Sciences et Technologies de l´Information et de la Communication pour l´Éducation et la Formation

Volume 12, 2005
Article de recherche

Un entrepôt pour stocker et rechercher des composants logiciels utiles aux EIAH

Issam Rebaï [Crip5, Université Paris 5 et ESIEA Recherche, ESIEA]

Nicolas Maisonneuve [Crip5, Université Paris 5]

Jean-Marc Labat [Lip6, Université Paris 6]

■ RÉSUMÉ : La visibilité des recherches en France dans le domaine des EIAH est restée réduite malgré la qualité de nombre de ces travaux, en particulier à cause de la difficulté à construire un environnement opérationnel suffisamment développé pour être expérimenté en situation réelle. Tester et faire fonctionner un outil obligent généralement les chercheurs à développer des programmes complémentaires souvent disponibles ailleurs mais non référencés. Pour mieux valoriser les travaux de recherche et éviter de re-développer plusieurs fois les mêmes fonctionnalités, une condition est donc que les réalisations logicielles soient décrites et indexées dans un espace commun, l'espace de stockage pouvant lui être distribué. Dans cet article, nous présentons la conception et la réalisation d'un entrepôt destiné à stocker, indexer et rechercher des composants logiciels nécessaires aux EIAH. Nous proposons également un schéma de métadonnées pour décrire ces composants logiciels en nous inspirant des standards existants mais en apportant des compléments car nous pensons que, en particulier les aspects techniques, ne sont pas suffisamment décrits.

■ MOTS CLÉS : Entrepôt, composant logiciel pédagogique, indexation, métadonnées.

■ ABSTRACT : Although there are numerous research works in France about e-learning environments with a great quality, their visibility is rather weak. This is due to the difficulty to construct operational environment, with enough development to be experimented in the real world. The problem is that one need to develop a lot of complementary software in order to test a tool. Very often, these tools exist somewhere, but without being referenced. To avoid this shortcoming, these tools must be described and indexed in a repository, or in a set of distributed repositories. In this paper, we present the design and the implementation of such a repository, dedicated to components for designing e-learning environments. We also propose a metadata schema to describe them, inspired of existing schema such LOM, but with some complements, due to the fact that there are software components.

■ KEYWORDS : Repository, pedagogical software component, indexation metadata.

1. Introduction

Depuis plus de 20 ans, des projets de recherche en France portent sur la conception et le développement d'environnements d'apprentissage ou d'outils logiciels offrant de nouvelles fonctionnalités. Cependant, le passage à l'échelle est resté réduit malgré la qualité de beaucoup de ces travaux. Les causes de cet état de fait sont multiples mais il nous semble que la raison majeure vient de la difficulté à construire un environnement opérationnel complet, c'est-à-dire incluant toutes les fonctionnalités nécessaires à un usage en situation réelle. C'est pourquoi, même si les résultats de ces recherches sont intéressants, souvent le produit final reste inexploité. En effet, la majorité des projets de recherche n'a pas pour objectif de construire un EIAH complet et opérationnel mais de se pencher sur un problème particulier et d'essayer de lui trouver une solution. Le résultat est souvent une application ou un prototype rendant un service particulier. Faute d'intégration dans un EIAH opérationnel, le fruit de ces recherches reste souvent limité à la phase d'expérimentation. D'un autre côté, tester et faire fonctionner un outil en situation réelle obligent généralement les chercheurs à développer des programmes complémentaires, éventuellement disponibles ailleurs mais non référencés et donc inaccessibles. Pour mieux valoriser les travaux de recherche et éviter de re-développer plusieurs fois les mêmes fonctionnalités, une première condition est donc que les réalisations logicielles soient décrites et indexées dans un espace commun, l'espace de stockage pouvant, lui, être distribué[1].

C'est dans ce contexte que le Réseau Thématique Pluridisciplinaire "Apprentissage, Formation et Education" a lancé une action spécifique pour définir une plate-forme de mutualisation des travaux de la communauté française, voire francophone, de recherche sur les EIAH. Grâce à un travail de réflexion mené collectivement, nous avons défini un ensemble de scénarios d'usage d'une telle plate-forme qui a abouti à définir les fonctionnalités souhaitées : Déposer/Rechercher, Publier/Souscrire, Assembler, Évaluer/Expérimenter, Gérer un projet, Participer à la communauté de pratique (AS Plate-forme, 2005).

Parallèlement à cette initiative, un travail a été entrepris au Crip5 depuis octobre 2001 d'une part pour définir un schéma de métadonnées décrivant les composants logiciels pédagogiques et d'autre part pour concevoir et réaliser un entrepôt de composants logiciels dédié aux EIAH s'appuyant sur ce schéma de métadonnées. Bien entendu, le groupe de recherche du Crip5 a participé à l'action spécifique dans la mesure où l'objectif de cette recherche portait sur l'une des fonctionnalités essentielles (déposer/rechercher) de la plate-forme de mutualisation.

L'objet de cet article est de décrire les travaux réalisés, travaux qui permettront dans un proche avenir de tester le dépôt incluant la description, l'indexation et la recherche de composants.

Cet article est structuré de la manière suivante : Le deuxième paragraphe fait un rapide tour d'horizon des viviers de connaissances existants en montrant que le besoin que nous avons énoncé ci-dessus n'est pas couvert. Dans le troisième paragraphe, nous proposons une catégorisation des composants logiciels nécessaires aux EIAH. Dans le quatrième paragraphe, nous proposons un ensemble de métadonnées pour les composants logiciels pédagogiques. Dans le cinquième paragraphe, nous présentons l'architecture générale du système, les mécanismes de dépôt et de recherche ainsi que l'outil d'indexation et de recherche. Dans le sixième paragraphe, nous terminons l'article par une conclusion, quelques perspectives ouvertes par ce travail et une annexe sur quelques considérations techniques.

2. Quand les viviers de connaissances et les métadonnées ignorent les composants logiciels

Les travaux que nous avons menés découlent d'un constat simple : La communauté EIAH ne s'est guère préoccupée du recensement des constituants applicatifs de leurs plates-formes de formation. Ces

constituants, rarement développés sous forme de composants logiciels, sont pourtant stratégiques. Ils sont la concrétisation du savoir-faire de la communauté et le résultat de son investissement en recherche. Combien de micro-mondes, de tuteurs intelligents, d'outils auteurs, de gestionnaires de scénarios pédagogique ne sont pas recensés ?

Cependant, la communauté s'est investie activement depuis une dizaine d'années pour créer des métadonnées, développer des outils d'indexation des ressources pédagogiques, des viviers de connaissances en vue de les stocker et des outils de recherches pour les retrouver.

À ce jour, il existe plusieurs schémas de métadonnées destinés à décrire les ressources pédagogiques. L'un des premiers à être utilisé fut le (Dublin Core, 2005). Il comporte 15 métadonnées destinées à décrire des ressources documentaires. L'alliance ARIADNE (Ariadne, 2005) est à l'origine du LOM (Learning Object Metadata, défini par le consortium IMS) (LOM, 2002), standard résultant d'un compromis entre les acteurs de la communauté à l'échelle internationale. Bien que doté de 80 éléments classés en 9 rubriques, ce schéma est critiqué parce qu'il n'est pas en adéquation avec les pratiques et les besoins du terrain pour décrire des ressources pédagogiques (Najjar et al., 2003), (Afnor, 2002). SCORM (ADL, 2001), qui rassemble des acteurs du ministère de la défense aux États-Unis, comporte aussi un jeu de métadonnées construit sur la base des travaux de l'Advanced Distributed Learning (ADL) et l'Aviation Industry CBT Committee (AICC, 2005).

Ces schémas ne sont pas adaptés à la description des composants logiciels constituant les EIAH car ils ne comportent que peu de métadonnées pour décrire l'aspect logiciel spécifique de ces composants comme les besoins techniques (matériels et logiciels) pour les faire fonctionner, leurs caractéristiques (services, propriétés, méthodes) et encore moins les règles et les dépendances d'assemblage nécessaires aux informaticiens développant les EIAH. Cette situation nous a amené à proposer, dans le paragraphe 4, un jeu de métadonnées dédié spécialement aux composants logiciels.

En ce qui concerne les viviers ou les entrepôts de ressources pédagogiques, il en existe plusieurs. ARIADNE (Ternier et al., 2003) dispose d'un vivier de ressources pédagogiques. La région Rhône Alpes finance, dans le cadre de GreCo, le projet ARPEM (Archivage de Ressources PEdagogiques Multimédia) (Mermet et al., 2002) de capitalisation, d'indexation et de mutualisation contrôlée des ressources pédagogiques universitaires de l'Académie de Grenoble. Au Canada, le CRSNG (Conseil de Recherches en Sciences Naturelles et en Génie) (CRSNG, 2005) a accordé une subvention de 7,5 millions de dollars canadiens pour une période de 5 ans à la Télé-université et à ses partenaires universitaires, au nombre de six, pour monter le réseau (LORNET, 2005) qui a commencé il y a un peu plus d'an. Parmi les objectifs de ce réseau, nous retrouvons le rassemblement dans un vivier des ressources pédagogiques produites. Ce réseau est un des rares qui intègre dans ses objectifs la gestion des composants logiciels intégrables dans le système d'opération du téléapprentissage TELOS. "Athabasca Digital Library in a Box" (ADLIB, 2005) d'Athabasca University est un autre exemple canadien d'entrepôt de ressources pédagogiques indexées selon le LOM.

L'un des problèmes avec ces viviers ou entrepôts de ressources pédagogiques est qu'ils ne supportent qu'un seul jeu de métadonnées, celui défini une fois pour toutes lors de leur création. De ce fait il est impossible de les utiliser pour indexer des composants logiciels sans leur apporter des modifications majeures. En effet, les outils de description, les bases d'index ou de stockage et les outils de recherches sont taillés sur le modèle des métadonnées qu'ils prennent en charge.

Récemment, le réseau thématique pluridisciplinaire, s'est engagé en lançant une action spécifique pour spécifier une plate-forme de mutualisation des travaux de la communauté de recherche sur les EIAH. Cette plate-forme est définie pour comporter un entrepôt hébergeant les composants logiciels utiles aux informaticiens concepteurs de plates-formes de formation. Elle permettra, à terme, de retrouver des composants logiciels pour les assembler et construire des environnements d'apprentissage à des fins d'expérimentation ou d'exploitation. La construction de cet entrepôt est une tâche à laquelle nous

participons. Dans le paragraphe 5 de cet article, nous explicitons le modèle que nous avons adopté pour le construire. Devant la complexité de la définition d'un jeu de métadonnées d'une part et d'autre part devant la nécessité absolue que ce jeu repose sur le plus large consensus possible, nous avons défini un modèle de conception de la plate-forme qui nous permet d'avoir une indépendance entre l'entrepôt et le schéma de métadonnées utilisé pour la description des composants logiciels. Ainsi, même si nous proposons un jeu de métadonnées, la plate-forme construite est générique dans le sens où elle peut être paramétrée relativement aisément par tout jeu de métadonnées.

3. Les composants logiciels dans les EIAH

3.1 Quels composants ?

La notion de composant est ancienne mais, encore aujourd'hui, plusieurs définitions sont proposées dans la littérature (Allen et al., 1998), (Brown et al., 1998). Même si ces définitions sont différentes selon le contexte et les technologies dans lesquelles elles ont été énoncées, elles ont fondamentalement le même objectif : la réutilisabilité (Ezran et al., 1999), (Sommerville, 1992). Par réutilisation, on entend la possibilité de construire une nouvelle application en récupérant le code et les parties de programmes développées auparavant. Pour ce qui nous concerne, nous proposons la définition suivante, inspirée de la définition de (Szyperski, 2002).

Définition D1

Un composant logiciel est une entité autonome de déploiement respectant un modèle de composant[2], qui encapsule des codes informatiques et qui décrit par des interfaces les interactions qu'il autorise avec d'autres composants.

Un avantage essentiel des composants est qu'il est possible de les assembler soit pour créer des applications soit, par recomposition, pour créer de nouveaux composants de granularité plus importante.

Cependant, contrairement aux ressources pédagogiques, où la notion de granularité paraît une notion essentielle, nous estimons que la granularité n'est pas un facteur clé. A priori, la granularité pourrait être liée au nombre de services offerts. Mais, on ne voit pas, contrairement aux ressources pédagogiques, comment définir des classes homogènes si ce n'est celle des composants unitaires ne comportant qu'un seul service. Nous pensons qu'il est préférable d'indiquer, comme mesure de la complexité du composant, la profondeur d'assemblage ce que nous appelons *ordre de composition*. Cet ordre de composition est défini par :

- ordre (composant élémentaire) = 1;
- ordre (composant) = 1 + max (ordre (composants utilisés pour le construire))

Cela nous paraît plus intéressant pour donner une idée sur la taille du composant. Quoi qu'il en soit, la granularité n'est pas un critère de sélection.

Dans le cadre de la plate-forme de mutualisation (décrite dans le paragraphe 5), nous ne voulons stocker que des composants logiciels utiles aux EIAH. Cependant, réaliser un logiciel complet nécessite des composants intervenant à différents niveaux du système. Nous introduisons ainsi, dans le cadre des EIAH, quatre classes de composants logiciels : les composants logiciels pédagogiques (CLP), les composants logiciels de services(CLS), les composants logiciels techniques (CLT) et les composants logiciels de fabrication (CLF).

Remarque

Dans la communauté EIAH, les développements logiciels sont encore rarement faits sous forme de composants. Cette situation rend, la réutilisation extrêmement délicate si ce n'est impossible. Certains

travaux (Rosselle, 2003) ont montré cette difficulté et ont fait des propositions de solutions pour y remédier.

3.2 Classification des composants participant à la composition des EIAH

Selon le rôle que les composants logiciels jouent dans l'EIAH, nous avons constaté que leurs descriptions sont pour une part spécifiques. Pour faciliter leurs recherches par les futurs utilisateurs de la plate-forme de mutualisation, nous les avons regroupés en classes, de telle sorte que les composants d'une même classe aient un schéma de description commun. Nous avons obtenu ainsi 4 classes, homogènes du point de vue de la description des composants logiciels y figurant.

3.2.1 Les composants logiciels pédagogiques

Définition D2

Un composant logiciel pédagogique (CLP) est un composant "métier" participant directement au processus d'apprentissage humain.

Cette notion de "composant métier" est conforme à l'approche Ingénierie logicielle Dirigée par les Modèles (IDM ou en anglais Model Driven Architecture – MDA) qui préconise de séparer les aspects métiers des aspects techniques lors de la construction d'une application (Blanc, 2005). Des exemples typiques sont les gestionnaires (ou players) de scénarios pédagogiques, les simulateurs ou les résolveurs de problèmes à vocation pédagogique, les outils d'évaluation de l'apprenant. Ce sont des composants réutilisables essentiellement dans les EIAH.

Remarque

Un problème de vocabulaire se pose car, généralement, les auteurs parlent de "composants pédagogiques" pour désigner des "ressources pédagogiques" ou des "objets pédagogiques" (Bourda, 2001), (Grandbastien, 2002), (Pernin, 2003). Une "ressource pédagogique" est un élément numérique contenant une information destinée à l'enseignement et susceptible d'intervenir dans la constitution d'un cours. Un "objet pédagogique" est un élément constitué d'un ensemble d'objets et/ou de ressources ayant un objectif pédagogique et une durée déterminée. Par exemple, une image, un schéma ou une démonstration de théorèmes sont des ressources pédagogiques alors qu'un ensemble de pages web avec cours, exercices, et démonstrations constitue un objet pédagogique. Les Assets dans SCORM sont d'autres exemples de ressources pédagogiques. Cependant la différence est parfois difficile à faire : une applet Java réalisant une simulation peut être considérée comme une ressource ou comme un objet selon son contenu.

D'après la définition D2, les CLP ayant des contenus disciplinaires peuvent être aussi, selon le cas, considérés comme des "objets pédagogiques". C'est le cas de certaines applets Java, de certains composants Flash. Il n'y a inclusion ni dans un sens, ni dans l'autre : un objet pédagogique ne joue le rôle d'un CLP que s'il comporte des instructions de traitements. De ce fait, un cours html en ligne n'est pas un CLP alors qu'une applet de simulation d'une expérience en physique l'est. Inversement, certains CLP ne sont pas des objets pédagogiques. Par exemple, un player de scénario pédagogique ne fait pas partie en tant que tel d'un cursus de formation. Ce n'est donc pas un objet pédagogique.

Il faut donc éviter de confondre les objets pédagogiques, les ressources pédagogiques et les composants logiciels pédagogiques (CLP).

3.2.2 Les composants logiciels de service

Définition D3

Un composant logiciel de service (CLS) est un composant apportant une plus-value fonctionnelle aux EIAH, mais non spécifique aux EIAH.

C'est un composant utile, mais pas indispensable, pour garantir la vocation pédagogique de l'EIAH. Il offre des services annexes et assure une meilleure ergonomie et maniabilité pour les utilisateurs. C'est le cas, par exemple, des synthétiseurs vocaux, des adaptateurs d'interfaces ou des outils de communication (messagerie, chat, forum, etc.). Il est utilisable dans les EIAH mais aussi dans d'autres domaines d'application.

3.2.3 Les composants logiciels techniques

Définition D4

Un composant logiciel technique (CLT) est un composant fournissant les mécanismes de base nécessaires à la constitution de l'infrastructure des EIAH pour garantir le bon fonctionnement des CLP et des CLS. C'est un composant n'apportant pas de fonctionnalités pédagogiques aux EIAH. Il est susceptible d'être utilisé dans tous les domaines.

Ces composants seront principalement utilisés pour la construction de l'infrastructure des plates-formes EIAH ou comme intermédiaires entre des CLP et des CLS peu interopérables. C'est le cas, par exemple, des composants de persistance (sauvegarde des données), d'impression, de formatage de texte (transformation d'un texte XML en pdf), d'adaptateur de structures de données (jouant le rôle de "glu" entre des composants), etc. Le constructeur d'EIAH disposera alors d'une large palette de composants logiciels techniques, utiles, lui facilitant sa tâche. La majorité de ceux-ci proviendront des environnements de développement ou des librairies des langages de programmation.

3.2.4 Les composants logiciels de fabrication

En plus des CLP, CLS et CLT, nous avons défini une quatrième classe appelée CLF (Composant Logiciel de Fabrication) regroupant les constituants des outils auteurs permettant de créer des scénarios pédagogiques, des images animées, des QCM, etc. Ces éléments, qui ont un rôle en amont de la phase d'utilisation des EIAH, sont néanmoins des composants qu'il est utile de partager dans la plate-forme de mutualisation.

3.3 Structure d'un EIAH

Selon les précédentes définitions, un EIAH peut être vu comme l'assemblage de :

1. un ensemble de composants logiciels interconnectés. Selon nos définitions, ces composants peuvent être des CLP, des CLS et des CLT;

2. un ensemble d'objets pédagogiques définissant les activités et les contenus propres à la formation en cours d'exploitation dans l'EIAH (voir figure 1).

Remarques :

- les objets pédagogiques peuvent comporter des CLP.
- les CLF ne font pas partie de l'EIAH.

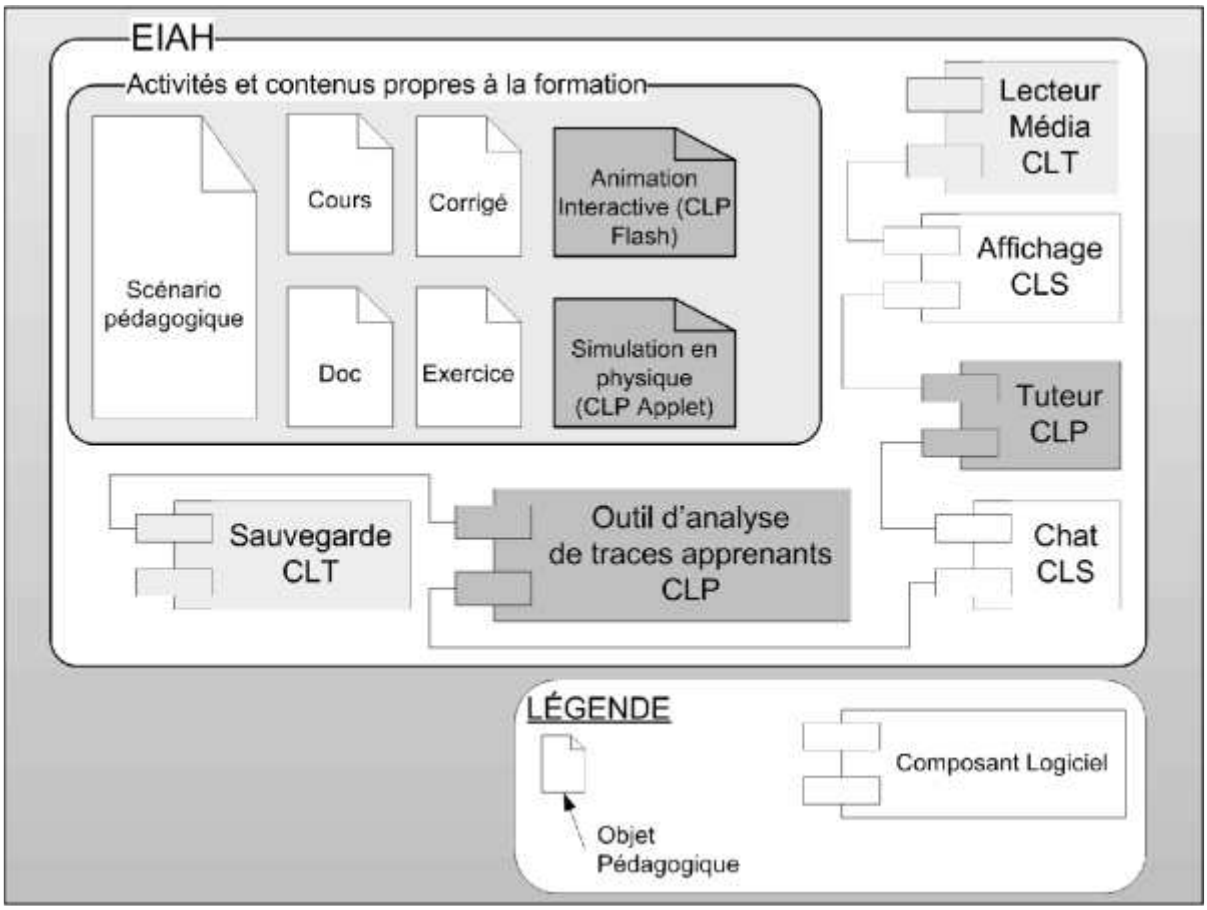

Figure 1 : Constitution d'un EIAH selon la vue composant

De ce fait, dans notre approche, un EIAH est vu comme un ensemble d'objets pédagogiques du domaine enseigné s'appuyant sur une architecture constituée par l'assemblage de CLP, CLS et CLT. Quant aux objets pédagogiques, ils peuvent être créés par les CLF.

Pour constituer un EIAH, il est non seulement nécessaire de trouver les objets pédagogiques mais aussi les composants logiciels de la plate-forme EIAH les supportant. Cela implique que les composants logiciels soient décrits et indexés finement grâce à un ensemble de métadonnées.

4. Une proposition de métadonnées pour les composants utiles aux EIAH

Les composants logiciels doivent être décrits avec des métadonnées adéquates assurant une double fonctionnalité. La première consiste à faciliter la recherche de ces composants selon plusieurs critères et la deuxième à connaître leurs caractéristiques pour savoir comment les manipuler. Définir ces métadonnées est une opération complexe car il faut être conforme aux règles reconnues de bonnes pratiques (Vidal et al., 2004).

4.1 Quel jeu de métadonnées ?

Pour que les métadonnées soient pertinentes, il faut, lors de leur création, veiller à respecter les points suivants :

1. Elles doivent être acceptées par une très large communauté. Chaque catégorie d'utilisateurs (concepteurs, développeurs, assembleurs, didacticiens, etc.) doit y trouver réponse à ses questions et ce à une échelle aussi large que possible. Idéalement elles

doivent être acceptées au niveau international sous la forme de standard.

2. Les valeurs permettant de remplir les champs doivent relever autant que possible d'un large consensus. C'est pourquoi, le plus souvent possible, il est souhaitable qu'une liste fermée de valeurs soit fournie. Selon les cas, cette liste peut être plate ou structurée, en particulier cette liste peut se présenter sous la forme d'une ontologie.

3. Les métadonnées doivent couvrir un maximum des caractéristiques de l'élément décrit. Cela engendre un nombre élevé de champs de description. Celles qui sont importantes pour la recherche doivent être obligatoires. Les autres doivent être facultatives afin que le processus ne soit ni trop lourd, ni bloqué par l'absence de valeurs.

4. Les métadonnées doivent être accompagnées de définitions claires évitant tout risque d'ambiguïté. Toutes les expériences montrent que les usagers sont réticents à passer du temps à remplir des champs dont la définition risque bien souvent de ne pas correspondre exactement à l'idée que se fait l'auteur de son composant.

5. La charge supplémentaire due au renseignement des métadonnées doit être réduite pour ne pas rebuter les usagers chargés d'indexer surtout s'ils n'ont pas un retour sur investissement rapide et perceptible. Cela passe par la mise à disposition de guides et d'outils pour faciliter le renseignement des champs.

Nous proposons, dans ce qui suit, un jeu de métadonnées, respectant au maximum les consignes précédentes, baptisé LSCM (*Learning Software Component Metadata*) pour décrire les composants logiciels intervenant dans la constitution des EIAH.

4.2 LSCM : Méthodologie d'obtention suivie

Face à la quasi absence de schéma de métadonnées dédié aux composants logiciels[3] et plus spécialement à ceux destinés à l'apprentissage et l'enseignement, nous avons décidé d'en construire un nouveau. Pour cela, nous avons commencé par recenser un maximum de schémas servant à décrire des documents numériques, des ressources pédagogiques et des logiciels. Nous avons également étudié des sites de téléchargement d'applications et les métadonnées contenues dans les descripteurs de déploiement des applications. À partir du Dublin Core, du LOM, de l'Open Software Description (OSD, 2005), du Basic Interoperability Data Model (BIDM, 1995), des champs de description trouvés dans les sites de développement collaboratif ou des sites de publication des Components-off-the-shelf (composant sur étagère) (COTS, 2005), nous avons établi une liste de tous les champs obligatoires et facultatifs qu'ils utilisent pour décrire leurs éléments. Cette liste, résultat de la fusion de tous les champs trouvés, nous l'avons "nettoyée" en supprimant les éléments synonymes. Après plusieurs itérations, nous avons classé les éléments obtenus dans des groupes homogènes (besoins logiciels, caractéristiques techniques, dépendances d'assemblage, etc.) facilitant leur organisation et leur repérage. Nous avons synthétisé ce premier résultat sur l'existant dans un document interne de travail non présenté ici.

Dans une deuxième phase, nous avons construit une autre liste sans utiliser l'étude précédente, c'est-à-dire en partant d'une liste vide. L'objectif de cette phase est de voir s'il est possible de trouver de nouvelles métadonnées utiles pour la description des composants logiciels mais non recensées dans notre étude de l'existant. Faute d'avoir pu mobiliser des experts motivés pour proposer de nouvelles métadonnées, nous avons joué le rôle de plusieurs acteurs (concepteurs, développeurs, assembleurs, utilisateurs de composants logiciels) en nous posant à chaque fois les questions suivantes : "Que voudrais-je savoir sur ce programme ou composant ?", "Que pourrais-je dire sur ce programme ou composant ?". Le résultat a été une liste de propriétés diverses et variées.

Dans une troisième phase, nous avons confronté ces deux listes en effectuant une étape de fusion manuelle, avec plusieurs cycles de nettoyage et une phase d'homogénéisation. Ensuite, en se basant sur

les comptes rendus des réunions présentant ce travail et les critiques faites dans des articles sur les schémas de métadonnées existants, comme le LOM, nous avons affiné l'organisation des éléments de notre liste, supprimé les éléments superflus et ajouté ceux qui étaient réclamés lors des présentations. Lors de cette phase, nous avons organisé plusieurs réunions restreintes de validation avec des chercheurs de différentes disciplines (informaticiens, didacticiens, pédagogues). Ces réunions ont permis de lever des ambiguïtés, d'ajouter d'autres éléments auxquels nous n'avions pas pensé auparavant et de retirer ceux qui étaient difficiles à renseigner. Cette opération a été itérée quatre fois avec à chaque fois une nouvelle version de travail de la LSCM. C'est ainsi qu'est née la première version de LSCM. C'est cette version que nous avons implémentée et qui sera mise à la disposition de la communauté pour expérimentation. Il reste à faire des tests d'usage auprès d'acteurs de terrain, ceux-ci nous ayant fait défaut dans ces trois phases.

4.3 Structure de la LSCM

Comme nous avons structuré les composants en quatre classes, nous avons choisi de subdiviser les métadonnées en deux sections :

- Une section commune appelée "SCM" (Software Component Metadata). Cette section est utilisée pour décrire les caractéristiques communes aux quatre catégories de composants présentées auparavant.

- Une section propre à chaque catégorie de composants pour décrire ses propriétés spécifiques. Cette section s'appelle xM avec x prenant les initiales du domaine. Pour les composants logiciels pédagogiques (CLP), cette section s'appelle LM (Learning Metadata), pour les composants logiciels de service (CLS), elle s'appelle SM. Ainsi chaque composant disposera normalement de deux jeux de métadonnées : Un jeu respectant le schéma SCM et un jeu respectant un schéma spécifique dépendant de sa catégorie. Dans les deux paragraphes suivants, nous présentons la section commune SCM et la section LM, spécifique aux CLP, car c'est celle qui concerne le plus la communauté EIAH.

4.4 SCM : Présentation succincte de la section commune

Les métadonnées de la section SCM permettent d'indexer tous les composants logiciels par rapport à leurs caractéristiques générales ou techniques indépendantes de leur domaine d'application. Elles sont organisées dans les douze rubriques suivantes, classées sous un élément racine nommée SCM :

1. Rubrique "Générale" : regroupant des informations d'identification et de description du composant ;

2. Rubrique "Meta-MetaData" : comportant des renseignements sur les métadonnées décrivant le composant ;

3. Rubrique "Cycle de vie" : décrivant l'état du composant, sa version et l'historique de son évolution ;

4. Rubrique "Contact" : permettant de trouver les coordonnées du fournisseur du composant et des services qu'il met à disposition des utilisateurs ;

5. Rubrique "Droit" : permettant de déclarer les droits de propriété intellectuelle et les conditions commerciales d'utilisation du composant ;

6. Rubrique "Informations Techniques" : renseignant sur quelques propriétés et caractéristiques techniques du composant logiciel comme sa structure ou l'approche de conception utilisée lors de son développement. C'est une des rubriques les plus

importantes du schéma SCM. Les informations qu'elle comporte sont essentiellement destinées à des informaticiens et permettent surtout de connaître les besoins matériels et logiciels nécessaires (par exemple le système d'exploitation, machine virtuelle,...) à l'utilisation du composant. Les éléments de premier niveau de cette rubrique sont présentés dans la figure 2. Le signe "+" devant certains d'entre eux indique l'existence d'une sous-structure masquée. C'est le cas de "Modele_Logiciel" permettant de préciser le modèle de conception logicielle utilisé lors du développement du composant comme le Model-View-Controller (Goldberg, 1984), (Improve, 2005), SEHEIM (Pfaf et Ten Hagen, 1993)... C'est aussi le cas de "Patron_Conception" permettant d'indiquer le Design Pattern utilisé lors du développement du composant logiciel. Enfin c'est le cas de "Modele_Developpement" qui décrit l'approche génie logiciel suivie ;

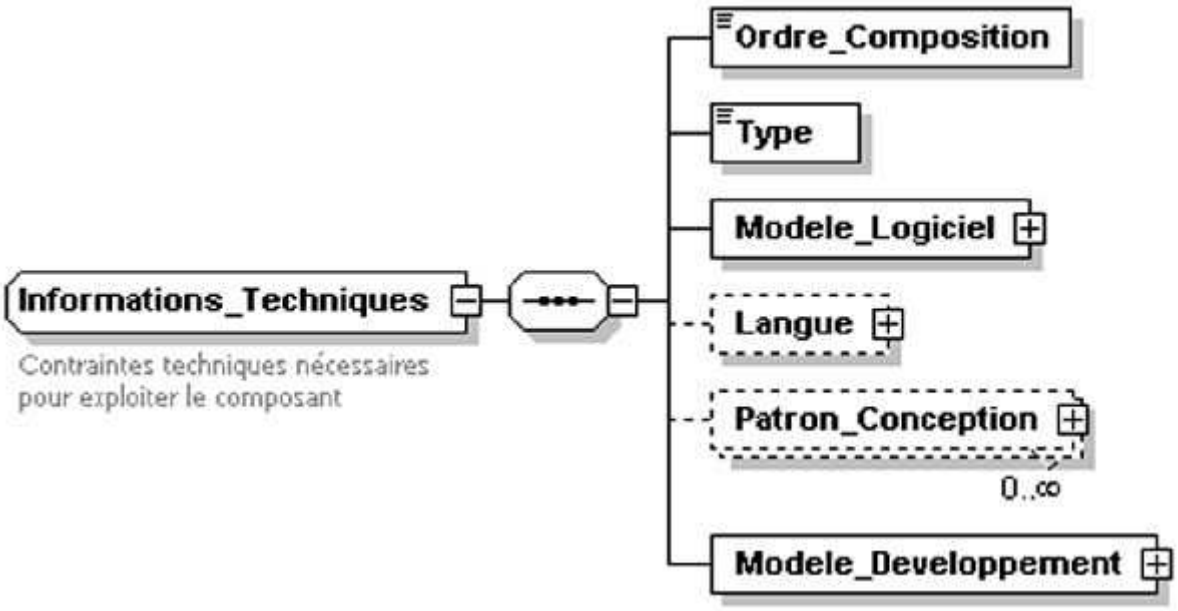

Figure 2 : Rubrique "informations techniques"

7. Rubrique "Relation" : énumérant les composants qui doivent être connectés avec le composant lors de l'assemblage pour le rendre fonctionnel. Elle contient des informations principalement destinées aux informaticiens assembleurs qui vont intégrer le composant par assemblage dans un projet. Ils retrouvent ici les connexions qu'ils doivent établir entre composants. Les éléments de cette rubrique sont donnés par le schéma de la figure 3. "Nature" qualifie le type de relation (requiert, exploite, est configuré, est visualisé...) entre le composant décrit et le composant ciblé par le lien. "Objet" permet de détailler le rôle et l'objectif de la relation entre les deux composants concernés par la connexion ;

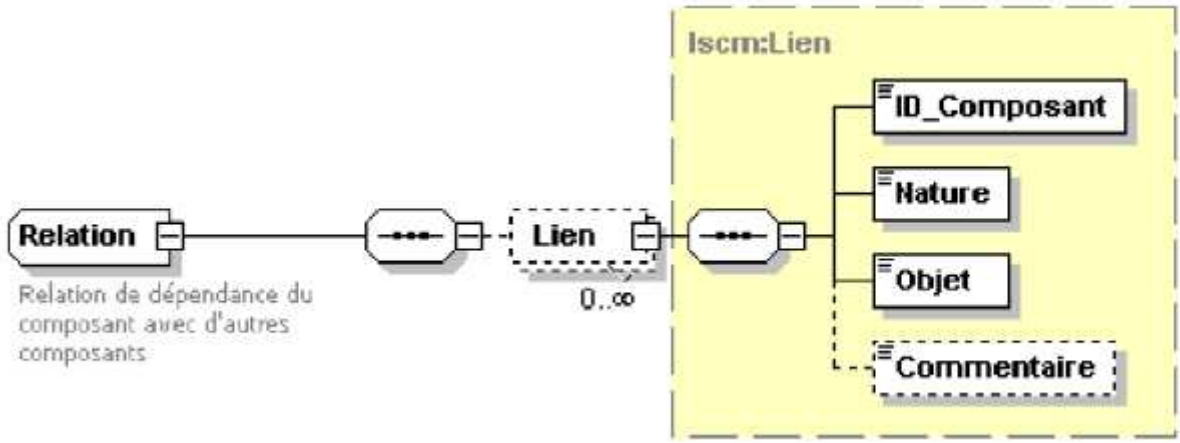

Figure 3 : Rubrique "Relation"

8. Rubrique "Caractéristiques composant" : listant les propriétés et les services offerts par le composant de point de vue informatique. Il s'agit des contrats signés entre le composant et son environnement extérieur. Ces contrats définissent comment les données doivent être échangées et comment les services doivent être appelés. Ils sont implémentés sous le concept d'interface au sens UML. Cette rubrique, la plus importante de la SCM, énumère donc les interfaces du composant. Les éléments d'une interface sont donnés par le schéma de la figure 4; L'élément Service possède la même sous structure que l'élément "Propriété" ;

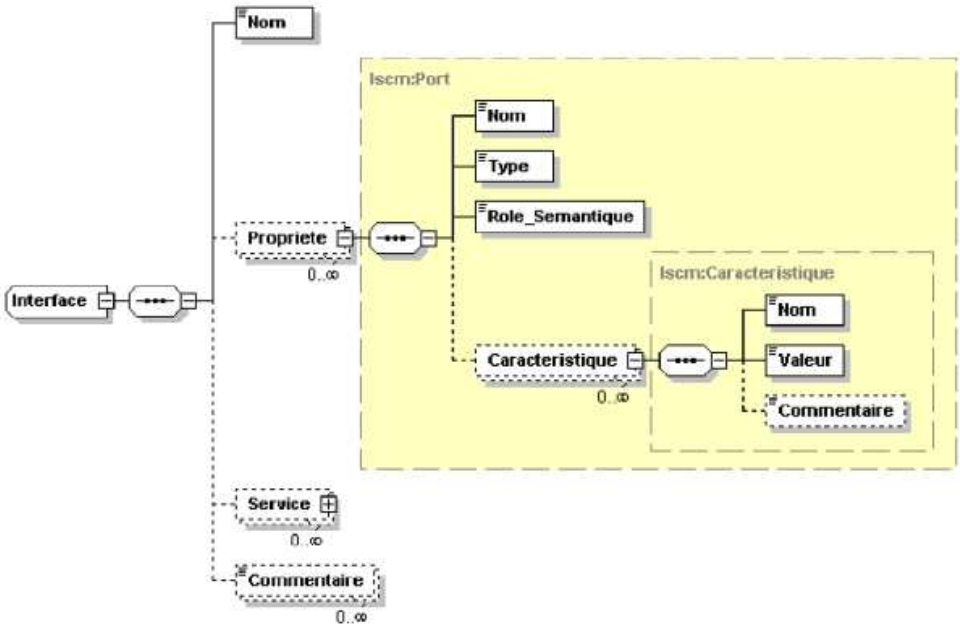

Figure 4 : Élément "interface" de la rubrique "Caractéristiques composant"

9. Rubrique "Documentation" : comportant des informations sur les documents ou les manuels joints au composant ;

10. Rubrique "Annotation" : regroupant des notes (de critique, de recommandation ou de suggestion) déposées par les personnes ayant manipulé le composant ;

11. Rubrique "Évaluation" permettant de noter le composant selon des critères de test

comme la performance, l'ergonomie, etc. ;

12. Rubrique "Extension" offrant le moyen de définir, de classifier le composant ou d'apporter des métadonnées supplémentaires.

Nous ne détaillons pas plus cette partie car cela nécessiterait de rentrer dans des détails techniques de bas niveau.

Si le LOM peut suffire à décrire des objets et des ressources pédagogiques, nous sommes convaincus qu'il ne permet pas de couvrir tous les besoins de description des composants logiciels pédagogiques. C'est pourquoi nous faisons cette proposition. Les différences principales portent sur la description des caractéristiques des composants.

Dans le LOM, les caractéristiques techniques d'un objet pédagogique se limitent aux sept métadonnées de la rubrique "Technique". C'est largement suffisant pour des entités simples comme la majorité des objets pédagogiques. Quand il s'agit d'éléments aussi complexes que les composants logiciels, cette rubrique devient rapidement insuffisante pour décrire les exigences matérielles et logicielles nécessaires au fonctionnement du composant. Comme nous venons de le voir, la LSCM comporte deux rubriques importantes par rapport au LOM : " Informations Techniques" et "Caractéristiques Composant". Ces informations sont essentielles pour les informaticiens car elles leur permettent de connaître les conditions nécessaires à la réutilisation et les connexions qu'ils doivent établir pour la réaliser. Ces deux rubriques disposent de plusieurs métadonnées dotées d'une structure générique permettant de définir les propriétés et de les renseigner. Dans le LOM, il n'existe aucune métadonnée permettant de décrire si finement ces aspects. Le schéma entier ainsi qu'un tableau établissant champ par champ une correspondance avec le LOM et DublinCore peuvent être consultés à l'adresse (LSCM, 2005).

4.5 LM : étude détaillée de la section spécifique aux CLP

La LM (*Learning Metadata*) est la section spécifique dédiée aux composants de la classe CLP. La LM comporte des métadonnées ayant trait au domaine de la formation, de l'apprentissage, de la didactique et de la pédagogie. Celles-ci sont décrites par six rubriques placées sous un élément racine nommée LM. Dans la suite nous détaillons le rôle et le contenu des champs principaux de ces six rubriques.

Remarques :

- Quand un champ peut prendre plusieurs valeurs, ce champ est répété dans la structure autant de fois que nécessaire.

- Certains champs, tels que les champs "commentaire" ou "identifiant" ne sont pas mentionnés pour alléger cette description mais, bien entendu, ils sont présents à chaque fois qu'ils sont nécessaires.

4.5.1 Rubrique Catégorie

Cette rubrique offre le moyen de classer les CLP en sous familles. Elle précise soit le type d'outils logiciels, soit la "classe d'environnements informatiques" (Tchounikine, 2002) dédiés à l'apprentissage dans laquelle le composant peut être rangé. Si le CLP est un outil logiciel, sa catégorie est précisée (par exemple "player de scénario", "traceur"). S'il est du type "classe d'environnements informatiques", la catégorie est définie en fonction du problème et du but ciblé par le composant et non par rapport aux disciplines d'enseignement (Mathématique, Physique, Anglais, etc.). La stratégie pédagogique sous-jacente supportée par le composant est dans ce cas un bon critère de qualification. Pour faciliter le renseignement de cette métadonnée, nous proposons une liste de valeurs devant être complétée pour couvrir toutes les classes d'environnements informatiques. Cette liste comporte entre autres les catégories "tuteur intelligent", "micro-monde", "composant support de pédagogie de projet".

4.5.2 Rubrique Utilisation

La rubrique "Utilisation" est obligatoire. Elle est principalement destinée aux usagers finaux du composant et aux concepteurs de logiciels de formation. Elle permet aux utilisateurs finaux, lors de la phase de sélection des composants candidats à un assemblage, de choisir ceux qui répondent à leurs contexte et conditions d'utilisation. Cette rubrique contient trois métadonnées précisant les conditions d'exploitation et les situations de formation prévues par les concepteurs et supportées par le composant. Ces métadonnées sont :

Cadre spatio-temporel

C'est un champ obligatoire indiquant le cadre temporel (synchrone, asynchrone, mixte) et le cadre spatial (à distance, présentiel, mixte). La combinaison de ces deux axes permet de construire une liste, associée à la métadonnée, comportant des termes comme : "Formation à distance synchrone".

Relation Inter Apprenants

C'est un champ obligatoire indiquant la nature des interactions apprenant-apprenant supportées par le composant. Plus précisément, cette métadonnée permet de répondre aux questions suivantes : Quels types de rapports peuvent avoir les apprenants entre eux en utilisant le composant ? Peuvent-ils travailler collectivement ou individuellement ? Auront-ils une relation de coopération ou de compétition ?

4.5.3 Rubrique Pédagogie

La rubrique "Pédagogie" est obligatoire. Elle décrit le composant selon un angle pédagogique en donnant des indications sur les méthodes et les démarches d'enseignement implémentées ou supportées par le composant et sur le contexte pédagogique dans lequel il est prévu de l'utiliser. Les informations de cette rubrique sont renseignées par les concepteurs du composant et servent aux utilisateurs pour décider s'ils veulent réutiliser le composant dans leur activité d'enseignement. Les valeurs de ces champs ne sont pas sujettes à évolution. Si le composant logiciel subit des modifications ayant un impact sur ses propriétés pédagogiques, alors il s'agit d'une nouvelle version. Dans ce cas, il faut créer une nouvelle instance de description.

Modèle

C'est le modèle pédagogique auquel les concepteurs du composant se sont référés pour programmer ses services. Le nombre de modèles pédagogiques est très réduit. Ils sont tous issues d'écoles de pensée. Nous associons à cette métadonnée une liste comportant les termes suivants : "constructiviste", "institutionnel", "transmissif", "appropriatif", etc.

Stratégie

Pour un modèle pédagogique donné, la stratégie pédagogique spécifie la méthode modélisée par le composant pour atteindre le but d'apprentissage. Une stratégie pédagogique est un ensemble de méthodes et de démarches formalisées et appliquées, qui vont déterminer des choix de techniques et de situations pédagogiques, par rapport au but de l'apprentissage. Bien sûr, la stratégie n'est pas complètement indépendante du modèle indiqué dans la métadonnée précédente, mais, à un modèle donné, peuvent, malgré tout, correspondre plusieurs stratégies. La liste de termes associée à cette métadonnée contient par exemple les stratégies : "découverte guidée", "découverte avec médiation", "résolution de problèmes", "exposition simple", "exposition avec dialogue", etc.

4.5.4 Rubrique Didactique

La rubrique "Didactique" est obligatoire. Contrairement à la rubrique "Pédagogie", on décrit ici les disciplines et les savoirs manipulés par le composant. Bien que le champ de la didactique soit bien élargi

aujourd'hui, l'objectif de cette rubrique se limite au renseignement de l'objet de l'enseignement, de la transmission des connaissances et des capacités ainsi que des moyens nécessaires au déroulement de l'enseignement. Cette rubrique comporte les éléments :

Domaine Enseignement

C'est un champ obligatoire précisant les disciplines ou les sous disciplines ciblées et manipulées par le composant. Ces disciplines sont généralement organisées sous forme d'une arborescence. La spécification d'une branche ou d'une sous discipline précise (feuille de l'arbre) est privilégiée afin d'affiner la description. Si le composant manipule plusieurs sous-disciplines n'appartenant pas à la même discipline générale, il faudra lui associer plusieurs instances de cette métadonnée. La figure 5 donne un extrait de l'arbre des disciplines. Bien entendu, ce sont les experts de chaque discipline qui devraient fournir une ontologie de leur domaine.

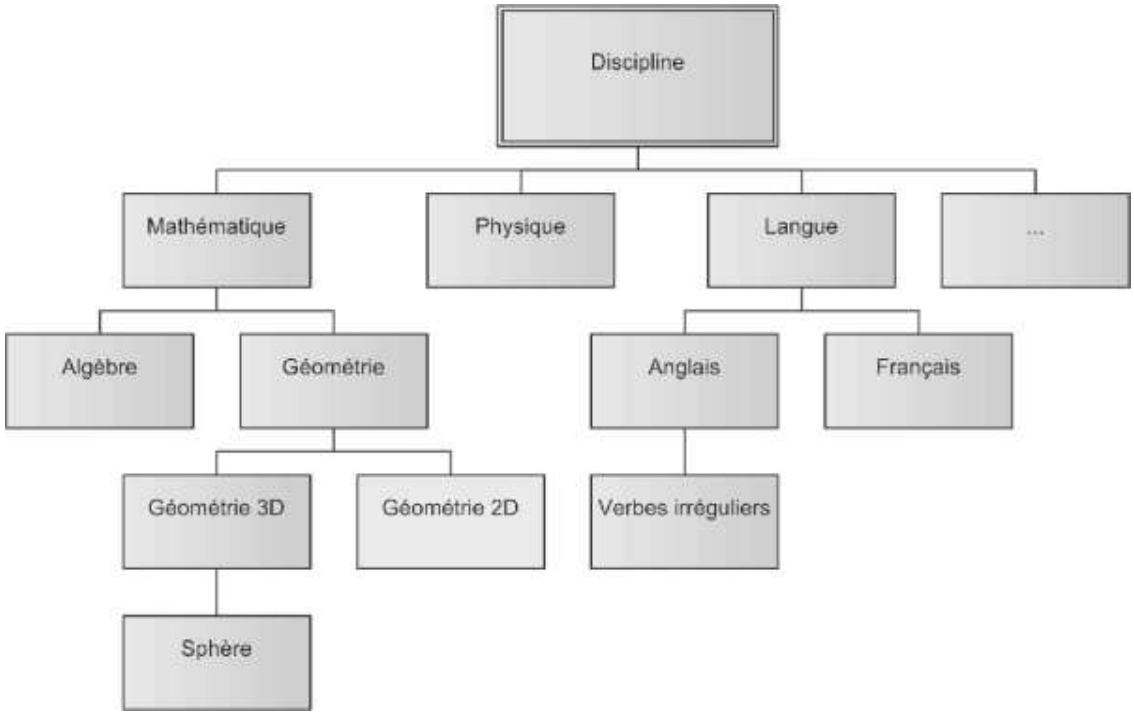

Figure 5 : Exemple d'arborescence des disciplines et sous-disciplines

But Didactique

C'est un champ obligatoire déclarant les effets attendus par l'utilisation du composant. Il permet entre autres d'outiller les futurs enseignants lors de la planification et de les sensibiliser au degré de difficulté variable que peut susciter l'usage du composant. Les buts sont déclarés par des phrases comportant un verbe d'action de type "être capable de". Par exemple, il est possible d'indiquer : "Maîtriser la résolution des équations du second degré". Chaque but doit être déclaré dans une instance spécifique de cette métadonnée.

Savoir

C'est un élément facultatif. En particulier, si le CLP est un outil logiciel ne manipulant aucun savoir particulier, cette rubrique n'est pas remplie. En revanche, si c'est un environnement dédié à l'apprentissage d'un domaine particulier, les connaissances théoriques, pratiques ou comportementales auxquelles le composant peut faire appel sont énumérées. Il s'agit des savoirs que le composant est capable d'analyser, d'interpréter, de traiter ou de communiquer à ses utilisateurs. Il est constitué des sept métadonnées suivantes :

- Pré-requis : champ permettant de recenser les connaissances qui doivent avoir été assimilées pour pouvoir acquérir celles qui sont présentées par le composant.

- Savoir enseigné : champ permettant de déclarer une connaissance théorique manipulée, enseignée ou transmise par le composant.

- Savoir-faire enseigné : champ permettant de déclarer une connaissance pratique manipulée, enseignée ou transmise par le composant.

- Savoir Être : champ permettant de déclarer une connaissance qui concerne les comportements et les attitudes à entreprendre dans une situation donnée.

- Savoir implicite : champ permettant de déclarer une connaissance théorique, manipulée implicitement par le composant, que l'apprenant doit de lui-même découvrir.

- Savoir-faire implicite : idem.

- Savoir Être implicite : idem.

Ces 7 métadonnées sont facultatives.

Ressource

C'est un élément facultatif permettant de renseigner les types de ressources pédagogiques manipulés par le composant pour fournir ses services. Pour chaque type de ressources, il faut créer une instance comportant les éléments suivants :

- Description : si l'élément ressource est instancié, l'élément description est alors un champ obligatoire renseignant de façon synthétique le type des ressources pédagogiques (format ou contenu). Par exemple si un composant sert à dessiner une pièce mécanique en 3D à partir d'un fichier contenant les caractéristiques de la pièce, il est possible d'écrire : "Propriétés en XML d'une pièce mécanique à dessiner en trois dimensions".

- Lien informations : c'est une structure facultative qui permet de retrouver une adresse url pour avoir accès à des informations complémentaires sur le type des ressources.

- Exemple : c'est une structure facultative donnant des références à des exemples concrets de ressource pédagogique manipulable par le composant.

Recommandation

C'est un champ facultatif permettant au fournisseur du composant de proposer des recommandations didactiques aux utilisateurs sur l'usage du composant dans une situation d'apprentissage.

4.5.5 Rubrique Exploitation

La rubrique "Exploitation" est facultative. Elle permet, dans le cas où le composant propose des services directement accessibles à des acteurs humains, de les énumérer et de décrire leurs propriétés. Il ne s'agit pas du manuel d'utilisation du composant. Son objectif n'est pas d'expliciter comment exploiter les services mais de préciser les interactions que peut avoir un acteur avec le composant. Cette rubrique comporte les éléments suivants :

Service

C'est un élément obligatoire permettant de décrire un service offert par le composant. Chaque service dispose d'un attribut "id" facultatif permettant de l'identifier de façon unique. Il servira pour faire des référencements par la suite. Les métadonnées de cette structure sont :

- Nom : Ce nom peut être une phrase telle que "Proposer des questions pour assister l'apprenant dans la démonstration de la perpendicularité de deux droites" ou "Dessin de figures géométriques simples".

- Caractéristique : Il sert par exemple à dire, pour le service précédent, que la "Dimension de dessin" est "2D".

Acteur

C'est un élément obligatoire permettant de décrire un acteur concerné par les services du composant et susceptible d'interagir avec. Les métadonnées de cette structure sont identiques aux métadonnées de "Service" :

- Nom : Il est possible d'associer à cette métadonnée une liste de valeurs comportant les termes : "apprenant", "co-apprenant", "enseignant", "tuteur", "formateur", etc.

- Caractéristique : ce sont les propriétés des acteurs qui doivent être décrites. Il est ainsi possible par exemple de dire que l'âge minimal d'un apprenant est de 8 ans.

Relation

C'est un élément facultatif permettant d'associer les services aux acteurs. Son objectif est de dire qu'un service "S" est destiné à un acteur "A". Il s'agit, en effet, de l'implémentation d'une association n-n, comme dans les bases de données relationnelles. Chacune doit comporter les métadonnées suivantes :

- Nom : Aucune contrainte n'est imposée sur son contenu. Il est toutefois recommandé, si possible, de lui affecter un verbe d'action, par exemple "Montrer...", "Offrir...", "Permettre...", "Corriger...".

- Service : champ obligatoire permettant d'identifier le service concerné par la relation.

- Acteur : champ obligatoire permettant d'identifier un acteur concerné par le service.

- Caractéristique : comme pour "Service" et "Acteur", cet élément facultatif permet de décrire les propriétés de la relation. Pour dire, par exemple, qu'un service n'est utilisable qu'une fois, il faut créer une caractéristique "Nombre de lancement" et lui attribuer la valeur "1".

4.5.6 Rubrique Durée

Cette rubrique est optionnelle. Elle permet d'énumérer les actions ou les usages du composant ayant ou nécessitant une durée d'exécution pouvant avoir un impact sur le bon déroulement de l'apprentissage. Les fournisseurs donnent, ici, une estimation de la durée d'exécution ou d'exploitation de l'action. La rubrique "Durée" comporte une seule métadonnée "Action".

Elle permet de renseigner l'intervalle de temps minimal et l'intervalle de temps recommandé d'une action. Une action correspond à l'exécution, en cascade, d'un ou de plusieurs services. Cette métadonnée comporte les champs suivants : Nom, Durée Minimale, Durée Recommandée, Unité de Mesure temporelle.

4.5.7 Bilan pour la LM

En ce qui concerne les CLP, nous retrouvons donc dans la LSCM toutes les métadonnées définies dans le LOM. De ce fait, il est possible, à partir d'une description LSCM, de renseigner toutes les métadonnées d'une description LOM. L'inverse n'est pas vrai car la description ci-dessus introduit des champs inexistants dans le LOM comme " But Didactique " ou " Pre-requis ".

Il est important de noter que, si nous avons essayé d'être exhaustifs dans la description de ces CLP, il y a seulement 7 champs obligatoires, a priori assez faciles à remplir, dans la mesure où à la plupart d'entre eux sont fournis avec une liste de valeurs. Il s'agit de : " catégorie ", " EspaceTemps ", " relationInterApprenants ", " Modèle ", " Stratégie ", " DomaineEnseignement ", et " ButDidactique ". La minimisation du nombre de métadonnées obligatoires est important car l'effort demandé pour les remplir doit absolument être limité.

Les métadonnées propres des composants logiciels appartenant aux classes CLS, CLT et CLF ne font pas l'objet d'une présentation dans cet article.

5. Plate-forme de mutualisation

Nous présentons dans cette partie l'architecture de la plate-forme de mutualisation ECR (Educational Component Repository) que nous avons développée. Par le terme plate-forme, nous n'entendons pas un environnement de formation, une n^{ième} plate-forme d'enseignement à distance, mais un environnement destiné aux chercheurs et membres de la communauté. De quoi s'agit-il ?

Dans le cadre de l'action spécifique lancée par le Réseau Thématique Pluridisciplinaire "Apprentissage, Formation et Éducation", plusieurs fonctionnalités souhaitées par la communauté ont été recensées. La plate-forme ECR vise à couvrir la fonctionnalité Déposer/Rechercher. Il s'agit de doter la communauté d'un espace de partage et d'échange des composants logiciels comme c'est le cas pour les objets pédagogiques dans ARIADNE.

Cette espace permet, par exemple, à un chercheur qui a développé un module de résolution d'exercices d'algèbre de le mettre à disposition de la communauté pour être intégré à un EIAH et expérimenté en situation réelle. Nous verrons dans le paragraphe 5.5 comment il faut procéder pour chercher un composant logiciel.

Mais, avant de décrire comment utiliser la plate-forme pour déposer et rechercher des composants logiciels, nous présentons la plate-forme elle-même (voir figure 7) que nous avons souhaité être le plus générique possible, comme nous l'avons déjà mentionné dans le paragraphe 2.

5.1 L'ECR, une instance de plate-forme générique

En ce qui concerne la plate-forme, nous nous sommes fixés, lors de la conception, une contrainte forte qui a eu pour conséquence de complexifier sa conception et sa réalisation. Cette contrainte postule que la plate-forme soit générique dans le sens précisé ci-après. Nous avons défini un modèle de conception de la plate-forme qui nous permet d'avoir une indépendance entre l'entrepôt lui-même et les divers éléments (schéma de métadonnées, base d'index, formulaires à remplir) utilisés pour la description et l'indexation des composants logiciels. Autrement dit, elle doit être totalement indépendante des métadonnées qu'elle va gérer[4]. Cet aspect nous a paru essentiel dans la mesure où :

1. Nous voulons que cette plate-forme puisse servir pour d'autres schémas de métadonnées que celui que nous proposons. Ainsi, comme indiqué dans la figure 6, si nous désirons réaliser un vivier pour des objets pédagogiques, semblable à celui d'ARIADNE, nous pouvons instancier la plate-forme générique avec le LOM. Si nous voulons réaliser une bibliothèque de documents numériques, nous pouvons l'instancier avec le Dublin Core. Cette plate-forme peut même servir pour un référentiel des publications de la communauté EIAH : Nous pouvons créer un schéma décrivant les caractéristiques des articles et l'utiliser pour instancier la plate-forme générique.

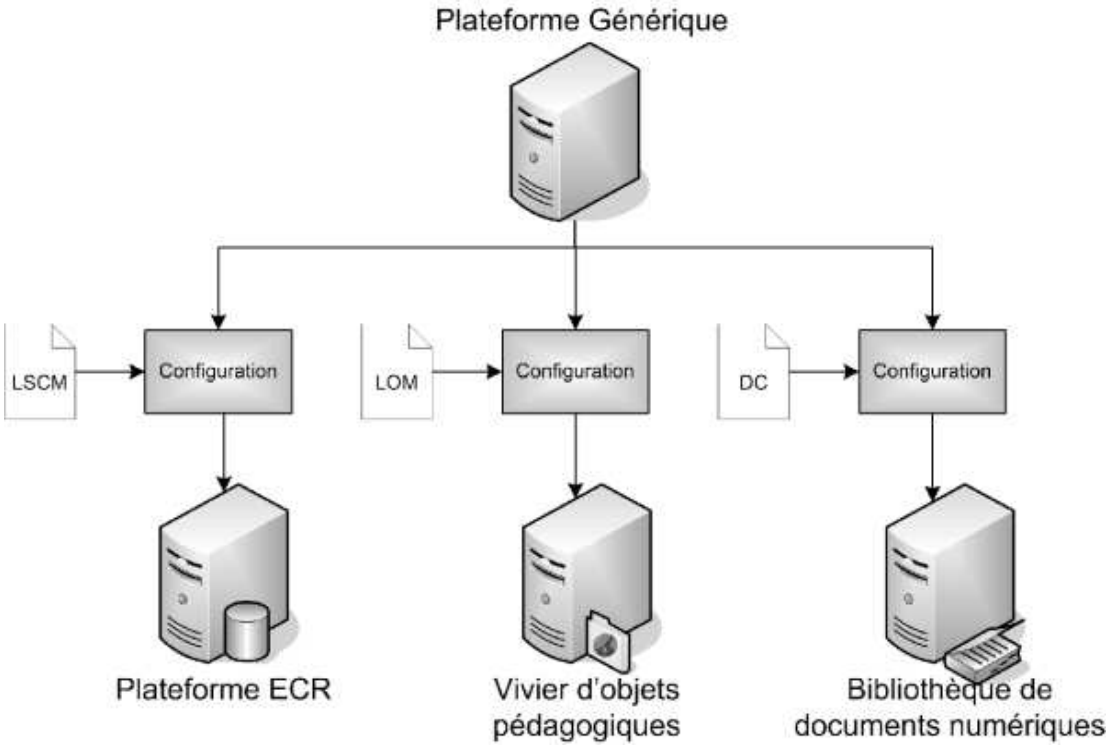

Figure 6 : Instanciation de la plate-forme générique dans divers contextes.

2. À supposer même que notre schéma de métadonnées soit accepté, un tel schéma de description est souvent sujet à des changements afin de suivre l'évolution des besoins. De plus, nous avons souhaité que notre plate-forme puisse accepter des composants déjà décrits avec d'autres schémas. Pour traiter ces deux situations, nous avons donc également conçu notre système d'indexation et de recherche pour qu'il puisse s'adapter à ces évolutions et gérer plusieurs schémas différents. Cette flexibilité lui confère une bonne généricité et une ouverture sur de futurs modèles de descriptions. Autrement dit, si nous trouvons des composants logiciels décrits avec d'autres schémas de métadonnées sémantiquement compatibles, il est possible de les indexer directement.

Cela a impliqué que :

- la base d'index et la base de données XML ne soient pas formatées. Autrement dit, elles doivent être configurées lors du paramétrage de la plate-forme;

- les formulaires d'interaction ne soient pas définis. Ils doivent être générés lors du paramétrage de la plate-forme;

- la plate-forme comporte des outils de configuration afin de l'instancier avec les propriétés des modèles des documents à indexer.

Comme le montre le schéma (figure 7), nous avons développé la plate-forme sous forme de composants (voir annexe pour quelques informations sur les composants logiciels utilisés) en respectant le modèle MVC (Model-View-Controller). Les composants les plus importants sont le composant d'indexation et celui de recherche. Les processus "déposer", "indexer" et "rechercher" offerts par la plate-forme sont décrits dans les paragraphes suivants.

Le composant "configuration Entrepôt" permet de paramétrer les deux composants d'indexation et de recherche ainsi que la base d'index et la base de données XML. L'administrateur l'utilise pour analyser le modèle (schéma XSD ou instance XML) des documents à indexer. L'analyse recense les éléments XML

(métadonnées) pouvant être indexés et détermine le type d'indexation le plus approprié pour chaque élément. Le résultat est affiché dans un formulaire à l'administrateur. Ce dernier ajuste le paramétrage automatique et valide la configuration. Le composant se charge alors de formater les bases de données et de configurer les paramètres des composants.

Le composant d'administration permet à l'administrateur de gérer la base de gestion des entités déposées. Cette dernière gère les connexions utilisateurs, leur droit d'accès aux entités et le suivi des statuts de ces dernières.

Enfin, l'application *e-package* (voir figure 7) est une application cliente destinée aux déposants de composants logiciels. Une fois installée sur le poste client, elle permet de renseigner les métadonnées des composants logiciels et de construire un paquetage contenant les composants à envoyer au serveur.

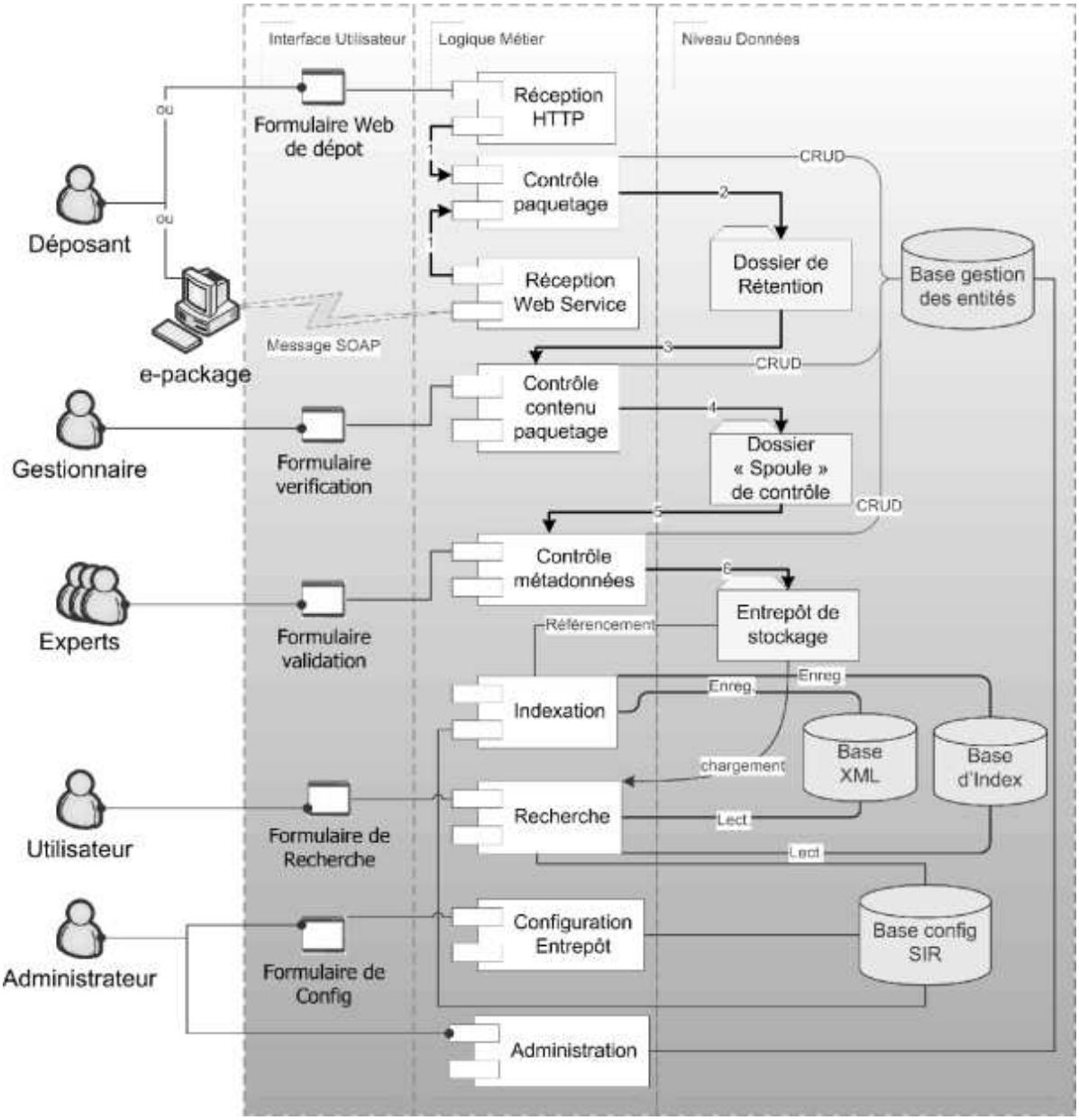

Figure 7 : Plate-forme ECR "Educational Component Repository"

Dans ce contexte, réaliser la plate-forme ECR revient à configurer la plate-forme générique avec le schéma LSCM et à développer les pages spécifiques (page d'accueil, documentation, etc.) du site

Internet.

5.2 Processus de l’action "déposer"

Revenons au cas du chercheur voulant déposer son logiciel de résolution d’exercices d’algèbre. Premièrement, il doit lister tous les composants qui sont nécessaires pour que son logiciel fonctionne. Pour cela, il doit créer, manuellement ou moyennant l’outil e-package une fiche de description, respectant le modèle LSCM, et ceci pour chaque composant logiciel. Il regroupe par la suite le code binaire du composant, la documentation, optionnellement le code source, et les fiches de métadonnées dans une archive au format ZIP portant l’extension .ecp (ecp : Educational Component Package). Un fichier *manifeste* décrivant le contenu du paquetage doit y être inclus. Il envoie ensuite ce paquetage au serveur moyennant l’application e-package qui établit une communication Web Service ou en se rendant sur le serveur à travers un formulaire de dépôt. Dès lors le paquetage est pris en charge par les responsables de la plate-forme de mutualisation.

5.3 Processus de contrôle du paquetage

Au niveau du serveur (voir figure 7), la réception du paquetage ecp déclenche un processus de contrôle de la validité du paquetage. S’il a le bon format, la bonne extension et si le manifeste est valide, il est mis en attente dans un dossier de rétention. Dans le cas contraire, il est retourné au déposant. À chaque entrée d’une archive dans le dossier de rétention un message est envoyé au gestionnaire de l’entrepôt. Celui-ci vérifie que le contenu du manifeste est cohérent et que les fiches de métadonnées sont valides par rapport au schéma LSCM. Si ce n’est pas le cas, il rejette le paquetage et le renvoie au déposant. En cas de succès du contrôle, le paquetage est mis dans un dossier "spoule"[5] de contrôle. Quand plusieurs paquetages sont présents dans le "spoule" de contrôle, un message est envoyé aux experts chargés de contrôler la validité de la description des composants pour leur signaler qu’il y a de nouveaux paquetages à traiter. Si des anomalies sont détectées dans les métadonnées, ils contactent le déposant du composant pour lui signaler les modifications à apporter. L’opération est itérée jusqu’à ce que les métadonnées soient valides. Quand c’est le cas, le paquetage est transféré à l’entrepôt de stockage. Les fichiers de métadonnées sont alors extraits et transmis vers l’outil d’indexation. Ils sont alors stockés dans la base XML et indexés. La base d’index est alimentée par le résultat de l’indexation. Le paquetage, lui, est archivé dans un dossier dépôt. Un message est ensuite envoyé au déposant pour lui signaler l’acceptation et la publication de son paquetage et de ses composants. À partir de cet instant, les composants peuvent être recherchés via l’outil de recherche intégré à l’ECR.

Remarque :

Les paquetages en attente dans le "spoule" de contrôle peuvent être indexés dans une base d’index secondaire. Si les composants de ces paquetages peuvent être recherchés, ils sont marqués "non valides". Ils n’ont pas la garantie de conformité donnée par les experts attestant de la conformité des métadonnées avec les caractéristiques réelles du composant.

5.4 Processus d’indexation

Indexer un ensemble d’entités permet une recherche plus performante (plus rapide et plus pertinente). Pour ce faire, il faut créer un index qui est une liste ordonnée d'unités d'information appelées "mots-clés" et jugées pertinentes pour la recherche d'une entité. Ainsi, chaque entité indexée est liée à un ensemble de mots-clés la caractérisant, permettant à l'utilisateur de la retrouver. Dans notre cas il va de soi qu'indexer un composant logiciel signifie indexer sa description. La description est mise sous format XML pour faciliter l’échange entre les institutions et assurer une meilleure interopérabilité avec des systèmes informatisés hétérogènes, comme le souligne Michel Crampes (Crampes, 2003).

5.4.1 Champs de recherche : l'entrée de l'index

Dans la mesure où nous souhaitons disposer d'une recherche multicritère, il faut disposer de plusieurs index, chacun décrivant une même entité selon un point de vue différent. À chaque index est associée une étiquette appelée *champ de recherche*. L'ensemble des champs de recherche forme la *base d'index*. Au moment de la configuration, l'administrateur doit construire la base d'index en déclarant les champs de recherche.

La généricité de la plate-forme est assurée par l'indépendance de la structure de la base d'index par rapport à celle des documents décrivant les entités à indexer. Pour cela, le système extrait de ces documents les informations sous forme de *mots-clés* qui vont alimenter les champs de recherche. Cette indépendance a pour autre avantage d'autoriser l'indexation d'informations provenant de formats hétérogènes de données (Base de données relationnelle, fichier texte basique, HTML).

Les champs de recherche sont a priori plus stables (mais modifiables si nécessaire) que les schémas de métadonnées. Quand des documents de description sont écrits dans des formats autres que XML, ils sont transformés pour devenir conforme à l'ensemble des champs de recherche acceptés par l'outil d'indexation (voir ci-dessous étape 1). Grâce à l'utilisation de feuilles XSL[6] (outil de manipulation de fichier XML) il est possible de modifier la structure originelle du document tout en gardant une structure sémantique minimale. Cette solution simple permet (1) de résoudre le problème de l'hétérogénéité des structures en attribuant à chacune une feuille XSL spécifique et donc (2) de rechercher précisément une information sans connaître sa localisation dans le document.

5.4.2 Les étapes d'indexation

L'indexation se passe en 2 étapes principales.

Étape 1 – Transformation du document à indexer dans le format d'indexation XML.

Le but de cette étape est de mettre le document à indexer sous la forme de couples *{champ, texte à indexer}* dans un format XML, nommé *format d'indexation*. Le format d'indexation est choisi par le responsable de la plate-forme lors de son instanciation. Il contient les champs et les informations qui vont alimenter la base d'index. Le format d'indexation est constitué d'une liste d'éléments de type : `<champ>texte à indexer</champ>`. Par exemple, soit un document 1 contenant des métadonnées à indexer (voir Listing 1). Pour pouvoir les introduire dans la base d'index, le document 1 sera transformé par le système dans le format d'indexation (voir Listing 2). Comme nous pouvons le constater les noms des balises dans le document 1 sont différents de ceux des balises du format d'indexation. Comme il existe une corrélation sémantique, il est possible de réaliser cette transformation (Exemples : author contributeur, keyword theme).

Ainsi, lors de la définition du format d'index, le responsable de la plate-forme a la possibilité de choisir les informations les plus pertinentes pour effectuer les recherches. Dans l'exemple, les champs de recherches sont "recherche", "creation", "contributeur", "theme", "langage" et "description". Cela permet d'optimiser la recherche tout en réduisant la taille de la base d'index.

```
<doc>
<meta>
<creation date=”03/03/03”>
<author>Raymonde Bayles</author>
</creation>
<modification>
<author>Céline Maisonneuve</author>
</modification>
<language>java</language>
</keywords>
<keyword>indexation</keyword>...
<keyword>composant</keyword>...
</keywords>
<summary>This indexing system is amazing !</summary>
</meta>
......
</doc>
```

Listing 1 : Document 1 à indexer

```
<document uid=”xmldb://description/id=12”>
<creation>03/03/2003</creation>
<contributeur>Raymonde Bayles</contributeur>
<contributeur>Céline Maisonneuve</contributeur>
<theme>indexation</theme>
<theme>composant</theme
<langage>java</langage>
<description>This indexing system is amazing !</description>
</document>
```

Listing 2 : Transformation du document 1 dans le format d’indexation

Cette transformation dans le format d’indexation est effectuée à l’aide d’une feuille XSL, responsable du mapping entre la structure du document et les champs de recherche. Ceci permet de résoudre le problème d’hétérogénéité explicité auparavant.

Remarque :

Pour les documents écrits dans des formats autres que XML, en amont de l’indexation, il faut utiliser les fonctionnalités du framework (voir annexe technique) qui permettent de transformer divers formats de données en flux XML. À partir de là, nous avons affaire à une nouvelle structure. Il suffit, comme vu précédemment, d’appliquer des feuilles XSL pour régler le problème d’hétérogénéité.

Étape 2 – Indexation sur les champs de recherche typés

Une fois le document à indexer mis dans le format d’indexation, il suffit d’effectuer une indexation pour chaque couple *{champ, texte à indexer}* afin d’en extraire des couples {champ, mot-clé} (notre document xml peut finalement être vu comme un vecteur de couples *{champ, mot-clé}*). Ce processus d’indexation dépend du type du champ (nombre, date, mot-clé libre ou contrôlé, texte à analyser). Ces informations sont définies dans un fichier de configuration lors de l’instanciation de la plate-forme.

Par exemple, pour un champ type "texte", nous pouvons indiquer au système l’analyseur analytique et linguistique à utiliser pour extraire les mots-clés à indexer (stoplist, stemming, etc.). Cela permet entre autre de définir le filtre anti-dictionnaire pour ignorer les mots fonctionnels fréquents et le filtre de lemmatisation à utiliser. Autre exemple, pour le type "mot-clé contrôlé", il est possible d’imposer une vérification de l’appartenance des mots-clés à une taxinomie avant d’effectuer l’indexation (Grandbastien, 2002).

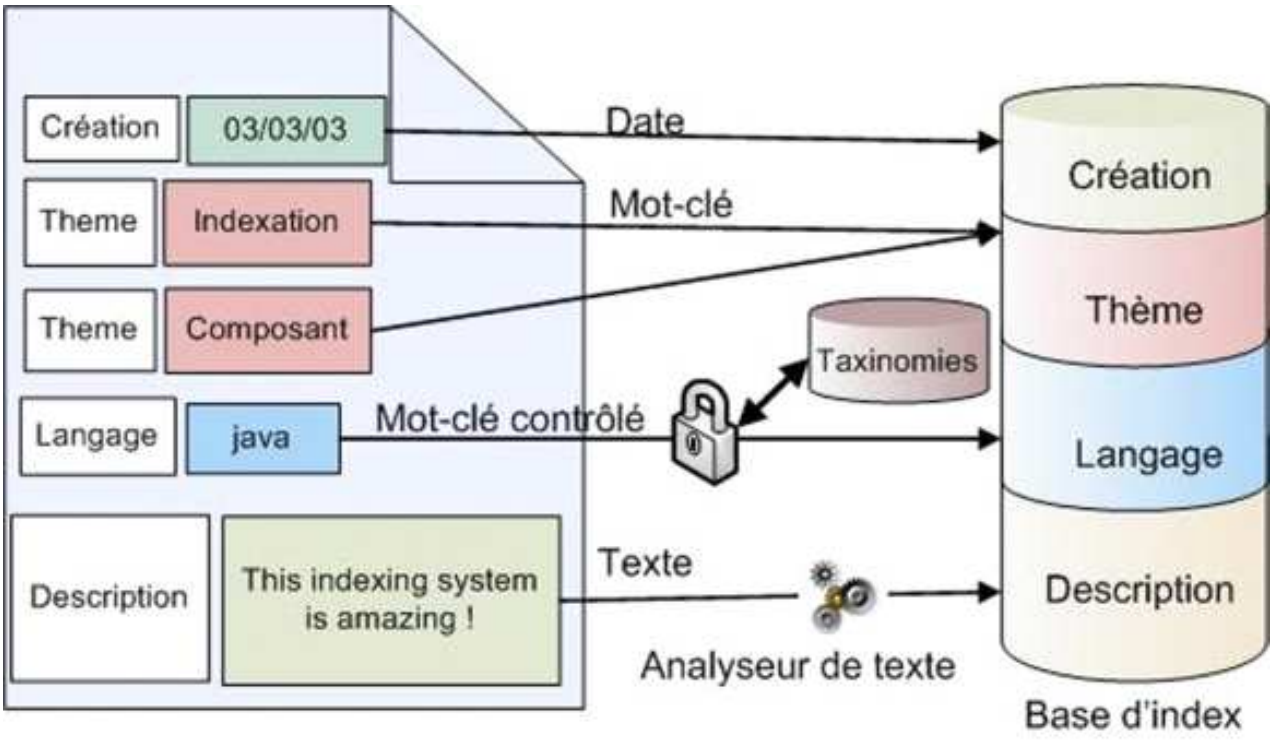

Figure 8 : Étape 2 – Indexation d'un document

5.5 Processus de l'action "rechercher"

Pour effectuer une recherche de composants, l'utilisateur sélectionne une combinaison de plusieurs champs de recherche appelée *profil de recherche*. Un profil de recherche définit essentiellement les champs sur lesquels sont réalisées les recherches et le *mode* utilisé (voir ci-dessous). Il est ainsi possible d'avoir un formulaire de recherche destiné à des informaticiens, des didacticiens, des enseignants, etc. Les recherches sont réalisées à partir de la base d'index. À la réception de la requête de recherche, le système fournit une liste des composants répondant aux critères de la requête. L'utilisateur consulte alors les métadonnées des composants qui l'intéressent. S'il décide de récupérer un composant, le paquetage le contenant lui sera transmis en entier.

La recherche dans la base d'index se fait selon deux modes complémentaires : la *recherche par mot-clé* (avec la possibilité d'une recherche multicritères) et la *recherche par navigation*. Dans la recherche par mot-clé, nous avons des fonctionnalités semblables aux moteurs de recherche sur Internet (genre Google). En ce qui nous concerne nous utilisons en interne le moteur de recherche fulltext *Lucene* (Lucene, 2003) d'Apache qui nous permet d'avoir une notion de pertinence dans les documents retournés.

Dans la recherche par navigation, se pose le problème de la classification de l'entité selon plusieurs critères en fonction des attentes de l'utilisateur. En effet, une entité complexe, comme un composant logiciel pédagogique, est assez difficile à classer selon un seul arbre hiérarchique car elle peut être vue de différentes manières. On peut par exemple souhaiter classer un CLP selon le critère modèle pédagogique, le domaine d'enseignement ou le langage de programmation dans lequel il est écrit. Pour répondre à cette difficulté de classement des résultats (entités complexes), nous utilisons le concept de classification par facette (Prieto-Diaz, 1991). Cela consiste en une collection de "petites" listes ou hiérarchies de classification orthogonales les unes aux autres (voir partie droite de la figure 9).

Dans l'exemple de la figure 9, nous avons deux facettes de classification : Celle de gauche permet de classer les entités selon le domaine d'enseignement de l'entité et celle de droite selon le niveau d'enseignement. Ces petites hiérarchies, nommées *facettes* sont des listes de mots-clés caractérisant un

aspect de l'entité. Chacune est alors classée suivant plusieurs vues indépendantes les unes des autres. L'un des avantages, par rapport à une classification hiérarchique unique, est qu'elle offre une meilleure compréhension de l'ensemble des données en séparant mieux les divers aspects de l'entité. Dans notre système, une facette est associée à un champ de recherche type *mot-clé* : elle peut donc être plate ou hiérarchique si le champ est contrôlé par une taxinomie ou une ontologie, par exemple exprimée avec OWL (OWL, 2005). En conjuguant plusieurs champs caractérisant chacun un aspect de l'entité et en classant les résultats suivant les mots-clés de ces champs, nous obtenons une classification multi facettes.

Figure 9 : Exemple de classification par facette – à gauche la facette thème, à droite la facette niveau

5.6 Langage de requêtes et de réponses XML

Nous avons créé un langage de requêtes, basé sur XML, et qui fournit un résultat dans une syntaxe compréhensible par Lucene. Par exemple, dans le listing 3, nous avons mis le code XML de l'exemple suivant de requête multicritères :

Rechercher les entités ayant dans les champs de recherche *info_technique* et *info_pedagogie* les mots-clés "Programme" et "logiciel" avec une priorité plus élevée pour le champ de recherche *info_technique*. Ces entités doivent avoir dans le champ de recherche *date* une valeur comprise entre "01/01/2000" et "01/12/2004" et, dans le champ de recherche *niveau*, pas le mot-clé "college".

On distingue trois types de requêtes unitaires :

- La requête définie par l'élément *fulltextquery*. Elle permet de rechercher les mots-clés présents dans l'attribut *query* dans les champs de recherche déclarés dans les sous-éléments *field.* Une syntaxe prédéfinie pour cet attribut (normalement rempli par l'utilisateur) offre des possibilités de recherche avancée : recherche booléenne, recherche par phrase exacte, avec joker, avec approximation de l'orthographe, par proximité de mots-clés, etc. L'importance des champs peut être pondérée (attribut *boost)* et ainsi influencer le calcul de la pertinence. Dans l'exemple donné, le champ *info_technique* est priorisé par rapport au champ *info_pedagogique*, ce qui aura pour effet de retourner en premier les documents contenant le mot-clé programme dans le champ *info_technique* plutôt que dans le champ *info_pédagogie*.

- La requête définie par l'élément *rangequery*. Elle permet de faire une recherche sur un intervalle dans un champ spécifié, en général des dates ou des nombres.

- La requête définie par l'élément *keywordquery*. Elle permet de rechercher les documents dans un champ type *mot-clé*. Elle est créée pour la recherche par navigation. S'il existe une taxinomie hiérarchique attachée au champ, le système la prend en compte en recherchant aussi les descendants du mot-clé. Dans le listing 3, la requête *keywordquery* réutilise la facette hiérarchique *niveau*, vue dans la figure 9, avec le mot-clé *college*. Cela a pour effet de rechercher l'ensemble des documents ayant les mots-clés *6eme*, *5eme*, etc. dans leur champ de recherche.

En plus des requêtes unitaires, il est possible de construire des requêtes composées. La requête définie par l'élément *group* groupe un ensemble de sous requêtes grâce à un opérateur logique ET/OU (attribut *operator*). À noter que chacune des requêtes précédentes possède l'attribut *prohibited* jouant le rôle d'un opérateur type *not*, comme c'est le cas pour la *keywordquery* dans le listing 3.

L'ensemble de ces types de requête permet une recherche aisée pour trouver l'entité souhaitée. Il est également possible de faire des recherches par affinement, c'est-à-dire d'exécuter une requête sur les résultats obtenus lors d'une précédente recherche grâce à l'attribut optionnel *onquery* de l'élément *query* où l'identifiant de la requête à affiner doit être précisé.

```
<search>
 <query onquery=123 >
  <group operator=”and”>
   <fulltextquery query=”programme AND logiciel”>
    <field name=”info_technique” boost=”2.0”/>
    <field name=”info_pedagogie” boost=”1.0”/>
   </fulltextquery>
   <rangequery field=”date” from=”01012000” to=”01012004” />
   <keywordquery field=”niveau” keyword=”college”
prohibited=true/>
   </group>
 </query>
 <results>
   <summary>
     <field name=”titre”/>
     <field name=”description_fr”/>
   <summary>
   <classified field=”theme”/>
   <classified field=”niveau”/>
 </results>
</search>
```

Listing 3 : Exemple de requête XML

La deuxième partie (élément *results*) d'une requête xml sert à configurer le formatage des résultats. Elle offre la possibilité de paramétrer le tri des résultats (pertinence ou selon un champ donné) de classer les documents suivant des facettes (élement *classified)*, de définir les champs à afficher pour le résumé des documents résultats *(élement summary),* de définir le nombre de résultats par page, etc. (voir listing 3).

L'exécution de la requête retourne, dans un format XML (voir listing 4), la liste des entités répondant aux critères de recherche. Pour l'exemple précédent, le système renvoie deux entités :

- Document "Programme et Logiciel" avec un score de 0.78

- Document " Programme Pédagogique pour le primaire" avec un score de 0.58

```
<results queryid=″1234″ numhits=″234″>
  <classified field=″theme″/>
   <keyword name=″Mass media″ numdoc=″12″/>
   <keyword name=″Museology″ numdoc=″4″/>
  </classified/>
  <classified field=″niveau″/>
   <keyword name=″Primaire″ numdoc=″234″>
     <keyword name=″CP″ numdoc=″12″/>
     <keyword name=″CE1″ numdoc=″222″/>
   <keyword/>
  <classified/>
  <list>
    <document uid=″http://document1″ score=″0.78″  idx=″1″ />
     <titre>Programme et Logiciel</titre>
     <description_fr> ... </titre>
    </document>
    <document uid=″http://document2″ score=″0.58″  idx=″2″ />
     <titre>Programme Pédagogique pour le primaire</titre>
     <description_fr> ... </titre>
```

Listing 4 : Résultat de la requête du listing 3

Comme nous l'avons vu, le système d'indexation et de recherche est flexible et permet des recherches complexes. De plus comme il a été développé sous forme de composant, il pourra lui aussi être décrit et indexé, et pourra servir dans d'autres domaines que les EIAH.

6. Conclusion et Perspectives

Capitaliser et réutiliser des composants logiciels pour construire des EIAH nécessitent de les décrire finement et de les mettre à disposition de la communauté dans une plate-forme de mutualisation. C'est à cette tâche que nous nous sommes attelés. Dans cet article, nous avons proposé sur le plan théorique et conceptuel :

- Une classification des composants logiciels dans les EIAH en quatre classes (Composant Logiciel Pédagogique – CLP, Composant Logiciel de Service – CLS, Composant Logiciel Technique – CLT et Composant Logiciel de Fabrication – CLF).

- Un modèle de métadonnées LSCM (Learning Software Component Metadata) pour décrire les composants logiciels destinés au EIAH. Il comporte deux sections : une section SCM (Software Component Metadata) commune aux quatre classes de composants et décrivant leur aspect génie logiciel et une section LM (Learning Metadata) spécifique à la classe CLP et décrivant l'aspect pédagogique, didactique et exploratoire du composant logiciel.

Sur le plan des réalisations logicielles nous avons implémenté la LSCM dans un schéma XSD du W3C. De plus nous avons développé les outils suivants :

- Un prototype d'une plate-forme générique permettant d'entreposer des entités et de les retrouver en réalisant des recherches sur leurs métadonnées.

- Deux composants d'indexation et de recherche d'informations dans des documents XML. Ils sont intégrés dans la plate-forme générique.

Comme il existe de nombreux schémas de métadonnées et plus encore de profils d'application (LOM-FR, CanCore, Normetic, etc.), pour que ce travail ait une chance de perdurer, il est essentiel que cette plate-forme soit *totalement* indépendante du schéma de métadonnées qu'elle utilise. "Totalement" signifie que

toutes les données utiles pour la configurer (schéma de métadonnées, champs de recherches, structure de la base d'index et de la base XML) sont externes à la plate-forme et surtout qu'à partir de ces données, toutes les interfaces utilisateurs (pour décrire et rechercher des composants) sont automatiquement générées par le composant de configuration de la plate-forme. Ceci permet de réutiliser la plate-forme avec d'autres schémas de métadonnées.

L'aspect générique de la plate-forme a été testé en produisant différentes instanciations de la plate-forme. D'une part en construisant la plate-forme dénommée ECR (Educational Component Repository) basée sur le schéma de métadonnées que nous proposons par ailleurs pour décrire les composants logiciels utiles pour construire des EIAH, d'autre part en construisant pour notre propre usage une plate-forme pour stocker des documents scientifiques. Dans les deux cas, nous avons obtenu une plate-forme qui fonctionne correctement pour déposer ou rechercher des entités numériques, que ce soit des composants logiciels ou des documents. Seule difficulté, les interfaces pour déclarer les fichiers de données nécessaires pour configurer la plate-forme ne sont pas totalement réalisées.

Il faudrait maintenant que l'entrepôt soit expérimenté par l'ensemble de la communauté française, tant sur le plan du jeu de métadonnées qui est proposé mais qui pourra évoluer grâce à l'indépendance de celles-ci vis-à-vis de l'entrepôt, que sur le plan de la réalisation informatique. Cet entrepôt devrait intervenir lui-même comme une brique de la plate-forme de mutualisation dont la spécification a fait l'objet de l'action spécifique présentée dans ce même numéro. Nous espérons même qu'à terme, si les tests se révèlent satisfaisants, il puisse être une brique du "Shared Virtual Lab", action fondamentale au sein du réseau Kaleidoscope.

Références

Références bibliographiques

AFNOR CN36 (2002). Afnor Commentaires français sur la LOM (draft 6.4) doc ISO, SC36N0255, 2002

ALLEN P., FROST S. (1998) Component-Based Development for Enterprise Systems: Apply the Select Perspective TM, SIGS Books/Cambridge University Press. ISBN 0521649994

IEEE Standard for Information Technology - Software Reuse - Data Model for Reuse Library Interoperability: Basic Interoperability Data Model (BIDM). IEEE Std 1420.1, 1995.

http://standards.ieee.org/reading/ieee/std/se/1420.1-1995.pdf (consulté en décembre 2005)

BLANC X., (2005) : MDA en action - Ingénierie logicielle guidée par les modèles, Édition Eyrolles, ISBN : 2-212-11539-3

BOURDA Y., (2001). Objets pédagogiques, vous avez dit objets pédagogiques ?. Cahier GUTenberg n° 39-40, p. 71-79.

BROWN A.W., WALLNAU K.C. (1998), The Current State of CBSE, IEEE Software September/October 1998

CRAMPES, M., RANWEZ, S., PLANTIE, M. et VAUDRY, C, (2003). Qualités d'une indexation portée par XML et une ontologie au regard d'un standard, in "Ressources numériques, XML et éducation", Sciences et Techniques Éducatives, Hermès, Hors série, 2003, pp. 105-135, ISBN 2-7462-0682-X

EZRAN M., MORSIO M., COLIN T. (1999). Réutilisation logicielle – Guide pratique et retour d'expériences. Paris, Édition Eyrolles. ISBN 2-212-09066-8

GOLDBERG A. (1984). Smalltalk 80, the Interactive Programming Environment. Addison-Wesley

GRANDBASTIEN .M.(2002) Quelques questions à propos de l'indexation et de la recherche de

ressources pédagogiques sur le Web. In Baron G.-L. et Bruillard E. Eds. Les technologies en éducation : Perspectives de recherche et questions vives. Paris : INRP - MSH - IUFM de Basse Normandie. p. 211-220.

LOM v1.0 (2002) Final Draft Standard for Learning Object Metadata. IEEE Standards Department. IEEE P1484.12.1-2002

MERMET J.-M., CARRERE C., BAILLY P. (2002) ARPEM : Où comment repérer les OPNIs ...Une expérience originale de capitalisation et de mutualisation d'objets pédagogiques multimédias, Conférence TICE, Lyon, 2002

NAJJAR J, DUVAL E., TERNIER S., NEVEN F.(2003), Towards Interoperable Learning Object Repositories: the ARIADNE Experience, IADIS International Conference WWW/Internet 2003, Algarve, Portugal, 5-8 November 2003, Vol. I, pp. 219-226.

PERNIN J.P.(2003) Objets pédagogiques : unités d'apprentissage, activités ou ressources ?. Revue "Sciences et Techniques Educatives", Hors série 2003.

PFAFF G., TEN HAGEN P.J.W. (1993). Seheim workshop on User Interface Management Systems. Berlin, Springer Verlag.

PRIETO-DIAZ R. (1991). Implementing Faceted Classification for Software Reuse. Communications of the ACM, 43(5):88 -- 97, May 1991.

ROSSELLE M. (2003) Étude des objectifs d'une plate-forme de coopération pour les EIAH, Actes de a conférences EIAH 2003 p379-390

SOMMERVILLE I., (1992). Le génie logiciel. Addison Wesley, France.

SZYPERSKI C. (2002), Component software beyond object-oriented programming. Addison Wesley, États-Unis.

TCHOUNIKINE P., (2002), Pour une ingénierie des Environnements Informatiques pour l'Apprentissage Humain, In: Revue I3, Vol. 2 n°1, p. 59-95

TERNIER S., NEUVEN F., DUVAL E., MACOWICZ M., EBEL N. (2003): Web Services for Learning Object Repositories : the case of ARIADNE Knowledge Pool System., Proceedings of WWW03 Conference, Budapest, 20-24 Mai 2003, Hongrie.

VIDAL Ph., ALIBERT A., BROISIN J. (2004), Normalisation et Standardisation des Objets d'Apprentissage: l'expÈrience ARIADNE, Colloque Miage et e-Mi@ge, Marrakech, Morocco 15-17 March 2004.

Références à des sites Internet

Advanced Distributed Learning, 2001. SCORM Metadata set. Available at: http://www.adlnet.org. (consulté en décembre 2005).

Athabasca Digital Library in a Box, http://library.athabascau.ca/, (consulté en juin 2005)

http://www.aicc.org/, (consulté en décembre 2005)

Ariadne Foundation, available on http://www.ariadne-eu.org/ (consulté en décembre 2005).

AS Plate-forme pour la recherche en EIAH, Contributions de l'Action Spécifique « Conception d'une Plate-forme pour la recherche en EIAH » à l'ingénierie des Environnements Informatiques pour l'Apprentissage Humain, *Revue STICEF*, Volume 12, 2005, ISSN : 1764-7223, mis en ligne le 21/02/2006, http://sticef.org

Projet Cocoon, Framework open source d'Apache, http://cocoon.apache.org (consulté en décembre 2005).

COTS A trader for COTS components. http://www.cotstrader.com. (consulté en décembre 2005)

Conseil de Recherches en Sciences Naturelles et en Génie, http://www.nserc.ca/indexfr.htm, (consulté en décembre 2005)

Dublin Core Metadata Initiative, http://dublincore.org/, (consulté en décembre 2005)

http://www.application-servers.com/articles/multicouches/index.1_0.html, (consulté en décembre 2005)

Réseau LORNET, http://www.lornet.org/, (consulté en décembre 2005)

Schéma de métadonnées: Learning Software Component Metadata, http://www.math-info.univ-paris5.fr/ECR/metadata/LSCM (consulté en décembre 2005)

Lucene, Apache Lucene project, http://lucene.apache.org (consulté en décembre 2005).

The Open Software Description Format, http://www.w3.org/TR/NOTE-OSD (consulté en décembre 2005)

OWL, Ontology Web Language www.w3.org/TR/owl-features/ (consulté en décembre 2005).

Exist DB, http://exist.sourceforge.net/ (consulté en 2005).

Xindice, http://xml.apache.org/xindice/ (consulté en 2005).

Annexe : Quelques considérations techniques

Choix du Framework

Parmi tous les frameworks[7] de développement web étudiés (Struts, Freemarker, Turbine, Cocoon), nous avons choisi Cocoon (Cocoon, 2005) qui est historiquement un moteur de publication web centré XML permettant de récupérer des sources de données variées (fichiers XML, HTML, SQL, LDAP, bases XML, etc.) et de les mettre sous forme XML. Il est particulièrement adapté pour la publication de données sur différents types de clients (navigateurs classiques, terminaux WAP, PDF, Web service) en utilisant des feuilles XSL. Il possède aussi des extensions pour la mise en oeuvre de portails, de l'internationalisation, de la génération d'images, etc. C'est une application web au sens J2EE, c'est à dire que Cocoon s'installe très facilement dans n'importe quel conteneur de servlets, tel que Tomcat. Avec la version 2 il est devenu un véritable framework de développement web MVC+, fondé notamment sur les concepts suivants :

- *Séparation des préoccupations.* Le contenu, le style de la présentation et la logique métier sont souvent créés par différents spécialistes. Cocoon vise à une séparation complète des trois couches, leur permettant d'être conçues, créées et gérées indépendamment, réduisant la charge de la gestion, augmentant la réutilisation du travail et réduisant le temps de mise en route.

- *Développement d'applications web à base de composants*. Cocoon implémente ces concepts autour de la notion de "component pipelines". Un pipeline est un processus de transformation à la chaîne de données composé d'un ensemble de composants. Chaque composant dans le pipeline est spécialisé dans une tâche particulière. Un pipeline doit être composé :

• d'un composant appelé *Generator*, responsable de la création de contenu XML;

• de 0 à plusieurs composants appelés *Transformer* responsables de la modification à la chaîne de ce contenu;

• et d'un composant appelé *Serializer* responsable du format de sortie du résultat.

- Un framework technique apporte les concepts, entités et mécanismes qui permettent de s'abstraire de certaines problématiques conceptuelles et techniques récurrentes (ex sécurité, logging, persistance des données).

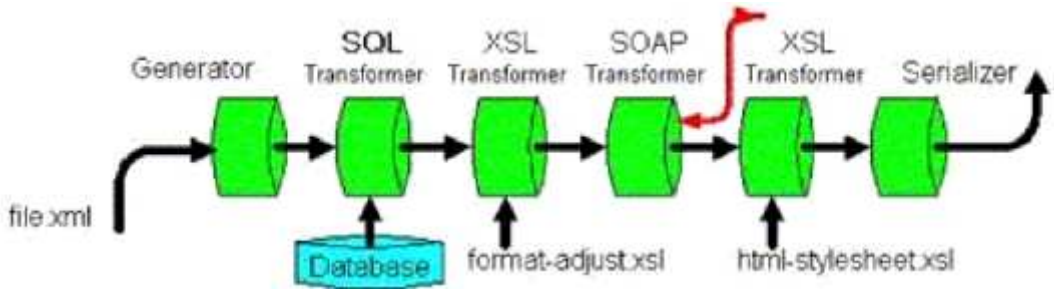

Figure 10 : Un pipeline comprenant un composant Generator, plusieurs composants Transformer (un transformateur relié à une base SQL, un autre relié à un web service) et un composant Serializer.

Choix du type de SGBD : XML ou relationnel

L'autre choix important (et plus discutable dans l'état actuel des standards et des réalisations logicielles) a été de choisir une base de données XML native.

Comme nous utilisons le standard XML comme format d'échange, nous disposons donc deux solutions :

- Soit mettre un "wrapper" entre XML et une base de données relationnelle, c'est-à-dire implémenter un mécanisme permettant de stocker des documents XML dans une base relationnelle de façon transparente. La majorité des SGBD commerciaux (Oracle ou DB2 d'IBM, SQL server) offrent ce service. Les SGBD gratuits (Mysql, PostGreSQL) commencent à intégrer cette fonctionnalité.

- Soit utiliser directement une base de données traitant nativement du XML. Nous avons testé deux bases XML natives gratuites et open source : Exist (Exist, 2005) et Xindice (Xindice, 2005) supportant les technologies XPath et XQuery et XUpdate (seulement pour Exist pour le moment).

Nous avons choisi la deuxième solution (la base de données Exist) car (i) les bases XML natives commencent à être matures et (ii) la structure de la base de données n'est pas statique contrairement aux bases relationnelles : on peut avoir dans la même collection, des documents de structures différentes. Lors d'un ajout ou d'une modification, on peut ou non valider un document XML suivant un schéma.

Choix du moteur de recherche interne pour l'outil d'indexation et de recherche

Enfin, nous avons choisi Lucene (Lucene, 2005) qui est une librairie java pour moteur de recherche et d'indexation "full-text". Il peut être utilisé comme noyau de toute fonctionnalité de recherche. Il fonctionne avec toutes sortes de données textuelles; toutefois, il n'offre pas de support intégré pour Word, Excel, PDF ou XML. Point important concernant Lucene : il ne s'agit que d'un moteur de recherche. Il

n'intègre pas de GUI web, ni de robot (web crawler). Il utilise un algorithme de recherche puissant et efficace comportant notamment un classement par score de pertinence, une recherche sur des intervalles de dates et sur plusieurs index. Lucene a été utilisé dans notre développement comme noyau du module d'indexation et de recherche dans l'entrepôt ECR.

---

[1] Remarquons que cette condition n'est pas suffisante : une autre difficulté est liée au fait que les logiciels développés, en tout cas dans la communauté EIAH, ne le sont que trop rarement sous forme de composants, ce qui rend la ré-utilisabilité difficile. Cet aspect de la question n'est pas l'objet de cet article.

[2] Il est important de noter qu'un composant logiciel ne respectant pas un modèle de composant est inexploitable dans les plates-formes de développement. Autrement dit, la notion de composant est indissociable de celle de *modèle de composants*. Une définition classique est : *Un modèle de composants définit des standards spécifiques d'interaction et de composition. La mise en œuvre d'un modèle de composants est un ensemble précis d'éléments logiciels requis pour permettre l'exécution de composants selon ce modèle*.
Plusieurs modèles de composants sont actuellement utilisés : J2EE, .Net, CCM, ActiveX, Java Beans, Fractal, Avalon, etc.

[3] La communauté génie logiciel s'est beaucoup moins investie que la communauté EIAH dans la définition de schémas de métadonnées.

[4] La seule contrainte que nous nous sommes fixée réside dans le fait que nous allons indexer des documents semi structurés, du XML en occurrence.

[5] Nous avons choisi ce terme de spoule que l'on voit dans certains dictionnaires d'informatique par analogie avec le "spool" d'impression qui est constitué de la liste des fichiers en attente d'impression (ici en attente de stockage définitif).

[6] XSLT est un langage de règles utilisé par les processeurs XSLT. Il permet de travailler sur le contenu d'un document XML. Il offre la possibilité de le transformer en divers documents XML ayant des structures différents. Outre cette transformation il permet également de générer des formats tels que HTML, RTF (Rich Text Format) ou XSL-FO (ex : pdf)

[7] Les frameworks sont des structures logicielles qui définissent des gabarits dans lesquels s'insèrent les objets spécifiques à une application. En pratique, un framework, ou squelette d'application, est un ensemble de classes qui fournissent, aux applications s'appuyant dessus, des services métiers et parfois des services techniques.
Un framework métier fournit des services à forte valeur fonctionnelle (gestion de portail, moteur de recherche, gestion de news, ...).
Un framework technique apporte les concepts, entités et mécanismes qui permettent de s'abstraire de certaines problématiques conceptuelles et techniques récurrentes (ex sécurité, logging, persistance des données).

A propos des auteurs

Issam Rebaï est ATER à l'université René Descartes (Paris 5). Il termine une thèse menée sous la tutelle conjointe du laboratoire CRIP5 et de l'ESIEA Recherche (École Supérieure d'Informatique Électronique Automatique). Son sujet de thèse porte sur la conception et le développement d'une plate-forme permettant la capitalisation de composants logiciels pour les environnements d'apprentissage humain.

Adresse : Laboratoire CRIP5 – Université René Descartes (Paris V), 45 rue des Saints Pères 75270 Paris

Cedex 06

Courriel : issam.rebai@math-info.univ-paris5.fr

Toile : http://www.math-info.univ-paris5.fr/~rebi

Nicolas Maisonneuve est ingénieur diplômé de l'ESIEA. Il a poursuivi en obtenant le DEA IARFA de l'université Paris6 au cours duquel il a participé à ce projet. Il travaille maintenant à l'université de Sydney.

Courriel : n.maisonneuve@gmail.com

Jean-Marc Labat a soutenu une thèse en intelligence artificielle (90) portant sur les environnements informatiques pour l'apprentissage humain, puis une habilitation à diriger des recherches (97), il a ensuite été promu professeur d'informatique à l'université René Descartes où il a dirigé pendant 4 ans le laboratoire Crip5. Depuis février 2005, il a rejoint l'université Pierre et Marie Curie. Il est responsable d'une équipe de recherche au sein du Lip6 et dirige L'UTES, un service commun de l'université. Il est l'actuel président de l'ATIEF. Ses travaux portent sur la modélisation et le suivi de l'apprenant, et plus généralement sur la modélisation cognitive de l'utilisateur en situation de résolution de problèmes.

Adresse : L'UTES, Université Paris 6, 12, rue cuvier 75005 Paris

Courriel : Jean-Marc.Labat@upmc.fr

Toile : www.lip6.fr/~labat/

Référence de l´article :

Issam Rebaï, Nicolas Maisonneuve, Jean-Marc Labat , Un entrepôt pour stocker et rechercher des composants logiciels utiles aux EIAH, *Revue STICEF*, Volume 12, 2005, ISSN : 1764-7223, mis en ligne le 08/03/2006, http://sticef.org