Rédacteurs	Nour EL MAWAS
Relecteurs	
Date	19/02/14
Référence	GRAPHIT-D4.3
Version	0.1

Spécification et architecture des APIs des plateformes

D4-3

■ Historique du document

Version	Date	Auteurs	Modifications
0.1	19/02/14	Nour EL MAWAS	
0.2			
0.3			
0.4			
0.5			

Table des matières

Index des Figures

1 Objectifs de ce document

Ce livrable sur les APIs a pour objectif de présenter les modules de communications (import/export) intégrés au sein de l'espace de cours des plateformes afin de simplifier l'opérationnalisation des scénarios pédagogiques sur les pfs. Ces modules se chargent de l'import des scénarios spécifiés en dehors de plateformes et de l'export des scénarios existants.

Les APIs sont développés pour les plateformes de formation suivantes :

- Moodle (2.0 et 2.4)
- Ganesha (version 4)

Ce travail devrait permettre de mettre en évidence une des hypothèses du projet : le langage de conception pédagogique d'une plateforme peut être identifié, formalisé et utilisé comme base pour le développement d'outils de conception pédagogiques externes et des modules d'import/export.

2 Développement du service d'opérationnalisation pour Moodle 2.0

2.1 Présentation de la fonctionnalité backup/restore

La plateforme Moodle dispose de certaines fonctionnalités pour sauvegarder, restaurer et importer des cours. La fonctionnalité backup de Moodle (figure 1) permet aux enseignants-concepteurs de sauvegarder leurs cours dans un archive Zip spécifique (figure 2) qui contient plusieurs fichiers xml recueillant toutes les informations relatives au cours : pages web, forum et leurs messages, devoirs, etc. Ces fichiers représentent les données extraites de la base de données. Nous y retrouvons la description des éléments composant le cours. Nous y retrouvons également des détails techniques de bas niveau tels que les informations des blocs situés dans l'espace du cours, les informations dédiées aux éléments de définition d'un cours (timecreated, legacyfiles, maxbytes, tags, requested, etc.), les informations des utilisateurs et des rôles (*shortname, nameincourse, description, sortorder, archetype,* etc.)

La fonctionnalité *restore* de Moodle permet de restaurer un cours en cas de sinistre ou de le placer sur un autre site Moodle via son service de restauration. Plusieurs modes de restauration sont possibles tels que restaurer le cours actuel en supprimant le précédent, restaurer le cours actuel en y ajoutant des données, créer un nouveau cours, etc. Toutefois, les fonctionnalités *backup* et *restore* ne sont pas toujours accessibles aux enseignants-concepteurs ; leur apparition dans l'espace de cours dépend de la configuration de la plateforme.

La modification d'un concept ou d'un fichier enregistré dans ce *backup* n'est pas une tâche aisée. D'une part, elle nécessite la modification de tous les fichiers XML décrivant ce concept. D'autre part, la fonctionnalité *restore* requiert le même format du paquetage de sauvegarde. Il n'est pas facile de produire de tels paquetages par des outils externes. Toutes ces difficultés ont représenté les raisons majeures pour proposer un nouveau module, intégré à l'espace cours sur Moodle et proposant l'opérationnalisation et l'export des situations d'apprentissage dans un format simplifié.

Illustration 1: Les fonctionnalités de sauvegarde et de restauration de Moodle

activities			Folder
course			Folder
files			Folder
sections			Folder
completion.xml	79	63	XML Document
files.xml	1 280	430	XML Document
gradebook.xml	1 737	542	XML Document
groups.xml	86	64	XML Document
moodle_backup.log	0	0	Text Document
moodle_backup.xml	5 363	742	XML Document
outcomes.xml	83	65	XML Document
questions.xml	83	65	XML Document
roles.xml	693	289	XML Document
scales.xml	79	63	XML Document
users.xml	3 145	844	XML Document

Illustration 2: Exemple d'archive Zip de la sauvegarde d'un cours

2.2 Description du module de communication

Afin que des enseignants-concepteurs puissent importer et exporter des scénarios pédagogiques avec la plateforme Moodle, il est nécessaire de développer un module de communication et de l'intégrer dans l'espace de conception des cours de la plateforme [Abedmouleh et al. 2011b] [Abedmouleh et al. 2011c] [Abedmouleh 2010]. La figure 3 illustre ce

module qui doit jouer le rôle de passerelle de communication entre les outils externes et la plateforme et garantir l'opérationnalisation des scénarios pédagogiques.

Le nouveau module est développé sous le format d'un bloc. Nous avons repris le langage de programmation utilisé pour le développement de la plateforme Moodle. Il s'agit du langage de programmation web le PHP. Nous avons également utilisé le langage XSLT (eXtensible Stylesheet Language Transformations) pour la définition de certaines instructions du processus d'import et d'export. Nous présenterons ci-après les procédures développées et basées sur ce langage.

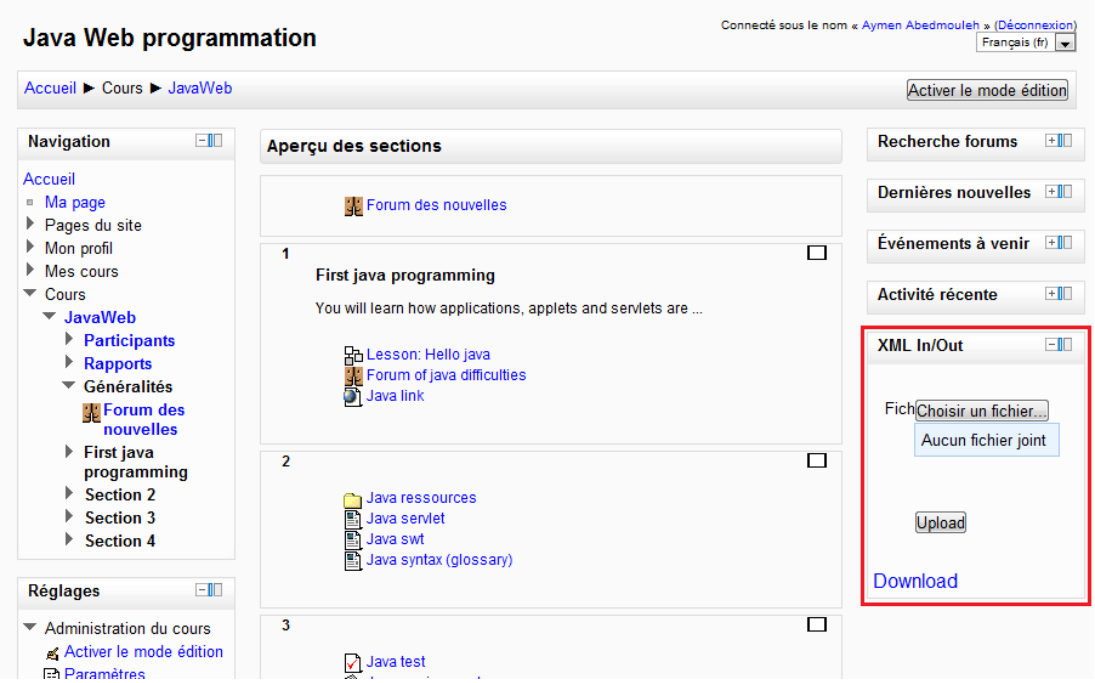

Illustration 3: Service d'opérationnalisation et d'export intégré à l'espace cours de Moodle

Pour définir le block d'import/export des cours dans Moodle, il est nécessaire de définir trois fichiers PHP principaux. D'autres fichiers peuvent être définis puis utilisés par les fichiers principaux. Tous les fichiers doivent être mis sous un répertoire bien spécifique *blocks/importandexportofcourse*. Les trois fichiers principaux sont les suivants :

block_importandexportofcourse.php : ce fichier présente la définition de classe pour le bloc. Il est utilisé pour gérer le bloc comme un plugin et pour le rendre à l'écran. Nous avons tout d'abord créé le fichier principal *importandexportofcourse.php*. Ensuite, nous avons défini les instructions nécessaires. Le développement impose plusieurs contraintes.

Par exemple, la première ligne d'instruction, représentant la définition de la classe du bloc, doit être définie exactement de la manière indiquée sur la figure 4, à l'exception du nom du bloc. Chaque classe doit débuter par la méthode *init()* qui permet d'instancier les valeurs des variables de la classe. La figure 4 montre un extrait du fichier *block_ importandexportofcourse.php* dont la fonction *init()* est consacrée à la définition du titre du bloc.

```php
<?php
require_once('importandexportofcourse_up_form.php');

class block_importandexportofcourse extends block_base {

  function init() {
    $this->title = get_string('pluginname',
    'block_importandexportofcourse');}

  function get_content() {
    global $CFG;
    if ($this->content !== NULL) {
      return $this->content;}

    $this->content = new stdClass;
    $this->content->footer = '';
    $this->content->text   = '<div class="xmlupform">';
```

Illustration 4: Extrait du fichier importandexportofcourse.php

lang/en/block_ importandexportofcourse.php : ce fichier est spécifique à la langue anglaise du bloc. Il est également possible de créer les blocs sous différentes langues. Ce fichier définit le nom du bloc qui sera affiché sur l'espace cours de la plateforme ainsi que tous les noms qui seront utilisés dans le bloc. La figure 5 illustre les instructions de ce fichier.

```php
<?php
$string['upload'] = 'Upload';
$string['uploadfilename'] = 'File to upload :';
$string['download'] = 'Download';
$string['newxmlinoutblock'] = '(new XMLInOut block)';
$string['pluginname'] = 'XML In/Out';
?>
```

Illustration 5: Les instructions du fichier lang/en/block_importandexportofcourse.php

version.php : ce fichier est générique pour tous les blocs de la plateforme. Il possède une seule instruction consistant à définir la version du bloc développé. La figure 6 illustre un exemple de ce fichier.

```php
<?php
   $plugin->version = 2011121010;
```

Illustration 6: Extrait du fichier version.php

Toutefois, la conception et le développement de ce module doit prendre en compte la contrainte liée à la modification et à l'adaptation des cours (existants sur la plateforme) à l'aide des outils externes. Ce choix apporte des éléments de réponses à une de nos problématiques : assurer une conception continue des situations pédagogiques. Ceci demande des utilisations successives de l'import/export. Après avoir exporté son cours de la plateforme, l'enseignant-

concepteur peut effectuer des modifications nécessaires sur l'outil de conception avant de l'opérationnaliser à nouveau par le module développé. Lors du développement, nous avons choisi d'utiliser deux fonctionnalités existantes (sauvegarde et restauration) tout en ajoutant les instructions nécessaires pour leurs utilisations et en les adaptant à nos besoins.

Le module intègre alors deux processus : un processus d'export consistant à générer un document (XML) spécifiant toutes les informations liées à la conception pédagogique du cours et un processus d'import pour "opérationnaliser", dans l'espace cours, le scénario spécifié dans un fichier XML. Les fichiers des scénarios importés ou exportés ont l'obligation d'être conformes au schéma XML du formalisme métier de la plateforme Moodle.

Nous avons ainsi spécifié un schéma XML explicitant formellement ce langage. La figure 7 montre un extrait du schéma XML. Ce schéma est dérivé du métamodèle de langage conception pédagogique de Moodle. Il représente un format de communication avec des outils extérieurs à la plateforme. Il pourra aussi servir pour valider les documents XML qui seront importés/exportés. Ce schéma a été spécifié en utilisant le design pattern russian dolls[1] parmi plusieurs patrons de conception de schéma XML tels que *Salami Slice Design*, *Venetian Blind Design*, *Garden of Eden Design*, etc.[2]. Aucun élément du document XML n'était partageable, il n'y a donc pas de nécessité d'avoir un mécanisme de réutilisation des éléments ; ce patron est très proche de la structuration du document et donc favorise sa compréhension.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  ...
    <xsd:element name="course">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="shortname" type="xsd:string"/>
          <xsd:element name="fullname" type="xsd:string"/>
          <xsd:element name="summary" type="xsd:string"/>
          <xsd:element name="sections">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="section" maxOccurs="unbounded">
              <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="name" type="xsd:string"/>
              <xsd:element name="summary" type="xsd:string"/>
              <xsd:element name="activities">
                <xsd:complexType>
                  <xsd:choice minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="resource">
                  ...
```

Illustration 7: Extrait du schéma XML équivalent au métamodèle du langage de conception pédagogique de Moodle

Afin d'assurer le bon fonctionnement de ces deux processus en termes de première création (premier import dans un cours vide) et d'adaptation (par ajouts et suppressions d'éléments), nous

1 Russian dolls est un patron de conception de schémas XML. Il représente le schéma qui reflète la structure des documents.

2 http://www.xfront.com/GlobalVersusLocal.html#FirstDesign

les avons testés dans un premier temps sur la base de deux concepts (forum et label). Puis, dans un deuxième temps, nous avons étendu le développement sur l'ensemble du métier explicité.

2.2.1 Description du processus d'export

Le premier processus (figure 8) se charge donc de la sauvegarde du cours et de sa formalisation conformément au métier que nous avons formalisé. Concrètement, il s'appuie sur la fonction de sauvegarde (*backup*) de Moodle pour générer le package du cours. Les données des fichiers du paquetage étant au format XML. Nous avons élaboré une transformation XSLT permettant de parcourir les différents fichiers XML décompressés du *backup* et de générer un fichier XML unique conformément au schéma XML (figure 7). La procédure XSLT est composée de deux parties. La première partie consiste à générer les règles pour la lecture des fichiers du *backup*. La deuxième partie consiste à générer les règles de la lecture du schéma XML. Pour chaque élément du schéma, une règle lui est associée. L'algorithme général de ce processus est le suivant :

- 1. Choisir le cours à exporter
- 2. Appeler la fonction backup de Moodle pour générer le paquetage du cours
- 3. Décompresser le paquetage du cours
- 4. Créer un fichier XML vide
- 5. Pour chaque élément du schéma XML de la plateforme,
    - 5.1. Lire les fichiers XML et identifier les informations dédiées à cet élément
    - 5.2. Ecrire ces informations au sein du fichier XML

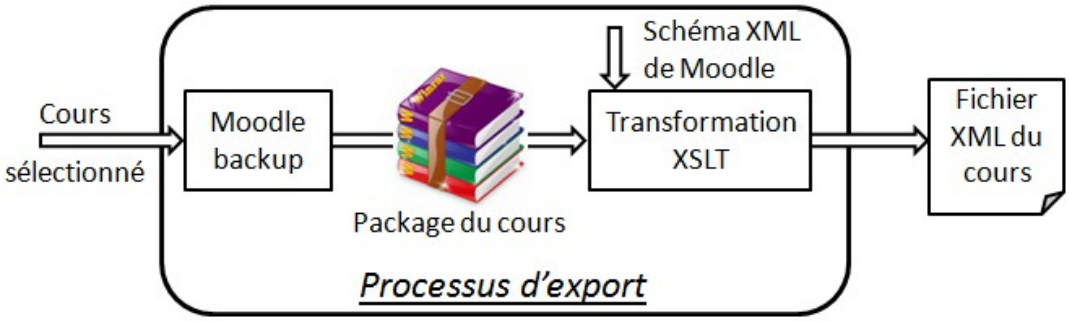

Illustration 8: Processus d'export de cours sur Moodle

Le fichier XML résultant du processus est directement manipulable par des outils de conception spécifiques. Ils sont développés sur la base du schéma XML/métamodèle du langage de conception pédagogique de la plateforme Moodle. Ils sont capables de charger les cours spécifiés dans un format XML conforme à ce schéma XML.

2.2.2 Description du processus d'import

Le deuxième processus (figure 9) se charge de l'opérationnalisation (l'import) de situations

pédagogiques spécifiées à l'externe des plateformes conformément au schéma XML. Nous avons choisi de nous appuyer sur la fonction *restore* existante dans Moodle pour concrètement opérationnaliser les situations pédagogiques. Pour cela, il faut réaliser deux opérations successives et complémentaires. La première consiste à compléter et transformer le scénario à importer dans le format requis par la fonctionnalité de *backup*. Ce format est un package compressé (format zip) dont la structure (figure 2) est conforme à celle du package de sauvegarde (nombreux dossiers et fichiers XML). La deuxième opération s'appuie sur la fonction *restore* pour établir l'opérationnalisation. Le processus est basé sur des transformations exprimées en langage XSLT. Ces règles de transformation modifient les aspects syntaxiques du scénario/cours à importer sans altérer leur sémantique.

A défaut de détailler techniquement ces règles, l'idée générale de l'algorithme de transformation que nous avons spécifié consiste à :

1/ réaliser le backup du cours concerné par l'import (via la fonction *backup*),

2/ compléter et modifier ce paquetage par les informations spécifiées dans le fichier XML du nouveau scénario (via la procédure XSLT développée),

3/ opérationnaliser le package modifié (via la fonction *restore*).

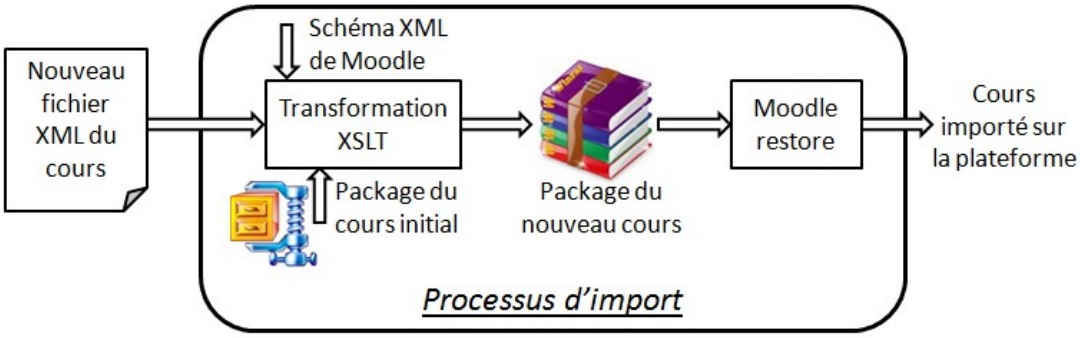

Illustration 9: Processus d'import de cours sur MOODLE

La procédure XSLT est basée sur le fichier XML du scénario à opérationnaliser ainsi que le schéma XML. Après avoir déterminé l'identifiant du cours (enregistré dans le fichier XML), la procédure réalise une sauvegarde du cours via la méthode *backup* de Moodle. Au sein de cette procédure sont définies les règles responsables de mettre à jour les données du paquetage du cours selon les informations spécifiées dans le fichier XML. La procédure possède également des règles permettant la création des nouveaux fichiers dans le paquetage lorsque des nouveaux éléments sont ajoutés dans le cours à opérationnaliser.

Le processus d'import prend en considération l'adaptation d'un cours existant et compatible avec un premier import. Concrètement, la transformation XSLT prend en compte quatre cas de figure :

- (1) la modification des informations d'un concept déjà existant dans le cours (ie. le concept apparaît dans le backup et dans le scénario à importer) ;
- (2) la création d'un élément de scénario non déjà présent dans le cours (ie. le concept n'apparaît pas dans le backup mais dans le scénario à importer) : ce point a nécessité de choisir des valeurs par défaut pour les identifiants des éléments ;

- (3) la suppression d'un élément du cours (ie. le concept apparaît dans le backup mais pas dans le scénario à importer) ;
- (4) l'omission d'un élément (ie. le concept apparaît dans le backup mais pas dans le scénario à importer mais l'outil externe ne gère pas cet élément métier) : ce point très complexe a nécessité l'introduction de l'information du « périmètre métier » abordé par l'outil externe à l'origine de la création du scénario. Nous avons pour cela choisi d'imposer aux outils externes de fournir le schéma XML formalisant le sous-ensemble du langage métier qu'ils manipulent, ce schéma ayant pour contrainte d'être un sous-ensemble de celui de la plateforme.

2.3 Expérimentation et évolution du service d'opérationnalisation

Nous avons expérimenté et testé le module d'opérationnalisation sur de nombreux cas afin de vérifier le bon fonctionnement des quatre possibilités de manipulations sur les scénarios/cours pour Moodle.

A titre d'illustration, voici quelques exemples. Dans le cas où l'enseignant a modifié une activité particulière du scénario telle que «leçon», le processus copie les parties XML du scénario initial en mettant à jour les données de cette activité. Par contre, si cette activité est supprimée, le processus ne copie pas les données de cette activité dans le nouveau package à opérationnaliser. Lorsqu'une activité est nouvellement ajoutée, le programme est capable de générer les parties XML de cette activité et d'initialiser les données associées.

Pendant le développement du module d'opérationnalisation, nous avons rencontré plusieurs difficultés. A titre d'exemple, il était indispensable de distinguer les éléments nouvellement ajoutés aux cours de ceux qui existent déjà. Une des solutions que nous avons adopté est d'initialiser l'identifiant des nouveaux éléments à zéro. Aussi, nous avons proposé une solution technique afin de palier au problème d'utilisation d'outils externes qui ne seraient capables de traiter que partiellement le langage métier que nous proposons pour la plateforme.

Les processus d'import et d'export ont l'avantage de palier au manque de données exclues pendant l'export initial du scénario vers les outils de conception externes puisque ces informations sont ajoutées au scénario avant sa restauration. Ainsi, le scénario à opérationnaliser est toujours complété en incluant tous les détails définis par défaut ou ajustés par le concepteur directement sur la plateforme.

3 Développement du service d'opérationnalisation pour Moodle 2.4

3.1 Description du module de communication

3.1.1 Description du processus d'export

3.1.2 Description du processus d'import

4 Développement du service d'opérationnalisation pour Ganesha

4.1 Description du module de communication

Nous avons choisi de mener une autre expérimentation sur la plateforme de formation Ganesha. Contrairement à la plateforme Moodle, Ganesha ne dispose pas de fonctionnalités de sauvegarde et de restauration de cours. Nous avons développé entièrement le service d'opérationnalisation (figure 10), intégré au sein de l'espace de cours, afin de la rendre capable de communiquer avec les outils de conception externes spécifiques. L'architecture de Ganesha est modulaire. L'espace administrateur de Ganesha propose l'ajout des extensions à cette plateforme via la configuration de certains paramètres (tels que l'adresse du serveur mail SMTP, le numéro du port SMTP, authentification, identifiant, mot de passe, etc.). Nous avons constaté que ces extensions sont difficiles à définir et ne conviennent pas à notre objectif pour la spécification d'un module d'import et d'export de cours.

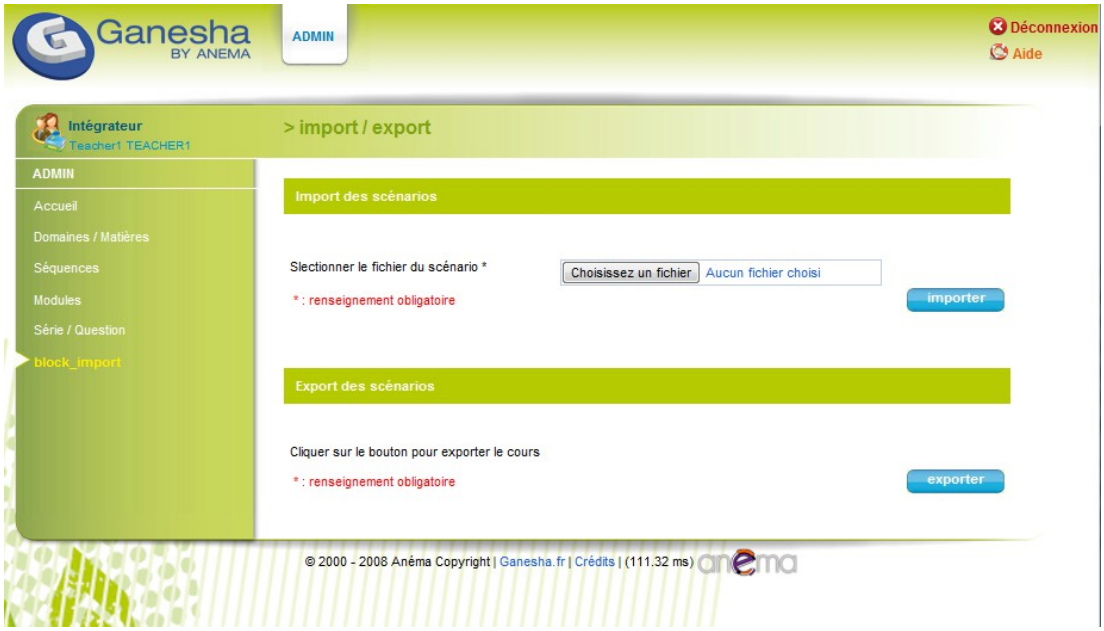

Illustration 10: Bloc import/export ajouté à Ganesha

Nous avons choisi d'intégrer ce module au sein de l'espace associé au profil d'un intégrateur parmi les profils proposés par la plateforme (admin, formateur, intégrateur, conseiller et gestionnaire). Ce profil est le plus proche du profil enseignant-concepteur de cours : l'usager de nos propositions. Nous avons du ainsi respecter deux contraintes principales concernant les droits associés à ce profil ainsi que la charte graphique de l'interface. Le langage de développement du module est le même langage de la plateforme Ganesha : le langage de programmation web PHP.

Le module d'opérationnalisation joue le rôle de passerelle de communication entre les outils externes et la plateforme. Il garantit l'opérationnalisation des informations des scénarios pédagogiques. Nous proposons un processus d'opérationnalisation des situations d'apprentissage (appelées module sur Ganesha). Nous proposons aussi un processus d'export afin de prendre en compte la possibilité de mener des modifications et des adaptations de ces situations à l'extérieur de la plateforme, en particulier avec les outils de conception développés.

Le module repose sur le schéma XML que nous avons spécifié en équivalence avec le métamodèle de langage de conception pédagogique de la plateforme Ganesha. La figure 11 représente un extrait de ce schéma XML. Le module intègre deux processus d'utilisation : un processus d'import pour opérationnaliser les scénarios spécifiés via des outils de conception externes et un processus d'exportation des cours (ou une partie) dans un format externe conforme au schéma.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
...
    <xsd:element name="Sequence" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="Titre" type="xsd:string"/>
            <xsd:element name="Objectifs pedagogiques" type="xsd:string"/>
            <xsd:element name="id" type="xsd:nonNegativeInteger"/>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
              <xsd:element name="SCORM 1.2" maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="Activite" type="xsd:string"/>
                    <xsd:element name="Description" type="xsd:string"/>
                    <xsd:element name="Mots clefs" type="xsd:string"/>
                    <xsd:element name="Visible" type="b01" default="1"/>
                    <xsd:element name="id" type="xsd:nonNegativeInteger"/>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
              <xsd:element name="Activite collaborative" maxOccurs="unbounded">
                <xsd:complexType>
                  <xsd:sequence>
                    <xsd:element name="Activite" type="xsd:string"/>
                    ...
```

Illustration 11: Extrait du schéma XML de Ganesha

Contrairement au cas de Moodle, le processus de conception et d'opérationnalisation proposé pour Ganesha n'impose pas la contrainte de commencer par exporter le cours avant de spécifier ou modifier son contenu sur l'éditeur. Nous avons tenu en compte de cette contrainte dans Moodle afin de réutiliser nos outils pour la plateforme UMTICE[3]. Nous avons eu besoin de l'identifiant du cours requis par les fonctionnalités *backup* et *restore*.

4.1.1 Description du processus d'import

Le processus d'opérationnalisation (import) consiste à implémenter les modèles de situations pédagogiques spécifiées en dehors des plateformes et conformes au schéma XML. La figure 12 illustre ce processus. L'idée générale de l'algorithme de ce processus consiste à parser le fichier XML du modèle et à insérer les données associées dans la base de données. Il ne requiert pas de transformations (XML) sur le modèle. Le processus prend en compte la contrainte d'adaptation d'un cours existant. Concrètement, le processus commence par parser le fichier XML du nouveau cours/module. Il se charge de la création des nouveaux concepts nouvellement ajoutés au cours (ceux qui apparaissent dans le scénario à importer mais pas dans le cours initial). Ce point a nécessité d'initialiser les identifiants des nouveaux éléments à zéro. Le processus se charge également de la mise à jour des informations des concepts existants (ie. le concept existe dans le cours mais il était modifié dans le nouveau cours). Nous avons également pris en compte le cas de la suppression d'un ou plusieurs élément(s) de cours (ie. le concept apparaît dans le cours initial mais pas dans le cours à importer). Le processus doit ainsi vérifier que certains éléments tels que

3 Umtice est une plateforme de formation à l'origine de la plateforme Moodle. Elle est utilisée au sein de l'université du Maine

les séquences n'appartiennent pas à un autre module avant leurs suppressions. Cette contrainte est due à la possibilité d'appartenance d'une séquence à plusieurs modules (prouvée par le métamodèle du langage de conception de Ganesha résultant de l'application du processus d'identification du langage de conception pédagogique sur Ganesha).

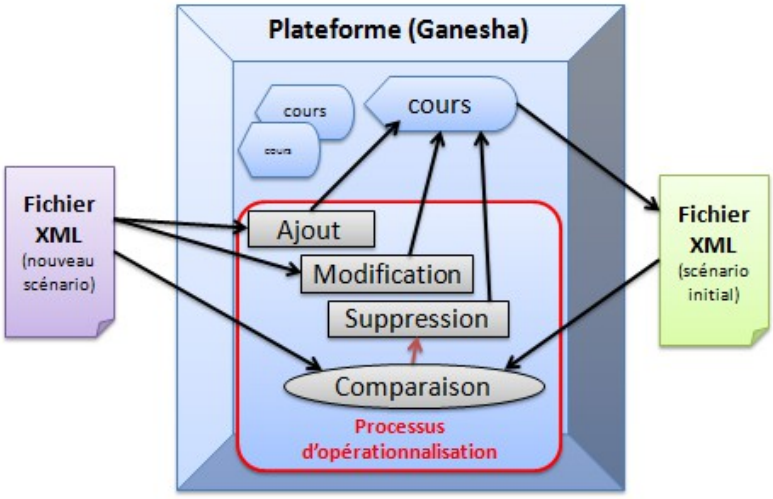

Illustration 12: Processus d'opérationnalisation développé pour Ganesha

4.1.2 Description du processus d'export

Les cours sur Ganesha sont désignés par le concept de module. Les modules sont associés à des matières qui sont également associées à des domaines. Le processus d'export (figure 13) permet de sauvegarder un module donné en format XML. Après avoir choisi un module, le processus se charge de générer le fichier XML correspondant. Dans le but de préserver la sémantique de la plateforme, nous avons tenu en compte la possibilité d'exporter les propriétés caractéristiques du domaine et de la matière. Le processus interroge la base de données de Ganesha, il examine les tables qui embarquent le contenu du module. Les tables sont consultées en suivant la hiérarchie du schéma XML (par exemple la table des séquences est interrogée avant celle des activités). Ce processus aboutit à la génération d'une instance de document XML conforme au schéma XML de la plateforme et directement manipulable par des outils de conception spécifiques à Ganesha.

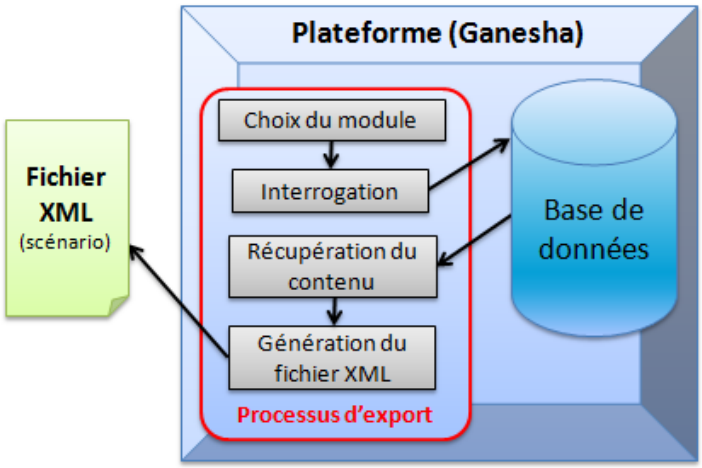

Illustration 13: Processus d'export développé pour Ganesha

5 Conclusion

Dans ce document, nous avons exposé les propositions portant sur le développement de services d'opérationnalisation et d'export dédiés à des plateformes spécifiques. Ces processus prennent en charge la mise en place et l'export des scénarios sérialisés sous un format XML et conformes au schéma XML du langage de conception pédagogique de la plateforme cible. Ces scénarios peuvent être spécifiés et/ou modifiés via des outils de conception externes présentés dans les documents D5.1 et D5.2.

Les processus d'import et d'export ont l'avantage de palier au manque de données exclues pendant l'export initial du scénario vers les outils de conception externes puisque ces informations sont ajoutées au scénario avant sa restauration. Ainsi, le scénario à opérationnaliser est toujours complété en incluant tous les détails définis par défaut ou ajustés par le concepteur directement sur la plateforme.

6 Références

Abedmouleh A. (2010, 6-7 mai 2010). Approche Domain Specific Modeling pour l'opérationnalisation de scénarios pédagogiques sur des plateformes de formation à distance. In: Rencontres jeunes chercheurs en EIAH, RJC 2010, Lyon, France, p.141-142.

Abedmouleh A., Oubahssi L., Laforcade P. et Choquet C. (2012 b, 25-27 juin 2012). Expressing the implicit instructional design language embedded in an LMS: motivations and process. In: Computers and Advanced Technology in Education, Naples, Italie, p. 774-064.

Abedmouleh A., Oubahssi L., Laforcade P. et Choquet C. (2012 c, 24-27 juillet 2012). An analysis process for identifying and formalizing LMS instructional language. In: International Conference on Software and Data Technologies, Rome, Italie, p. 218-223.