Pourquoi il n'est pas possible actuellement d'utiliser des objets pédagogiques plastiques respectant la norme SCORM

Antoine Bouyer *José Rouillard* *Alain Derycke*

Laboratoire Trigone
CUEEP – Bat B6
USTL de Lille
59655 Villeneuve d'Ascq Cedex

antoine.bouyer@gmail.com jose.rouillard@univ-lille1.fr alain.derycke@univ-lille1.fr

RESUME
Dans un contexte de multimodalité et d'interfaces abstraites, cet article fait émerger des verrous liés à la dynamicité des interfaces, à travers une étude de cas. Celle-ci se situe dans le domaine de l'enseignement à distance, qui est très orienté vers l'Internet classique. Nous montrons pourquoi, aujourd'hui, il n'est pas possible de déployer des interfaces concrètes, à la fois basées sur un modèle générique, et utilisables sur des plateformes d'enseignement à distance respectant la norme SCORM.

MOTS CLES : Plasticité, E-learning, objets pédagogiques, norme SCORM.

ABSTRACT
In a context of multimodality and abstract interfaces, this paper shows ploblems related to the dynamicity of interfaces, through a case study. This one is applied to the distance learning field, which is very directed towards the traditional Internet. We show why, today, it is not possible to deploy concrete interfaces, at the same time based on a generic model, and usable on learning management systems respecting SCORM.

CATEGORIES AND SUBJECT DESCRIPTORS: H.5 [Information Interfaces and presentation]

GENERAL TERMS: Design, Experimentation

KEYWORDS: Plasticity, E-learning, pedagogical objects, SCORM.

INTRODUCTION

Le laboratoire Trigone travaille depuis quelques années avec les interfaces adaptatives. Celles-ci permettent de concevoir une application indépendamment du support sur lequel elle va être éxécutée (ordinateur de bureau, PDA, téléphone, ...). Néanmoins, pour être utilisables, ces applications nécessitent d'être dynamiques, plutôt que constituées de « pages » statiques.

Travaillant aussi dans le domaine des EIAH (Environnements Informatiques pour l'Apprentissage Humain), notre équipe s'est intéressée à l'étude de la plasticité des objets pédagogiques (OPs). La transformation de ces éléments – qui sont la base de l'apprentissage à distance – afin de pouvoir les utiliser dans un contexte de mobilité et de multimodalité nous à permis de découvrir certains verrous associés à la dynamicité des interfaces abstraites. Cela rend cette étude de cas intéressante de par les problèmes généraux qu'elle soulève, plutôt que par les résultats directs obtenus. Ainsi, cet article reviendra d'abord sur la définition et l'utilisation des OPs, puis sur les possibilités existantes pour les rendre plastiques ainsi que les problèmes scientifiques et techniques rencontrés. Enfin, nous présenterons les verrous plus généralement liés au domaine.

QU'EST-CE QU'UN OBJET PÉDAGOGIQUE ?

Avec le développement d'Internet, l'intérêt pour l'enseignement à distance a considérablement augmenté. Dès lors, des plateformes d'e-formation (dites LMS pour *Learning Management System*) [4], [6] ont été créées et distribuées, librement ou non. Pour chaque plateforme, l'enseignant peut créer des modules (leçon de cours, QCM ou exercices à trous, par exemple), qui seront ensuite assemblés pour réaliser un cours complet. Comme il n'existait pas de standard structurant ces modules, chaque plateforme était indépendante et possédait son propre protocole de communication.

Lorsque l'on souhaitait changer de plateforme, tous les modules existants devaient donc être re-codés.

Pour remédier à ce problème, et permettre la mise en commun de matériel pédagogique, la norme SCORM (Sharable Content Object Reference Model) [14] a été proposée par ADL (http://www.adlnet.org/), à l'initiative du département de la défense Américaine (*Departement of Defense*). Son objectif est de permettre aux systèmes d'apprentissage en ligne d'importer, de partager, de réutiliser, et d'exporter des contenus d'apprentissage, de manière normalisée. Pour ce faire, SCORM regroupe un ensemble de spécifications déjà existantes ([1], [2], [8]), définissant ainsi l'agrégation d'objets pédagogiques (SCO), la navigation de l'un à l'autre, ainsi que la communication entre ceux-ci et les plateformes d'apprentissage à distance *Scorm-compliant*.

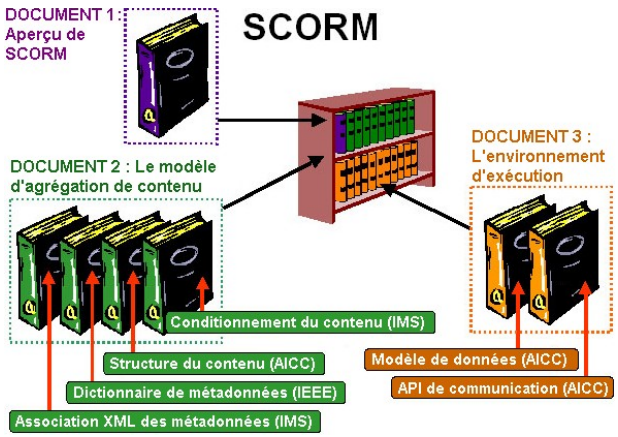

Figure 1 : Vue synthétique de la norme SCORM

Concrètement, SCORM peut être découpé en plusieurs parties différentes (cf. figure 1). Tout d'abord, le « modèle d'agrégation du contenu » assure la promotion de méthodes cohérentes en matière de stockage, d'identification, de conditionnement d'échange et de repérage du contenu. Il définit des méta-données décrivant le contenu (spécification LOM : http://ieeeltsc.org/), ainsi que l'empaquetage de l'ensemble des fichiers et documents joints. Ensuite, l'« environnement d'exécution » décrit les exigences nécessaires du système d'apprentissage quant à la gestion de l'environnement d'exécution. Il s'agit d'une API Javascript permettant une communication normalisée dans les deux sens, et indépendante des outils utilisés. Elle permet principalement à l'OP d'utiliser ou d'enregistrer des variables sur la plateforme, à l'aide de quelques fonctions Javascript. Enfin, le « modèle de séquencement et de navigation » permet une présentation dynamique du contenu. Il décrit comment le système interprète les règles de séquencement exprimées par un développeur de contenu, ainsi que les événements de navigation déclenchés par l'apprenant ou par le système.

Ces plateformes et la norme SCORM étant pensées pour produire un résultat graphique, cela pose des problèmes d'IHM. Par exemple, il peut apparaître un effet mosaïque [15], [9], c'est-à-dire qu'il n'y a pas d'unité graphique entre la plateforme et les différents cours visualisés pourtant en même temps sur l'écran. De plus, dans le cas d'étudiants déficients (visuels, par exemple), aucun outil ou méthode ne permet de transformer ou de réadapter l'interface. Par la suite, ce sont les transformations des interfaces vers d'autres canaux qui vont être étudiées.

POURQUOI VOULOIR OBTENIR DES OBJETS PÉDAGOGIQUES PLASTIQUES ET MULTICANAUX ?

Avec l'essor de l'informatique ubiquitaire (ordinateurs portables, PDA, téléphones portables, ...), les utilisateurs doivent pouvoir utiliser leurs applications sur les différents modes de communications qui leur sont proposés à un instant donné (Internet, wap, téléphonie, ...). Cependant, pour développer une application de N pages destinée à M périphériques, il faut en principe coder N*M pages. Pour éviter ces redondances, le laboratoire Trigone a défini un langage de description d'IHM abstraites : PlasticML [12]. Le principe, comme pour d'autre langage ressemblant à USIXML [17], est de concevoir l'interface de façon unique, indépendamment de son contexte d'utilisation [16], [19]. Il n'y a alors plus que N pages à développer. Ensuite, une transformation XSL permet de traduire celles-ci vers le format voulu. Dans le cas de PlasticML, il s'agit de HTML [7], WML (wap) [20] et VoiceXML (téléphonie) [18], [13].
Etant donné le potentiel de l'informatique ubiquitaire pour le futur de l'e-Formation, en particulier le *mobile-learning* [5], nous nous intéressons aux OPs présentant une IHM « plastique ». De plus, cela permet d'envisager à terme des interfaces multimodales pour les OPs, et pas seulement sur d'autres périphériques graphiques [10] (PDA, Smartphone, ...).

Néanmoins, cette création a rapidement été limitée par la norme SCORM elle-même. D'une part, l'API de communication se fait grâce à des fonctions Javascript. Or les différents langages utilisés (HTML, WML et VoiceXML) possèdent chacun leur propre langage de script (respectivement Javascript, WMLScript et ECMA Script.) Nous ne pouvons donc pas créer *a priori* d'OPs SCORM accessibles vocalement ou via le WAP. D'autre part, étant donné que SCORM est orienté vers l'Internet « classique », les systèmes d'apprentissages compatibles SCORM ([4], [6]) ne proposent pas de portail Wap ou vocal. A l'inverse OpenUSS [11] permet une navigation sur le wap, mais n'est pas *Scorm-compliant*.

COMMENT RENDRE UN OBJET PÉDAGOGIQUE PLASTIQUE ? MÉTHODES ET PROBLÈMES ASSOCIÉS.

Plusieurs possibilités d'utilisation d'OPs plastiques ont été identifiées. Elles sont décrites ci-dessous, et les difficultés liées à chacune d'elles sont analysées.

La Rétro- engineering

Tout d'abord, la première transformation qui peut venir à l'esprit consisterait à utiliser un phénomène de rétro-engineering. Il faut dans ce cas faire transiter le flux HTML, envoyé par le système pour l'apprenant, à travers

un proxy. Celui-ci peut alors transformer en temps réel le code HTML vers un langage abstrait, qui sera ensuite retransformé vers le langage supporté par l'utilisateur [3].

Aussi, cette méthode a été écartée de nos objectifs dès le départ, notamment à cause des problèmes qu'elle engendre. La première difficulté apparaît à travers la richesse du code HTML et du CSS, par rapport aux autres langages. Ainsi, une page Internet peut devenir rapidement très complexe (cadres, tableaux imbriqués, largeur des éléments imposés, ...). Cela peut freiner, voire empêcher sa transposition vers des interfaces pour d'autres périphériques (petits écrans, interfaces vocales, etc.). Un autre verrou réside dans le fait que certaines pages Web ne sont absolument pas transformables. En effet, les animations flash ou les applets java, par exemple, ne pourront pas être abstraits, ni ré-exécutés telles quelles.

Utilisation D'objets Pédagogiques Plastiques Sur Des Plateformes « Classiques »

Rappelons que notre objectif est de créer des OPs plastiques, dans le but de les transformer par la suite vers un format cible voulu. Pour cela, nous utilisons le langage d'interfaces abstraites PlasticML, conçu par le laboratoire Trigone [12].

Dans un premier temps, le langage PlasticML a été étendu afin qu'il supporte l'utilisation d'événements et de scripts. Dans un souci de simplicité, il a été décidé de ne pas abstraire le langage de script. Les scripts sont ainsi stockés dans autant de fichiers externes que de langages cibles visés. Le nom des fichiers est le même pour tous les langages, seule l'extension change. De la même façon, les noms des fonctions définis dans chaque fichier et utilisés par les évènements sont les mêmes, et seule l'implémentation diffère (voir Code 1). Cela permet de ne pas s'occuper de l'implémentation des scripts dans le code de l'interface abstraite. Après réflexion, nous avons retenu cette solution, car les fonctions définies par les scripts, dans le cadre de SCORM, sont peu nombreuses, et une fois les fichiers pour chaque script créés, ils peuvent être réutilisés d'un objet à l'autre.

Code PlasticML :

```
<plasticml version="2.0">
  <interface title="SCORM" load="open()"
             unload="close()">
    <script source="sco"/>
    ...
    <userevent name="Valid" action="check()"/>
    ...
  </interface>
</plasticml>
```

Code HTML après la transformation :

```
<html>
  <head>
    <title>SCORM</title>
    <script language="JavaScript" src="sco.js">
    </script>
  </head>
  <body onload="javascript:open();"
        onunload="javascript:close();">
    ...
    <input type="button" value="Valid"
           onClick="javascript:check();" />
    ...
  </body>
</html>
```

Code 1 : Les scripts et les évènements en PlasticML

Plus globalement, les modifications apportées à PlasticML autorisent plus de dynamicité, à travers l'utilisation de scripts prédéfinis pour un canal donné. Cette possibilité de gérer des scripts, lors de l'exécution des pages Web, permet de réaliser des tâches en fonction des évènements rencontrés. Ceux-ci peuvent être déclenchés soit par l'ouverture ou la fermeture d'une page, soit par l'action d'un utilisateur (qui clique sur un bouton par exemple). Dans notre étude de cas, cet évènement peut, par exemple, lancer une fonction qui initialise la communication entre l'OP situé chez l'apprenant et la plateforme sur le serveur, ou bien qui corrige un QCM proposé sur la page.

Un ensemble de pages PlasticML a été créé. Elles ont été regroupées dans un fichier zip, avec le fichier php permettant d'effectuer la transformation et un fichier xml décrivant la structure de ce *package*. Puisque ce paquet respectait la norme SCORM, il a pu être déployé sans problèmes. Ensuite, les pages possédaient des fonctions de scripts similaires à celles de l'API SCORM, si bien que la navigation des différentes pages jusqu'à un QCM de compréhension, s'est déroulée sans encombre sur un navigateur " classique ".

Un souci de compatibilité survient au moment d'utiliser toutes les possibilités de PlasticML au travers d'un système d'apprentissage et de prendre connaissance du cours sur du wap ou au travers d'un téléphone normal. Deux solutions sont envisageables. La première consiste à accéder à cet objet directement et sans passer par un système d'apprentissage (comme la plateforme Claroline, développée en PHP, par exemple). La transformation interne à PlasticML fonctionne alors parfaitement, mais la puissance de la plateforme n'est pas utilisée. En particulier, il n'y a plus d'identification, ni de persistance des données (résultats aux QCM ou temps passé par un étudiant sur un cours). La seconde consiste à utiliser une plateforme qui apporte ces fonctionnalités ; mais celle-ci n'affiche pas seulement l'OP. En effet, l'écran est divisé en cadres (frame HTML), et l'OP n'est affiché que dans l'un d'eux, les autres permettant d'afficher la bannière du système ou le sommaire des chapitres du cours parcouru. Ainsi, même si cet objet peut être transformé vers du WML ou du VoiceXML, le cadre principal reste toujours en HTML.

Pour récapituler, le sujet reste ouvert, et le verrou technique présenté dans cet article demeure, puisqu'à notre connaissance, personne n'a montré qu'il était possible d'utiliser une plateforme standard sur des

canaux autres que ceux de l'Internet « classique ». La norme SCORM et donc les plateformes existantes sont tournées vers ce canal de manière relativement contraignante, notamment à cause de l'utilisation d'une API Javascript.

CONCLUSION

Tout au long de cet article, notre approche visant à permettre l'exécution d'un objet pédagogique sur un canal différent de l'Internet classique a été expliquée. Nous avons dû pour cela faire évoluer le langage PlasticML afin qu'il puisse supporter plus de dynamicité, grâce à la prise en compte de scripts.

La méthode présentée ici n'a pas aboutie. En effet, SCORM a été pensée pour être exécuté seulement via un navigateur Web. Ainsi une utilisation graphique, quel que soit le périphérique sous-jacent (PC, PDA, Smartphone, ...) fonctionne. En revanche, l'intégration de techniques, comme les frames HTML et le Javascript, l'empêche d'être utilisée sur d'autres modes de navigation, comme le vocal par exemple. Voilà pourquoi, selon nous, le format actuel de cette norme ne permet pas l'usage d'objets plastiques sur les LMS. De plus, la version SCORM 2004, qui commence à être implémentée par les éditeurs de plateformes, propose des améliorations d'ordre pédagogique. Néanmoins, aucune évolution des aspects interfaces n'est annoncée (dynamicité, multicanalité, plasticité…).

Plus globalement, cette étude de cas, nous a permis de faire émerger un verrou lié aux scripts dans la conception d'une application abstraite dynamique. En effet, chaque langage résultant d'une transformation possède son propre langage de script, qui possède lui-même ses propres caractéristiques techniques. Ainsi, il est difficile pour l'instant de définir à l'avance le comportement d'une application, disjointement du support d'exécution.

BIBLIOGRAPHIE

1. Avation Industry CBT Committee. (http://www.aicc.org/ *)
2. Ariadne (http://www.ariadne-eu.org/ *)
3. Bouillon, L. and Vanderdonckt, J. *Retargeting Web Pages to other Computing Platforms with VAQUITA*. Proceedings of the 9th Working Conference on Reverse Engineering WCRE'02 (October 29 - November 01, 2002, Richmond), IEEE, 2002, pp. 339 – 348.
4. Claroline. (http://www.claroline.net/ *)
5. Derycke, A., Chevrin, V. et Rouillard, J. *Intermédiations Multicanales et Multimodales pour l'E-Formation : l'Architecture du Projet Ubi-Learn.* Conférence EIAH'2005, ATIEF (25-27 Mai 2005, Montpellier), INRP (eds), 2005, 6p.
6. Ganesha. (http://www.anemalab.org/ganesha/ *)
7. HTML (HyperText Markup Language) : langage support d'Internet. (http://www.w3.org/MarkUp/ *)
8. IMS Global Learning Consortium, Inc. (http://www.imsglobal.org/ *)
9. Ip, A., Radford, A. and Canale, R. *Overcoming the presentation Mosaic Effect of Multi-Use Sharable Content Objects*. Proceedings of the 20th Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education ASCILITE"03 (7-10th Décembre 2003, Adelaide, Australie), ASCILITE, 2003, pp. 256 – 262.
10. Lin, N. H., Shih, T. K., Hsu, H., Chang, H., Chang, H., Ko, W. C., and Lin, L. J. 2004. *Pocket SCORM.* Proceedings of the 24th international Conference on Distributed Computing Systems Workshops - W7: EC (Icdcsw'04) - Volume 7 (March 23 - 24, 2004). ICDCSW. IEEE Computer Society, Washington, DC, 274-279.
11. OpenUSS. (http://openuss.sourceforge.net/ *)
12. Rouillard, J. *Plastic ML and its toolkit*. Proceeding of the 10th International Conference on Human - Computer Interaction HCI'03 (22-27th June 2003, Heraklion, Crete, Greece), 2003.
13. Rouillard, J. *VoiceXML - Le langage d'accès à Internet par téléphone*, Editions Vuibert, 2004, ISBN : 2-7117-4826-x, 197 pages.
14. Sharable Content Object Reference Model. (http://www.adlnet.org/scorm/ *)
15. St-Pierre, R. and Hope, P. *Dynamic Appearance Model Analysis and Alternatives*. Canadian Department of National Defence, Publié par Sun Microsystems Inc, 2003.
16. Thévenin, D. *Adaptation en Interaction Homme-Machine : Le cas de la Plasticité*. Thèse en informatique, Université Joseph Fourier, Grenoble, 21 Décembre 2001. (http://iihm.imag.fr/thevenin/these/TheseDavidThevenin.pdf *)
17. USIXML (USer Interface eXtensible Markup Language). (http://www.usixml.org/ *)
18. VoiceXML : langage permettant d'accéder à Internet via un téléphone. (http://www.voicexml.org/ *)
19. Vanderdonckt, J., Limbourg, Q., Michotte, B., Bouillon, L., Trevisan, D., Florins, M. *UsiXML: a User Interface Description Language for Specifying Multimodal User Interfaces*. Proc. of W3C Workshop on Multimodal Interaction WMI'2004 (Sophia Antipolis, 19-20 July 2004)
20. WML : langage support du wap. (http://www.wapforum.org/ *)

** Tous les liens Internet ont été visités le 20 Février 2006*