This report describes work done by the Learning Systems Architecture Lab as a participant in the CLEO Lab project. Required deliverables for the Advanced Distributed Learning Initiative, as specified in the statement of work, are:

- Participate in the Customized Learning Experiences Online project as the leader of the specification track that will work to modify and re-architect the CSF specification to accommodate a broader set of instructional and content models.
- Provide a technical report identifying existing requirements and defining anticipated future requirements for the CSF. These requirements are described in the Requirements section of the report.
- Develop a technical report that includes a candidate successor specification to the current CSF specification. The elements of the candidate successor specification are described in the Speculations section of the report.

Project Objectives and Challenges

Customized Learning Experiences Online (CLEO) is a collaborative project among vendors, academia, and the ADL initiative. The participants assume that content consumers want more than the ability to pick and reuse course content from multiple sources (simple content interoperability and reuse). They want to create consistent learning experiences that reflect a specific learning style, learning methodology, situation or learning culture, and learning experiences that are created to meet the needs of the individual learner. To do so requires a method to describe learning content in a way that supports both interoperability and customization, both to the learner and to the instructional model.

CLEO begins with the existing Sharable Content Object Reference Model (SCORM) V1.1 as its point of departure. The goal is to improve the Content Structure Format (CSF) specification element of this model so that it adequately supports content vendor and user requirements to deliver a rich, customized, e-learning experience that incorporates content from multiple sources.

As currently structured, CSF has limited support for pedagogy and is designed primarily to describe single-user, pre-sequenced (pre-planned by the content author) conventional CBT content. CSF is based on a single, immutable, predefined learning model and a simple, predefined notation of content flow or sequencing. CSF does not (and cannot by design) support or describe content reuse in diverse learning contexts such as: self-directed, case-based,
scenario-based, adaptive, performance support, constructivist, intelligent tutoring, simulation, or collaborative learning. CSF has limited support to describe how to sequence content, and cannot describe sequencing for different learning models. CSF content embeds control behavior within content elements, making content behavior opaque. CSF does not include a method to describe how to render, layout or brand content, leaving all interaction look-and-feel choices to either content object authors or to the learning management system developers. These limitations make it much harder to reuse content elements in different contexts.

Requirements Overview

The requirements section identifies existing requirements and anticipated future requirements for a replacement Content Structure Format (CSF) Specification as an element of a future version of the Sharable Content Object Reference Model (SCORM) or other e-learning specifications and standards. These requirements are used to drive the development of a candidate to replace the current CSF specification.

The requirements described herein are based on the background work to date. It is critical to note that this work has not yet been substantively informed by the detailed requirements and use-case scenarios of the vendors in the CLEO Lab Project, and has not been aligned with the outcomes from the e-Learning "Style" Models work from the pedagogy track in the CLEO Lab Project. It is anticipated that the candidate replacement CSF specification will be influenced by these requirements and outcomes, and thus the requirements listed below are not the exclusive set that will be used to produce the candidate specification.

When describing the candidate replacement specification, CSF does not imply a single or direct replacement for the current CSF. Rather it should be interpreted as a generic term describing the collection of data models and other specifications needed to meet the stated requirements.

Speculations Overview

The speculations section describes a collection of elements that are part of the candidate replacement for the Content Structure Format (CSF) Specification. The candidate replacement is not a monolithic specification, but a collection of smaller parts. These elements are designated speculations or speclets to indicate that they are both speculative in nature and fine-grained.

The speculations are based on the background work to date and the requirements. It is critical to note that they have not yet been substantively informed by the detailed requirements and use-case scenarios of the vendors in the CLEO Lab Project, they have not been aligned with the outcomes from the e-Learning "Style" Models work from the pedagogy track in the CLEO Lab Project, and they have not been validated via the CLEO test-bed activities. It is anticipated that the speculations will be influenced by this work, and the speculations will be revised before being put forth to the larger community for further refinement and standardization.

- The data models are described in a tabular form, using the notation in the data model descriptions.
- The content models describe the learning content, both as individual elements and as collections or content aggregations.
- The behavior models describe different behaviors (pre- and post-conditions, rendering, etc.) associated with content objects.
- There is a set of basic data submodels and support models that are used in other data models.

Working Notes Overview

The working notes section contains a collection of working notes related to the development of a candidate replacement for the CSF specification.
The **Content Aggregation** note describes some of the issues associated with creating and describing a content aggregation and sequencing the delivery of objects within the aggregation.

The **Sequencing Light** note describes a simple set of extensions to SCORM content aggregations to support the basic sequencing of the items (SCOs or aggregations) in the aggregation.

The **Temporal Issues** note describes the temporal relationships between events that occur when a student interacts with one or more learning objects.

The **Content Handles** note describes how the concept of handles can abstract a collection of version, variation and preference functions from a content model.

### Next Steps

The candidate successor specification to the current CSF specification herein focuses on the overall framework for a content representation model. Additional work is required to extend and validate the proposed framework.

- Develop examples to illustrate how the framework accommodates other models, e.g., EML, the Click2Learn **Adaptive Content Framework**.
- Examine use cases from the CLEO lab project to determine if the framework accommodates these scenarios.
- Extend and detail the **content behavior** and **basic data submodels and support** models, including adding new models, refining data types and model framework.
- Refine the use of **content handles** throughout the framework.
- Define data models needed to record and track the information needed to instantiate, create and deliver the learning experience. The models as defined focus only on the definition of learning content, not dynamic information used in delivery.
- Develop test bed implementations.
- Extend the framework to encompass intelligent tutoring systems. The framework will require the addition of a **learner model**.
Requirements

Overview

This report identifies existing requirements and anticipated future requirements for a replacement Content Structure Format (CSF) Specification as an element of a future version of the Sharable Content Object Reference Model (SCORM) or other e-learning specifications and standards. These requirements are used to drive the development of a candidate to replace the current CSF specification.

The requirements described herein are based on the background work to date. It is critical to note that this work has not yet been substantively informed by the detailed requirements and use-case scenarios of the vendors in the CLEO Lab Project, and has not been aligned with the outcomes from the e-Learning "Style" Models work from the pedagogy track in the CLEO Lab Project. It is anticipated that the candidate replacement CSF specification will be influenced by these requirements and outcomes, and thus the requirements listed below are not the exclusive set that will be used to produce the candidate specification.

It is assumed that the reader is familiar with the current relevant specifications and draft standards, including SCORM, IMS Content, IMS/IEEE Metadata (LOM), and AICC/IEEE Computer Managed Instruction (CMI), and with the notations used therein.

Throughout the following, the acronym CSF is used to represent the CSF specification (the word specification is dropped). When describing background, CSF refers to the current version. When describing requirements, CSF refers to the candidate replacement specification under development unless otherwise noted. When describing the candidate replacement specification, CSF does not imply a single or direct replacement for the current CSF. Rather it should be interpreted as a generic term describing the collection of data models and other specifications needed to meet the stated requirements.

Additionally, words such as system and service are used throughout. They should be interpreted as generic terms, not descriptions of a technological approach used in any implementation.
Requirements

Requirements for both the immediate candidate replacement Content Structure Format (CSF) specification and anticipated future requirements are presented below. Overall requirements are decomposed into a set of functional areas, and detailed requirements are given for each of the functional areas. Included are a set of exclusions or requirements deemed out of scope for the current work.

No attempt has been made to differentiate between requirements for the immediate candidate replacement CSF and anticipated future requirements. The features included in the immediate replacement specification will be based on feedback and input from the CLEO project participants. The immediate replacement specification will be designed to accommodate those additional features that are anticipated but not essential for the immediate replacement specification.

Meta Requirements

The top-level overriding requirements for the candidate replacement CSF specification are:

- **Learning Model Requirement**: The specification must include a mechanism to accommodate multiple learning models or instructional models. Meeting this requirement addresses the current limit of the single, built-in learning model. **Req: O-01**

- **Sequencing Requirement**: The specification must include a declarative mechanism to define content sequencing behavior. Meeting this requirement addresses the current limited, predefined content sequencing model. **Req: O-02**

- **Content/Behavior Separation Requirement**: The specification must include a mechanism to separate learning content from control and behavior. Meeting this requirement addresses the current limitation of having control and behavior embedded in content. **Req: O-03**

- **Presentation (aka "Skins") Requirement**: The specification must include a declarative mechanism to define basic content presentation and rendering. Meeting this requirement addresses the lack of a mechanism to separate content from look and feel. **Req: O-04**

The candidate replacement specification will be presented as a collection of data models, including models for learning content, learning models, sequencing, and presentation. Associated with the data models for these elements will be a set of associated operations, behaviors or constraints on behaviors, and use of the elements of each model.

These operations and behaviors describe the required operational characteristics that a system or service that implements the specification must provide. Actual conformance to the specification is out of scope for the requirements. Associated with the models and behaviors will be a collection of practice guidelines, e.g., recommended vocabularies and taxonomies of elements described in the data models.

Detailed Requirements

Note: In the description that follows, learning content includes all content elements represented and used by the specification, e.g., learning or content objects, behavioral specifications, presentation specifications, content metadata.

The general requirements listed above are mapped to the more detailed set below. These detailed requirements are categorized into a set of functional areas.
Use and user requirements: Use and user requirements outline the target use and user communities for the specification. **Req: U**

Content representation requirements: Content structure model derived from the current CSF. **Req: CS**

Learning model requirements: Support for a collection of different learning models. **Req: LM**

Metadata requirements: Metadata information derived from LOM. **Req: MD**

Behavior and sequencing model requirements: Mechanism to control the learning experience without embedding control in learning objects or opaque control algorithms. **Req: SEQ**

Data model requirements: Other data models required for customized learning. **Req: DM**

Repository requirements: Services and storages for content management. **Req: REP**

Content rendering and presentation requirements: Models and behaviors to control content look and feel. **Req: RP**

Communications requirements: Mechanisms for communications during content experience delivery. **Req: CM**

Learning requirements: Learning content and learning results issues. **Req: L**

Implementation requirements: Implementations versus models and behaviors. **Req: I**

**Use and User Requirements**

The specification is designed for a specific set of users and uses.

- **Run-time Requirement**: The specification must support the run-time delivery of customized learning experiences. **Req: U-01**

  The specification shall enable the creation of technology systems or services to deliver content to the learner in a manner that is consistent with the described behavior represented in the CSF. A prime use of the specification is to provide the mechanism to map between the behavioral and presentation intent described and the experience presented to the learner. Thus the specification must either:

  - explicitly describe the required run-time behavior and presentation of learning content; or **Req: U-01-01**
  - incorporate a mechanism to describe the required run-time behavior and presentation of learning content such that a learning management system or content delivery service can create the appropriate user experience. **Req: U-01-02**

- **Authoring Requirement**: The specification must support the creation of customized learning experiences. **Req: U-02**

  The specification shall enable the creation (i.e., authoring) of learning experiences or instructional collections, sequences and contexts (e.g., courses) that can then be represented in CSF. Authoring uses content from a repository or repository service. Authoring may be either:

  - a computer-mediated process directed by the content or instructional author, e.g., an authoring tool; or **Req: U-02-01**
  - an automated process, e.g., experience description generated automatically by selecting appropriate content, either prior to the learner beginning to access the instruction, or in real time, based on user requirements, user profile or situation. **Req: U-02-02**

- **Interoperability Requirement**: The specification must support the storage, exchange and transport of CSF data model instances. **Req: U-03**

  The specification shall enable the transport of learning content between different content processing and storage systems and services, e.g., movement of content between repositories, LMSs, or other content and delivery services. CSF model instances or elements of instances may be moved between technological elements of learning
environments or learning services at any time, including during content delivery (run
time), content authoring, or as a back-end content management process.

All other user and uses of learning content are out of scope and will not be considered. For
example, the specification will not address:

- rights management issues;
- repository issues such as packaging, search or access;
- exchange issues such as packaging;
- authoring issues except for creating learning experiences;
- content and learner management issues beyond run-time delivery;
- knowledge management.

## Content Representation Requirements

The specification is based on a model of content structure derived from the current CSF.

- **Learning Resource Requirements:** The content representation model must include learning
  resources. **Req: CS-01**

  Learning resources (learning assets, content assets, content resources) are elements of
digital content used by learning objects. Learning resources are learning context-free.

  - The content model must represent an arbitrary unstructured collection of learning
    resources. **Req: CS-01-01**
  - Each learning resource must be uniquely identifiable. **Req: CS-01-02**
  - The resource identifier must be sufficient to globally locate and identify a resource
    for any conforming use. **Req: CS-01-03**

- **Learning Objects Requirements:** The content representation model must include
  structured learning objects. **Req: CS-02**

  Learning objects (e.g., SCOs) are elements of digital content used to create learning
  experiences. Learning objects present material for a particular learning context.

  - Each learning object may incorporate zero or more learning resources. **Req: CS-02-01**
  - A learning object may incorporate a number of other elements for learning,
    behavior and rendering. **Req: CS-02-02**
  - Each learning object must be uniquely identifiable. **Req: CS-02-03**
  - The learning object identifier must be sufficient to locate and identify a learning
    object for any conforming use. **Req: CS-02-04**

- **Learning Object Collection Requirements:** The content representation model must include
  structured collections of learning objects. **Req: CS-03**

  - Each learning object collection may incorporate one or more learning objects. **Req:
    CS-03-01**
  - The collection model must represent a nested hierarchy of learning objects. **Req:
    CS-03-02**
  - The collection model should represent a networked collection of learning objects.
    **Req: CS-03-03**
  - The collection model should be extensible to represent other structures of learning
    objects, such as unstructured collections. **Req: CS-03-04**

- **Metadata Requirements:** The content representation model must support metadata
  association with each content model element. **Req: CS-04**

  - The content model must provide a mechanism to associate a learning resource
metadata instance with each learning resource or resource variant. \textbf{Req: CS-04-01}

- The content model must provide a mechanism to associate a learning object metadata instance with each learning object or object variant. \textbf{Req: CS-04-02}

- The content model must provide a mechanism to associate a learning object collection metadata instance with each learning object collection or collection variant. \textbf{Req: CS-04-03}

- \textit{Content Variant Requirements}: The content representation model should support a collection of resource variants for each content model element. \textbf{Req: CS-05}

Variants provide \textit{equivalent} versions of an element, i.e., one variant may be substituted for another, based on the learning delivery or context use.

- The content model should represent a taxonomy or vocabulary of variant types. Variant vocabulary examples might include: \textbf{Req: CS-05-01}
  - Language
  - Technical requirements (e.g., software, bandwidth)
  - Duration
  - Cost/Price
  - Accessibility
  - Pedagogical approach
  - Localization

- The content model may define an explicit variant taxonomy or vocabulary. \textbf{Req: CS-05-02}

- The content model may include a mechanism to replace or extend the explicit variant taxonomy or vocabulary. \textbf{Req: CS-05-03}

- The content model must not exclude a mechanism to replace or extend the explicit variant taxonomy or vocabulary. \textbf{Req: CS-05-04}

\textbf{Contents}

\textbf{Learning Model Requirements}

The specification is designed to accommodate a collection of learning models.

- \textit{Multiple Learning Models Requirements}: The specification must support a collection of different learning models. \textbf{Req: LM-01}

The learning models specify the run-time delivery behavior of learning experiences. Learning models, combined with content representations, learner state behavior, learner profiles and system state instantiate events of the learning experience for delivery to the learner.

- The specification must accommodate a specific set of learning models. Potential learning models include: \textbf{Req: LM-01-01}
  - conventional CBT
  - instructor-led
  - self-directed
  - case-based
  - scenario-based
  - adaptive
  - performance support
  - constructivist
  - intelligent tutoring
  - simulation
  - collaborative learning

The list of models does not pretend to be exhaustive or even well specified.

- The specification should include a taxonomy or vocabulary of learning model types. \textbf{Req: LM-01-02}

- The specification may define an explicit, fixed, learning model taxonomy or
vocabulary. **Req: LM-01-03**
- The specification may include a mechanism to replace or extend the explicit learning model taxonomy or vocabulary. **Req: LM-01-04**
- The specification must not exclude a mechanism to replace or extend the explicit learning model taxonomy or vocabulary. **Req: LM-01-05**

- **Learning Model Representation and Behavior Requirements:** The specification should accommodate or anticipate different approaches to learner model representation and behavior. **Req: LM-02**
  - The specification must accommodate a structure consisting of: **Req: LM-02-01**
    - an implicit learner model, i.e., a closed or opaque representation, possibly proprietary.
    - an implicit proprietary controller, i.e., hard-coded or plugin content sequence controller.
    - an open CSF content representation.

  In this approach, the different models and their behaviors are not explicitly *defined* in the specification but are hardwired in an implementation. The models are simply defined by name in the specification, and their behavior is described implicitly through a statement of the expected behavior of the model. The model is not explicit; a change in the model or behavior requires a change in the specification and a change in the implementation. For example, a specific model might be the current CMI model, defined in detail. The only controls of the model are the fixed parameter set; its behavior is defined by the specification, not via any set of data.

  - The specification should accommodate a structure consisting of: **Req: LM-02-02**
    - an explicit learner model representation
    - an implicit proprietary controller, i.e., hard-coded or plugin content sequence controller.
    - an open CSF content representation.

  In this approach, each different model is implicit (defined only by name), but details of each model's behaviors are explicitly defined via a representation framework that is part of the specification. The description and behavior of each is defined by a limited set of rules or attributes that govern the behavior of the model (e.g., *playrules*). In this approach, each different model still requires additions to the specification, but the behaviors of the model can be described with a common set of mechanisms.

  - The specification should accommodate a structure consisting of: **Req: LM-02-03**
    - an explicit learner model representation.
    - an explicit controller.
    - an open CSF content representation.

  In this approach, the controller is described via a general framework, rather than being attributes and rules associated with implicit controllers. Thus, rather than limiting the approach to a fixed set of models and behaviors, any model that can be described via the representation and controller declarative systems can be used.

### Metadata Requirements

Metadata elements, based on LOM, are used throughout the data models.

- Metadata should be used throughout to express all learning and technology characteristics of all CSF data model elements. **Req: MD-01**
- Alternatives to existing learning object metadata schemes should not be used. **Req: MD-02**
- Metadata should be represented using the LOM data model. **Req: MD-03**
- Metadata may require extensions to the LOM data model. **Req: MD-04**
Elements of the metadata model may be required (not optional). **Req: MD-05**

The existing metadata model is the preferred mechanism to describe content. Alternative representations, e.g., MPEG7, should not be used and a new metadata model should not be developed unless the metadata model is inadequate. Metadata should be placed in a LOM instance, not added as elements of other data models.

### Behavior and Sequencing Model Requirements

The specification provides a model to sequence learning objects to create learning experiences for different learning models. Sequencing is based on creating contexts, i.e., collections of related learning objects targeted at a learning goal, and further classifying learning content.

- **Content Behavior Model Requirements**: The learning content model must include data items to enable sequencing and control. **Req: SEQ-01**
  - The model must be able to associate zero or more learning purposes or roles with each learning object or learning object collection. **Req: SEQ-01-01**
  - The model must represent a taxonomy or vocabulary of learning purposes. Potential learning purposes include: **Req: SEQ-01-02**
    - content presentation
    - simulation
    - procedure
    - assessment
    - remediation
    - feedback
  - The model may define an explicit learning purpose taxonomy or vocabulary. **Req: SEQ-01-03**
  - The model may include a mechanism to replace or extend the learning purpose taxonomy or vocabulary. **Req: SEQ-01-04**
  - The model must not exclude a mechanism to replace or extend the learning purpose taxonomy or vocabulary. **Req: SEQ-01-05**
  - The model must be capable of associating zero or more contexts with each learning object or learning object collection. **Req: SEQ-01-06**
  - The model must be able to associate one learning objective outcome with each learning object or learning object collection within a context. **Req: SEQ-01-07**
  - The model may be able to associate multiple learning objective outcomes with each learning object or learning object collection within a context. **Req: SEQ-01-08**
  - The model must be able to associate multiple learning objective prerequisites with each learning object or learning object collection within a context. **Req: SEQ-01-09**
  - The model must be able to represent relations between learning objects within a context. **Req: SEQ-01-10**
  - The model must be able to represent relations between learning object collections within a context. **Req: SEQ-01-11**

Relations may be applied between elements in a single collection (i.e., within the leaf level) or between collections at the same level in a content hierarchy. Relations across levels are not permitted.

- The model must represent a taxonomy or vocabulary of learning object relations. Potential relations include: **Req: SEQ-01-12**
  - pretest of
  - posttest of
  - intervention for
  - elaboration for
  - The model may define an explicit relation taxonomy or vocabulary. **Req: SEQ-01-13**
  - The model may include a mechanism to replace or extend the relation taxonomy or vocabulary. **Req: SEQ-01-14**
  - The model must not exclude a mechanism to replace or extend the relation taxonomy or vocabulary. **Req: SEQ-01-15**
- **Content Sequencing Model Requirements**: The sequencing model must define how to associate sequencing and control information with content within a context. **Req: SEQ-02**
  - The model must be able to associate zero or more sequencing methods with any collection of content. **Req: SEQ-02-01**
  - The model must be able to associate each sequencing method with a particular context. **Req: SEQ-02-02**
  - The model must provide a scoping mechanism to define how a sequencing method is associated with the structural model of content. **Req: SEQ-02-03**
  - The model must include a scoping method that is capable of associating a sequence (control) method with any learning object collection within a context. **Req: SEQ-02-04**

Sequencing is applied to only a single collection, not across collections. Sequencing applies to only that collection. For the simple case when a collection is a set of SCOs, sequencing applies to only that set. For the more complex case when the entire content collection is just a set of content objects, a global sequencing method can be used on the entire set (e.g., forward chaining on objectives).

  - The model must include a scoping mechanism that is capable of associating the sequencing method with the learning object collection when the collection is authored for the learning experience. **Req: SEQ-02-05**
  - The model should provide a way to associate the sequence with the learning object collection dynamically while the learning experience is being delivered to or generated for the learner. **Req: SEQ-02-06**

Sequencing approaches may be pre-defined or may be dynamically assigned.

- **Sequencing Method Requirements**: The sequencing method must define how to sequence content. **Req: SEQ-03**
  - Each sequencing method must provide a predictable description or state model for the method. **Req: SEQ-03-01**
    - The model must describe the actions for a series of states for the learning object (applied to each instance of the object). Possible states include: **Req: SEQ-03-01-01**
      - initial state
      - entry state
      - restart state
      - suspend state
      - preempt state
      - completed state
    - The model must specify a default action or exception behavior that is applied when the normal actions fail. **Req: SEQ-03-01-02**
    - The model must specify a default action or exception behavior when access to data from a different data model fails. **Req: SEQ-03-01-03**
  - The model must represent a taxonomy or vocabulary of sequence or control behaviors. Possible behaviors include: **Req: SEQ-03-02**
    - select the unit of content to present
    - personalize the content
    - select the method to present the selection
    - monitor tracking and results from the content
    - select the next element in the sequence
    - exception handling for selection in case of failure
  - The model must represent a taxonomy or vocabulary of sequence or control actions. **Req: SEQ-03-03**
  - The model may include a mechanism to replace or extend the control action taxonomy or vocabulary. **Req: SEQ-03-04**
  - The model must not exclude a mechanism to replace or extend the control action taxonomy or vocabulary. **Req: SEQ-03-05**
  - The model must represent a taxonomy or vocabulary of sequence or control events. **Req: SEQ-03-06**
  - The model may include a mechanism to replace or extend the control event taxonomy or vocabulary. **Req: SEQ-03-07**
The model must not exclude a mechanism to replace or extend the control event taxonomy or vocabulary. **Req: SEQ-03-08**

- The model must support a single active sequencing process. **Req: SEQ-03-09**
- The model should support concurrent asynchronous active sequencing processes. **Req: SEQ-03-10**
- The model must anticipate non deterministic behavior. **Req: SEQ-03-11**
- The model should include a meta mechanism to control the sequencing methods, e.g., scope control, exception handling, inheritance. **Req: SEQ-03-12**

**Sequencing Behavior Requirements:** The model must support a variety of sequencing behaviors. **Req: SEQ-04**

- The model must support sequencing based on content variants. **Req: SEQ-04-01**
- The model must support sequencing independent of user differentiation. Possible undifferentiated sequencing behaviors include: **Req: SEQ-04-02**
  - sequence based on explicit rules, e.g., page turn
  - sequence based on content only, e.g., prerequisite sequence
  - sequence based on user selection, e.g., *user choice*
  - sequence based on instructor selection
  - sequence based on context, e.g., schedule
- The model must support sequencing based on learner profiles. Possible profile-based sequencing behaviors include: **Req: SEQ-04-03**
  - sequence based on user identity
  - sequence based on user selected preferences
  - sequence based on user demographics
  - sequence based on user cognitive style
  - sequence based on user situation
- The model must support sequencing based on learning models. Possible learning-model based sequence behaviors include: **Req: SEQ-04-04**
  - sequence based on prerequisites
  - sequence based on learner model
  - sequence based on expert learner model tracing
  - sequence based on adaptive testing
- The model must represent a taxonomy or vocabulary of sequencing behaviors. **Req: SEQ-04-05**
- The model may include a mechanism to replace or extend the sequencing behavior taxonomy or vocabulary. **Req: SEQ-04-06**
- The model must not exclude a mechanism to replace or extend the sequencing behavior taxonomy or vocabulary. **Req: SEQ-04-07**
- The model must define the sequencing method for each sequencing behavior. **Req: SEQ-04-08**
- The model should include the definition of the sequence method for a collection of sequencing behaviors. **Req: SEQ-04-09**
- The model must support an explicit method for defining the sequencing of a learning object collection. **Req: SEQ-04-10**
- The model may use the explicit method of defining a sequence method to define the other sequencing behaviors. **Req: SEQ-04-11**
- The model should not require the use of the explicit method to define the other sequencing behaviors. **Req: SEQ-04-12**

An example of an explicit method is a rule-based strategy. All behaviors could use a single rule-based approach, or each of the different models could be implemented directly, i.e., programmed as part of the content sequencing service.

---

**Data Model Requirements**

The specification assumes the existence of a collection of other data models, used in learning experience control and sequencing.

- **Tracking Model Requirements:** The specification assumes a model for tracking content experience delivery. **Req: DM-01**
The tracking model is used to record information about delivery and sequencing of the learning experience so that the behavior and sequencing process can properly create the experience without relying on the content-to-process tracking communications.

- The tracking model must record all information about the learning experience for each individual learner, as required by the learning model. **Req: DM-01-01**
- The tracking model must record all information about the learning experience for each individual learner, as required by the behavior and sequencing model. **Req: DM-01-02**
- The tracking model should record all information, events, and control operations of all components. **Req: DM-01-03**
- All tracking model data must be available for use by the behavior and sequencing models. **Req: DM-01-04**
- Tracking model data should not be available to individual content objects at delivery time. **Req: DM-01-05**
- Certain tracking model data (currently undetermined) should not be available to individual content objects at delivery time. **Req: DM-01-05/a**
- The tracking model must differentiate between current data and historic data. **Req: DM-01-06**

- **Learner Profile Model Requirements**: The specification assumes a model of the learner or a learner profile. **Req: DM-02**

The learner profile provides learning experience independent information about the learner (e.g., name, preferences) so that either content or controllers can create the appropriate learning experience.

- The learner profile model should provide data needed to adapt the learning experience to the learner. **Req: DM-02-01**
- All learner profile model data must be available for use by the behavior and sequencing models. **Req: DM-02-02**
- Demographic learner profile model data should be available to individual content objects at delivery time. **Req: DM-02-03**
- Non-demographic learner profile model data should not be available to individual content objects at delivery time. **Req: DM-02-04**
- The profile model should incorporate the following data (other data may be needed to support other learning models) **Req: DM-02-05**:
  - learner demographics
  - learner needs (what the user wants to learn, e.g., content topic)
  - learner objectives (why the user wants to learn: e.g., formal education and training, informal learning, job aid; and when the user wants to learn: e.g., immediate, deferred)
  - learner background (what the user knows)
  - learner learning style
  - learner technology preferences
  - current physical learning environment (e.g., location, current connectivity and access)

- **System State Model Requirements**: The specification assumes a model of system state. **Req: DM-03**

The system state model provides global information (e.g., date, time, location, connectivity) so that either content or controllers can create the appropriate learning experience.

- The system state model should provide data needed to adapt the learning experience to the system environment. **Req: DM-03-01**
- The system state model should record system data for all components. **Req: DM-03-02**
- The system state model must differentiate between current data and historic data. **Req: DM-03-03**
- **Learner State Model Requirements:** The specification may require a model of learner state. **Req: DM-04**

  The learner state model describes what the learner knows. An instance of the learner state model is used to aggregate and abstract the tracking model data for a learning experience. The learner state model is used so that either content or controllers can create the appropriate learning experience. The learner state model data may be used in conjunction with the problem state model data and the expert learner state model data.

  - All learner state model data must be available for use by the behavior and sequencing models. **Req: DM-04-01**
  - Certain learner state model data (currently undetermined) should be available to individual content objects at delivery time. **Req: DM-04-02**
  - Certain learner state model data (currently undetermined) should not be available to individual content objects at delivery time. **Req: DM-04-03**
  - The learner state model must represent the history of learning states. **Req: DM-04-04**

- **Expert Learner Model Requirements:** The specification may require a model of an expert learner. **Req: DM-05**

  The expert learner model describes how an expert learns, and is used to guide sequencing and control in certain learning models.

  - The expert learner model should provide data needed to create a learning experience to match the learner state model with the expert learner model. **Req: DM-05-01**
  - All expert learner model data must be available for use by the behavior and sequencing models. **Req: DM-05-02**
  - Certain expert learner model data (currently undetermined) should be available to individual content objects at delivery time. **Req: DM-05-03**
  - Certain expert learner state model data (currently undetermined) should not be available to individual content objects at delivery time. **Req: DM-05-04**
  - The expert learner model must represent a collection of distinct expert learning states. **Req: DM-05-05**
  - The expert learner model must represent a collection of paths through the collection of expert learning states. **Req: DM-05-06**
  - The expert learner model must represent transitions between the collection of expert learning states. **Req: DM-05-07**
  - The specification may require a collection of expert learner models, with one or more for each learning model. **Req: DM-05-08**

- **Problem Space Model Requirements:** The specification may require a model of problem space and problem state. **Req: DM-06**

  The problem space model describes the components of the problem space and problem-solving environment, e.g., both the physical resources and the learning scaffolding, along with an actual state model of the learner. The problem space model is used in conjunction with a particular learning model and behavior model to provide elements of the learning experience and guide the learner through the problem space.

  - The problem space model should provide data needed to adapt the learning experience to the problem space environment. **Req: DM-06-01**
  - The problem space model may be learning model-specific. **Req: DM-06-02**
To support learner model specific problem space models, a mechanism is needed to define and detail the problem space models and possibly to associate them with the learning models.

- The specification should include a taxonomy or vocabulary of problem space model types. **Req: DM-06-03**
- The specification may define an explicit, fixed, problem space model taxonomy or vocabulary. **Req: DM-06-04**
- The specification may include a mechanism to replace or extend the explicit problem space model taxonomy or vocabulary. **Req: DM-06-05**
- The specification must not exclude a mechanism to replace or extend the explicit problem space model taxonomy or vocabulary. **Req: DM-06-06**
- All problem space model data must be available for use by the behavior and sequencing models. **Req: DM-06-07**
- Certain problem space model data (currently undetermined) should be available to individual content objects at delivery time. **Req: DM-06-08**
- Certain problem space model data (currently undetermined) should not be available to individual content objects at delivery time. **Req: DM-06-09**
- The problem space model should include a description of elements of the problem solving and learning environment (e.g., scaffolding). **Req: DM-06-10**
- The specification should include a taxonomy or vocabulary of the problem solving and learning environment. **Req: DM-06-11**
- The specification may define an explicit, fixed, problem solving and learning environment taxonomy or vocabulary. **Req: DM-06-12**
- The specification may include a mechanism to replace or extend the explicit problem solving and learning environment taxonomy or vocabulary. **Req: DM-06-13**
- The specification must not exclude a mechanism to replace or extend the explicit problem solving and learning environment taxonomy or vocabulary. **Req: DM-06-14**

These data models are in addition to the other models detailed throughout:

- Learning Model
- Behavior and Sequencing Model
- Rendering and Presentation Model
- Communications Model
- Content Model
- Repository Model

Repository Requirements

The specification assumes a content repository.

- The specification must support storage of all CSF model elements in a repository service. **Req: REP-01**
- All repository elements must have metadata. **Req: REP-02**
- All repository elements must be uniquely identifiable. **Req: REP-03**
- The element identifier must be sufficient to globally locate and identify a resource for any conforming use. **Req: REP-04**
- The repository service must support versioning. **Req: REP-05**
- The repository service must support variants. **Req: REP-06**

Except for the use and user requirements stated, the specification will not describe any other repository requirements.

Repositories are abstract services. There may be one or more repositories, either individual or confederated. From the view of content or other models, such distinctions are hidden and not discernible.
Content Rendering and Presentation Requirements

The specification provides models to describe presentation and controls to render content.

- **Rendering Model Requirements**: The learning content model must include data items to enable rendering and presentation control. **Req: RP-01**

  Rendering models specify how content is structured and what role it provides in presenting the learning experience. The presentation structure of the learning experience may differ from the learning model structure of the experience. For example, the content model may treat a learning presentation and the related learning assessment as two separate learning objects, while the content presentation treats the pair as a single unit of presentation.

  - The content representation model may require a structured collection of learning objects or learning object collections. **Req: RP-01-01**
  - The rendering object collection may be isomorphic with the learning object collections. **Req: RP-01-02**
  - The rendering object collection may be a separate collection model. **Req: RP-01-03**
    - Each rendering object collection may incorporate one or more learning objects. **Req: RP-01-03-01**
    - Each rendering object collection may incorporate one or more learning object collections. **Req: RP-01-03-02**
    - The collection model must represent a nested hierarchy of rendering collections. **Req: RP-01-03-03**
    - The collection model should represent a networked collection of rendering collections. **Req: RP-01-03-04**
    - The collection model should be extensible to represent other structures of rendering collections, such as unstructured collections. **Req: RP-01-03-05**
  - The model must be able to associate a presentation role with each learning object or rendering object collection within a context. **Req: RP-01-04**
  - The model must represent a taxonomy or vocabulary of presentation roles. Potential presentation roles include: **Req: RP-01-05**
    - syllabus
    - calendar
    - news
    - lectures
    - assignments
    - tutorials
    - readings
    - student status
    - glossary
    - references
    - faq
    - help
    - content presentation
    - menu
    - assessment
    - search
    - remediation
    - intervention
  - The model may define an explicit learning presentation role taxonomy or vocabulary. **Req: RP-01-06**
  - The model may include a mechanism to replace or extend the presentation role taxonomy or vocabulary. **Req: RP-01-07**
  - The model must not exclude a mechanism to replace or extend the presentation role taxonomy or vocabulary. **Req: RP-01-08**
  - The model must be able to associate zero or more rendering behaviors with each rendering object collection. **Req: RP-01-09**

- **Content Markup Requirements**: The specification should include a content markup
The content markup language provides a mechanism to customize learning objects and describe the elements of a learning object that should be rendered in a particular style.

- The markup language should include a set of features to describe learner personalization of a learning object, e.g., name. Req: RP-02-01
- The markup language should include a set of features to describe context customization of a learning object, e.g., course name. Req: RP-02-02
- The markup language should include a set of features to describe system state customization of a learning object, e.g., time, date. Req: RP-02-03
- The markup language should include a set of features to describe problem space customization of a learning object. Req: RP-02-04
- The markup language must include a taxonomy or vocabulary of learning object content personalization and customization features. Req: RP-02-05
- The model may define an explicit rendering feature taxonomy or vocabulary. Req: RP-02-06
- The model may include a mechanism to replace or extend the rendering feature taxonomy or vocabulary. Req: RP-02-07
- The model must not exclude a mechanism to replace or extend the rendering feature taxonomy or vocabulary. Req: RP-02-08
- The markup language may include a set of features to render any data item of any other data model. Req: RP-02-09
- The markup language may include a set of features to define the learning content components within each learning object. Possible learning components include: Req: RP-02-10
  - objective
  - procedure
  - concept
  - summary
  - definition
  - example
  - practice
  - demonstration
  - test item
  - distractor
- The model may define an explicit learning content components taxonomy or vocabulary. Req: RP-02-11
- The model may include a mechanism to replace or extend the learning content components taxonomy or vocabulary. Req: RP-02-12
- The model must not exclude a mechanism to replace or extend the learning content components taxonomy or vocabulary. Req: RP-02-13
- The markup language may include a set of features to define the navigation and sequencing controls. Possible navigation controls include: Req: RP-02-14
  - next
  - previous
  - first
  - last
  - exit
  - contents
  - top
- The model may define an explicit navigation control taxonomy or vocabulary. Req: RP-02-15
- The model may include a mechanism to replace or extend the navigation control taxonomy or vocabulary. Req: RP-02-16
- The model must not exclude a mechanism to replace or extend the navigation control taxonomy or vocabulary. Req: RP-02-17
- The model must provide a set of named rendering styles. Req: RP-02-18
- For each named rendering style, the model should provide a mechanism to specify how to render each feature of the markup language. Req: RP-02-19
- For each named rendering style, the model must provide a default mechanism to specify how to render any feature of the markup language. Req: RP-02-20

The content markup language could be implemented as a set of XML elements for content
or XML models for the different data models, combined with XSLT models to render the features of the objects.

- **Content Presentation Behavior Requirements:** The specification must define how rendering and markup models are used to create rendered content. **Req: RP-03**
  - The model must provide a set of named presentation behaviors. **Req: RP-03-01**
  - For each presentation behavior, the model should provide a mechanism to define how to present each role within the named behavior. **Req: RP-03-02**
  - For each presentation behavior, the model must provide a default mechanism to define how to present any role within the named behavior. **Req: RP-03-03**
  - The model must provide a scoping mechanism to define how a presentation behavior is associated with the structural model of content. **Req: RP-03-04**
  - The model must be able to associate zero or more presentation behaviors with any rendering object collection of content. **Req: RP-03-05**
  - The model must define the presentation look and feel for each presentation behavior. **Req: RP-03-06**
  - The model should support an explicit method for defining the look and feel for each role and each presentation behavior. **Req: RP-03-07**
  - The model may use the explicit method of defining presentation to define the other presentation behaviors. **Req: RP-03-08**
  - The model should not require the use of the explicit method to define the other presentation behaviors. **Req: RP-03-09**
  - The model may include a global presentation behavior used independently of any rendering object collections. **Req: RP-03-10**

An example of an explicit method is a style sheet-based strategy. All presentation behaviors could use such an approach, or each of the different behaviors could be implemented directly, i.e., programmed as part of the content rendering and presentation service.

- **Content Presentation Requirements:** The specification must define overall presentation requirements. **Req: RP-04**
  - The model must support content without any rendering specifications. **Req: RP-04-01**
  - The model should support content with embedded (opaque) rendering and navigation specifications. **Req: RP-04-02**
  - The model should support content that is rendered before presentation. **Req: RP-04-03**
  - The model should support content that is dynamically rendered during presentation. **Req: RP-04-04**
  - The model must support content that has no notion of communicating with the content delivery system or service to control events and navigation. **Req: RP-04-05**
  - The model should support content with explicit communications to the content delivery system or service to handle events and navigation. **Req: RP-04-06**
  - The model must support a single active content object presentation. **Req: RP-04-07**
  - The model should support concurrent asynchronous active content object presentations. **Req: RP-04-08**
  - The model should include a meta mechanism to control the content presentation. **Req: RP-04-09**

**Communications Requirements**

The specification assumes a mechanism for data transfer between various components during content experience delivery or in support of learning and content management for content experience delivery.

- **Content Communications Requirements:** Learning content must be able to exchange data with other components during delivery of the learning experience. Requirements for the communications model include: **Req: CM-01**
o A mechanism for a learning object to initiate and manage communications with the delivery system or service. **Req: CM-01-01**

This is the current model of content communications; all data transfer is under control of the content.

o A mechanism for the system or service to initiate communications with a learning object. **Req: CM-01-02**

This is the mechanism for system-controlled content and system-controlled communications. Control requirements are listed below.

o The communications model and delivery environment must not require that a learning object support communications to enable customized learning. **Req: CM-01-03**

Content communications is optional, i.e., dumb content is permitted. The overall control and behavior models must support dumb content via delivery system mechanisms for tracking, control, etc.

o The communications model must support data transfer (movement of data model items and attributes). **Req: CM-01-04**

o The communications model should support control communications. **Req: CM-01-05**

Control mechanisms include:

- Initiating a learning experience.
- Monitoring a learning experience (independent of content control).
- Terminating a learning experience.
- Preempting or suspending a learning experience.
- System trapping of communications exceptions.
- Content triggering an event.
- System triggering an event.
- Content trapping an event.
- System trapping an event.

The system controlled communication model should support all of the mechanisms listed.

o The communications model must provide a single communications method to provide access to all data models and elements of the CSF. **Req: CM-01-06**

o The communications model must support disconnected use. **Req: CM-01-07**

o The communications model must abstract connected versus disconnected use from content. **Req: CM-01-08**

o The communications model must include a state model defining communications behavior. **Req: CM-01-09**

o The communications model must provide unidirectional point-to-point communications control. **Req: CM-01-10**

Unidirectional control can be either content initiated or system initiated. The current API is unidirectional content initiated.

o The communications model should provide bidirectional point-to-point communications control. **Req: CM-01-11**

Bidirectional communications may be equivalent to a pair of unidirectional controls.

o The communications model should provide broadcast communications control. **Req: CM-01-12**
Broadcast or collaborative control may be 1:N or N:N communications.

- The communications model must support one content-to-system communications channel. **Req: CM-01-13**
- The communications model should support multiple content-to-system communications channels, one channel per active content element. **Req: CM-01-14**
- The communications model may support multiple content-to-system communications channels with multiple channels per active content element. **Req: CM-01-15**

A single content channel provides for a single active SCO. Multiple channels provide for concurrent SCOs or multiple links to a single SCO.

### Data Model Communications Requirements:

Various communication links are required. **Req: CM-02**

- Content-to-data model communications must be supported. **Req: CM-02-01**
- Content-to-content communications should be supported. **Req: CM-02-02**
- Data model-to-data model communications should be supported. **Req: CM-02-03**

The same communications model is used for all data models. Potential data models include:

- Content model
- Learning model
- Tracking model
- Learner/profile model
- System state model
- Learner state model

### Learning Requirements

The specification is learning results neutral.

- The specification makes no requirements about the quality of any learning object, learning model or learning experience. **Req: L-01**
- The specification makes no requirements about the effectiveness of any learning object, learning model or learning experience. **Req: L-02**
- The specification makes no requirements about the granularity of any learning object or learning experience. **Req: L-03**
- The specification should permit content and learning objects that are opaque. **Req: L-04**

### Implementation Requirements

The specification is designed to focus on models and behavior and not implementations.

- **Implementation Independence Requirements**: The specification must be neutral to implementations. **Req: I-01**

The specification shall not detail an implementation and shall not constrain a conforming implementation to a particular strategy or technology.

- The specification must not constrain an implementation to a particular architectural approach e.g., client-server, peer-to-peer, server-centric. **Req: I-01-01**
- The specification shall not specify bindings and representations for data models and processing used with implementations. **Req: I-01-02**
The specification must enable data model or data model element binding to an external form for transport and exchange. **Req: I-01-03**

The specification shall not constrain the data model binding to a single external form. **Req: I-01-04**

It should be possible to bind the data models or data model elements to XML for external use. **Req: I-01-05**

It should be possible to transform existing CSF instances represented in XML to XML instances of the new specification. **Req: I-01-06**

The specification shall not specify communications transport protocols or bindings. **Req: I-01-07**

The specification shall not specify repository structure. **Req: I-01-08**

- **Conformance Requirements**: The specification will not specify conformance requirements. **Req: I-02**
  - Separate application profiles should specify conformance. **Req: I-02-01**
  - Conformance to the specification should only be measured by performance. **Req: I-02-02**
  - Conformance to bindings may be prescriptive. **Req: I-02-03**

  Prescriptive conformance to the specification may imply an implementation bias.

- **Transport Independence Requirement**: The specification must be neutral to CSF instance transport. **Req: I-03**

  The specification shall not detail the mechanism used to transport content between different content processing and storage systems. Actual mechanisms to package and transport content are out of scope, and assumed to be covered in areas such as the IMS Content Packaging Specification.

- **Viability Requirement**: All elements of the specification should be shown to be demonstrably viable. **Req: I-04**

  Viability can be shown by trial implementations and empirically testing that the model works across different instructional models and provides interoperability of content among systems and services.

**Discussion**

Setting requirements for sequencing and behavior is not straightforward due to the different approaches that are possible. One approach is to ignore all learning model and associated global strategies and represent sequencing only via a set of localized rules applied to a content object within the content collection.

This approach should always be feasible, if not practical. For example, an authoring tool might interpret a global strategy and develop the appropriate local control rules. However, unless the content objects are tagged to indicate the type of global strategy expressed in the local rules, it is impossible to determine the strategy for content collection reuse.

Another approach is to establish a global control strategy. In this approach, rules are not associated with the content collection, but with the defined experience or with an overall learning model. The global strategy or global rules find and sequence content from a larger collection of learning objects. Localized control is not applied.

Even when using only local rules, there is still some overriding control strategy, i.e., there is always some global control strategy (e.g., graph traversal with rule firing on each node). Similarly, in the global strategy, there may be a need for local control or overrides, either via rules or parameters. Additionally, a learning experience may be composed of content for which both local control is appropriate and a global strategy is needed. These different control behaviors and methods are needed at different levels in the content structure, and are a natural result from attempting to create reusable content aggregations.
Thus there is always the requirement for both a global and a local control regime. Since local and global control may be freely intermixed, having two different approaches and models of control seems problematic. The requirements above attempt to provide a framework for a single approach that can meet both needs.

The problem is further compounded by layering a separate set of presentation or rendering models on content, with possibly a separate organizational structure. Rendering is also both a local problem and a global problem, i.e., how to display sequencing (next/back) buttons for a sequence collection of content and how to display that content within an overall presentation framework or problem-solving scaffold (content, navigation, glossary, etc.).

Thus there are two potentially competing local strategies (sequencing and rendering) and two competing global strategies in place, all of which need to be coordinated to properly present the learning experience.
Speculations

Overview

This report describes a collection of elements that are part of the candidate replacement for the Content Structure Format (CSF) Specification. The candidate replacement is not a monolithic specification, but a collection of smaller parts. These elements are designated *speculations* or *speclets* to indicate that they are both speculative in nature and fine-grained.

The speculations are based on the background work to date and the requirements. It is critical to note that they have not yet been substantively informed by the detailed requirements and use-case scenarios of the vendors in the CLEO Lab Project, they have not been aligned with the outcomes from the e-Learning "Style" Models work from the pedagogy track in the CLEO Lab Project, and they have not been validated via the CLEO test-bed activities. It is anticipated that the speculations will be influenced by this work, and the speculations will be revised before being put forth to the larger community for further refinement and standardization.

Data Model Descriptions

The speculations include a collection of data models. These data models are described in a tabular form, using the notation described in the data models descriptions. These data models form a set of abstract data types that can be used to form other data models. The core data types used to build all models are also listed.

Data Models

The speculations include the different data models listed below. In the current version, the emphasis is on the overall framework for representing content and supporting a diverse collection of learning behaviors. The speculations focus on information models, and the top-level structure of content representation. Additional work is required to fill out the details of all models, and to describe the details of behaviors and associated run time data models. Rendering and skins are not addressed.

- Content Models
  - Aggregate Learning Object Model
  - Contextualized Learning Object Model
  - Purposed Learning Object Model
  - Learning Object Model
- Behavior Models
Supporting Data Models and Submodels

The speculations include a set of basic submodels and supporting models that are used in other data models.

- Object ID Submodel
- Competencies Submodel
- Objectives Submodel
- Prerequisites Submodel
- Handle Submodel
- Aggregate Collection Model
- Metadata Model
- Resources Model
- Learning Context Model
The speculations include a collection of data models. These data models are described in a tabular form, using the notation described below. These data models form a set of abstract data types that can be used to form other data models. The core data types used to build all models are listed below.

Bindings will detail how to map the abstract data types and data models to an actual implementation representation.

Data Model Description

The elements of a data model are presented in a simple tabular form. Each element of the data model is described using the following attributes.

- **Name**: Descriptive name of the data element. The name will be bound to a more specific identifier in a binding of the data model.
- **Description**: A brief description of the data element.
- **Obligation**:
  - **O**: the element is optional
  - **R**: the element is required
- **Multiplicity**:
  - **0**: the element is reserved and is not used
  - **1**: a single occurrence
  - **0:n**: zero or more occurrences
  - **1:n**: one or more occurrences
  - **m**: *m* is a integer value designating a fixed number of occurrences
- **Ordering**:
  - **O**: the ordering of the elements in the data model instance is significant
  - **U**: the ordering of the elements in the data model instance is not significant
- **Representation**: The data model or (abstract) data type of the data element.
- **Notes**: Additional descriptive information.

Hierarchies of elements (submodels) are not explicitly represented in the tabular form. Each
submodel is presented as its own data model or structure that can then be reused as the representation for any other model element. Binding of a nested model to an actual data structure is not determined by the data model, e.g., the parent structure may contain a reference to an instance of the nested data model; the element in the parent structure may be a complex data item that represents the nested data item, or the elements may be flattened into a simple normalized form.

Nothing precludes use of complex data models as elements of a data model to create self-referential or cyclic structures.

Unless otherwise specified, the ordering of the elements in a data model is insignificant and simply the result of presentation in the tabular form. A binding should not impose an ordering of the elements of the model unless explicitly specified.

An example of the tabular form is shown below. Several of the column headers are abbreviated to improve the layout.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>Boolean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Integer</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decimal</td>
<td>Exact numeric value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Float</td>
<td>Float or real number, approximate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Date/time, specific instance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Duration</td>
<td>Date/time range</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Octet String</td>
<td>String of octets</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Character String</td>
<td>String of characters, language undesignated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lang String</td>
<td>String of characters, language designated</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>URI</td>
<td>String of characters interpreted as a URI (URI/URL/URN)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Core Data Types

The data model descriptions start with the following core or base data types. These base types are not defined further herein. Many of the data types are similar to those used in XML Schema (http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/).
The speculations require a description of content, both as individual elements, and as collections or content aggregations. The models that describe the content elements and content structure follow. These include elements to describe the behavior of the content when delivered, but the details of the behaviors are outside of the scope of the structural description of the elements themselves. The content models only describe the learning content and its organization. They do not include elements or data models needed to instantiate the content for delivery (e.g., tracking, history, scores).

As noted, the focus is on a framework to represent the content. Both the terms content and structure should be interpreted in their most general sense. Content is the collection of materials and behaviors used to describe a learning experience. Structure is a logical organization, not a particular physical representation of the content. How one uses the content and structure to build and deliver a learning experience is not specified.

The overall content structure model is described in a top-down fashion. Most elements of the model are instances of other models, i.e., an abstract data type. The higher-level models are independent of the details of the referenced elements. This produces a structure for the speculations where the lower-level elements can be defined outside of their use in the content structure, and the referencing models are immune to changes in the lower-level models. For example, some content elements reference prerequisites. Prerequisites are defined as an independent model. The Prerequisite Model may be a simple list of elements, or it may be a complex boolean relationship on a set of prerequisites. The referencing content element is independent of these alternatives. This approach is designed to permit modifications or extensions to elements of the overall model without breaking other parts of the structure.

The overall model is organized into four levels:

- An aggregation, consisting of a collection of objects. Each element of the collection is:
  - A contextualized object, consisting of a single reusable content object within a specific learning context with associated control methods. This content object is:
    - A purposed object, consisting of a single reusable content object with a specific educational purpose. The content object is either a collection (an aggregation) or a basic learning object. A basic object is:
      - A learning object with associated learning resources. It may be reused in different ways to fulfill different purposes. Its internal structure is opaque.

Each model has associated identification data. Thus, each instance of an element of the content
structure could be independently reusable in the proper context, and can be referenced by identifier. For example, the same basic learning object can appear in different purposed learning objects, each with different competencies, or a different purposed object can be used in different contexts with different controls.

The framework assumes that a Content Handle system will be used to resolve all references to actual content object instances. Content handles abstract alternative selection, versioning and variants from the overall content framework.

A binding of the content model may choose to ignore part of the conceptual structure. For example, a binding could combine a conceptualized learning object and a purposed learning object into a single referenced and identified physical entity.

The association of data in the model to instances is also not specified, and should not be implied. Any items, e.g., roles, pre- and post-processing behaviors, all could be instantiated with particular values at content delivery time on a per instance basis, and once instantiated, value could be changed. Such overall use of the model is under the control of the associated environment and the values of the behaviors in the model. No other constraints on behavior or use of the framework should be implied.

The Sequencing Light proposal can be described as a specific instance of the data model. Notes describe how to map this particular sequencing approach to the overall content representation model.

The overall model is illustrated below (to enlarge click on the image for a .pdf file).

Legend:

- Content objects (models) are shown in yellow.
- Other models are shown in green.
- Submodels are shown in light blue.
- Vocabularies are shown in red.
- Double boxes indicate required handle resolution.
Aggregation Learning Object Model

The Aggregation Learning Object Model describes a collection (aggregation) of learning content used within a particular context. It associates behavior, and rendering information with a collection of contextualized learning objects.

The *collection* represents the organization of the content elements in the aggregation. The collection is used by the processing and rendering methods, but the collection is not the implied view or sequence for the content.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifying information for the object</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Object ID Submodel</td>
<td></td>
</tr>
<tr>
<td>Aggregation Type</td>
<td>The type of aggregation learning object</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Aggregation Learning Object Type</td>
<td>See below.</td>
</tr>
<tr>
<td>Learning Content Collection</td>
<td>Description of the content elements within the learning aggregation</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Aggregation Collection Model</td>
<td>E.g., a graph representation. Elements of the collection are content objects or handles of content objects.</td>
</tr>
<tr>
<td>Rendering</td>
<td>Description of the render information used to display the content in the aggregation</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Rendering Model</td>
<td>Value is inherited from the parent if missing.</td>
</tr>
<tr>
<td>Sequencing Behavior</td>
<td>Description of the process used to sequence the content in the aggregation</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Sequencing Model</td>
<td>Value is inherited from the parent if missing.</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>Description of the pre-processing methods applied before the learning aggregation can be delivered</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Pre/Postprocessing Model</td>
<td></td>
</tr>
<tr>
<td>Postprocessing</td>
<td>Description of the post-processing methods applied after the learning aggregation is delivered</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Pre/Postprocessing Model</td>
<td>E.g., a roll-up rule.</td>
</tr>
<tr>
<td>Metadata</td>
<td>The metadata for the aggregation learning object</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Metadata Object Model</td>
<td></td>
</tr>
</tbody>
</table>

The *Aggregation Type* item is used to differentiate kinds or classes of aggregations. Tokens of the vocabulary are not specified. Behavior and pre- or post-processing methods can use the *Aggregation Type* to control the content delivery experience.

**Mapping to Sequencing Light**

- The Sequencing Light content structure is a simple ordered list of content elements. The *Aggregation Collection Model* used to represent the content collection would be a simple
ordered list of content objects (Learning Object Model instances or Aggregation Learning Object Model instances). The details of the Aggregation Collection Model would determine if Order is explicit or implicit in the representation.

- The Sequencing Type in Sequencing Light corresponds to the Sequencing Behavior element. The Sequencing Model used to represent the Sequencing Behavior would be the Sequencing Type vocabulary.
- The Completion Status Rule in Sequencing Light corresponds to the Postprocessing element. The Pre/Postprocessing Model used to represent the Postprocessing would be the Completion Status Rule vocabulary.

## Contextualized Learning Object Model

The Contextualized Learning Object Model describes an element of learning content used within a particular context. It associates context, use role, resources, and pre- and post-conditions with a purposed learning object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifying information for the object</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Object ID Submodel</td>
<td></td>
</tr>
<tr>
<td>Context</td>
<td>The learning context</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Learning Context Model</td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>Role of the learning object in the context</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Learning Object Role Vocabulary</td>
<td>See below.</td>
</tr>
<tr>
<td>Learning Content</td>
<td>The learning object used in the context</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Purposed Learning Object Model</td>
<td>May be either a basic object or an aggregation.</td>
</tr>
<tr>
<td>Associated Resources</td>
<td>The resources required to use the learning object in this context</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Resources Model</td>
<td>Resources are additional requirements, such as a tutor, a simulator, a chat service. It is not the resource elements of an IMS Content Package.</td>
</tr>
<tr>
<td>Preprocessing</td>
<td>Description of the pre-processing methods applied before the learning object can be delivered in this context</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Pre/Postprocessing Model</td>
<td></td>
</tr>
<tr>
<td>Postprocessing</td>
<td>Description of the post-processing methods applied after the learning object is delivered in this context</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Pre/Postprocessing Model</td>
<td>E.g., a next rule.</td>
</tr>
<tr>
<td>Metadata</td>
<td>The metadata for the contextualized learning object</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Metadata Object Model</td>
<td></td>
</tr>
</tbody>
</table>

The *Role* item is used to differentiate how the content should behave in a particular context. For example, a content object that is an assessment might take on the role of *pretest*, *posttest*, *practice* or *certification*. Values of the vocabulary are not specified. Pre- or post-processing methods can use the role to control the content delivery experience.
Mapping to **Sequencing Light**

- The *Next Rule* in Sequencing Light corresponds to the *Postprocessing* element. The *Pre/Postprocessing Model* used to represent the *Postprocessing* would be the *Next Rule* data model (the unordered collection of sequence branching descriptions).

**Purposed Learning Object Model**

The Purposed Learning Object Model describes a learning object with a particular educational purpose, described via an associated set of objectives, competencies and other learning outcome-specific information. This information is layered on a learning object (either a basic learning object or an aggregation learning object).

The learning content is a basic object or aggregation that could be reused in different contexts and for different learning purposes. Converting learning content from that which is *purposed* is done by layering the additional contextual information on the object. This information is placed in the object model as described.

Alternatively, the Metadata Model for the objects could be extended to encompass the additional information about purpose. Both alternatives are functionally equivalent and the framework is neutral to how an actual binding will treat this information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifying information for the object</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Object ID Submodel</td>
<td></td>
</tr>
<tr>
<td>Learning Content</td>
<td>The learning object</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Learning Object Model or Aggregation Learning Object Model</td>
<td>The actual object is resolved via a content handle.</td>
</tr>
<tr>
<td>Objectives</td>
<td>The objectives associated with the learning object for this purpose</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Objectives Submodel</td>
<td></td>
</tr>
<tr>
<td>Competencies</td>
<td>The competencies associated with the learning object for this purpose</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Competencies Submodel</td>
<td></td>
</tr>
<tr>
<td>Prerequisites</td>
<td>The prerequisites associated with the learning object for this purpose</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Prerequisites Submodel</td>
<td></td>
</tr>
<tr>
<td>Metadata</td>
<td>The metadata for the purposed learning object</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Metadata Object Model</td>
<td></td>
</tr>
</tbody>
</table>

Mapping to **Sequencing Light**

- The *Simple Prerequisites* in Sequencing Light corresponds to the *Prerequisites* element. The *Prerequisites Submodel* used to represent the *Prerequisites* would be an unordered list of zero or more content objects.

**Learning Object Model**
The Learning Object Model describes an individual learning object (not an aggregation), i.e., an element of content that has specific educational characteristics and can be reused. The learning object is **opaque** in that the internals of the object are not exposed to any control process.

The object metadata describes a single set of educational characteristics of the core learning or content object, i.e., it describes only what the object is. The metadata does not describe how the object is used in a learning experience, i.e., it does not associate objectives or competencies with the learning object.

The content handle provides the reference to the content. The handle is processed by a resolution service to access an instance of the content.

The **Learning Object Type** item is used to differentiate between content objects that behave in different ways. For example, the content object might be a **SCO** that uses the API for communications and to pass state to the control process, or it might be **dumb** content that does not communicate with the control process. **Learning Object Type** can be used to control the content delivery experience. Tokens of the vocabulary are not specified. The **Learning Object Type** information could be embedded in the **Metadata**.

### Content Delivery Process

The content delivery process begins with an aggregation learning object, instantiated for the learner. Somehow, via an interface to the delivery environment, this particular content object is designated for delivery to the learner. The delivery process is:

- Invoke the designated **preprocessing** methods of the learning object instance.
- Using the designated **sequencing behavior** of the learning object, sequence the delivery of the elements of the object’s content **collection**.
  - An individual content element is selected from the learning object’s **collection** according to the **sequencing behavior** of the learning object. The selected object will be a **contextualized learning object**.
    - Invoke the designated **preprocessing** methods of the learning object instance.
    - Resolve the handle of the content object instance. The **contextualized learning object** instance will resolve to a **purposed learning object** instance, which in turn will resolve to either an aggregation learning object instance or a learning object instance.
    - If the content object resolves to an aggregation learning object, recursively process the instance of the content object.
    - If the content object resolves to a learning object, deliver the content object to the learner.
    - Invoke the designated **postprocessing** methods of the learning object instance.
    - Repeat selection process until the **sequencing behavior** of the learning object terminates.
- Invoke the designated **preprocessing** methods of the learning object instance.
Speculations: Behavior Models

Daniel R. Rehak, PhD  
Technical Director  
rehak@cmu.edu

William H. Blackmon, PhD  
System Architect  
huit@cmu.edu

Learning Systems Architecture Laboratory  
Carnegie Mellon University

Overview

The behavior models describe different behaviors (pre-, post-conditions, rendering, etc.) associated with content objects. In a traditional LMS, these models and behaviors are typically hardwired into the system. Approaches to providing formal descriptions of models as part of a content modeling and sequencing solution vary from simply listing a collection of different explicit alternatives to providing a generalized model definition language.

The current speculation describes a **simple** declarative version for these models. The nature of the overall content model is that these models can be changed or replaced with more comprehensive models without impacting the overall logical organization of the content model.

In this simple version, the models are just a vocabulary of names. The associated behaviors are defined explicitly for each alternative, i.e., they remain hardwired with implicit behavior, but the overall content representation framework provides the mechanism to chose a model from the explicitly named set, rather than accepting an overriding single implicit model.

A more comprehensive model would include multiple behaviors for rendering, sequencing and pre/postprocessing. Each of the models would include not only the collection of behaviors but how the information is used to select or sequence the alternatives.

Rendering Model

The Rendering Model describes the skins or window display environment for presenting the content and a content style model. The skin or *outer environment* describes the overall framework, e.g., windows, menus, controls, and the content style describes how to display a content object within the outer environment, e.g., color scheme, fonts, backgrounds. No further details are provided at this time.

<table>
<thead>
<tr>
<th>Rendering Model: Simple Vocabulary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>Skin Model Name</td>
</tr>
</tbody>
</table>

Behavior is explicitly associated with a particular token from the vocabulary.
**Sequencing Model**

The Sequencing Model describes how to select an element from the content aggregation for presentation to the learner. The Sequencing Model could be just either a procedural representation (possibly just the name of the method), or it could be a declarative collection of information that augments the rest of the content model and is used in conjunction with the method.

The simple sequencing model is based on the vocabulary of sequencing types defined in the **Sequencing Light Proposal**.

**Mapping to Sequencing Light**

The **Model Name** would designate the vocabulary as the **Sequencing Light Sequencing Type**. The **Sequencing Light Sequencing Type Model Name Vocabulary** consists of the following tokens. The behavior of each is detailed in the **Sequencing Light Proposal**.

- User Sequencing
- Ordered Sequencing
- Successful Sequencing
- Prerequisites Sequencing
- Next Rule Sequencing

**Pre/Postprocessing Model**

The Pre/Postprocessing Model describes what actions to take before delivering an element of content (either an element within an aggregation, or a contextualized object) and what actions to take after the content is delivered. The simple Pre/Postprocessing Model is based on the vocabulary of behaviors defined in the **Sequencing Light Proposal**.

The model is instantiated separately for object pre- and post-conditions. In the simple model, different vocabularies would be used for each case and each object type.
Mapping to Sequencing Light

- The Completion Status Rule in Sequencing Light corresponds to the Postprocessing element. The Model Name would designate the vocabulary as the Sequencing Light Completion Status. The Sequencing Light Completion Status Model Name Vocabulary consists of the following tokens. The behavior of each is detailed in the Sequencing Light Proposal.
  - Explicit passed
  - Explicit failed
  - Explicit completed
  - Explicit incomplete
  - Last
  - Passed all
  - Passed or completed all
  - Passed any
  - Failed all
  - Failed any
  - Completed all
  - Completed any

- The Next Rule in Sequencing Light corresponds to the Postprocessing element. It requires a more complex model.

### Postprocessing Model: Next Rule

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifying information for the object</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Object ID Submodel</td>
<td></td>
</tr>
<tr>
<td>Rule Set</td>
<td>Collection of next rules.</td>
<td>R</td>
<td>0:N</td>
<td>U</td>
<td>Rule (See below)</td>
<td>Behavior is explicitly associated with the Sequencing Light Next Rule model.</td>
</tr>
</tbody>
</table>

Each rule is modeled as two parts, a condition and an action. The condition is based on the vocabulary of behaviors defined in the Sequencing Light Proposal. The action is either a learning object or it is a token based on the vocabulary of behaviors defined in the Sequencing Light Proposal.

### Rule Submodel: Next Rule

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule Condition</td>
<td>Result status to match</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Status Vocabulary token</td>
<td></td>
</tr>
<tr>
<td>Rule Action</td>
<td>Next item in sequence.</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Content Handle Submodel or Special case vocabulary token (See below)</td>
<td>If the action is to branch to a specific content object, the object must resolve to an object within the same content aggregation.</td>
</tr>
</tbody>
</table>

The Status Vocabulary consists of the following tokens. The behavior of each is detailed in the Sequencing Light Proposal.

- Ab initio
- Not attempted
- Browsed
- Passed
- Failed
- Incomplete
- Completed
- Other
The *Special Case Vocabulary* consists of the following tokens. The behavior of each is detailed in the *Sequencing Light Proposal*.

- Exit
- Exit All
- Exit User
- Exit Top
- First
- Last
- Random
Overview

The speculations include a set of basic supporting models and submodels that are used in other data models. These are essentially abstract data types, or collections of elements used with other models. Instances of these submodels are not full objects, i.e., they do not have identification data and are not designed for reuse.

There are different possible structures for many of the submodels. The current speculation describes a simple version for most of the submodels. As illustration, a more comprehensive alternative is outlined for most submodels. The nature of the overall content model framework is such that the structure of these submodels can be changed without impacting the overall logical organization of the content model.

Object ID Submodel

Individual instances of data models need to be uniquely identified. The Object ID Submodel provides a common set of elements that can be attached to any other data model to provide a mechanism for identifying the data model instance.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUID</td>
<td>A globally unique identifier for the object</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>URI</td>
<td>The GUID scheme is defined externally.</td>
</tr>
<tr>
<td>Object ID</td>
<td>A human readable identifier for the object</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Lang String</td>
<td>The ID scheme is not specified. Any language may be used.</td>
</tr>
<tr>
<td>Data Model Version</td>
<td>The version of the data model of the object</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Char String</td>
<td></td>
</tr>
</tbody>
</table>

A more comprehensive Object ID Submodel might include more object descriptions and data model versioning information.
Competencies Submodel

The Competencies Submodel describes the collection of competency definitions and a satisfaction condition. A Sequencing Behavior or Preprocessing Model can use the status of the satisfaction condition for an instance of a learning object as part of a control process to select and deliver content. The satisfaction condition is based on status or other data from the competency records associated with each competency. The postprocessing method can use the results or state returned from delivery of a learning object to set the status or competency record for an individual competency instance for the learner.

The Competencies Submodel describes only the competencies and satisfaction condition. The submodel is not used to record satisfaction, and does not include the model of competency records. The Competencies Submodel is used to associate competency requirements with individual learning objects. It does not describe an overall competency framework for learning or a competency driven learning program.

In the simple Competencies Submodel, the competencies are a list of competency definitions. In the simple Competencies Submodel, the satisfaction condition is not specified, but is assumed to be implicit in the behavior model, e.g., the behavior model might assume that each (i.e., all) competency must be satisfied individually for the learning object competencies to be satisfied. Note: defining the satisfaction criteria is implicit in the behavior model or method.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Competency</td>
<td>A reusable competency definition</td>
<td>R</td>
<td>1:N</td>
<td>U</td>
<td>Competency Definition Model</td>
<td>The Reusable Competency Definition Model is defined externally.</td>
</tr>
</tbody>
</table>

A more comprehensive Competencies Submodel might specify how to combine individual competency record values via a logic statement. It would include the set of competency definitions, the logic to combine the values from different competency records to determine satisfaction. Satisfaction includes not only how to compute the value, but how to interpret the value, e.g., the learner has the required competency if they have the collection of competency A with grade X and competency B with grade Y and competency C with grade Z, or alternatively, they have achieved competency D within the last year.

Objectives Submodel

The Objectives Submodel describes the collection of learning objectives and a satisfaction condition. A Sequencing Behavior or Preprocessing Model can use the status of the satisfaction condition for an instance of a learning object as part of a control process to select and deliver content. The satisfaction condition is based on status or other data from the individual objectives.

The Objectives Submodel describes only the objectives that might be associated with a learning object, not the model used to record satisfaction, or the model to define the objectives themselves.

In the simple Objectives Submodel, the objectives are a list of other learning content objects. In the simple Objectives Submodel, the satisfaction condition is not specified, but is assumed to be implicit in the behavior model, e.g., the behavior model might assume that each (i.e., all) objective must be satisfied individually for the learning object objectives to be satisfied. Note: defining satisfaction is implicit in the behavior model or method.
A more comprehensive Objectives Submodel might combine individual objective values via a logic statement and satisfaction condition.

**Prerequisites Submodel**

The Prerequisites Submodel describes the collection of prerequisites and a *satisfaction* condition. A Sequencing Behavior or Preprocessing Model can use the status of the satisfaction condition for an instance of a learning object as part of a control process to select and deliver content. The satisfaction condition is based on status or other data from the individual prerequisites. The postprocessing method can use the results or state returned from delivery of a learning object to set satisfaction conditions and to propagate prerequisite completion.

The Prerequisites Submodel describes only the prerequisites themselves, not the model used to record satisfaction.

In the simple Prerequisite Submodel, the prerequisites are a list of other learning content objects. In the simple Prerequisite Submodel, the satisfaction condition is not specified, but is assumed to be implicit in the behavior model, e.g., the behavior model might assume that each (i.e., all) prerequisite must be satisfied individually for the learning object prerequisites to be satisfied. Note: defining *satisfaction* is implicit in the behavior model or method.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisite</td>
<td>Prerequisite learning object</td>
<td>R</td>
<td>1:N</td>
<td>U</td>
<td>Learning Object Model or Aggregation Learning Object Model</td>
<td></td>
</tr>
</tbody>
</table>

A more comprehensive Prerequisites Submodel might combine individual prerequisite values via a logic statement and satisfaction condition.

**Mapping to Sequencing Light**

The simple list model is a direct mapping and provides the data model for *simple prerequisites*. Prerequisites are associated with individual content elements (basic elements or aggregations).

**Handle Submodel**

The simple Handle Submodel is just a single URN or equivalent encoding of the handle string. The associated handle resolution service will process the handle to return one or more locations or other information. The client would then select the appropriate alternative.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle</td>
<td>The actual handle</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>URI</td>
<td></td>
</tr>
</tbody>
</table>

A more comprehensive version of a Handle Submodel would represent constraints and other information needed to select or filter the alternatives returned to the client from the resolution.
The resolution process begins with the **Start Handle**. The set of resolved results are filtered. All or a subset of results are returned to the client. If the filter process returns no results, then the **Default Handle** is resolved and its unfiltered results are returned to the client.

The **Resolution Type** is used to control the overall resolution process. For example, it might be appropriate to resolve the handle to a final object, or it might be appropriate to resolve only to the location handle (resolving all alternatives). The tokens of the vocabulary are not fully specified. A sample vocabulary might include the tokens:

- **Version**: Resolve to a single version.
- **Variant**: Resolve to a single variant.
- **Alternatives**: Resolve to a single alternative (version and variant).
- **Location**: Resolve to the location handle.
- **Full**: Full resolution.
- **Single**: Resolve only one step.

Each object has a set of attributes and values that differentiate the alternatives, e.g., version is an attribute, and values of the version attribute would be the version numbers. Objects that match the filter are returned.

### Filter Submodel

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter Attribute</td>
<td>The name of an attribute of an object</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Filter Attribute Vocabulary</td>
<td></td>
</tr>
<tr>
<td>Filter Value</td>
<td>The name of a value of an attribute of an object</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Filter Value Vocabulary</td>
<td>Vocabulary is dependent on the Filter Attribute Vocabulary.</td>
</tr>
</tbody>
</table>

Tokens of the **Filter Attribute** and **Filter Value** vocabularies are not specified.

### Aggregation Collection Model

An Aggregation Collection is the logical organization of the Contextualized Learning Objects in an Aggregation Learning Object. The Aggregation Collection Model describes the organization of content elements that are part of a content aggregation. The content objects may have an
arbitrarily complex organization, e.g., they may be a directed graph of objects.

The structure of the graph does not imply a particular content presentation or delivery sequence. The aggregation collection is used by the Sequencing Behavior Model and the Rendering Model to determine which elements to present and how to present them to the learner. One behavior might use all of the content objects in the aggregation collection as part of the learning experience or as the content table of contents. A different behavior might dynamically select content according to the objects in the aggregation.

It is important to recognize that the elements of the collection always map to an actual content object (again, either a basic content object or a complex aggregation) via a content handle. Thus the aggregation collection can model proxies for content, and the handle resolution strategy can (dynamically) convert the proxies to actual content objects for delivery.

Two alternative models are presented: a simple list structure and a more complex graph structure. The simple list model is the common approach of a fixed, ordered structure of content in the aggregation. The Contextualized Learning Object in the aggregation collection can represent either a simple object or a content aggregation.

The more comprehensive model represents a complete directed graph of content objects, by self-referencing the Aggregation Collection Model. This permits modeling of content hierarchies directly within a single aggregation, rather than requiring multiple aggregation objects. The model recursively references itself to create any kind of structure. Leaf or terminal nodes of the graph are Contextualized Learning Objects.

The Aggregation Collection Type Vocabulary consists of the following tokens:

- List: The Content is an ordered set of Contextualized Learning Objects
- Bag: The Content is an unordered set of Contextualized Learning Objects
- Tree: The Content is an ordered set of Contextualized Learning Objects or Aggregation Collection Models (directed graph version).

The model can also support logical combinations by using type AND, OR, XOR, or NOT.
Note: the model is defined in a self-referential form. This does not imply that instances of the model are created; the model could be expanded to define a formal graph structure, but this transform is part of the model binding.

**Mapping to Sequencing Light**

The simple list submodel is a direct mapping and provides the data model for collection of content in the aggregation. The data submodel represents the *ordering* attribute through the definition of the submodel; a binding could introduce a specific ordering attribute.

**Metadata Model**

The current Metadata Model is the IEEE LOM model. The model is not detailed further. Proposed extensions and modifications from IMS, EASEL, etc., need to be considered.

A more comprehensive Metadata Model would include different types of metadata, e.g., LOM versus DCMI versus MPEG7, either as part of an individual instance or as a combined structure.

**Resources Model**

The Resources Model describes a set of learning resources that are used or required for delivery or processing of a learning object. For example, resources could be a chat room, text book, simulator software, grader, etc. How resources are used to control learning content delivery is not specified by the Resource Model. The Sequencing Behavior and Preprocessing Models could use resource availability information as part of a control process to select and deliver content.

The Resources Model describes only the set of resources themselves, not the model used to determine resource availability or use.

In the simple Resources Model, the resources are a list of handles of resource objects. The handles can resolve to digital or non-digital objects. Use of the Resources Model is not specified but is assumed to be implicit in the behavior model.
Resource Type to control the content delivery experience.

A more comprehensive Resources Model would include different collections of resource options, and alternatives and conditions for use of each option.

**Learning Context Model**

The Learning Context Model describes the environment and contextual parameters needed to deliver or process a learning object. Details of the context model are not specified.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Ob</th>
<th>Mult</th>
<th>Ord</th>
<th>Representation</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifying information for the object</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Object ID Submodel</td>
<td></td>
</tr>
<tr>
<td>Learning Context Type</td>
<td>The type of learning content</td>
<td>R</td>
<td>1</td>
<td>U</td>
<td>Learning Context Object Type Vocabulary</td>
<td>See below.</td>
</tr>
<tr>
<td>Description</td>
<td>A description of the learning context</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Lang String</td>
<td></td>
</tr>
<tr>
<td>Context Definition</td>
<td></td>
<td>R</td>
<td>1:N</td>
<td>U</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Metadata</td>
<td>The metadata for the Context Object</td>
<td>O</td>
<td>1</td>
<td>U</td>
<td>Metadata Object Model</td>
<td></td>
</tr>
</tbody>
</table>

The Learning Context Type item is used to differentiate types or classes of Learning Contexts. Tokens of the vocabulary are not specified. Behavior and pre- or post-processing methods can use the Learning Context Type to control the content delivery experience.
Overview

The requirements and framework are based on and have been informed by a collection of activities and the review of a variety of materials as outlined below. The description is primarily a list of activities, not a detailed work review.

Current Specification Review

SCORM and the current CSF are based on a collection of interrelated specifications and draft standards that are still undergoing changes. These include:

- ADL SCORM V1.1 (including CSF)
- ADL Content Packaging Profiles
- IMS Content Packaging Specification V1.0 and V1.1
- IMS Content Management Specification V1.0
- IMS Content Communications Specification V1.0
- IMS GUID Paper
- IMS Competency Definition Specification
- IMS Learning Information Package Specification
- IMS Question Test Interoperability Specification
- AICC Computer Managed Instruction Guidelines
- IEEE LTSC Computer Managed Instruction Draft Standard
- IEEE LTSC Learning Object Metadata Draft Standard

Ongoing review of the evolution of this work and participation in work groups have helped to determine specific current problems, if and where there may be substantial problems that would limit development of a replacement CSF, and how a candidate replacement CSF would interface with the ongoing work. The limitations of the current CSF of particular concern are:

- Ad hoc model for representing content flow and sequencing.
- Embedding of behavioral and sequencing control in content.
- No model for representing skins (i.e., content and interface rendering).
- Need for contextualized versus learning context-free metadata.
- Lack of support for content versioning (for personalization, accessibility, internationalization, etc.).
- Harmonization and an upgrade path from the existing packaging model to one that incorporates different types of learning content, services and models.
- Need for a communications model that extends from the current simple SCO-to-LMS data
Learning Model Literature Review

A limited literature review of different learning models and reuse of learning objects, including informal discussions with authors, has been conducted. There are no specific outcomes.

A limited review of current approaches to intelligent tutoring, simulation and adaptive content within learning systems was conducted. This included an exploration of a basic scheme for incorporating such adaptive content into CSF via extension (including pre- and post-condition competency requirements on each node, adding control rules, and adding overall control algorithms for learning, e.g., goal seek to find content to satisfy required competency, forward chain competency results through the content map).

A review of Gagne's seminal work in instructional design was conducted to refocus on the meaning of learning in learning objects. Many of today's systems are modified versions of previous systems. The review of Gagne's work will help us to identify weaknesses in current systems that are due to technological shortcuts of the 1980's and 90's; the hope is that CLEO will break out of any ruts that current systems may have found themselves in.

Approaches to Extending CSF and SCORM

Several content and LMS vendors have written white papers describing their needs and desires for future versions of CSF and SCORM or have developed solutions to some problems. This work has been reviewed, including limited discussions with these parties.

- Review of a white paper from Peer3, *Pushing the SCORM Envelope*. The paper advocates an XML approach for declarative content representations to help identify presentation and behavior, along with a dynamic content delivery services model.
- Review of model for content management used by Peer3. Review to assess if the model contains the needed elements to separate content from behavior and presentation to enable content reuse in different learning scenarios.
- Discussions with Avilar about their *SCORMFront* approach to providing a declarative model for content presentation and separating content from presentation.
- Review of white paper from Avilar, *An Extensible Runtime Data Model Capability for SCORM*. The paper describes an approach to permitting content to use any data model.
- Review of white paper from KnowledgePlanet, *Towards an Adaptive SCORM*. The paper advocates a server-centric (versus content-embedded) model of adaptive content delivery, based on a core collection of instructional models.
- Review of Cisco's RIO/RLO model of content structure and separation of content elements from control.
- Review of the ALIC (Advanced Learning Infrastructure Consortium) model of content, content sequencing, and content control used in adaptive learning systems.
- Review of Click2Learn's IMS submission on *Playrules*. This presents one model of content structure to provide a better approach to sequencing, but is based on a direct encoding of the instructional model in the rules (but not in the content). Thus while the control is not opaque in the learning objects, the instructional model is still opaque in the overall rule set.
- Review of Click2Learn's IMS submission on *Fancy Content*. The proposal addresses a mechanism for content to override the generic navigation controls of an LMS, or to provide the ability for content authors to include navigation controls as part of their visual design, instead of relying on the generic controls of the LMS.
- Review of Click2Learn's and Microsoft's IMS submission on *Content Variants*. Content variants address how to incorporate *learning equivalent* but technically different content within an IMS Content Package.
Run-Time Environment. The white paper presents a model of using SOAP as a replacement for the communications API.

- Review of Online Courseware Factory's white paper, Template Method For Designing and Delivering Learning Objects.
- Review on NYU Online's white paper, Feedback to the ADL on the SCORM Standard.

Alternative Models for Content Representation and Behavior

Various alternative approaches for content representation and behavior have been developed. This work has been reviewed, including limited discussion with some of the parties.

- Review of LML (Lesson Markup Language) as an approach for declarative models for simulation content sequencing and control.
- Review of EML (Educational Markup Language) as an alternative approach to declarative models for content presentation and control.
- Review of PML (Procedural Markup Language) as an approach for defining structure of learning objects and their use.
- Review of LMML (Learning Material Markup Language) as an alternative model for declarative models for content structure.
- Discussions on intelligent tutoring systems with Carnegie Learning.
- Discussions on intelligent tutoring systems with University of Pittsburgh Learning Research Development Center.
- Review of David Wiley's book on Learning Objects including discussions with the author.
- Review of Allen Munro's work on simulations.
- Review of OKI (Open Knowledge Initiative), a proposed open-source framework for learning management.
- Review of Wes Reigan's work on Intelligent Tutoring Systems, including interpretations by Eric Roberts.
- Review of EASEL model on extending metadata, including a focus on metadata for adaptive learning.

Workshop on Requirements for Representing Adaptive Simulation

A workshop on requirements to support simulation and adaptive learning was conducted on January 31, in conjunction with the AICC January 2001 meeting. Attending: Dan Rehak (Carnegie Mellon), Bill Blackmon (Carnegie Mellon), Philip Dodds (ADL), Eric Roberts (ADL), Tyde Richards (IBM Mindspan), Claude Ostyn (Click2Learn), Thor Anderson (IMS), Kris Rockwell (Usairways), Mark Winter (Simulis), Allen Munro (USC), Tom King (Macromedia), Jean-Louis Bravo (AirBus), Ben J. Drinkwater (Flight Safety Boeing), Luc Ethier (CAE), Gilbert Deziel (CAE), Schawn Thropp (CTC/ADL), Rob Ball (CTC/ADL), Jack Hyde (Flight Safety Boeing), Avi Weiss (Simulis), Joselyn Mane (IBM Mindspan), Paul Munro (Apren), Jay Dempsey (Starmountain), Bill MacDonald (Flight Safety Boeing). Discussion topics included:

- What are exemplars of content reuse in simulations?
- What is a SCO for simulation?
- How is a SCO used in simulation?
- What are requirements for sequencing simulation SCOs?
- What information and data does a simulation SCO need to separate control from content?

CLEO Retreat

A weeklong retreat of all the CLEO participants was held from February 26 to March 2, 2001, to discuss the project goals and work. Topics included:
- Review of status and current directions from all participants.
- Development of learning scenarios.
- Review of requirements versus learning scenarios.
Working Notes

This report consists of a collection of working notes related to the development of a candidate replacement for the CSF specification. Each working note is a separate document.

Content Aggregation

The Content Aggregation note describes some of the issues associated with creating and describing a content aggregation and sequencing the delivery of objects within the aggregation.

Sequencing Light

The Sequencing Light note describes a simple set of extensions to SCORM content aggregations to support the basic sequencing of the items (SCOs or aggregations) in the aggregation. The objective is to describe a common collection of sequencing behaviors with minimal impact on the current CSF, CMI, IMS Content Packaging, and SCORM data models to meet the immediate sequencing needs of SCORM 1.2/1.3. The solution is not meant to outline a direction for addressing content aggregation and sequencing in SCORM 2.0+ or to preclude any alternative solutions in the future.

Temporal Issues

The Temporal Issues note describes the temporal relationships between events that occur when a student interacts with one or more learning objects. Understanding these relationships is important to creating instances of tracking data for different events during content delivery.

Content Handles

Handles are a way to create persistent identifiers that allow the physical location of objects to move or different objects to be returned for a single identifier. Handles can abstract a collection of version, variation and preference functions from a content model. The Content Handles note
describes how to apply handles for this use.
Overview

This note describes some of the issues associated with creating and describing a content aggregation and sequencing the delivery of objects within the aggregation.

Introduction

No one model of content behavior or object representation will be acceptable to everyone. Content aggregations must be able to combine content objects with different behaviors, representations and structures. This requires an open framework to author and describe the aggregations, while letting the internals of any content object within the aggregation be private. The sequencing problem is only at the interface between the objects in the aggregation, not within any single object. Sequencing passes control of presenting the learning experience between the objects of the aggregation. You turn control over to an object, and it does what it was designed to do. All you care about is understanding the communications between the objects and the aggregation, not what or how an individual learning object does what it does.

Within a conventional LMS, sequencing and behavior consists primarily of maintaining state for a known set of data models instantiated for each content object, and letting content objects at different levels within a content structure share access to these data models via a known communications protocol within an overriding global control strategy.

SCORM 1.x hardwires a single set of data models, a particular hierarchical content structure and a single communications protocol together with a somewhat ambiguous control strategy.

Goal

To be able to combine existing, diverse content objects that use different data models and protocols to create other content objects, which can then be used to deliver a learning experience.
Content (aggregations) is represented as a nested collection of content objects. This is consistent with Carnegie Mellon Online, SCORM, Tabula, OKI, LRN, etc. There appears to be sufficient consensus on this representation and it appears to have the necessary flexibility in what it can represent such that we can accept it as a baseline.

There are many different content objects; CSF objects (e.g., SCOs), CMI objects, QTI objects, simulation objects, mime-type objects (i.e., dumb content), Flash objects, etc. The set of types of objects is not closed.

Each content aggregation includes a *structure* (or *collection*) that includes zero or more content objects (aggregations). Self and cyclic inclusion can be excluded in the simple case, but cyclic structures are probably needed for dynamically scoped, self-adaptive learning.

A content aggregation is an object of type *aggregate*, i.e., it is not some other well known and well defined content type.

Declared content aggregation structures are lexically scoped, defined at content aggregation authoring time. Instantiated content objects are dynamically scoped, independent of the lexical scope. (This is probably controversial, or not well understood.)

Each content aggregation or content object uses a set of data models to describe state, tracking, etc., e.g., CMI model, LIP model, IMS CM micromodels. The overall set of data models is not closed. A particular type or authored instance of a content object will use a fixed (known) set of data models.

Each content aggregation has its own sequencing method that is applied to the elements of the aggregation, e.g., linear, user choice, adaptive, scripted, rule-based. The overall set of sequencing models is not closed. A particular type of content object will use a fixed set (known) sequencing models (most likely just one).

How a single content object treats or uses the data models in combination with sequencing is private to that object or object type (e.g., is it scripted or interactive, is it static content, does it use a global intelligent controller, does it use play rules). Instantiated content has a private internal state and behavior.

Behavior and sequence control is all about propagating data and control messages between levels of the aggregation structure. It is complex because the objects are different.

Creating and using aggregations of different types of objects is focused on discovery and mapping of data between the levels or different types of learning objects in the aggregation structure.

A content object or aggregation must be able to discover (e.g., at authoring time, at instantiate time) the data models and communications methods used by its elements and dynamic context.

Within a content aggregation, the elements of the aggregation must be controlled and controllable. An object must be able to access the data model of the outer scope of its content and the inner scope of its elements.

A content aggregation must be able to look one level down, to push data and control messages onto its instantiated direct descendants. Since aggregates are opaque and complex on a level-by-level basis, it appears unreasonable to look down multiple levels, or at least we take this as a simplifying assumption.

A content object or aggregation must be able to look one level up, to propagate data to its controlling environment. A content object or aggregation must also be able to look at or propagate data to what it thinks is its most global level of control scope. Up one level and global top are conceptually available to any object; up $n$ levels is not well defined, so again, we assume a limited propagation scope.
A mechanism to map data and control messages between levels is needed. Who has the responsibility for the mapping; the sender, the receiver? An aggregation should always pass messages down in the data model of the receiver (it knows the destination), but pass messages up in its own data model (it does not know about its outer environment except at run time). This permits static authoring with tight coupling and means that objects (not aggregations) need not do discovery or data model mapping.

Each level in the content structure or hierarchy is identical in nature except for the lowest level at which content is considered to have a monolithic structure for purposes of aggregation and sequencing. Conceptually you can stop thinking about aggregations at any level; anything below is considered opaque.

Each content object or aggregation has metadata.

The content objects within a content aggregation are proxies (pointers, references, etc.) for the real content.

When someone or something (e.g., controller, sequencer, author, generator) decides to utilize a particular content object within the aggregation, the content proxy must be bound to a particular real content object. Utilize does not mean initialize; utilize may be author time (early bind), or run time initialize (late bind).

The mechanics of how to resolve the proxy and do the binding are specific to a particular aggregation. There may be multiple mechanisms to handle the binding; a particular aggregation will use a particular method.

All binding and proxy methods must be fail soft; content must continue to work if the binding fails to resolve the proxy to a real content object.

There may be a general-purpose service to help content instances maintain state across re-entry or re-launch of the content instance.

There may be parallel, concurrent content within the sequencing of a single aggregation. This is controlled by the private aggregation sequencing method. The interesting problem is how to extend the communications across layers of a single scope thread to communications between objects in different scope threads.

Requirements

The discussion and assumptions above lead to requirements for a solution. The solution must provide the framework for aggregations, not a single solution.

- Agreement to limit the scope to the general hierarchical/graph-structured aggregation model.
- A mechanism to describe the structure of the aggregation (a data model).
- A mechanism to describe the sequencing models that an aggregation uses (a data model).
- A mechanism to describe the data models and communications messages that an aggregation uses (a data model).
- A single message transport structure (API, XMPL, XMLQuery?) (a transport model).
- A mechanism to do cross-walks between the corresponding data models and message models (this is a complex processing model).
- A mechanism to describe the actual sequencing within an aggregation (a data model and a processing model). A scripting language for sequencing within aggregations is a possible solution.
- A mechanism to discover and map data between the data models used in the different levels of the aggregations (a processing model).
- A mechanism to describe the method used to bind a content proxy to the real content object (a data model).
CLEO and SCORM 2.* Solution

CLEO and SCORM 2.* must pick a fixed data model (or set of models with fixed cross-walks). General data model mapping is too much of a semantic mapping issue for now. CLEO and SCORM 2.* focus on aggregations of content with known interoperable data models.

CLEO and SCORM 2.* add a local sequencing algorithm to each aggregation object and provide a simple content proxy binding procedure. The solution provides an approach for sequencing algorithm representation.

Miscellaneous Notes

Can we develop a simple taxonomy for three things --- (1) types of communications, (2) types of data models, (3) types of sequencing --- and then plot or tabulate how different approaches (e.g., SCORM, Online) fall into this overall structure? Such a plot and taxonomy could express the range of potential solutions.
The following note describes a *simple* set of extensions to SCORM content aggregations to support the basic sequencing of the *items* (SCOs or aggregations) in the aggregation. The objective is to describe a common collection of sequencing behaviors with minimal impact on the current CSF, CMI, CP, and SCORM data models to meet the immediate sequencing needs of SCORM 1.2/1.3. The solution is not meant to outline a direction for addressing content aggregation and sequencing in SCORM 2.0+ or to preclude any alternative solutions in the future.

This document describes only extensions to the descriptions of SCOs and content aggregations and the associated required behavior, i.e., it presents an information model and processing model. Actual binding to an XML representation for content exchange is not addressed.

### Assumptions

- An aggregation is a mixed collection of SCOs and content aggregations. For the purpose of sequencing, the content aggregation is simply considered as a set of learning objects of arbitrary complexity, i.e., all are *items*.
- Learning objects or items in the aggregation that are SCOs follow conventional launch and communicate behavior. Each SCO returns a *status* value (i.e., lesson status) that is available to the sequencing processor.
- Learning objects in the aggregation that are themselves aggregations are treated like recursively processed SCOs. Each aggregation returns a *status* value that is available to the sequencing processor.
- The current CMI data model for SCOs is used as is. While known to be inadequate for the long term, changes to the data model are outside of the scope of the light sequencing solution.
- The solution is based on explicit declarations of sequences or sequences controlled only by the *status* value returned by a SCO or content aggregation. Score-based or other complex
state rollup is outside of the scope of the light sequencing solution.

- The solution does not address any aspects of skins, rendering, fancy content, metadata, objectives, competencies, play rules, etc., but is strictly limited to simple sequencing.
- Prerequisites sequencing using a scripting solution (e.g., AICC script) is too complex for the light sequencing solution.

Contents

Information Model

The SCORM CP data model requires additional items to describe sequencing. Except for the item status data item, all data items are part of the aggregation or SCO description. The binding of the information model to the SCORM CP extensions must incorporate these new fields.

Content Aggregation Data Model

A content aggregation has the following additional data (the data does not exist for an item that is a SCO):

- **Sequencing type** -- a fixed vocabulary of behaviors used to sequence the items within the aggregation. The sequencing type is defined as part of the aggregation description, e.g., it appears in the content package. Only a single value from the vocabulary may be present. The value is optional. If not present, the default sequencing type is user.

  Values of the vocabulary are:
  - **User**: sequencing is user selected.
  - **Ordered**: use the explicit ordering values to sequence and present all items.
  - **Successful**: use the explicit ordering values to sequence and present items. Present items until the sequence completes or the learner fails, i.e., continue as long as the learner continues to make progress through the sequence.
  - **Simple Prerequisites**: use prerequisite between SCOs and items combined with the status values of the SCOs and items to sequence the items.
  - **Next Rule**: use the individual next rule for each item combined with the status values of the SCOs and items to sequence the items.

Sequencing behavior for each sequencing type is detailed below.

- **Completion status rule** -- a fixed vocabulary of rules or methods used to determine the item status or state of the aggregation. The item status value is used to propagate status for use with the next rules and prerequisites in the parent aggregation. The completion status rule is defined as part of the aggregation description, e.g., it appears in the content package. Only a single value from the vocabulary may be present. The value is optional. If not present, the default completion status rule is defined by each sequencing type.

  Values of the vocabulary are:
  - **Explicit passed**
  - **Explicit failed**
  - **Explicit completed**
  - **Explicit incomplete**
  - **Last**
  - **Passed all**
  - **Passed or completed all**
  - **Passed any**
  - **Failed all**
  - **Failed any**
  - **Completed all**
  - **Completed any**

  How the sequencing processor sets the item status for each value of the completion status rule vocabulary is detailed below.

- **Item status** -- a fixed vocabulary representing the current status of the aggregation.
Values are a subset of the SCO *lesson status* values. The value of *item status* is determined solely by the sequencing processor and is available only to the sequencing processor. Content, i.e., SCOs, do not have access to the *item status* of the aggregation. *Item status* is part of the item internal data model of the sequencing processor, e.g., it *does not* appear in the description of the aggregation in the content package.

Values of the vocabulary are:
- **Incomplete**
- **Completed**
- **Passed**
- **Failed**

The other *lesson status* values for a SCO (i.e., *browsed, ab initio, not attempted*) are not used to sequence aggregations.

**Content Element Data Model**

Each item (SCO or aggregation) within the aggregation has the following additional data:

- **Ordering** -- an integer value specifying an explicit ordering of the item within the aggregation. The *ordering* is defined as part of the item description, e.g., it appears in the content package. The value is optional. The value need not be unique within the aggregation.

- **Simple Prerequisites** -- a list of zero or more identifiers of other items within this aggregation. The prerequisites must be completed before the corresponding item can be launched. The *simple prerequisites* is defined as part of the item description, e.g., it appears in the content package. The value (list) is optional. An item identifier in the list should (but need not) match the identifier of another item within the aggregation.

- **Next rule** -- a list of zero or more pairs of sequence branches. Each pair in the list consists of a *status* value (from the SCO *lesson status* or aggregation *item status* vocabulary), and the identifier of an item within the aggregation. Based on the *status* value, branch to the corresponding item. The *next rule* is defined as part of the aggregation description, e.g., it appears in the content package. The next rule (list) is optional. An item identifier in the list should (but need not) match the identifier of another item within the aggregation.

**Sequencing Behavior**

Sequencing is defined only by the *sequencing type* of the aggregation. If an item in the aggregation is a SCO, it is launched, tracked and the *lesson status* and other results are returned to the LMS. If the item in the aggregation is a (sub)aggregation, the (sub)aggregation is started or launched, using the *sequencing type* for the (sub)aggregation, with an *item status* value returned according to the described behavior of the (sub)aggregation sequence.

**User Sequencing**

Let the user choose the sequencing.

- All of the items in the aggregation are sorted in ascending order using the *ordering* value of each item in the aggregation.
- Items without an *ordering* value are placed at the start of the sequence.
- The ordering of items with duplicate *ordering* values is indeterminate. All items with duplicate *ordering* values appear in the sequence.
- The sequence is presented to the user showing the ordering based on the sequence (the actual display is not specified, e.g., it may be a hierarchical table of contents showing the items in the ordered sequence, a drop down list). The user may select any item from the sequence in any order.
- When the launched item terminates, the user may select any item from the sequence.
- The user must have an option to make a selection to *exit* from the sequence or return to...
the parent aggregation. Such an exit behavior terminates the sequence.

- The final item status value for the aggregation is determined by the completion status rule for the aggregation.
- If the completion status rule is missing, the item status is set to completed when the sequence terminates.
- If the ordered sequence is empty, no content is presented, and the item status for the aggregation is determined using the completion status rule or its default value.

**Ordered Sequencing**

Present the items according to the specified explicit sequence.

- All of the items in the aggregation are sorted in ascending order using the ordering value of each item in the aggregation.
- Items without an ordering value are omitted from the sequence.
- The ordering of items with duplicate ordering values is indeterminate. All items with duplicate ordering values appear in the sequence.
- Starting with the item with the smallest ordering value, the items in the sequence are presented in order. All items in the sequence are presented unconditionally. As each item terminates, the next item in the sequence is launched.
- When the last item in the sequence terminates, the aggregation exits.
- The final item status value for the aggregation is determined by the completion status rule for the aggregation.
- If the completion status rule is missing, the item status is set to completed.
- If the ordered sequence is empty, no content is presented, and the item status for the aggregation is determined using the completion status rule or its default value.

**Successful Sequencing**

Present the items according to the specified explicit sequence until the sequence completes or the learner fails, i.e., continue as long as the learner continues to make progress through the sequence.

- All of the items in the aggregation are sorted in ascending order using the ordering value of each item in the aggregation.
- Items without an ordering value are omitted from the sequence.
- The ordering of items with duplicate ordering values is indeterminate. All items with duplicate ordering values appear in the sequence.
- Starting with the item with the smallest ordering value, the items in the sequence are presented in order. If the item terminates with a status of failed, the sequence is terminated and the aggregation exits. If the item terminates with any other status value, the next item in the sequence is launched.
- When the last item in the sequence terminates, the aggregation exits.
- The final item status value for the aggregation is determined by the completion status rule for the aggregation.
- If the completion status rule is missing, set the item status using a default completion status rule of last.
- If the ordered sequence is empty, no content is presented, and the item status is determined using the completion status rule or its default value.

**Prerequisites Sequencing**

Present the items according to a set of prerequisites that must be satisfied. All items with satisfied prerequisites are presented.

- Examine the simple prerequisites for each item. If a simple prerequisite refers to an item that is not in this aggregation, drop the prerequisite for the item.
- Perform a topological sort on all of the items in the aggregation based on the remaining simple prerequisites.
- Items that are topologically equivalent are sorted by the ordering value of the item.
- Items that are topologically equivalent and that do not have an ordering value are placed at the start of the sequence of topologically equivalent items.
- The ordering of items with duplicate ordering values is indeterminate. All items with
duplicate ordering values appear in the sequence.

- Form the list of items that have satisfied prerequisites (initially those items with no prerequisites).
- Order the items in the list of those with satisfied prerequisites by the ordering value of the item.
- Items without an ordering value are placed at the start of the sequence.
- The ordering of items with duplicate ordering values is indeterminate. All items with duplicate ordering values appear in the sequence.
- Launch the first item in the list of those with satisfied prerequisites. If the item completes with a status of completed or passed, it satisfies the prerequisite for other dependent items. Remove the item from the list available to launch, i.e., those with satisfied prerequisites.
- Compute the additions to the list of items with satisfied prerequisites, and add them to the end of the list. If an item has already been launched, do not add it to the list.
- Repeat by launching the first item in the list of those with satisfied prerequisites.
- When the list of items with satisfied prerequisites is empty, the sequencing process terminates and the aggregation exits.
- The final item status value for the aggregation is determined by the completion status rule for the aggregation or its default value.
- If the completion status rule is missing, set the item status using a default completion status rule of last.
- If the set of items in the aggregation sequence is empty, no content is presented, and the item status is determined using the completion status rule or its default value.

**Next Rule Sequencing**

Start with an item, and conditionally branch between items based on individual next rules associated with each item. Next rules provide a way to match the status value of any item and branch to another item based on the condition. Branching may be to a specified item or may be processed as a special case.

- Examine the simple prerequisites for each item. If a simple prerequisite refers to an item that is not in this aggregation, drop the prerequisite for the item.
- Perform a topological sort on all of the items in the aggregation based on the remaining simple prerequisites.
- Items that are topologically equivalent are sorted by the ordering value of the item.
- Items that are topologically equivalent and that do not have an ordering value are placed at the start of the sequence of topologically equivalent items.
- The ordering of items with duplicate ordering values is indeterminate. All items with duplicate ordering values appear in the sequence.
- Form the list of elements that have satisfied prerequisites (initially those with no prerequisites).
- Order the items in the list of those with satisfied prerequisites by the ordering value of the item.
- The ordering of items with duplicate ordering values is indeterminate. All items with duplicate ordering values appear in the sequence.
- Launch the first item in the set of those with satisfied prerequisites.
- Examine the status value returned by the item and branch to the item specified by the next rule for the item. Details of branching follow the overall description of the process.
- Continue launching and branching between items until you reach an item for which the next branch is one of the special exit cases.
- The final item status value for the aggregation is determined by the completion status rule for the aggregation.
- If the completion status rule is missing, set the item status using a default completion status rule of last.
- If the set of items in the aggregation sequence is empty, no content is presented, and the item status is determined using the completion status rule or its default value.

Each next rule case consists of two parts, a status case vocabulary value and the identifier of the next item to launch. Acceptable values of the status case vocabulary and corresponding actions for the associated next item are:

- Ab initio: If the SCO returns a lesson status value of ab initio, branch to the specified item
in the aggregation.

- **Not attempted**: If the SCO returns a lesson status value of not attempted, branch to the specified item in the aggregation.
- **Browsed**: If the SCO returns a lesson status value of browsed, branch to the specified item in the aggregation.
- **Passed**: If the SCO returns a lesson status or an aggregation returns an item status value of passed, branch to the specified item in the aggregation.
- **Failed**: If the SCO returns a lesson status or an aggregation returns an item status value of failed, branch to the specified item in the aggregation.
- **Incomplete**: If the SCO returns a lesson status or an aggregation returns an item status value of incomplete, branch to the specified item in the aggregation.
- **Completed**: If the SCO returns a lesson status or an aggregation returns an item status value of completed, branch to the specified item in the aggregation.
- **Other**: If the SCO returns a lesson status or an aggregation returns an item status value that does not match any other case, branch to the specified item in the aggregation. This is the default branching rule if the complete set of values for all status cases is not specified.
- **Any**: This case matches any value of lesson status or item status. Unconditionally branch to the specified item in the aggregation.

Multiple next rules cases are processed in the following order.

- If there is an any case, it is used first.
- Match the actual status returned by the SCO or item against the status case of each next rule.
- If the status does not match the status case of any next rule, use the other case.
- If there is no other case, exit the aggregation.

The next item in each next rule case is generally an identifier within the aggregation. Special cases are processed as described:

- **Valid item**: If the next item is an element in the aggregation, branch to the specified item.
- **Invalid item**: If the next item is not an element in the aggregation, exit the aggregation.
- **Exit**: This is a special value for the next item indicating that the sequencing processor is to exit the current aggregation sequence.
- **Exit All**: This is a special value for the next item indicating that the sequencing processor is to exit all aggregation sequences, i.e., to terminate the sequence process.
- **Exit User**: This is a special value for the next item indicating that the sequencing processor is to exit to the first parent aggregation that has a sequence type of user. If there are no active sequences with sequence type of user, terminate the sequence process, i.e., exit all.
- **Exit Top**: This is a special value for the next item indicating that the sequencing processor is to exit to the outermost parent aggregation that has a sequence type of user. If there are no active sequences with sequence type of user, terminate the sequence process, i.e., exit all.
- **First**: This is a special value for the next item indicating that the sequencing processor is to relaunch the first item in the aggregation sequence.
- **Last**: This is a special value for the next item indicating that the sequencing processor is to branch to the last item in the topologically ordered sequence of the aggregation items.
- **Random**: This is a special value for the next item indicating that the sequencing processor is to branch to some item in the aggregation sequence at random.

**Completion Status Rule**

When an aggregation exits, the value of the item status for the aggregation is determined by the completion status rule. The different completion status rules and their corresponding behaviors are:

- **Explicit passed**: Unconditionally set the item status for the aggregation to passed when the aggregation exits.
- **Explicit failed**: Unconditionally set the item status for the aggregation to failed when the aggregation exits.
- **Explicit completed**: Unconditionally set the item status for the aggregation to completed
when the aggregation exits.

- **Explicit incomplete**: Unconditionally set the *item status* for the aggregation to *incomplete* when the aggregation exits.
- **Last**: Set the *item status* for the aggregation based on the *status* of the last item that terminated.
  - If the last item that terminated or exited was an aggregation, set the *item status* to the *item status* of the aggregation.
  - If the last item that terminated was a SCO, and the *lesson status* is *passed*, *failed*, *completed*, or *incomplete*, set the *item status* to the *lesson status* of the SCO.
  - If the last item that terminated was a SCO, and the *lesson status* is *browsed*, set the *item status* to *completed*.
  - If the last item that terminated was a SCO, and the *lesson status* is *not attempted* or *ab initio*, set the *item status* to *incomplete*.
  - If the aggregation terminates with no items being launched, set the *item status* to *completed*.
- **Passed all**: Set the *item status* for the aggregation to *passed* if all items in the aggregation terminated with *status* of *passed*. Otherwise set the *item status* to *failed*.
- **Passed or completed all**: Set the *item status* for the aggregation to *passed* if all items in the aggregation terminated with *status* of *passed* or *completed*. Otherwise set the *item status* to *failed*.
- **Passed any**: Set the *item status* for the aggregation to *passed* if any item in the aggregation terminated with *status* of *passed*. Otherwise set the *item status* to *failed*.
- **Failed all**: Set the *item status* for the aggregation to *failed* if all items in the aggregation terminated with *status* of *failed*. Otherwise set the *item status* to *passed*.
- **Failed any**: Set the *item status* for the aggregation to *failed* if any item in the aggregation terminated with *status* of *failed*. Otherwise set the *item status* to *passed*.
- **Completed all**: Set the *item status* for the aggregation to *completed* if all items in the aggregation terminated with *status* of *completed*. Otherwise set the *item status* to *incomplete*.
- **Completed any**: Set the *item status* for the aggregation to *completed* if any item in the aggregation terminated with *status* of *completed*. Otherwise set the *item status* to *incomplete*.

### Implementation Notes

The implementation requires a stack to track the current locations and *status* of the active items in the content hierarchy.

All of the sequencing options can be implemented by mapping the sequencing to individual next rules that are applied to each SCO or aggregation, i.e., when a package is loaded, the data can be mapped to an equivalent set of local next rules.

The *last* version of the *completion status rule* can be implemented by setting the *item status* for the aggregation to *completed* when the aggregation is launched. When an item returns a different value (e.g., *failed*, *passed*, or *incomplete*), update the *item status*.

The *item status* value for an aggregation that has not been launched is not defined because the value is not needed to control sequencing.

When exiting multiple levels, the *completion status rule* is applied at each level as the levels are popped from the tracking stack.

The *next rules* may encode loops. The sequencing processor must allow looping.

### Issues

There is no learning context. Different aggregations that use the same content but with different sequencing behavior are explicitly different, i.e., you cannot share or reuse behavior and
structure as independent items.

We need to verify that the topological and other sorting rules are sufficient to produce a definitive, reproducible sequence when prerequisites are used. If there are no prerequisites, the topological sort should reduce to a simple ordering sort.

The sequencing behavior described applies within a single session. SCO and item termination and exit conditions are defined for normal cases in which the next item is to be launched immediately after the current item terminates. Sequencing behavior is not defined herein for the condition of a SCO which terminates with a status of incomplete, and exit value of suspend or logout or similar condition. In such cases, the LMS must suspend the entire sequencing stack. An appropriate LMS behavior would be to resume the current sequence and resume the current SCO upon restart. However, other resume and restart rules may apply. Termination, suspend, and error state behavior are outside of the scope of the actual sequencing within the aggregation.
Contents

Overview

The following note describes the temporal relationships between events that occur when a learner interacts with one or more learning objects. Understanding these relationships is important when creating instances of tracking data for different events during content delivery.

Introduction

A strawman proposal from the IMS Content Management Working Group presented a model of the temporal relationship between events that occur when a learner interacts with a learning object:

To read the diagram, only pay attention to the vertical relationships between items. For example, the diagram shows that Activity 1 and Attempt 1 begin at the same time, and that Activity 1 and Attempt 2 end at the same time. The vertical spacing between items is only for legibility.

These relationships are important for several reasons. For example, it is not sufficient for a learning object to report to a tracking service that a learner earned a grade of 90 on the assessment activity; which attempt is a required piece of information.

When describing data models, e.g., a tracking or grade submodel, the relationship between the basic data and the data instances should not be expressed within the data model. For example, there may be a model that defines time limits, e.g., maximum duration, earliest start time, latest...
start time. The time limits might be different for different attempts, or the same for all attempts, but the data model is the same. It is important to understand the temporal relationships between the events so that the data models can be properly instantiated either for content descriptions or as part of a content tracking service.

The model presented in the IMS strawman is not incorrect; this working note makes it more general.

**Assumptions and Definitions**

The base model is as follows:

- A learner logs into and out of a Learning Management System (LMS) or otherwise connects to a content delivery and presentation service; a Login is a continuous connection of the learner to the environment.
- Within a Login session, the LMS starts one or more Activities for the learner; an Activity is an instantiation of a Learning Object.
- A learner interacts with an Activity via a Window, i.e., a display or presentation and interaction interface, e.g., a window presented by an operating system, a command line prompt, a touchpad, a simulator interface.
- An Activity is Attempted one or more times by the learner, even if the Attempt is merely presenting some text. An Attempt is a time interaction with the Activity that produces a set of results (e.g., grades) and completion status.
- The time for the Activity begins at the first Attempt at the Activity and ends when the Activity is completed. The time for an Attempt is defined by the state transition model of the LMS or delivery environment.
- An Attempt may be spread over more than one attempt Session; an attempt Session is a contiguous period of time devoted to a specific Attempt at an Activity, as maintained or recorded by a tracking service.
- An Attempt may be Suspended, i.e., the learner stops working on that Activity. When an Attempt is terminated, if the Attempt is not complete, it is Suspended.
- An attempt Session may be Paused, i.e., the learner takes a short break from working on the Activity during an Attempt. A tracking service would not record the time during a Pause as time the learner spent working on the activity or attempt.
- All interactions such as looking up a glossary item, entering a chat session, checking status in a course, or reading the syllabus are modeled as Activities. The learner may have multiple sessions with such Activities; differentiating attempts is meaningless and the Activity can never be Completed.

The following briefly describes scenarios that were not illustrated in the base model. Each scenario is developed below in this note.

- An Activity (as a time span) may encompass another Activity, such as a remediation Activity or a syllabus Activity. This scenario also represents nested contents and activities with a content hierarchy.
- The original model did not include the learner logging in and out of the LMS.
- An LMS may display multiple concurrent Activities simultaneously, e.g. in separate Windows. Normally, a learner only interacts with Activities through a single Window.
- Two Activities may be combined and presented together in a single Window.

**Simultaneous Activities; Single Window**

The base model does not illustrate Simultaneous activities. In the diagram below, Activity 2 could be something as simple as examining the Syllabus for the course, or it could be as complex as a remediation Activity that recursively calls other Activities. In these cases, Activity 1 is suspended while the learner attempts the other Activities.
Logging in

The model does not illustrate when a learner logs into and out of the LMS. This information could be useful in various tracking systems. The diagram presents the following scenario in which a learner:

- logs in, begins Attempt 1, logs out;
- logs in, finishes Attempt 1, begins Attempt 2, logs out; and
- logs in, finishes Attempt 2, logs out.

In other diagrams in this note, if the Login events are not shown, then a single Login is implied.

Simultaneous Activities; Multiple Windows

The base model does not illustrate simultaneous Activities occurring in separate Windows. The diagram below could represent a scenario in which a learner starts a Session in Activity 1. He then starts Activity 2, which is a simulation that helps him with Activity 1. The simulation in Activity 2 could be a multi-user environment that continues independently of the learner completing Activity 1.
Joined Activities

It is possible that the environment could present two or more Activities at the same time, where the joint Activity is more than the sum of the parts. The diagram below could represent a scenario where a presentation-only Activity is displayed with an assessment Activity, as in an expositional text followed by an inline assessment.

Description of Relationships

Based on the diagrams above, we can determine the cardinality of the relationships between items, which is summarized below. Read the table as RowName:ColumnName; the first data cell shows that the relationship of Login to Session is 1:N. "NDR" stands for "No Direct Relationship" between the components (although a relationship may be inferred, as in Rendering to Session is 1:N because Rendering to Attempt is 1:N and Attempt to Session is 1:N). An "S" indicates that there may be multiple simultaneous relationships. Details of the relationships follow this table.
The following table provides more information about the table above. All relations in this table imply synchronous beginnings and endings; for example, a Login begins at the same time that a Session begins, and the Login ends at the same that a Session ends.

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Cardinality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login:Window</td>
<td>1:N, S</td>
<td>A single Login has one or more Windows. A Window or presentation interface terminates when the learner logs out of the LMS.</td>
</tr>
<tr>
<td>Login:Activity</td>
<td>N:N, S</td>
<td>A learner can login and logout of the LMS independently of starting or completing activities. A learner can initiate multiple activities at the same time.</td>
</tr>
<tr>
<td>Login:Session</td>
<td>1:N, S</td>
<td>A Session cannot span multiple Logins. A single Login may persist through sequential Sessions. There may be simultaneous Sessions if there are multiple Windows.</td>
</tr>
<tr>
<td>Window:Activity</td>
<td>1:N, S</td>
<td>Normally, a Window displays one or more Activities sequentially. A Window may display two or more joined Activities simultaneously.</td>
</tr>
<tr>
<td>Window:Session</td>
<td>1:N</td>
<td>A Window is essentially a sequence of Sessions because the Windows are what allow the learner to interact with the Sessions.</td>
</tr>
<tr>
<td>Activity:Window</td>
<td>1:N</td>
<td>An Activity may be displayed in different Windows sequentially, but an Activity cannot be displayed in two Windows at the same time. Usually an Activity is displayed in a different Window in different Login sessions.</td>
</tr>
<tr>
<td>Activity:Attempt</td>
<td>1:N</td>
<td>By definition, an Activity is a sequence of Attempts.</td>
</tr>
<tr>
<td>Attempt:Session</td>
<td>1:N</td>
<td>By definition, an Attempt begins with the beginning of a Session and ends with the ending of a Session.</td>
</tr>
</tbody>
</table>

**Issues**

The model as presented assumes a single user context. It does not cover collaborative Activities such as multi-person simulations or interactions between coach and student.

The model also does not allow an Activity to be opened in multiple Windows. It's not clear if this is a common occurrence that needs to be supported.
Overview

Handles are a way to create persistent identifiers that allow the physical location of objects to move or different objects to be returned for a single identifier. The classic example for the first type is the Domain Name System (DNS) that translates a computer host name to an IP address, e.g., www.cmu.edu to 128.2.11.43. An example for the second type is the Akamai system that replicates websites around the world; connecting to a website such as www.yahoo.com seamlessly redirects your request to a server closer to you on the network for faster download times.

Using handles for content and learning objects allows a service to choose a learning object that is most appropriate, based on the current platform the student is using, his connection speed to the run-time service, his preferred learning style, his field of interest, etc., so that appropriate examples are displayed, etc. Handles can abstract a collection of version, variation and preference functions from a content model.

The Handle System

The Handle System (http://www.handle.net/) is a "a general purpose distributed information system designed to provide an efficient, extensible, and secured global name service for use on networks such as the Internet." It was originally developed for digital libraries, but was generalized as a naming schema for digital objects.

"Every handle consists of two parts: its naming authority, otherwise known as its prefix, and a unique local name under the naming authority, otherwise known as its suffix. The naming authority and local name are separated by the ASCII character "/". A handle may thus be defined as

< Handle> ::= < Handle Naming Authority> ""< Handle Local Name >

"For example, "10.1045/january99-bearman" is a handle for an article published in the D-LIB magazine. It is defined under the Handle Naming Authority "10.1045", and its Handle Local Name is "January99-bearman".

10.1045 / january99-bearman

Naming Authority Item Identifier

[Prefix] [Suffix]
"As a general purpose indirection system that resolves identifiers into state information, the Handle System can be used to advantage in any dynamic network environment as part of the overall process of managing digital objects."


The basic model of use of the handle system is to take a handle and resolve it into a source for the corresponding object, permitting distributed resolution and storage location independence. A unique handle is always passed to the resolution. A specific locator for the corresponding object is returned.

Handles for Learning Objects

The concept of a unique identifiers and resolution can be applied to learning objects. The simplest approach is that of a location resolution service. An element of a content model will contain the handle of the object, rather than the location of the object. When the handle is passed to the resolution service, the actual location is returned. The handle may be represented as a GUID, a URN, a DOI, etc. Given the actual location, the resulting object may then be accessed.

The concept can be generalized by enabling the resolution service to return more than the single location for an object. A simple extension is to enable the resolution service to return a set of alternative locations for the object. The requesting client can then make a choice from the given alternatives. Creating a distributed content repository requires a handle resolution mechanism.

Another extension would be for the resolution service to return multiple versions of the object to the requesting client. By analogy, one could ask for a book without requesting a particular edition. The resolution service would return the list of all editions, i.e., all versions, and the client would then choose the appropriate one to request.

The same content object could exist in similar or functionally (pedagogically) equivalent variants. The object might exist in different languages, or in variants with different technical characteristics such as minimum display resolution. By generalizing the concept of a handle from being a unique identifier to being a resolvable identifier for an object, a collection of different variants of the object could be accessed via a single handle. Examples include:

- Language
- Technical requirements
- Duration
- Cost/Price
- Pedagogical approach
- Accessibility
- Localization

Thus handles can take on some of the role of content customization. Within a content model, the object handle can identify the super class of all equivalent objects. Resolution finds the set of matching objects, and customization returns the appropriate single object from the resolution service. Thus aspects such as customization, variant or version handling can be abstracted into a single handle model.

Similarly, the handle may reference a collection of related objects. A single handle might reference both a learning object and its corresponding metadata object. The resolution strategy could return either or both objects.

Resolution may also be an iterative process. Rather than returning the final location (or alternative locations) for an object, the resolution service may return a handle, not an object. The process would then repeat, until resolved to an object. As an example, when requesting an object that exists in multiple versions, the first request would return the handle of the different
versions (e.g., the handles of the different editions), but not the actual location of any copy of a particular edition or version. Once the version is selected, a second resolution would find the location of the object.

Note: the concepts of client and resolution service can be layered. The end-user client may make a request of a resolution service. This intermediate resolution service may know how to pick from alternatives. It will in turn make a request from another resolution service that will return the list of alternatives. The intermediate service will perform the alternative selection, and return a single result to the end-user client. As far as the end-user client is concerned, the levels of intermediate resolution are not apparent.

A content handle system may also function as a query interface. Rather than passing the handle to the resolution system and letting the client select from the alternatives returned, the client could request only those objects that match a set of criteria be returned. This approach is functionally equivalent to adding another layer to the resolution strategy. Thus the choice is one of implementation, not a difference in concept of content handles.

Thus the handle approach can also be used to identify content object alternatives. A learning object could contain the handle of the set of alternatives. The resolution strategy could use the learning context to select the appropriate content, abstracting such selection from the overall content model.

**Using Content Handles**

The simple rule is that any reusable object, e.g., any object that has a GUID, should be resolved via a handle.

Content handles abstract versioning and variants from an overall data model. Thus handles should be used to provide indirection for any element (content or other) for which there could be alternatives.

The Content Models speculation explicitly calls out the use of content handles in several places.

- All of the elements in the Learning Content Collection in a Aggregation Learning Object are represented by a handle. The instances in the Aggregation Collection will resolve via the handle to a particular alternative for a Learning Object or a Aggregation Learning Object.
- The Learning Content in a Purposed Learning Object is represented by a handle. The instance will resolve via the handle to a particular alternative for a Learning Object or a Aggregation Learning Object.
- The Learning Content in a Learning Object is represented by a handle.

Note: the above does not explicitly mention the resolution of the Learning Content within the Contextualized Learning Object. This nesting of structure may only be logical, but if a physical layer is inserted in the binding, use of a handle here would also be appropriate.