP:

```
function writeText($text) {
 makeHumanReadable($text); // formats the string
 echo $text; // Then we can output
}
```

Les commentaires de lignes devraient être utilisés dès qu'un opération délicate est effectuée qui demande une explication. Ils devraient également apparaître dans les pages procédurales qui utilisent d'autre librairies pour faciliter la compréhension de leur fonctionnement. Cependant, si ils sont trop nombreux, ils peuvent avoir l'effet contraire. Ceci peut aussi indiquer que le code n'est pas bien construit et pourrait être simplifié (regrouper les tâches récurrentes dans des fonctions, etc.).

8.5.4 Organisation générale des projets de développement

De façon générale, les développeurs devront essayer de structurer les différents projets de telle sorte que leur organisation soit claire, compréhensible et maintenable. Cet objectif et assez difficile à réaliser car il demande de réfléchir au développement d'une application sous différents point de vue. Nous pouvons cependant proposer les pistes de réflexion suivantes:

- Nous avons déjà signalé plus haut l'importance des commentaires dans le code (voir chapitre 8.5.3 page 90) et le choix des identifiants (voir chapitre 8.5.2 page 89). Ces deux points important doivent être discutés entre les developpeurs afin de choisir le standard à utiliser. Mais la discussion doit également inclure d'autre problématique et devrait déboucher sur un véritable standard pour l'écriture de code (coding standard) à utiliser à TECFA (indentation du code, structure des noms de variables et de fonction, format de message de log, construction générale des objets...). De nombreux standards existent[1], pour un langage particulier, une entreprise particulière, et peuvent servir de base pour la mise en place d'un standard qui doit être respecté à TECFA.
- Les developpeurs devrait s'astreindre à regrouper toute les fonctionnalités récurrentes de leur application dans des librairies. Ces librairies doivent ensuite être rendu disponible pour les autres développeurs au travers du dépôt CVS. Le code contenu dans ces librairies doit être utilisé depuis son emplacement original et non copié de façon physique à l'intérieur de la nouvelle application. De cette façon, s'il y a un changement à apporter pour améliorer le fonctionnement d'un méthode, d'un objet... la modification ne devra se faire qu'à un seul endroit pour que toute les applications puissent en profiter de façon simultanée.
- Enfin, le code est difficile à comprendre dès que le niveau d'indentation devient trop élevé et/ou que sa longueur devient trop importante. Un trop grand niveau d'imbrication de boucles, de sélection et/ou un code trop long sont souvent le signe que des amélioration peuvent être apportées à la structure même du code. Quand ces situations surviennent, les developpeurs devraient alors réfléchir à la simplification de leur code en écrivant des fonctions supplémentaires pour qui prendront en charge les opérations récurrentes ou en regroupant un certains nombre de fonctionnalités au sein d'objets qui seront réutilisés par la suite.

Bien d'autre pistes de réflexion sont à explorer comme l'organisation des applications en couches indépendantes successives ou l'utilisation de modèles de design pour séparer la logique des applications et leur données de leur affichage - comme par exemple le modèle MVT (Model View Controler).

8.6 Documentation

La documentation est un outil absolument indispensable qui doit retenir une attention particulière. En effet, sans documentation, la programmation devient extrêmement compliquée voire impossible. Toute la documentation disponible pour le dispositif à mettre en place devrait être déposée dans un endroit central et stable dans le temps afin

1.Voir par exemple "*Kristiansen, F. (2001). PHP coding standard. http://utvikler.start.no/code/php_coding_standard.html"* ou le standard PEAR ("*Site officiel de PEAR (PHP Extension and Application Repository). http://pear.php.net/"*)

que tous les développeurs puissent la consulter et l'utiliser.

Toute la documentation doit être maintenue à jour. Le répertoire central pour la documentation devrait contenir plusieurs catégories de documents:

- Les manuels des différents logiciels et technologies utilisés pour le dispositif. Bien que ces manuels soit généralement accessibles depuis leur site officiel, il est important d'en conserver un copie locale qui soit cohérente avec la version du logiciel ou de la technologie utilisé dans le dispositif.
- Une explication des standards et des règles de développement choisies. Pour que les consignes décidées en groupe soit respectées, chaque développeur doit pouvoir s'y référer losqu'il en ressent le besoin.
- La documentation du code développé à TECFA. Celle-ci sera directement généré à partir du code source et de ses commentaire. Il est donc vital, comme nous l'avons déjà souligné (voir chapitre 8.5.3 page 90), que le code créé soit commenté dans les règles de l'art. Différents outils existent pour générer de la documentation selon le langage utilisé: javadoc pour java, cdoc pour C/C++, phpdoc ou phpdocumentor pour PHP. Ces outils permettent, à partir de blocs de commentaire formaté selon un standard, de générer la documentation sous différents formats (HTML, XML, pdf...). L'utilisation de mot-clefs spéciaux dans les commentaires permet également de faire des liens vers de documents extérieurs ou vers d'autre partie de la documentation pour cette même application.

  Malheureusement, le format de documentation et les mots clefs à utiliser dépendent encore de l'outil utilisé pour la générer (bien que la tendance soit à l'uniformisation). Le format utilisé pour les commentaire à l'intérieur du code devra donc être choisi en fonction de l'outil qui sera retenu pour la générer par la suite.

9 Développement d'un prototype

Les propositions que nous venons de faire pour la mise en place d'un nouveau dispositif à TECFA sont quelques peut abstraites et s'intéressent surtout à une couche de bas niveau. Nous avons essayé de les mettre en pratique en réalisant des essais de développement que nous allons présenter ici. Tous ces essais sont dans un état de premier prototype et ne sont absolument pas utilisables dans leur état actuel. Pour certains d'entre eux, il ne s'agit même pas de prototypes fonctionnel mais de propositions de structure pour leur développement ultérieur.

9.1 Base utilisateur LDAP et le schéma TECFA

Nous avons mis en place un serveur LDAP pour stocker les informations utilisateurs. Pour cela, nous avons installé openLdap sur un des serveurs centraux de TECFA en utilisant des schémas déjà existants pour les attributs les plus courants (nom, prénom, téléphone, mail...). Les données déjà existantes sur les différents utilisateurs ont été retirées d'un serveur LDAP existant à TECFA utilisé pour l'accès aux newgroups.

Nous nous sommes ensuite attachés à la confection d'un schéma spécifique à TECFA et à ses différents membres pour définir les attributs (page travaux, groupes de travail, promotion, responsable de cours) et les types d'objets (membre de TECFA, étudiant, collaborateur...) dont nous avions besoin.

L'écriture d'un schéma n'est pas une chose facile et demande le respect de quelques règles. Tout d'abord, d'après la spécification LDAP version 3, chaque attribut et chaque type d'objet déclaré dans un schéma LDAP doit avoir un identifiant unique: le OID (Object Identifier). Les OID sont représentés par des nombre formant une hiérarchie. Chaque organisation ou chaque structure susceptible de créer des schémas LDAP doit demander qu'on lui assigne un OID de base à partir duquel elle pourra créer sa propre hiérarchie[1].

Nous avons donc contacté l'autorité responsable de l'affectation des OIDs, l'*Internet Assigned Number Authority* (*IANA* http://www.iana.org). Nous avons obtenu un numéro de bases qui désigne l'unité TECFA[2] à partir duquel nous avons été libre de créer notre propre hiérarchie. La figure 18 explicite la hiérarchie que nous avons mis en place pour construire notre schéma. Nous proposons de l'utiliser comme base pour le développement des futurs schémas.

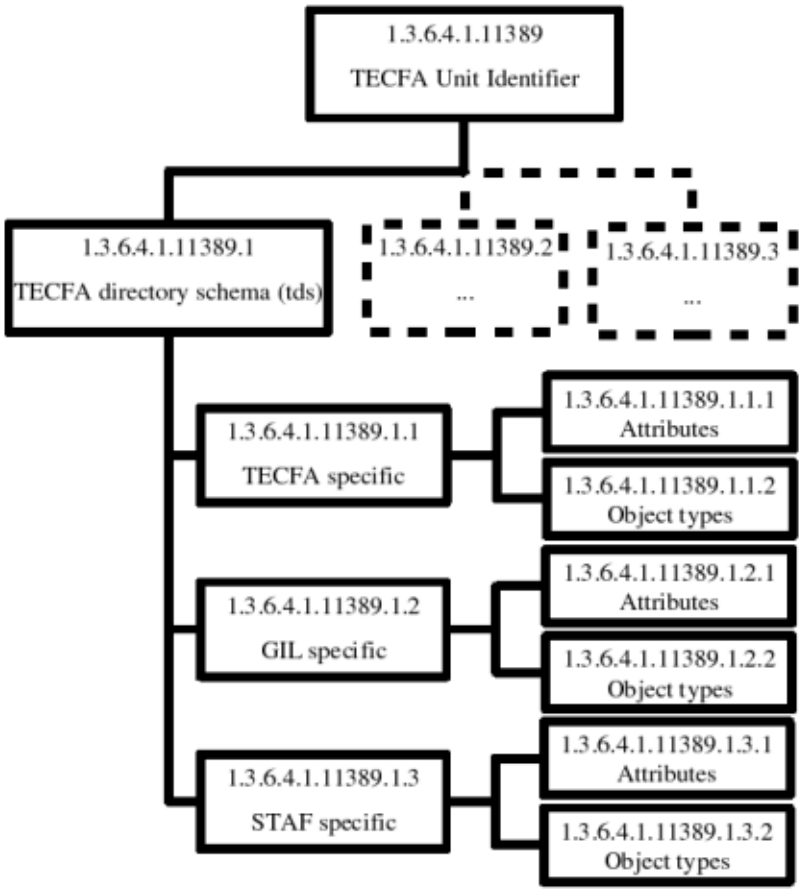
Pour l'instant, la liste des OIDs utilisés a été déclarée directement dans le fichier de schéma et mise à jour au fur et à mesure de son développement. Cette approche peut rester valable tant qu'il n'y a qu'un seul schéma et qu'il contient un nombre d'objets et d'attributs restreints. Cependant, si d'autre schémas doivent être développés, il est très important de garder une trace de tous les OIDs utilisé afin de ne pas dupliquer leur

1.Les OID ne sont pas seulement dédiés aux schémas LDAP. Ils sont également utilisés pour des protocoles réseaux (comme SNMP). Leur utilisation étant planétaire, il est extrêmement important de les respecter et de ne pas utiliser de faux numéros qui pourraient rentrer en conflit avec des numéros existants.

2.Le numéro de base que nous avons obtenu pour TECFA est le 1.3.6.1.4.1.11389

utilisation. Un simple fichier texte peut faire l'affaire mais on peut également réfléchir à mettre en place d'autre solutions (comme une base de données permettant la consultation et la déclaration des nouveau OIDs utilisé avec l'objet qu'ils représentent). Nous n'avons pas exploré la possibilité d'utiliser des outils capables de construire ce genre de listes directement à partir des schémas LDAP et ne savons donc pas s'il en existe.

Fig 18.Proposition de hiérarchie des OIDs à TECFA.



Un fois que nous avons établie les grandes lignes de notre hiérarchie de OID, nous sommes passé à la déclaration du schéma lui même. Nous avons la encore suivit des consignes données pour le choix des identifiants pour les attributs et les types d'objets. Afin d'éviter la confusion avec tout types d'objets ou attributs existants, il est conseillé que chaque identifiant commence par un préfixe. Ce préfixe doit permettre, dans la mesure du possible, d'identifier l'organisation qui a créé le type d'objet ou l'attribut.

Pour aller plus loin et éviter toute confusion, nous avons choisi d'utiliser un préfixe général pour tous les objets LDAP de TECFA et de le compléter pour savoir si l'objet en question est spécifique à STAF, GIL ou TECFA[1]. Nous avons donc à l'arrivée quatre préfixes pour construire nos identifiants:

- *tds*, abréviation de *TECFA Directory Schema* est notre préfixe général
- *tdsTecfa* pour les attributs et les types d'objets spécifiques à TECFA

1.GIL (Geneva Interaction Lab) est un laboratoire de recherche hébergé à TECFA.

- *tdsGil* pour les attributs et les types d'objets spécifiques au GIL
- *tsdStaf* pour les attributs et les types d'objets spécifiques à STAF

Donnons un exemple qui nous montre à quel point ces préfixes peuvent être utiles: nous pouvons maintenant sans ambiguïté définir deux attributs *HomePage*:

- *tdsStafHomePage* définie la page personnelle d'un étudiant
- *tdsTecfaHomePage* définie la page personnelle d'un membre de TECFA

Ainsi, un ancien étudiant devenu membre de TECFA peut conserver son ancienne page personnelle (a diffuser auprès des étudiants des années précédentes et suivantes) tout en ayant une autre page personnelle officielle pour ses nouvelles fonctions, le tout sans ambiguïté de sens et sans risque de confondre les deux.

Nous n'allons pas passer ici en revue les différents types d'objets et d'attribut que nous avons créé. Nous avons inclus dans l'annexe A le schéma que nous avons développé.

9.2 Modification du portail DaCode pour fonctionnement avec LDAP

Nous avons choisi d'installer un portail d'information open source et de le modifier pour l'adapter à notre besoin d'authentification avec LDAP. L'objectif de ce développement était double:

- Nous voulions vérifier la faisabilité d'une modification d'un portail open source et la difficulté effective de l'adapter.
- Nous voulions tester la possibilité d'utiliser CVS pour stocker notre propre version tout en étant en mesure d'intégrer les modification ultérieures apporté par l'équipe de développement originelle.

Notre choix s'est porté sur le produit DaCode (http://www.dacode.com. Ce choix et particulièrement discutable par rapport à certaines propositions que nous avons faites. Notamment, DaCode ne propose pas une véritable API permettant de l'étendre en intégrant des modules[1]. Cependant, nous justifions ce choix pour nos essais en apportant ces quelques précisons.

- La justification principale est que DaCode est le seul produit que nous ayons trouvé qui intégrait l'idée d'authentification par LDAP. D'autre produit y songent. Cependant, DaCode utilisait la meilleure approche à notre point de vue. De plus, l'intégration de LDAP était déjà relativement avancée, ce qui nous permettait de parvenir rapidement à notre objectif
- L'équipe de DaCode stocke son code source sur un dépôt CVS librement accessible en lecture par les visiteurs anonymes. Nous avons ainsi pu travailler directement avec ce serveur avant de mettre en place notre propre dépôt CVS.

1.Cet API existe en fait mais n'est pas encore stabilisée et surtout n'est absolument pas documentée.

- La structure même du programme DaCode est relativement bien faite, assez compréhensible et bien documentée. Les différents développeurs suivent bien les règles édictées pour son développement, notamment en ce qui concerne la production et le formatage des commentaires à l'intérieur du code pour les fonctions/méthodes, les variables et les objets.

Notre expérience a été relativement concluante. Tout d'abord, nous avons réussi à modifier comme nous le souhaitions le mécanisme de création des utilisateurs. Dans la version originale de DaCode, les utilisateurs sont créés, selon la configuration, par un administrateur ou directement par un visiteur en fournissant un login et une adresse email. Le mot de passe généré par le système est ensuite envoyé à l'utilisateur à l'adresse fournie. L'utilisateur doit ensuite confirmer son inscription sous 48 heures pour que son compte soit validé.

Nous avons modifié ce processus afin que les comptes créés dans DaCode soit générés à partir des informations contenues dans notre base LDAP d'utilisateurs. Dans le cas ou l'utilisateur n'existe pas encore dans LDAP, nous avons laissé l'option de créer un compte depuis le portail. Le nouvel utilisateur est alors ajouté à notre base LDAP avec des attributs minimaux. Le résultat n'est pas encore satisfaisant. Notamment, le mot de passe est toujours généré par le portail alors qu'il serait préférable qu'il puisse être librement choisi. D'autre part, si le mot de passe existe déjà dans la base LDAP il devrait être réutilisé ce qui n'est pas encore le cas.

La deuxième source de satisfaction est que nous avons effectivement réussi pendant tout le processus de modification à intégrer les changements effectués par ailleurs par l'équipe de DaCode. En une seule occasion, la mise à jour de notre dépôt CVS a occasionné un conflit entre deux versions de fichier. Cependant, ce problème a pu être rapidement résolu en éditant les fichiers à la main après leur mise à jour.

Les modification que nous avons effectués concernaient surtout l'intégration de LDAP au portail. Durant la développement, nous avons cependant corrigé des erreurs que nous avons trouvées dans le programme lui même. Ces corrections était importantes pour la version originale de DaCode et améliorait son fonctionnement. L'emploi de CVS nous a permis de retourner à l'équipe de DaCode de petit fichiers différentiels (des *patchs*) à intégrer à leur code source pour corriger ces erreurs. Certains de ces patchs ont été accepté et inclus dans la version originale.

On peut donc légitimement penser que, si l'intégration de LDAP que nous avons commencé était finalisée correctement, elle pourrait également faire l'objet d'un patch et être intégré à la version standard. Ceci aurait l'avantage de rendre cette amélioration disponible pour d'autre utilisateurs et de leur permettre également de l'améliorer. Nous pourrions profiter en retour de ces nouvelles améliorations.

Le choix de DaCode comme portail, comme nous l'avons indiqué plus haut, reste fortement discutable. Cependant, le travail que nous avons mené sur ce portail est en théorie transposable à tout autre portail open source et on peut légitimement en attendre les même résultats.

9.3 L'API de base pour la gestion utilisateur et l'accès à LDAP: TECFAPI

Pour cet essai, nous nous sommes retrouvés face à un écueil. Notre première idée était de développer un couche d'authentification de base, puis de la tester en adaptant une

activité existante du campus pour l'utiliser. Cette approche était une erreur et nous nous sommes retrouvé dans l'impossibilité de présenter quelque chose de fonctionnel à intégrer à ce document. Le premier problème est qu'il est très difficile, départ le design du campus virtuel, d'en retirer une activité dans son entier. Le deuxième problème et que ces activités utilisent généralement l'authentification du campus ce qui nous contraint à implémenter une méthode similaire pour notre API. Nous nous retrouvons donc dans la situation inverse de celle que nous cherchions a créer: adapter l'API à l'activité au lieu de faire le contraire.

Nous pouvons cependant tirer quelques conclusions de l'impasse dans laquelle nous nous sommes trouvé. Vu l'importance centrale de cette couche d'authentification et les implications de son utilisation par les activités pédagogiques qui l'utiliseront, il est absolument nécessaire qu'elle soit développée en priorité, avant que tout autre programme ne se base sur ses fonctionnalités. D'autre part, une fois cette couche mise en place, il semble fortement improbable de pouvoir récupérer facilement des activités qui ont déjà été développées par le passé sans en ré-écrire un grande partie, voire de recommencer leur développement depuis le point zéro

Nous avons inclus en annexe B la classe Ldap programmée en PHP (largement inspiré de celle trouvé dans le portail DaCode) pour gérer la communication avec le serveur LDAP. Cette classe est relativement aboutie est peut probablement être utilisée. L'annexe C présente une première structure pour une classe User qui contiendrait les différentes fonctionnalités d'authentification ainsi que les données caractérisant un utilisateur.

9.4 Génération de la documentation

Nous avons voulu essayer un outil de génération automatique de documentation pour le code PHP. Nous nous sommes tournés vers PHPdocumentor (http://phpdocu.sf.net) qui paraissait être un projet relativement abouti, fonctionnel et surtout bien suivi au niveau de son développement avec un équipe de production solide. De plus, les commentaires de code présents dans le portail DaCode étaient compatibles avec cet outils. Nous avons également respecté ce format de commentaires pour nos propres productions.

Nous avons été très agréablement surpris par le résultat obtenu. Nous avons pu, sans aucune difficulté, générer la documentation complète de notre version modifiée de DaCode ainsi que celle de notre API de base (beaucoup moins volumineuse...).

PHPdocumentor permet de générer de la documentation sous format XML, HTML et PDF en utilisant des feuilles de style différentes selon le résultat recherché. Il est également possible de produite ces propres feuilles de style. La génération de la documentation peut se faire à partir d'un formulaire en ligne dans un navigateur Web ou en ligne de commande (sous réserves que PHP soit disponible en ligne de commande). Cette dernière option est particulièrement intéressante car elle permet de fournir un fichier de commande pour générer la documentation complète d'une application. Ainsi, dès qu'un changement est effectué, il suffit d'exécuter ce fichier pour que la documentation soit mise à jour sans avoir à indiquer de nouveau quels fichiers et répertoires prendre en compte.

10 Conclusion

L'éducation est un art difficile. Comme nous avons pu nous en rendre compte à travers les différentes philosophies d'enseignements et les différents modèles qui en ont été tiré, les processus de transmission, d'acquisition et de construction du savoir sont nombreux et complexes. Pour autant, il n'existe pas de "recette générique miracle". Pour chaque situation d'apprentissage, le type d'enseignement à mettre en place sera différent et dépendra de beaucoup de facteurs (sujet de l'enseignement, but recherché par l'enseignant, situation en présence ou à distance, contraintes de temps...).

L'étude des différents dispositifs existants à travers le cadre de référence que nous nous sommes fixé nous a permis d'entrevoir les différents outils qui existaient pour mettre en oeuvre des dispositifs pédagogiques. Pour autant, aucun de ces produit ne permet de supporter toute la variété de situations et de modèles d'enseignement qui existent.

Au sein de TECFA pourtant, un environnement commun de développement des applications pédagogiques doit être ouvert au maximum afin de pouvoir supporter tout types d'interactions. Ce but n'étant pas atteignable en utilisant un seul outils générique, nous avons cherché à savoir quel était le "plus petit dénominateur commun" des différents développements d'applications pédagogiques menés par cette équipe dans le cadre de l'enseignement et de la recherche. Nous en sommes arrivé à la conclusion que, si il est effectivement envisageable d'utiliser un environnement commun, celui ci doit proposer des outils génériques minimaux de bas niveau. De cette manière, les développements peuvent s'effectuer sur une base commune stable et suffisamment flexible pour convenir à tous les développeurs.

Nous avons ensuite passé en revue (sans pour cela être exhaustifs) les différentes solutions techniques qui s'offraient à nous pour poser les bases d'un tel dispositif. A partir de cette étude, nous avons été en mesure de proposer une première architecture technique et des directives de développement pour créer, maintenir et faire évoluer ce dispositif dans le temps. Cette proposition à été élaborée en essayant de répondre aux quelques questions de base que nous étions posées a savoir:

- Quelles doivent être les composantes du dispositif pour qu'elles soient à la fois suffisantes et non contraignantes ?

  Nous avons proposé comme première approche la mise en place d'une base LDAP d'utilisateurs et d'outils pour l'authentification et la gestion. D'autre fonctionnalités pourront être proposées au fur et à mesure des développements mais leur intégration doit résulter d'une négociation entre les différents développeurs et d'un consensus pour leur utilisation.

- Quels outils minimaux de bas niveau doivent être disponibles pour les développeurs ?

  Pour répondre aux besoins de maintenance et pour faciliter le travail en groupe sur une même application, nous avons proposé, outre les élément ci-dessus, la mise en place d'un dépôt central de code source géré par le programme CVS. Egalement, une attention particulière devra être porté à la documentation, notamment celle du code créé localement de façon automatique à partir des commentaires dans les programmes.

- Comment faut-il concevoir l'architecture technique du dispositif pour qu'elle soit adaptable, maintenable avec un minimum de ressources humaines et évolutive ?

  Nous avons proposé une architecture de bas niveau (apache, LDAP, PHP, MySql...) qui permettent de s'appuyer sur un petit ensemble d'outils robustes et stables. L'utilisation de ces outils de bas niveau permet un grande flexibilité pour qu'elle convienne à tous les programmeurs.

- Comment faciliter l'utilisation du dispositif et quels processus mettre en place pour qu'il soit utilisé et maintenu de façon durable ?

  Nous avons proposer quelques directives de développement qui devront par la suite être discutées et améliorées. Ces directives concernent notamment l'hygiène de programmation et l'utilisation du dépôts CVS. Le respect de certaine règles commune permet de rendre les programmes plus lisibles, plus compréhensibles. Notamment, la production systématique de commentaires formatés à l'intérieur du code permet par la suite de générer automatiquement la documentation de l'application programmée. Cette documentation peut ensuite être consulté par les développeurs, ce qui leur fait gagner un temps précieux pour la modification ou la réutilisation de ressources existantes.

Enfin, nous avons présenté quelques essais de développement pour mettre en pratique quelques unes ses propositions que nous avons faites.

Le travail que nous avons mené ici n'a pas abouti en une véritable proposition concrète. Nous avons plutôt mené une exploration des différentes solutions envisageables et proposé des pistes de réflexion pour la véritable mise en place de ce dispositif. Il reste donc beaucoup de points en suspens qui devront faire l'objet d'une véritable discussion entre les différents acteurs amenés à utiliser cette plateforme.

En effet, pour que ce dispositif voit le jour de façon concrète, qu'il réponde aux attentes des différents utilisateurs, qu'il soit effectivement utilisé et maintenu en état de marche, un véritable consensus doit émerger de la négociation entre les différents acteurs qui seront amenés à l'utiliser. Un compromis satisfaisant doit être accepté par tous non seulement pour l'architecture exacte à mettre en place mais également pour les règles à suivre et à respecter pour sa bonne utilisation et son évolution. En d'autres termes, ce dispositif restera à notre avis un voeu pieu s'il n'emporte pas l'adhésion d'une communauté suffisamment large et active pour son utilisation et son développement.

Références

Les ouvrages publiés de façon classique se trouvent dans la section “Publications papiers”. Ils sont éventuellement suivis d’un URL permettant de se procurer une copie électronique du document ou de sa table des matières. Les références situées dans la section “Documents en ligne seulement/ Sites de référence” sont disponibles uniquement à l’URL indiqué (ou n’ont pas été trouvés en référence papier). Tous les URL publiés ici étaient valides lors de l’impression de ce document. L’auteur n’est pas responsable de leur éventuelle expiration.

Publications papier

Bargh, J.A., Schul, Y. (1980). *On the cognitive benefits of teaching*. Journal of Educational Psychology, 72, 593-604.

Bergsten, H. (2001). *Java Server Pages*. O’Reilly: Sebastopol, CA

Brandau, D.T., Chi, X. (1996). *The development of a computer-mediated academic communication system*. In P. Carlson & F. Makedon (Eds), Educational telecommunications ‘96 (pp. 46-50). Boston, MA: AACE.

Bruner, J. (1960). *The process of Education*. Cambridge, MA: Harvard University Press.

Bruner, J. (1966). *Towards a Theory of Instruction*. Cambridge, MA: Harvard University Press.

Chi, M.T.H., Bassok, M., Lewis, M.W., Teinman, P., Glaser, R. (1989). *Self-explanations: how students study and use examples in learning to solve problems*. Cognitive Science, 13, 145-182.

Crowder, N.A. (1959). *Automatic tutoring by means of intrinsic programming*, in E. Galanter (ed.), Atomatic Teaching: the State of the Art, New York: Wiley.

Dillenbourg, P., Baker, M., Blaye, A., O’Malley, C. (1996). *The evolution of research on collaborative learning*. In E. Spada & P. Reiman (Eds), Learning in Humans and Machine: Towards an interdisciplinary learning science. (pp. 189-211). Oxford: Elsevier.

Johner, H., Brown, L., Hinner, F.S., Reis, W., Westman, J. (1998). *Understanding LDAP*. Raleigh, N.C.: IBM Publications. http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg244986.pdf

Joyce, B., Weil, M. & Calhoun, E. (2000). *Models of teaching*, 6th edition. Boston: Allyn & Bacon.

McLean, N. (2001), *Interoperability convergence of online learning and information environments*. The New Review or Information Networking, 7. Cambridge: Taylor Graham Publishing. http://www.colis.mq.edu.au/news_archives/convergence.pdf

O’Shea, T. & Self, J. (1983). *Learning and Teaching with Computers - Artificial Intelligence in Education*. Brighton: Harvester Press.

Paquette, G. (2000) *Construction de portails de télé-apprentissage - Explor@: Une diversité de modèles pédagogiques*. Science et Techniques Educatives, 7 (1), 207-226. http://www.licef.teluq.uquebec.ca/gp/doc/publi/campus/steexplora.doc

Perelman, L.J. (1992). *School’s out: Hyperlearning, the new technology, and the end off education*. New York: William Morrow.

Piaget, J. (1964). *Six Etudes de Psychologie*. Genève: Editions Gonthier

Reeves, T.C. & Reeves, P.M. (1997). *Effective Dimensions of Interactive Learning on the World Wide Web*. In B.H. Khan (Ed.), Web-Based Instruction (pp. 59-66). Englewood Cliffs, NJ: Educational Technology Publications.

Reuchlin, M. (1990). *Psychologie* - 8eme édition mise à jour. Paris: Presses Universitaires de France.

Skinner, B.F. (1938). *The Behaviour of Organisms: an Experimental Analysis*. New York: Appleton-Century-Crofts.

Skinner, B.F. (1968). *The Technology of Teaching*, New York: Appleton-Century-Crofts.

Vygotsky, L.S. (1962). *Thought and Language*. Cambridge, Massachusetts: MIT Press

Vygotsky, L. S. (1978). *Mind in Society: The Development of Higher Psychological Processes*. Harvard University Press.

Webb, N.M. (1991). *Task related verbal interaction and mathematics learning in small groups*. Journal for Research in Mathematics Education, 22 (5), 366-389.

Documents en ligne seulement / Sites de référence

ADL (Advance Distributed Learning) (2002). *SCORM Overview*. http://www.adlnet.org/index.cfm?fuseaction=scormabt&cfid=303063&cftoken=81276702

Class, B. (2001). *De l'éducation présentielle à l'éducation distancielle: quelques concepts et études de cas*. Mémo, Avril 2001. TECFA, Faculté de Psychologie et des Sciences de l'Education, Université de Genève. http://tecfa.unige.ch/~class/avril_01/doc_travail.pdf

Class, B. & Schneider D.K. (2001). *Outils pédagogiques pour l'élaboration d'un cours*. Transparents de cours. TECFA, Faculté de Psychologie et des Sciences de l'Education, Université de Genève. http://tecfa.unige.ch/guides/tie/html/pedago/pedago.html

Cederqvist, P. et al.*Version Management with CVS for CVS 1.11.2*. Signum Support AB. http://www.cvshome.org/docs/manual/

Fox, G.C. (2000). *Portals and Frameworks for Web Based Education and Computational Science*. School for Computational Science and Information Technology - Florida State University. http://www.new-npac.org/users/fox/documents/pajavaapril00/

Hoover, W.A. (1996). *The Practice Implications of Constructivism*. SEDLetter, IX (3). http://www.sedl.org/pubs/sedletter/v09n03/practice.html

IBM Corporation (2002). *Introduction to the Darwin Information Typing Architecture (DITA)*. http://www-106.ibm.com/developerworks/library/x-dita1

OVAREP (2000). *Etude Comparative Technique et Pédagogique des Plates-Formes pour la Formation Ouverte et à Distance*, mise à jour novembre 2000. http://www.algora.org/kiosque/publicat/doc/pdf/pf2000.pdf

Kristiansen, F. (2001). *PHP coding standard*. http://utvikler.start.no/code/php_coding_standard.html

Programmers Resource (2000). *What are Active Server Pages (ASP)?*. http://www.programmersresource.com/articles/whatisasp.asp

Kearsley, G. *The Theory into Practice Database*. http://tip.psychology.org/

Site officiel CVS http://cvshome.org/

Site officiel DocBook http://docbook.org

Site officiel Java http://java.sun.com/

Site officiel de PEAR (PHP Extension and Application Repository). http://pear.php.net/

Site officiel php http://php.net

Site officiel de phpdocumentor http://phpdocu.sf.net/

SURFnet. *Introducing a Directory Service*. http://www.surfnet.nl/innovatie/afgesloten/x500/introducing/

Liste des figures

ANNEXE ALe schéma LDAP pour TECFA

```
# $Header: /export/data2/tecfacvs/tecfapi/ldap-schema/tecfa.schema,v 1.5
2002/09/11 23:54:12 clavel Exp $

#######################################################
# This is the TECFA OpenLDAP Schema description      #
# http://tecfa.unige.ch                              #
#                                                    #
# This is a draft version under test                 #
#                                                    #
# Please do not use unless you experiment with TECFA #
# This schema is subject to change anytime without   #
# any prior notification                             #
#                                                    #
# The final schema will be made public.              #
#                                                    #
# This openLDAP schema depends upon                  #
# - core.schema                                      #
# - cosine.schema                                    #
# - inetorgperson.schema                             #
#######################################################



# Author: Olivier Clavel
# Olivier.Clavel@laposte.net
#
# ToDo:
# - Make a real page somewhere to manage our OID hierarchy
#   IMPORTANT: until this is done, please continue
#   documenting branches in this document


# OID Hierarchy for TECFA
#
# ************************************************************
# 1.3.6.1.4.1.11389 => TECFA Unit identifier (tecfa, gil, staf...)
# ************************************************************
#
#########################################################################
# 1.3.6.1.4.1.11389.1 => Tecfa Directory Schema (starts with "tds")
#########################################################################
#
#-----------------------------------------------------------------
#-----------------------------------------------------------------
# 1.3.6.1.4.1.11389.1.1 => TECFA specific (starts with "tdsTecfa")
#-----------------------------------------------------------------
#-----------------------------------------------------------------
#
# 1.3.6.1.4.1.11389.1.1.1 => TECFA specific attributes
#-----------------------------------------------------
#

# The HomePage discribing the person.
```

```
attributeType ( 1.3.6.1.4.1.11389.1.1.1.1 NAME 'tdsTecfaHomePage'
                            DESC 'The HomePage of the person'
                            SUP labeledURI)

# A string representing the category of a tecfa member: teacher, assistant,
etc...
attributeType ( 1.3.6.1.4.1.11389.1.1.1.2 NAME 'tdsTecfaMemberCategory'
                            DESC 'Category of a tecfaMember'
                            EQUALITY caseIgnoreMatch
                            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)


#
# 1.3.6.1.4.1.11389.1.1.2 => TECFA specific objects types
#---------------------------------------------------------
#

# A person that is or was working at TECFA, or is in relation
# with TECFA a way or another and logs on our machines.
objectClass ( 1.3.6.1.4.1.11389.1.1.2.1 NAME 'tdsTecfaPerson'
                            DESC 'A person related to TECFA who logs on
machines'
                            SUP inetOrgPerson
                            STRUCTURAL
                            MUST ( uid $ userPassword $ mail)
                            MAY ( tdsTecfaHomePage $ personalTitle ))

# A person which is or has been a member of TECFA
objectClass ( 1.3.6.1.4.1.11389.1.1.2.2 NAME 'tdsTecfaMember'
                            DESC 'A member of TECFA'
                            SUP tdsTecfaPerson
                            STRUCTURAL
                            MUST ( tdsTecfaHomePage $ manager $
tdsTecfaMemberCategory $
                            telephoneNumber)
                            )

#
#-------------------------------------------------------------
#-------------------------------------------------------------
# 1.3.6.1.4.1.11389.1.2 => GIL specific (starts with "tdsGil")
#-------------------------------------------------------------
#-------------------------------------------------------------
#
# 1.3.6.1.4.1.11389.1.2.1 => GIL specific attibutes
#-------------------------------------------------
#
# None at the moment
#
# 1.3.6.1.4.1.11389.1.2.2 => GIL specific objects types
#-------------------------------------------------------
#

# A member of GIL
objectClass ( 1.3.6.1.4.1.11389.1.2.2.1 NAME 'tdsGilMember'
                            DESC 'A member of GIL'
                            SUP tdsTecfaMember)
#
```

```
#-------------------------------------------------------------
#-------------------------------------------------------------
# 1.3.6.1.4.1.11389.1.3 => STAF specific (starts with "tdsStaf")
#-------------------------------------------------------------
#-------------------------------------------------------------
#
# 1.3.6.1.4.1.11389.1.3.1 => STAF specific attributes
#-----------------------------------------------------

# A single letter, initial of the chosen name for the promotion.
# Length size 2 in case we have to change the way we name those promotions.
attributeType ( 1.3.6.1.4.1.11389.1.3.1.1 NAME 'tdsStafPromotion'
                            DESC 'STAF Student promotion, single letter'
                            EQUALITY caseIgnoreMatch
                            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{2} )

# Specifies if the student is active or not
attributeType ( 1.3.6.1.4.1.11389.1.3.1.2 NAME 'tdsStafIsActive'
                            DESC 'Is the student active - True/False'
                            EQUALITY booleanMatch
                            SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )

#######
##
## Deprecated (before it was even born :) Use tdsTecfaHomePage instead
##
#######
## attributeType ( 1.3.6.1.4.1.11389.1.3.1.3 NAME 'tdsStafHomePage'
##             DESC 'Labeled URL of the student HomePage @ TECFA'
##             SUP labeledURI )
#######


# Student's Work Page @ tecfa.
attributeType ( 1.3.6.1.4.1.11389.1.3.1.4 NAME 'tdsStafWorkPage'
                            DESC 'Labeled URL of the student WorkPage @
TECFA'
                            SUP labeledURI )

# Student's Test Page @ tecfa
attributeType ( 1.3.6.1.4.1.11389.1.3.1.5 NAME 'tdsStafTestPage'
                            DESC 'Labeled URL of the student TestPage @
TECFA'
                            SUP labeledURI )

# The student mail assigned by the University. This is for info only.
# We should use the "mail" attribute to send emails
attributeType ( 1.3.6.1.4.1.11389.1.3.1.6 NAME 'tdsStafStudentMail'
                            DESC 'The email address assigned by University.
Info only'
                            SUP mail)

# Indicates if the student completed his diploma
attributeType ( 1.3.6.1.4.1.11389.1.3.1.7 NAME 'tdsStafIsCompleted'
                            DESC 'Did the student complete his diploma -
True/False'
                            EQUALITY booleanMatch
                            SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 )
```

```
#
# 1.3.6.1.4.1.11389.1.3.2 => STAF specific objets types
#-------------------------------------------------------


# A Staf Student
ObjectClass ( 1.3.6.1.4.1.11389.1.3.2.1 NAME 'tdsStafStudent'
                            DESC 'A STAF Student'
                            SUP  tdsTecfaPerson
                            STRUCTURAL
                            MUST ( tdsStafPromotion $ tdsStafIsActive $
                            tdsTecfaHomePage $ tdsStafWorkPage )
                            MAY ( tdsStafTestPage $ tdsStafStudentMail $
                            tdsStafIsCompleted ))

# $Log: tecfa.schema,v $
# Revision 1.5  2002/09/11 23:54:12  clavel
# test for for the cvs log tag
#
```

ANNEXE BTECFAPI: classe LDAP

```
<?php
/**
* $Header: /export/data2/tecfacvs/tecfapi/src/ldap.php,v 1.3 2002/09/11
21:54:50 clavel Exp $
* @package tecfapi
*/
/**
* Abstraction Class to manage ldap connection and communication.
*
* Mostly inspired from {@link http://www.dacode.org/ DaCode} ldap class to
manage ldap users.
* @copyright Eric Kifoil <eric@ipass.net>
* @author Eric Kilfoil <eric@ipass.net>
* @author Pascal Pucci <Pascal@deenoo.com>
* @author Olivier Clavel <Olivier.Clavel@laposte.net>
* @package tecfapi
*
*/


class Ldap {
                /**
                * Ldap hostname
                * @var string The ldap hostname
                */
                var $hostname="localhost";

                /**
                * Ldap base dn
                * @var string The ldap base DN
                */
                var $basedn="o=tecfa.unige.ch";

                /**
                * Ldap bind dn
                * @var string The ldap bind DN
                */
                var $binddn="";

                /**
                * Ldap bind
                * @var string The ldap bind password
                */
                var $bindpw="";

                /**
                * Ldap server connection id
                * @var int The ldap connection id
                */
                var $cid;

                /**
                * Ldap server bind id
```

```
                *
                * Ldap server bind id. 0 (FALSE) if problem, bind id (ressource)
otherwise
                * @var int bind id (evaluates to false if problem)
                */
                var $bid;

                /**
                * Ldap search result
                * @var integer The search result
                */
                var $sr;

                /**
                * Ldap result entry
                * @var integer Result entry
                */
                var $re;

                /**
                * link status.
                *
                * True if connection AND bind where successfull
                * (user loged in with {@link $binddn} and {@link $bindpw})
                * @var boolean link status
                */
                var $status=false;

                /**
                * Ldap error
                *
                * Any error messages to be returned can be put here
                * @var string The error message
                */
                var $error = null;

                /**
                * Determines if we are fetching the first entry of result
                *
                * Determines if we are fetching the first entry of result.
                * 0 if we are fetching the first entry, 1 otherwise
                * @var integer 0 or 1
                */
                var $start = 0;

                /**
                * The attributes to return in the search.
                *
                * Contains the attributes to be returned when
                * completing a search(). All are returned if empty.
                * @var array The attributes to search for or empty for all
                */
                var $attributes = array();


                /**
                * Class constructor.
                *
```

```
                * Performs an annonymous bind on "localhost" if called with no
argument.
                * @param string The hostname of the LDAP server
                * @param string The dn for the bind (kind of username)
                * @param string The passwd used for the bind
                */
                function Ldap($binddn=null, $bindpw=null, $hostname=null) {
                                if (isset($hostname)){
                                                $this->setLdapHost($hostname);
                                }
                                if (isset($binddn)) {
                                                $this->setBinddn($binddn);
                                }
                                if isset($bindpw) {
                                                $this->setBindpw($bindpw);
                                }

                                $this->link($binddn, $bindpw, $hostname)
                }

                /**
                * Sets hostname of LDAP server
                *@param string
                *@access public
                */
                function setLdapHost($hostname)
                {
                                $this->hostname = $hostname;
                }

                /**
                * Returns hostname of LDAP server
                * @return string The ldap hostname
                * @access public
                */
                function getLdapHost() {
                                return($this->hostname);
                }

                /**
                * Sets distinguished name for the bind
                * @param string The bind DN
                */
                function setBinddn($binddn) {
                                $this->binddn=$binddn;
                }

                /**
                * Returns distinguished name for the bind
                * @return string The bind DN
                * @access public
                */
                function getBinddn() {
                                return($this->binddn);
                }

                /**
                * Sets base distinguished name
```

```
                * @param string The base DN
                * @access public
                */
                function setBasedn($basedn) {
                                $this->basedn = $basedn;
                }

                /**
                * Returns base distinguished name
                * @return string The base DN
                * @access public
                */
                function getBasedn() {
                                return($this->basedn);
                }

                /**
                * Sets bind password
                * @param string The bind password
                * @access public
                */
                function setBindpw($bindpw) {
                                $this->bindpw = $bindpw;
                }

                /**
                * Returns bind password
                * @return string The bind password
                * @access public
                */
                function getBindpw() {
                                return($this->bindpw);
                }

                /**
                * Sets current node in ldap tree.
                * @param string The dn of the node (or '..' to move one level
up)
                * @access public
                */
                function cd($node) {
                                if ($node == "..") {
                                                $this->basedn = $this-
>getParentNode();
                                } else {
                                                $this->basedn = $node;
                                }
                                $this->dis;
                }
                /**
                * Returns the name of the parent node in LDAP hierarchy.
                * @param string dn to compute parent.
                * @return string
                * @access public
                */
                function getParentNode($dn = "") {
                                if (empty($dn)) {
                                                $dn = $this->basedn;
```

```
                        }
                        if ($this->basedn == LDAP_BASEDN) {
                                return("");
                        }
                        return(ereg_replace("[^,]*[,]*[ ]*(.*)", "\\1",
$dn));
        }

        /**
        * Adds an attribute to be returned in the search
        *
        * Adds one more attribute to the $attributes array so that it
gets
        * returned in the next search performed
        * @param string The attribute to be returned
        * @access public
        */
        function add_return_attribute($attr) {
                $this->attributes[] = $attr;
        }

        /**
        * Returns the attributes to be returned by search()
        * @access public
        * @return array The attributes
        */
        function get_return_attributes() {
                return $this->attributes;
        }

        /**
        * Empties the $attributes array (all attributes will be
returned in next search)
        * @access public
        */
        function empty_return_attributes() {
                $this->attributes = array();
        }

        /**
        * Connects and bind to the LDAP server.
        *
        * If the connection is already opened, just does the bind.
        * To change user on the same connection, simply use {@link
setBinddn()}
        * and {@link setBindpw()}, then call link() again.
        * @return boolean The link status
        * @see $status
        */
        function link() {
                $e = error_reporting(0);
                if (!$this->cid) {
                        if (!$this-
>cid=ldap_connect($this->hostname)) {
                                $this->error =
"Could not connect to LDAP server on ".$this->hostname;
                                $this-
>status=false;
```

```
error_reporting($e);
                                                        return $this-
>status;
                                        }
                                }
                                if (!$this->bid = ldap_bind($this->cid, $this-
>binddn, $this->bindpw)) {
                                        $this->error = "Could not bind
with binddn " . $this->binddn . "wrong password ???";
                                        $this->status=false;
                                        error_reporting($e);
                                        return($this->status);
                                }
                                $this->status=true;
                                error_reporting($e);
                                return($this->status);
                }

                /**
                * Closes the connexion to the LDAP server
                * @access public
                */
                function unlink() {
                                ldap_close($this->cid);
                }

                /**
                * Performs a search in the directory. This is quite poorly
written...
                * @param string the filter for the search
                * @return integer a search result identifier or 0 on failure.
                * @access public
                * @todo implement a search filter variable and functions in the
class.
                */
                function search($filter) {
                                $e = error_reporting(0);
                                $result = array();
                                if (!$this->link()) {
                                        error_reporting($e);
                                        return(0);
                                }

                                $this->sr = ldap_search($this->cid, $this-
>basedn, $filter, $this->attributes);
                                $ldap->error = ldap_error($this->cid);
                                $this->resetResult();
                                error_reporting($e);
                                return($this->sr);
                }

                /**
                * Lists all objects at the current level of LDAP hierarchy.
                * @param string the filter for the search
                * @param string the base dn to use, if not the current one.
                * @access public
                */
```

```
                function ls($filter = "(objectclass=*)", $basedn = "") {
                                if (empty($basedn)) {
                                                $basedn = $this->basedn;
                                }
                                if (empty($filter)) {
                                                $filter = "(objectclass=*)";
                                }

                                $e = error_reporting(0);
                                $result = array();
                                if (!$this->link()) {
                                                error_reporting($e);
                                                return(0);
                                }

                                $this->sr = ldap_list($this->cid, $basedn,
$filter);
                                $ldap->error = ldap_error($this->cid);
                                $this->resetResult();
                                error_reporting($e);
                                return($this->sr);
                }

                /**
                * Reads an entry in the directory
                * @param string the dn of the object to read
                * @return integer a search result identifier or 0 on error.
                * @access public
                */
                function cat($dn) {
                                $e = error_reporting(0);
                                $result = array();
                                if (!$this->link()) {
                                                error_reporting($e);
                                                return(0);
                                }
                                $filter = "(objectClass=*)";

                                $this->sr = ldap_read($this->cid, $dn, $filter,
$this->attributes);
                                $ldap->error = ldap_error($this->cid);
                                $this->resetResult();
                                error_reporting($e);
                                return($this->sr);
                }

                /**
                * Fetches an entry in the current result set.
                * @return array The attributes of the entry
                * @access public
                */
                function fetch() {
                                $e = error_reporting(0);
                                if ($this->start == 0) {
                                                $this->start = 1;
                                                $this->re =
ldap_first_entry($this->cid, $this->sr);
                                } else {
```

```
                                        $this->re =
ldap_next_entry($this->cid, $this->re);
                        }
                        if ($this->re) {
                                $att = ldap_get_attributes($this-
>cid, $this->re);
                        }
                        $ldap->error = ldap_error($this->cid);
                        error_reporting($e);
                        return($att);
        }

        /**
        * Resets the array of results
        * @access public
        */
        function resetResult() {
                $this->start = 0;
        }

        /**
        * Gets dn of current result entry
        * @return string
        * @access public
        */
        function getdn() {
                $e = error_reporting(0);
                $rv = ldap_get_dn($this->cid, $this->re);
                $ldap->error = ldap_error($this->cid);
                error_reporting($e);
                return($rv);
        }

        /**
        * Counts the number of entries in the result set
        * @return int The number of results
        * @access public
        */
        function count() {
                $e = error_reporting(0);
                $rv = ldap_count_entries($this->cid, $this->sr);
                $ldap->error = ldap_error($this->cid);
                error_reporting($e);
                return($rv);
        }

        /**
        * Adds a new entry in directory of objectClass top
        * @param string Name of the attribute (for the DN)
        * @param string Name of the entry (for the DN)
        * @param string Dn where to add entry
        * @return boolean true on success, false otherwise
        * @access public
        */
        function mknode($attrname, $dirname, $basedn = "") {
                if (empty($basedn)) {
                        $basedn = $this->basedn;
                }
```

```
                        $e = error_reporting(0);
                        $info["objectclass"] = "top";
                        //$info[$attrname] = $dirname;
                        $r = ldap_add($this->cid, "$attrname=$dirname, "
. $basedn, $info);
                        $ldap->error = ldap_error($this->cid);
                        error_reporting($e);
                        return($r ? $r : 0);
                }

                /**
                * Deletes attributes of entry
                * @param mixed Array of attributes to delete or empty string
(all attributes)
                * @param string The entry (default: current entry)
                * @return boolean true on success, false otherwise
                * @access public
                */
                function rm($attrs = "", $dn = "") {
                        if (empty($dn)) {
                                $dn = $this->basedn;
                        }

                        $e = error_reporting(0);
                        $r = ldap_mod_del($this->cid, $dn, $attrs);
                        $ldap->error = ldap_error($this->cid);
                        error_reporting($e);
                        return($r);
                }

                /**
                * Replaces attributes for specified dn. Wrapper for
ldap_mod_replace
                * @param array the arrtibutes to set
                * @param string the dn; current one if empty.
                * @return boolean true on success
                * @access public
                */
                function rename($attrs, $dn = "") {
                        if (empty($dn))
                        $dn = $this->basedn;

                        $e = error_reporting(0);
                        $r = ldap_mod_replace($this->cid, $dn, $attrs);
                        $ldap->error = ldap_error($this->cid);
                        error_reporting($e);
                        return($r);
                }

                /**
                * Deletes an entry from directory
                * @param string DN to delete. NO DEFAULT
                * @return boolean true on success
                * @access public
                */
                function rmnode($deletedn) {
                        $e = error_reporting(0);
                        $r = ldap_delete($this->cid, $deletedn);
```

```
                        $this->error = ldap_error($this->cid);
                        error_reporting($e);
                        return($r ? $r : 0);
                }

                /**
                * Modifies the current entry
                * @param string The attributes to modify
                * @return boolean true on success, false otherwise
                * @access public
                */
                function modify($attrs) {
                        $e = error_reporting(0);
                        $r = ldap_modify($this->cid, $this->basedn,
$attrs);
                        $this->error = ldap_error($this->cid);
                        error_reporting($e);
                        return($r ? $r : 0);
                }
}

?>
```

ANNEXE CTECFAPI: classe User

```
<?php
/**
* $Header: /export/data2/tecfacvs/tecfapi/src/user.php,v 1.3 2002/09/11
21:54:50 clavel Exp $
* @package tecfapi
*/
/**
* Handles users for TECFA campus.
*
* This class should be the first one instanciated in the calling
* script, before anything else is done. It takes care of authentying
* the user if he did not login yet and get attributes from session
* or ldap.
*
*
*
* @author Olivier Clavel <olivier.clavel@laposte.net>
* @package tecfapi
* @copyright Clavel 2002
*/
class User {

               /**
               * Stores error message if any
               * @var string The error message
               */
               var $error;

               /**
               * Instance of ldap class
               *
               * This variable is for internal use.
               * @var object Ldap
               * @access private
               * @see Ldap
               */
               var $ldap;

               /**
               * User login
               *
               * @var string user unique identifier
               */
               var $login;

               /**
               * User personal title
               *
               * Contains the string M., Mlle or Mme.
               *
               * Matches personalTitle ldap attribute
               * @var string User personal title
               * @see $title
```

```
                */
                var $personalTitle;

                /**
                * User title
                *
                * Contains a string such as director, secretary, assistant...
                * Empty if user is not a tecfa member.
                *
                * Matches title ldap attribute
                * @var string User title (for work)
                * @see $personalTitle
                */
                var $title;


                /**
                * User first name
                *
                * Matches givenName ldap attributes
                * @var string User first name
                */
                var $firstName;

                /**
                * User last name
                *
                * Matches sn ldap attribute
                * @var string User last name
                */
                var $lastName;

                /**
                * User common name
                *
                * Contains a concatenation of {@link $fistName} and {@link
$lastName}.
                * It represents an ldap mandatory attributes and is here mainly
                * for update purpose when modifying {@link $fname} or {@link
$lname}.
                * Can be usefull thow to display the user full name.
                * Matches the cn ldap attribute
                *
                * @var string User common name
                */
                var $commonName;

                /**
                * User phone number
                *
                * Matches the telephoneNumber ldap attribute
                * @var string phone number
                * @see $homePhoneNumber
                */
                var $phoneNumber;

                /**
                * User home phone number
```

```
*
* An alternate phone number to reach user
* at home
*
* Matches the homePhone ldap attribute
* @var string Home phone number
* @see $phoneNumber
*/
var $homePhoneNumber;

/**
* User fax number
*
* Matches the facsimileTelephoneNumber ldap attribute
* @var string fax number
*/
var $faxNumber;

/**
* User mobile phone number
*
* Matches the mobile ldap attribute
* @var string mobile phone number
*/
var $mobileNumber;

/**
* User email
*
* Matches the mail ldap attribute
* @var user email
*/
var $email;


/**
* User address
*
* Contains the street and number, and eventually extra info
* (such as residence name....)
*
* Matches the street ldap attribute
* @var string address ([extra info\n] street num)
*/
var $address;

/**
* User postal code
*
* Postal code in international format (eg CH-1205)
*
* Matches the postalCode ldap attribute
* @var string international format postal code
*/
var $postalCode;

/**
* User city
```

```
*
* Matches the l ldap attribute
* @var string city
*/
var $city;

/**
* Room number for tecfa employees
*
* Mainly use for tecfa members to indicate
* office number.
*
* Matches the roomNumber ldap attribute
* @var string Room number
*/
var $roomNumber;

/**
* User's manager
*
* Contains the dn of the user's manager in ldap
*
* Matches the manager ldap attribute
* @var string manager's cn in ldap
* @todo find some more user friendly presentation.
*/
var $manager;

/**
* Tecfa Member category
*
* The category of member: teacher, guest, assistant, student...
*
* Matches the tdsTecfaMemberCategory ldap attribute
* @var string The category
*/
var $memberCategory;

/**
* User home page
*
* Matches the tdsTecfaHomepage ldap attribute
* @var string Home page URL
*/
var $homePage;

/**
* Student work page
*
* Matches the tdsStafWorkPage ldap attribute
* @var string Work page URL
*/
var $workPage;

/**
* Student test page
*
* Matches the tdsStafTestPage ldap attribute
```

```
                * @var string Test page URL
                */
                var $testPage;

                /**
                * Student promotion (name)
                *
                * Matches the tdsStafPromotion ldap attribute
                * @var string Name of promotion
                */
                var $promotion;

                /**
                * Student university email
                *
                * This is the email provided automatically by the
                * the university uppon registration. This is for info purpose
                * only. {@link $email email} should be used for communications.
                *
                * Matches the tdsStafStudentMail ldap attribute
                * @var string Student email
                */
                var $studentMail;

                /**
                * Is the student active ?
                *
                * This variable refers to the activity of the student.
                * See {@link $isCompleted} for diploma completion
                * @var bolean
                */
                var $isActive;

                /**
                * Did the student complete is diploma ?
                *
                * This refers to diploma completion only. For activity
                * see {@link $isActive}.
                */
                var $isCompleted;

                /**
                * Class constructor
                */
                function User() {

                                require("ldap.php");
                                $this->ldap = new Ldap();


                                $this->ldap->setLDAPHost("localhost");
                                $this->ldap-
>setBindDN("uid=root,o=tecfa.unige.ch");
                                $this->ldap->setBindPassword("iapiaget");
                                $this->ldap->connect
                }

                /**
```

```
                * Get user title
                * @return string title
                */
                function get_title() {
                                return $this->title;
                }

                /**
                * Sets user title
                * @param string title
                */
                function set_title($title) {
                                $this->title = $title;
                }


                /**
                * Authentifies a user
                * @param string login The user’s login
                * @param string password The user’s password
                * @return boolean TRUE on success, false otherwise
                */
                function authentify($login, $password) {
                                return;
                }

                /**
                * retreives user info in defined variables
                *
                */
                function retreive_user_info() {

                }
}
        ?>
```