
SCORM, DITA, AND S1000D

A contribution to the SCORM TWG

Robby Robson, Eduworks
rrobson@eduworks.com
September 18, 2007

THREE CONTENT MODELS

SCORM, DITA and S1000D are three standards that are used to express and transport digital content. Each was designed to solve a different problem and therefore takes a different view of the way that content is structured and how systems process it.

SCORM – DESIGNED FOR INSTRUCTIONAL CONTENT

A SCORM 2004 package (more properly, a SCORM 2004 content aggregation) can be thought of as a container in which three things can be found:

1. *Resources*, which are to be delivered to the learner by the RTE.
2. *Metadata*, whose function is to define the *context* of the content in the package,
3. *Processing instructions*, whose function is to tell an RTE what to do with the content,

SCORM neither assumes nor provides any structure to resources, beyond the assumption (implicit in the API approach) that they are Web-deliverable and that some can issue the appropriate API calls. Put differently, SCORM resources are not *typed*. SCORM does not know or care if a resource is a quiz or an exposition or a simulation. Information about the type of a resource, to the extent it exists at all, can be encoded in metadata, and SCORM does not provide a structured XML format for encoding different types.

S1000D – DESIGNED FOR TECHNICAL MANUALS IN AVIATION

An S1000D DM contains two components

1. *Metadata*, whose function is administrative
2. *Content*, which is defined in an XML format

The S1000D *Data Module Code* (DMC) is also metadata. It defines the *type* of a DM, and each type of DM has its own XML format. The processing instructions are assumed to be apparent from the type, so an S1000D player knows what to do with every DM. S1000D content is also assumed to come from a *common source database* (CSDB). This means that S1000D content is assumed to be come from an *authoritative source*. The notions of a single authoritative source and associated metadata that encodes versioning and type information are common in the field of content management and not in any way specific to S1000D.

DITA – DESIGNED FOR HELP SCREENS AND PERFORMANCE SUPPORT SYSTEMS

DITA provides for two structures: *topics* and *maps*. Both are encoded in XML. A topic contains:

1. *Content*, which is structured XML and which contains a set of specified elements
2. *Metadata*, which is administrative in nature

Topics are *typed* much the same as in S1000D, but the content types are fewer, more general, and contain some common elements (e.g. a title and short description). Also, topics can contain other topics – indeed, the name, Darwin Information Typing Architecture, comes from the concepts of types and inheritance.

The second DITA concept is a *map*. A map is an XML-encoded association among objects, which could be DITA topics but could be anything else and are only required to be identified. Maps come in three flavors: Hierarchical, Ordered and Family (mutually linked).

As with S1000D, the system displaying DITA content is expected to know what to do with a particular type of topic or how to display content related by a map. The DITA community also is committed to the notion of authoritative sources but believes that this is best handled as part of a general content management strategy.

COMPARISON

SCORM views content as unstructured and provides processing instructions by grouping resources into units (SCO's) and telling a delivery system how to sequence the units. SCORM requires the content to communicate information (e.g. the status of objectives) to the delivery system that the system can use in making sequencing and navigation decisions. The advantages of this view are that the instructions can be complex and there are no restrictions on the type of content that can be used. One does not have to define a new type every time a new instructional structure is invented. The disadvantages are that the sequencing instructions are proscriptive (see the discussion of sequencing below) and the content is stored in a delivery format (e.g. HTML, PDF®, or Flash®). These are *serious* disadvantages that have negatively impacted reuse, interoperability and adoption.

S1000D views content as divided into types. The types must be further specialized for specific products and customers and retain some separation of content and presentation. The advantage of this view is that theoretically different viewers can display the same content in different ways. The CSDB and DMC code aid searching and enable changes in diagrams and procedures to automatically propagate to electronic technical manuals (or training). The disadvantages of S1000D are that it takes a very fine grained approach to types, in practice much of the content contained in an S1000D is in a delivery format, and there are no sequencing mechanisms, mechanisms for defining alternate versions of the same resource, or mechanisms for defining hierarchical content. In addition, it is based on a single model of content management that is not appropriate for all contexts. For example, the notion streaming content from a single source in nomadic and disconnected learning is questionable. Although S1000D Data Modules may be excellent sources of content for training and job aids, these disadvantages make S1000D an inappropriate choice for a general content interchange format for learning, education and training.

DITA views content as divided into general types that can be nested and aggregated via DITA maps. There are several clear advantages of the DITA approach. First, content can in practice be bound to

a delivery format at different stages. Second, DITA types are at a level of granularity that can be conveniently matched to *instructional objects* such as quizzes, modules, lessons (in one ontology) or facts, principles, practices, procedures, etc. (in a different ontology). Third, relationships among objects can be expressed. This does not directly provide the type of sequencing that is contained in Simple Sequencing, but I will argue below that *proscriptive sequencing is a bad idea anyway*.

SEQUENCING – TWO APPROACHES

In its simplest form the sequencing problem is that of deciding what content to show the user at runtime. Michael Priestly came up with the term *displayable unit*, and using this term the simplest sequencing problem can be phrased as the problem of deciding which displayable unit comes next.

In fact, sequencing is more complex. Embedded in the sequencing is the problem of determining how user interactions with current and past displayable units affect the decision process. At its most abstract level, the sequencing problem becomes a *process interaction* problem that is partially dependent on the content and partially dependent on context that is independent of the content. This is the difference between a *displayable* unit and (using AICC terminology) an *assignable* unit or (using SCORM terminology) a SCO.

Because of this complexity, the IMS and SCORM communities (and the AICC before) took the approach that sequencing is a programming problem. One writes an algorithm that every LET delivery system uses to sequence content. Being an algorithm implies that *every LET system will make the same decision given the same data*. Let us call this *prescriptive sequencing*. Underlying prescriptive sequencing is the assumption that every LET system *should* make the same sequencing decision given the same data.

Given the situated nature of learning and the unpredictable diversity in physical, cognitive and affective contexts in which sequencing takes place, this assumption either requires too much data to be tenable or is wrong.

A different approach to sequencing is to ascribe a modicum of intelligence to the designers of delivery systems and to provide them with the information their systems need to make reasonable and reliable decisions. This is the approach that instructional designers, academics, intelligent tutoring systems designers, authoring tool vendors, delivery system vendors, and the gaming and simulation communities have been clamoring for over the past ten years, and it is clearly superior in the LET context. Let us call this approach *context based sequencing*.

SCORM, S1000D, DITA AND CONTEXT-BASED SEQUENCING

SCORM cannot support context based sequencing because one of the vital pieces of context is the *type* of the content. A system simply cannot do the same thing with a quiz as with an exposition or a simulation. S1000D *does* enable context based sequencing, but only because the sequencing problem for IETM's is the simple version. Simply knowing the DMC and a bit of administrative metadata provides sufficient information to an S1000D player to make intelligent decisions, but the problem being handled by an S1000D player is relatively context-free.

DITA also is not built for context-based sequencing as is because there is insufficient contextual metadata provided with topics (or with maps), but *there is no reason that this metadata cannot be included*. Once the metadata is there, DITA becomes very close to ideal in that it provides the

appropriate level of typing, organization, inheritance and other features. For example, a DITA topic has a title and short description that can be used as alternative displays for small handheld devices, and DITA maps can be used to elegantly solve the “alternative resource” problem present in the existing SCORM Content Aggregation Model. Furthermore, there is nothing preventing a DITA map from simple sequencing code, AICC files, or some other expression of prescriptive sequencing to a collection of resources. This gives a path back to legacy content and a path forward that allows communities of practice to develop their own prescriptive sequencing approaches which may, in some cases, be desirable or more efficient than context-based sequencing.

CONCLUSIONS

Based on the high level considerations presented in this position paper, it makes sense to take a close look at the work being done by the DITA Learning Types group and to map out a complete DITA-based solution to packaging and sequencing for future versions of SCORM. The first step is to explore the edge and corner cases, including