SCORM 2.0: Content Authoring Standards and Services

by Aaron Silvers

SUMMARY: SCORM 2.0 should expose content authoring and packaging services to promote interoperability, relying on existing and developing technologies, standards and specifications.

The Current Challenge

Many are familiar with the challenge of content interoperability when it comes to migrating content from one Learning Management System (LMS) to another. The issues are evident with SCORM 1.2, which had three different levels of conformity, presenting content developers and often adopters with a challenge (and just plain ignorance) that such differences either existed or were of any relevance. Other issues expanded into SCORM 2004, which continued a tradition of offering no demands on implementation of the specifications outlined -- only demanding conformance to the test. This results in real-world differences across systems in terms of presentation rules with respect to Sequencing and Navigation and handling of Run-Time Data Model Elements like *datafromlms* which breaks on some LMSs that, despite their certifications, produce failures with respect to optional elements in a manifest that some LMSs require. Improved testing would decrease issues like this; however, it's predictable that problems like this will continue despite the specification and despite the testing tool(s). There must be a more endurably way of dealing with the issue of content interoperability.

It is my belief that the notion of conforming to the specification promotes the same challenges to interoperability for SCORM content as the challenges we see in providing interoperable Run Time Environments. Because there are so many ways to construct content, many combinations may pass the test, but the Conformance Test does not validate for context -- in other words, is the Data Model or the Sequencing instructions being used as a shared understanding of the SCORM specifications would suggest or anticipate?

Providing a Content Authoring Standard

Looking at common practices in both plain-jane web content development and in the E-Learning world, the use of XML as a basis for data is now commonplace. Though it may be difficult to obtain data points to this end, XML-based web content may account for as much as half of all web content delivered to a user; in E-Learning, XML-based content probably accounts for 80%. The overwhelming majority of Learning Content authoring tools export the core, unskinned content in XML that conforms to some proprietary schema and then displays in a UI and template -- usually harnessing Adobe Flash. This method of content deployment solved the problem of getting content to display the same in different systems (they cheated and stuck everything inside of Flash), but different tools produce different tracking experiences even with Run-Time data in different LMSs. Some tools track both a success_status and a completion_status even when running in (supposedly) SCORM Version 1.2 environments. Very, very few tools support Sequencing and Navigation in SCORM 2004.

A Content Authoring Standard based on an XML core was heralded by OASIS in April, 2008 (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=dita-learningspec). The use cases for DITA in the context of Learning and Training content were identified by the panel as:

Enable indexing, searching, and retrieval of learning content
By structuring content with DITA topics and maps as self-contained learning objects matched with appropriate DITA metadata, it is possible to enable fast index, search, and retrieval of learning content that meets specific learning goals and objectives.

Creating custom courses quickly
A company has a large inventory of topic-based content that is used to provide technical and troubleshooting information about a set of componentized software products. It desires to enable field engineers to quickly identify technical content that is suitable for providing on-site training. With DITA learning topics and maps, the field engineers are able to quickly identify the specific technical content that matches specific learning objectives, and pull together the learning content that meets specific customer problems.

Making technical content available for direct sharing and reuse in learning and training
A DITA learning specialization makes it possible to define a context for and directly assemble and use existing technical content for delivery as learning and training. The DITA approach identifies consistent structures and patterns, and leverages them to enable a consistent

approach for sharing content across teams. The result is much more opportunity to share content between different providers and across areas of expertise, to learn from each other, and to deliver content and the learning experience consistently. As a result, instead of copy, paste, and make unique as the norm, we have write once and share with others as the new norm.

Templates: Sculpting the Content Model

Whether or not DITA is "the" standard that should be championed for Content Authoring is subject for healthy discussion. If we assume that Content Authoring uses some XML schema as a core, the nature of the content is then subject to validation. This inevitably makes predictable transformation of such content at the presentation layer.

Recent discussions on this subject have yielded exciting ideas on how such content might be transformed into readable content, how it could be stored in a repository and how it could be delivered to users.

Steve Flowers comments on the Flash for Learning blog...

"Picture a standard runtime packaging that consumes [a content] organization specification to run a course. [With] common source pooling [the system] can start to access bits of [the content organization] (like DITA) from within a package...

Move that backwards to a design standard (prior to final transformation) that can be handed off between centers, between organizations, and becomes a (potentially unused) part of the content package.

[To] see the design specifications, edit the design, etc.. it's suddenly really easy if it's contained in the content package. In the case of service driven assemblies, mash-ups, etc.. these specs could be functions of reports.

Now to move a design from one authoring system to another, conceptually it shouldn't matter what the original source for the design was."

Standardizing the abstraction of design and presentation data from core content provides efficiencies that are commonly practiced by content developers. The notion of design layer abstraction described above is intended to describe what we currently call the "content package," but Steve continues to highlight in the blog comments that template elements provided by a SCORM 2.0 system could be placed in a "shared level" in a repository allowing common elements, like GUI, to

be reused across all content. This would allow for run-time assembly of content.

Transporting such content across federated repositories could be enabled by content packaging services. CETIS (http://wiki.cetis.ac.uk/Get_Involved_with_Transcoder) is developing a web service specifically to package content to a variety of specifications. Such a service could be employed here for runtime assembly of content.

The core content XML could, as Ethan Estes describes it (FFL)...

"define a link to an external content service and the [content package] would have the template for the presentation side to know how to display it. It would need changes as the manifest would need to have attributes for the content address and an attribute for the template address which might be outside the [content package] as well. [This would allow the LMS] to adapt/switch the template based on what is requesting the service-cellphone, flash runtime, html browser on a display in a car."

Template Behaviors

It would also be good to have a mechanism in place so that a content package could contain a default template if none is defined via the LMS or by the requesting organization.

Perhaps such templates could control assets and presentation functions provided by the LMS. Some ideas include control over whether or not to hide or show a Table of Contents; whether or not the content should reside inside a frameset or not; what navigation controls need to be provided by the LMS, etc.
A hierarchy of templates should be defined by SCORM 2.0 should this idea gain traction. The LMS system templates are invoked before Package-level assets are used, as one example.

Why all the talk about templates? When it comes to making the template assets available to an authoring tool or service, a packaging service (like the one proposed by CETIS) can then use templates as a run-time display of content meeting a standard like DITA's. One use case: A request for content comes to the LMS from a mobile device. The LMS determines the type of browser and at delivery uses a packaging service to deliver the XML-based content and associated media to the mobile device with a template specifically made to render appropriately for ease-of-use on that particular device. At the same time, another learner on a different platform requests the same content object from the LMS and the LMS recognizes the screen resolution and the ability (or even preference) for rich, Flash-based interfaces --

allowing such a template to be packaged at delivery to that particular learner. At the same time, a SME is engaged in maintenance and a special template is provided to allow the SME to experience the content in a "review mode" on whatever platform it's being reviewed in.

Idea: Order of Operations for Templates

Manifest calls out a specific template address OR Error returned.
If (error) system verifies if LMS has a default template OR Error returned.
If (error) verify if template file is at root of content package OR Error returned
If (error) returned assume it's a totally custom User Interface (or is legacy content) and launch from the href (not using template system).

LMS level template overrules any content package-level templates values so a content-package level value could be used if no asset or template in the LMS template chain defines that value.

The LMS can chain templates together to allow for modular structure, then the system can apply any values from the content-package-level template.

Idea: Sample Template Structure

XML-based.
Top level:
- Configuration
  - window sizes
  - browser requirements
  - available plugins speced out as content elements
  - os requirements
  - window controls disabled
- LMS elements -
  - Course TOC
  - SCO navigation controls
  - Organizational branding
- Content elements
  - Internal content navigation
  - Screen elements (sizes, positions)

Idea: Delivery of LMS template/assets to Authoring tool

The LMS delivers assets/templates back to a request from an authoring tool via content package; this package has a structure that the authoring file will respect with reserved file and folder names.

To reduce clutter, these assets and templates would remain segmented from the repository since the LMS would already have assets.

Preferences can be set for an authoring application or service to provide special customization for content (i.e. logo graphics, fonts needed, etc).

Attribution

This paper was made possible by the combined thought sharing by Ethan Estes, Steve Flowers and Tom King. Where very clear, I have made a best attempt to attribute specific quotes to the proper person. With people bouncing ideas off one another, it is difficult (if not impossible) to pin any one idea in this paper to a specific person. Any misrepresentation of an idea and/or its origins is a fault of the Editor of this work, and is completely unintentional.