Sensible Sequencing

by Mike Rustici August 5, 2008

Introduction

What follows is an attempt to outline some basic concepts that could be fleshed out into a sequencing and navigation specification for SCORM 2.0. It assumes that SCORM 2.0 looks a lot like SCORM 2004. It is not complete by any means, but I believe the concepts are solid and will be easily extended into a full specification.

The intent of sensible sequencing is to provide a straight-forward and simple mechanism for controlling learner navigation between SCOs. Sensible sequencing should be:

- Intuitive and straight-forward
- Congruent with the existing SCORM data models
- Easy to implement
- Easy to extend and improve

Concepts

Items

Every item in the SCORM manifest participates in sensible sequencing. These are called "items". In sequencing rules, items are referenced by the SCORM manifest's item identifier. Every item has a small set of tracking data as defined in the Data Model. Whenever possible, the data used by Sensible Sequencing is taken straight from the CMI runtime data held by a SCO. There is a direct correspondence and in reality there is only one single authoritative data store. A SCO's runtime data is only reset when an explicit rule triggers this reset, by default, runtime data is never reset.

Rules Only, No Attributes

In Sensible Sequencing all behavior is specified by explicit rules attached to an item. There are no "attributes" which statically define behavior for or characteristics of an item. Static attributes only serve to limit and confuse. There isn't any behavior that can't benefit from being allowed to vary based on conditional statements. If an author needs an unconditional behavior, it can easily be represented by a conditional that always evaluates to true. Attributes that describe behavior are in fact really just implicit rules. In keeping with the goal of making Sensible Sequencing straight-forward, anything implicit is bad. In Sensible Sequencing, there is no default logic, everything is defined as an explicit set of rules.

Data model

For clusters, the following data is defined and derived from rollup rules.

- Completion Status
- Success Status
- Score (scaled)

For items linked to SCOs the data model accessible for Sensible Sequencing includes all of the above in addition to:

- Launch Count
- Attempt Index (An attempt is a reset of sequencing data. The attempt index is a zero-based counter identifying a set of runtime data.)

For SCOs, Completion Status, Success Status and Score are all derived from the SCO's CMI runtime data. Since Launch Count and Attempt Index are maintained for SCOs, I would propose that they also be accessible via the runtime API. Furthermore, sensible sequencing rules can access any other element of the CMI runtime data.

For items linked to launchable Assets, Launch Count is tracked, Attempt Index will always be 0. Completion Status, Success Status and Score are always unknown.

Properties

Properties are data points associated with items that are derived from the execution of rules. The LMS uses properties to determine behavior. The following properties are maintained for each item:

- Visible in Selection
- Enabled in Selection
- Directed Continue Navigation Enabled
- Directed Previous Navigation Enabled
- Included in Directed Navigation
- Show Exit SCO Navigation
- Show Exit Course Navigation
- Participates in Satisfaction Rollup
- Participates in Completion Rollup
- Percent Contribution to Score Rollup
- Run Time Data To Be Reset - Indicates that the next time the item is launched, its data should be reset
- Ordinal - Specifies the order in which items are displayed within the context of their parent.
- Used - Was this one of the items selected randomly to be included
- Current - Is this item currently being displayed

Course level properties

- Resume From - The result of a "resume from" rule action defined below
- Initialized - False before any rules are evaluated, True once the course has been launched and all rules have been evaluated once
- Pending navigation request

> *Selection* - The ability for the user to select an item from a table of contents
> *Directed Navigation* - The ability to navigate the course using previous and next buttons

I would also propose to extend the runtime API to make sensible sequencing properties available to the SCO

Shared Containers

There is often a need to store "course level" data that is shared across SCOs. This data could store learner preferences or it could be used as a variable to control sequencing behaviors. A course many define zero or many shared containers that store data that is shared amongst all SCOs in the course. Each container will have the following properties:

- Container Id - an identifier that uniquely represents the container within the scope of the course
- Data - A string field that can hold any arbitrary set of characters

Advanced LMS systems may desire a method of sending data to a sequenced package. For instance, learner preferences, such as learning style, may be stored globally. Or perhaps, the LMS has prior knowledge of the learner's competencies that the content should know about. To facilitate such interactions, I would proposed that the container model be made extensible so that different categories of data can be exchanged between the LMS and the course to accommodate different communities of practice. This data exchange could easily be accomplished by adding a third property called Type. Type would be a GUID that identifies the structure of the information held in the Data property. For instance, the Type could be "http://www.letsi.org/defined_competencies" and the Data could be something like "Knows how to tie shoes:Mastered".

Containers should be accessible via an extension to the runtime API as well as via rules.

Rules

Any item can have zero to many rules associated with it. A rule is composed of a condition clause and an action clause. The condition clause is evaluated to determine if the LMS should perform the action. An action may direct how sequencing is performed, how rollup happens or

what navigational elements are present for the user. Conditions can be combined using logical operators (AND / OR / NOT) and grouped for precedence.

The intent of a rule is to form a statement like "if this/these item(s): __(item reference)__ are in this state: __(state)__ then do this: __(action)__. Rule conditions fill in the item reference and state. Rule actions fill in the action.

Item Reference Rule Component

There are two types of item references, the first is to an explicit single item and the second is to a collection of children items:

Direct reference components:

- Referenced Item - Identifier for a specific item to reference if other than the current item.
- Referenced Container - Identifier for a specific container to reference.

Child reference components:

- Any / All / At least __
- Immediate Children Only / All Descendants

State Rule Component

The State Rule Component has three parts:

Data Model Reference

- Data model element - Which of the CMI or Sensible Sequencing data model elements is being referenced (if applicable to the condition, for instance "Always" doesn't need a data model element reference). Sensible sequencing properties can also be referenced.

- Element Id – If referencing a CMI data model collection (objectives, interactions, comments) the identifier of the collection element to be referenced.

- Is Currently / Has previously been / Was during attempt index ___ - This is a flag that can be placed on a condition to look at previous attempt data (by implication I am suggesting that attempt history be maintained)

Condition

- Equal to
- Greater than
- Greater than or equal to
- Less than
- Less than or equal to
- Text Contains
- Text Starts with

- Text Ends with
- Always
- Never
- Is Current (only one item may be "current" at any given time, that is the SCO that is currently in front of the learner)
- Is Active (all parents of the current item are "active")
- Is Used (was this item chosen when only a subset of children are used according to a "randomly use __" sequencing behavior action)

Operator

- Not - This is a true logical NOT to mean "all other conditions". For example the condition "NOT Success Status Equal To Satisfied" evaluates to true if success status is "not satisfied" or "unknown".
- And
- Or

Action Component

There are several categories of actions than can be performed for each rule:

- *Rollup Actions* change the state data for an item
- *Presentation Actions* alter the LMS's UI and control what the user can and cannot select
- *Sequencing Behavior Actions* control what is available for directed navigation and selection
- *Navigation Request Actions* control how the user is navigated through the course after exiting an item
- *Data Management Actions* control when and how state data is reset

Actions affect only the item on which the rule is defined.

Rollup Actions (may be defined on any item, including SCOs)

- Satisfaction is Passed
- Satisfaction is Failed
- Satisfaction is Unknown
- Completion is Completed
- Completion is Incomplete
- Completion is Unknown
- Participates in completion rollup
- Participates in satisfaction rollup
- Does not participate in completion rollup
- Does not participate in satisfaction rollup
- Counts __% towards score rollup
- Score is __
- Score is as provided or weighted average of children
- Container ____ Data is _____

Presentation Actions – These actions only disable the user's ability to access these events via the LMS GUI, if a sequencing rule results in a navigation request or if a SCO generates a navigation request, it is honored no matter what the rules say about the availability of that request to the user through the LMS GUI.

- Disable Selection
- Disable Selection for Item and All Children
- Disable Selection for Item and All Descendants
- Hide From Selection
- Hide Item and All Children from Selection
- Hide Item and All Descendants from Selection
- Disable Directed Continue Navigation
- Disable Directed Previous Navigation
- Hide Exit SCO Navigation
- Hide Exit Course Navigation

(we also need the opposite conditions to remain consistent)

- Enable Selection
- Enable Selection for Item and All Children
- Enable Selection for Item and All Descendants
- Show in Selection
- Show Item and All Children in Selection
- Show Item and All Descendants in Selection
- Enable Directed Continue Navigation
- Enable Directed Previous Navigation
- Show Exit SCO Navigation
- Show Exit Course Navigation

Sequencing Behavior

- Bypass in Directed Navigation (automatically bypasses all children as well)
- Include in Directed Navigation
- Shuffle Order of Child Items
- Maintain Specified Order of Child Items
- Randomly choose __ child items to use
- Randomly choose __ new child items to use
- Use all child items

Navigation Requests

- Next
- Back
- Select (go to a specific item)
- Exit Course (There is no distinction between exit types, by default all data is saved for next attempt. To reset data, create a rule with a data management action.)
- Exit to Interstitial Resource

- On Resume Start From Last Position
- On Resume Start From Beginning
- On Resume Start With Interstitial Resource (can be used to allow the user to select where to start)

Navigation request actions are only evaluated on items which correspond to a SCO.

Data Management

- Reset runtime data
- Reset runtime data for this item and all descendants
- Maintain runtime data

Rule Evaluation

- All rules are evaluated for all items upon initial launch of the course.
- Rules are evaluated whenever the SCO calls Commit (if this proves unfeasible, this could change to Terminate, but I think that rule evaluation will scale fairly linearly so performance shouldn't be a huge concern). Rules resulting in navigation request actions and data management actions are only evaluated when the SCO calls Terminate. Navigation request actions are only evaluated when defined on a SCO (except for "On Resume __" actions).
- For a given item, rules are evaluated in the order in which they are defined in the manifest.
- If two rules produce conflicting actions, then the action resulting from the last rule to be evaluated wins.
    - For example, if the first rule evaluates to an action of "Satisfaction is Failed" and the second rule evaluates to an action of "Satisfaction is Passed", the the item will be passed.
- Data resulting from the actions of prior rules is used when evaluating later rules.
    - For example, if the first rule is "IF always THEN satisfaction is passed" and the second rule is "IF cmi.success_status equals passed THEN Disable Selection", the item will always be disabled in selection.
- Rules on the current SCO are evaluated first, followed by rules on the SCO's parent, then it's parent, etc until the root item is reached. (Note: There is still an open question remaining about how to evaluate items with rules that explicitly reference items that are changed during rule evaluation. The issue gets tricky and performance suffers if there is a lot of cross referencing.)

Rule Definition

- Every item has a *rule container* that is essentially a list of rules to be evaluated in order.
- A rule container contains an ordered list of *rule elements*.

- A *rule element* is either a defined rule or a pointer to another *rule container*. If the *rule element* is a pointer to a *rule container*, the *rule elements* in the referenced *rule container* are simply substituted into the original *rule container* in-line.
- *Rule containers* can be defined outside of the context of an item so they can be shared across many items. These are called *shared rule containers*.
- When a *rule element* references a *rule container* it can reference either a *shared rule container* or the *rule container* in another item.

The Default Rules

Every manifest will contain at least one rule container that contains a set of default rules. Every item must explicitly reference the default rule container as a rule element if it wants any rollup or other default behavior to occur. It is suggested that the default rule container be evaluated first, but it is up to the content developer to make this determination. In fact, the content developer can even change the rules specified in the default rule container. The default rule container is really only a convenient way to inform the content developer of rules that should be implemented to achieve commonly expected behaviors.

Rollup

- If always, then participates in completion rollup
- If always, then participates is satisfaction rollup
- If always, then score counts 100% towards rollup
- If any children have (satisfaction status of failed AND participates in satisfaction rollup), then satisfaction status is failed
- If all children have (satisfaction status of passed AND participates in satisfaction rollup), then satisfaction status is passed
- If any children have (satisfaction status of unknown AND participates in satisfaction rollup), then satisfaction status is unknown
- If all children have (completion status of unknown AND participates in completion rollup), then completion status is unknown
- If any children have (completion status of incomplete AND participates in completion rollup), then completion status is incomplete
- If all children have (completion status of completed AND participates in completion rollup), then completion status is completed
- If always, then score is as provided or weighted average of children
- If not used, then does not participate in completion rollup
- If not used, then does not participate in satisfaction rollup
- If not used, then score counts 0% towards rollup

Presentation

- If always, then enable selection
- If always, then show in selection
- If always, then enable directed continue navigation
- If always, then enable directed previous navigation

- If always, then show exit SCO navigation
- If always, then show exit course navigation
- If not used, then disable selection
- If not used, then hide from selection

Sequencing
- If always, then include in directed navigation
- If always, then maintain specified order of child items
- If always, then use all child items
- If not used, then bypass in directed navigation

Navigation Request
- If always, then next
- If always, then on resume start from last position
- If not initialized, then resume from beginning

Data Management
- If always, then maintain runtime data

Interstitial Resources

Interstitial resources are simply SCOs or launchable assets that are designed to provide context or information between SCOs. They could include a menu of the course, information about your progress, or a simple message to the learner. Nothing special is required to define interstitial resources in a content package, they are simply resources that are listed in the manifest like any other SCO or asset (with a defined launch URL). Navigation Request Rule Actions can specify that the LMS should launch an interstitial resource. The rule action can specify which resource to launch using its resource identifier. Once launched, the interstitial resource may make GetValue calls to the SCORM API to query the status of any data model element for any SCO or item in the course (I would propose that the runtime specification be extended to provide this functionality to all SCOs). The interstitial resource may only make SetValue calls to initiate navigational requests or alter Container data. A resource in the manifest may be marked as the default interstitial resource. This resource will be launched when the sequencer does not have an item to deliver. If the manifest does not identify a default interstitial resource, the LMS may provide a page of its own choosing. The runtime API should be extended to allow for another data model element to be used in a read-only capacity by interstitial resources. This data model element could be called something to the effect of "reason you were launched" and the return value would be a series of tokens such as "reached end of course", or "called by rule action".

Execution Path

Below are the high level algorithms that an LMS would need to execute to process Sensible Sequencing.

Begin

Executed when the course is launched.

1. Form an ordered list of launchable assets according to a pre-order traversal of the tree (the directed navigation set)
2. Form an ordered list of all items according to a bottom up traversal of the tree (the rule evaluation set)
3. If Initialized is false
    1. For each item in the rule evaluation set
        1. Invoke the Rule Container Evaluation Process, passing in the item's rule container
2. Invoke Reset Runtime Data Process on the root item
3. If resume from = Start From Last Position
    1. Find item that has Current=true and invoke launch process on it, if no matches, launch interstitial resource
2. If resume from = Start From Beginning
    1. invoke launch process on first item in directed navigation set
2. If resume from = Start With Interstitial Resource
    1. Launch specified interstitial resource

Rule Container Evaluation Process

1. Form an ordered list of rules to be evaluated, by combining the rule elements in the passed in rule container with any referenced rule containers.
2. For each rule
    1. Invoke the Condition Evaluation Process
    2. If the condition evaluates to true
        1. Invoke the Action Process

Condition Evaluation Process

1. If condition references an item
    1. Add the referenced item to the evaluation set
2. Else, the condition references a child set
    1. Add all children or descendants to the evaluation set
2. Set number of successful conditions = 0
3. For each item in the evaluation set
    1. Evaluate the condition against the data model element, combining sections using operators and order of precedence
    2. If the condition evaluates to true
        1. increment number of successful conditions
3. If child reference is any
    1. If number of successful conditions > 0, return true
2. Else if child reference is all

        1. If number of successful conditions = the size of the evaluation set, return true
    2. Else if child reference is at least __
        1. If number of successful conditions >= __, return true

Action Process

For most actions, performing the action is simply a matter of the item's property to a value. For the following actions, more processing is necessary:

Score is as provided or weighted average of children

1. If the item is a SCO, the score remains whatever was set by the SCO
2. Else, if the item is a container
    1. Set total score value = 0
    2. For each immediate child of the item
        1. Get its score
        2. Multiply the score by the percent contribution to score rollup property
        3. Add this value to total score value
    4. Set the item's score to total score value divided by the number of children

Shuffle Order of Child Items

1. Set the ordinal of each child item to a new random value so that when sorted by ordinal, the child items will be in a random order

Randomly choose __ child items to use

1. Set the used property of all child items to false
2. Randomly select __ items and set their used property to true

Randomly choose __ new child items to use

1. Set the used property of all child items to false
2. Form a set of child items that have never been used. If the size of the set is less than __, the set becomes all child items.
3. Randomly select __ items and set their used property to true

Reset Runtime Data Process

Resets the runtime data when an item is launched. Data is not reset when an item is exited so that it is maintained for reporting and so that the display of status is intuitive to the user

1. Form a list of the item that was passed in and all of its parents
2. Traverse the list looking for the highest parent (if any) that has the runtime data to be reset property set to true
3. Form a list of all descendants of this highest parent
4. For each descendant item of the highest parent
    1. If the descendant item has its runtime data to be reset property set to true

        1. Reset the item's runtime data
        2. Increment the item's current attempt index
    2. Reevaluate rules referencing the reset activities

Launch Item

1. Set item's current property to true
2. Invoke reset runtime data process
3. Invoke update presentation

Commit

Invoked when the SCO calls Commit.

1. Form list of affected activities (algorithm TBD)
2. Loop until no more affected activities
    1. Run Rule Container Evaluation Process
    2. Form new list of affected activities (activities whose rules reference the recently changed rules)
3. Invoke Update Presentation

Terminate

Invoked when the SCO calls Terminate.

1. Form list of affected activities (algorithm TBD)
2. Loop until no more affected activities
    1. Run Rule Container Evaluation Process
    2. Form new list of affected activities (activities whose rules reference the recently changed rules)
3. Invoke Update Presentation
4. Invoke Navigate

Response to User Navigation

Invoked when the user clicks on an LMS provided navigation control

1. Set pending navigation request
2. Unload currently displayed SCO / Interstitial resource
3. Invoke the navigate process once unloading is complete

Navigate

1. Look for a pending navigation request
2. If there is no pending navigation request, launch the default interstitial resource

3. If the pending navigation request is Next
    1. Find the current activity in the directed navigation set
    2. Traverse the directed navigation set forwards until an item is found with the include in directed navigation property set to true
    3. If an activity is found, invoke the launch process on that activity
    4. Else, launch the default interstitial resource
5. Else if the pending navigation request is Back
    1. Find the current activity in the directed navigation set
    2. Traverse the directed navigation set backwards until an item is found with the include in directed navigation property set to true
    3. If an activity is found, invoke the launch process on that activity
    4. Else, launch the default interstitial resource
5. Else if the pending navigation request is Select
    1. If the specified item is a SCO
        1. Invoke the launch process on the specified item
    2. Else (the specified item is a cluster)
        1. Follow the process for a Next request, started at the first SCO in the directed navigation set that is a child of the specified item
    2. If no SCO is identifier for delivery, launch the default interstitial resource
2. Else if the pending navigation request is Exit Course
    1. Exit the course
2. Else if the pending navigation request is Exit to Interstitial Resource
    1. Launch the specified interstitial resource

Update Presentation

1. For each item
    1. Set table of contents (TOC) visibility according to Visible in Selection property (does not affect children, just the current item)
    2. Set TOC enabled according to Enabled in Selection property
    3. Reorder children if necessary to match ordinals
4. Set enabled status of continue button according to Directed Continue Navigation Enabled property
5. Set enabled status of previous button according to Directed Previous Navigation Enabled property
6. Set enabled status of exit SCO button according to Show Exit SCO Navigation property
7. Set enabled status of exit course button according to the Show Exit Course Navigation property

Guidance for Interpreting Completion and Satisfaction

- The LMS should interpret the status defined on the root activity to be the overall status for the course.

- While the completion status for the root activity is unknown or incomplete, the course is still in progress and the learner may continue to attempt it, regardless of the value of success status.
- Only when the value of completion status becomes completed, should the course be "graded", "submitted" or "moved to transcript" by the LMS.
- After the course is completed, it is up to LMS policy to decide whether or not the learner can re-launch it. The course should anticipate relaunch after completion and behave appropriately. A better mechanism for signaling to a course that it is being relaunched could be quite useful. Perhaps the current cmi.mode data model element should be made into a course property and extended to accommodate more modes.

User Interface Guidelines

- The LMS should provide a navigable table of contents to enable selection navigation. This may be hidden if all items are hidden from selection, otherwise, the table of contents should always be visible to the learner.
- The LMS should provide continue and previous buttons to enable directed navigation, unless these are explicitly disabled by rules.
- The LMS should provide buttons to close the current SCO and exit the course, unless these are explicitly disabled by rules.
- The LMS should provide some indication of which item is current.
- The LMS should provide a mechanism for the learner to know the completion status, satisfaction status and score for all activities.

Guiding Principals

The concepts below guided me in creating this beginning to Sensible Sequencing. For it to flourish, I believe it is imperative that we continue to adhere to them as (or, if) we evolve this specification any further.

- Straight forward
- Intuitive
- Simple
- No hidden dependencies
- Extensible
- Congruent with the rest of SCORM
- No such thing as a defined exception state. If we are are aware of the state, there will be defined behavior.