A MODEL FOR MAINTAINING INTEROPERABILITY OF COARSE XML SHARABLE LEARNING OBJECTS AFTER RE-AUTHORING IN A STANDARDS-BASED EDITOR

Rajendra G. Singh
rajsingh@fsa.uwi.tt

Margaret Bernard
mbernard@fsa.uwi.tt

Department of Mathematics and Computer Science
Faculty of Science and Agriculture
The University of the West Indies
St. Augustine, Trinidad, W.I.

ABSTRACT

Learning Objects are being packaged for interoperability using internationally agreed standards such as the Sharable Content Object Reference Model (SCORM). A course-designer aggregating heterogeneously created Sharable Learning Objects (SLOs) from a repository when challenged with coarse SLOs would find a standards-based editor to be a useful tool for re-authoring if the interoperability of an edited coarse SLO could remain intact after the edits. Further, since many authoring applications are beginning to use XML to define SLOs (XSLOs) it is inevitable that a repository will contain (solely) heterogeneously authored XSLOs and as a result, we focused our research on maintaining the interoperability of re-authored coarse XSLOs.

This paper presents research conducted to determine if a coarse XSLO could be edited in a standards-based editor without affecting its interoperability. Initially a model for the XSLO was developed and titled *SIM*. We found that if the SIM is applied during the authoring process, it will afford protection to the interoperability of the XSLO when the XSLO is subsequently edited in a corresponding SIM-aware DOM editor.

We describe how a DOM editor (based on the Document Object Model) can be transformed into a SIM-aware DOM editor such that it is still standards-based. In addition, we present the application *eLearnPro*, which was developed to test the interoperability of XSLOs that were authored using the SIM, then re-authored by a SIM-aware DOM editor.

1. INTRODUCTION

Instructional designers and developers of e-learning today face new challenges and opportunities for developing course content. Currently, designers are being asked to design small units of stand-alone instruction, called Learning Objects, that can be tagged and managed in a repository and which can later be assembled into learning modules. These Learning Objects are being packaged for interoperability using internationally agreed standards such as the Sharable Content Object Reference Model (SCORM) [1], resulting in Sharable Learning Objects (SLO). In addition, many authoring applications are using XML to define SLOs and eventually a repository will contain (solely) SLOs authored with different applications but with the commonality of XML.

One consistent problem with the many definitions given to a SLO is that the granularity of the SLO has remained a subjective feature [2]. The granularity specifies how fine or coarse the SLO should be and holds a direct relationship to the number of learning objectives, ideally one. The coarseness of the SLO is therefore generally left to the guided discretion and experience of the instructional-designer. As a result, a successful initial use of a SLO in one course can sometimes later be discovered to be coarse when it is being considered for reuse in another instance. If such an instance of reuse requires only a subset of the original learning objectives then the course-designer is challenged to re-authoring the SLO or finding another SLO. If re-authoring is to take place, the consensus is that the instructional-designer is responsible for re-authoring the SLO [3, 4], usually in the original authoring application. This re-authoring task has never been placed in a framework and given to standards-based software.

In this paper, we present a model for dynamic internal access to coarse XML Sharable Learning Objects that reduces re-authoring efforts [5]. A coarse XML SLO (XSLO) can be dynamically accessed outside its original authoring application and changes made to the internal elements and attributes using its DOM architecture. This type of editing would normally expose the XSLO to (unintentional) damage to its interoperability. However, we introduce a *Sharable Learning Object Interoperability Model, SIM,* which provides a mechanism for editing a SCORM conformant XSLO while protecting the interoperability of the XSLO. We describe a proof-of-concept prototype application, *eLearnPro* (*Pro* short for *protection*) that was created to assist with the authoring of XSLOs and to validate the extent of protection that is afforded by the SIM. The eLearnPro application therefore included a SIM-aware DOM editor to effect changes to coarse XSLOs, and facilities to verify interoperability both before and after editing.

One key factor in developing the SIM was to differentiate a Reusable Learning Object (RLO) from a Sharable Learning Object (SLO), which is presented in the next Section.

2. REUSABLE AND SHARABLE LEARNING OBJECTS

Researchers in the e-learning community are yet to agree on a universal and comprehensive definition of a Learning Object (LO) [6]. There are however enough common ground among the definitions to consider LOs that are *reusable* and *sharable*, although Feldstein [7] question the need for reusability. In the research, we decided that the definition given in [8], which although highly subjective in parts was useful to explore the concepts of a coarse SLO. The definition is as follows: *A SLO is a reusable chunk of content with the following three fundamental properties:*

- *instructionally sound content with a focused learning objective*
- *facility that allows the learner to practice, learn and receive assessment*
- *metadata or keywords that describe the object's attributes and mechanisms for communicating with any Learning Management System (LMS).*

Using this base definition, we differentiated between a RLO and a SLO by dividing the above original definition to give the following: *A **RLO** is a reusable chunk of content with the following two fundamental properties:*

- *instructionally sound content with a focused learning objective*
- *facility that allows the learner to practice, learn and receive assessment*

*and, a **<u>SLO is a RLO</u>** with the following additional property:*

- *metadata or keywords that describe the object's attributes and mechanisms for communicating with any LMS.*

The latter part of this last property is what determines if a SLO is sharable. The first part ensures that a specific SLO can be discovered among others in a repository of SLOs, and the entire property can be defined as providing a level of interoperability. The research was concerned with maintaining this level of interoperability when re-authoring is done outside the original authoring application.

3. ACHIEVING AND MAINTAINING INTEROPERABILITY

The SCORM version 1.2 was applied to achieve the interoperability of XSLOs. The SCORM provides a framework for defining SLOs and corresponding e-learning applications interoperability. It does this through its Content Aggregation Model (CAM) and Runtime Environment (RTE). The SCORM CAM handles three aspects: the Content Model, Metadata, and Content Packaging. The Content Model assists in defining the reusable properties of LOs, the metadata allows for the discovery of parts (e.g. Assets) or the whole SLO and information about the aggregation, and the Content Packaging guides the organization and structure of content for the learning experience. The RTE on the other hand specifies how the SLOs are launched and the mechanism for communicating with any LMS, which is essentially defining the sharable properties of the SLO. The choice of using the SCORM was based on this well-defined framework. It is relatively easy to identify the various components that form the level of interoperability that we wanted to examine within a SLO and corresponding LMS.

Further, the definitions given in Section 2 were instrumental in developing an approach to the problem. We decided that it would be obvious to author XRLOs and then package them to be interoperable, which lead us to develop the SIM to guide the process. Re-authoring would then be done using a SIM-aware DOM editor, which ensures that the interoperability of the XSLO is maintained.

3.1 The SIM (XML Sharable Learning Object Interoperability Model)

The SIM is our model for a XSLO, which is designed to remain interoperable after editing in a SIM-aware DOM editor. The model is depicted in Figure 1.

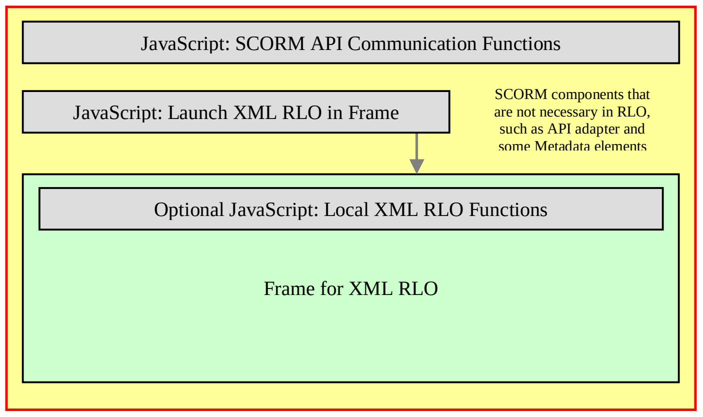

JavaScript: SCORM API Communication Functions
JavaScript: Launch XML RLO in Frame
SCORM components that are not necessary in RLO, such as API adapter and some Metadata elements
Optional JavaScript: Local XML RLO Functions
Frame for XML RLO

Figure 1 – The SIM

We stated earlier in Section 2 that a SLO is a RLO packaged for interoperability. The interoperability of the XSLO was achieved by applying the SCORM version 1.2. The model isolates the SCORM API communication functions, API adapter and XSLO metadata from the XRLO, and as a result a DOM editor can be used on the XRLO component of the SIM to effect changes. Further, since the RLO was defined to contain the learning objective(s) then a coarse XRLO could be edited by a DOM editor without affecting the SCORM functionalities.

3.2 The SIM-aware DOM Editor

When we speak of a SIM-aware DOM editor, we mean that the file-management capabilities were extended in a standard DOM editor to autonomously identify the XRLO documents and present them for editing, and for subsequently saving the documents. The identification of the XRLO documents was done using a metadata element <*rlodoclist/*> in our XSLOs, which contains information on names and relative locations of XRLO documents used by the given XSLO. The DOM editor file-management facility must be updated to include a search-component to locate the <*rlodoclist/*> metadata element (traversing the XSLO DOM-tree), and retrieve the names and relative locations of the XML documents that comprises the XRLO. If document names and locations change then the added components of the file-management facility must update the respective attributes of the <*rlodoclist/*> metadata element in the XSLO. Figure 2 shows the <*rlodoclist/*> metadata element and corresponding attributes that were used.

```
<rlodoclist numdocs="1" name1="default" location1="default" />
```

Figure 2 – The <*rlodoclist/*> Metadata Element and Attributes

4. VERIFYING CONCEPTS OF THE SIM

To actually construct a XSLO we decided to use the XML User Interface Language (XUL) [9] for authoring the RLOs thus producing XML RLOs (XRLOs), and to package for interoperability using the SCORM version 1.2, resulting in SCORM conformant XSLOs. The application eLearnPro was then developed and used to test the interoperability of coarse XSLOs both before and after editing in a SIM-aware DOM editor.

4.1 The XML User Interface Language (XUL)

XUL is a programming language that uses XML to define *widgets*, which are common for building Graphical User Interfaces. A XUL document is therefore a XML document. XUL documents can be rendered by Mozilla and Mozilla-based browsers [10] in the same way that HTML documents are rendered. In addition, Mozilla offers a facility for rendering a XUL document as a *window-type* application.

Programming the event-handlers in XUL is done using JavaScript programming. The choice of XUL combined with JavaScript meant that we could produce XSLOs with both content and event-handling capabilities for any operating system platform that runs Mozilla, which include many flavors of Microsoft Windows, MAC OS, Linux and Unix Operating Systems. In addition, the eLearnPro application, inclusive of the SIM-aware DOM editor could be developed as a window-type application using the same base of programming technologies and skills. The good support for DOM Level-1 and DOM Level-2 functionalities along with some unique *tree* data structures [11] available in XUL contributed further to the choice for developing the DOM editor in XUL.

A major disadvantage of the XUL choice is that XUL documents cannot be rendered by the Microsoft Internet Explorer browser, however the Mozilla organization has active projects attempting to develop a transformation tool that would convert a XUL-document to a HTML-document.

4.2 THE eLearnPro APPLICATION

Figure 3 shows the Main Interface of the eLearnPro application that was developed. The application was built using XUL and JavaScript programming. The application

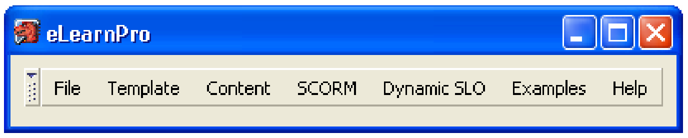
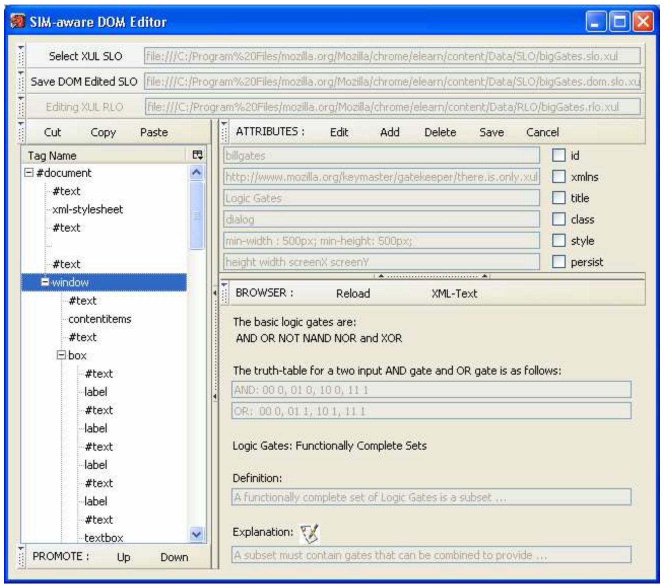


Figure 3 – The eLearnPro Application

is therefore a Mozilla-based application and will execute on any platform that supports Mozilla.

The *Template* and *Content* menus provide access to tools for authoring XUL RLOs. The *SCORM* menu includes a menu item for packaging the XUL RLOs as SCORM version 1.2 conformant XUL SLOs and structured according to the SIM. Also included is a menu item for validating that the XUL SLOs are in fact conformant to the SCORM version 1.2. The next menu *DynamicSLO* provides access to a SIM-aware DOM editor, which is shown in Figure 4 with a XUL SLO loaded. After editing a XUL SLO with the SIM-aware DOM editor, it can be re-checked for conformance to the SCORM version 1.2 using the respective menu item in the *SCORM* menu.



Select XUL SLO: file:///C:/Program%20Files/mozilla.org/Mozilla/chrome/elearn/content/Data/SLO/bigGates.slo.xul
Save DOM Edited SLO: file:///C:/Program%20Files/mozilla.org/Mozilla/chrome/elearn/content/Data/SLO/bigGates.dom.slo.xu
Editing XUL RLO: file:///C:/Program%20Files/mozilla.org/Mozilla/chrome/elearn/content/Data/RLO/bigGates.rlo.xul

```
#document
    #text
    xml-stylesheet
    #text
    #text
  window
      #text
      contentitems
      #text
      box
          #text
          label
          #text
          label
          #text
          label
          #text
          label
          #text
          textbox
```

id	billgates
xmlns	http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul
title	Logic Gates
class	dialog
style	min-width : 500px; min-height: 500px;
persist	height width screenX screenY

The basic logic gates are:
AND OR NOT NAND NOR and XOR

The truth-table for a two input AND gate and OR gate is as follows:
AND: 00 0, 01 0, 10 0, 11 1
OR: 00 0, 01 1, 10 1, 11 1

Logic Gates: Functionally Complete Sets

Definition:
A functionally complete set of Logic Gates is a subset …

Explanation:
A subset must contain gates that can be combined to provide …

Figure 4 – A XSLO Loaded for Editing in the DOM-Editor
Note that only the XRLO DOM-Tree is exposed for editing when the XSLO is selected. Saving the XSLO saves the edited XRLO and updates the corresponding metadata element <*rlodoclist/*>. The DOM editor allows the user to cut/copy/paste elements and edit/add/delete attributes of correspondingly selected elements. Elements within the DOM-tree can also be repositioned. The XRLO document is rendered in real-time while editing or the user can view the XRLO document as XML-text.

5. RESULTS

The research found that when the SIM is applied in the authoring process to create SCORM version 1.2 conformant coarse XSLOs, the coarse XSLOs remained conformant to the SCORM version 1.2 after editing the corresponding coarse XRLO in the SIM-aware DOM editor. This result was expected since the SCORM API Adapter and SCORM functions *LMSInitialize(), LMSCommit(), LMSFinish()* and *LMSGetDiagnostic()* were all inaccessible to the editor. By not interfering with these LMS *communication mechanisms* the XSLO was always rated as at least RTE1 (minimally) conformant to the SCORM version 1.2.

The SCORM allows for data exchange between the XSLO and LMS using the functions *LMSGetValue()* and *LMSSetValue()*, and for error checking and reporting using the functions *LMSGetLastError()* and *LMSGetErrorString()* which must sometimes be called from within the XRLO. The accuracy of information communicated by these functions is not really an interoperability issue. It is an integrity issue when communicated information can be affected if the XRLO is edited. The SIM does not address this issue. It remains the responsibility of the course-designer to ensure that data communicated by the XRLO remains consistent after editing a coarse XSLO.

5.1 Selective Editing of the XSLO

Although our SIM-aware DOM editor did not allow the course-designer to access and edit any other components but the XRLO, it would be a simple task to extend it to allow optional access to selected elements and attributes outside the XRLO, for example, the SCORM CAM *Information Model Metadata* elements. This feature can be an advanced option in the SIM-aware DOM editor for experienced course-designers. Including such an option would however expose the XSLO interoperability to intentional edits as well as unintentional damage. The course-designer that reduces the coarseness of a XRLO can easily argue that the facility is necessary to access and edit metadata that is used for discovery of the newly edited XSLO within a repository, and such access should rightly be facilitated. Selecting which metadata elements can be updated must be carefully considered.

6. CONCLUSIONS AND FUTURE WORK

It is expected that SLOs will be created in mass amounts and held in repositories for subsequent discovery and aggregation by course-designers. Further, many authoring applications are beginning to use XML to define SLOs and eventually a repository will contain (solely) SLOs authored with different applications but with the commonality of XML. The conclusion is that the SIM affords a generally good level of protection to XSLO interoperability when the DOM is utilized to effect internal edits. As a result, we can say that it is also established that heterogeneously created coarse XSLOs authored according to the SIM and residing in repositories, can be edited in a common standards-based application without affecting interoperability, thus reducing re-authoring efforts.

In planned future work, we will attempt to utilize (and refine) the SIM in other advantageous ways, such as, to facilitate real-time autonomous (and artificially intelligent) editing by course-aggregating software modules in any standards-based LMS.

7. REFERENCES

1. ADL Technical Team. (2001*)*. Sharable Content Object Reference Model (SCORM) Version 1.2 - The SCORM Overview [online]. http://adlnet.org
2. Quinn, C. & Hobbs, S. (2000). Learning Objects and Instructional Components, *Educational Technology and Society*, *3(2).* http://ifets.ieee.org/periodical/vol_2_2000/discuss_summary_0200.html
3. Robson, R. (2001). All about learning objects [online]. http://www.eduworks.com/LOTT/tutorial/learningobjects.html
4. Barron, T. (2000). Learning Object Pioneers, *ASTD Learning Circuits*. http://www.learningcircuits.com/mar2000/barron.htm
5. Singh, R. (2003). Dynamic Internal Access to Coarse Sharable Learning Objects: An approach for reducing re-authoring efforts, *M.Sc. thesis*, *The University of the West Indies, Trinidad*.
6. Wiley, D. A. (2000). Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. In D.Wiley (Ed.), *The Instructional Use of Learning Objects* [online]. http://reusability.org/read/chapters/wiley.doc
7. Feldstein, M. (2002). Do You Really Need Reusability? [online]. http://www.elearnmag.org/subpage/sub_page.cfm?section=8
8. Cuthbert, A., Himes, F. (2002). Creating Learning Objects with Macromedia Dreamweaver MX [online]. http://www.macromedia.com/resources/elearning/objects
9. Deakin, N. (2003). XUL Tutorial [online]. http://www.xulplanet.com/tutorials/xultu
10. Mozilla. (2003). Why use Mozilla? [online]. http://www.mozilla.org/why
11. XULPlanet. (2002). XUL Element Reference [online]. http://www.xulplanet.com/references