EML and LMS related standard

Onderwijstechnologisch expertisecentrum OTEC
Open Universiteit Nederland

EML and LMS related standard

OTEC 2002/19

Colofon

Titel:	EML and LMS related standard
Auteurs:	Arjan Loeffen, Jocelyn Manderveld en Rob Koper
Projectleiding	Jocelyn Manderveld
Projectondersteuning	Mieke Haemers
Uitgifte	OTEC
Datum druk	11 June 2004

Onderwijstechnologisch expertisecentrum (OTEC)
Open Universiteit Nederland

EML and LMS related standard

Table of contents

Introduction

This section supplies an overview of several initiatives that may affect the design of a containing educational model. This concerns:

Instructional Management System (IMS)

Introduction

IMS Global Learning Consortium, Inc. (IMS) is developing and promoting open specifications for facilitating online distributed learning activities such as locating and using educational content, tracking learner progress, reporting learner performance, and exchanging student records between administrative systems

The IMS project defines the following separate specifications.

- *Learning Resource Meta-data* (p. 9). This is a specification of meta-data used to identify "learning resources".
- *Content packaging* (p. 13). A specification of how to assemble and distribute content in "packages".
- *Resource identifiers* (p. 17). This defines persistent, location independent resource identifiers.
- *Question & Test Interoperability (QTI)* (p. 19). This defines the structure of questions and tests, and the grouping of these.
- *Enterprise* (p. 33). This defines the way information on the learning 'enterprise' (instructional processes) is shared.
- *Learner information packaging* (p. 37). This specifies how to record and share information on the learner.
- *Reusable Competency Definitions* (p. 40). An information model for describing, referencing and exchanging definitions of competencies, primarily in the context of online and distributed learning.
- *Simple Sequencing* (p. 42). This defines how to associate sequencing information with *content packs* (p. 13) and its default behaviour.

Each specification has (or will have) at least three main parts:

- Information model — an abstract description of the area modelled
- Binding — binding to a particular language. For all specifications XML is the language of choice
- Best practice — explanation of how to apply the model.

IMS — Learning resources meta-data

General

The meta-data specifications create a uniform way for describing learning resources so that they can be more easily found (discovered), using meta-data aware search tools that reflect the unique needs of users in learning situations

A learning resource is any information- or tool object that can be reused in different environments.

Note 1	IMS does not define 'learning resource', so this definition is provisional.

In order to reuse the resources, several descriptions of the resource are required. The meta-data proposal offers descriptive 'layers' that are represented as consecutive XML elements in the XML binding document.

Components

A meta-data instance is a single specification, that is: a single XML document. This is a 'conforming LOM meta-data instance'. The components of a single meta-data specification are:

- General — Context independent features of the resource. Offer handles for search and retrieval.
- Lifecycle — Features of the lifecycle of the resource. Manage the (change history and) version of the resource
- Metametadata — Features of the description rather than the resource. Manage the meta-data entry itself.
- Technical — Technical aspects of the resource. The resource is assumed to be available in an electronic form; a hardback book is hard to describe using this scheme.
- Educational — Educational aspects of the resource. This includes the level of interactivity, for what user (type and level) the resource is intended, and such.
- Rights — Legal aspects of using the resource, i.e. costs and copyright.
- Relation — Possible typed relations with other resources.
- Annotation — Comments on the educational use of the resource.
- Classification — Some classification of the resource, based on a taxonomic path, keywords within the taxonomy.

These components are represented as subelements of the <lom> root element which is required for all meta-data instances. If the set of constructs is not sufficient, extensions can be made. The proposal suggests the use of alternative namespaces to identify these extensions; it does not provide a strategy.

Datatypes and taxonomies

The values of the meta-data components are defines by datatypes, field length, and/or taxonomic location. Datatypes are:

DateType — a combination of ISO8601 date and time specification and a description, which allows for a more verbose description of a moment in time.

LangStringType — a combination of a String and a language code taken from ISO639 (language) and ISO3166 (sublanguage). Example: "en-GB".

A taxonomy is a controlled vocabulary of terms and or phrases.

Note 2	For example, the IMS Meta-Data general.structure field (1.8) has a restricted vocabulary (i.e., Collection, Mixed, Linear, Hierarchical, Networked, Branched, Parceled, Atomic) from which the single field value can be drawn

The most typical form of a taxonomy is a hierarchy. A taxon is a node in the hierarchy. A path to a sub-taxon is a taxonpath, and therefore comprises a list of taxons. Each taxon has an id and entry. The id is an alphanumeric reference. Each taxon node in a taxonomy has a descriptive term, which is contained in the entry. Each classification instance may contain multiple taxonpaths. There is a source specification that identifies the base scheme of the taxonomy, for example the LCC (Library of Congress Classification):

```
source: LCC
        taxon
                id: B
                entry: PHILOSOPHY. PSYCHOLOGY. RELIGION
        taxon
                id: F
                entry: Psychology
        taxon
                id: 180
                entry: Experimental psychology
```

IMS does not endorse any particular taxonomy or set of taxonomies. There are however several such vocabularies and taxonomies suggested by the meta-data "best practices" part of the recommendation.

Vcards

The IMS spec introduces visiting cards for identifying persons and organizations. The <vcard> element is a non-XML specification and is specified by http://www.imc.org/pdi/.

Relations with other standards

General

Meta-data is not bound to a particular language. It can for example be represented in HTML as follows:

```
<META name="description" content="The IMS meta-data system.">
<META name="keywords" CONTENT="IMS, Metadata, Meta Data, meta-data,
fields, online, on-line, on line, knowledge, distributed, instruction,
education, learning">
```

Relation with Dublin Core

IMS meta-data covers the Dublin Core and a mapping is provided within the specification.

Relation with SCORM

Within the SCORM, the SCORM Meta-data Application Profiles are specializations of the IMS Learning Resource Meta-data Specification Version 1.1 The SCORM imposes additional constraints on the application of the specification.

The SCORM Version 1.2 contains three meta-data application profiles: 1/ Resource (SCORM Version 1.1 Raw Media Meta-data) 2/ SCO (SCORM Version 1.1 Content Meta-data) 3/ Content Aggregation (SCORM Version 1.1 Course Meta-data)

Integration with EML

EML provides content that can be reused and exchanged, much in the sense of the IMS, and therefore requires meta-data. EML meta-data is represented using <metadata> and attributes on elements.

The following mappings from EML to IMS van be envisioned:

— `Metadata/Title` >> *general/title*

— `Metadata/Subtitle` >> *None*

— `Metadata/Creator` >> *lifecycle/contribute* with *role=author*

— `Metadata/Description` >> *general/description*

— `Metadata/Keywords` >> *general/keywords*

— `Metadata/Copyright` >> *rights/copyright*

— `Metadata/Study-load` >> *None*

— `Metadata/Extra-meta` — *This is a wrapper element and has no special meaning.*

— `Metadata/Extra-meta/Object-type` >> *None*

— `Metadata/Extra-meta/Supplied` >> *None*

— `Metadata/Extra-meta/Contributor` >> *lifecycle/contribute* with *not(role=author)*

— `Metadata/Extra-meta/History` >> *Partly within lifecycle; this doesn't provide means to descibe what is contributed*

— `Metadata/Extra-meta/Status` >> *lifecycle/status*

— `Metadata/Extra-meta/Creation-date` >> *May be mapped onto lifecycle/contribute/date for contributor role=author*

— `Metadata/Extra-meta/Date-last-change` >> *May be mapped onto lifecycle/contribute/date for any contributor role*

— `Metadata/Extra-meta/Min-completion-time` >> *None*

— `Metadata/Extra-meta/Max-completion-time` >> *None*

— `Metadata/Extra-meta/Meta` >> *None*

— `Metadata/Comment` >> *None*

— `.[@version]` >> *lifecycle/version*

Some technical meta-data is part of the IML application.

Many of the unmapped specifications can be expressed using the <relation> element. This is however non-standard.

References

See *[IMS-LR-INFO-1.1]*, *[IMS-MD-BEST-1.1]*, *[IMS-MD-BEST-1.2]*

IMS — Content packaging

General

The IMS Content Packaging Specification describes how to collect reusable content objects such that they are useful in a variety of learning systems. It describes data structures that are used to provide interoperability of Internet-based content with content creation tools, Learning Management Systems (LMS), and run-time environments. The objective of the IMS Content Packaging Specification is to define a standardized set of structures that can be used to exchange content. The scope of the IMS Content Packaging Specification is on defining interoperability between systems that wish to import, export, aggregate, and disaggregate packages of learning content.

Components

An IMS Content Package contains two major components:

– 1 a (required) special XML document describing the content organization and resources of the package. The special file is called the Manifest file (`imsmanifest.xml`) because package content and organization is described in the context of manifests.

– 2 the physical files referenced in the Manifest.

This is shown in the following figure, taken from *[ims-cp-best-1.1.2]*.

Figure 1 - IMS Content Package Components

IMS Content Package shown on the left, and related IMS specifications on the right.

The CP defines the following components:

```
Package
  Manifest
    Metadata?
      Schema
      Schemaversion
      #any
    Organizations
      Organization*
        Title?
        Item*
          Title?
          Item*
          metadata?
          #any
        Metadata?
        #any
      #any
    Resources
      Resource*
        Metadata?
        File*
          Metadata?
          #any
        Dependency
          #any
    Manifest*
    #any
  Physical resources...
```

- A *package* represents a unit of usable (and reusable) content. It is typically stored in a folder and passed on as a zip file. This conatins the physical resources and the manifest file
- A *manifest* is a description in XML of the resources comprising meaningful instruction. Submanifests may occur, in which case these are interpreted in the context of the outer manifest. Submanifests are intended to be independent of their containing manifests, just like boxes within a box are still capable of holding stuff on their own.
- Within the manifest a *metadata* section is available and may, for example be used to merge the resource descriptions into a catalog. A bias exists toward the IMS meta-data but is not part of the specification.
- A Manifest may also contain zero or more static ways of organizing the instructional resources for presentation using *organizations*. The organization consists of items that 1) reference a resource, 2) reference a manifest, and/or 3) hold one or more subitems. More than one organization may be supplied which iare deemed to be equivalent in learning outcomes; a default organization is required.

  The *Simple Sequencing* (p. 42) initiative introduces and alternative approach here; a single organization is expressed using a SS specification.

- The *resources* component can describe external resources, as well as the physical files that the package consists of. These files may be media files, text files, assessment objects or other pieces of data in electronic form.

Relations with other standards

IMS CP has been extended for *SCORM conformance* (p. 58) using ADL specific information elements. ADL specific elements necessary for packaging SCORM content are defined within an ADL namespace.

Integration with EML

EML can be expressed in part using content packages. There are two alternative setups.

Setup 1 — http://file:///E:\p-sal\OUH007\work\eml20\sample3\imsmanifest4.xml

Manifests and submanifests can cover the aspects of a UOL specification:

- Metadata — As far as the meta-data concerns the unit of study itself, and not the content package, the meta-data can be recorded using a *IMS Learning Resource Meta-data* (p. 9) specification.
- Role information — Resources are preferrably *IMS Learner Information profiles* (p. 37).
- Prerequisites and objectives — Resources are preferrably *IMS Reusable Competency Descriptions* (p. 40).
- Content/Environment — Sets of resources holding or describing objects in the environment can be expressed using submanifests. This may include *IMS QTI assessments, sections or items* (p. 19)
- Content/activity — Activities are described using manifests. These reference resources holding the activity description. They may also reference or hold environments as expressed using submanifests.
- Method/activity-structure — This can be mapped onto 1) subitems within the organization section of the manifest, or 2) a *simple sequencing specification* (p. 42) referenced from within the organization. The second approach is preferred as sequencing logic is best represented there.
- Method/play — There is no equivalent to this structure in IMS. It seems best to define a separate resource for such specifications.
- Method/conditions — There is no equivalent to this structure in IMS. It seems best to define a separate resource for such specifications.

Setup 2 — file:///E:\p-sal\OUH007\work\eml20\sample2\examplepack\imsmanifest.xml

Note that setup 1 does not comply with the rule of independent submanifests. So, we may equally define packages in stead of manifests that hold part of a complete UOL. This would result in a package with sub-packages for the aspects mentioned above. Each sub-package holds a single manifest which identifies the resources:

```
-------------------------------------------------------------------------
manifest           - for complete OUL
  resource         - student description file
  resource         - activity pack for the student.
                     This is a directory with an imsmanifest:
    ---------------------------------------------------------------------
    manifest       - for activity pack
     organization  - for activity pack
       item        - for activity
       item        - environment for the activity
     resource      - activity description
     resource      - environment.
                     This is a directory with an imsmanifest:
```

```
      ------------------------------------------------------------------
      manifest    - for the environment
        resource  - article itself
        resource  - book itself
      ------------------------------------------------------------------
    ----------------------------------------------------------------------
  resource         - assessment for the UOL
------------------------------------------------------------------------
```

References

See *[IMS-CP-BIND-1.1.2]*, *[IMS-CP-BEST-1.1.2]*, *[IMS-CP-INFO-1.1.2]*.

IMS — Resource Identifiers

General

There is a need for persistent, location-independent, resource identifiers across multiple IMS specifications. A persistent, location-independent, resource identifier is defined as an instance of a data type or data format associated with an item which provides a persistent, immutable label with global scope and indefinite lifetime

Components

The RI handbook defines 1) Globally Unique Identifiers (GUIDs) and 2) Universally Unique Identifiers (UUIDs). The IETF Uniform Resource Name (URN) is proposed as a candidate for the IMS UID scheme. The URN is a calculated string and is not registered. The URN form is: *URN:[nid]:[nss]* where [nid] is the Namespace Identifier and [nss] is the Namespace Specific String. Example: *URN:ISBN:0-395-36341-1*

The following application strategy is followed.

— Existing URN schemes should be used for all objects which have a formal URN scheme.

— Large organizations should obtain their own NID. This includes the IMS itself. IMS develops a NSS scheme for associated organisations.

— An organization receiving an object with a URN will determine if it trusts the uniqueness of the URN; it must preserve the URN and upon export or transmission, it must label the object with the original URN.

— Two objects with the same URN are identical. Lexical equivalence of URNs is based on their encoding scheme.

— An object can only have 1 URN.

An example of an IMS GUID would be (URN within the IMS NID using a sourced NSS without a scheme):

```
URN:IMS-PLIRID-VO:DUNS:05-218-4116::6ba7b8149dad11d180b400c04fd430c8
```

Relations with other standards

The IEEE Learning Object Meta-data (LOM) Specification has placeholders to store pan-organizational identifiers.

IMS Content Packaging has identifier fields which are recommended to be unique across all organizations. The IMS Competency Working Group intends to use unique identifiers as the fundamental identification mechanism for reusable competency definitions. There are numerous existing and proposed repositories for competency, skills, and outcomes definitions. If this specification is to have any practical effect, it will be necessary to reference these definitions in a unique and machine-retrievable way. Note that none of the IMS specifications identify an adequate scheme to define the data type or representation for such a unique identifier.

Note 3	Note that a common IMS solution for persistent identiers has not yet been identified. Note also that the Handbook, though it states that it is the final version, is actually work in progress.

The handbook references existing identification schemes such as ISBN, DOI, DUNS, MAC, URN.

Integration with EML

The UID strategy can be applied where IMS integration is foreseen. For example, in the QTI modules and content packaging. Integration of EML within the IMS strategy requires EML to align with these directions. Therefore, the EML atribute `Wwuid=` will have a URN value as described.

References

See *[IMS-RI-HB–1.0]*.

IMS — Question & Test

General

The IMS Question & Test Specification addresses the need to share test items and other assessment tools across different systems. It builds upon an envisioned workflow which includes authors, assessors, candidates, tutors and so on. The question and test system itself holds an assessment engine which reads a repository of questions and tests, and information on eligibility and performances. It then evaluates the responses in terms of producing scores and feedback.

There is a clear bias of the QTI proposal toward the XML binding.

Figure 2 - IMS representation of the assesment system.

Taken from *[IMS-QTI-INFO-1.1]*

Components

There are three kinds of reusable data objects in the QTI model, representing *assessments, sections, items (ASI)*. The ASI objects all have meta-data in accordance with IMS Meta-data specifications.

- *ASSESSMENT Data Object* — A complete *assessment*, for example: a test. This may be a complex collection of several parts, known as Sections and Items. An Assessment object contains all of the information to make the use of individual Items meaningful. This means that, apart from the

sections themselves, it includes the relationships between the sections, the group evaluation processing and the corresponding feedback.

The user interacting with an assessment is known as the *Participant*.

Figure 3 - Graphical representation of `assessment` object

— The assessment meta-data is defined in terms of 1) general QTI meta-data fields, that have no specific internal structure, 2) single specifications for typing and processing the assessment (not for retrieval).

— Objectives and rubric are textually defined (materials and text flows).

— Assessment control holds control flags for presenting the assessment, using switches for showing feedback, hints and solutions. It also defines for whom this material is available ('view', such as All, Administrator, Assessor, Candidate)

— Assessment processing is implemented using 1) variable declaration structure and 2) a score expression language that reads and writes these variables. The language is completely score variable based, for example: "If variable A has value V, set variable B, and show feedback F".

— Assessment feedback is material shown on completion of the assessment.

— The assessment specification is completed by an ordered set of sections.

- *SECTION data object* — A *Section* object contains all of the information to meaningfully group together Items. Apart from the Items it includes the relationships between the Items and the selection criteria of the Items.

Figure 4 - Graphical representation of `section` object

— Section meta-data is 1) general QTI meta-data, followed by 2) processing meta-data for this section (number of items, selection sequence etc).

— Objectives and rubric, see elsewhere.

— Secton control see elsewhere

— Sections are embedded.

— Section processing is the processing of accumulated responses and scores of the embedded sections. Again the score expression language is used.

— Feedback, see elsewhere.

— The section specification is completed by an ordered set of items.

- *ITEM Data Object* — An *Item* object contains information on how to present a question and its subsequent processing to the user. The structure of the Item includes one or more actual questions and responses as well as its presentation format, the range of possible responses, the ways in which the responses are to be processed, and the possible solutions and hints to the Item.

The item is of a particular *response-type*, i.e. the item is typed in accordance with the kind of response required, such as a *logical ID* (for MC questions) or a *string* (for fill-in-the-blanks).

Figure 5 - Graphical representation of `item` object

— Item meta-data is composed like the assessment and section meta-data. It includes directions like how many hints are permitted, level of difficulty, rendering type etc.

— Objectives, see elsewhere.

— Item control, see elsewhere

— (Item) rubric holds material for specific views, see assessment.

— Presentation, which is a container for response types and rendering forms. A simple presentation has just one response type.

— Response processing is defined in terms of outcomes (declare and set variables) and conditions as described for assessment and section.

— Item feedback is textual by nature, and may include the solution (for self-tests for example) and hints.

Reponse types

An author must determine for a question the *response type* and map this onto a *rendering type*. The response type can be described using three orthogonal feature sets: response structure, multiplicity and timing.

Figure 6 - QTI response types

The following levels are inherent to the QTI response type (see *figure* (p. 23)):

- Basic response — One single type of response is expected. For example: a string.
- Composite response — Several grouped responses. For example: three answers that are thematically related.
- Time dependent — The time used to generate the response is part of the response handling. For example, when 'slow' offer different question.
- Time independent — Time is of no importance to the reponse handler.
- Single response — A single response is returned for the item, whether or not the item is multiple response.
- Multiple response — Several responses are returned for each item.
- Ordered response — Multiple, where order is significant.

Response types denote the kinds of results of processing a question. This is any of Logical ID (LID), XY coordinate (XY), String (STR), Numerical (NUM) or Logical Group (GRP). Per response type several rendering types can be chosen. This is expressed in the schema as shown in *separate figure* (p. 23) (string example)

Figure 7 - Graphical representation of `reponse_str` object

The response type values are also associated with a *duration* and an *identifier*. The duration element is the period between the item being triggered and the response(s) being supplied. The identifier ensures that the scoring attributes can be correlated to the generating response.

Rendering types

Each of the reponse types may be based on a item rendered in a particular way. For example, a MC question may be rendered as a list of, say, 4 items, or as a hotspot representation. Render types are part of the IMS specification. These rendering types all produce a response type.

Note 4	Note that rendering 'subtypes' can be defined. As stated, a MC question can be rendered in several ways. MC is still considered to be a rendering format.

The following rendering types are defined (with associated response types).

- True/false LID
- Multiple choice LID
- Multiple response LID
- Image hotspot XY
- Fill-in-blanks STR NUM
- Select text STR
- Slider LID NUM
- Drag object LID GRP
- Drag target LID GRP
- Order objects LID XY
- Match items LID GRP
- Connect the points LID XY
- Short answer STR (not in response types overview)
- Essay STR (not in response types overview)

Object model

The above constructs (assessments, sections, items, responses), are part of the QTI object model. This also includes objects for dealing with the outcomes. These objects are all reflected in the XML binding (through elements and attributes). The object model includes

- Assessment / <assessment> — as described.
- Section / <section> — as described.
- Item / <item> — as described
- Activity selection — selection of the next activity determined by the progress and results obtained upto the moment of activity selection
- Accumulation process — the reconciliation of all the evaluation outputs to produce an overall Assessment/Section evaluation
- Scoring weights — the scoring weights that are to be assigned to the results output from the response processing
- Response processing — the processing and evaluation of the user responses
- Presentation — the rendering of the content and the possible responses
- Examinee record — the set of collated results that are output from the complete process. This is a 'life-long' record in that it contains the historical progress of the individual
- Outcomes — the set of outcomes that are to be evaluated by the response processing object. These determine the scoring metrics to be applied to the response evaluations
- Reponse — the responses that are supplied by the user of the Items i.e. the input user selections

- Flow — the underlying presentation structure that defines the block relationship between the different *material* components
- Material — the content that is to be displayed

  The content that is part of a question rendering can be text, image, audio or video, or any combination of these.

Textual structure

Much of the content is displayed as text. The QTI specifies the following constructs for representing textual material:

- Mime type — *[RFC1521]*
- Character set — *[ISO10646]*
- Language — specified through *xml:lang*=, see *[XML]*
- White space handling — specified through *xml:space*=, see *[XML]*
- Emphasis — specified through <*matemtext*>.
- Paragraphs — specified through flow structures, and through `<matbreak>`.

The specification states that other more sophisticated textual structures are outside its scope.

Meta-data

Meta-data specifications are either

- standard IMS meta-data as used in packaging content. The *[ims-qti-info-1.1]* specifies how to apply this meta-data specification for ASI instances.
- QTI specific meta-data fields, ie.. assessment meta-data, section meta-data, item meta-data.

  Not yet available

- A specific QTI vocabulary used in any meta-data specification.

  Not yet available

Conformance

Vendors claiming conformance must provide a "Conformance Summary", detailing their level of conformance to the QTI specification. This is defined on three levels.

Publish – (write) this implies that the system produces an instance that contains QTI elements. Note that QTI presupposes the bidning to be in XML.

Accept – (read) the system is able to read QTI elements and process tjhem in accordance with the QTI specification.

Repackage – this is the ability to import QTI XML instances from one or more sources and to create a new instance that combines the imported information.

Examples

We provide some XML code examples below

An Item: a True/false question with CHOICE rendering, without reponse processing:

```
<?xml version="1.0" standalone="no"?>
<!-- Author:  Colin Smythe    -->
<!-- Date:    30th January, 2001   -->
<!-- Version 1.0 Compliant Example: BasicExample001a -->
<!-- Basic Example  -->
<!-- True/false with CHOICE rendering (w/o reponse processing). -->
<questestinterop>
  <qticomment>
     This is a simple True/False multiple choice example.
     The rendering is a standard radio button style.
     No response processing is incorporated.
  </qticomment>
  <item ident="IMS_V01_I_trfl_i_001">
     <presentation label="BasicExample001a">
     <material>
       <mattext>Paris is the Capital of France</mattext>
     </material>
     <response_lid ident="TF01" rcardinality="Single" rtiming="No">
       <render_choice>
         <response_label ident="A">
            <material><mattext>Agree</mattext></material>
         </response_label>
         <response_label ident="D">
            <material><mattext>Disagree</mattext></material>
         </response_label>
         </render_choice>
     </response_lid>
  </presentation>
  </item>
</questestinterop>
```

Colin Smythe, 9th March, 2001

An Item: Multiple-choice (text) with CHOICE rendering and response processing:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE questestinterop SYSTEM "IMS_QTIv1p0.DTD">
<!-- Author:  Colin Smythe  -->
<!-- Date:      30th January, 2001  -->
<!-- Version 1.0 Compliant Example: BasicExample002a  -->
<!-- Multiple-choice (text) with CHOICE rendering and response
     processing -->
<questestinterop>
 <qticomment>
       This is a simple multiple choice example.
       The rendering is a standard radio button style.
       Response processing is incorporated.
   </qticomment>
 <item title="Standard Multiple Choice Item" ident="IMS_V01_I_mchc_ir_001">
  <presentation label="BasicExample002a">
   <material>
   <mattext>Which one of the listed standards committees is
                 responsible for developing the token ring
                 specification ?
             </mattext>
   </material>
   <response_lid ident="MCb_01" rcardinality="Single" rtiming="No">
   <render_choice shuffle="Yes">
     <response_label ident="A">
     <material>
      <mattext>IEEE 802.3</mattext>
     </material>
     </response_label>
     <response_label ident="B">
     <material>
      <mattext>IEEE 802.5</mattext>
     </material>
     </response_label>
     <response_label ident="C">
```

```
    <material>
     <mattext>IEEE 802.6</mattext>
    </material>
    </response_label>
    <response_label ident="D">
    <material>
     <mattext>IEEE 802.11</mattext>
    </material>
    </response_label>
    <response_label ident="E" rshuffle="No">
    <material>
     <mattext>None of the above.</mattext>
    </material>
    </response_label>
  </render_choice>
  </response_lid>
 </presentation>
 <resprocessing>
  <outcomes>
  <decvar vartype="Integer" defaultval="0"/>
  </outcomes>
  <respcondition title="Correct">
  <conditionvar>
    <varequal respident="MCb_01">A</varequal>
  </conditionvar>
  <setvar action="Set">1</setvar>
  <displayfeedback feedbacktype="Response" linkrefid="Correct"/>
  </respcondition>
 </resprocessing>
 <itemfeedback ident="Correct" view="Candidate">
  <material>
  <mattext>Yes, you are right.</mattext>
  </material>
 </itemfeedback>
</item>
</questestinterop>
```

Colin Smythe, 9th March, 2001

Example of a single Section with two multiple-choice (text) questions:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE questestinterop SYSTEM "IMS_QTIv1p0.dtd">
<!-- Date:    30th January, 2001           -->
<!-- Version 1.0 Compliant Example: AdvExample201     -->
<!-- Advanced Example with S(1) I(2)           -->
<questestinterop>
  <qticomment>
    This example consists of two Sections.
  </qticomment>
  <section title="European Capitals" ident="IMS_V01_S_Example201">
    <objectives view="Candidate">
      <material>
        <mattext>To assess your knowledge of the capital cities in
                 Europe.</mattext>
      </material>
    </objectives>
    <objectives view="Tutor">
      <material>
        <mattext>To ensure that the student knows the difference
        between the Capital cities of France, UK, Germany, Spain and
        Italy.
        </mattext>
      </material>
    </objectives>
    <item title="Capital of France" ident="I01" maxattempts="6">
      <!-- item structure goes here -->
    </item>
  </section>
  <section title="European Rivers" ident="IMS_V01_S_Example202">
    <objectives view="Candidate">
      <material>
        <mattext>To assess your knowledge of the rivers in
```

```
        Europe.</mattext>
      </material>
    </objectives>
    <objectives view="Assessor">
      <material>
        <mattext>Questions on the rivers in Germany, Spain, Italy
        and France.</mattext>
      </material>
    </objectives>
    <item title="Rivers in France question" ident="I02">
      <!-- item structure goes here -->
    </item>
  </section>
</questestinterop>
```

Colin Smythe, 9th March, 2001

Example of acomplex single Assessment containing several multiple-choice questions:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE questestinterop SYSTEM "IMS_QTIv1p0.dtd">
<!-- Date:    30th January, 2001           -->
<!-- Version 1.0 Compliant Example: AdvExample     -->
<!-- Advanced Example with A(1)S(1)I(2)S(1)I(1)     -->
<questestinterop>
  <assessment title="European Geography" ident="A01">
    <qticomment>
      A Complex Assessment example.
    </qticomment>
    <objectives view="Candidate">
      <material>
        <mattext>To test your knowledge of European geography.</mattext>
      </material>
    </objectives>
    <objectives view="Assessor">
      <material>
        <mattext>Tests the candidate's knowledge of European
                    geography.</mattext>
      </material>
    </objectives>
    <assessprocessing>
      <qticomment>
        Processing of the final accumulated assessment.
      </qticomment>
      <scores>
        <decvar/>
      </scores>
      <scorecondition>
        <conditionvar>
           <varlte respident="SCORE">9</varlte>
        </conditionvar>
        <displayfeedback feedbacktype="Response" linkrefid="A01_FDB01"/>
      </scorecondition>
      <scorecondition>
        <conditionvar>
           <vargte respident="SCORE">10</vargte>
        </conditionvar>
        <displayfeedback feedbacktype="Response" linkrefid="A01_FDB02"/>
      </scorecondition>
    </assessprocessing>
    <assessfeedback title="Failed" ident="A01_FDB01">
      <material>
        <mattext>You failed the test.</mattext>
      </material>
    </assessfeedback>
    <assessfeedback title="Passed" ident="A01_FDB02">
      <material>
        <mattext>You passed the test.</mattext>
      </material>
    </assessfeedback>
    <section title="European Capitals" ident="S01">
      <objectives view="Candidate">
```

```
        <material>
           <mattext>To assess your knowledge of the capital cities
                    in Europe.</mattext>
        </material>
      </objectives>
      <objectives view="Tutor">
        <material>
           <mattext>To ensure that the student knows the difference
           between the Capital cities of France, UK, Germany, Spain
           and Italy.
           </mattext>
        </material>
      </objectives>
      <item title="Capital of France" ident="I01" maxattempts="6">
        <!-- item stucture goes here -->
      </item>
    </section>
    <section title="European Rivers" ident="SO2">
      <objectives view="Candidate">
        <material>
           <mattext>To assess your knowledge of the rivers in
                    Europe.</mattext>
        </material>
      </objectives>
      <objectives view="Assessor">
        <material>
           <mattext>Questions on rivers in Germany, Spain, Italy and
                            France.</mattext>
        </material>
      </objectives>
      <item title="Rivers in France question" ident="I02">
        <!-- item stucture goes here -->
      </item>
      <item title="Rivers in Germany" ident="I03">
        <!-- item stucture goes here -->
      </item>
    </section>
  </assessment>
</questestinterop>
```

Colin Smythe, 9th March, 2001

Relations with other standards

The IMS QTI will be presented to Aviation Industry CBT Committee, which works with ADL specifications.

ISO has a JTC1/SC36 — learning technology. This forum does however not yet define standards in the QTI field.

IMS is actually the only formal QTI specification body.

Integration with EML

The following mappings can be envisioned.

- All EML *question* elements can be mapped onto QTI *items*, very likely by automated transformation. EML is however less powerfull and will benefit from the introduction of QTI items. Conformance however will be only partial unless the complete QTI strategy is adapted.

  We provide the mappings below.

  — `Multiple-choice-question` >> *render_choice*

— `True-false-question` >>*render_choice*

— `Multiple-response-question` >> *render_choice* (within <*response_lid rcardinality='Multiple'*>)

— `Sequence-question` >> *ims_render_object* with an ordered response cardinality

Example of an order object question (textual)

```
<presentation label="BasicExample009a">
  <material>
   <mattext>What is the correct order for the days of the week ?</mattext>
  </material>
  <response_lid ident="OB01" rcardinality="Ordered" rtiming="No">
    <render_extension>
     <ims_render_object shuffle="No" orientation="Row">
      <response_label ident="A">
       <material><mattext>Monday</mattext></material>
      </response_label>
      <response_label ident="B">
       <material><mattext>Tuesday</mattext></material>
      </response_label>
      <response_label ident="C">
       <material><mattext>Friday</mattext></material>
      </response_label>
      <response_label ident="D">
       <material><mattext>Thursday</mattext></material>
      </response_label>
      <response_label ident="E">
       <material><mattext>Wednesday</mattext></material>
      </response_label>
     </ims_render_object>
    </render_extension>
  </response_lid>
</presentation>
```

Colin Smythe, 9th March, 2001

— `Matching-question` >> *match item*?

— `Short-answer-question` >> *render_fib*.

The short answer question is a combination of a pattern matched against the reponse and the rendering by way of a *fill-in-the-blank*.

```
<?xml version="1.0" standalone="no"?>
<questestinterop>
  <qticomment>
     This is a standard fill-in-blank (text) example.
  </qticomment>
  <item title="Standard FIB string Item" ident="IMS_V01_I_fibs_ir_001">
     <presentation label="BasicExample012b">
       <flow>
         <material>
            <mattext>Complete the sequence: </mattext>
         </material>
         <flow>
            <material>
              <mattext>Winter, Spring, Summer, </mattext>
            </material>
            <response_str ident="FIB01" rcardinality="Single"
                          rtiming="No">
              <render_fib fibtype="String" prompt="Dashline"
                          maxchars="6">
```

```
                <response_label ident="A"/>
                <material>
                  <mattext>.</mattext>
                </material>
              </render_fib>
            </response_str>
          </flow>
        </flow>
      </presentation>
      <resprocessing>
        <outcomes>
          <decvar varname="FIBSCORE" vartype="Integer"
                  defaultval="0"/>
        </outcomes>
        <respcondition>
          <qticomment>Scoring for the correct answer.</qticomment>
          <conditionvar>
             <varequal respident="FIB01" case="Yes">Autumn</varequal>
          </conditionvar>
          <setvar action="Add" varname="FIBSCORE">1</setvar>
          <displayfeedback feedbacktype="Response"
                           linkrefid="Correct"/>
        </respcondition>
      </resprocessing>
      <itemfeedback ident="Correct" view="Candidate">
        <flow_mat>
          <material><mattext>Yes, the season of
                        Autumn.</mattext></material>
        </flow_mat>
      </itemfeedback>
    </item>
</questestinterop>
```

Colin Smythe, 9th March, 2001

— `Question-answer-question` >> *No mapping?*

— `Prompt` >> *no mapping?*

Many question types and constructs in IMS cannot be expressed using EML constructs. Example: ordered responses, sliders etc.

EML does not separate the response types *described earlier* (p. 23).

- — *interactions* >> *Section*
- Structure of the questions (common constructs)

— *Metadata* >> mapped onto item metadata. *Not yet specified.* Note that in QTI items have objectives, in EML they don't.

— *Question* | *Statement* >> *material*

— *Hint* >> *hint*

— *Feedback* >> *itemfeedback*

Example hint and feedback

```
<itemfeedback ident="I01_IFBK03" view="All">
  <hint feedbackstyle="Multilevel">
  <hintmaterial>
   <flow_mat>
```

```
    <material>
     <mattext>One of the choices is not in Europe.</mattext>
    </material>
    </flow_mat>
   </hintmaterial>
   <hintmaterial>
    <flow_mat>
    <material>
     <mattext>Berlin is the Capital of Germany.</mattext>
    </material>
    </flow_mat>
   </hintmaterial>
   <hintmaterial>
    <flow_mat>
    <material>
     <mattext>The Eiffel tower is in the Capital of France.</mattext>
    </material>
    </flow_mat>
   </hintmaterial>
   </hint>
</itemfeedback>
```

Colin Smythe, 9th March, 2001

— *Score >> scores/decvar*.

Note 5	Feedback and scores can be provided conditionally; QTI provides an expression language for this within *scorecondition*:
	
	`<scorecondition title="Assessment Summary Scoring">`
	`  <conditionvar>`
	`  <varequal respident="SECTIONSCORE">5</varequal>`
	`  </conditionvar>`
	`  <setvar action="Add" varname="ASSESSSCORE">100</setvar>`
	`  <displayfeedback feedbacktype="Response" linkrefid="AssesSummary"/>`
	`</scorecondition>`

- In general, the QTI specification can be integrated with EML on all ASI levels. We can envision the EML questionnaire by QTI assessment, interactions module QTI section, and individual questions by QTI items. Based an an integrated system for handling the dossiers (mapping of variable declaration as in QTI onto properties as in EML) it is possible manage these domains separately and unify the handling of the EML workflow (activities) with the outcomes of the ASI parts. Note that EML does not claim to model complete assessments.
- It is to a limited extent possible to express IMS QTI questions in EML. A repository of QTI compliant questions can be transformed to EML questioons but only with much loss of function. Therefore we consider this option not viable.

References

See *[IMS-QTI-INFO-1.01]*, *[IMS-QTI-BEST-1.01]*, *[IMS-QTI-INFO-1.1]*, *[IMS-QTI-BEST-1.1]*, *[IMS-QTI-BIND-1.1]*.

IMS — Enterprise

General

The IMS Enterprise Specification is aimed at administrative applications and services that need to share data about learners, courses, performance, etc., across platforms, operating systems, user interfaces, and so on. It defines a standardized set of structures that can be used to exchange data between Learning Management systems (LMS) and systems mentioned below. These structures provide the basis for standardized data bindings that allow software developers and implementers to create Instructional Management processes that interoperate across systems developed independently by various software developers.

The targeted systems are:

- Human Resource Systems — track skills and competencies and define eligibility for training programs.
- Student Administration Systems — support the functions of course catalog management, class scheduling, academic program registration, class enrollment, attendance tracking, grade book functions, grading, and many other education functions.
- Training Administration Systems — support course administration, course enrollment, and course completion functions for work force training.
- Library Management Systems — track library patrons, manage collections of physical and electronic learning objects, and manage and track access to these materials.

Note that this is confined to the same enterprise or organisation; it does not cross enterprises.

Components

The process components that the enterprise model focusses on are:

- Profile management — personal information.
- Group management — group related information, including class creation and scheduling.
- Enrollment management — information on registering and assignment of instructors.
- Result processing — dealing with the outcomes of the learning processes.

The specification builds upon groups and persons, and persons in groups (members).

- Group — This object contains elements describing a group of interest to the Learning Management environment. The most common is a Course Instance, but they may also include Training Programs, Academic Programs, Course sub-groups, clubs, etc. A group can also have any number of relationships with other groups. Properties include type of group, description, organisation, time frame in which this group is active, enrollment info, common contact info, relationships with other groups.
- Person — This data object contains elements describing an individual of interest to the Learning Management environment. This includes user's ID, name, demographical info, contact info, photo.

- Group Member — This data object contains elements describing the membership of a person or group within a group. Group members may be instructors, learners, content developers, members, managers, mentors, or administrators. This includes member ID's, role of that member within the group, subroles, status, timeframe for this member, final result of membership, and member contact info.

Example enterprise specification

This is an example of 2 persons-- 1 group and a group membership for the group consisting of the 2 persons. (1 a student and the other the professor).

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ENTERPRISE SYSTEM "IMS-EP01.dtd">
<ENTERPRISE>
  <PROPERTIES>
    <DATASOURCE>California State University San Marcos</DATASOURCE>
    <TARGET>Computing and Telecommunications LMS</TARGET>
    <TYPE>REFRESH</TYPE>
    <DATETIME>1999-02-03</DATETIME>
  </PROPERTIES>
  <PERSON transaction="1">
    <SOURCEDID>
      <SOURCE>California State University San Marcos</SOURCE>
      <ID>88-99-0102</ID>
    </SOURCEDID>
    <NAME>
      <FN>Stanley Wang</FN>
    </NAME>
  </PERSON>
  <PERSON transaction="1">
    <SOURCEDID>
      <SOURCE>California State University San Marcos</SOURCE>
      <ID>111-22-3344</ID>
    </SOURCEDID>
    <NAME>
      <FN>Wayne Veres</FN>
      <SORT>Veres, Wayne</SORT>
      <NICKNAME>Wayne</NICKNAME>
      <N>
        <FAMILY>Veres</FAMILY>
        <GIVEN>Wayne</GIVEN>
        <PREFIX>Mr.</PREFIX>
      </N>
    </NAME>
    <DEMOGRAPHICS>
      <GENDER>2</GENDER>
      <BDAY>1956-02-03</BDAY>
    </DEMOGRAPHICS>
    <EMAIL>veres@mailhost1.csusm.edu</EMAIL>
    <TEL teltype="1">7607504785</TEL>
    <TEL teltype="2">7607503257</TEL>
    <ADR>
      <STREET>Twin Oaks Valley Rd</STREET>
      <LOCALITY>San Marcos</LOCALITY>
      <REGION>CA</REGION>
      <PCODE>92096-0001</PCODE>
    </ADR>
  </PERSON>
  <GROUP transaction="1">
    <SOURCEDID>
      <SOURCE>College of Arts and Sciences</SOURCE>
      <ID>CS 697C Section 1 Fall 1999</ID>
    </SOURCEDID>
    <DESCRIPTION>
      <SHORT>Security In Computing</SHORT>
      <LONG>Graduate Level Special Topics course covering security in
computing today.</LONG>
      <FULL>This course will examine threats and security issues in today's
common computing environments. Prerequisites: Advanced Networks (CS 622) and
Cryptography (CS 633).</FULL>
```

```
    </DESCRIPTION>
    <ORG>
      <ORGNAM>College of Arts and Sciences</ORGNAM>
      <ORGUNIT>Computer Science</ORGUNIT>
      <TYPE>Academic</TYPE>
    </ORG>
    <TIMEFRAME>
      <BEGIN restrict="0">1999-08-26</BEGIN>
      <END restrict="0">1999-12-20</END>
      <ADMINPERIOD>Fall 1999</ADMINPERIOD>
    </TIMEFRAME>
    <ENROLLCONTROL>
      <ENROLLACCEPT>1</ENROLLACCEPT>
    </ENROLLCONTROL>
  </GROUP>
  <MEMBERSHIP>
    <SOURCEDID>
      <SOURCE>College of Arts and Sciences</SOURCE>
      <ID>CS 697C Section 1 Fall 1999</ID>
    </SOURCEDID>
    <MEMBER>
      <SOURCEDID>
        <SOURCE>California State University San Marcos</SOURCE>
        <ID>111-22-3344</ID>
      </SOURCEDID>
      <IDTYPE idtype="1"/>
      <ROLE transaction="1" roletype="01">
        <STATUS>1</STATUS>
        <COMMENTS>This student has no special needs.</COMMENTS>
        <FINALRESULT>
          <MODE>Letter Grade requested</MODE>
          <VALUES listrange="0">
            <LIST>A</LIST>
            <LIST>C</LIST>
            <LIST>F</LIST>
          </VALUES>
        </FINALRESULT>
      </ROLE>
    </MEMBER>
    <MEMBER>
      <SOURCEDID>
        <SOURCE>California State University San Marcos</SOURCE>
        <ID>88-99-0102</ID>
      </SOURCEDID>
      <IDTYPE idtype="1"/>
      <ROLE transaction="1" roletype="02">
        <SUBROLE>PRIMARY</SUBROLE>
        <STATUS>1</STATUS>
      </ROLE>
    </MEMBER>
  </MEMBERSHIP>
</ENTERPRISE>
```

Taken from *[IMS-EN-BIND-1.1]*

Relations with other standards

This specification shares data objects and model with IMS meta-data and IMS profile specification.

The specification lists relations with other standards as part of *[IMS-EN-BEST-1.1]*. In this case 'mappings' to these related standards are envisioned.

Integration with EML

In EML no persons are defined, only roles. 'Groups' are defined by allowing learners or staff members to be assigned to a role which have multiple persons

asigned to it. In short, management of student and staff information is outside the scope of EML.

Opinion Note that the specification is of 1999 and no recent update seems to exist. The issue described in the proposal is very complex but the proposal is small and seems too require more work. Is this a dead spec?

References

See *[IMS-EN-BEST-1.1]*, *[IMS-EN-INFO-1.1]*, *[IMS-EN-BIND-1.1]*.

IMS — Learner information packaging

General

The Learner Information Package (LIP) specification provides a means to package learner information to the point that the resultant data is ready for exchange between disparate systems. “Learner Information” is the broad range of information that may be used by different systems to support the learner's activities. The systems using the specification may all provide part of the information needed for fully operational Learning systems. Flexibility of framework is tehrefore essential: the most elements in the LIP are optional, and the specification can be extended.

The LIP does not provide a protocol to exchange the packages. Also, LIPs are packaged as *Content Packaging specifications* (p. 13) and exchanged in that fashion.

Components

The components (called segments) of the LIP are as follows:

- Identification — Basic information that helps identify an individual. Elements like name and address (Vcard) are contained in this area.
- Goal — Learner's personal goals and aspirations. It allows status tracking related to any item in this area. A nested structure provides facilities for capturing sub-goals.
- QCL — Qualifications, certifications and licenses. It reflects accomplishments already completed along with a structure to indicate the source of the QCL and level attained.
- Accessibility — Learner preferences, language information, disability/accessibility information and technical/physical preferences.
- Activity — Education/training work and service of the learner. It is designed for flexibility for capturing disparate activities. This area goes beyond the simple recording of the activity and result by providing a space to include activity digital representations related to the activity, e.g. a code sample or a digital representation of a work of art.
- Competency — Elements for capturing skills the learner has acquired. Skills contained in this segment are associated with formal or informal training or work history. These skills may be related to other information reflected in the Activity and/or QCL segments.
- Interest — Information on hobbies and other recreational activities. These items may be related to QCL data and may also contain digital representations.
- Transcript — A placeholder for emerging standards from other organizations. This area introduces the concept of an exrefrecord that might be used to store another data format. One example might be the inclusion of an ANSI X.12 U.S. University Academic Transcript in its native EDI format. Similarly, one might store a PDF of the same document.
- Affiliation — Information on the descriptions of the organizations associated with the learner. This may include work groups, clubs or professional associations.
- Security Key — Learner information such as passwords or security keys.
- Relationship — Description of the relationships of data contained in the other segments. All relationships in LIP have been moved from the other segments to focus the collection of such information in this segment. This

segment is also important for targeting previously provided information for update or deletion.

We give an example of a XML bounded LIP specification which expresses a QCL for a learner.

```
<?xml version="1.0" standalone="no"?>
<learnerinformation>
  <comment>An example of LIP Qualification, Certification and Licence.</comment>
  <contentype>
    <referential>
      <sourcedid>
        <source>Company X</source>
        <id>1001</id>
        <!-- identification of the source of the LIP -->
      </sourcedid>
    </referential>
  </contentype>
  <qcl>
    <typename>
      <tysource sourcetype="imsdefault"/>
      <tyvalue>Qualification</tyvalue>
      <!-- Taken from Qualification, Certification, Licence, Degree -->
    </typename>
    <contentype>
      <referential>
        <indexid>qcl_01</indexid>
        <!-- establish unique name for this QCL  -->
      </referential>
    </contentype>
    <title>MA Criminology</title>
    <organization>
      <typename>
        <tysource sourcetype="imsdefault"/>
        <tyvalue>Educational</tyvalue>
        <!-- Taken from: Professional, Employer, Government,
             Recreational, Educational, Training, Military -->
      </typename>
      <description>
        <short>Cambridge University</short>
      </description>
    </organization>
    <level>
      <text>First Class Honours</text>
    </level>
    <date>
      <typename>
        <tysource sourcetype="imsdefault"/>
        <tyvalue>Award</tyvalue>
        <!-- taken from: Effective, Birth, Start, Finish, Expiry,
             Death, Update, Create, Renewal, Delete, Publish, Award,
             Enrol, Join -->
      </typename>
      <datetime>1920</datetime>
    </date>
    <description>
      <full>
        <media mediamode="Image" mimetype="image/gif"
               contentreftype="uri">holmes/degree.gif</media>
      </full>
    </description>
  </qcl>
</learnerinformation>
```

Colin Smythe, 9th March, 2001

Relations with other standards

Student Educational Record (Transcript), ANSI ASC X.12-TS130, ANSI, April 1998.

Profile Format: Design Specification, Daniel Lipkin, Saba Inc, May 2000.

IEEE PAPI Specification - Learning Technology: Public and Private Information, Version 6.0, IEEE LTSC P1484, June 2000.

IMS *Content Packaging* (p. 13), *Meta-data* (p. 9), *QTI* (p. 19)

Integration with EML

EML does not directly access student information, but EML players are however assumjed to have access to personal information such as accessibility, preferences, portfolio information. LIP may provide a framework for mapping dossier information onto learner profiles.

References

See *[IMS-LIP-BEST-1.0]*, *[IMS-LIP-BIND-1.0]*, *[IMS-LIP-INFO-1.0]*, *[IMS-LIP-PRIMER]*

IMS — Reusable Competency Definitions

General

This specification defines an information model for describing, referencing and exchanging definitions of competencies, primarily in the context of online and distributed learning. "Competency" is used in a very general sense that includes skills, knowledge, tasks, and learning outcomes. This specification gives a way to formally represent the key characteristics of a competency independent of its use in any particular context. It enables interoperability among learning systems that deal with competency information by providing a means for them to refer to common definitions with common meanings.

The core information in a reusable competency definition is an unstructured textual definition of the competency with a globally unique ID. This information may be refined using a user-defined model of the structure of a competency. The specification is intended for interchange by machines.

Components

The specification is build on the following components:

```
GUID
Title
Description
Optional Definition (consists of:)
   Model
   Zero or more Statements (consists of:)
      StatementID
      StatementName
      StatementText
Metadata
```

Using this structure a human readable as well as automated record can be made of the competency. The author of a competency definition is free to use the Definition element in the way that best describes the competency. The *model* part identifies the model that the statements are based on.

Relations with other standards

Competencies are defined and structured in many ways in different communities of practice (ACRL, CASAS, CPA, Mager, NOICC, O*Net, PASS, SCANS, TATS). This specification allows communities of practice to exchange information according to the model they use. Extensibility can be achieved by defining the structure of the competency definition or by including LOM elements in the Meta-data portion.

This specification aligns with other IMS specs where the following constructs are concerned:

— IMS GUID guidelines and practices.

— IMS LR meta-data for extension.

Integration with EML

The RCD or a reference to a declared RCD (through GUID) can be mapped on (or replace) de EML competence level intended by the learning-objectives and

prerequisites elements. In this case the competency is thought of as either a target compenence, or a competence expected from any candidate learner or staff:

— *Learning-objectives/Learning-objective >> RCD*

— *Learning-objective/Metadata/Title >> Title*

— *Learning-objective/Objective-description >> Description*

— *Learning-objective/Objective-type >> Statement*

— [idem prerequisite and objective]

Note that the competencies are not typed, and therefore we need to maintain the distinction between objective and prerequisite. Also, this comparison does not include dossier information through *Performance-property*.

References

See *[IMS-RCD-01]*.

IMS — Simple Sequencing

General

Notice
This text is based on work in progress (*[IMS-SS-SPEC-0.7]*, *[IMS-SS]*, dd. april 2002), not on any accepted or in any way finalized text.

The IMS Simple Sequencing Specification provides a way to describe an intended behavior and resulting learning experience from a collection of learning content when the Content Resources are delivered to a learner in a managed environment. Simple Sequencing defines the relative order in which elements of content are to be presented and the conditions under which a piece of content is selected or skipped in the presentation. It incorporates rules that describe the branching or flow through the content according to the outcomes and interactions of a learner with the different pieces of content. It also describes how learner actions and events cause the sequencing process to select and deliver Content Resources to the learner.

Note 6	The term "simple" is used to clarify the constrained set of sequencing behaviors it covers, not that the specification itself is simple.

Content packs are in themselves static, and do not specifiy how content should be offered in a chain in order to express didactical structure or logic. When an IMS Simple Sequencing specification is recorded as part of a *Content Packaging* (p. 13) specification, it records the *organization* of items within the pack, and the organization of items within items. Sequencing and navigation information will enable systems to present elements of aggregate content in a predictable manner, while reacting consistently to learners' interactions with learning resources.

Note 7	An XML binding is a non-normative part of the specification.

SS goals are:

- Describe the "intended behavior of content" — the way content should be consumed by the learner. To this end SS presents a *content sequencing definition model*.
- Describe the expected behavior of a learning technology system. To this end SS presents a *behavior model*.
- Describe the kind of interactions a learner has made with the system, which may influence the sequencing. This results in a *status tracking model*.
- Describe the format to encoding the sequencing descriptions.

Note 8	SS activities are mapped onto items, groups of items, and organizations in CP. The way this is done is specified as part of the CP specification.

The specification is abstract and independent of learning activity type, i.e. format or intent. Content need *not* use a communications adapter, such as an API implementation (cf. *ADL SCORM "Shareable Content Objects"* (p. 58). The model does not prescribe an implementation.

Components

The SS model is based on the following assumptions, see also *figure* (p. 43).

Figure 8 - Representation of a simple sequencing specification.

- The SS defines sequences of *learning activities*. A learning activity may be loosely described as an instructional event or events embedded in a content resource, or as an aggregation of activities that eventually resolve to discrete content resources with their contained instructional events.
- Learning activities are modelled (conceptually) as a tree (*activity tree*). The tree consist of nodes (= activity descriptions) that may themselves be sectioned into sub-activities. Example: A lesson is followed by three tests.

  The content sequencing definition model defines the sequence of these nodes: under what conditions may the "next" node be accessed. For example: first test can be accessed only when lesson is "completed".

  Activities are always interpreted in the context of the parent activity (parents and children form a *cluster*). For example, when the parent activity is disabled, the child activities are inaccessible.

- Each node has a state for each learner, which is the basis for sequencing. Due to a number of events states may change and therefore sequencing may be evaluated differently. Example: Learner skips a test.

  The sequencing state model defines the data model for recording and updating such states.

- The sequencing model is build out of four processes: 1/ navigation, 2/ sequencing, 3/ delivery, 4/ rollup, described below. These processes take part in a "sequencing loop", which activates the processes (in that order).

Figure 9 - Simple Sequencing — the *sequencing loop*.

The sequencing loop

- 1/ *Navigation* — move through activity tree by user choice, external triggers such as timeout, etc. The specification only focusses on logical navigation events, i.e. those that trigger sequencing requests.

Note 9	For comparison: Historical events trigger navigation though a "history" of delivered activities (much in the sense of web browsers history listings); this is outside the scope of the SS.

- In a web browser, this can be visualized on screen as buttons *[go to previous activity]* (history listing) or *[stop]*

Note 10	Navigation events are: Start, Continue, Previous, Forward, back, Choose, Abandon, AbandonAll, Suspend, SuspendAll, UnqualifiedExit, ExitAll.

- Sequencing *control modes* allow a particular way to access the nodes in a cluster to be defined: modes are *Flow* (system chooses next activity based on "continue" or "previous" sequencing request), *Choice* (learner chooses an activity), and *AutoAdvance* (next activity in sequence is selected).

- 2/ *Sequencer* — determine the candidate node (i.e. node queued to be delivered). This is done as a result of a sequencing request. A *sequencing request* is the expression of a desire to traverse the conceptual activity tree in a particular direction relative to the current activity; to a particular activity; or to exit a cluster or the entire aggregation. A sequencing

request occurs as the result of an event such as a navigation event triggered by a learner, or a system generated event.

This could be visualized on screen as a button: *[get next activity]*

Note 11	Sequencing requests are: Start, Retry, RetryAll, Exit, ExitParent, ExitAll, Continue, ContinueFromParent, Previous, Choose.

The sequencing process is guided by *sequencing rules*, that take the form of [condition, action] pairs (with possible subrules). The condition tests for status information, such as mastery, progress and 'limit conditions' status. The action is to disable, skip, hide, deny forward progress or hide all activities. Sequencing rules may take precedence over sequencing requests.

Limit conditions are defined on activities and impose a constraint on access to the activity. For example: the maximum number of attempts is reached.

Note that activities can be available concurrently. The 'main' sequence is always available; parallel sequences can be defined at will. Activities are associated with one sequence only.

Finally, based on *tracking status* information the selected activities can be processed in a particular way (ignored, highlighted, recorded in a dossier etc.). Such status information includes mastery status (pass, fail, score) and progress status (activity completion status, duration, time-spans, counts etc.).

- 3/ *Delivery* — determine if node's content can and may be delivered to the learner (“validation”).

  In a browser this may result in a selected activity to be shown on screen (content, task description etc.).

Note 12	Note that while the sequencer may pass on the activity node, the delivery process may stall this node due to particular constraints (e.g. learner profile). Note also that the sequencer may pass several nodes in parallel.

  Delivery may follow a *delivery mode* associated with the activity. Such modes include “browse”, “review”, “normal”. This ensures that the same sequence and delivery can be processed in different “user roles”.

- 4/ *Rollup* — determine the “results” of a sequencing process. The rollup information is the accumulation of tracking information on subactivities.

  This could be visualized on screen as an updates score frame.

The learning designer may explicitly define the way rollup information should be accumulated. *Rollup states* determine if the activity contributes to the rolled up date values of the parent. *Rollup rules* determine the way the rollup information should be accumulated. Examples of rollup rules are "passed if", "failed if", "completed if", "incomplete if". The conditions are expressed in terms of a predefined number of subactivities that meet a condition ("passed if 60% of subactivities passed").

Three statusses can be "rolled up": Completion status (e.g. parent is complete when 2 children are complete), scores (e.g. parent scores grand total of child scores), mastery status, and duration (e.g. total duration is duration of selected activities).

The specification also introduces *control modes*. These are superimposed rules on how to interpret events and deliver content to the learner.

- Event-driven mode — sequences are traversed based on learner and navigation events.
- Completion-driven mode — sequences are traversed based on learning activity completion and exit.
- Selection-driven mode — sequences are traversed based on learner's choice.

Sequence definition model

Figure 10 - *Activity* overall structure.

The SS specification introduces the following "features" of sequences and sub-structures. The format is as follows:

- Optional multiplicity indicator (? = optional, * = zero or more, + is one or more)
- The name of the feature
- Possible values of the feature, defaults are placed between [...].
- Explanation of the feature.

The features are as follows.

XML example

An example XML binding (non-normalive) shows the static nature of the SS specification.

An XML schema for the XML binding of SS is shown below.

Figure 11 - Tree-based representation of SS schema. Attributes not included.

A sample XML file is given here. The fragment shows how the CP <manifest> is filled with an <organization> and <item> subelements, and how SS sequences are interwoven (within their own namespace). Irrelevant details are left out. See comment lines for explanation.

Note that the exact syntax of SS schema is not yet determined.

```
<!--
     There are three 'module' items in the organization, and several
     sections and paragraphs within the modules. A path is created
     through these items.

     The SS spec is inserted into the manifest as defined by IMS CP.
-->
<manifest ...namespaces...>
 <organizations>
  <organization identifier="GIUNTIUseCase1">
   <item identifier="Introduction">
    <!--
         Modules are: "introduction", "general aspects" and
```

```
        "procedural aspects". Subitems eventually point to
        resources, comprising content for the activities.
    -->
    <item identifier="IMandHI">
     <!--
          Sections "Involved materials" and "historical issues" can
          be done in any order; the "consumption factors" can be done
          only when the first 2 modules are completed.

          This is why we introduce a "dummy" item IMandHI, which we
          can bind to (end with) a SS specification.
     -->
     <item identifier="InvolvedMaterials">
      <item identifier="Presentation_Materials"/>
      <item identifier="ExerciseTest_Materials"/>
     </item>
     <item identifier="HistoricalIssues">
      <item identifier="Presentation_Historical"/>
      <item identifier="ExerciseTest_Historical"/>
     </item>
     <!--
          This is the SS part: by default the mode for
          accessing the first two activities is "choice".

          The learner can continue with next activities ('never block
          the evaluation of navigation events').

          The completion of these two items do not set the mastery
          (the complete course is not yet mastered).

          The sequence item can be 'rolled up' when all recorded
          results read "completed".
     -->
     <imsss:sequencing forwardblock="never">
      <imsss:isrolledup ismasteryrolledup="false"/>
      <imsss:rolluprules>
       <imsss:passedif setqualifier="all" statusqualifier="completed"/>
      </imsss:rolluprules>
     </imsss:sequencing>
    </item>
    <!--
        The learner gets here only when all items have been "completed".

        Completion "opens up" the activity on consumption factors.
    -->
    <item identifier="ConsumptionFactors">
     ......etc ......
    </item>
   </item>
   <item identifier="ProceduralAspects">
     ....items go here...
     <!--
          The following SS part states that within the previous items
          it is not allowed to go back in the activity list (i.e. one
          cannot 'try again').
     -->
     <imsss:sequencing forwardonly="true"/>
   </item>
   <!--
       The last SS part states that when mastery is reached, one
       cannot navigate back into the activity sequence ('previous'
       events are ignored).
   -->
   <imsss:sequencing forwardblock="mastery">
    <imsss:controlmode flow="true"/>
   </imsss:sequencing>
  </organization>
 </organizations>
 <resources>
  <!-- resources go here -->
 </resources>
</manifest>
```

GIUNTI_SSv0p6p5.xml

Relations with other standards

Through content packaging relations exist with *IMS content packaging* and *ADL SCORM* (p. 58).

Through sequencing relations exists with EML (OUNL), *IMS QTI*, and probably Carnegie Mellon Univ. LSAL/CLEO project.

Integration with EML

It is clear that a large part of the <*method*> could be represented using a sequencing specification as described. SS goes to the heart of the EML, which is the script for how to deliver content and messages to and between actors and react on the outcome of their actions. The EML strategie is to define scenario's for these interactions by defining activity sequences and contitional structures in plays.

We can compare the EML and SS specifications as shown below. A more general comparison is given *elsewhere* (p. 51).

Provisional comparison of EML and SS: element mapping

Topic	*EML*	*SS*
Activity	<Activity>	(Learning) Activity
Activity metadata	<Activity / metadata; Activity-structure/metadata>	N.a. / Set using MD specifications within CP.
Activity description in terms of "what to do and how to do it"	<Activity-description>	<Cp:Item>, points to <cp:Resource>
Activity prerequisites and objectives	<Activity / objectives activity / sequence>	n.a. / Can be implemented using IMS Reusable Competency definitions.
Activity: content objects needed to perform the activity	<Activity/environment>	<Cp:resource>
Activity sequences.	<Play> Activities are sequenced for a particular role. Continuation may require completion of tasks for other roles.	<Ss:Sequence>
Activities have inherent subactivities	<Activity-structure>	<Ss:Sequence> (same as activity sequences)
User has a choice between several "didactically equivalent" activities (A1 | A2) (user request)	<Activity-selection>	<ss:Sequencing controlmode="choice">
User must follow predefined path A1->A2->A3 (user request)	<Activity-sequence>	<ss:Sequencing controlmode="flow">

Automatically follow predefined path A1->A2->A3 (system request)	<Continue / when-completed>	<ss:Sequencing controlmode="autoAdvance">
User cannot go back to previous activity	This is against the EML philosopy that an activity visited can always be accessed again.	<ss:Sequencing forwardOnly="true">
Specification of when the activity is considered "done" in the context of the activity sequence.	<Play / continue> (may refer to completion status of activity, a condition that evaluates succesfully, or a person that determines the completion of the activity such that the sequence can continue)	<ss:CompletedIf> and <ss:PassedIf>
Traversal from activity to activity is conditional (Condition/Action C/A).	<Play / Continue / When-condition-true>	<ss:SequencingRule / ss:Condition>
C If activity is completed	<Continue / when-completed> The completion status of an activity is defined using activity-description / completed	<ss:Condition Completed>
C If a person in a particular role decides so.	<Continue / role-choice>	n.a.
C If for a predefined part of the users in role the condition is true	<Continue / when-condition-true> (users-in-role)	n.a.
C If a dossier property has no value	<Is>	n.a.
C If two or more conditions are true (AND); when one of two or more conditions are true (OR); when a condition is false (NOT)	<And>, <Or>, <Not; Is-not>	<ss:sequencingRule / Evaluation> & <ss:operator>
C If a calculated value relates to a predefined value	<Sum, subtract, multiply; divide; greater-than; less-than>	n.a.
C If no value is available for property	<No-value>	n.a.
C If elapsed time	<Time-activity-started;	<SequencingRule: TimedOut

relates to predefined time	current-time>	| OutsideAvailableTimerange> / Part of navigation model.
C If the max. number of attempts is reached	n.a.	<SequencingRule: Attempted | MaxAttemptsMet> / Part of navigation model.
A Then skip this activity	Partly by <Completed / unrestricted>	<SequencingRule / action: Skip>
A Then exit all activities	n.a.	<SequencingRule/action: ExitAll>
A Then retry, retry all	n.a.	<SequencingRule/action: Retry | RetryAll>
A Then continue, continue from parent activity	n.a.	<SequencingRule/action: Continue | ContinueFromParent >
A Then go to previous activity	n.a. Eml does not have the concept of "activity history"	<SequencingRule/action: Previous>
A Then block forward progress	Using <hide> identified activities/structures after checking dossier.	<SequencingRule/action: DenyForwardProgress>
Define sequences of activities where these activities are distributed over roles ("threads").	<Play>	n.a.
Activities can be defined for groups.	<Play/role-ref>	n.a.
Dossier: record the result of the activity	<Completed/change-property-value>	n.a. (not accessible using SS specs; may be implemented in LMS).
Offer feedback when activity is completed.	<Feedback-description>	n.a. (may be implemented using a "feedback" activity)
Activity sequence is completed when...	No more continue-conditions found.	Completion status is True
Activities are visible or hidden	Conditions / then / show | hide	<Cp:Item isVisible>
Activities have a delivery mode	n.a.	<ss:DeliveryMode>, i.e. normal, browse or review.

The following general comparison can be made at this stage.

Provisional comparison of EML and SS: global aspects

Nature of the specification	Data directed	Proces-directed
Navigational event handling	No model; not biased to any navigational system.	Browser-biased; Traversal back into previous activities (browser history) can be blocked.
Role handling	All activities are associated	No roles. Sequencing model is

	with roles; role-based activities are essential to sequencing model.	based on single learner.
Dossier building	Dossiers created for all roles; direct access to dossier through properties.	Dossier not part of the specification; no direct read/write access to any system-defined dossier.
Informal specifications	Lack of formal methods can be circumvented by textual explanations ("comments").	No informal means to express lacking constructs.
Rollup	Rollup information is calculated by author of the UoS.	Defines a model for determining the "achievement so far".
Mastery	Mastery is equivalent to completion, i.e. completion of a sequence of learner activities.	Mastery is the completion of learner activities that result in the "mastery" status (Pass, Fail). A more general "completion" status is available.
Specification modularity	Single specification	Sequencing + packaging (+ metadata)
Feedback	Feedback on an activity state (completion) can be defined.	No feedback information specified as such (could be modelled as an activity?)
Stop / resume	Sequences cannot be stopped.	A sequence can be stopped. It cannot be resumed (?)

It seems possible to integrate EML and SS, allowing each of these specifications to exist independently of eachother. This is shown in the figure below. Here we integrate EML, SS, MD and CP.

Figure 12 - EML, SS, CP and MD combined.

The figure shows that EML adds group-based logic (collaboration) into the SS specification.

References

See *[IMS-SS-SCOPE-1.0]*, *[IMS-SS-SPEC-0.7]*, *[IMS-SS]*.

Open Knowledge Initiative (OKI)

General

Open Knowledge Initiative (OKI) focusses on *tools* that are sustainable, open source, and web-based to support teaching and learning (such as discussion forums, project team notebooks, portfolios, and grade-books). It also offers a system for *assembling sets of web tools that can work together* and with other campus systems. It covers a community of planners and developers who create the basic system, tool builders (both commercial and academic), service providers who make the tools available, and students and instructors who teach and learn with the tools.

> The Open Knowledge Initiative (OKI) addresses what is perceived by many in Higher Education as a critical need: meaningful, coherent, modular, easy to use web-based environments, for assembling, delivering and accessing educational resources. OKI is building a scalable, sustainable open source reference system for Web enabled education. MIT and its partners have been working to define the parameters of an architecture having components addressing key educational management functions. The solutions pursued, however, have implications and potential benefits far beyond Cambridge and Palo Alto. By addressing the needs of a more diverse range of schools, OKI seeks to drive collaboration and spark an open-source developer community to build a sustainable support model. The generous support of the Andrew W. Mellon Foundation provides start-up funds for an initial two-year period. The Open Knowledge Initiative identifies, designs, and packages a set of web enabled learning components that will be of service to the widest range of educational environments. This solution will enhance and streamline the development, delivery, and sustainability of interesting and advanced knowledge components in the future. The products of this effort include strategies for engaging the educational community to inform the design of the product, as well as supporting its ongoing development and use. This initiative will look towards existing or proposed industry tools, open-source solutions, and consortium developed standards. A key characteristic of the project is its adherence to the open source approach to software development.
>
> OKI Website at http://web.mit.edu/oki

Components

The architecture has two main parts:

- *Enterprise Environment Specifications*: OKI provides a suite of Enterprise Environment Specifications. Through this OKI aims to *implement a reference for information exchange* based on various standards such as those coming out of the IMS project as well as various digital library initiatives.
- *Knowledge Component Specifications*: Instructors are beginning to require advanced pedagogical tools, such as voice capture, adaptive systems and systems to support team and project based learning. The OKI reference platform will support a common API and presentation specification to support this sort of development. Applications components built on top of the OKI "Core" can take advantage of this API to access and manipulate core elements such as *user profiles, roles, groups, logs, schedules, and content*. Additionally components such as grade books will be able to expose APIs to their peers. Among other benefits, this will also allow

support for pedagogical process specifications such as those coming out of the SCORM project or the OKI's own research.

Basically, OKI identifies, designs, and packages a set of web enabled learning components.The core deliverable is an architecture and application programming interface (API) specification that supports learning management systems and educational tool development.

Relations with other standards

OKI states that it incorporates IMS; as for SCORM, those parts that are not IMS are implemented in the OKIAPI that are needed for higher education. A cooperation has been set out between IMS, ADL and OKI:

> "CAMBRIDGE, Mass. and WASHINGTON, D.C. - July 11, 2001 - Leaders of the MIT's Open Knowledge Initiative (OKI), the Advanced Distributed Learning Co-Lab (ADL), and the IMS Global Learning Consortium (IMS) today announced their intention to cooperate to close the gap between innovative pedagogical technology and production learning resources. OKI, IMS, and ADL will collaborate where appropriate to accelerate the development and deployment of innovative as well as practical online learning tools, systems and techniques."
>
> Press — June 29, 2001 — MIT Open Knowledge Initiative, ADL Co-Laboratory, and IMS Cooperate to Advance Learning Technology

Integration with EML

Information is too unspecific to be able to make a comparison. OKI makes use of standard expressions such as IMS and SCORM specifications but does not define any of its own. It focusses on the API. EML, when developed and accepted, may be provided with an OKI API.

References

http://web.mit.edu/oki/about/index.html

Hot Potatoes

General

The Hot Potatoes suite includes six applications for Javascript/DHTML based internet browsers, that create interactive multiple-choice, short-answer, jumbled-sentence, crossword, matching/ordering and gap-fill exercises for the World Wide Web. The exercises use JavaScript for interactivity, and will work in Netscape Navigator and Internet Explorer versions 4 and above on both Windows and Macintosh platforms. Hot Potatoes is not freeware, but it is free of charge for non-profit educational users who make their pages available on the web.

Components

Generator for the following question types (all in Javascript/DHTML):

- Multiple Choice
- Short-answer
- Jumbled-sentence. Letters are in wrong order, put them right.
- Crossword
- Matching-ordering. For example, put 5 steps in right order.
- Gap-fill.

Relations with other standards

None specified by producers.

Integration with EML

Figure 13 - Input screen of the Hot Potatoes JQuiz submodule.

This tool seems primarily useful for allowing teachers to build their own simple exams and tests. Question rendering is completely scripted using Javascript and generated by the special interactive Hot Potatoes tools. The tool does not read XML specs but requires interactive editing.

EML question types do not include the Hot Potatoes 1/ Crossword and 2/ Jumbled sentence.

References

http://web.uvic.ca/hcmc/halfbaked/;http://www.bvenet.nl/~potatoes/ (dutch)

ADL Sharable Content Object Reference Model (SCORM)

General

SCORM is a product of the *Advanced Distributed Learning (ADL)* initiative. The purpose of the ADL is to ensure access to high-quality *education and training materials* (resources) that can be tailored to individual learner needs (personalization) and made available whenever and wherever they are required (medium neutral). ADL provides a common technical framework for computer and net-based learning that will foster the creation of reusable learning content as "instructional objects", i.e. objects used for instructional purposes, in an instructional environment

The ADL vision can be summarized as follows.

> ADL development envisions the creation of learning "knowledge" libraries, or repositories, where learning objects may be accumulated and cataloged for broad distribution and use. These objects must be readily accessible across the World Wide Web, or whatever form our global information network takes in the future.
>
> It is expected that the development of such repositories will provide the basis for a new instructional object economy that rewards content creators for developing high quality learning objects and encourages the development of whole new classes of products and services that provide accessible, sharable and adaptive learning experiences to learners.
>
> The development of reusable, sharable learning objects is key to ADL's long term vision.
>
> *[scorm-1.1]*

The *Sharable Content Object Reference Model (SCORM)* defines a 1/ Content Aggregation Model for learning, and a 2/ Runtime Environment for leaning objects.

- 1/ *Content Aggregation Model* — A model for collecting content objects such that is can be referenced, passed and reused in different learning environments. This is the basis for forming large repositories of learning resources.
- 2/ *Runtime Environment* — A specification of the interface to the actual software objects that access the learning materials. Defines API and data model for these objects.

This focus should support the construction and interoperation of learning management systems.

A *Learning Management System (LMS)* is a set of functionalities, possibly implemented in a (suite of) software tools(s), that deliver, track, report on and administer learning content, student progress, and student interactions. Such a system references content objects or aggregations of these as *Content Structure Format (CSF)* objects. The LMS is intended to track the learner. This includes gathering student profile information, delivering content to learner, monitoring interactions, and deterrmining what the learner should do next.

Components

"SCORM conformant content" conforms to particular types and categories defined within the SCORM Conformance Matrix. This includes conformant Sharable Content Objects (SCOs), XML Meta-data documents and XML Content Structure Format (CSF) documents. The following terms are used:

- *Asset* — An asset is the smallest addressable, reusable and/or trackable piece of information that can be delivered to web clients. Example: a single web page; an applet.
- *Sharable Content Objects (SCOs)* — A collection of one or more assets that use the runtime environment to communicate with the LMS. Independent of a particular learning context. Intended to be small such that reuse is feasible. Example: a set of web pages.
- *Content Structure Format (CSF)* — A "map" that aggregates learning content into a cohesive unit of instruction (e.g. course, module, lesson), applying structure to the components, and associating learning taxonomies. The map consists of a 1/ *content element*, 2/ *blocks* and sub-blocks, and finally 3/ *references to SCOs*. (A block is therefore a hierarchical representation of blocks and/or SCO's.)

  Note that in future versions this will be replaced by IMS content packaging; see [SCORM-1.1]2–42.

Figure 14 - Depiction of the SCORM data structure (XML DTD)

The result may be seen as a specific sequencing of pieces of content and a specification on how to "start that content up". This sequence is then treated as a hierarchical "lesson plan" (p. 2–58).

A CSF must not be confused with content packaging: the latter is a transport structure ('bundle and ship'), while CSFs are specifications of content ('find and launch'). A CSF however can be expressed using the

IMS content packaging specification, as this extends the bundling idea and introduces organisation of that content such that is constitutes a lesson, module and so on.

Figure 15 - Relation between assets, meta-data tagging, and API's in SCORM SCO's.

Meta-data

Alle components can be 'tagged' — assigned meta-data by a separate "meta-data" XML specification. The structure of the meta-data specification follows *IMS meta-data* (p. 9) (General, Lifecycle, etc.). Each meta-data element is "flagged' as mandatory, required or reserved within either raw media, content or course meta-data. The XML specifications are assumed to be stand-alone XML documents.

- Tagged Asset — *raw media meta-data*. Intended for course repositories.
- Tagged SCO and Tagged Block — *content meta-data*. Intended for content repositories.
- Tagged Content Aggregation (for CSF specs) — *course meta-data*. Intended or course repositories.

Runtime environment

Learning content must be interoperable across multiple *LMS*s. The runtime environment therefore must provide for three functions:

- Start the content — Launch.
- API (communication with LMS) — Initialize, finish, check error conditions, ask for a score, etc.
- Data model (predefined set of elements, or vocabulary in communication). For example, a student's score. The SCORM defines a complex set of data elements to be passed between LMS and SCO's.

Figure 16 - Launch, API and Data Model as they apply to the SCORM Run-Time Environment.

The datamodel holds the following elements (informational items pased back/forth between SCOs and LMS; notation is javascript):

- Get/set core information: *core.student_id, core.student_name, core.lesson_location, core.credit, core.lesson_status, core.entry, core.score, core.score.raw, core.score.max, core.score.min, core.total_time, core.lesson_mode, core.exit, core.session_time*
- Suspend of launch data: *suspend_data, launch_data*
- Get(/set) comments on the SCO: *comments, comments_from_Ims*
- Get/set information on the objectives N: *objectives.N.id, objectives.N.score, objectives.N.score.raw, objectives.N.score.max, objectives.N.score.min, objectives.N.status*
- Get/set information on students: *student_data.mastery_score, student_data.max_time_allowed, student_data.time_limit_action*
- Get/set student preferences: *student_preferences.audio, student_preferences.language, student_preferences.speed, student_preferences.text*
- Get/set interactive information ("questions" N): *interactions.N.id, interactions.N.objectives, interactions.N.objectives.N.id, interactions.N.time, interactions.N.type, interactions.N.correct_responses, interactions.N.correct_responses.N.pattern, interactions.N.weighting, interactions.N.student_response, interactions.N.result, interactions.N.latency*

Relations with other standards

Sharable Content Object Reference Model (SCORM) has been produced in respons to a number of meetings with IMS and LTSC on "Web-based learning

technologies". Version 1.0 of SCORM was released on January 31, 2000. Version 1.1 of SCORM was released on January 16, 2001.

New submissions and materials intended to comply with the specs can be passed to the ADL *Co-labs* that forster collaborative research and assessment of tools, standards, content and guidelines.

Integration with EML

SCORM aggregates solutions for modelling content (content structure format) and the classification of content (meta-data) with a *implementation* of that model data model and communication strategy / API. The first is well within the boundaries of EML; however, EML does not reference the implementation aspects of content or the use thereof. For example, EML does not address issues of medium neutrality or interfacing; it does address personalization and the registration of the kind of activities that can be done using the content. Within this domain, EML may be the basis for a LMS in the sense of the SCORM content aggregation model. EML adds "workflow" based roles and dependencies that are not part of the SCORM.

Note 14	For example, CSF blocks may be launched in any way the LMS desires. In this way this gives the impression of [EML play] / Continue. There is no part of the CSF that defines when or how to continue or alter the launching sequence. A natural way to present the assets would be using a table of contents.

SCORM is also focussed on modelling (parts of) *courses* (as is made clear by the fact that 1.0 was called the Sharable *Courseware* Object Reference Model), while EML focusses on activities in general, and is much more versatile in that sense.

References

See http://www.adlnet.org and *[SCORM-1.1]*, *[SCORM-PP-1.0]*

Procedural Markup Language (PML)

General

The *Procedural Markup Language (PML)* is an XML application defined in 1999. It focusses on the representation of 1/ content and the 2/ knowledge about that content. It follows the notion of the representation of content being decoupled from the design of the presentation and navigational structure, both to facilitate modular system design and to permit the construction of dynamic multimedia systems that can determine appropriate presentations in a given situation on the fly. PML allows the content of a multimedia presentation to be represented in a flexible manner by specifying the knowledge structures, the underlying physical media, and the relationships between them using cognitive media roles. The PML description can then be translated into different presentations depending on such factors as the context, goals, presentation preferences, and expertise of the user.

Components

Figure 17 - Example PML objects and relations

This image shows the relation between knowledge objects, knowledge links and cognitive media roles.

The PML introduces three basic components shown in a *separate figure* (p. 63).

- *Knowledge nodes* — Objects that represent information about a domain. Subtypes are:

    - Thing — Represents a system, physical object, part, or substance in its normal state. Things may be composed of other things. Example: Hot water system.
    - State — Each thing has one or more states that it can be in. Example: Pilot light off.
    - Procedure — Represents a sequence of actions carried out by the user that operate on a thing in some manner. Example; Lighting a pilot lamp.
- *Knowledge links* — represent conceptual relationships between knowledge nodes. Subtypes are: Is-a, Has-a, Connects-to, Steps, Problem-state, Repair-procedure, Outcome, Precondition, Uses, Related-to.
- *Cognitive media roles* — the roles a medium cluster plays in the problem-solving task in which the user is engaged. For example, an "example" (media role) is expressed through text and video media (media cluster).

  A *media cluster* contains the actual information about a knowledge node that the system can display to the user. Media are stored in external files (except for textual info). A media cluster may contain more than one type of physical media. A knowledge node may contain one or more media clusters; these are organized using cognitive media roles that provide the connections between the knowledge structures and the media clusters.

  Types of cognitive media roles are Name/title, Definition/description, Example, Counter-example, Justification.

The specification is expressed as an XML DTD, listed below.

```
<!ELEMENT pml (procedure | thing | state)*>

<!-- Things -->
<!ELEMENT thing  (title, (author | description | justification | link
                 | appspecific | example | counterexample )*) >
<!ATTLIST thing  id ID #REQUIRED>

<!-- States -->
<!ELEMENT state  (title, (author | description | justification | link
                 | appspecific | example | counterexample )*) >
<!ATTLIST state  id ID #REQUIRED>

<!-- Procedures -->
<!ELEMENT procedure   (title, (author | description | justification |
                      link | appspecific | example | counterexample )*) >
<!ATTLIST procedure id ID #REQUIRED>

<!-- Cognitive Media Types & Identifying Information -->
<!ELEMENT title          (#PCDATA | media)* >
<!ELEMENT author         (#PCDATA | media)* >
<!ELEMENT description    (#PCDATA | media)* >
<!ELEMENT justification  (#PCDATA | media)* >
<!ELEMENT example        (#PCDATA | media)* >
<!ELEMENT counterexample (#PCDATA | media)* >

<!ATTLIST description    type   CDATA #IMPLIED>
<!ATTLIST justification  type   CDATA #IMPLIED>
<!ATTLIST example        type   CDATA #IMPLIED>
<!ATTLIST counterexample type   CDATA #IMPLIED>

<!-- Media -->
<!ELEMENT media  (#PCDATA)>
<!ATTLIST media src CDATA #REQUIRED
          caption   CDATA #IMPLIED>

<!-- Links & Targets -->
<!ELEMENT link   (target+)>
```

```
<!ATTLIST link   type   (uses | is-a | has-a | connects-to |
                        related-to | steps | precondition | outcome |
                        problem-state | repair-procedure) #REQUIRED>

<!ELEMENT target  EMPTY>
<!ATTLIST target id IDREF #REQUIRED>

<!-- Application-Specific Key/Value Pairs -->
<!ELEMENT appspecific   EMPTY>
<!ATTLIST appspecific key           CDATA    #REQUIRED
                      value         CDATA    #REQUIRED>
```

[RAM99] (errors corrected)

For clarity we also provide an example XML specification.

```
<!DOCTYPE pml SYSTEM "pml.dtd">
<pml>
  <!-- All PML documents must start with the PML tag -->
  <procedure id="cake-1">
    <!-- This says we're beginning a procedure. We've given it an id of "cake 1." -->
    <!-- This is the name you use to refer to this procedure in other places. -->
    <title>How to Bake a Cake</title>
    <author>Colleen Kehoe</author>
    <!-- We define the title of this procedure and the author. Title is required. -->
    <!-- The title may be the same as the id in the procedure tag, if desired. -->
    <description>
     This procedure tells you how to bake your basic cake. It assumes
     you're at or near sea level. You'll need a different procedure
     if you're at a high altitude.
    </description>
    <!-- We give a description of the overall procedure. -->
    <!-- Since it is text, we can include it here or reference an external file via a
MEDIA tag. -->
    <justification>
     Everybody likes cake.
    </justification>
    <!-- We give a justification for this procedure. This is optional. -->
    <appspecific key="difficulty" value="easy"/>
    <!-- Here we may associate some application-specific information with this node.
-->
    <!-- This may be used for indexing purposes or for deciding how to display this
node. -->
    <example>
      <media src="cake.gif" caption="here is a picture of someone baking a cake."/>
      <media src="cake.mov" caption="here is a movie of a baker at work."/>
    </example>
    <!-- An example containing two physical media files. -->
    <!-- Any number of examples or counterexamples are allowed. -->
    <!-- Now we list all of this links from this node to other nodes in the system. -
->
    <link type="uses">
      <!-- This is a list of the equipment this procedure uses. -->
      <target id="mixer"/>
      <!-- This is a "thing" node. -->
    </link>
    <link type="problem-state">
      <!-- The following nodes are problems. -->
      <target id="cake-didnt-rise"/>
      <!-- Each is a "state" node. -->
      <target id="cake-burnt"/>
      <target id="cake-tastes-salty"/>
    </link>
    <link type="outcome">
      <!-- The following node is an outcome. -->
      <target id="cake-is-done"/>
      <!-- This is a "state" node. -->
    </link>
    <link type="steps">
      <!-- This is a list of the steps in this procedure. -->
      <target id="mix-ingredients"/>
```

```
      <!-- These are "procedure" nodes. -->
      <target id="put-in-oven"/>
      <target id="test-for-doneness"/>
      <target id="cool"/>
    </link>
    <link type="related-to">
      <!-- Other related nodes. -->
      <target id="baked-good"/>
    </link>
  </procedure>
  <!-- This marks the end of this procedure. -->
  <procedure id="mix-ingredients">
    <!-- This is the beginning of a new procedure. Notice that this is the first step
in the -->
    <!-- procedure we just defined above. Procedures are defined hierarchically. -->
    <title>Mix the Ingredients</title>
    <author>Colleen Kehoe</author>
    <description>
     Get all the ingredients together and mix them.
    </description>
    <!-- For this application, we've provided two descriptions, both text. -->
    <description type="high">
     You will need: 2 eggs, 2 c. flour, 1/2 c. milk, 3 tbsp. butter,
     1 tsp. baking soda, 1/4 tsp. salt, 1/4 c. water, 1c. sugar.
     Combine the dry ingredients in one bowl. Combine the wet
     ingredients in another bowl. Gradually add the dry to the wet,
     blending with an electric mixer.
    </description>
    <!-- Here, we provide a very detailed description. -->
    <!-- As before, we list the links from this node to other nodes in the system. --
>
    <link type="uses">
      <!-- This is a list of the equipment this procedure uses. -->
      <target id="mixer"/>
      <!-- This is a "thing" node. -->
    </link>
  </procedure>
  <!-- These are the rest of the steps in the "cake 1" procedure. -->
  <!-- Details are omitted in the interest of space, but they would be similar to the
one above. -->
  <procedure id="put-in-oven">...</procedure>
  <procedure id="test-for-doneness">...</procedure>
  <procedure id="cool">...</procedure>
  <thing id="baked good">
    <!-- Now we define a "thing" node. This is referred to in the "cake 1" procedure
above. -->
    <title>Baked Good</title>
    <author>Colleen Kehoe</author>
    <description>
     A baked good is usually found in a bakery. They are things like:
     bread, cookies, cakes, muffins, etc.
    </description>
    <counterexample>
      <media src="fish.mov" caption="while a fish can be baked, it is not considered
to be a baked good."/>
    </counterexample>
    <!-- Here, we provide a counterexample to a baked good. -->
    <!-- It is in an external media file, but we provide a textual caption as well. -
->
  </thing>
  <!-- This marks the end of the thing node. -->
  <state id="cake-didnt-rise">
    <!-- Now we define a "state" node. This was one of the problem states referred to
in the "cake 1" procedure. -->
    <title>Cake didn't rise properly</title>
    <author>Colleen Kehoe</author>
    <description>
      The cake didn't rise above the edge of the pan. This is usually
      caused by accidentally leaving out the baking powder or salt.
    </description>
    <link type="repair-procedure">
      <!-- These are links to repair procedures. -->
      <target id="eat it anyway"/>
      <!-- These are "procedure" nodes. -->
      <target id="feed to birds"/>
```

```
    </link>
    <link type="related-to">
      <!-- This is a link to a related node (here, a similar problem). -->
      <target id="cake-cracked"/>
      <!-- This is a state. -->
    </link>
  </state>
  <!-- Other procedures, things, and states are defined similarly. -->
</pml>
```

[RAM99] (errors corrected)

Relations with other standards

There are no relations with educational modelling standards. The specification offers an interpretation of a generic property specification syntax, much like RDF *[XML-RDF]*. Also, clear resemblance with Topic Maps *[XML-XTM]* in the design of knowledge nodes (*topics*), links (*associations*), roles (*roles*), and media clusters (similar to *occurrences*).

Integration with EML

The PML and EML connect only on the level of the representation of knowledge objects. PML does not define the semantics/pragmatics of activities in a learning environment. It offers a static representation of resources though re-assembled for each profile. EML does not itself separate media profiles but allows for the introduction of alternative resources to be used for similar purposes, through activity definitions and preferences set in the dossier.

Note 15	The associated language IML does introduce alternation of media objects. De choice depends on the media constraints that apply for the electronic learning environment at hand, e.g. thin web client, paper, oral. IML is not an integrated part of EML but part of the technical solution (“player”).

In short, EML uses activities to model associations between knowledge items, not (static) links.

References

See *[RAM99]*.

Tutorial Modelling Language (TML)

General

Tutorial Modelling Language (TML-5) (derived from Tutorial Markup Language 4.0) presents a simple, open framework for delivering question-based content in a cross-platform environment. TML separates content representation from delivery mechanism. The TML delivery system uses HTML to provide an interactive interface to the question sets stored on the server. It is based on the HTML syntax.

The development if TML–5 is at a *very very* early stage.

Relation with other standards

TML could quite easily make use of meta-data standards such as the Dublin Core, IMS Educational Meta-data as well as domain specific vocabularies and data formats - eg. MeSH headings for medical content, Chemical Markup Language for representing molecules or MathML for representing Maths.

http://www.ilrt.bris.ac.uk/netquest/about/lang/motivation.html

Integration with EML

TML focusses on question formulation on the web on a purely semantic level, and therefore allows for integration with EML interactions.

References

See *[TML web]*

PALO

General

PALO is a educational modelling language to describe and design learning content and learning environments at a high level of abstraction. PALO is based on a reference framework for designing educative content on 5 levels of abstraction. Each level identifies a certain group of related components or elements of a learning resource. The language allows for the definition of teaching strategies by means of specific XML DTD's called *instructional templates*. These templates are a general type of PALO document that specially suits for a given instructional or teaching purpose.

PALO offers a technology-independent representation of a learning resource

Components

PALO files are processed using a PALO Compiler called *paloc*. This tool has been developed in PERL and based in the SGML parser developed by Earl Hood.

Relation with other standards

PALO is part of European Commission: Fifth Framework Program DiviLAB project and the CEN/ISSS Learning Technologies Workshop

Integration with EML

Unclear.

References

See *[Palo web]* and *[Rodríguez-Artacho 1999]*

Customized Learning Experience Online (CLEO)

General

The Customized Learning Experience Online (CLEO) Lab is a one-year research collaboration between corporations interested in e-learning, academic researchers and the U.S. Advanced Distributed Learning Initiative (ADL). The CLEO Lab goal is to conduct focused, applied research on technical and pedagogical issues related to the *ADL Sharable Content Reference Model (SCORM)* (p. 58). The CLEO Lab is organized as a program of the IEEE Industry Standards and Technology Organization (IEEE-ISTO). CLEO is supported by Carnegie Mellon University and The Open University (U.K.)

The CLEO labs do not produce standards. The CLEO lab goal is a collaborative R & D project. Its objective is to provide focused applied research on critical areas for e-learning interoperability where specification development is premature. CLEO Lab research reports are intended to inform and accelerate the activities of initiatives developing e-learning interoperability specifications.

References

http://www.cleolab.org/

Universal Learning Format (ULF)

General

The Universal Learning Format (ULF) is a modular set of XML-based formats developed by Saba for capturing and exchanging various types of e-learning data, including online learning content, catalogs of learning resources, certification libraries, competency libraries, and learner profile information.

ULF borrows from a wide spectrum of industry standards for exchanging learning data in a web environment and brings together the key elements of these standards into a comprehensive and fully integrated solution. This solution is based on transformation using style sheets (such as XSLT). ULF is fully compatible with its constituent standards and provides a two-way path for conversion and reconversion. This ensures that data described in ULF is universally portable across all systems and taxonomies that are designed to support virtually any recognized industry standard. It also means that the ULF will shadow new developments in its constituent standards, thus providing a direct path for future extensibility.

Components

The ULF spec focusses on *reuse* and *visibility*. It makes use of XML and RDF syntax for representing the information and making it accessible in a retrieval environment. The specifications include formats for 5 domains:

- Catalog Format — Describes learning catalog entries. Catalog Format is an XML-based representation designed to capture the information associated with an e-learning product catalog. A class hierarchy is used to classify the learning resources available for purchase through the product catalog.

  This follows the LOM specifications.

- Learning Content Format (LCF) — Describes online course content
- Certification Format — Describes certification library definitions
- Competency Format — Describes competency library definitions
- Profile Format — Describes learner profile data

Relations with other standards

Explicit references are made to *ADL* (p. 58), *IMS* (p. 8), LRN, IEEE LTSC, Dublin Core, and vCard. Saba holds the viewpoint that it shopuld be logically compatible with ongoing standardisation efforts, but not on the level of binding. It is stated that tranformations are the basis for connecting to these standards. The *Saba Publisher* works on a local, proprietary format.

Integration with EML

None forseen.

References

http://www.saba.com/standards/ulf/Overview/Frames/overview.htm

Reusable Learning Objects (RLO)

General

The Reusable Learning Objects (RLO) architecture is developed by Cisco Systems. It is part of their E-learning solutions described in *[Cisco 2001]*. RLO builds upon the notion of reusable, granular objects that can be written independent of a delivery media and accessed dynamically through a database. This includes Educational objects, Learning objects, Content objects, Training components and parts thereof. The RLO implements a lesson, a task definition, or such. It may be embedded in a larger containment framework such as:

```
- Curriculum
  - Unit
    - Module
      - Lesson (RLO)
        - Topic/Page (RIO)
```

Components

The RLO Strategy is built upon the Reusable Information Object (RIO). An RIO is granular, reusable chunk of information that is media independent. An RIO can be developed once, and delivered in multiple delivery mediums. Each RIO can stand alone as a collection of content items, practice items and assessment items that are combined based on a single learning objective.

A Reusable Learning Object is created by combining an Overview, Summary, Assessment and five to nine (7 ± 2) RIOs. An RLO is based on a single objective, derived from a specific job task. Each RIO is built upon an objective that supports the RLO's objective.

The RIO itself holds content items, practice items and assessment items. Each RIO is built upon a single objective. As defined by Information Mapping, each RIO is classified as either being a Concept, Fact, Process, Principle or Procedure. For each of these RIO's a schema is defined for items that may occur in content.

The *overview* section of the RLO is used to introduce the RLO and act as an advanced organizer for the Learner by listing the objective, outline, and job-based scenario for this "lesson." It contains the following parts: Introduction, Importance, Objectives, Prerequisites, Scenario (optional), Outline. These part are all textual by nature. The scenario is related to a job function, and tells the reader how to process the RIOs.

The meta-data of a RIO holds RIO title and RIO level objective, RIO type, Job function / job task, Author name / owner name, Creation date / published date / expiration date, and Prerequisites.

Relations with other standards

None specified.

Integration with EML

The buildup of the overview resembles the EML specification, but these are not formal by nature and therefore cannot be made part of an automated learning system that tracks levels, objects, results and provides interaction between persons. Overviews should be taken as descriptive information on how to consume the rio's. The following mapping is therefore purely conceptual.

— *Introduction >> Metadata/Description*

— *Importance* >> (none). This element is persuasive by nature, trying to interest the learner in accessing and participating in the lesson.

— *Objectives >> Objectives + Learning-objectives*

— *Prerequisites >> Prerequisites*

— *Scenario >> Play*

— *Outline* >> (none)

References

See *[Cisco 2000]* and *[Cisco 2001]*.

AICC Computer-Managed Instruction (CMI)

Introduction

The Aviation Industry CBT (Computer-Based Training) Committee (AICC) is an international association of technology-based training professionals. The AICC develops guidelines for aviation industry in the development, delivery, and evaluation of CBT and related training technologies. The AICC CMI subcommittee introduces the concept of *Computer-managed Instruction (CMI)*. CMI systems manage both courseware and students in a training environment. It is primaily a scheduler of CBT materials. Through *[CMI001]* the project offers guidelines on 1/ Communication between a CMI system and a lesson, 2/ Moving a course between different CMI systems, 3/ Storing lesson evaluation data. Guidelines for the format and content of files are also described.

The relation between CMI and CBT can be depicted as follows.

Figure 18 - Relation between CMI and CBT.

While the CBT is focussed on providing information to a learner in an interactive way, the CMI deals with sequencing of lessons, prerequisites and objectives, rostering and student assignment, and feedback analysis. It can also provide a basis for testing.

Components

A CMI has the following components.

- *Course Structure Development* — this is the heart of the CMI system. The structure of the CMI is that of Curriculum, Course, Block, and Assignable unit (this is the smallest unit the CMI system assigns and tracks, "lesson"). The CMI defines Curriculum and Course on the level of management and rostering, while it allows for the sequencing of blocks and lessons, to be processed in predefined order. In this sequence the 'lessons' may have the following attributes: 1/ associated training objective(s) and classification 2/ test id(s) 3/ number of attempts allowed

4/ lesson type (on-line/off-line) 5/ lesson assignment rules 6/ need for instructor intervention 7/ resources required 8/ equipment, classrooms, instructors, consumables (e.g. workbooks) 9/ teaming requirements 10/ number of members of a team 11/ team resources 12/ remediation strategies and remedial lessons.

Course structures are based on lessons, that may be collected into blocks.

Figure 19 - Lesson structure.

Taken from *[CMI001]*

Information on the lessons is passed in files (much like windows INI files). Sequencing information is expressed in structure files and prerequisite files.

Prerequisites are lists of *logic statements*. For example: "A4 & A3=P | A3=C" which means: Lesson A4 must be completed and status of lesson A3 must be P or C. Each block or assignable unit may have a prerequisite statement.

- *Testing* — used for the development and administration of off-line and on-line tests. Types of tests include mastery, performance checklists, and attitude questionnaires. Tests are composed of test items. Test items relate to objectives.

Note 16	Testing is often done as part the lesson and is usually handled separately as a function of the CBT system. Making test definition part of the CMI system provides more flexibility in test administration, tracking, and data analysis and does not preclude the use of the CBT system in development of the test if it is to be on-line. CBT systems may pass test results back to the CMI for analysis.

- Test scores are determined acording to 1/ percentage , 2/ Number of items within an objective to pass the objective 3/ Number of objectives within the test to pass the test 4/ Item weighting 5/ Critical items (pass item to pass test) 6/ Critical objectives (pass objective to pass test)
- Tests are assigned as pre-tests or post-tests, which covers intakes.
- *Roster Operations* — entering student names and demographic data. Student classes and class assignments are also made with this component. Also the components provides reports (tables, resource overviews and so on)
- *Student Assignment Management* — lesson assignment, performance recording, certification, and assignment override by instructors. It can be the primary interface to the various CBT systems providing presentation of CBT lesson material. Assignments can be made by instructor and the system. Student login functions are also part of the component; the CMI is the single point of entrace to the student. It provides mail facilities.
- *Data Collection and Management* — The data collection component provides automated collection and management of data. This component also provides for both standard and ad-hoc reports on the data collected. The types of data collected can include the following: 1/ lesson and course summary data 2/ test item response 3/ student performance data

Related standards

Information found on the relation between CMI and ADL, IEEE/ LTSC and IMS is outdated (1999). The CMI is recently (2001) updated to conform to *ADL SCORM 1.1* (p. 58).

CMI is not much described or debated on the internet. No relevant entries found.

There are conformance tests and some systems pass these; relatively recent conformance tests have been reported. Examples are: THINQ TrainingServer Learning Management System, IBM Mindspan Solutions, Question Mark Perception, Solstra 2000, Saba LMS, and such; the conformance tests are listed at http://aicc.org/pages/aicc_ts.htm.

Integration with EML

The CMI approach much resembles the EML aproach. It deals with roles, activities, perequisites and objectives, conditional completion of taks, portfolios, and such. It extends this domain by introducing rostering and staff/student management facilities. It is sometimes very much concerned with particular technical solutions such as windows setup (the binding of management level data; initialisation information; parametrization) and mail facilities. On many other levels the specification does not dive into specific formats or bindings for content, interactions, or the like. It does not deal with meta-data other than that used to start up or maintain learning processes.

We can provide the following mapping on a general level. Note that most the components mentioned are represented as individial initialisation files in the CMI specification.

— Course >> unit of study. The files containing the structure of a course need to answer the question, "What information does a CMI system need, to present the training material to the student in the way desired by the designer?" The answer is considered to be the complete list of assignable units or blocks, with their prerequisites.

— Descriptor >> Meta-data+ID? A short description of the CBT resource: a lesson, an objective, a block.

— Assignable unit or "lesson" >> activity. Lesson structure is much like the activity structure of EML, providing a framework for accessing the functions needed to reach a set of objectives.

— Course structure >> activity structure

— Completion requirements in combination with Prerequisites and Objectives relationships >> Play. The specification defines which structure element (Iesson or block) is considered completed under what condition, and what lesson is to be provided next. Note that CMI prerequisites and objectives are always expressed in terms of other lessons or blocks, that have to be completed in advance or may be provided next within the course.

References

See *[AICC web]* and *[CMI001]*.

Bibliographic references

AICC web

Home page of the IACC: http://www.aicc.org/

Cisco 2000

Reusable Learning Objects Strategy. Cisco Systems, Inc. april 22, 2000.

Available through: http://www.cisco.com/warp/public/10/wwtraining/elearning/

Cisco 2001

Model of an E-Learning Solution Architecture for the Enterprise. Cisco Systems, Inc. april, 2001.

Available through: http://www.cisco.com/warp/public/10/wwtraining/elearning/

CMI001

CMI Guidelines for Interoperability AICC. AICC, DOCUMENT NO. CMI001, ORIGINAL RELEASE DATE 25-Oct-93 Revision 3.5 release 2 Apr 2001.

IMS-CP-BEST-1.1.2

IMS Content Packaging Best Practice Guide. Final specification, Version 1.1.2 IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-CP-BIND-1.1.2

IMS Content Packaging XML binding. Final specification, Version 1.1.2 IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-CP-INFO-1.1.2

IMS Content Packaging Information Model. Final specification, Version 1.1.2 IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-EN-BEST-1.1

IMS Enterprise Best Practice and Implementation Guide. Version 1.01. IMS global learning consortium, Inc., 1999.

Available at: http://www.imsproject.org/specifications.html

IMS-EN-BIND-1.1

IMS Enterprise XML Binding specification. Version 1.01. IMS global learning consortium, Inc., 1999.

Available at: http://www.imsproject.org/specifications.html

IMS-EN-INFO-1.1

IMS Enterprise Information Model. Version 1.01. IMS global learning consortium, Inc., 1999.

Available at: http://www.imsproject.org/specifications.html

IMS-LIP-BEST-1.0

IMS Learner Information Packaging Best Practice & Implementation Guide. Final specification, Version 1.0. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-LIP-BIND-1.0

IMS Learner Information Packaging XML Binding. Final specification, Version 1.0. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-LIP-INFO-1.0

IMS Learner Information Packaging Information Model Specification. Final specification, Version 1.0. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-LIP-PRIMER

Primer for the IMS Learner Information Package. Version 1.0. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-LR-INFO-1.1

IMS Learning Resource Meta-data Information Model. Final specification, Version 1.1. IMS global learning consortium, Inc., 2000.

Available at: http://www.imsproject.org/specifications.html

IMS-MD-BEST-1.1

IMS Learning Resource Meta-data Best Practice and Implementation Guide. Final specification, Version 1.1. IMS global learning consortium, Inc., 2000.

Available at: http://www.imsproject.org/specifications.html

IMS-MD-BEST-1.2

IMS Learning Resource Meta-data Best Practice and Implementation Guide. Public draft specification, Version 1.2. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-QTI-BEST-1.01

IMS Question & Test Interoperability Best Practice & Implementation Guide. Final specification, Version 1.01. IMS global learning consortium, Inc., 2000.

Available at: http://www.imsproject.org/specifications.html

IMS-QTI-BEST-1.1

IMS Question & Test Interoperability ASI: Best Practice & Implementation Guide. Final specification, Version 1.1. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-QTI-BIND-1.1

IMS Question & Test Interoperability ASI: XML binding Specification. Final specification, Version 1.1. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-QTI-INFO-1.01

IMS Question & Test Interoperability Information Model Specification. Final specification, Version 1.01. IMS global learning consortium, Inc., 2000.

Available at: http://www.imsproject.org/specifications.html

IMS-QTI-INFO-1.1

IMS Question & Test Interoperability ASI: Information Model Specification. Final specification, Version 1.1. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-RCD-01

IMS Reusable Competency Definitions Information Model. Version 0.1. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-RI-HB–1.0

IMS Persistent, Location-Independent, Resource Identifier Implementation Handbook. Version 1.0 Final Handbook. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-SS

IMS Simple Sequencing. Multiple materials used in initial phases of IMS specification development.

IMS-SS-SCOPE-1.0

IMS Simple Sequencing Scope document 1.0. IMS global learning consortium, Inc., 2001.

Available at: http://www.imsproject.org/specifications.html

IMS-SS-SPEC-0.7

IMS Simple Sequencing Specification. Draft version 0.7.4, april 2002.

iso10646

Information technology -- Universal multiple-octet coded character set (UCS). Part 1: Architecture and basic multilingual plane. With amendments 1-9. 1993 (ISO/IEC 10646-1:1993(E)

See also http://www.unicode.org

Palo web

PALO language Home Page on http://sensei.ieec.uned.es/palo/

RAM99

PML: Adding Flexibility to Multimedia Presentations. Ashwin Ram [etc] 1999. Available as http://www.cc.gatech.edu/faculty/ashwin/papers/git-gvu-98-20.pdf

Rodríguez-Artacho 1999

M. Rodríguez-Artacho [etc]: *Using a High-Level Language to Describe and Create Web-Based Learning Scenarios*. Published in Frontiers In Education Conference FIE '99 IEEE Computer Society. Available at http://sensei.ieec.uned.es/~miguel/articulos/fie99/1075.pdf

SCORM-1.1

Advanced Distributed Learning Sharable Content Object Reference Model Version 1.1 (January 16, 2001). Available at http://www.adlnet.org

SCORM-PP-1.0

Sharable Content Object Reference Model (SCORM) Packaging Application Profiles. Version 1.0 DRAFT (April 20, 2001)

TML web

Home page of the NetQuest: http://www.ilrt.bris.ac.uk/netquest/about/lang/

xml

T. Bray, J. Paoli, M. Sperberg-McQueen: *Extensible Markup Language 1.0*. W3C recommendation 10-february-1998. Archived at http://www.w3.org/TR/REC-xml. Second edition published as *[xml2]*

xml-rdf

Resource Description Framework. W3C recommendation of febr 11, 1999. http://www.w3.org/TR/REC-rdf-syntax

xml-xtm

XML Topic Maps. An initiative of Topicmaps.Org http://topicmaps.org/xtm/index.html. Parts: *[xml-xtm1] [xml-xtm2] [xml-xtm3] [xml-xtm4]*

xml-xtm1

XML Topic Maps (XTM) 1.0 Core Deliverables TopicMaps.Org Specification, TopicMaps.Org XTM Authoring Group, 4 Dec 2000. http://www.topicmaps.org/xtm/1.0/core.html

xml-xtm2

XML Topic Maps (XTM) 1.0 TopicMaps.Org AG Review Specification TopicMaps.Org XTM Authoring Group, 4 Dec 2000. http://www.topicmaps.org/xtm/1.0/

xml-xtm3

XTM: XML Topic Maps (XTM) Processing Model 1.0 TopicMaps.Org AG Review Specification, TopicMaps.Org XTM Authoring Group, 4 Dec 2000. http://www.topicmaps.org/xtm/1.0/xtmp1.html

xml-xtm4

TopicMaps.org Charter and By-Laws XTM Authoring Group, 9 Jun 2000. http://www.topicmaps.org/org/XTM-charter-final-20000610.html

xml2

T. Bray, J. Paoli, M. Sperberg-McQueen, E. Maler: *Extensible Markup Language (XML) 1.0 (Second Edition)* W3C Recommendation 6 October 2000 http://www.w3.org/TR/REC-xml

List of abbreviations

ADL: Advanced Distributed Learning

ASI: assessments, sections, items

CMI: Computer-managed Instruction

CSF: Content Structure Format

LMS: Learning Management System

OKI: Open Knowledge Initiative

PML: Procedural Markup Language

SCORM: Sharable Content Object Reference Model