Content Aggregation Issues and SCORM 2.0

By John Campbell and Don Holmes

Summary

Existing content aggregation practices in SCORM 2004 are not sufficient for the content size, quantity of files, and delivery requirements heading forward. With packages containing thousands of files, sizes growing beyond the 1GB range, and the requirement for staggered incremental deliveries from clients, we need to develop new methods of aggregating and delivering our courseware.

The Problem

The size of content is growing at a rapid pace. The “size” refers to both the number of files and also the physical (bytes) of the files in an aggregation. It is not uncommon for courseware to be in the hundreds of megabytes range containing thousands and thousands of files. We have produced courses over 1.6GB and some containing over 8000 files.

With packages being so large, importing into an LMS has become an issue. Even at 100 megabytes, we have seen LMSes that do not handle this load. And many systems are hosted with the expectation that content packages are uploaded over the internet. Workarounds typically involve a local administrator breaking up the delivered package into smaller components. A “shell” packages consisting of only the manifest and possibly metadata is created and imported into the LMS. After import, the remaining content is then copied to the file system into the course location. Though functional, this is not ideal. It requires knowledge of the proprietary file structures of the LMS and also is a dangerous, uncontrolled practice that could result in overwritten content or system files. New import mechanisms and requirements should be discussed to solve this problem.

New delivery requirements for US Army courseware consists of potentially four delivered versions for every course. These consist of one skeleton (minimal content) package to prove sequencing, one fully sequenced course package, one lesson (pre-test, knowledge lesson (KL), exercise, and post-test) package for credit, and individual packages for each KL. All but the skeleton package potentially can exist on the production schoolhouse LMS simultaneously. This means that some content resources will exists no less than three times on the server. Shared lesson content can exist many more times depending on how many lessons there are in a course. We have courses with around thirty lessons. This amounts to a tremendous level of resource duplication on the LMS.

Obviously, disk usage is problematic with this approach.   However, a more costly and complex issue is that of maintenance, versioning, and practicality. With multiple versions of courseware, it becomes increasingly difficult to update files, fix bugs, and keep track of what is where. As we develop new courseware, many time we have a desire to re-purpose content that is already developed. In some cases this means refreshing the old content to contain new doctrine, updated graphics, etc. It's currently not possible for multiple courses to easily share resources without duplication. One of our courses, MICCC, has over 40 SCORM 2004 content packages. Maintenance was a nightmare even before delivery had occurred.

Solutions?

We need the ability to package, reference, import, and share common resources across content packages. We have discussed the idea of breaking content packages into pieces. The main course package could consist of the manifest, metadata, and possibly minimal skeleton content. Resource packages could be created that consist of the remaining content. It seems that the concept of a "resource package" exists, but in practice it isn't being supported. The ability to reference external content in a standard way is needed. It may or may not exist on the same system. I think a more common use case for our development is content on the same system, but we should probably not limit it to that.

This solves many of the problems we are currently experiencing. It would be possible to create multiple, manageably sized packages. Though not ideal, this would allow most courses to be imported into an LMS over normal upload mechanisms. A bigger win, though, comes in dealing with multiple offerings that share resources. The content could be imported once, and only smaller "manifest" packages would need to be installed for each offering that used the resource. Having content that is no longer duplicated makes maintenance much easier for the developer and the LMS admin. It saves disk space too.

We hope that SCORM 2.0 can help to solve some of these issues. Content isn't getting any simpler. There will be more and more files to maintain and they aren't getting smaller anytime soon.