SCORM: The Baby and the Bathwater

Mike Rustici, July 28, 2008

Here in the United States, we have a saying "Don't throw out the baby with the bathwater". It means, "don't throw away the good with the bad". In this paper I'll talk about what I think SCORM has done really well over the years that should be maintained (the baby) and where I think SCORM has come up short and needs cleaning up, or perhaps cleaning out (the bathwater).

The Babies

It Works

Despite all the complaints and all the criticism, there's a dirty little secret that isn't often discussed: SCORM WORKS. There's a reason why SCORM is the de facto industry standard and it doesn't have anything to do with DoD instructions or mandated policy. Industry has adopted SCORM because, by and large, SCORM actually works really well. Sure, there are problems and yes there are some things that are less than ideal, but as we get into this process, let's not lose sight of the fact that **we are here to define SCORM 2.0 because there's something here worth keeping.**

Basic Interoperability

SCORM has a number of design goals often referred to as the "-ilities". Chief among the "-ilities" is interoperability.

> "Interoperability: The ability to take instructional components developed in one location with one set of tools or platform and use them in another location with a different set of tools or platform. "

SCORM 1.x has done a fantastic job of providing interoperability to the e-learning industry. Prior to SCORM, there was no such thing as interoperability between different learning systems. Content created in or for one system had to be completely revamped in order to port it to another system. SCORM has completely altered the eLearning landscape, now content can easily be moved from one LMS to another. Sure, there are some occasional bumps in the road, but let's not forget that when we got started there wasn't even a road. **This tremendous increase in interoperability is SCORM's greatest success story and the primary reason why it has become the de facto industry standard. There are things we can do to continue to improve interoperability and SCORM 2.0 should not lose sight of the fact that interoperability is the driving force behind adoption of SCORM.** The content packaging and run time specifications are the workhorses of SCORM's interoperability. They could use some modernization, but let's remember how well they've served us over the years.

Clearly Defined Design Goals and Direction

Philip Dodds and the other early visionaries behind SCORM had a clear vision of what they sought achieve. They articulated their ideas through the "-ilities" and they foresaw learning systems that would adaptively assemble individually tailored instructional material to optimize the learning experience. This vision set a clear direction for the design and evolution of SCORM. Only with a clear destination can the correct path forward be known. Early on, ADL did an excellent job of defining that destination. **Our imperative is now to update and redefine those goals to chart a course for a new destination. What are the new design goals? Are the "-ilities" still relevant? What do we hope to accomplish with SCORM 2.0? These are the most fundamental and important questions we have to answer.** Whatever the goals are, all aspects of SCORM 2.0 should be consistent with achieving these goals. Some aspects of SCORM 2004 have unfortunately drifted from complete consistency with the original design goals; that cannot be allowed to continue.

Appropriately Allocated "Burden of Complexity"

I believe the most unheralded genius of SCORM is that it gave all the hard implementation work to just a few people (LMS developers) in order to make life easy for everybody else (content developers). I often say that SCORM placed the "burden of complexity" on the LMS developers. Considering these facts:

- There are perhaps thousands of LMS's in the world, but there are probably millions of SCOs.
- LMS's are systems with a long shelf life that tend to be in service for many years, SCOs often have a shelf-life of only a few years at most.
- LMS's are usually developed by large teams of talented software developers; SCOs are often written in PowerPoint

It only makes sense that the **LMS's should have to do the hard work and that SCORM should be nearly transparent to content developers**. Maintaining this complexity distribution is essential to the future of SCORM.

Graduated Complexity

In keeping with the idea that the burden of complexity should be placed on the LMS. It is also **important to allow content developers to bite off only as much as they can chew**. Creating a simple SCO should be very simple, even simpler than it is today. A content developer should not require any understanding of sequencing or any other advanced concepts in order to create a simple SCO. It is nice to have advanced functionality available to content developers, but only if they want it. This requirement speaks as much to the technical implementation of the specification as it does to the organization and layout of the specification's documentation. On the other hand, I believe it is hugely important to interoperability that LMS vendors be required to support all functionality (even the most advanced functionality) in order to be labeled as SCORM conformant. Without guaranteed LMS support, content is only interoperable when implemented at the level of the lowest common denominator. A partially implemented SCORM solution is worse than none at all.

Well Written Specification Documents

I distinctly remember how positive my first impression of SCORM was. While it wasn't like reading Tom Clancy, my initial experience reading the SCORM specifications was much better than I ever expected from a technical specification. The specifications included an appropriate balance of technical rigor with explanatory narrative (all of them except for sequencing and navigation that is...that was a tough document to understand). Other specifications I've read have either been too technicaly rigorous or not organized well enough to be easily comprehendable. **Excellent writing makes a specifcation more accessible to all and significantly increases its chances of being widely and correctly adopted.**

The Bathwater

Sequencing

I believe that the concept of including sequencing in SCORM is a good one, however **the current sequencing specification is a big black eye on the face of SCORM 2004**. While *it does work*, IMS Simple Sequencing is overly complex, fragile, difficult to implement and very hard to use. Unfortunately the benefit to using IMS Simple Sequencing does not exceed the cost, even for the content developer who carries the substantially lighter burden. The motivation for continuing the inclusion of sequencing in SCORM is a topic of discussion I will put forth in a seperate white paper. I will also publish the beginnings of a specification I call "Sensible Sequencing" for consideration in SCORM 2.0.

Sometimes Forgetting the Basics

Having a clear vision of the future to strive for is a great thing, however **it is important not forget the present when focusing on the future**. SCORM has sometimes focused too much on advanced capabilities and ignored simple, basic things that people need to do today. For instance, we have advanced sequencing capabilities, yet it is hard to provide good reporting on how a user did on a quiz and it is impossible for an instructor's evaluation of a learner's performance to count towards the learner's score.

Ignoring the Learner's Experience

SCORM has long focused on technical interoperability but ignored the learner experience. For good reason, SCORM was silent on how the LMS should present an interface to the learner. SCORM also includes some counter-intuitive behaviors such as resetting attempt data by default and it doesn't provide any mechanism for providing feedback to the user during sequencing. User experience is everything...especially these days. **If a system can't provide a friendly, intuitive user experience it is not going to be used.** SCORM must face this challenge head on or it will eventually be left behind.

"Reference Model"

This last comment will be controversial and, frankly, I'm not sure if I even agree with my own opinion, but it does merit discussion. The concept of creating SCORM as a reference model makes a lot of sense when trying to quickly create a specification, when you are trying to bring communities together, when you are trying to foster adoption and when you

are hoping to not rock too many boats. In other words, it made a whole lot of sense when ADL first conceived of SCORM. From a technical perspective however it has it's downfalls. Consider the addition of the reference to IMS Simple Sequencing. To harmonize Simple Sequencing with the existing set of specifications required a lot of square pegs to be inserted in round holes. It required duplicate data models to be maintained and it sometimes used different terminolgy to mean the same thing (objective normalized measure == score.scaled) and it sometimes used the same terminology to mean different things (can anybody define an objective?). In short, **referencing an existing specification can get sloppy**. Intellectual property concerns aside, are the technical disadvantages of a reference model worth the logistical advantages? ADL has long said that it is not in the business of creating specifications, I don't think that LETSI should so adamantly refuse to create its own intellectual property when needed.