

SCORM 2.0

The Lucas Proposal – SCO-based Sequencing

Abstract

The "Lucas Proposal" for SCORM 2.0 calls for the elimination of simple sequencing. This proposal describes its replacement called SCO-based sequencing. SCO-based sequencing allows courses to be modular at the SCO level but eliminates all of the problems with simple sequencing which include confusing user interfaces, increased time to create courses, limitations to sequencing and increased time to integrate courses with a LMS.

SCO-based sequencing allows a SCO to call another SCO by providing a runtime API to the child SCO. The parent SCO receives all of the SCORM data sent by the child SCO and makes sequencing decisions based on the data. The parent SCO passes the child's SCORM data onto the LMS for storage and reporting. All SCOs are separated into folders for ease of identification and reuse.

Problem definition

Since ADL introduced SCORM 1.2, our e-learning community has enjoyed these 3 benefits:

1. A standard way to describe a course (through metadata and packaging).
2. A standard way to launch a course.
3. A standard way to track important events within a course (completion, score, bookmarks, other state information, interactions, ...).

SCORM 1.2 has been a very positive force within our community. It reduced the cost to deploy courses and provides increased choice to organizations that deliver courses. However, SCORM 1.2 did not provide a standard way to control the sequencing and navigation between SCOs.

SCORM 2004 introduced "simple sequencing" as a standard way to provide sequencing between the SCOs within a course. Simple sequencing has these significant flaws which have limited its adoption:

1. The LMS is forced to provide the navigation between SCOs. Every LMS is free to implement a different navigational user interface so the learner is often presented with an overall user interface that is a confusing mix of interfaces. The course producer has to consider blending the course user interface with each LMS he/she supports.
2. It is difficult to implement simple sequencing in raw XML. There are few sequencing tools available to the course authors and those tools are not very easy to use because of the difficulty in implementing simple sequencing.
3. Simple sequencing makes it difficult or impossible to implement some types of instruction. Example:
 - a. "Simple sequencing" does not allow a cluster-node to have content.
 - b. It is difficult to design a course with "simple sequencing" that allows a learner to jump from a question in one SCO to a tutorial in another SCO.

This white paper describes a new SCORM 2.0 implementation that would provide all of the benefits of SCORM 2004 while eliminating the flaws of the simple sequencing model. There may be several SCORM 2.0 proposals so refer to this whitepaper as the "Lucas Proposal" when discussing alternatives.

Use cases

1. **Course developers and learners** – A company produces a course using SCORM-based sequencing. The course has the same navigation functionality and user interface in every LMS.
2. **Course developers** – A course developer can learn how to create a course using SCO-based sequencing in a fraction of the time it takes to learn how to create a course using simple sequencing. The course developer creating SCO-based sequencing would read the runtime specification plus a small SCO-based sequencing specification. The SCO-based sequencing specification would likely be 10% of the size of the Sequencing and Navigation specification and much less complex.
3. **LMS vendors** – The time to implement SCO-based sequencing is a fraction of the time to implement SCORM 2004.

Stakeholders

1. **Learners** – easier to use courses because there is no confusion over inter and intra SCO navigation.

2. **Organizations** – reduced time and cost to get courses to work correctly with their LMS because the LMSs no longer have a role in navigation.
3. **Course developers** – lower cost because SCO-based sequencing courses are easier to develop and easier to make work with a LMS compared to SCORM 2004 courses using simple sequencing. Better quality product because the course can have full control over navigation
4. **LMS vendors** – The time to implement SCO-based sequencing is a small fraction of the time to implement simple sequencing.
5. **LETSI members** – The time to create a compliance test for SCO-based sequencing is tiny compared to simple sequencing.

Proposed solution

SCORM 2.0 should eliminate simple sequencing. Simple sequencing should be replaced with SCO-based sequencing. Here is how SCO-based sequencing works:

1. The LMS launches the root SCO of the course. This root SCO (like any other SCO in the course) can launch one or more child SCOs by providing a runtime API to the child SCOs. The root SCO (and all other SCOs in the course) can provide any type of sequencing and navigation it desires using the SCORM data returned by the child SCOs. The child SCOs are launched in frames or iframes in the root SCOs HTML page.
2. The root SCO passes SCORM set/get data item requests onto the LMS.
3. The LMS keeps track of all of the SCORM data items for each SCO. Since a SCO can launch a child SCO, we will need to make a minor modification to the SCORM runtime API. The runtime functions will have to include a new parameter. The new parameter is a globally unique ID that the SCO provides. The globally unique ID for a SCO is defined in its manifest file.
4. The root SCO can launch more than one SCO at a time. For example, the root SCO can launch one SCO to provide a tutorial and a second SCO to provide a simulation exercise.
5. Lower level SCOs can launch child SCOs just like the root SCO can do.
6. The course session is complete when the root SCO calls its terminate function.
7. Generally each SCO is defined in its own folder (a directory) so files from one SCO are distinct from files of another SCO (several SCOs could use common resources from another folder if needed). Separating SCOs into folders makes it easy to identify the SCOs in a course and reuse those SCOs in other courses.

Advantages of SCO-based sequencing

1. The LMS does not provide any portion of the navigation interface. So, the learner gets a better experience when taking the course.
2. The course builder can use the same technology to build the navigation between SCOs as he/she uses to create the SCOs. The course developer does not have read the difficult-to-understand sequencing and navigation specification (how long did it take you to fully understand the sequencing and navigation specification?). The course developer's understanding of the SCORM runtime API will allow him/her to quickly understand how to use the runtime API to control sequencing in his/her course.
3. SCO-based sequencing can provide a more sophisticated model versus simple sequencing. Cluster-nodes can contain content. It is much easier to implement a course that jumps between assessments and tutorials. It is much easier to implement adaptive tutorials based on the performance on a pre-test.
4. The LMS vendors can support SCO-based sequencing with little effort. A SCORM-based LMS application already knows how to keep track of data for multiple SCOs within a course.
5. The current version of the SCORM runtime API is only available in JavaScript. This means that a course developer cannot use Flash or Silverlight to create the navigation within a course containing multiple SCOs. The SCO-based sequencing could be implemented in Flash, SilverLight and other interactive-multimedia environments. For example, an author could implement an all-Flash course that has 100% Flash-based SCOs (only the root SCO requires a HTML launch page).

Disadvantages of SCO-based sequencing

1. Simple sequencing is defined in the course manifest which makes the sequencing rules easy to find (but not easy to understand for most course developers). SCO-based sequencing is defined in JavaScript that could be included within any file in the SCO. We could implement a standard naming convention for a JavaScript file that defines the sequencing and navigation to minimize this problem.
2. SCO-based sequencing does not provide a standard way to add or subtract SCOs from a course. The addition and subtraction of SCOs is done in a SCO specific way. However, every course that uses multiple SCOs will have centralized algorithm for the launching of multiple SCOs so the modifications to sequencing can be done in one location.

The author believes that the advantages of SCO-based sequencing far outweigh the disadvantages.

Integration and other technical issues

Changes to the runtime API

The runtime functions will include a parameter that uniquely identifies the SCO. This parameter is a Universally Unique Identifier (UUID). A UUID is easy to generate (example: <http://www.famkruihof.net/uuid/uuidgen>) and is an IETF standard (<http://www.ietf.org/rfc/rfc4122.txt>). Here are examples:

- Initialize("2f23ffe0-6196-11dd-ad8b-0800200c9a66")
- GetValue("2f23ffe0-6196-11dd-ad8b-0800200c9a66", "cmi.suspend_data");

Specification of the UUID

Each SCO will have a manifest similar in function to the imsmanifest.xml file. The UUID will be defined in the manifest. A SCO can read its manifest file to retrieve its UUID.

SCO organization

A course must have a root SCO. The root SCO can call one or more child SCOs. These child SCOs are organized into sub folders. The child SCOs can also call SCOs in sub folders. Example of the file structure of a course:

- Root SCO
 - SCO1 – a child SCO of the root SCO
 - SCO2 – another child SCO of the root SCO
 - SCO1 – a child SCO defined above (SCO2)
 - SCO2 – a child SCO defined above (SCO2)
 - SCO3 – another child SCO of the root SCO

Existing implementations/prototypes

No prototypes exist at this time. e-Learning Consulting, LLC will produce an implementation if it is helpful to support this proposal.

Summary and recommendations

SCORM 2.0 should eliminate simple sequencing and replace it with SCO-based sequencing. SCO-based sequencing allows courses to be modular at the SCO level but eliminates all of the problems with simple sequencing which include confusing user interfaces, increased time to create courses, limitations to sequencing and increased time to integrate courses with a LMS.

SCO-based sequencing allows a SCO to call another SCO by providing a runtime API to the child SCO. The parent SCO receives all of the SCORM data sent by the child SCO and makes sequencing decisions based on the data. The parent SCO passes the child's SCORM data onto the LMS for storage and reporting. All SCOs are separated into folders for ease of identification and reuse.

About the author

Leo Lucas is the founder of e-Learning Consulting, LLC (<http://www.e-learningconsulting.com>). Leo has 25 years of experience developing e-learning authoring tools, learning management systems, developer tools and courses. Prior to founding e-Learning Consulting, Leo served in several senior management roles at Click2learn (now SumTotal Systems), Aimtech and the US Army.