

EadGen, un environnement ouvert de production en ligne basé sur XML et XSL-T

Marc Nanard, Jocelyne Nanard

▶ **To cite this version:**

Marc Nanard, Jocelyne Nanard. EadGen, un environnement ouvert de production en ligne basé sur XML et XSL-T. Technologies de l'Information et de la Communication dans les Enseignements d'ingénieurs et dans l'industrie, Nov 2002, Villeurbanne, France. pp.157-165. edutice-00000653

HAL Id: edutice-00000653
https://edutice.hal.science/edutice-00000653

Submitted on 7 Oct 2004

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

EadGen, un environnement ouvert de production en ligne basé sur XML et XSL-T

Marc Nanard, Jocelyne Nanard

1 : Conservatoire National des Arts et Métiers, Paris
1 et 2 : LIRMM, UMR CNRS/ Université Montpellier II
161 rue Ada, 34392 Montpellier Cedex 5
nanard@lirmm.fr

Abstract

This article describes EadGen, an authoring environment designed and implemented for producing documents for distant learning. It results from a collaboration between CNAM and CNRS. EadGen offers an open production chain which is fully programmable by specialists (author, collection director, graphist…), each one being responsible of his own expertise and working with his own specialized tools. The production chain runs from the content authoring to the delivering of on-line documents. It automatically transforms learning material into interactive documents, according to the specifications elaborated by the collection director in order to build customizable learning paths. These specifications result from negotiation between the authors and the collection director and correspond to mediatisation requirements which are common to a set of courses. The requirements are expressed by means of open authoring languages which can be adapted on demand, owing to XML extensibility. XSL-T transforms are used to express how final documents are produced. The original point of our approach is to make XML use transparent to users by providing adaptable mark-up languages which offer a light notation and the semantics of which can be easily specified.
Keywords: Authoring tools, XML, XSL-T, Open architecture.

Résumé

Cet article décrit l'environnement auteur EadGen que nous avons conçu, développé et mis en service pour le CNAM, avec le concours du CNRS pour la production de documents pédagogiques en ligne. Il offre une chaîne de production « ouverte », totalement paramétrable par des spécialistes (auteur, responsable de collection, graphiste…), chacun étant responsable de sa propre expertise et disposant de ses propres outils spécialisés. La chaîne de production va de la production du contenu par l'auteur à la mise en ligne des documents. Elle transforme, de façon automatique, une matière première pédagogique en des documents interactifs, conformément aux spécifications d'un responsable de collection pour construire des parcours pédagogiques personnalisables. Elle s'appuie sur la spécification des besoins de médiatisation utiles dans un ensemble de matières, ces besoins étant négociés entre les auteurs et le responsable d'une collection. Ils sont exprimés au moyen de langages auteur «ouverts », adaptables à la demande, grâce à l'extensibilité de XML. La spécification de la production du document cible est exprimée au moyen de transformations XSL-T. L'originalité de cette approche est d'alléger pour l'auteur l'emploi de XML en lui substituant des langages de balisage adaptables, dont la notation est beaucoup plus légère et dont la sémantique peut être facilement spécifiée.
Mots-clés : Environnements auteurs, XML, XSL-T, langages ouverts.

Introduction

L'évolution rapide des techniques a entraîné une prise de conscience croissante du besoin de formation tout au long de la vie et de parcours pédagogiques individualisables. Parallèlement, la démocratisation de l'accès à Internet a permis un essor très rapide de l'enseignement à distance et, par voie de conséquences, de la production massive de documents pédagogiques en ligne. Pour être de qualité, cette production doit être spécifique et adaptable. Elle ne peut se limiter à mettre en ligne d'anciens supports de cours, et d'exercices. Toutefois, leur existence ne peut être négligée car ils constituent une source importante de connaissances déjà pré-organisées. Le coût de rétro-ingénierie pour convertir des documents pédagogiques existants en documents en ligne de haute qualité et les intégrer de façon méthodique dans des cursus personnalisables peut être réduit de façon significative, si les techniques informatiques mises en jeu sont suffisamment flexibles et extensibles pour s'adapter facilement à la diversité de forme de la matière première à traiter. XML, The eXtensible Markup Language (xml 1998 ; Michard 1999), est reconnu comme un des moyens les plus efficaces pour prendre en compte cette diversité (de La Passardière 2001). Toutefois, son usage direct reste fastidieux. Un environnement de travail simple et

facilement configurable est nécessaire pour rendre son usage accessible à des auteurs de plus en plus nombreux et non spécialistes de l'informatique (Murray 1999).

Cet article décrit l'environnement auteur EadGen conçu pour le CNAM (Conservatoire National des Arts et Métiers) avec le concours du LIRMM (CNRS). Il est utilisé par le SIFOD, service de formation à distance du CNAM, pour la production de documents pédagogiques en ligne. Une version de démonstration est disponible en ligne (Nanard 2002-b). EadGen permet, à partir d'un balisage extrêmement léger introduit dans la forme textuelle de la matière première (cours, exercices, etc..), de produire de façon totalement automatique des ensembles de documents homogènes dans leur forme et leur style d'interaction, constituant de véritables «collections» d'ouvrages au sens éditorial du terme. Ceci est rendu possible par le caractère ouvert, configurable et extensible du langage de balisage. Ce dernier est défini conjointement par le responsable de la collection et les auteurs, en fonction des besoins et des contraintes de la collection. Ce langage de balisage a pour rôle de décrire de façon déclarative les propriétés de la matière première pédagogique, indépendamment de toute mise en forme. Il sert à identifier les rôles structurels, sémantiques, rhétoriques des éléments que manipule l'auteur, et rend ainsi possibles les traitements automatiques qui assurent une mise en forme de qualité. La technique interne utilisée repose sur l'emploi systématique de XML et vise à réduire de façon drastique les coûts de production et à rendre la maintenance très aisée et bon marché. Parallèlement, un effort de réflexion méthodologique a été conduit pour proposer les méthodes de travail associées à cette nouvelle technologie.

L'article est composée de trois parties. La première présente le contexte et les motivations de ce travail. La seconde présente l'organisation de l'environnement EadGen. La troisième s'intéresse aux langages mis en jeu dans EadGen. Enfin une discussion évalue l'intérêt de ce type d'approche.

Contexte et Objectifs

Production de Documents Pédagogiques en Ligne

La production de documents pédagogiques en ligne destinés à l'EAD peut tirer profit de l'existence de supports de cours, mais ne peut se limiter à une simple adaptation de ceux-ci. En effet, le terme même de « support de cours » suggère que de tels documents ne sont que les compléments, souvent légers, à un cours existant en propre. Les mettre en ligne, sans en adapter à la fois la structure, le contenu et la forme, ne serait que changer leur mode de distribution. A l'opposé, les documents en ligne d'un cours en EAD doivent contenir tout ce dont a besoin l'apprenant, tant en ce qui concerne les exercices d'entraînement, d'auto évaluation, que le contenu détaillé de l'information qui serait donnée en cours. Plus encore, l'auteur d'un cours en EAD doit y inclure ce qu'il transmettrait oralement ou même par gestes en amphithéâtre. Pour rendre cet effort éditorial supplémentaire économiquement possible, il faut pouvoir transférer une part significative du budget de production sur celui de la conception. Ceci n'est possible qu'en mettant en jeu des techniques de production automatisées mais flexibles.

Médiatisation d'un Document

Le terme de *médiatisation* traduit la transformation que l'on doit effectuer pour introduire dans le document et transmettre efficacement au moyen des divers médias disponibles, la part du message que l'enseignant transmet usuellement sous forme non écrite. Il s'agit, par exemple des anecdotes qui rendent un cours plus attrayant et de sa façon de mettre en valeur un point important par un simple changement d'intonation, par une mimique ou toute autre technique. L'existence du tutorat, indispensable à la FOAD, ne peut servir de prétexte pour diffuser des documents en ligne médiocres au sens de leur médiatisation. En effet, les documents d'EAD sont la vitrine de l'établissement.

Les techniques informatiques aujourd'hui utilisables sur le Web pour le graphisme, l'interaction, les animations, la typographie, permettent déjà une médiatisation d'excellente qualité. De nombreux exemples de cours remarquablement médiatisés (Saporta 2002) prouvent que la forme « à distance » peut être d'aussi bonne qualité et aussi attractive que la forme « en présentiel ». Hélas, les coûts de production de documents médiatisés « par page » sont prohibitifs tant en ce qui concerne la réalisation effective sous traitée à des professionnels qu'en ce qui concerne le travail de l'auteur. De ce fait, seules quelques expériences, réalisées comme « preuves de faisabilité », ont été poussées à ce niveau, le plus souvent par des bénévoles enthousiastes qui ne comptabilisaient pas leur travail. Une telle approche n'est évidemment pas durable. De plus, elle se prête mal à la reconfiguration souhaitable pour construire des cursus personnalisables.

La production pérenne de gros volumes de documents spécifiquement conçus pour la FOAD, implique non seulement une maîtrise des coûts de production, mais aussi la garantie de maintenabilité et de modularité des documents. Ceci est particulièrement indispensable, aussi bien pour des disciplines à évolution très rapide (génétique, télécoms, interaction homme-machine) que pour la personnalisation des cursus. La conception et la production de documents pour la FOAD est une discipline complexe qui met en jeu conjointement les acquis de l'informatique, de la pédagogie, et des sciences de la communication.

Pratiques Usuelles de Production pour la FOAD

La Production Page par Page. Cette technique donne souvent de remarquables résultats, car elle permet une très

fine médiatisation des contenus. Elle rend possible beaucoup de subtilité dans la mise en page, car celle-ci est traitée au coup par coup et peut prendre en compte le contexte et les subtilités de la leçon. Par contre, elle est extrêmement coûteuse, car, même en utilisant des éditeurs de pages interactifs, elle requiert un travail de production important pour chaque page. De plus, la maintenance de tels documents est souvent très délicate : toute retouche significative dans le contenu d'une page revient presque à sa ré-élaboration. Pire encore, des modifications générales dans la charte graphique ou dans le modèle de navigation du site conduisent à mettre à jour manuellement la totalité des pages si l'on veut assurer la cohérence.

Les Pages Dynamiques. Cette technique consiste à faire produire les pages sur le serveur. Elle repose sur l'emploi de scripts (php, asp, perl, ou autres) qui matérialisent sous forme algorithmique la structure de navigation, de présentation et d'interaction des pages, tandis que le contenu textuel ou graphique est obtenu par des requêtes à une base de donnée. Souvent associée à une très forte structuration des contenus et de la forme, elle est bien adaptée à la production de documents pédagogiques de certaines disciplines scientifiques où les notions à introduire sont clairement individualisables, peuvent être illustrées de façon brève et sont susceptibles de mises en pratique isolées et immédiates. Toutefois, son emploi peut difficilement être généralisé à toutes les matières. Par exemple, les sciences de l'homme, l'économie et le droit, et même certaines disciplines des sciences dures ont souvent besoin de faire appel, au sein d'une même leçon, à un ensemble de notions complexes interdépendantes dont la subtilité ne peut être exprimée sans perte dans un cadre structurel rigide et stéréotypé. Par contre, le caractère dynamique de la production de pages dynamiques rend très faible le coût d'ajout de nouvelles pages ou de maintenance des contenus une fois que le développement des scripts est achevé et tant que l'on ne touche pas à ceux-ci : l'auteur se contente de saisir au moyen de formulaires les éléments à ranger dans la base de données… Alors que la mise à jour des contenus est simple et bon marché, toute modification dans la structure du document cible, ou dans la présentation, reste très coûteuse car elle conduit à modifier les scripts, activité typique de programmation qui induit les coûts et les risques associés.

Malheureusement, cette technique ne peut pas être étendue ou améliorée sensiblement au delà d'un certain niveau de souplesse ou de complexité. En effet, le niveau de compétence requis pour l'élaboration de scripts qui ne se limitent pas à la simple mise en place répétitive d'éléments dans un cadre prédéfini, reste extrêmement élevé. Il requiert des spécialistes en informatique ayant aussi un très bon niveau en science de la communication. L'observation des sites à base de pages dynamiques met nettement en évidence le caractère très répétitif, stéréotypé et monotone de leur organisation et ceci quelle que soit la qualité du graphisme associé. Les quelques rares exemples de génération automatique de bonne qualité sont actuellement des prototypes de laboratoire et requièrent alors un travail très important de description des données et des méta-données qui pilotent la génération.

Une Troisième Voie : la Transformation de Structures. Entre ces deux extrêmes, une troisième voie est possible : celle des approches par transformation de structure. La construction de sites Web de qualité rencontre des difficultés qui ne sont pas propres à l'EAD mais communes à tous les développements de sites Web. Le W3C a donc recommandé une autre approche reposant sur la famille de langages bâtis sur XML et XSL-T (XSLT 1999). Le langage XSL-T sert à spécifier des transformations entre une structure de contenu décrite dans un langage source appartenant à la famille XML et une structure cible, dans un autre langage, en particulier HTML. Ainsi, les modèles (ou xsl-templates) d'éléments structurels du document tels que des définitions, des exemples, des leçons, etc. peuvent être décrits en XSL une fois pour toute. La production effective des pages cibles à partir de leur contenu et de ces modèles d'organisation peut alors être totalement automatisée. L'invocation des modèles se fait en exprimant, dans le texte source XML, la structure abstraite du document, la production effective revient à instancier cette structure conformément aux modèles qu'elle invoque.

L'approche XML/XSL-T présente de nombreux intérêts pour la production de cours (Changtao-Qu 2001). Elle combine la performance économique des pages dynamiques et la souplesse de la composition par page. En étant une approche dirigée par le contenu, elle permet de prendre en compte toute la souplesse et la subtilité qui manque aux techniques de production de pages dynamiques. En permettant la production automatisée du document cible grâce aux modèles, elle se place dans la même fourchette de coût, de maintenabilité et de fiabilité que la production par scripts. Ce type d'approche, suggérée par le W3C, serait idyllique si l'effort requis pour spécifier un contenu en XML n'était, hélas, presque aussi élevé que la production de pages en HTML. La lourdeur syntaxique de XML (dont XHTML n'est qu'une instance) rend malheureusement cette approche quasi impraticable de façon directe par un enseignant. De la même façon, l'expression des xsl-templates est, elle-même, presque aussi délicate que la programmation de scripts, et il n'existe encore que peu de spécialistes de ce langage. L'ensemble de ces deux faits explique la relativement lente, mais inexorable, montée de XML dans le public. A l'opposé, son essor dans le monde industriel est déjà considérable, car l'emploi de XML est quasi systématiquement encapsulé au sein d'applications dédiées qui en allègent l'emploi. Pour pouvoir utiliser efficacement XML pour la production de documents pour l'EAD, nous avons suivi cette stratégie d'encapsulation au sein d'un environnement de travail afin de tirer profit de la puissance de XML, sans en avoir les inconvénients.

L'environnement EadGen

EadGen est un environnement de production de cours que nous avons conçu et mis au point, ainsi que sa méthodologie associée, dans le cadre d'une recherche conjointe CNAM / LIRMM pour les besoins en EAD du CNAM et de l'UOM-LR (Université Ouverte Montpellier Languedoc Roussillon). Il est utilisé comme un service en ligne du SIFOD et est accessible en mode démonstration (Nanard 2002-b), ainsi que sa documentation (Nanard 20 02-a). On trouvera un extrait de cours produit avec cette technique sur (Nanard 2002-c). Dans cette section, après avoir présenté les fondements de EadGen et son organisation, nous montrons son intérêt sur un exemple. La section suivante détaille la conception du langage auteur ouvert, son traitement dans EadGen et l'accès à ce service de production de documents.

Fondements de Eadgen

Un Langage Auteur Ouvert, Basé sur XML. Cet environnement évite le piège de langages auteur figés ou d'applications dédiées. Au contraire, il offre une chaîne de production « ouverte », totalement paramétrable par des spécialistes (auteur, responsable de collection, graphiste…), chacun étant responsable de sa propre expertise et disposant de ses propres outils spécialisés. La chaîne de production va de la production du contenu par l'auteur jusqu'à la mise en ligne des documents. Elle transforme une matière première pédagogique en des documents interactifs, de façon homogène, conformément aux spécifications d'un responsable de collection. Elle s'appuie sur la spécification des besoins de médiatisation utiles dans un ensemble de matières, ces besoins étant négociés entre les auteurs et le responsable d'une collection. Ils sont exprimés au moyen de langages auteur «ouverts », adaptables à la demande, grâce à l'extensibilité de XML. L'environnement EadGen met en jeu un ensemble de pré-traitements dont le but est d'épargner aux auteurs et aux responsables de collections, les difficultés inhérentes à l'emploi direct de XML, tout en leur offrant une très grande souplesse et un pouvoir expressif identique à celui d'une production directe en HTML.

Production de Documents Basée sur la Transformation de Structure avec XSL-T. Produire un document consiste à mettre en relation la structure du contenu de la matière première pédagogique et celle du document à produire. Dans cette structure cible, on peut mettre en évidence plusieurs aspects structurels : l'organisation pédagogique, la présentation, la navigation, l'interaction. Automatiser la production suppose de rendre explicite l'ensemble de ces structures en les modélisant et en spécifiant formellement leurs règles de correspondance. Une collection de cours est un ensemble de cours qui partagent le même modèle de structure cible, indépendamment de tout contenu. La transformation entre la structure de la matière première (exprimant les intentions de l'auteur) et la structure du document cible (correspondant à ce qui est fourni à l'apprenant) est spécifiée dans le langage de spécification de transformation XSL-T.

Architecture et Fonctionnement de EadGen

Le principe général de la chaîne de production consiste à rendre simple la définition d'un langage auteur extensible (au sens de XML) et à fournir les traducteurs associés pour convertir en HTML des documents sources écrits dans ce langage (voir figure 1).

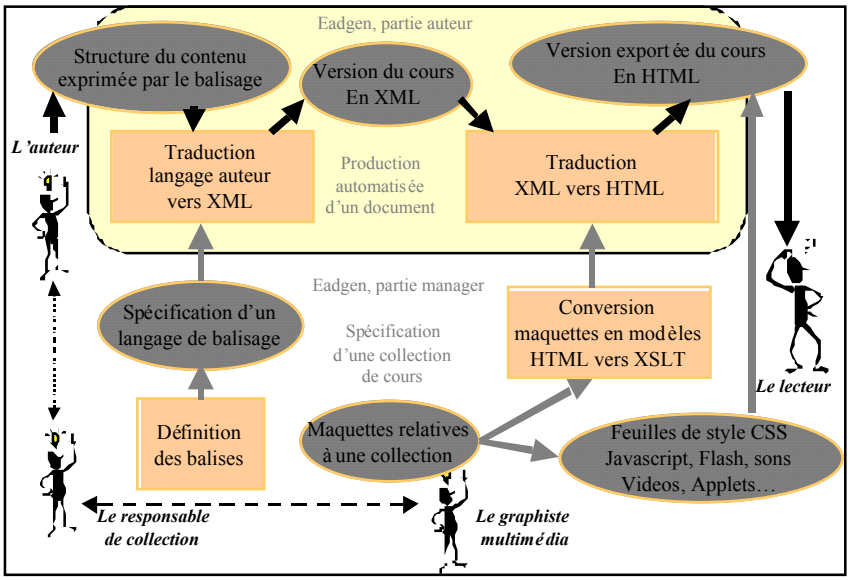


Figure 1. Traitements, rôles et documents sous Eadgen

Lors de la définition d'une nouvelle collection, le responsable de la collection définit, de préférence conjointement avec les auteurs, les concepts nécessaires à l'expression de leurs cours. Il spécifie un nouveau jeu de balises. Pour définir leur effet sous forme de transformations XSL-T, il collabore au préalable avec un graphiste ne maîtrisant pas nécessairement XSL-T : De plus, il peut bénéficier au sein de EadGen d'un outil assurant la tâche de rétro-ingénierie pour passer de maquettes de pages élaborées par le graphiste en HTML à un ensemble de xsl-templates qui seront utilisés pour la traduction XML vers HTML.

Ceci présente les avantages suivants :

- Le concepteur-auteur du cours peut exprimer ses intentions directement dans le contenu textuel des documents sources au moyen d'un balisage léger référant les concepts définis pour la collection.
- Un compilateur transforme automatiquement ce balisage en XML, puis en HTML (ou dans un autre langage).

Un exemple

Une des originalités du langage ouvert de EadGen est la possibilité de définir des éléments liés à la sémantique du domaine, en plus des éléments classiques de structuration logique et de description de méta-données pédagogiques. Supposons par exemple que, dans les exercices interactifs qui seront simulés dans le cours en EAD, un enseignant de chimie souhaite rappeler automatiquement la dangerosité de certains produits. Il décide, avec le responsable de collection, d'introduire dans le vocabulaire du langage, une balise pour matérialiser ce concept, notée d'un commun accord «$tox» avec la sémantique suivante :

> « Lorsqu'un produit a été déclaré *toxique* et que les précautions d'emploi associées ont été indiquées, ceci une fois pour toutes dans le cours, ces dernières sont automatiquement rappelées à l'apprenant dans une bulle d'aide dynamique partout où le nom de ce produit est invoqué dans le document, et ce, sans que l'auteur n'ait quoi que ce soit à ajouter pour déclencher cet effet local. De même, une table de ces produits et de leur emploi dans les exercices sera construite automatiquement».

Une fois le concept introduit dans le langage par le responsable de collection, l'auteur n'aura plus qu'à déclarer, une seule fois quelque part dans le cours (voir figure 2), que le chlore est un produit « toxique », pour qu'à chaque instanciation d'emploi de ce produit dans un quelconque exercice, ou dans le cours, l'ensemble des règles de sécurité associées seront automatiquement rappelées sous forme de bulle d'aide.

```
$tox chlore
```

```
Le chlore est un gaz irritant pour les bronches. Toute
réaction dégageant du chlore doit se faire sous hotte.
```

Un peu plus loin dans le corps d'un exercice l'auteur a écrit :

```
$manip
Verser lentement 10ml de …
on observe un dégagement de chlore…
```

L'auteur n'a rien de plus à dire ici pour déclencher le rappel.

Figure 2. Exemple de légèreté du langage de balisage

La notion de toxicité, exprimée ici une fois pour toute devant le mot chlore par la présence de la balise légère `$tox` concerne l'auteur, elle est un concept associé au contenu. Le choix de la *présentation* et de *l'interaction* associée à l'expression de ce concept sous forme de bulle d'aide dynamique, réalisée par un script dans une page HTML, ne relève plus de l'auteur mais du responsable de collection. La réalisation totalement automatique de ces traitements assure leur fiabilité et la maintenance facile du cours : tout nouveau passage citant un produit déclaré toxique sera à jour. Dans la pratique, le graphiste décide et illustre dans une maquette de page en HTML comment est représenté pour l'apprenant la notion de toxicité d'un produit et quelle sont les interactions et la navigation associées. L'outil de conversion automatique HTML vers XSL-T produit le code XSL-T associé à la balise $tox épargnant ainsi au graphiste une large partie de l'effort de spécification de la traduction en XSL-T.

La séparation entre la spécification du contenu et celle de la forme finale permet, pour un même contenu, balisé une fois pour toutes, de produire des documents spécifiquement adaptés aux disfonctionnements de divers navigateurs ou même une version imprimable. En continuant l'exemple précédent, le responsable de collection peut décider que, dans la version imprimée, toute occurrence de produit toxique sera en rouge, avec une tête de mort dans la marge portant l'indication du numéro de page où se reporter pour lire les consignes de sécurité. Cet exemple montre bien la possibilité d'une médiatisation automatique d'habitudes pédagogiques de l'enseignant et le caractère ouvert du langage.

Conception et Traitement des Langages de Eadgen

Le langage auteur de EadGen est totalement ouvert et extensible au sens de XML. Par cela, on entend que seule sa syntaxe est spécifiée de façon formelle, mais que son vocabulaire et sa sémantique sont totalement libres. Il est donc très facile, une fois la syntaxe simple et générale comprise, de composer le langage de son choix, et de lui donner la sémantique que l'on souhaite, dans la limite des combinaisons des traitements élémentaires disponibles sous EadGen.

Forme Générique du Langage Auteur

Les objectifs et contraintes que nous nous sommes fixés dans la conception de la forme du langage sont les suivants :

- réduire l'effort nécessaire au marquage à son niveau le plus bas, en particulier pour permettre la reprise d'informations provenant d'anciens documents.
- disposer d'un langage très facilement extensible.

Syntaxe. Les balises sont identifiées par la présence d'un symbole de début de balise, usuellement le « $ ». Celui-ci n'est interprété comme tel que s'il apparaît en début de ligne. Cette mise en page augmente la lisibilité du balisage en évitant la recherche de balises disséminées dans les lignes. Syntaxiquement, une balise se compose d'un *nom*, d'un champ optionnel de *méta-données* et d'un *paramètre principal* optionnel et a une *portée* qui définit son contenu (voir figure 3) :

```
$sonNom (ses méta données) son paramètre principal
sa portée ou zone d'influence, pouvant contenir d'autres
balises
```

Figure 3 : Forme syntaxique générale

La figure 4 montre le balisage d'une image commentée :

```
$img (LadyDiana.jpg) La princesse Lady Diana
Lady Diana fut l'épouse du prince de Galles et mourut à Paris
dans un accident de voiture.
```

Figure 4 : Syntaxe d'une balise

Dans cet exemple, la balise $img rassemble les informations associées à une image. La chaîne de caractères qui spécifie le fichier image est utilisée ici en position de méta-donnée, en ce sens qu'elle ne figure pas en tant que telle dans le document cible, mais y agit seulement par son effet. Le titre de cette image est le paramètre principal de cette balise. La *portée* de cette balise est la portion de document source balisé qui s'étend de la ligne qui suit cette balise jusqu'à rencontre d'une autre balise plus prioritaire. La sémantique de cette balise est que cette portion de document constitue le commentaire d'image. La construction syntaxique associée aux balises est systématique, et n'a aucune exception. Elle est facile à retenir même pour un novice et se résume très simplement en une phrase : « *ce qui est entre parenthèses n'apparaît jamais tel quel dans le texte, ce qui est sur la même ligne est ce sur quoi porte la balise, et son effet perdure jusqu'à rencontre d'une balise plus importante qu'elle*». Cette règle s'applique à toutes les balises, quel que soit leur nom et leur sémantique.

Priorités. Pour réduire l'effort de balisage, un mécanisme de *priorité* a été choisi de préférence à une notation parenthésée comme en XML, plus lourde. Il permet de décrire des structures d'arbres de façon très naturelle : une balise de priorité plus importante agit implicitement comme parenthèse fermante de toute balise de priorité moindre qui la précède. Par exemple dans la figure 5, l'auteur déclare un « exemple », puis une « illustration commentée » comme suit :

```
$exemple L'oxygène active une combustion
L'oxygène intervient dans de nombreuses réactions appelées
oxydation, ou combustion en langage courant.
$illustration (Fil_de_fer_dans_oxygène.avi) Combustion d'un
fil métallique en présence d'oxygène
L'extrémité du fil de fer préalablement chauffé au rouge est
introduit dans un ballon contenant de l'oxygène. Il s'y
consume vite en émettant des étincelles constituées de
particules de fer brûlant dans l'oxygène.

$def réaction exothermique
On appelle réaction exothermique…
```

Figure 5 : Balisage structurel implicite par priorités

Cette *illustration* (ici une vidéo) est incluse dans l'*exemple* dans la mesure où la priorité d'une figure est inférieure à celle d'un exemple. La *définition* qui suit provoque implicitement la fin de l'exemple, car une *définition* a une priorité supérieure à celle d'une figure et à un exemple. Par contre, elle ne clôt pas la *section* en cours, car une définition peut apparaître dans une section, ce qui a été exprimé, une fois pour toutes, à la création du langage, en affectant leurs priorités aux balises. L'absence de parenthèsage réduit considérablement l'effort de balisage, non seulement en terme de frappe, mais aussi parce qu'il dispense l'auteur de se préoccuper des balises ouvertes en un point donné du document. Le mécanisme de priorités s'est révélé naturel et simple à comprendre pour les auteurs. Avec cette technique, l'écriture d'un document structuré en arbre semble être d'un coût cognitif quasi identique à celle d'un document structuré en râteau. Ainsi le passage suivant se comprend sans commentaires supplémentaires (voir figure 6) :

```
$qcm Vos connaissances en histoire de France
$q Quelle est la date de la bataille de Marignan ?
$f 1914
$rep non ceci serait la date du début de la seconde guerre
mondiale
$j 1515
$rep en effet, cette bataille a été gagnée par François
Premier
$f 1789
$rep non ceci est la date de la révolution française
$q quelle est la couleur du cheval blanc d'Henri IV ?
$j blanc
```

Figure 6 : Exemple de balisage léger possible pour noter le rôle des éléments d'un QCM

Remarquons que la dissociation des rôles entre le responsable de collection et l'auteur évite à ce dernier, lorsqu'il décrit un QCM, d'avoir à se préoccuper de problèmes de représentation ou d'interaction. Ceci n'est pas le cas habituel de nombreux environnements auteur. Par contre, s'il décidait d'associer des points de notation aux questions, ce qui est de sa responsabilité, il

l'exprimerait en tant que méta-donnée, comme dans la figure 7 :

```
$q(3pt) Quel est le numéro atomique du zinc ?
```

Figure 7 : Exemple d'expression directe de méta-donnée au sein d'une balise

De façon générale, l'expression de méta-données structurées, comme proposé dans LOM (LOM 2001) est très facile et naturelle avec notre technique de balisage léger extensible (voir figure 8).

```
$exercice calculer la surface d'un polygone
$prerequis (calcul de la surface d'un triangle)
$duree (6 minutes)…
```

Figure 8 : Expression de méta-donnée structurées en balisage léger

Langage de Spécification du Vocabulaire et de la Sémantique du Langage Auteur

La liste des balises disponibles et leur sémantique associée est spécifiée par le responsable de la collection. Pour construire le langage, il délivre simplement un prototype de chaque balise, dans lequel chacune des trois zones de la balise, si sa présence est autorisée pour cette classe de balise, est remplie sous forme d'un commentaire lisible par un humain. Une liste d'attributs prédéfinis spécifie quels traitements, parmi ceux disponibles dans EadGen, sont effectués par le pré-processeur transformant ce langage en XML. Ces attributs constituent la notation formelle de la sémantique de la balise. Enfin, le nom du xsl-template associé clôt la définition de la balise. Reprenons l'exemple du concept de produit toxique. La définition du nom et de la sémantique de cette balise est exprimée en XML (voir figure 9).

```
<Balise
   nom = «tox» priorité = «50»
   concerne = «nom du produit concerne»
   porte = «consignes de sécurité associées»
   detecte_occurrence= «bulle»
   cree_bulle_avec= «contenu»
   table_associée = «Liste des Produits Toxiques»
   references_croisée = «oui»
   nom_xml= «prodtox»
/>
```

Figure 9 : Définition d'une balise par le responsable de collection

Dans cet exemple, les attributs indiqués spécifient que le traducteur construira une table de ces produits, et que dans cette table figureront les références croisées des endroits où figure ce terme. Ils spécifient de fait que le pré processeur effectue les traitements nécessaires à la reconnaissance dans le source des mots déclarés par cette balise et la mise en place automatique dans le code XML de la marque qui repère une occurrence de ce mot, et la génération des fichiers associés. Notons qu'une spécification tout à fait similaire peut être utilisée pour donner le nom d'un théorème et son énoncé, puis faire apparaître ou non automatiquement un rappel de l'énoncé chaque fois que ce théorème est invoqué. L'attribut « masque_aides » peut être utilisé en particulier dans la portée de la balise $qcm pour inhiber localement cette aide. Les traitements correspondants ont un caractère générique. Le langage conçu par le responsable de collection est ainsi défini sous forme d'un fichier XML utilisé pour paramétrer le traducteur. Un utilitaire de création de balises, utilisant un formulaire web, permet d'effectuer cette tâche sans devoir connaître XML, ni devoir se souvenir des noms des attributs.

Fonctionnement interne de EadGen

Du Document Balisé par l'Auteur à un Document Source en XML. Un pré-traitement épargne à l'auteur de devoir écrire son document source directement en XML. Pour cela, le pré-processeur de EadGen utilise les fichiers de spécification des balises propres au langage auteur choisi et effectue les traitements spécifiés par les attributs de chacune des définitions de balises lors de la rencontre de leurs instances dans le document balisé. La liste des traitements sémantiques possibles par le pré-processeur n'a besoin d'être connue que par le responsable de collection lorsqu'il définit les balises. L'auteur n'a pas à la connaître. Il utilise les balises, connaît leur effet, mais n'a pas besoin de se soucier de leur implémentation. Le traducteur transforme alors les textes sources en un ensemble de fichiers XML, par nature bien formés au sens de XML, et conformes à la DTD implicitement définie pour ce langage auteur par le fichier des balises. La structure du document XML résultant est le reflet direct de l'arbre implicitement codé dans le fichier source par les priorités des balises. Pour chaque balise source, un nœud constitué de plusieurs balises XML est construit dans le document cible XML conformément à la syntaxe et aux attributs de la balise source. Dans l'exemple précédent, la rencontre d'une balise $tox génère la sous-structure XML donnée dans la figure 10.

```
<prodtox >
<item_prodtox> nom du produit concerné </item_prodtox>
<content_prodtox>
consignes de sécurité associées
</content_prodtox>
</prodtox >
```

Figure 10 : Déclaration de produit toxique traduit en XML

Traduction de XML vers HTML. Le pré-processeur délivre un document XML. La production d'une représentation de ce document se fait en lui associant une feuille de style XSL. Rappelons que XSL-T est un langage de spécification de transformations, bâti sur XML. Il définit des xsl-templates qui indiquent comment les sous-arbres du document source XML se transforment en sous- arbres du document cible. Tout processeur XSL-T du commerce, par exemple Saxon (SAXON) ou Xalan

(XALAN), peut être utilisé pour la production du code final.

Le responsable de collection définit conjointement avec un graphiste la présentation qu'il souhaite associer à chaque balise qu'il a défini dans le langage auteur. Ceci est accompli en définissant les xsl-templates correspondants. En pratique, l'écriture de xsl-templates est peu attractive et plutôt lourde. Pour épargner cette tâche au responsable de collection, il a été développé un second pré-processeur dont le rôle est de transformer en xsl-templates les maquettes que le graphiste fournit en HTML. Pour cela, le graphiste indique dans sa maquette de page, sous forme de commentaire HTML, que telle partie du code HTML sert de modèle pour telle balise. Au sein de cette partie, il identifie les éléments variables par la notation conventionnelle `??source??` et les paramètres d'un xsl-template par le nom qu'ils ont en XSL. Cette convention est assez simple pour pouvoir être utilisée sans effort notable. Pour assurer une meilleure observabilité de la maquette sans nuire à sa traduction automatique, la chaîne de caractères du logo du Cnam en latin « *omnes docet* » peut être utilisée librement comme simulateur de gris dans la maquette, et n'est pas retranscrite dans le document cible.

Utiliser Eadgen

L'environnement que nous venons de décrire est en service en interne au CNAM depuis février 2001 (voir figure 11). Il a été mis à disposition du réseau CNAM sous forme d'un service accessible sur le Web. En pratique, plusieurs ensembles de balises et les transformations XSL associés sont déjà disponibles. D'autres sont en cours de conception dans le cadre de séries spécifiques.

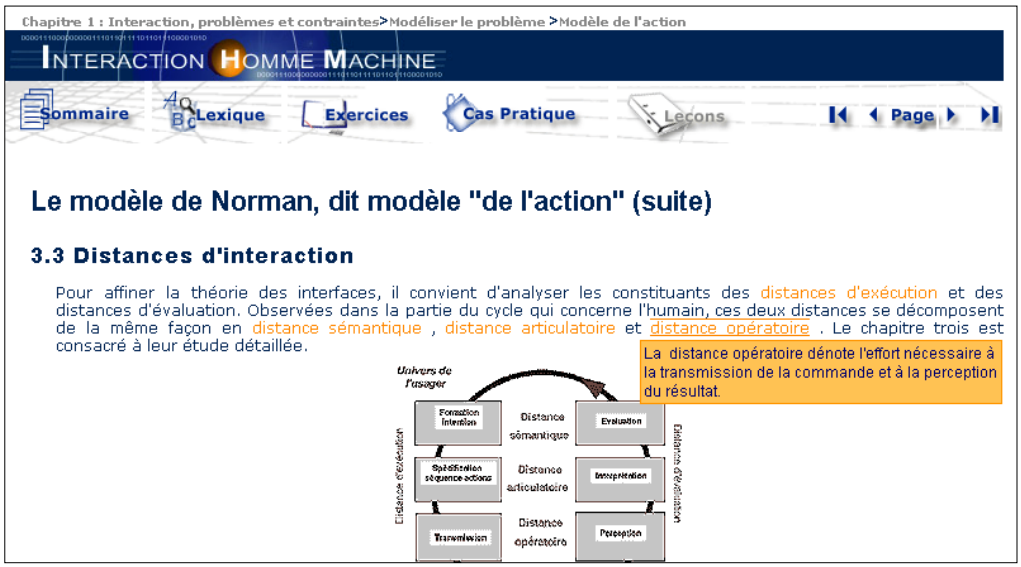


Figure 11 : Un exemple de page générée par Eadge.n

Les usagers de EadGen disposent d'un espace de travail privé, personnalisable et protégé où ils placent leurs sources et récupèrent leurs traductions. Aujourd'hui, un auteur peut préparer un document source balisé selon le principe énoncé dans les sections précédentes. Dans la majorité des cas, ceci peut se faire directement dans un document Word, mais un simple éditeur de texte suffit. Le ou les documents à traduire sont téléchargés au format texte sur le site de EadGen et y sont associés à un des langages disponibles selon les préférences de l'auteur. L'auteur peut aussi sélectionner les fichiers à traduire pour construire des parcours pédagogiques à la carte. Une fois la traduction effectuée, l'ensemble du site produit est restitué à son auteur sous forme d'un fichier compacté, contenant l'ensemble des pages HTML, les scripts, les constituants de la charte graphique, etc. que l'auteur déploie sur son propre site. Le temps de production sur le serveur de EadGen est de l'ordre de quelques minutes pour générer un site de quelques centaines de pages HTML.

Du fait de la rapidité de la traduction, une approche incrémentale de l'élaboration des documents est possible, l'auteur pouvant, dans un délai très bref, effectuer un cycle d'écriture ou de correction de ses documents sources et observer l'effet sur le document produit. Il est bien évident que toutes les mises à jour doivent être effectuées dans le document source et non dans le document cible. Par suite l'activité de maintenance d'un cours n'est plus qu'un cas particulier de sa production.

Evaluation et Discussion

Les apports de XML et XSLT à la pédagogie sont nombreux (de La Passardière 2001). L'un des points

majeurs est de séparer les spécifications concernant le contenu de celles concernant la forme. Ceci facilite une description fortement structurée du document source rendant possible l'identification du rôle pédagogique ou sémantique de chaque élément, et par suite leur réemploi dans le cadre de parcours pédagogiques personnalisables (Hiddink 2001). Cette description ne préjuge en aucune manière de l'usage qui est fait de l'expression de la structure.

Symétriquement, la traduction effective vers la forme utilisée par l'apprenant, spécifiée en XSL-T, est donnée sous forme de règles (xsl-templates) et a un caractère générique : toutes les pages cibles sont produites automatiquement de façon homogène par rapport à ces règles. Le contenu des templates, exprimé en HTML, a par conséquent le même pouvoir expressif qu'une écriture directe en HTML, donnant ainsi accès sans restrictions aux scripts, aux animations, aux applets, ou à l'appel de CGI.

Le langage de balisage léger utilisé dans Eadgen allège considérablement l'activité de codage de l'auteur, en lui épargnant l'aspect fastidieux du codage XML, sans pour autant l'enfermer dans le cadre rigide des formulaires de saisie déduits automatiquement d'une DTD, ou des éditeurs dirigés par la syntaxe. En conservant une forme de balisage léger, utilisable « au fil de la plume », on préserve la créativité et l'initiative de l'auteur en phase de rédaction, sans pour autant perdre l'intérêt et la richesse expressive d'une description structurée. Un autre aspect important de EadGen n'est pas l'objet principal de cet article : la spécification de la navigation. Les mécanismes mis en jeu dans le pré-processeur et dans les transformations XSL-T permettent de spécifier de façon abstraite et générique le modèle de navigation et dispensent l'auteur de devoir expliciter des liens. Ce dernier peut se concentrer sur la description intrinsèque des propriétés de son contenu pédagogique, plutôt que sur la navigation dans une matérialisation particulière de celui-ci.

Le caractère « ouvert » du langage de Eadgen rend possible son adaptation très facile aux spécificités de chaque type de discipline et même aux extensions souhaitables pour organiser facilement et économiquement des expériences pédagogiques. Il est bien adapté à l'expression des méta-données caractérisant des éléments de cours, comme ceux suggérées par LOM (LOM 2001). Le mécanisme de traduction par des feuilles de style XSLT est avant tout une transformation de structure. Produire du code HTML n'est qu'un cas très particulier pour XSLT. Bien que cela soit beaucoup moins connu, il est tout aussi facile de construire des spécifications XSLT pour faire générer du code d'accès à une base de donnée, et ainsi stoker ou extraire de celle-ci des éléments de base en fonction des méta-data ou de propriétés structurelles décrites dans le langage de balisage léger.

Remerciements

L'implémentation associée au présent travail a été financée par le SIFOD. Nous exprimons notre gratitude à l'équipe qui a collaboré au développement de Eadgen : Karine Bianciotto, Séverine Trabbia et Nathalie Astruc.

Références

Changtao-Qu, 1999. Collaborative courseware authoring and publishing based on WebDAV, XML and XSLT. In Proc. Int. Conf. on Trends in Communications, Eurocom 01, IEEE, Piscttaway NJ, USA.

de La Passardière, B., Giroire, H. XML au service des applications pédagogiques. In Proc. EIAO 2001.

Hiddink, G. 2001. Solving reusability problems of online learning matériels. *Campus Wide Information System*s, 18(4): 146-52.

LOM 2001. Draft Standard for Learning Object Metadata. IEEE P1484.12/D6.1. 18.

Michard, A. 1999. *XML, langage et applications*. Paris: Eyrolles.

Murray, T. 1999. Authoring Intelligent Tutoring Systems : an Analysis of the State of the Art. *IJAIE* 10: 98-129.

Nanard, M. 2002-a. Documentation en ligne de Eadgen V1.4 . http://www.lirmm.fr/~mnanard/EadCnamTools/ DocsEadgen/ie/intro_docead.html

Nanard, M. 2002-b. Environnement de production de cours pour l'Ead. http://www.lirmm.fr/~mnanard/EadCnamTools/OpenSession.hml

Nanard, M. 2002-c. Cours en ligne d'interaction homme-machine. http://www.lirmm.fr/~mnanard/ihm/ie/chap1/ cas-prat_c1_p4.html. (Le cours complet n'est accessible qu'aux apprenants enregistrés sur la plateforme Plei@d du CNAM).

Saporta, G. 2002. Techniques de la statistique. Module Cnam en Ead, SIFOD ref. 01435, CNAM, AGRO & UMII.

SAXON. The Saxon XSLT engine. http://users.iclway.co.uk/mhkay/saxon/

XALAN Apache. The Xalan XSLT engine. http://xmlfr.org/actualites/tech/010206-0003 http://xml.apache.org/xalan-j/ (ref)

Xml 1998. XML, The eXtendible Markup Language, http://www.w3.org/TR/1998/REC-xml-19980210 (ref) http://babel.alis.com/web_ml/xml/REC-xml.fr.html (version française).

XSLT 1999. The eXtendible Markup Language , http://www.w3.org/TR/1999/REC-xslt19991116 (ref) http://xmlfr.org/w3c/TR/xslt/ (version française)