SCORM® 2004 3rd EDITION

Sharable Content Object Reference Model

Sequencing and Navigation

NOVEMBER 16, 2006
VERSION 1.0

Advanced Distributed Learning

このページは空白である.

Advanced Distributed Learning (ADL)

SCORM® 2004 3rd Edition
シーケンシング＆ナビゲーション(SN)
バージョン 1.0

ADLNet.gov から入手可能
(http://www.adlnet.gov/)

質問やコメントは ADL 問い合わせセンタ (ADLNet.gov) まで

このページは空白である.

日本語版訳者

eLC 標準化推進委員会

太田 衛	エネゲート㈱
大仲 輝	日立電子サービス㈱
仲林 清	NTT レゾナント㈱
増島 涼子	㈱富士通ラーニングメディア
宮内 浩	㈻産業能率大学
本村孝則	アベイズム㈱

このページは空白である.

チーフ テクニカル アーキテクト
Philip Dodds

テクニカル エディタ
Angelo Panar

謝辞:

ADL は, 相互運用可能なeラーニングの標準および仕様の作成に関する下記の組織およびメンバーの継続的･献身的な協力に感謝したい.

Alliance of Remote Instructional Authoring & Distribution Networks for Europe (ARIADNE) (http://www.ariadne-eu.org/)

Aviation Industry CBT (Computer-Based Training) Committee (AICC) (http://www.aicc.org/)

Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Committee (LTSC) (http://ltsc.ieee.org/)

IMS Global Learning Consortium, Inc. (http://www.imsglobal.org/)

ADL は ADL コミュニティに関しても SCORM の進化への貢献に対して感謝したい.

著作権

配給

本書の配給を許可するには以下の条件を満たす必要がある:

1. 本書および本書の図版および例示の使用は, 非営利目的で教育目的もしくは情報提供目的に限る.
2. 本書および本書の図版および例示は, 修正せずそのままの形であること. これには表紙および著作権, 配給, 再配布セクションも含まれる.

再利用

本書を全てまたは部分的に再利用するには以下の条件を満たす必要がある:

1. 再利用は, 非営利目的で教育目的もしくは情報提供目的に限る.
2. 情報元として以下を適切に引用する

   Source: Advanced Distributed Learning (ADL), Sharable Content Object Reference Model (SCORM®) 2004 3rd Edition Sequencing and Navigation Version 1.0, 2006.

著作権, 配給および再利用に関するより詳細の情報および質問は下記まで:

ADL Co-Laboratory Hub
1901 North Beauregard Street, Suite 600
Alexandria, Virginia 22311
USA
703-575-2000

目次

このページは空白である.

セクション1
SCORM シーケンシング＆ナビゲーション (SN)

このページは空白である.

1.1. SCORM シーケンシング＆ナビゲーション (SN) ブック概説

SCORM は，しばしば本棚の一組の本に例えられる．シーケンシング＆ナビゲーション (SN) は，その一組の本の中の一冊である．(図 1.1.a：SCORM ブックシェルフの一部としてのシーケンシング＆ナビゲーション参照)．他の本および本の間の関係についてのより詳細な情報は SCORM2004 概要に記述されている．SCORM SN は，学習者ないしシステム主導のナビゲーションイベントにより，SCORM コンテンツが学習者にどのように順序だてて提示される (sequenced) か，を記述する．コンテンツの分岐(branching)と流れ(flow)は，予め定義された一連のアクティビティ (Activity) により規定される．

図 1.1.a：SCORM ブックシェルフの一部としてのシーケンシング＆ナビゲーション

1.1.1. SCORM シーケンシング＆ナビゲーションの範囲

SCORM シーケンシング＆ナビゲーション (SN) では主要なコンセプトがいくつか導入される．その中では，実行時のコンテンツオブジェクト (SCO またはアセット) のシーケンシング，および，SCO からのナビゲーション要求に関する LMS の役割について取り扱う．さらに，学習者へナビゲーション制御を提供するための手引きも取り扱う．概要は以下の通りである．

- シーケンシングの概念と用語 (例：学習アクティビティ，アクティビティツリー，クラスタ)
- シーケンシング定義モデル (例：学習アクティビティに適用可能なシーケンシング情報の詳細定義および要件)
- シーケンシング動作モデル (例：規定されたシーケンシング情報および学習コンテンツにおける学習者の学習行為(learner's experience)に対する LMS の動作の詳細な記述
- ナビゲーション制御および要件
- ナビゲーションデータモデル

実行時の学習者の選択と成績(performance)に基づいてコンテンツを学習者に提示するために，コンテンツ･LMS 間通信において，SCORM シーケンシング＆ナビゲーションが活用される．この通信により，コンテンツが学習者に提示されている間，LMS は学習者の進捗状況と成績を記録することができる．本書は，シーケンシング動作が学習者の進捗状況をどのように記録するかを詳細に説明する．

1.1.2. SCORM シーケンシング＆ナビゲーション (SN) ブックの利用

本書は，自分たちの製品で SCORM をサポートしたい LMS ベンダおよびオーサリングツールベンダ，そして，シーケンシングと LMS の関係やどのようにシーケンシングがコンテンツに適用可能かを理解したい人，つまり SCORM コンテンツ開発者などに役立つであろう．

本書の冒頭の「セクション 1: SCORM シーケンシング＆ナビゲーション (SN)」および「セクション 2: シーケンシングの概念」では，SCORM シーケンシングに適用される概念を取り扱う．これらのセクションは，SCORM シーケンシングの背景となる概念を知りたい人，および技術的な詳細までは必要としない人にお薦めである．

「セクション 3:シーケンシング定義モデル」は，シーケンシングの技術的な詳細を提供する最初のセクションである．このセクションでは，コンテンツ開発においてシーケンシング戦略の記述に使用されるシーケンシング情報，およびそれらの使用例について説明する．

「セクション 4:シーケンシング動作」は，シーケンシングにおいてどのような情報が記録されるのか，コンテンツオブジェクトに対する学習者の進捗がどのようにトラッキング情報に影響するか，を詳細に記述する．このセクションは，SCORM シーケンシング動作を詳細に取り上げる．これには，シーケンシング情報をトラッキング情報に適用するための詳細な LMS 動作要件も含まれる．

「セクション 5:SCORM ナビゲーションモデル」は，コンテンツオブジェクトが LMS にシーケンシングの状態を問い合わせ，また，LMS にナビゲーション要求を行うためのランタイムデータモデルを記述する．本セクションでは，LMS が学習者に適切なナビゲーションコントロールを提供するためのガイドラインも提供する．

さらに，「付録 C」は，SCORM シーケンシング動作を明示的に定義するための，最新の詳細な標準擬似コードを提供する．

1.1.3. 他の SCORM ブックとの関係

SCORM SN は，学習実行時の学習者に対するコンテンツオブジェクトのシーケンシングに関する LMS の役割を記述する．SCORM では，コンテンツオブジェクトは，実行時に LMS と通信を行う SCO か，通信しないアセットかのどちらかである．SCORM SN は，様々なシーケンシング戦略を定義するのにシーケンシング情報がどのように適用されるのか，実行時にシーケンシング評価をおこなうためにシーケンシング情報がどのように解釈されるのか，そして，次のコンテンツオブジェクトを配信 (起動) するために，学習者のコンテンツオブジェクトに対するインタラクションによって発生したナビゲーション要求がどのように処理されるのかについて記述する．コンテンツオブジェクトの実際の起動に関しては，本書の対象外であり，SCORM RTE ブックに記されている[4].

以下のセクションは，SCORM SN ブックと他の SCORM ブックの関係を記述する．さらに，SCORM 全体についての詳しい知識がなくても本ブックを理解できるように，良く使われる用語を紹介する．LMS ベンダ，コンテンツ開発者，編集ツール開発者は，全ての SCORM 構成要素の目的，詳細，関係，および，利点を良く理解するために，SCORM の各ブックを読むよう強く推奨する．

1.1.3.1 SCORM コンテンツアグリゲーションモデルブック

SCORM コンテンツアグリゲーションモデル (CAM) ブックは, メタデータ, コンテンツパッケージ, および ADL シーケンシング&ナビゲーションのコンテンツパッケージ拡張に関する情報が含まれる. SCORM CAM ブックと SCORM SN ブックは相互に関連している部分がいくつかある.

メタデータは, 「データに関するデータ」である. 簡単に言うと, SCORM メタデータは, SCORM コンテンツモデルの異なるコンポーネント (コンテンツ構成, アクティビティ, SCO およびアセット) を記述する情報である. メタデータはコンテンツオブジェクトを検索・発見するために必要となる. 現時点で, SCORM SN ブックは, SCORM メタデータを使用していない. SCORM メタデータはナビゲーション要求もしくはシーケンシング動作の処理に影響を及ぼさない.

コンテンツパッケージは, 一般的な意味では, 規定されたコンテンツ構造でコンテンツオブジェクトをまとめることを指す. SCORM コンテンツパッケージは, SCORM コース, レッスン, モジュールもしくは単に SCORM レポジトリに保存された関連するコンテンツオブジェクトの集合を指す. 全ての SCORM コンテンツパッケージは, imsmanifest.xml ファイルを含む. このファイルは, パッケージされたコンテンツを表し, オプションでコンテンツ構造の記述を含む.

SCORM コンテンツパッケージは追加情報として, コンテンツパッケージの処理と, コンテンツを管理に関する LMS の意図された動作の記述を含む場合がある. この情報のいくつかは SCORM SN ブックで使用されている.

- SCORM コンテンツパッケージのいくつかの要素は, コンテンツオブジェクトのランタイムデータモデルの初期化および管理に影響がある. これらの要素は, SCORM SN ブックで記述される動作に影響がない.
- SCORM コンテンツパッケージの他の要素は, コンテンツオブジェクトのランタイムデータモデルの特定の要素の初期値を記述する. シーケンシングは, 配信するためのコンテンツオブジェクトを特定するに過ぎない. 従って, これらの要素は, SCORM SN ブックで記述される動作に影響がない.
- コンテンツオブジェクト起動ロケーションおよび起動パラメータも, SCORM コンテンツパッケージでは要素として記述される. シーケンシングは, 配信するコンテンツオブジェクトを特定するだけである. 従って, これらの要素は, SCORM SN ブックで記述される動作に影響がない.
- SCORM コンテンツパッケージがコンテンツ構造の記述を含む場合, シーケンシング情報が追加され, パッケージのコンテンツオブジェクトのシーケンシングに関する意図した処理方法が定義される. SCORM SN は, コンテンツパッケージで定義されたコンテンツ構造がどのようにアクティビティツリー (Activity Tree. シーケンシングにおいて使用される基本構造) として解釈されるかを定義する.
- SCORM コンテンツパッケージは, LMS がどのようにそれぞれの UI ナビゲーションコントロールを提示するか, 実行可能にするか, もしくは隠すかに関する指示を提供するためのユーザインターフェース (UI) 要素を含むことがある. SCORM SN で記述されるシーケンシング&ナビゲーション動作は, UI ナビゲーションコントロールが (ナビゲーションイベントを) 実行可能にするかいなかには依存しない. UI ナビゲーションコントロールは LMS が発行するナビゲーション要求の処理だけに関与する.

シーケンシング&ナビゲーション規格の要素が SCORM コンテンツパッケージでどのように指定されるかを充分に理解するには, SCORM CAM ブックを参照する必要がある[3].

1.1.3.2 SCORM ランタイム環境ブック

SCORM ランタイム環境は, 実行時の LMS (Learning Management System) とコンテンツオブジェクトの責任範囲について記述する. SCORM では, コンテンツオブジェクトは, 実行時に通信を行う SCO か, 通信

しないアセットかのずれかを指す. SCORM RTE は, 共通コンテンツオブジェクト起動メカニズム, コンテンツオブジェクトと LMS 間の共通通信メカニズム, コンテンツオブジェクトで学習者履歴を記録するための共通データモデルを記述する. これらの観点は, ADL 高レベル要件のいくつかを満たす環境を作る. 例えば, 標準通信メカニズムによって通信するコンテンツオブジェクトは, 通信手段に修正を加えることなく, LMS から他の LMS へ移植することが出来る. これは, 学習オブジェクトの移植性, 耐用性を増し, それにより開発, 実装, 保守コストが低減する.

1.2. SCORM シーケンシング概要

SCORM SN の一部は, IMS シンプルシーケンシング (SS) 仕様[1]に基づいている. IMS SS 仕様は, どんな LMS においても一貫性のある手順で学習アクティビティをシーケンスするように, 作成した学習コンテンツの意図した動作を表現する方法を定義している. IMS SS がシンプルと呼ばれるのは, 数あるシーケンシング動作のうち限られたものを定義するからであり, 仕様そのものが単純だからではない. IMS SS は全てを対象にしているわけではない. たとえば, 人工知能に基づくシーケンス, スケジュールに基づくシーケンス, 閉じた外部システムおよびサービスからデータを必要とするシーケンシング (例:組み込まれたシミュレーションのシーケンシング), 協調学習, カスタマイズされた学習, 複数の並行の学習アクティビティの間の同期を取ることなどについては除外もしないが言及もしない.

IMS SS は, 学習者の役割だけを定め, 講師, メンター, 同僚などその他のアクターに依存したり, 使用したりするようなシーケンシング機能を定義しない. SCORM SN ブックは他のアクターを含んだ文脈での使用を禁止しないが, 他のアクターの参加によって生じるシーケンシング動作や, 他の関係者の役割を定義しない.

SCORM SN ブックは, IMS SS 仕様が SCORM 環境でどのように適用され, 拡張されるかについて定義する. SCORM に対応している LMS が, 実行時にシーケンシング情報を処理できるように実装しなければならない動作や機能に関して定義する. 具体的に言うと, 起動されたコンテンツオブジェクトと作成されたシーケンシング戦略に対する学習者のインタラクションの結果に基づく, 学習アクティビティのアクティビティツリーの中での分岐と流れを記述する.

SCORM は, いつどのようにアクティビティツリーを作成するかについて, あるいは, アクティビティツリーの内部表現や実行時のアクティビティツリー管理について, LMS にどのような要求もしない. しかし, SCORM CAM は, SCORM コンテンツパッケージの拡張によりシーケンシング情報のひとつの表現を定義し, これによって異なる実行時要素, すなわち LMS 間で, コンテンツ構造とシーケンシング情報を交換する相互運用性のある方法を提供する.

総括すると, SCORM シーケンシングは, 学習アクティビティの定義された構造であるアクティビティツリー, シーケンシング戦略を定義したシーケンシング定義モデル, そして, 外部ないしシステムが発生したイベントに対する定義された動作を適用した SCORM シーケンシング動作により規定される.

1.3. SCORM ナビゲーション概要

SCORM SN は, 学習者やシステムからナビゲーションイベントがどのように発生して処理され, 最終的に配信される学習アクティビティが特定されるかについて記述する. 配信のために特定された各学習アクティビティは, 関連付けられたコンテンツオブジェクトを持っている. SCORM RTE ブック[4] (セクション 2.1.2:コンテンツオブジェクトの起動) は, 指定されたコンテンツオブジェクトがどのように起動されるかを記述している. ある学習者とコンテンツ構造に対して, 起動したコンテンツオブジェクトのシーケンスは, 固有の学習行為 (学習者とコンテンツオブジェクトとのインタラクション) を提供する. SCORM RTE ブックは, 実行された学習行為を SCO のために LMS がどのように管理するか, および, 学習行為がアクティビティツリーにどのように影響するかについて記述する.

ナビゲーションは, ナビゲーションイベントを発生させるユーザインターフェース機能の存在を前提とする. これらの機能は, LMS により提供されるかコンテンツオブジェクトに組み込まれている. 学習者がこのような機能を操作すると, LMS はそのイベントを対応するナビゲーション要求に変換し, 要求を処理し, そして次に配信する学習アクティビティを特定する. SCORM SN は, ナビゲーション要求を SCO が LMS へ伝えるのに使用するランタイムデータモデルを記述する.

SCORM SN は, ナビゲーションおよび補助サービスにアクセスするユーザーインターフェース機能を含めて, 実行時に学習者に提示するユーザインターフェースの種類もしくはスタイルに対して何も要求をしない. ユーザーインターフェースの性質および学習者と LMS 間の対話メカニズムは, あえて規定されていない. ルック&フィール, 提示スタイル, ユーザインターフェース機能ないしコントロールの配置といった課題は SCORM の対象外である. しかし, 公式のナビゲーション (およびプレゼンテーション) 仕様もしくは標準が開発されるまでの間, SCORM ナビゲーションモデルを解釈する労力を減らすための推奨事項が提供される.

セクション2
シーケンシングの概念

このページは空白である.

2.1. コンテンツ構成およびアクティビティツリー

コンテンツ構成図は，学習行為の階層的な関係を定義するために，学習設計関係者が使用する一般的なツールである．IMS SS 仕様は，学習アクティビティの構成を定義するのに，アクティビティツリーと呼ばれる同様のコンセプトを定義し使用している．アクティビティツリーにより，SCORM シーケンシング&ナビゲーションモデルにおいて，シーケンシングアルゴリズムや動作といった情報および処理に関する要件を，実装とは分離した形で定義することが可能になる．図 2.1a はアクティビティツリーの例である．アクティビティツリーのルートはコース（Course）である．アクティビティツリーのルートは，上記に定義された学習アクティビティでもあり，より具体的には（殆どのケースで）クラスタである．

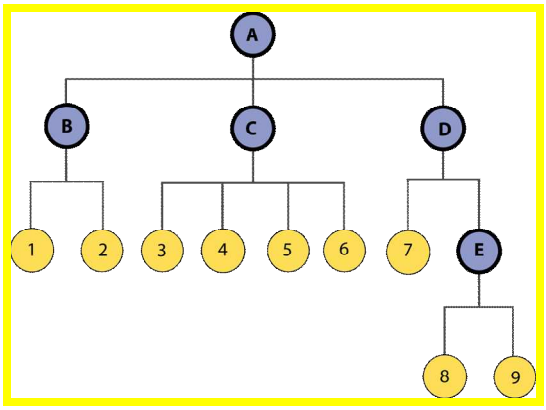

図 2.1a：アクティビティツリーの例

シーケンシングを実装するシステムは，そのように要求されてはいないが，アクティビティツリーの独自の内部表現を有すると想定される．その内部表現はツリーデータ構造になっていることもなっていないこともありうる．SCORM は LMS が，いつどのようにアクティビティツリーを作成するかについて定義していない．さらに，SCORM は，アクティビティツリーが常に静的な構成であることを要求していない．シーケンシング定義モデル（セクション 3：シーケンス定義モデル参照）およびシーケンシング動作（セクション 4：シーケンシング動作参照）に基づいている実行しているかぎり，アクティビティツリーの構成およびアクティビティツリーのアクティビティに適用されたシーケンシング情報を適時ダイナミックに変えることができる．もし，学習者がそのアクティビティと関連したコンテンツオブジェクトとやりとりしている間に，アクティビティツリーをダイナミックに変更することを選択した場合，LMS が実行中の学習行為を中断しない形で行うことが推奨される．

繰り返し述べると，SCORM では，オーサリングツールおよび LMS がどのようにアクティビティツリーを実装するか，もしくはインストラクショナルデザインの手法がアクティビティツリーを活用するために，どのように修整されるかは制約されない．アクティビティツリーとは，指定されたシーケンシング動作を相互運用可能な形で適用するための，階層的な学習アクティビティとそれに対応したシーケンシング情報を表す一般的な用語である．

2.1.1. コンテンツパッケージからのアクティビティツリーの誘導

SCORM CAM [3]は，学習コンテンツの階層型の構造を定義する．これは，コンテンツパッケージで単一の<organization>要素で表現されるコンテンツオーガニゼーション(Content Organization)である．階層構造の各アイテムは学習単位を表す．アイテムは，任意の深さにネストでき，学習分類上の名称をつけるこ

とが可能である. 例えば, アイテムは, コース, モジュール, ユニット, レッスンなどを示すことがある. 階層型コンテンツ構造は, コンテンツ交換のために, 従来からコンテンツパッケージにおけるオーガニゼーションという形で表現されている.

SCORM シーケンシング動作は構造化された学習アクティビティを用いて定義されているため, コンテンツ構造は, アクティビティツリーを導出するための開始点を提供する. シーケンシングに関しては, コンテンツオーガニゼーションが一つの相互運用可能なアクティビティツリーの構造を表し, 各々の<item>要素が学習アクティビティに対応する. 望ましい学習行為に合致した固有のシーケンシング実行時動作を定義するために, シーケンシング定義モデル要素がアイテムに適用される.

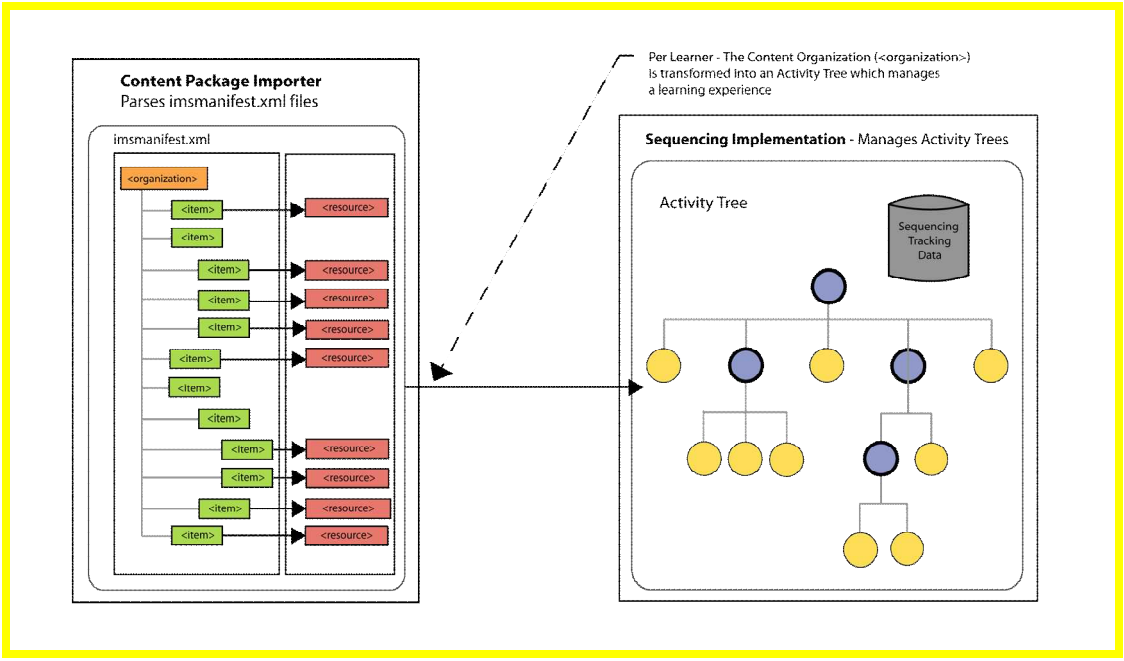

図 2.1.1a: コンテンツオーガニゼーションとアクティビティツリーの関係

コンテンツオーガニゼーションとアクティビティツリーの関係は 図 2.1.1a に図示されており, 以下のようにまとめることができる:

1. アクティビティツリーは, コンテンツ設計, 編集および集約プロセスから得られる概念的なコンテンツ構造を表す. アクティビティツリーは, SCORM コンテンツパッケージでは最終的にコンテンツオーガニゼーション(<organization>要素)として表現され, シーケンシング情報の相互運用性のある交換が可能になる. 例えば, オーサリングツールは, コンテンツ階層を現す内部データ構造を固有のフォーマットで実装することができる. この構造は, 開発者が学習行為を定義するために用いたインストラクショナルデザインのプロセスもしくは手法から得られる. 開発プロセスが完了すると, オーサリングツールは固有の内部フォーマットを SCORM CAM で定義されたフォーマットに変換する. これは, コンテンツ・アグリゲーション・パッケージング・アプリケーション・プロファイル[3]に準拠したフォーマットで, SCORM コンテンツパッケージを理解するすべてのシステムに取り込むことができる.

2. SCORM 対応 LMS は, コンテンツオーガニゼーションをアクティビティツリーに変換する. アクティビティツリーは, LMS が実装するデータ構造を表わし, 定義された学習アクティビティの階層的な内部表現を反映しており, 学習者個人ごとに各アクティビティの状態記録情報を含んでいる.

3. 学習者がアクティビティツリーに示されるコンテンツとやりとりすることを選択したとき, LMS はシーケンシングおよびトラッキング情報を評価して, 学習アクティビティの相対的シーケンスを決定するとともに, 学習者が実行しようとしている学習アクティビティが適切か否かをコンディションを用いて評価する. このとき同じコンテンツ構造でも各学習者の経験が異なることがある. これは, コンテンツ開発者が定義したシーケンシング情報および学習者のコンテンツオブジェクトとの具体的なやりとりに依存する.

2.1.2. シーケンシングコレクションの利用

「コンテンツパッケージからのアクティビティツリーの誘導」(2.1.1 参照) で述べたように, SCORM コンテンツアグリゲーションモデル[3]は構造化された学習コンテンツの交換の構造を提供している. この構造中の各ノードは, 誘導されたアクティビティツリーを構成する学習アクティビティを表す. アクティビティツリー全体を通して, 共通するシーケンシングのねらいが繰り返しあらわれ, シーケンシング情報のパターンとなることがよくある. このような場合, 全く同じでないとしても類似のシーケンシング情報の組が, アクティビティツリー中の複数の学習アクティビティに適用される.

複数ノードにまたがる冗長なシーケンシング情報の記述を最小化するため, SCORM コンテンツアグリゲーションモデル[3]は, この共通なシーケンシング情報の組を宣言するコンテナを提供している. <sequencingCollection>要素によって, 指定されたシーケンシング情報の組を, コンテンツ構造中の複数のノードから参照・再利用することができる.

ある学習アクティビティに適用されるシーケンシング情報を導く時, 以下のルールが適用される.

- <sequencingCollection>からの参照されたシーケンシング情報は, 学習アクティビティに直接適用されているシーケンシング情報と「統合」しなくてはならない. この「統合」は「最上位レベル」の IMS SS XML 要素, つまり, Sequencing Control Modes, Sequencing Rules, Rollup Rules, Objectives,などで行う.
- 学習アクティビティ (<item> ないし <organization>) に直接適用されている「最上位レベル」の IMS SS XML 要素が, 参照されたシーケンシング情報中の同じ要素よりも優先されなくてはならない. すなわち,「最上位レベル」の IMS SS XML 要素が, ノードと参照されたシーケンシング情報の双方に含まれる場合, 参照された要素とそのすべての子要素を「統合」してはならない.
- シーケンシング情報を「統合」したとき, 統合した要素およびそのすべての子要素は, 学習アクティビティ適用されるシーケンシング情報の一部とみなさなくてはならない.
- ADL ネームスペース (adlseq および adlnav) 拡張要素は,「最上位レベル」要素とみなして, IMS SS XML「最上位レベル」要素と同様の方法で統合しなければならない.

2.1.3. クラスタ

クラスタ (Cluster) は, サブアクティビティを持つ特殊な形の学習アクティビティである. この用語は様々なシーケンシング動作で使用される. クラスタは, 一つの親アクティビティおよび直下の子アクティビティを含むが, それ以下の子孫は含まない. クラスタの子は, 葉アクティビティもしくは他のクラスタである. 葉アクティビティはクラスタではない.

図 2.1.2a は, 5 個のサンプルクラスタを表す. 各クラスタは, 破線で囲まれているように定義される.「コース (Course)」のクラスタであるクラスタ A は, 4 個のアクティビティだけを含む. すなわち,「コース」アクティビティとクラスタ B, C および D の親アクティビティである. 各「モジュール」クラスタであるクラスタ B, C および D は,「モジュール (Module)」アクティビティおよびモジュールの「レッスン (Lesson)」から成り立つ.「モジュール 3」の「レッスン 2」を除いた全「レッスン」アクティビティは, コンテンツオブジェクトと関連する

葉学習アクティビティである．「モジュール 3」の「レッスン 2」は 2 個の「チャプター (Chapter)」葉学習アクティビティから成るクラスタである．

クラスタはアクティビティツリーの基礎ブロックとみなすことができ，シーケンシング定義モデル (セクション 3: シーケンシング定義モデル参照) の多くの要素は実際にはクラスタに適用される．クラスタの親アクティビティは，クラスタのシーケンシング戦略に関する情報を含んでいる．クラスタのクラスタでない子 (葉アクティビティ) は，定義されたシーケンシング戦略に従って配信されるコンテンツオブジェクトと関連する．

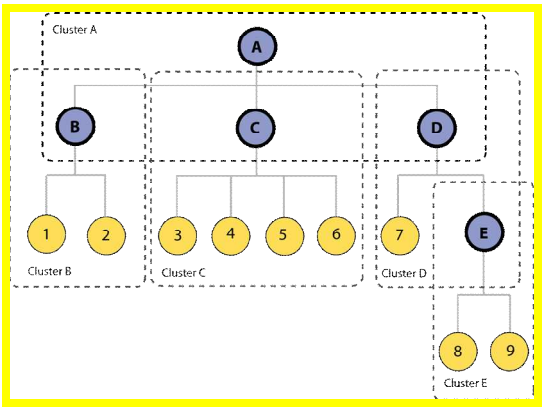

図 2.1.2a: クラスタの例

2.1.4. コンテンツパッケージにおける (サブ) マニフェストの使用

SCORM コンテンツアグリゲーションモデル[3]は，現在，コンテンツパッケージ中の (サブ) マニフェストのサポートを要求していない．そのため，(サブ) マニフェスト中のシーケンシング情報の使用，および，(サブ) マニフェストを含むコンテンツパッケージからのアクティビティツリーの誘導は未定義である．(サブ) マニフェストをサポートするか否か，また，どのようにサポートするかは LMS 実装に任されている．

ADL ノート: IMS グローバルコンソーシアムが IMS コンテンツパッケージ仕様の新しいバージョンの作業を行っている．IMS が解決しようとしている大きな課題のひとつで (サブ) マニフェストの使用法に関する要件および XML 文法に関する要件が扱われる．現時点で，IMS の作業が完了するまで (サブ) マニフェストを使わないよう ADL は推奨する．(サブ) マニフェストに関する質問，懸案，新たな推奨は ADL に送ることが望ましい．

2.1.5. 学習アクティビティ

IMS シンプルシーケンシング (SS) 仕様は, 学習アクティビティ (Learning Activity) の概念に依存する. 学習アクティビティ (図 2.2a) は, おおまかには学習に関する意味のある単位と言うことができる. つまり, 概念的には, 学習を進めている間, 学習者が行う何かである. 学習アクティビティは, 学習者へ学習リソースを提供するか, もしくは, いくつかのサブアクティビティから構成される. この文書では, 「アクティビティ」という用語は「学習アクティビティ」と同義語である.

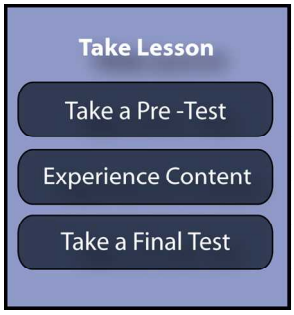

図 2.2a: 学習アクティビティの例

図 2.2a で, 「レッスンを受ける (Take Lesson)」アクティビティは, 3 つのサブアクティビティから成り立つ. 「プリテストを受ける (Take a Pre-Test)」, 「コンテンツを経験する (Experience Content)」および「修了テストを受ける (Take a Final Test)」. 学習者は, 「レッスンを受ける」というアクティビティの学習の中でこれらのサブアクティビティを学習する.

サブアクティビティは, さらに何層にもネストしたサブアクティビティから成り立つ事がある. 下位のサブアクティビティがないサブアクティビティは, 葉アクティビティと呼ばれる. 葉アクティビティは, コンテンツオブジェクトと関連付けられている. LMS は, 前に学習した学習アクティビティにおける学習者の進捗, 学習者の意図および作成されたシーケンシング情報に基づいて, ランタイム時に決定された順序で配信するための学習アクティビティを特定する.

コンテンツオブジェクトは, 葉学習アクティビティの学習において, 学習者によって学習される. 葉学習アクティビティの試行 (Attempt) が始まると, 関連するコンテンツオブジェクトが学習者に対して起動され, 学習者の試行と学習者セッションの両方がそのコンテンツオブジェクトに対して開始する. 学習者が経験する連続するコンテンツオブジェクトは学習行為と呼ばれる.

全ての学習アクティビティは以下の特徴を持っている:

- 学習アクティビティは, 区分された開始と終了を持つ
- 学習アクティビティは, 明確に定義された完了と習得の条件を持つ
- 学習アクティビティは, 下に何階層にもネストしたサブアクティビティを持つ事ができる
- 学習アクティビティ (の試行) は, 親アクティビティがあればその (試行の) 中で発生する

2.1.6. 試行

試行 (Attempt) はアクティビティを完了するための動作と定義され, 試行している間, ゼロもしくはそれ以上の学習目標が習得されることがある. アクティビティへの試行は, 常にその親アクティビティへの試行の中で発生する. どのアクティビティツリーにおいても, どの時間を取っても, 試行される葉アクティビティは一つだけであること, および葉アクティビティが試行されている間, その上位のルートまでの全てのアクティ

ビティの試行が全て進行していることに留意しなければならない．葉アクティビティが試行されているとき，このアクティビティに対応するコンテンツオブジェクトが起動していると考えてよい．

試行は，アクティビティが配信のために選択された時に始まり，LMS のシーケンシング実装が次の配信用アクティビティを選択しているときに終了する．アクティビティへの試行は，そのアクティビティに対応するコンテンツオブジェクトへの学習者試行と密接に関係している．SCORM RTE ブック[4]（セクション 2.1：ランタイム環境（RTE）管理）は，コンテンツオブジェクトの時系列的なモデルを詳細に説明している．必ずしも1回の試行でアクティビティを完了できるとは限らない．多くの場合，学習者はアクティビティを一旦中断し，後に再開する．殆どのケースで，中断されたアクティビティを再開する場合，新しい試行を開始するわけではなく，現在の試行が継続される．

アクティビティへの試行の結果として，もしくは，いくつかの外部の処理を通して，アクティビティのトラッキング状態を変えることができる．（セクション 4.2：トラッキングモデル参照）．アクティビティのトラッキング状態が変わるとき，上位のトラッキング状態に影響を及ぼすことがある．これはロールアップと呼ばれる（セクション 4.6：ロールアップ動作参照）．

2.2. シーケンシングセッションの開始と終了

シーケンシングセッションは，アクティビティツリーのルートアクティビティへの試行が始まってから試行が終わるまでの期間を指す．SCORM シーケンシング動作は，どのナビゲーション要求がシーケンシングセッションを始めることができるかを規定するだけで，いつどのようにこのナビゲーション要求を発生させるかは規定しない．一般的に，LMS は何らかのシステムイベント（例：ログイン，コース開始など）を認識して Start ナビゲーション要求を出す．前シーケンシングセッションが Suspend All ナビゲーション要求によって終了した場合，LMS は Start ではなく Resume All ナビゲーション要求を出さなくてはならない．

いくつかのケースでは，Start もしくは Resume All ナビゲーション要求がうまく行かず，有効な Choice ナビゲーション要求だけがシーケンシングセッションを開始する場合がある．有効な Choice ナビゲーション要求を発生するメカニズムを提供するのは LMS の役割である．シーケンシングセッションは，アクティビティツリーのルートアクティビティで Exit シーケンシング要求が処理されると終了する．これは，Exit All（ログアウト）もしくは Suspend All（ポーズ）ナビゲーション要求，あるいは，アクティビティツリーのルートへの *exit action sequencing rule*（セクション 4.5：終了動作）の適用によって引き起こされる．

2.3. アクティビティ状態トラッキング

SCORM シーケンシング動作は，シーケンシングトラッキング状態モデル（セクション 4.2:トラッキングモデル参照）の値に基づいてシーケンシング動作を制御する．学習者のアクティビティへの各試行に対して，そのアクティビティは対応するトラッキング状態データを持つ．学習者のコンテンツオブジェクトとのインタラクションは，コンテンツオブジェクトに対応するアクティビティのトラッキングデータに影響を与える．トラッキングデータは様々なシーケンシングプロセスで使用され，その動作に影響を与える．

2.3.1. 通信型および非通信型コンテンツ

SCORM シーケンシングでは，通信型コンテンツと非通信型コンテンツが区別される．通信型コンテンツは，学習者のコンテンツとのやりとりに関する情報を SCORM ランタイム API[4]（セクション 3: アプリケーションプログラミングインターフェース）を通してやりとりする．一方，非通信型コンテンツは，SCORM ランタイム API を使用しない．SCORM シーケンシングは，両方のコンテンツをアクティビティ別にサポートする．

SCO は，SCORM ランタイム API および SCORM ランタイム環境データモデル[4] (セクション 4: SCORM ランタイム環境データモデル)によって学習者の進捗状況を通信しなければれならない．LMS は，学習者の進捗状況に関して通信されない情報にはいかなる仮定もおかない．アセットの場合，LMS は定義された既定値およびデフォルトの動作に基づき，自動的に学習者の進捗情報を設定する．

2.3.2. アクティビティの中断および再開

アクティビティへの試行は，中断され，後に再開されることがある．中断されたアクティビティを再開することは，新しい試行とは数えない．アクティビティが中断されている間に他のアクティビティが試行されることがある．一つ以上の試行が同時に中断されることもある．

アクティビティツリーのルートアクティビティへの試行を中断すると，LMS は学習者が最後に学習したアクティビティを記憶し，シーケンシングセッションを中断状態で終了する．学習者が後でアクティビティツリーのルートへ試行を再開すると，学習者が最後に経験したアクティビティも再開される．

2.3.3. データ保持

管理，トラッキング，状態情報を，少なくともアクティビティツリーのルートアクティビティへの現在の試行が終わるまでは保持する必要がある．このような試行は複数のシーケンシングセッションにまたがる場合がある．SCORM も IMS SS 仕様も，試行の間に保持すべきデータ（例：シーケンシング情報およびトラッキング状態データ）をどのように格納するか規定していない．これは，セッション中，および，セッションの間保持すべきデータについても同様である．管理，トラッキング，状態情報を，アクティビティツリーのルートアクティビティへの試行が終わった後，保持する必要はない．そのようなデータを，例えば，学習者の活動の監査，分析，履歴記録のために保持するかどうかは LMS の方針が決めるものである．そのような方針は SCORM の範囲外である．

2.3.4. 学習目標

学習目標は学習アクティビティとは別のものである. SCORM はどのように学習目標が学習アクティビティと関連するかについては制約しない. また, コンテンツオブジェクトがどのように学習目標を使用するかについても定義していない. SCORM シーケンシング動作は, 学習目標をどのように解釈するか (例:それは, コンピテンシーなのか, 習得なのか, もしくは単に共有値なのか) についても仮定しない. トラッキングの観点から, 学習アクティビティと関連した各学習目標に, 一連の学習目標ステイタス情報 (学習目標習得値および学習目標習得度) が設定されている.

アクティビティは, 一つ以上の学習目標と関連することがある. しかし, SCORM SN モデルは, 一つのアクティビティに関連する複数の学習目標の意味については仮定してしない. アクティビティの学習目標として保持される学習目標ステイタス情報は, デフォルトではそのアクティビティに固有 (local) のものである. 学習目標ステイタス情報を共有するために, 一つのアクティビティは, 複数の共有グローバル学習目標を参照する場合がある. 複数のアクティビティが, 同じ共有グローバル学習目標を参照する, つまり, その学習目標ステイタス情報を共有することができる. 共有グローバル学習目標は, 単体のアクティビティツリー内で共有化されたり, LMS 内の複数のアクティビティツリーに渡って共有化されることもある. 一つのアクティビティが, 共有グローバル学習目標をどのように参照するのかに関しては, 二つの制約がある.

1. ローカル学習目標は, 一つの共有グローバル学習目標からのみ学習目標ステイタスデータを読み取ることができる.
2. ある特定のアクティビティに定義された一組のローカル学習目標では, 二つのローカル学習目標が, 同一の共有グローバル学習目標に学習目標ステイタスデータを書き込むことはできない.

このページは空白である.

セクション3
シーケンシング定義モデル

このページは空白である.

3.1. シーケンシング定義モデル概要

SCORM シーケンシング定義モデルは, IMS シンプルシーケンシング (SS) 仕様[1]から派生した情報モデルである. IMS SS シーケンシング定義モデルは, 様々なシーケンシング動作を記述したり, 変更したりするのに使用できる一連の要素を定義する. さらに, SCORM に固有な要素がいくつか定義され, これらはアプリケーションプロファイルに特有の拡張された動作と制約で, IMS SS 仕様で現在定義されている以上のものを提供する.

SCORM シーケンシング定義モデルは, コンテンツ開発者が意図するシーケンシング動作を定義するのに使用する一連の要素を定義する. 定義モデル要素は, アクティビティツリーの中で学習アクティビティに適用される. 各要素は, 明示的に定義された値がない場合, シーケンシング機能が前提とする既定値を持つ. SCORM シーケンシング定義モデル要素の効果は, SCORM シーケンシング動作 (セクション 4: シーケンシング動作参照) の適用中にのみ有効である. SCORM 対応 LMS は, 全ての定義されたシーケンシング定義モデル要素に関連付けられた値の結果として得られる動作をサポートしなければならない. これらの値は, 明示的に定義された値と既定値の両方を含む. 正規のシーケンシング動作の詳細は, シーケンシング動作擬似コード (付録C参照) に記述されている.

SCORM は, アクティビティに適用されたシーケンシング定義モデル要素の値が, ある期間, 静的である, または静的になる, または静的に留まるということを示唆もしないし要求もしない. 要素の値空間に従う限り, LMS は要素の値を必要に応じて変更することができる. しかし, シーケンシング定義モデル要素のいくつかのグループは, SCORM シーケンシング動作を通じて, 互いに強く結び付いている. SCORM シーケンシング定義モデル要素の値を変える場合, 特に学習行為の実行中は, 操作に細心の注意を払うことが強く望まれる.

SCORM は, SCORM シーケンシング定義モデル要素が, いつもしくはどのように学習アクティビティに適用されるかについては何の要請もしていない. しかし, SCORM CAM ブック[3]は, これらの要素が SCORM コンテンツパッケージに含まれたコンテンツオーガニゼーションにどのように適用されるかを記述している. コンテンツパッケージからアクティビティツリーの導出 (セクション 2.1.1 参照) に記述されているように, SCORM シーケンシング定義モデル要素は, コンテンツパッケージが処理されたとき, 導出されたアクティビティツリーのアクティビティに適用される. これは, 教材作成時に宣言した意図したシーケンシング動作が, コンテンツオーガニゼーションを通して通信されることを可能とし, これにより, SCORM コンテンツパッケージを使用したシステム間でシーケンシング情報が相互運用できる形で交換できるようになる.

3.2. シーケンシングコントロールモード

シーケンシングコントロールモードによって, ナビゲーション要求がクラスタにどのように適用されるか, および, シーケンシング要求が処理されている時にクラスタのアクティビティをどのように扱うかを, コンテンツ開発者が制御することができる. シーケンシングコントロールモードは, 必要に応じて, 望ましい学習行為を制約するために適用される. コントロールモードは以下の方法で使用される:

- ナビゲーション要求 (セクション 4.4: ナビゲーション動作参照) を処理中, 要求が有効なシーケンシング要求に変換されるかどうかを決定するため
- 様々なシーケンシング要求がサブプロセス (セクション 4.8: シーケンシング動作参照) を実行している間, 配信対象となるアクティビティを選択する方法を制御するため

- 様々なシーケンシング動作の間，どのようにトラッキング状態情報が管理されるかを制御するため（セクション 4.2：トラッキングモデル参照）

表 3.2a に適用可能なシーケンシングコントロールモードが説明されている．シーケンシングコントロールモードはアクティビティツリーのどのアクティビティにも適用することができるが，*Sequencing Control Choice, Sequencing Control Flow* および *Sequencing Control Forward Only* モードは葉アクティビティに適用された場合は効果がない．複数のモードを同時に使用してコントロールモード動作の組み合わせを作成できる．どのような場合にも，あるアクティビティのコントロールモードが他のアクティビティに影響を与えることはない．つまり，コントロールモードは受け継がれない．もしコントロールモードがアクティビティに対して明示的に定義されていない場合，表 3.2a に示したデフォルト値が適用される．

表 3.2a: シーケンシングコントロールモードの説明

No.	名称	説明	値空間	既定値
1	Sequencing Control Choice	*Choice*ナビゲーション要求をアクティビティの子に対して発行してよいことを示す．	論理型	True
2	Sequencing Control Choice Exit	*Choice*シーケンシング要求が処理されたときに，このアクティビティが終了してよいことを示す．	論理型	True
3	Sequencing Control Flow	フローサブプロセスをこのアクティビティの子に適用してもよいことを示す．	論理型	False
4	Sequencing Control Forward Only	（アクティビティツリーの探索に関して）後もどりがアクティビティの子に対して許されないことを示す．	論理型	False
5	Use Current Attempt Objective Information	あるアクティビティの子の学習目標進捗情報が，そのアクティビティの現在の試行中に記録された場合だけ，ルール評価とロールアップで使われることを示す．	論理型	True
6	Use Current Attempt Progress Information	あるアクティビティの子の試行進捗情報が，そのアクティビティの現在の試行中に記録された場合だけ，ルール評価とロールアップで使われることを示す．	論理型	True

3.2.1. Sequencing Control Choice

Sequencing Control Choice 要素は，学習者がクラスタのすべてのアクティビティをいかなる順序でも制約を受けずに自由に選ぶことができることを示す．この要素は，論理型（True/False）の値を持つ．デフォルトでは，アクティビティツリー全体で，親の *Sequencing Control Choice* が True の子アクティビティはすべて *Choice* ナビゲーション要求の有効な対象となる．いくつかのケースにおいて，コンテンツ開発者が，ある特定の条件の下に，学習者にアクティビティを選択させることがある．*Choice* ナビゲーション要求の対象は，*Sequencing Control Choice Exit* 要素（セクション 3.2.2 参照），*Constrained Choice* コントロール要素（セクション 3.3 参照），もしくは *Hidden From Choice* プリコンディションシーケンシングルール（セクション 3.4 参照）を適用することで，条件的に制約することができる．

LMS は，親アクティビティの *Sequencing Control Choice* が True に設定されていて *Choice* ナビゲーション要求の対象となりうるアクティビティを，学習者が「選択する」ための何らかのメカニズム（メニュー，マップ，目次といったユーザーインターフェースナビゲーションコントロール）を提供する必要がある．学習者が選択可能なアクティビティを選択したとき，シーケンシング動作（セクション 4.8.6.7：*Choice* シーケンシング要求サブプロセス参照）は要求されたアクティビティをアクティビティツリーで探索する．要求されたアクティビティは，他のシーケンシング情報によって妨げられない限り配信対象として特定され，アクティビティに関連するコンテンツオブジェクトが学習者のために起動される．

Sequencing Control Choice コントロールモードは，葉アクティビティに定義された場合，無効である．

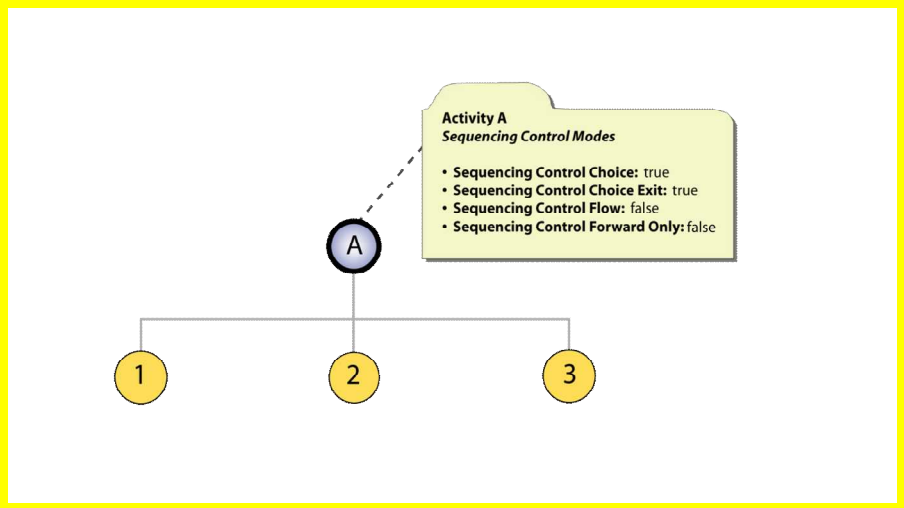

図 3.2.1a：デフォルト Sequencing Control Choice 動作

図 3.2.1a は，*Sequencing Control Choice* 要素のデフォルト動作を示している．親アクティビティ A の *Sequencing Control Choice* は True に設定されているので，アクティビティ 1, 2, 3 は Choice ナビゲーション要求の有効な対象である．アクティビティ A は，その親の *Sequencing Control Choice* が True に設定されているか，もしくは，自身がアクティビティツリーのルートアクティビティでない限り，*Choice* ナビゲーション要求の有効な対象ではない．

もし学習者が，*Choice* ナビゲーション要求の有効な対象であるクラスタを選択した場合，以下の二つのうちのいずれかの結果となる：

1. 図 3.2.1b に示すように，*Choice* ナビゲーション要求の対象（アクティビティ B）の *Sequencing Control Flow* が True である．この場合，アクティビティ B の子アクティビティを，葉アクティビティが配信対象に指定されるまで，順序付きツリー検索を行う必要がある．この例ではアクティビティ 1 が配信対象として特定される．

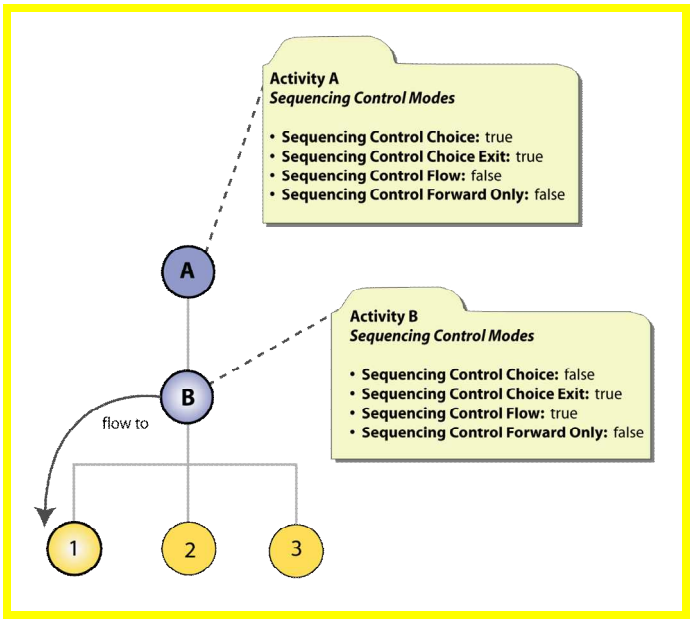

図 3.2.1b: 有効なフローとクラスタアクティビティの選択

2. 図 3.2.1c に示すように, *Choice* ナビゲーション要求の対象 (アクティビティ B) の *Sequencing Control Flow* が False である. この場合, 配信対象アクティビティがない (クラスタが配信されない). アクティビティ B の *Sequencing Control Choice* は True なので, 学習者がアクティビティ B ではなく, その子の一つを直接選択する (ナビゲーション要求を発行する) ような何らかのメカニズムを LMS が提供しなくてはならない.

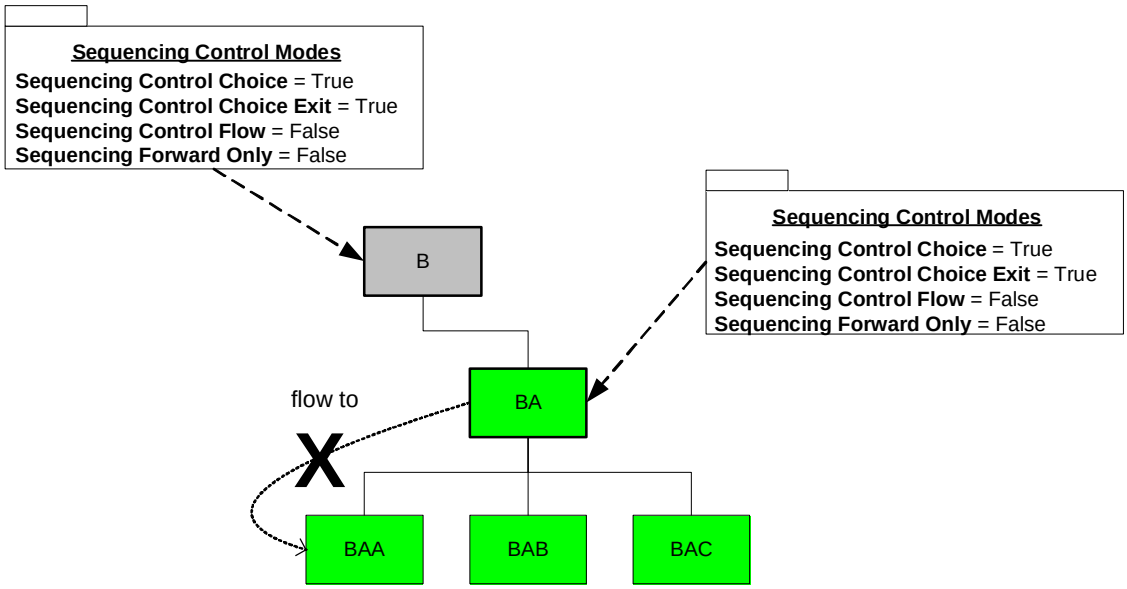

図 3.2.1c:無効なフローとクラスタアクティビティの選択

3.2.2. Sequencing Control Choice Exit

Sequencing Control Choice Exit 要素は，以後 *Choice Exit* と呼ぶが，設定されたアクティビティの子孫でないアクティビティが *Choice* ナビゲーション要求の対象となり，それにより設定されたアクティビティを終了させることができるかどうかを示す．*Choice Exit* はアクティブなアクティビティだけに適用される．この要素は，論理型 (True/False) の値を持つ．アクティビティに対し明示的に定義されていないならば，*Choice Exit* の既定値は True である．これは，アクティビティがアクティブな間に，子孫でないアクティビティに対する *Choice* ナビゲーション要求を学習者が発行できることを示している．

例えば，図 3.2.2a で，*Choice Exit* が False と定義されたアクティビティ 3 を現在学習者が学習しているものとする．アクティビティ 3 の親の *Sequencing Control Choice* は True だが，アクティビティ 3 の兄弟のいずれも *Choice* ナビゲーション要求の有効な対象ではない．アクティビティ 2 もしくはアクティビティ 4 を配信することは，アクティビティ 3 を終了することになり，*Choice Exit* コントロールの意図に反する．この例では，アクティビティ B の *Sequencing Control Flow* (セクション 3.2.3 参照) も True であり，学習者は学習行為を進行するのにアクティビティ 3 から Continue もしくは Previous ナビゲーション要求を発行できる．

LMS は，*Choice Exit* コントロールモード違反となるアクティビティを学習者が「選択する」メカニズムを提供しないことが推奨される．

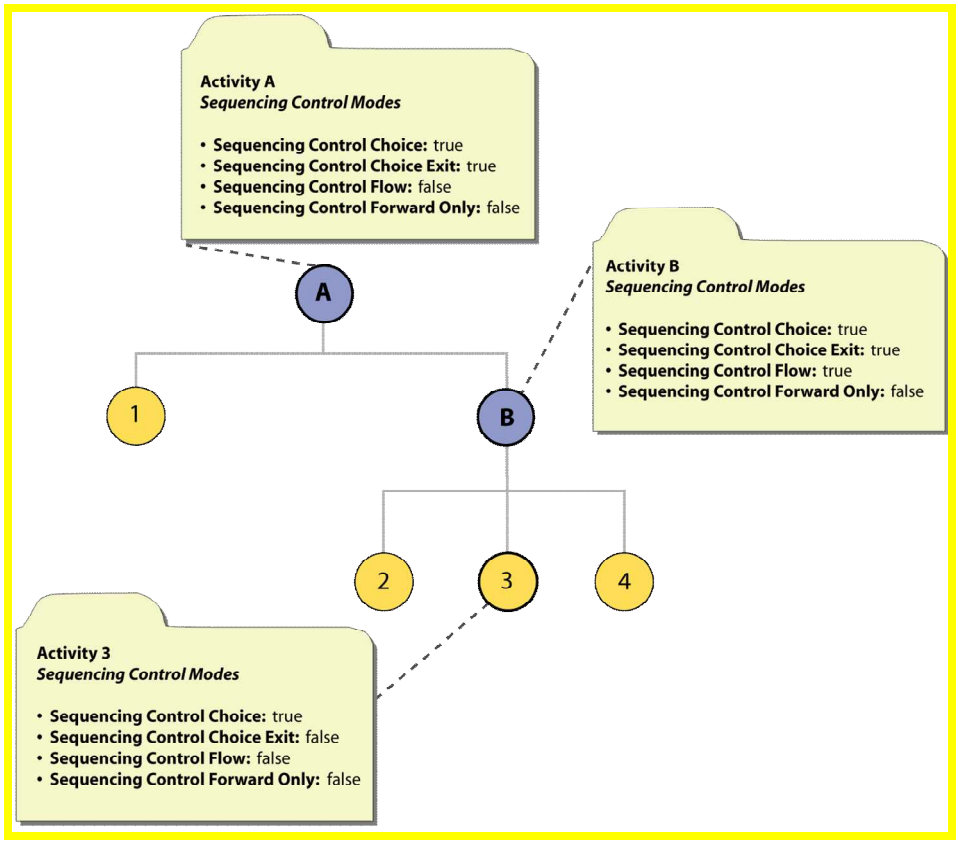

図 3.2.2a：Choice Exit の例

3.2.3. Sequencing Control Flow

Sequencing Control Flow 要素は，クラスタの子アクティビティに対するシステム主導のシーケンシングがサポートされている事を示す．この要素は論理型 (True/False) の値を持つ．アクティビティに対して明示的に定義されていないならば，*Sequencing Control Flow* の既定値は False である．これは，*Continue* および *Previous* ナビゲーション要求に基づいてアクティビティの子が学習される順序を，シーケンシング実装が自動的に評価しないということを示す．

もし，あるクラスタの *Sequencing Control Flow* コントロールモードが True と定義されたら，LMS は，学習者が次のアクティビティへ「継続する」もしくは「前の」アクティビティに戻る要望を示すための何らかのメカニズム提供しなくてはならない．

いくつかのケースでは，コンテンツ開発者は，コンテンツオブジェクト内から *Continue* および *Previous* ナビゲーション要求を始動したいと思うことがある．もし，あるクラスタの *Sequencing Control Flow* コントロールモードが True と定義され，コンテンツ自身が *Continue* もしくは *Previous* ナビゲーション要求を発行するメカニズムを提供するとコンテンツ開発者が定義した場合，*Continue* および *Previous* ナビゲーション要求を学習者に指定させる余分なメカニズムを LMS が提供しない事が望ましい．LMS がこのようなメカニズムを提供することは，2 組のナビゲーションコントロールが存在することになり，学習者を混乱させる恐れがある．

Sequencing Control Flow コントロールモードは，葉アクティビティに定義された場合無効である．

図 3.2.3a で，アクティビティ A の *Sequencing Control Flow* は True に設定されており，アクティビティ 1—3 はアクティビティ 1 から始まって Continue および Previous ナビゲーション要求に対応して，LMS のシーケンシング実装によって順序付けられる．

ADL ノート：この例で，もしアクティビティ A がアクティビティツリーのルートであれば，学習者がアクティビティ 1 を学習しているとき，*Previous* ナビゲーション要求は無効である．なぜならアクティビティ 1 はアクティビティツリーの最初の葉アクティビティだからである．一方，学習者がアクティビティ 3 を学習しているとき，*Continue* ナビゲーション要求は有効である．これにより，フローに基づく一貫した学習経験を維持するのに役立つ．しかし，アクティビティ 3 はアクティビティツリーの最後の葉アクティビティであり，学習者がアクティビティ 3 から *Continue* ナビゲーション要求を発行すると，LMS は *Exit All* を処理し，学習者のアクティビティツリーにおける現在の試行を終了させる．

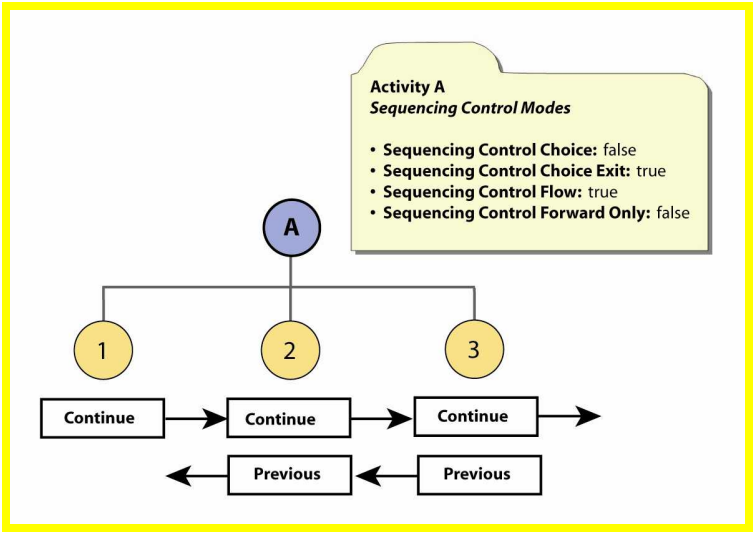

図 3.2.3a：Sequencing Control Flow 動作

3.2.4. Sequencing Control Forward Only

Sequencing Control Forward Only 要素は，以後 *Forward Only* と呼ぶが，クラスタの子アクティビティに対するシステム主導のシーケンシングにおいて，*Previous* ナビゲーション要求や後戻りする *Choice* 要求を禁止するよう制約されていることを示す．この要素は，論理型 (True/False) の値を持つ．アクティビティに対して明示的に定義されていないならば，*Forward Only* の既定値は False である．

学習者が現在学習しているクラスタの *Forward Only* が True の場合，LMS は，学習者が *Previous* ナビゲーション要求を発行するメカニズムを提供しない事が推奨される．

Forward Only コントロールモードは，葉アクティビティに定義された場合無効である．

図 3.2.4a では，アクティビティ A の *Forward Only* は Ture であり，学習者はアクティビティ 1 から始まり，アクティビティ 1—3 を連続した (前) 方向にだけ学習できる．この例では，いかなる *Previous* ナビゲーション要求も *Forward Only* コントロールモード違反により無効であるため許されない．

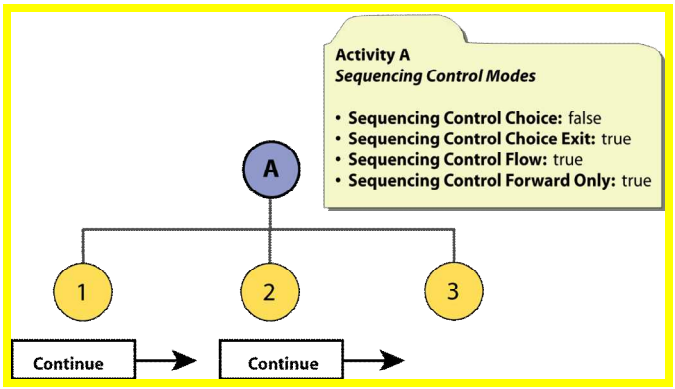

図 3.2.4a：Sequencing Control Forward Only の例

もし，あるアクティビティの *Forward Only* が True の場合，このノードの子の探索は常に前方向に行われる．例えば，*Previous* シーケンシング要求の結果としてクラスタに入った場合，最後の子アクティビティではなく最初の子アクティビティが最初に検索される．また，もし現在学習者が学習しているクラスタの *Forward Only* および *Sequencing Control Choice* の両方が True の場合，学習者は，現在学習しているアクティビティの前の兄弟であるアクティビティを *Choice* ナビゲーション要求の対象にできない．

3.2.5. Use Current Attempt Objective Information

Use Current Attempt Objective Information 要素は，アクティビティの子の学習目標進捗情報 (セクション 4.2：トラッキングモデル参照) が，様々なシーケンシング動作中にどのように管理され使用されるかを指定する．この動作は表 3.2.5a に集約されている．この要素は論理型の(True/False) 値を持つ．アクティビティに対して明示的に定義されていないならば，*Use Current Attempt Objective Information* の既定値は True である．

クラスタの *Use Current Attempt Objective Information* が False だと，クラスタに対する前回の施行で記録されたものだとしても，LMS はクラスタの子アクティビティの最も最近の試行の学習目標進捗情報を用いる．

もし値が True (デフォルト) の場合, クラスタに対する現在の施行で記録されたものでない, いずれの子アクティビティの学習目標進捗情報もデフォルト (未定) とならなくてはならない.

もし値が False の場合, クラスタに対する現在の施行で記録されたものでない, いずれの子アクティビティの学習目標進捗情報も前回の施行で記録されたものとならなくてはならない.

Use Current Attempt Objective Information 要素は, 葉アクティビティに定義された場合無効である.

表 3.2.5a: Use Current Objective Information に基づくトラッキング情報の評価

親の Use Current Attempt Objective Information	アクティビティの学習目標情報がいつ記録されたか	記録された学習目標情報と進捗度	学習目標情報がどのように評価されるか	
			アクティビティのプレ, ポスト, 終了シーケンシングルール	親のロールアップルール
True	クラスタの現在の試行の前	True	True	Unknown
		False	False	
		Unknown	Unknown	
	クラスタの現在の試行中	True	True	True
		False	False	False
		Unknown	Unknown	Unknown
False	関係なし	True	True	True
		False	False	False
		Unknown	Unknown	Unknown

3.2.6. Use Current Attempt Progress Information

Use Current Attempt Progress Information 要素は, アクティビティの子の試行進捗情報 (セクション 4.2: トラッキングモデル参照) が, 様々なシーケンシング動作中にどのように管理され使用されるかを指定する. この動作は表 3.2.6a に集約されている. この要素は論理型の(True/False) 値を持つ. アクティビティに対して明示的に定義されていないならば, *Use Current Attempt Progress Information* の既定値は True である.

クラスタの *Use Current Attempt Progress Information* が False だと, クラスタに対する前回の施行で記録されたものだとしても, LMS はクラスタの子アクティビティの最も最近の試行の試行進捗情報を用いる.

もし値が True (デフォルト) の場合, クラスタに対する現在の施行で記録されたものでない, いずれの子アクティビティの試行進捗情報もデフォルト (未定) とならなくてはならない.

もし値が False の場合, クラスタに対する現在の施行で記録されたものでない, いずれの子アクティビティの試行進捗情報も前回の施行で記録されたものとならなくてはならない.

Use Current Attempt Progress Information 要素は, 葉アクティビティに定義された場合無効である.

表 3.2.6a: Use Current Attempt Progress Information に基づくトラッキング情報の評価

親の Use Current Attempt Progress Information	アクティビティの試行情報がいつ記録されたか	記録された試行情報	試行情報がどのように評価されるか	
			アクティビティのプレ，ポスト，終了シーケンシングルール	親のロールアップルール
True	クラスタの現在の試行の前	True	True	Unknown
		False	False	
		Unknown	Unknown	
	クラスタの現在の試行中	True	True	True
		False	False	False
		Unknown	Unknown	Unknown
False	関係なし	True	True	True
		False	False	False
		Unknown	Unknown	Unknown

3.3. 選択制限コントロール

IMS SS 仕様は，アクティビティの親の *Sequencing Control Choice* が True の場合，アクティビティツリーのいかなる場所にあるいずれのアクティビティも，デフォルトで *Choice* ナビゲーション要求の有効な対象となることを認める．この柔軟性はいくつかのシーケンシング戦略では有効であるが，他の戦略では重大な問題となる．ADL は，*Choice* シーケンシング要求の処理について，追加の条件および動作を定めた選択制限 (Constrain Choice) コントロール (表 3.3a 参照) を定義した．

図 3.3a: 選択制限コントロールの説明

No.	名称	説明	値空間	規定値
1	*Constrain Choice*	*Choice* シーケンシング要求は，*Flow* において論理的に次のアクティビティだけを配信するように特定すべきであることを示す．	論理型	False
2	*Prevent Activation*	*Choice* シーケンシング要求は，アクティビティが既にアクティブであれば，アクティビティの子孫だけを配信するように特定すべきであることを示す．	論理型	False

3.3.1. Constrain Choice

アクティビティの *Constrain Choice* 要素が True の場合，アクティビティツリー中で制約されたそのアクティビティの論理的にひとつ「後ろ」もしくはひとつ「前」にあるアクティビティ，および，それらの子孫アクティビティだけが，*Choice* シーケンシング要求の対象となる．Choice シーケンシング要求サブプロセス (セクション 4.8.6.7 参照) のアクティビティツリー探索のどの時点でも，カレントアクティビティの祖先で *Constrain Choice* 要素が True のアクティビティに遭遇することがある．*Choice* ナビゲーション要求は有効なアクティビティであればどれを対象にしても良いが，True の *Constrain Choice* 要素に遭遇すると，対象アクティビティが配信対象となることを妨げる．この要素は，論理型の (True/False) 値を持つ．アクティビティに対して明示的に定義されていないならば，*Constrain Choice* の既定値は False である．

Constrain Choice 要素の目的は，アクティビティツリーにおいて有効な「選択」対象を，制約されたアクティビティ (*Constrain Choice* が True) の論理的に (いずれの方向にも) 次に位置するものに制約することである．これは学習者が，前提条件アクティビティを先に学習することなく，遠くへ「飛び」過ぎることを防ぐ．制約されたアクティビティの論理的に次のアクティビティは，制約されたアクティビティの前および後ろの方向に関して，Choise Flow サブプロセス (付録 C－SB.2.9.1 参照) によって決定される．Choise Flow サブプロセスで特定されたアクティビティ，および，その子孫のみが Choice の対象となり，他の シーケンシング情報の評価は保留される．例えば，図 3.3.1a では，意図されたシーケンシング戦略は，学習者がアクティビティを順番に試行することである： アクティビティ B，アクティビティ 4，アクティビティ 5，そして最後にアクティビティ C の順番にどのアクティビティも飛ばすことなくということである．アクティビティ B の *Constrain Choice* が True であり，アクティビティ B の子アクティビティからアクティビティ C の子アクティビティにジャンプすることはできない．学習者はアクティビティ B の試行を実行中に (アクティビティ B がアクティブなとき)，アクティビティ 1，2 および 3 だけが *Choice* シーケンシング要求の有効な対象となる．アクティビティ B を通り抜けるには，学習者はアクティビティ 3 からアクティビティ 4 へ continue (Flow) で移動しなければならない．

ADL ノート: *Constrain Choice* 要素を適用すると *Choice* ナビゲーション要求の有効な対象が減少する. 例えば, アクティビティ B の試行を実行中, アクティビティ C のどの子アクティビティも *Choice* ナビゲーション要求の有効な対象では無い. LMS は, ユーザーインターフェースナビゲーション装置を制御して, *Constrain Choice* 要素によって配信対象とならないアクティビティを (*Choice* ナビゲーション要求の発行の) 対象外にしなくてはならない.

Constrain Choice 要素は, 葉アクティビティに定義された場合無効である.

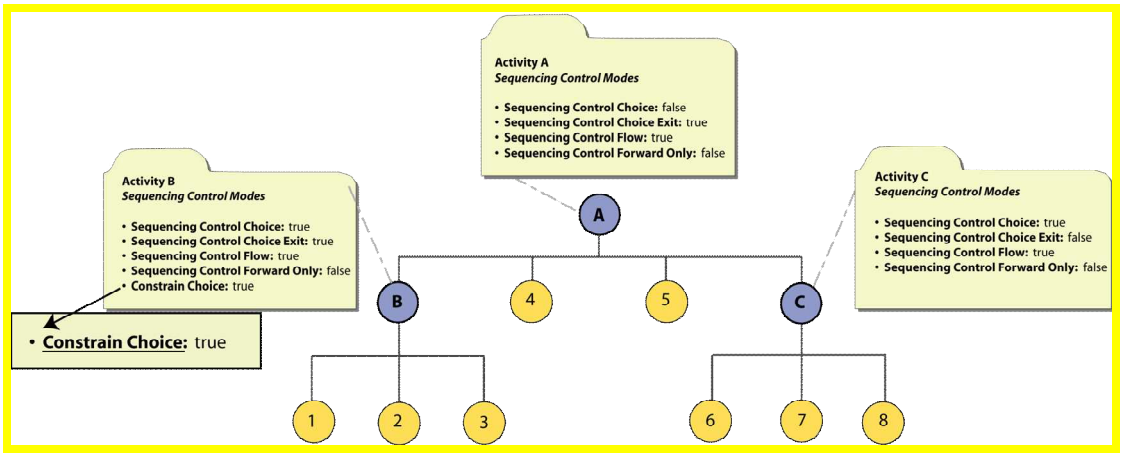

図 3.3.1a: Constrain Choice の例

3.3.2. Prevent Activation

アクティビティの *Prevent Activation* 要素が True の場合, アクティビティの子孫を対象とする *Choice* シーケンシング要求によってアクティビティへの試行を開始できないことを示す. すなわち, Prevent Activation 要素が True のアクティビティの子孫は, 禁止されたアクティビティに既に到達して (アクティビティがアクティブ, もしくは Current Activity である) いない限り, 配信用に指定されないということである。この要素は論理型の (True/False)値を持つ. アクティビティに対して明示的に定義されていないならば, *Prevent Activation* の既定値は False である。

Prevent Activation 要素の目的は, 有効な「選択」対象をアクティビティ直下の子に制約することである. これは学習者が, まず前提条件アクティビティへ到達することなく深くへ「飛び」過ぎることを防ぐ. 例えば, 図 3.3.2a では, 意図されたシーケンシング戦略は, 学習者がアクティビティ C の子アクティビティを選択する前に, まずアクティビティ C へ到達することである.

ADL ノート: *Prevent Activation* 要素を適用すると *Choice* ナビゲーション要求の有効な対象が減少する. 例えば, アクティビティツリーのどのアクティビティを試行中でも, アクティビティ C のどの子アクティビティも, まずアクティビティ C に到達した後でのみ *Choice* ナビゲーション要求の有効な対象となる. LMS は, ユーザーインターフェースナビゲーション装置を制御して, *Prevent Activation* 要素によって配信対象とならないアクティビティを (*Choice* ナビゲーション要求の発行の) 対象外にしなくてはならない.

Prevent Activation 要素は, 葉アクティビティに定義された場合無効である.

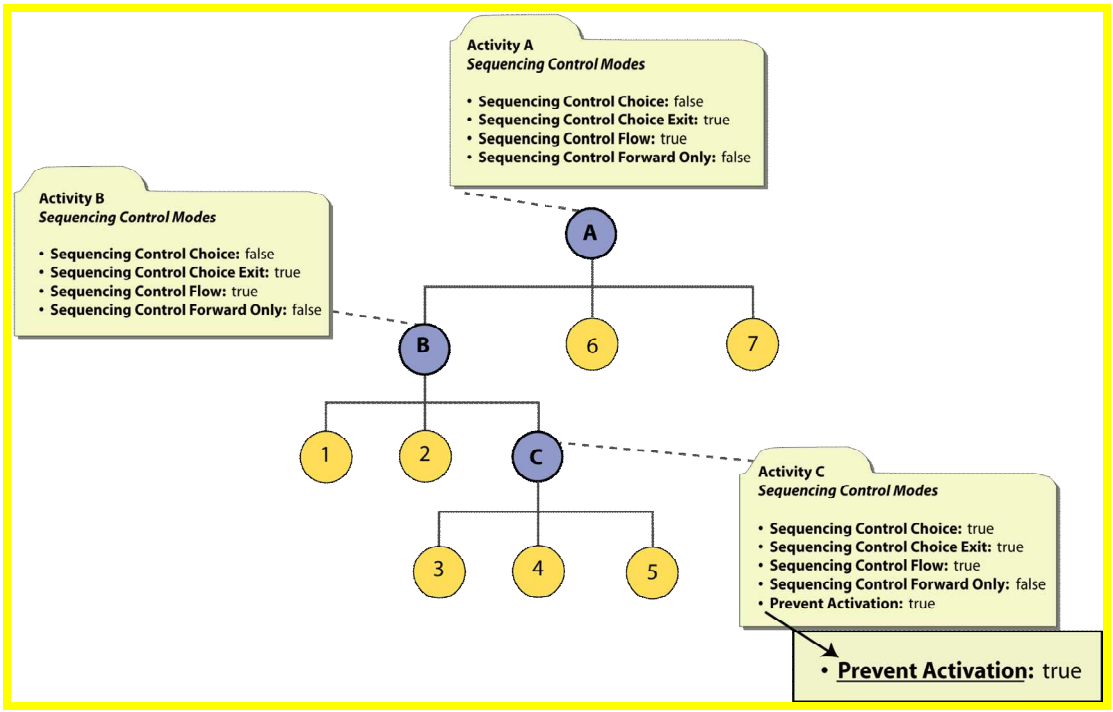

図 3.3.2a: Prevent Activation の例

3.4. シーケンシングルール記述

IMS SS 仕様は，ルールベースのシーケンシングモデルを用いている．ゼロもしくはそれ以上のシーケンシングルールの組がアクティビティに適用され，ルールは様々なシーケンシング動作（付録 C：シーケンシング動作擬似コード参照）の中の決められた時点で評価される．各シーケンシングルールは，一組のコンディション（Condition）およびそれに対応するアクション（Action）から成り立つ．コンディションはアクティビティに関連付けられたトラッキング情報（セクション 4.2：トラッキングモデル参照）を使って評価される．ルールのアクションに対応する動作は，ルールの condition-set が True であると評価された場合に実行される．図 3.4a は，シーケンシングルールの(if [condition_set] then [action])構造を示している．

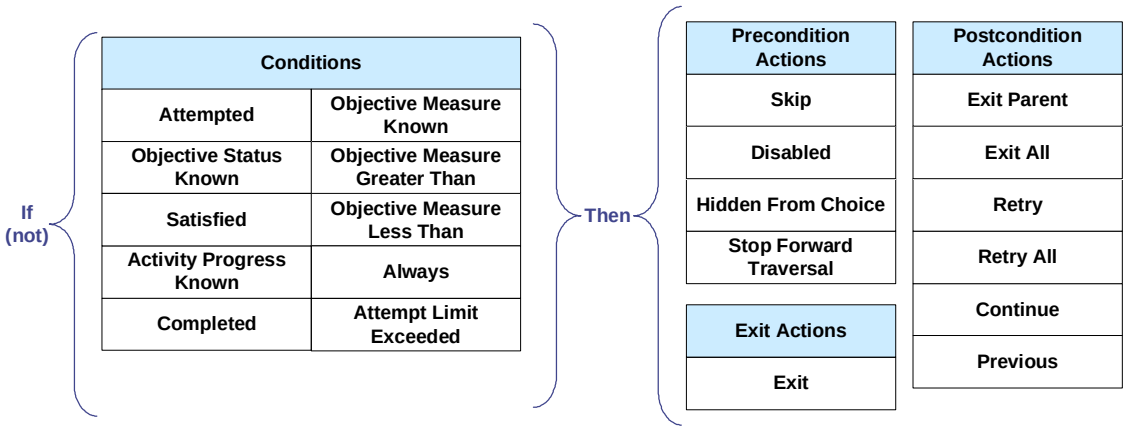

図 3.4a: シーケンシングルールコンディションおよびアクション

3.4.1. Condition Combination

個々のコンディションが組み合わされて，評価の対象となるコンディション集合になるが，結果としてアクションが発行されるには，どれか一つのコンディションが True でなければいけない場合と，全てが True でなければいけない場合がある．*Condition Combination* 要素は表 3.4.1a で定義される．

- **All**（既定値）－個々のコンディション全てが True の場合にのみ，コンディションの集合は True と評価される．論理的には ***And*** の役目をする．
- **Any**－個々のコンディションのどれか一つが True の場合，コンディションの集合は True と評価される．論理的には ***Or*** の役目をする．

表 3.4.1a: Condition Combination

No.	名称	説明	値空間	既定値
1	*Condition Combination*	ルールを評価するためにどのようにルールコンディションが組み合わされるか • *All* － 個々のルールコンディションが全て True の場合にのみ，全体のルールコンディションが True と評価される．（論理的 And）	語彙	All

		• *Any* – 個々のルールコンディションのどれか一つが True であれば, 全体のルールコンディションが True と評価される. (論理的 Or)		

3.4.2. Rule Conditions

Rule Conditions 要素はコンディションの組からなり, これはシーケンシングルールが定義されたアクティビティにおいて評価される. *Rule Conditions* 要素は, 一つもしくはそれ以上の個別 *Rule Condition* 要素から成り立ち, その組み合わせはシーケンシングルールに適用された *Condition Combination* (セクション 3.4.1 参照) によって定義される. 各 *Rule Condition* 要素は, トラッキングモデル (セクション 4.2 参照) の要素に基づいた特定の語彙 (表 3.4.2a 参照) の中の一つでなければならない.

ADL ノート: SCORM は, LMS が時間によるトラッキング情報を管理もしくは保持することを要求していない. 従って, 時間によるコンディションの評価は定義される必要がない. シーケンシング実装は, シーケンシングルールを評価するとき, 全ての, もしくは, いくつかの時間によるコンディションを考慮しなくても構わない. もしシーケンシングルールが時間によるコンディションだけを使用した場合, シーケンシング実装はシーケンシングルール全てを無視しても構わない事になる. コンテンツ開発者は, 時間によるコンディションをシーケンシングルールに適用しても, LMS が認識しない可能性があることに注意する必要がある.

表 3.4.2a: Rule Conditions

コンディション	説明
Satisfied	アクティビティに付随する学習目標の *Objective Progress Status* が True かつアクティビティに付随する学習目標の *Objective Satisfied Status* が True だった場合, True に評価される.
Objective Status Known	アクティビティに付随する学習目標の *Objective Progress Status* が True だった場合, True に評価される.
Objective Measure Known	アクティビティに付随する学習目標の *Objective Progress Status* が True かつアクティビティに付随する学習目標の *Objective Measure Status* が True の場合, True に評価される.
Objective Measure Greater Than	アクティビティに付随する学習目標の *Objective Measure Status* が True かつアクティビティに付随する学習目標の *Objective Normalized Measure* が *Rule Condition Measure Threshold* より大きい場合, True に評価される.
Objective Measure Less Than	アクティビティに付随する学習目標の *Objective Measure Status* が True かつアクティビティに付随する学習目標の *Objective Normalized Measure* が *Rule Condition Measure Threshold* より小さい場合, True に評価される.
Completed	アクティビティの *Attempt Progress Status* が True かつアクティビティの *Attempt Completion Status* が True の場合, True に評価される.
Activity Progress Known	アクティビティの *Activity Progress Status* が True かつアクティビティの *Attempt Progress Status* が True の場合, True に評価される.
Attempted	アクティビティの *Activity Progress Status* が True かつアクティビティの *Activity Attempt Count* が potitive の場合 (アクティビティが試行された場合), True に評価される.
Attempt Limit Exceeded	アクティビティの *Activity Progress Status* が True かつアクティビティの Limit Condition Attempt Limit Control が True かつアクティビティ

	の *Activity Attempt Count* がアクティビティの *Limit Condition Attempt Limit* より大きいか等しい場合, True に評価される.
Always	常に True に評価される

3.4.3. Rule Condition Referenced Objective

Rule Condition Referenced Objective 要素 (表 3.4.3a 参照) は特定の *Rule Condition* で用いる. アクティビティに対して定義された一組の学習目標のうち, どの学習目標を *Rule Condition* の評価に使うべきかを指定するのに使用される. *Rule Condition Referenced Objective* 要素は, 学習目標進捗情報 (セクション 4.2:トラッキングモデル参照) に適用する以下のコンディションでのみ使用される:

- Satisfied
- Objective Status Known
- Objective Measure Known
- Objective Measure Greater Than
- Objective Measure Less Than

上記の *Rule Conditions* が, シーケンシングルールにおいて明示的に学習目標を参照していない場合, アクティビティの *Objective Description* (セクション 3.10 を参照) が True の学習目標がデフォルトで使用される.

ADL ノート: *Rule Condition Referenced Objective* 要素は, 上記に挙げた以外の *Rule Conditions* に定義た場合, 無効である.

表 3.4.3a: Rule Condition Referenced Objective

No.	名称	説明	値空間	既定値
2.2	*Rule Condition Referenced Objective*	アクティビティに付随する, 条件の評価において使用される学習目標の識別子. アクティビティのルールが識別子によって陽に学習目標を参照しない場合, ルールはアクティビティのロールアップに寄与する学習目標をデフォルトで参照する.	固有識別子	None

3.4.4. Rule Condition Measure Threshold

Rule Condition Measure Threshold 要素 (表 3.4.4a 参照) は, 特定の *Rule Condition* で用いる. *Rule Condition Referenced Objective* 要素と組み合わせて使用され, *Rule Condition* の評価時に習得度の比較に使用される閾値を定義する. この要素は, 以下のコンディションでのみ使用される:

- **Objective Measure Greater Than**: [objective measure] > [measure threshold]
- **Objective Measure Less Than**: [objective measure] < [measure threshold]

コンテンツ開発者は, *Rule Condition Measure Threshold* 要素で実行される比較は, より大きい(>)およびより小さい (<)であるということに留意する必要がある. 「等しいかより大きい」 (>=) また「等しいかより小さい」(<=) という比較演算子に対して明示的に定義されたルールコンディションはないが, これらは適切なコンディションを否定 (「Not」演算子を適用 (セクション 3.4.5:*Rule Condition Operator* 参照) することにより実行可能である.

ADL ノート: *Rule Condition Measure Threshold* 要素は, Objective Measure Greater Than および Objective Measure Less Than 以外の *Rule Conditions* に定義しても作用しない.

表 3.4.4a: Rule Condition Measure Threshold

No.	名称	説明	値空間	既定値
2.3	*Rule Condition Measure Threshold*	習得度に基づくコンディション評価時に閾値として使用される値	実数型[-1.0..1.0] 有効数字 4 桁の実数	0

3.4.5. Rule Condition Operator

Rule Condition Operator 要素は, 各 *Rule Condition* 要素に適用される任意要素である. *Rule Condition* の評価後に適用される単項式の論理演算を示す. 表 3.4.5a は, IMS SS がサポートする二つの単項論理演算を記述する.

- **NO-OP** (既定値) – *Rule Condition* 評価結果はそのまま使用される
- **Not** – *Rule Condition* 評価結果は使用される前に否定される

表 3.4.5a: Rule Condition 演算子

No.	名称	説明	値空間	既定値
2.4	*Rule Condition Operator*	評価に適用される単項論理演算子 • *Not* – コンディション評価結果の否定がルール評価に使用される • *NO-OP* – コンディション評価結果がそのままルール評価に使用される	語彙	NO-OP

3.4.6. Rule Action

Rule Action 要素(表 3.4.6a, 3.4.6b および 3.4.6c)は, 様々なシーケンシング動作中, シーケンシングルールのコンディションの集合が True の時に LMS が実行する責任を持つ, 意図されたアクションおよび動作を表す. アクションは, 3 つの評価タイミング状況により分類される:

- **Precondition Actions**: アクティビティツリーを移動して配信対象アクティビティを特定する際に用いる
- **Post condition Actions**: アクティビティの試行が終了したときに用いる
- **Exit Actions**: 子孫のアクティビティの試行が終了した後に用いる

表 3.4.6a: Precondition Rule Actions

No.	名称	説明	値空間	既定値
Precondition Actions				
3	*Rule Action*	ルールが True と評価されたときにとられる望ましいシーケンシング動作 • *Skip* – そのアクティビティは *Flow* シーケンシング要求中に配信対象の候補とはみなされない • *Disabled* – アクティビティはシーケンシング要求または配信要求の対象にならない • *Hidden from Choice* – アクティビティは *Choice* シーケンシング要求の対象にならない • *Stop Forward Traversal* – シーケンシング要求中の アクティビティから「前向き」方向のアクティビティが配信候補になることを妨げる	語彙	Ignore

表 3.4.6b: Postcondition Rule Actions

Postcondition Actions				
3	*Rule Action*	ルールが True と評価されたときにとられる望ましいシーケンシング動作 • *Exit Parent* – *Exit Parent* 終了要求を実行する • *Exit All* – *Exit All* 終了要求を実行し *Exit* シーケンシング要求を返す • *Retry* – *Retry* シーケンシング要求を返す • *Retry All* – *Exit All* 終了要求を実行し *Start* シーケンシング要求を返す • *Continue* – *Continue* シーケンシング要求を返す • *Previous* – *Previous* シーケンシング要求を返す	語彙	Ignore

表 3.4.6c: Exit Rule Actions

Exit Actions				
3	*Rule Action*	ルールが True と評価されたときにとられる望ましいシーケンシング動作 *Exit* – 無条件でアクティビティを停止する	語彙	Ignore

ADL ノート: SCORM は, フローベースのシーケンシング要求(*Continue*, *Previous*, *Start*, および *Retry*)を行うときは, *Stop Forward Traversal Rule* アクションを使用しない. このアクションは, *Choice* シーケンシング要求により, *Current Activity* から前方のアクティビティが対象となった場合にのみ適用される. さらに, シーケンシング動作は, *Stop Forward Traversal* ルールアクション (付録 C 参照) の評価を削除するように更新されている.

3.5. 制限コンディション

コンテンツ開発者は，制限コンディション（Limit Condition）を定義して，アクティビティの配信を不許可とするコンディションを表すことができる．制限コンディションは，アクティビティに関連付けられ，アクティビティのトラッキング状態情報（セクション 4.2：トラッキングモデル参照）に依存する．制限コンディションが満たされた，もしくは超えたとき，アクティビティは配信できなくなる．

SCORM は，*Limit Condition Attempt Limit* 要素へのサポートだけを要求する．SCORM は，いかなる時間ベースの制限コンディションの評価も要求しない．従って，LMS は，制限コンディションチェックプロセス（付録 C: UP.1 参照）の任意部分に関しては，データ管理を要求されない．

3.5.1. 試行制限

コンテンツ開発者が，学習者に許される学習アクティビティ試行回数を制限したいと思うケースがある．*Limit Condition Attempt Limit* 要素は，非負整数値を含み，この値でアクティビティの最大試行回数を指定する．コンテンツ開発者が *Limit Condition Attempt Limit* 値を定義しないと，アクティビティの試行回数に制約がないことになる．表 3.5.1a に，*Limit Condition Attempt Limit* 要素を記述する．

表 3.5.1a: Attempt Limit

No.	名称	説明	値空間	既定値
1	*Limit Condition Attempt Control*	アクティビティに関する試行回数に関する制限コンディションが設定されていることを示す 値が False なら，アクティビティに関する試行回数制限はない	論理型	False
2	*Limit Condition Attempt Limit*	アクティビティへの最大試行回数．ゼロ値の場合，アクティビティにはアクセスできないことを意味する． *Limit Condition Attempt Control* 値が True 以外の場合，この値は意味を持たない．	非負整数型	0

ADL ノート：*Limit Condition Attempt Limit* 要素の説明は，データモデルペアを使用している．つまり，一つの要素が意図された制限を表し，もう一つがその制限が有効かどうかを表す．例えば，*Limit Condition Attempt Limit* がアクティビティへの試行の制限が何かを表し，*Limit Condition Attempt Control* が *Limit Condition Attempt Limit* の値は有効かどうかを表す．これらの二つの要素を初期化し，同期するよう維持するのは LMS の役割である．

3.5.2. Attempt Absolute Duration

コンテンツ開発者が学習アクティビティの一回の試行に費やされる時間を制限したいシナリオがある場合がある．*Attempt Absolute Duration Limit* 要素は，学習者が一回の試行で許される最大の時間を定義する値を含む．この時間は，その間のシステムまたは学習者アクションに関係なく，LMS がアクティビティに

試行を開始してから終了するまでの時間である. コンテンツ開発者が学習アクティビティの *Attempt Absolute Duration Limit* を定義しない場合, 学習者がそのアクティビティをどれだけ長く使えるかに制約はない.

ADL ノート: SCORM は, 時間による制限コンディションの評価を要求しない. *Attempt Absolute Duration Limit* 要素は, アクティビティに関連付けられた SCO の cmi.max_time_allowed ランタイムデータモデル要素[4] (セクション 4.2.15: *Maximum Time Allowed*) の初期化を行なうためだけに含まれている. LMS は, 制限コンディションチェックプロセス (付録 C: UP.1 参照) の間, この要素の評価を使用することを要求していない.

表 3.5.2a: Attempt Absolute Duration Limit

No.	名称	説明	値空間	既定値
1	*Limit Condition Attempt Absolute Duration Control*	学習者が一度のアクティビティ試行に費やすのを許される最大時間の制限コンディションが設定されていることを示す 値が False なら, 学習者がアクティビティで費やす時間について制限がない.	論理型	False
2	*Limit Condition Attempt Absolute Duration Limit*	学習者が一度のアクティビティ試行に費やすのを許される最大時間. アクティビティが *active* なときこの制限が適用される—アクティビティが始まってから終わるまでアクティビティが *suspend* している時間も含む. 値がゼロなら, アクティビティにはアクセスできないことを意味する. *Attempt Absolute Duration Control* が True 以外の場合, この値は意味を持たない.	期間 – 精度 0.1 秒	0.0

ADL ノート: *Limit Condition Attempt Absolute Duration Limit* 要素の記述は, データモデルペアを使用している. つまり, 一つの要素が意図された制限を表し, もう一つがその制限が有効かどうかを表す. 例えば, *Limit Condition Attempt Absolute Duration Limit* がアクティビティへの試行の制限が何かを表し, *Limit Condition Attempt Absolute Duration Limit Control* が *Limit Condition Attempt Absolute Duration Limit* は有効かどうかを表している. これらの二つの要素を初期化し, 同期するよう維持するのは LMS の役割である.

3.6. 補助学習資源

アクティビティは学習者に追加サービスまたは学習資源を提供する補助学習資源を持つことがある. IMS SS 仕様は, これらの補助学習資源にどのような意味も定義していない. IMS SS 仕様は, どの学習資源が使用可能か, もしくはどのように学習資源が使用されるかを定義していない. 唯一 IMS SS 仕様が提供するのは, 補助学習資源をアクティビティに関連付ける方法だけである.

SCORM は, LMS に補助学習資源をサポートするようには要求してしない. もし LMS が補助学習資源を実行もしくは提供することを選択した場合, 相互運用性に保証はない.

3.7. ロールアップルール

クラスタアクティビティはコンテンツオブジェクトとは関連しないため, 学習者進捗情報がクラスタアクティビティに直接適用されることはない. IMS SS 仕様は, クラスタアクティビティに対する学習者の進捗状況をどのように評価するか定義する方法を提供している. ゼロもしくはそれ以上のロールアップルールの組をクラスタアクティビティに適用することができ, それらはオーバーオールロールアッププロセス(セクション 4.6: ロールアップ動作参照)中で評価される. 各ロールアップルールは, 一組の評価対象の子アクティビティ, 子アクティビティのトラッキング情報に対して評価されるコンディションの集合, および, コンディションの集合の評価が True の際にクラスタのトラッキング状態情報を設定するアクションから成り立つ. 図 3.7a は, ロールアップルールの構造(if [condition_set] True for [child activity set] then [action])を示す.

ロールアップルールは葉アクティビティに定義された場合無効である.

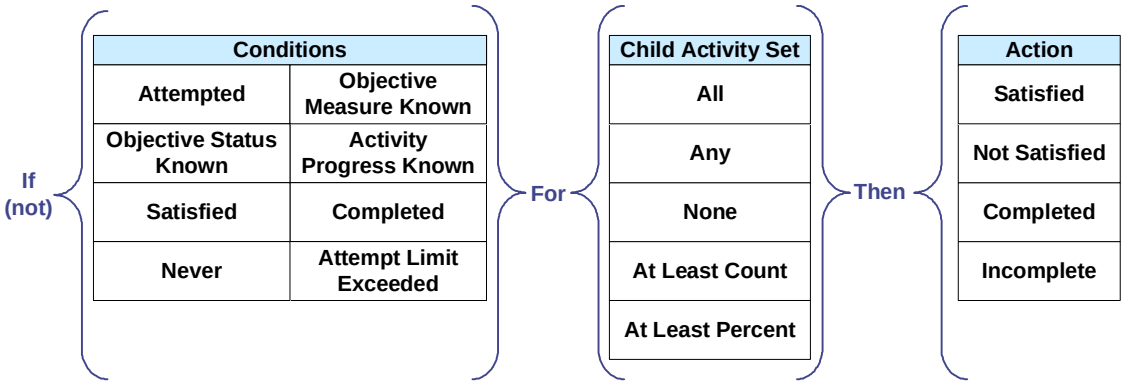

図 3.7a: ロールアップルール 子アクティビティ集合, コンディションおよびアクション

3.7.1. Condition Combination

ロールアップに含まれる各アクティビティに対して, 個々のコンディションを組み合わせ, コンディションのどれか一つが True, もしくは, 全てのコンディションが True にならなければならないコンディションの集合を作る. ロールアップに含まれる各アクティビティの *Condition Combination* は, ロールアップルールに定義された *Child Activity Set* に対して評価され, 結果として起こるアクションを発行すべきか否かを決定する. *Condition Combination* 要素は表 3.7.1a で定義される.

- **All** –個々のロールアップコンディション全てが Ture と評価される場合のみ, コンディションの集合が True になる. 論理的に ***And*** の役目をする. .
- **Any** (既定値) – 個々のロールアップコンディションのうちどれか一つでも True と評価されれば, コンディションの集合は True になる. 論理的に ***Or*** の役目をする.

表 3.7.1a: Condition Combination

No.	名称	説明	値空間	既定値
1	*Condition Combination*	ロールアップコンディションが評価ルールにおいてどのように結合されるか. • *All* – 個々のロールアップコンディションの	語彙	Any

		全てが True と評価される場合に限り，ルールコンディションは True と評価される．(論理 And) • *Any* – 個々のロールアップコンディションのどれか一つでも True と評価されれば，ルールコンディションは True と評価される．(論理 Or)		

3.7.2. Rollup Conditions

Rollup Conditions 要素はコンディションの集合を含み，これらのコンディションはロールアップルール評価に含まれる各アクティビティにおいて評価される．*Rollup Conditions* 要素は，一つもしくはそれ以上の *Rollup Conditions* 要素で成り立ち，これらの要素はロールアップルールに適用される *Condition Combination* (セクション 3.7.1 参照)の定義の通りに組み合わされる．各 *Rollup Condition* 要素には，トラッキングモデル (セクション 4.2 参照) 要素の語彙 (表 3.7.2a 参照) を使用しなければならい．

ADL ノート：SCORM は，LMS に対して，時間に関するトラッキング情報を管理もしくは保持することを要求していない．従って，時間に関するコンディションの評価は定義しない．シーケンシングの実装に関して，ロールアップルールを評価する際，全てのもしくはいくつかの時間に関するコンディションを考慮しなくても構わない．もしロールアップルールが時間に関するコンディションだけを使用した場合，シーケンシングの実装では全てのロールアップルールを無視し，代わりに，デフォルトロールアップルールを使っても構わない事になる (セクション 4.6：ロールアップ動作参照)．コンテンツ開発者は，時間に関するコンディションをロールアップルールに適用しても LMS で実行できない可能性があることに注意する必要がある．

表 3.7.2a: Rollup Conditions

コンディション	説明
Satisfied	子アクティビティに付随するロールアップ学習目標の *Objective Progress Status* が True かつ子アクティビティに付随するロールアップ学習目標の *Objective Satisfied Status* が True のとき，True と評価される
Objective Status Known	子アクティビティに付随するロールアップ学習目標の *Objective Progress Status* が True のとき，True と評価される
Objective Measure Known	子アクティビティに付随するロールアップ学習目標の *Objective Measure Status* が True のとき、True と評価される
Completed	子アクティビティの *Attempt Progress Status* が True かつ子アクティビティの *Attempt Completion Status* が True のとき，True と評価される
Activity Progress Known	子アクティビティの *Activity Progress Status* が True かつ子アクティビティの *Attempt Progress Status* が True のとき，True と評価される
Attempted	子アクティビティの *Activity Progress Status* が True かつ子アクティビティの *Activity Attempt Count* が正のとき (例：アクティビティが試行されたとき)，True と評価される
Attempt Limit Exceeded	子アクティビティの *Activity Progress Status* が True かつ子アクティビティの *Limit Condition Attempt Limit Control* が True かつ子アクティビティの *Activity Attempt Count* が子アクティビティの *Limit Condition Attempt Limit* 以上のとき，True と評価される

Never	常に False と評価される

3.7.3. Rollup Condition Operator

Rollup Condition Operator 要素は，各 *Rollup Condition* 要素に適用される任意要素で，*Rollup Condition* の評価後に適用される単項論理演算を示す．表 3.7.3a は，IMS SS でサポートされる二つの単項論理演算を示す．

- **NO-OP** (既定値) – *Rollup Condition* 評価の結果をそのまま使用する．
- **Not** – *Rollup Condition* 評価の結果を否定する．

表 3.7.3a: Rollup Condition 演算子の説明

No.	名称	説明	値空間	規定値
3.2	*Rollup Condition Operator*	コンディションの評価に適用される単項論理演算子 • *Not* – コンディションがルール評価において否定される • *NO-OP* – コンディションがルール評価においてそのまま使用される	語彙	NO-OP

3.7.4. Rollup Child Activity Set

デフォルトでは，クラスタのロールアップ評価では，クラスタの全ての子のトラッキング状態情報が評価される．コンテンツ開発者は，アクティビティをロールアップ評価にどのようにいつ対象とするか，明示的に制限することが可能である．

- *Tracked* (セクション 3.13.1 参照)を False に定義－アクティビティがどんなトラッキング情報も保持しないことを示す．従って，ロールアップにこのアクティビティが対象となることは決してない．
- *Rollup Objective Satisfied* (セクション 3.8.1 参照)を False に定義－*Rollup Action* が Satisfied もしくは Not Satisfied のロールアップルールの評価に，このアクティビティが対象とならないことを示す．
- *Rollup Objective Measure Weight* (セクション 3.8.2 参照)を 0.0 に定義 – アクティビティの習得度が親の加重平均習得度に寄与しないことを示す．
- *Rollup Progress Completion* (セクション 3.8.3 参照) を False に定義 –*Rollup Action* が Completed もしくは Incomplete のロールアップルールの評価に，このアクティビティが対象とならないことを示す．
- *Measure Satisfaction If Active* (セクション 3.9.1 参照)を定義 – この要素は，アクティビティのロールアップ学習目標の習得度がロールアップ学習目標の習得にいつ適用されるかを示す．
- さまざまな *Required For Rollup Elements* (セクション 3.9.2 参照)を定義 – この要素は，特定の *Rollup Actions* のロールアップルールの評価にアクティビティがいつ対象となるかを条件付で示す．

ロールアップルール評価の際に，*Rollup Conditions* は (上記の基準に基づいた) 全てのロールアップ対象アクティビティに適用される．*Rollup Action* が適用されるどうかを決定する際，ロールアップ対象アクティビティのコンディションの評価の結果がどのように使用されるのかが，*Rollup Child Activity Set* 要素(表

3.7.4a 参照)によって定義される. *Child Activity Set* は, *Rollup Action* がいつ適用されるべきか示す決められた語彙から成り立つ:

- **All** (既定値) –対象アクティビティの *Condition Combination* の全てが True のとき, 指定された *Rollup Action* を適用する.
- **Any** –対象アクティビティの *Condition Combination* のいずれかが True のとき, 指定された *Rollup Action* を適用する.
- **None** –対象アクティビティの *Condition Combination* のいずれもが True でなければ, 指定された *Rollup Action* を適用する.
- **At Least Count** – *Rollup Minimum Count* 要素で指定した最低数の対象アクティビティの *Condition Combination* が True のとき, 指定された *Rollup Action* を適用する.
- **At Least Percent** – *Rollup Minimum Percent* 要素で指定した最低限の割合の対象アクティビティの *Condition Combination* が True のとき, 指定された *Rollup Action* を適用する.

表 3.7.4a: Rollup Child Activity Set

No.	名称	説明	値空間	規定値
1	*Rollup Child Activity Set*	ロールアップコンディションを評価する際に, データの値が使われる子アクティビティのセット • *All* - 子のロールアップコンディション (*Condition Combination*の結果)がすべてTrueのとき, ロールアップルールがTrueになる • *Any* - 子のロールアップコンディション (*Condition Combination*の結果)のどれかがTrueのとき, ロールアップルールがTrueになる • *None* – 子のロールアップコンディション (*Condition Combination*の結果)のどれもがTrueにならないとき, ロールアップルールがTrueになる • *At Least Count* - *Rollup Minimum Count*で指定した最低限の数の子のロールアップコンディション(*Condition Combination*の結果)が Trueのとき, ロールアップルールがTrueになる • *At Least Percent* - *Rollup Minimum Percent*で指定した最低限の割合の子のロールアップコンディション(*Condition Combination*の結果)がTrueのとき, ロールアップルールがTrueになる	語彙	All

Rollup Child Activity Set の記述において **At Least Count** 語彙を使用する際, *Rollup Minimum Count* 要素の値が使用される. *Rollup Minimum Count* 要素は, *Rollup Conditions* の *Condition Combination* が True でなくてはならないアクティビティの最低数を示す整数値であり, これは定足数のように機能する. *Rollup Minimum Count* 要素の既定値はゼロである. 値が定義されないと, ロールアップ評価時にアクティビティは要求されず, *Rollup Action* が無条件で適用されることになる.

Rollup Child Activity Set の記述において **At Least Percent** 語彙を使用する際, *Rollup Minimum Percent* 要素の値が使用される. *Rollup Minimum Percent* 要素は, *Rollup Conditions* の *Condition*

Combination が True でなくてはならないアクティビティの数の最低の割合を示す実数値である. *Rollup Minimum Percent* 要素の既定値は 0.0 である. 値が定義されないと, ロールアップ評価時にアクティビティは要求されず, *Rollup Action* が無条件で適用されることになる.

3.7.5. Rollup Actions

Rollup Action 要素は, ロールアップルールが定義されたクラスタアクティビティに適用される望ましいアクションを表す. コンディションの集合が, ロールアップルールの *Rollup Child Activity Set* に定義されたようにロールアップ評価に含まれるアクティビティに適用される場合, *Rollup Action* はロールアップ動作 (セクション 4.6 参照) 時に適用される. 表 3.7.5a で定義されるように, *Rollup Action* はロールアップルールが対応するアクティビティのトラッキング状態モデル (セクション 4.2:トラッキングモデル参照) に影響をおよぼすことがある.

表 3.7.5a: Rollup Actions

ロールアップアクション	説明
Satisfied (*default value*)	下記のとおり設定する: ▪ アクティビティに付随するロールアップ学習目標の *Objective Progress Status* を True に ▪ アクティビティに付随するロールアップ学習目標の *Objective Satisfied Status* を True に
Not Satisfied	下記のとおり設定する: ▪ アクティビティに付随するロールアップ学習目標の *Objective Progress Status* を True に ▪ アクティビティに付随するロールアップ学習目標の *Objective Satisfied Status* を False に
Completed	下記のとおり設定する: ▪ アクティビティの *Attempt Progress Status* を True に ▪ アクティビティの *Attempt Completion Status* を True に
Incomplete	下記のとおり設定する: ▪ アクティビティの *Attempt Progress Status* を True に ▪ アクティビティの *Attempt Completion Status* を False に

3.8. Rollup Controls

IMS SS 仕様では, コンテンツ開発者が広いレベルで, アクティビティの親のロールアップへの寄与を条件付きで制限することができる. 表 3.8a は, ロールアップ時に制限できる3つのトラッキング状態情報 (セクション 4.2:トラッキングモデル) の詳細を示す.

表 3.8a: Rollup Controls

No.	名称	説明	値空間	既定値
1	*Rollup Objective Satisfied*	親の *Satisfied* と *Not Satisfied* ロールアップルールの評価にアクティビティが含まれるかどうかを示す	論理型	True
2	*Rollup Objective Measure Weight*	親アクティビティへのロールアップの際に, 子アクティビティに付随する(*Objective Contributes to Rollup* が *True* の)学習目標の *Objective Normalized Measure* に適用される重み	実数型[0..1] 有効数字 4 桁の実数	1.0
3	*Rollup Progress Completion*	親の *Completed* と *Not Incomplete* ロールアップルールの評価にアクティビティが含まれるかどうか示す	論理型	True

3.8.1. Rollup Objective Satisfied

Rollup Objective Satisfied 要素は, アクティビティのトラッキング状態情報 (セクション 4.2:トラッキングモデル) が, 親の *Rollup Actions* が Satisfied および Not Satisfied のロールアップルールに適用されるかどうかを示す. この要素は論理型(True/False)の値を持つ. アクティビティに対して明示的に定義されていないならば, *Rollup Objective Satisfied* の既定値は True である.

アクティビティの *Rollup Objective Satisfied* が False の場合, アクティビティのトラッキング情報が記録されても, *Rollup Action* が Satisfied もしくは Not Satisfied の親のロールアップルールすべてに対して, LMS はアクティビティのトラッキング状態情報を考慮しない.

3.8.2. Rollup Objective Measure Weight

Rollup Objective Measure Weight 要素は, アクティビティの *Objective Normalized Measure* (セクション 4.2:トラッキングモデル参照) が, 親の *Objective Normalized Measure* 評価時にどのように使用されるかを示す. この要素は実数値([0.0..1.0])を持つ. アクティビティに対して明示的に定義されていないならば, *Rollup Objective Measure Weight* の既定値は 1.0 である.

Rollup Objective Measure Weight 要素は, 一つのアクティビティで得られた習得度が他のアクティビティで得られた習得度に比べてどの程度関連性があるかを指定する方法をコンテンツ開発者に提供する. 習得度ロールアッププロセス (付録 C:RB.1.1.参照) は, クラスタの正規化された習得度を決定するのに, 全てのクラスタの子の重み付き平均習得度を計算する. アクティビティの *Rollup Objective Measure Weight* が 0.0 の場合, アクティビティの *Objective Normalized Measure* が記録されても, 習得度ロールアッププロセス時に, LMS はアクティビティの *Objective Normalized Measure* を考慮しない.

3.8.3. Rollup Progress Completion

Rollup Progress Completion 要素は，アクティビティのトラッキング状態情報（セクション 4.2：トラッキングモデル参照）が，親の *Rollup Actions* が Completed および Incomplete のロールアップルールに適用されるかどうかを示す．この要素は論理型(True/False)の値を持つ．アクティビティに対して明示的に定義されていないならば，*Rollup Progress Completion* の既定値は True である．

アクティビティの *Rollup Progress Completion* が False の場合，アクティビティのトラッキング情報が記録されても，*Rollup Action* が Completed もしくは Incomplete の親のロールアップルールすべてに対して，LMS はアクティビティのトラッキング状態情報を考慮しない．

3.9. Rollup Consideration Controls

デフォルトでは, IMS SS 仕様は, 以下の条件でない限り, 全ての子アクティビティは親のロールアップに関与するとしている:

- アクティビティがトラッキングされていない (セクション 3.13.1: *Tracked* 参照) , もしくは,
- アクティビティが全くロールアップに貢献しない (セクション 3.8: *Rollup Controls* 参照)

アクティビティがロールアップ評価に関与しているにも関わらず, そのトラッキング状態情報の評価(*Rollup Condition*)が「unknown」の場合, 殆どのケースで, ロールアップ評価の結果は「unknown」値になる. ADL は実装とコミュニティのフィードバックを通して, 多くの共通ロールアップシナリオにはとってこの動作は厳格すぎることを見出した. ADL は, 表 3.9 に定義する通り, アクティビティが親のロールアップに関与するコンディションをさらに洗練した *Rollup Consideration Controls* を定義した.

表 3.9a: Rollup Consideration Controls

No.	名称	説明	値空間	既定値
1	*Measure Satisfaction If Active*	アクティビティがまだアクティブでも, アクティビティのロールアップ習得度は, アクティビティの *Minimum Normalized Measure* に対して評価すべきことを示す	論理型	True
2	*Required For Satisfied*	アクティビティのトラッキング情報が, 親のロールアップ Satisfied 状態にいつ寄与するか示す • always – 子は常に親のロールアップ評価に寄与する • ifNotSuspended –子が評価時に試行されおり中断されていないとき, 子は親のロールアップ評価に寄与する • ifAttempted –子が評価時に試行されていたとき, 子は親のロールアップ評価に寄与する • ifNotSkipped –子が評価時にスキップされていないとき, 子は親のロールアップ評価に寄与する	語彙	always
3	*Required For Not Satisfied*	アクティビティのトラッキング情報が, 親のロールアップ Not Satisfied 状態にいつ寄与するか示す • always – 子は常に親のロールアップ評価に寄与する • ifNotSuspended –子が評価時に試行されており中断されていないとき, 子は親のロールアップ評価に寄与する • ifAttempted –子が評価時に試行されていたとき, 子は親のロールアッ	語彙	always

		プ評価に寄与する • ifNotSkipped –子が評価時にスキップされていないとき, 子は親のロールアップ評価に寄与する		
4	*Required For Completed*	アクティビティのトラッキング情報が親のロールアップ Completed 状態にいつ寄与するか示す • always – 子は常に親のロールアップ評価に寄与する • ifNotSuspended –子が評価時に試行されており中断されていないとき, 子は親のロールアップ評価に寄与する • ifAttempted –子が評価時に試行されていたとき, 子は親のロールアップ評価に寄与する • ifNotSkipped –子が評価時にスキップされていないとき, 子は親のロールアップ評価に寄与する	語彙	always
5	*Required For Incomplete*	アクティビティのトラッキング情報が親のロールアップ Incomplete 状態にいつ寄与するか示す • always – 子は常に親のロールアップ評価に寄与する • ifNotSuspended –子が評価時に試行されて中断されていないとき, 子は親のロールアップ評価に寄与する • ifAttempted –子が評価時に試行されていたとき, 子は親のロールアップ評価に寄与する • ifNotSkipped –子が評価時にスキップされていないとき, 子は親のロールアップ評価に寄与する	語彙	always

3.9.1. Measure Satisfaction If Active

定義された習得度しきい値の評価 (*Objective Satisfied by Measure*) の評価は条件付きで行われる. *Measure Satisfaction If Active* 要素は, ロールアップ学習目標の習得にアクティビティのロールアップ学習目標習得度がいつ適用されるかを示す. この要素は, アクティビティの習得状態が必要で, 習得状態が習得度しきい値の評価 (*Objective Satisfied by Measure*) で決定されるときに適用される. この要素は論理型の (True/False) 値を持つ. アクティビティに対して明示的に定義されていないならば, Measure Satisfaction If Active の既定値は True である.

Measure Satisfaction If Active 要素が False の場合, LMS は, アクティビティへの試行が終了したときにだけ, *Objective Minimum Satisfied Normalized Measure* をアクティビティのロールアップ学習目標に適用する. それまで, アクティビティのロールアップ学習目標の習得状態は“unknown”と評価されなくてはならない.

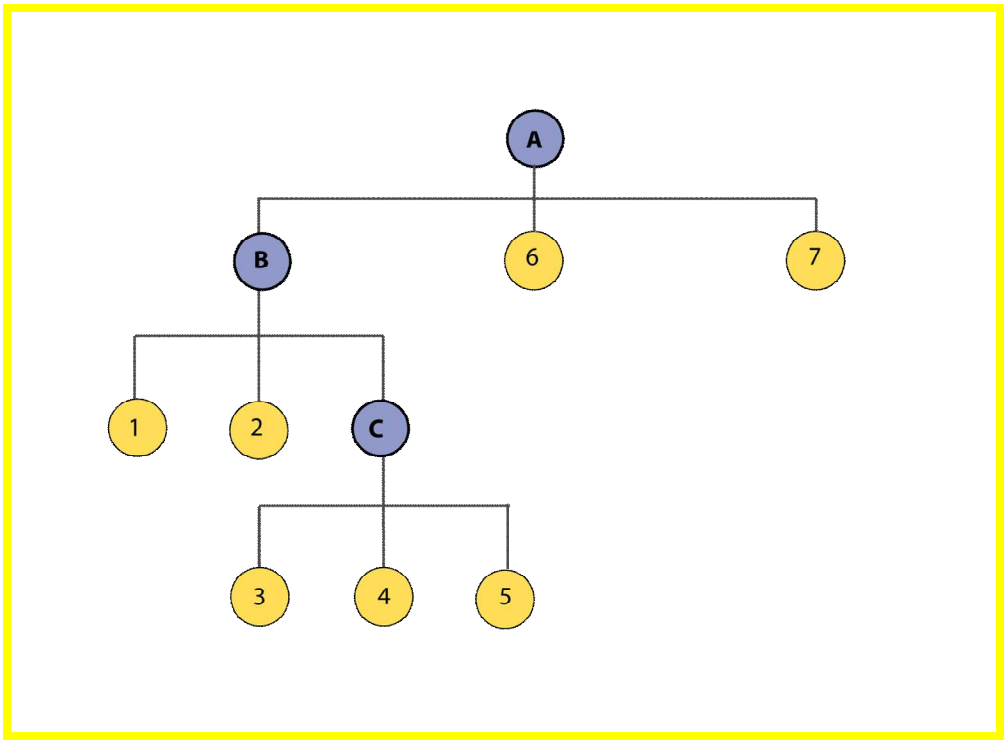

図 3.9.1a: Measure Satisfaction If Active の例

この値が学習目標習得の評価にどのように影響を与えるのか，例を図示する．図 3.9.1a で図示されたクラスタ，アクティビティ B を見てみよう．学習者がアクティビティ B の子を実行すると，子（アクティビティ 1，アクティビティ 2）またはアクティビティ C の子孫への試行が終わるたびにロールアップが呼び出される．それぞれの子は，アクティビティ B の学習目標の習得に対してなんらかの習得度を寄与し，アクティビティ B の習得度を変えていく．

アクティビティ B のひとつの子が終了してロールアッププロセスが呼び出されたとき，アクティビティ B は依然アクティブである．アクティビティ B の *Measure Satisfaction If Active* 要素が True（既定値）の場合，アクティビティ B の学習目標のロールアップ習得度は *Objective Minimum Satisfied Measure* と比較されて，アクティビティ B の学習目標は Satisfied もしくは Not Satisfied になる．このケースでは，アクティビティ B の子アクティビティのいずれかが試行された後，アクティビティ B の学習目標の状態が「unknown」となることはない．この動作は，すべての子に基づいてアクティビティ B の学習目標の習得を決定したいのであれば，望ましくない．

一方，*Measure Satisfaction If Active* 要素が False の場合，アクティビティ B のロールアップ（主）学習目標の習得は，アクティビティ B がアクティブでなくなるまで評価されず，アクティビティ B の子の習得度に依らず「unknown」に留まる．アクティビティ B がアクティブでなくなるひとつの場合は，アクティビティ B に適用された *Sequencing Exit* アクションルールが True で明示的に終了される場合である．

学習目標が一旦 Satisfied になったらクラスタを終了し，一方，クラスタの全ての子が試行されるまで学習目標を Not Satisfied とみなさないことが望まれる動作である場合，以下のステップが推奨される：

1. アクティビティの *Measure Satisfaction If Active* 要素を False に設定する
2. 学習目標に *Objective Minimum Satisfaction Measure* 要素を定義する

3. 学習目標習得度が *Objective Minimum Satisfaction Measure* に使われた同じ値より大きいという条件で *Sequencing Exit* アクションルールを適用する

3.9.2. Required For Rollup

IMS SS オーバーオールロールアッププロセスは, ロールアップに寄与するいずれかの子が「unknown」状態の場合, ロールアップ評価を行わない. この動作は, アクティビティがスキップ, 中断もしくは無効になるといくつかの問題を引き起こす. 子アクティビティを, いつ親のロールアップ評価に含むかに関して, コンテンツ開発者がより明示的な定義を提供可能とするために, 次の4つの要素が追加された: *requiredForSatisfied, requiredForNotSatisfied, requiredForCompleted,* および *requiredForIncomplete*

これらの *Required for Rollup* 要素は, 親のロールアップ評価の特定のロールアップルールに対して, 対応するアクティビティが対象となる状況を示す. これらの要素は, ロールアップ子チェックサブプロセス(セクション 4.6: ロールアップ動作参照)で評価される. これらの *Required for Rollup* 要素の値は以下のとおりである:

- **always** (既定値) –子は親のロールアップ評価に常に寄与する
- **ifNotSuspended** –評価時に試行されたが中断していないとき, 子は親のロールアップ評価に寄与する
- **ifAttempted** –試行されたとき, 子は親のロールアップ評価に寄与する
- **ifNotSkipped** –評価時にスキップされていないとき, 子は親のロールアップ評価に寄与する

Required for Rollup 要素は, *Tracked, Rollup Objective Satisfied* もしくは *Rollup Progress Completion* を False と定義することで親のロールアップに明示的に含まれないアクティビティには影響を与えない.

Required for Rollup 要素は, ロールアップ時にステイタス情報を提供する子だけに影響を与える. *Rollup Child Activity Set* は, ロールアップルール評価時に, 評価された各ロールアップルールに対して提供されたステイタス情報が, どのように適用されるかを定義する.

3.10. 学習目標記述

IMS SS 仕様の SCORM への導入とともに, 学習目標をアクティビティに関連付けるメカニズムが成立した. アクティビティは一つもしくは複数の学習目標と関連付けることが可能である. 各学習目標は, 表 3.10a に示す要素を使用して, 記述されなければならない.

表 3.10a: 学習目標記述

No.	名称	説明	価空間	既定値
1	*Objective ID*	アクティビティに付随する学習目標の識別子. IDは対応する学習目標の学習目標進捗情報へのリンクである.	固有識別子	値は要求されない
2	*Objective Satisfied by Measure*	アクティビティの学習目標が習得されたかどうか決定するために, 他の方法ではなく *Objective Minimum Satisfied Normalized Measure*が使われること (真または偽) を示す.	論理型	False
3	*Objective Minimum Satisfied Normalized Measure*	学習目標に対する最小習得度を示す。-1以上, 1以下に正規化されている. 学習目標の *Objective Measure Status*が真で, *Objective Normalized Measure*がこの値以上の場合, *Objective Progress Status* が真, *Objective Satisfied Status*が真に設定される. 学習目標の*Objective Measure Status*が真で, *Objective Normalized Measure*がこの値未満の場合, *Objective Progress Status* が真, *Objective Satisfied Status*が偽に設定される. *Objective Satisfied by Measure*が真でなければ, この値は未定である.	実数型[-1..1] の 有効数字4桁の実数	1.0
4	*Objective Contributes to Rollup*	ロールアップの間, 学習目標の *Objective Satisfied Status* と *Objective Normalized Measure*が使われること (真または偽) を示す.	論理型	False

SCORM は, どのように学習目標が定義され, 使用され, 解釈されるかについて言及しないが, シーケンシングのために, アクティビティに付随する各学習目標は, 学習目標に対する学習者進捗の追跡を行うトラッキング状態情報を持ち, 条件付きシーケンシング判断を可能にする. 図 3.10a は, 学習目標記述と, アクティビティの学習目標の使用に関連付けられた学習目標進捗情報の関係を図示する.

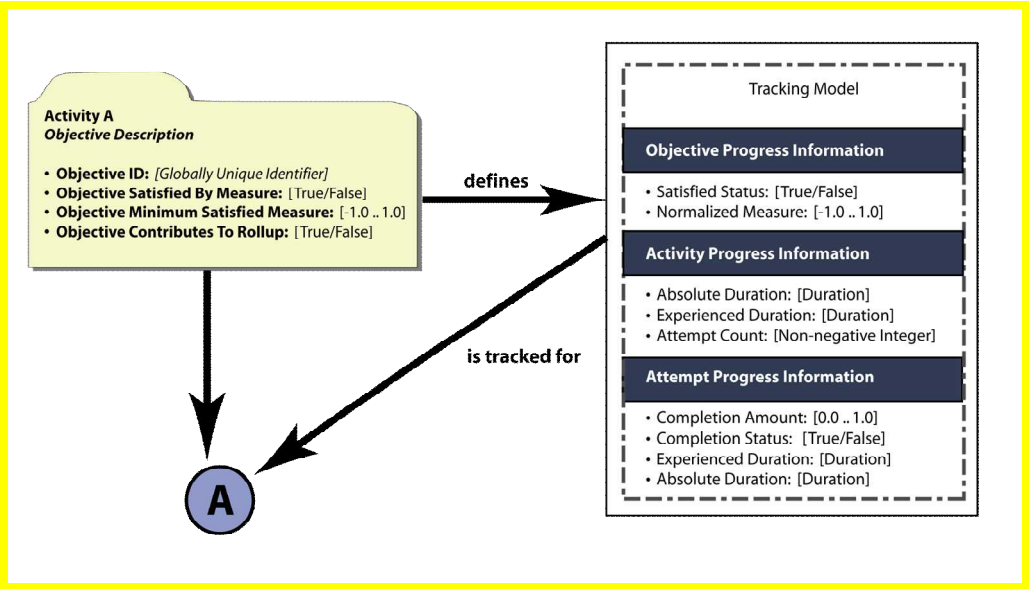

図 3.10a: 学習目標記述と学習目標進捗情報の関係

各学習目標記述は以下の情報から成り立つ:

- **Objective ID**: *Objective ID* 要素は, 学習目標進捗情報とアクティビティの間のリンクの役割を果たす. *Objective ID* 要素には既定値は定義されない. *Objective ID* 要素は, 学習目標に対して学習目標マップ (3.10.3 学習目標マップ参照) が定義された場合のみ必要となる.
  ADL ノート: *Objective ID* のデータタイプはグローバルな固有識別子 (globally unique identifier) であるが, これは *Objective ID* がアクティビティツリー内の全てのアクティビティにわたって識別可能でなければいけないということではなく, 学習目標が使用 (アクセス) される範囲内に限り識別可能でなければいけないということである. デフォルトでは, アクティビティは, アクティビティに対して定義された一組の学習目標 (これらはローカル学習目標と呼ばれる) の *Objective Progress Information* にだけアクセスできる. シーケンシング評価時に学習目標進捗情報が明確に判別できるように, ひとつのアクティビティに対応する全ての *Objective ID* は識別可能でなければならない. 2つもしくはそれ以上のアクティビティツリー内のアクティビティが同じ *Objective ID* の学習目標を持つ場合がある.

- **Objective Satisfied by Measure** (*False* – 既定値): *Objective Satisfied by Measure* 要素は, 学習目標の習得(セクション 4.2.1.2: *Objective Progress Information* 参照)を決定するために, 他の方法ではなく, *Objective Minimum Satisfied Normalized Measure* 要素と学習目標の習得度 (つまりスコア) を用いることを示す. もし *Objective Satisfied By Measure* 要素が True であれば:
  - 学習目標の *Objective Measure Status* が True かつ学習目標の *Objective Normalized Measure* が *Objective Minimum Satisfied Normalized Measure* と同じもしくは超える場合, *Objective Progress Status* が True に設定されかつ *Objective Satisfied Status* が True に設定される.
  - 学習目標の *Objective Measure Status* が True かつ学習目標の *Objective Normalized Measure* が *Objective Minimum Satisfied Normalized Measure* より小さい場合, *Objective Progress Status* が True に設定されかつ *Objective Satisfied Status* が False に設定される.
  - 学習目標の *Objective Measure Status* が False の場合, *Objective Progress Status* が False に設定される.

- **Objective Minimum Satisfied Normalized Measure**(*1.0* – 既定値): *Objective Minimum Satisfied Normalized Measure* 要素は, 学習目標を達成するための閾値を示す. この要素は-1.0 および 1.0 間の実数値を有する. アクティビティに対して明示的に定義されていなければ, *Objective Minimum Satisfied Normalized Measure* 要素の既定値は 1.0 である.
- **Objective Contributes to Rollup** (*False* – 既定値): *Objective Contributes to Rollup* 要素は, 学習目標の *Objective Normalized Measure* と *Objective Satisfied Status* が, ロールアップ評価時に使用されるかどうかを示す.

ADL ノート: この要素は IMS SS シーケンシング定義モデルで定義されているが, 直接, 設定や変更はできない. SCORM コンテンツパッケージ (SCORM CAM 分冊, 5.1.7.1: <primaryObjective> 要素参照) の XML 要素<primaryObjective>がロールアップに関与する単一の学習目標を指定するのに使われる. 一つのアクティビティに対して, <primaryObjective>と定義された学習目標だけがロールアップに関与する (*Objective Contributes to Rollup* が True となる).

3.10.1. ローカル学習目標 vs 共有グローバル学習目標

学習目標は, それぞれ, 学習目標進捗情報(セクション 4.2:トラッキングモデル参照)からなるローカルおよびグローバルなスコープのデータ項目の組で, 学習アクティビティとは別のものである. アクティビティは, そのアクティビティに対して定義された学習目標 (このような学習目標はローカル学習目標と呼ばれる) の学習目標進捗情報だけにアクセスできる. アクティビティは, 別のアクティビティの学習目標の学習目標進捗情報を直接参照することはできないが, 学習目標マップを定義することで, ローカル学習目標を共有グローバル学習目標に関連付けることが可能である. アクティビティは, ローカル学習目標を一つ以上持ち, 複数の共有グローバル学習目標を参照することが可能である. 複数のアクティビティが同一の共有グローバル学習目標を参照し, これによって学習目標進捗情報を共有することができる. このような例を図 3.10.1a に示す. Objective 5 を除いた全学習目標が, 対応するアクティビティに対してローカルである. Objective 5 はアクティビティ B とアクティビティ C に共有された共有グローバル学習目標である.

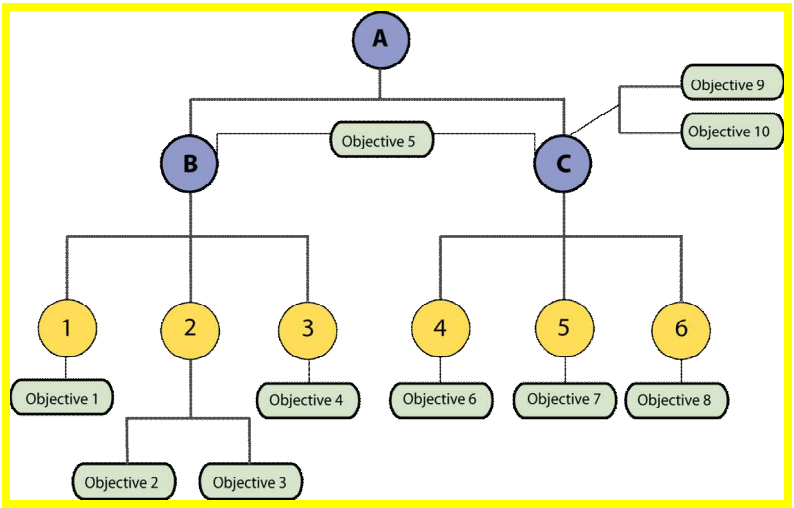

図 3.10.1a: 学習目標共有例

3.10.2. Objectives Global to System

Objectives Global to System 要素は，アクティビティツリーに対して適用され，アクティビティツリーで参照される共有グローバル学習目標 ID のスコープを示す．また，アクティビティツリーに関連付けられた共有グローバル学習目標の学習目標進捗情報(セクション 4.2: トラッキングモデル参照)の存続時間も示す．しかし，この要素は，学習目標進捗情報をどのように管理するか，どのようにローカル学習目標 ID と共有グローバル学習目標 ID の決定が行われるかについては定義しない．この要素は論理型(True/False)の値を持つ．アクティビティツリーに対して明示的に定義されないならば，*Objective Global to System* の既定値は True である．共有グローバル学習目標情報は，学習者毎に，システム内の学習者の存続期間中存在する．

アクティビティツリーの *Objectives Global to System* 要素が False と定義された場合，LMS は，共有グローバル学習目標 ID のスコープをアクティビティツリー内のみに絞る．共有グローバル学習目標情報は学習者毎かつアクティビティツリーの試行毎に存在する．

Objectives Global to System 要素は，アクティビティツリーにのみ適用でき，アクティビティに定義しても効果がない．

ADL ノート: SCORM コンテンツパッケージ（セクション 2.1.1: コンテンツパッケージからのアクティビティツリーの導出参照）からアクティビティツリーを導出するとき，ゼロもしくは一つの Objectives Global to System 要素が，アクティビティツリーに対応する<organization>要素に定義されている．この要素はアクティビティツリー内で使用される全ての objective ID のスコープを定義する．

3.10.3. 学習目標マップ

表 3.10.3a は，LMS が，どのようにローカル学習目標を共有グローバル学習目標に対応させるか，そして，いつ参照する学習目標進捗情報にアクセスすべきかを説明している．

表 3.10.3a: 学習目標マップ

No.	名称	説明	値空間	規定値
1	*Activity Objective ID*	アクティビティに関連付けられたローカル学習目標の識別子 **注意:** Activity Objective ID に既定値はない．アクティビティに学習目標マップを定義する場合，Activity Objective ID は必須である	固有識別子	なし 値を設定しなくてはならない
2	*Target Objective ID*	マッピングの対象となる共有グローバル学習目標の識別子 **注意:** Target Objective ID に既定値はない．アクティビティに学習目標マップが定義される場合 Target Objective ID は必須である	固有識別子	なし 値を設定しなくてはならない
3	*Read Objective Satisfied Status*	ローカル学習目標の進捗が未定義（ローカル学習目標の *Objective Progress Status* が False）の時，ローカル学習目標（*Activity Objective ID*）の *Objective Progress Status* と *Objective Satisfied Status* の値を，指定された共有グローバル学習目標（*Target Objective ID*）から読み出すこと（True または False）を示す．	論理型	True

		この操作はローカル学習目標の学習目標情報を変更しない.		
4	*Write Objective Satisfied Status*	アクティビティ試行の終了時に, ローカル学習目標 (*Activity Objective ID*)の *Objective Progress Status* と *Objective Satisfied Status* の値を, 指定された共有グローバル学習目標 (*Target Objective ID*)に転送すること (True または False) を示す.	論理型	False
5	*Read Objective Normalized Measure*	ローカル学習目標の習得度が未定義 (ローカル学習目標の *Objective Measure Status* が False) の時, ローカル学習目標 (*Activity Objective ID*) の *Objective Measure Status* と *Objective Normalized Measure* の値を, 指定された共有グローバル学習目標 (*Target Objective ID*) から読み出すこと (True または False) を示す. この操作はローカル学習目標の学習目標情報を変更しない.	論理型	True
6	*Write Objective Normalized Measure*	アクティビティ試行の終了時に, ローカル学習目標 (*Activity Objective ID*)の *Objective Measure Status* と *Objective Normalized Measure* の値を, 指定された共有グローバル学習目標 (*Target Objective ID*)に転送すること (True または False) を示す.	論理型	False

ADL ノート: SCORM コンテンツパッケージマニフェストでは, 表 3.10.3a で指定される *Activity Objective ID* は学習目標マップが定義されているコンテクストで暗に前もって定義されている. XML<mapInfo>要素は<primaryObjective> 要素ないし<objective>要素の中で学習目標マップを定義するために用いられる. XML<mapInfo>要素は SCORM コンテンツパッケージの中で学習目標マップを定義するために用いられる (SCORM CAM, 5.1.7.1.2: <*mapInfo*>要素参照) . 各<mapInfo>要素は, 学習目標に対して, その学習目標の Objective ID を *Activity Objective ID* として使用するひとつの学習目標マップを定義する.

各学習目標マップは, アクティビティのローカル学習目標進捗情報と共有グローバル学習目標との出入りのマッピングを定義する. 学習目標マップは, アクティビティ間で学習目標進捗情報を共有するためのカギである. 学習目標マップを適用する際, いくつかのルールがある:

- 各アクティビティの学習目標マップの数に制限はない.
- デフォルトでは, 学習目標進捗情報はアクティビティ間で共有されない. ローカル学習目標情報を共有グローバル学習目標にどのようにマッピングするかを表すためには, 各アクティビティで学習目標マップを定義しなければならない.
- トラッキング動作 (セクション 4.2.1.7 参照) に記述されているとおり, ローカル学習目標情報が変更される際は, 常に学習目標マップが使用される.
- あるアクティビティにおいて, 各ローカル学習目標は一つの共有グローバル学習目標だけから状態や学習目標習得度を読み取ることが可能である.
- あるアクティビティにおいて, 複数のローカル学習目標が同一の共有グローバル学習目標に情報を書き込むことはできない.

3.11. 選択コントロール

コンテンツ開発者は, アクティビティを選択するタイミングおよび数を示すシーケンシング情報を定義することができる. これは, 例えば「あるアクティビティの最初の試行で, 6 個のアクティビティから 4 個を選べ」というルールを書いて LMS のシーケンシング実装に伝えることが可能になるということである. 表 3.11a は選択コントロールを説明する.

表 3.11a: 選択コントロール

No.	名称	説明	値空間	既定値
1	*Selection Timing*	選択がいつ起こるべきかを示す. • *Never* – 選択が一度も適用されない; すべての子アクティビティがデフォルトとして選択される. • *Once* – 選択がアクティビティの最初の試行の前に適用される. • *On Each New Attempt* – 選択がアクティビティの新しい試行の前に適用される. *On Each New Attempt* オプションとその関連動作はこの SCORM バージョンでは規定されていない.	語彙	Never
2	*Selection Count Status*	アクティビティの *Selection Count* データが意味のあるものである (真か偽) ことを示す.	論理型	False
3	*Selection Count*	アクティビティの子アクティビティ集合から選択されなければならない子アクティビティの数を示す. *Selection Count* が子アクティビティの数より大きい場合, すべての子アクティビティが選択される. *Selection Count Status* が True でなければこの値は意味を持たない. *Selection Count Status* が False の場合, すべての子アクティビティが選択される.	非負整数型	0

3 つの選択コントロール要素は密接に関係している. 第一の *Selection Timing* 要素は, いつ (もし発生すれば) クラスタの子が選択されるべきかを示す. *Selection Timing* 要素は以下の値を持つ語彙である:

- **Never**(既定値): クラスタに選択が適用されることは決してない. アクティビティツリーで定義されたクラスタの全ての子がデフォルトで対象となる.
- **Once**: クラスタへの最初の試行の前に選択が適用される.
- **On Each New Attempt**: クラスタへ新たな試行が起こされる前に必ず選択が適用される.

標準シーケンシング動作擬似コード(付録 C 参照)は, いつ子選択プロセスが呼び出されるか明示的に定義していない. LMS は, 選択ランダム化動作(セクション 4.7 参照)に記述されている通り, 子選択プロセスの適用が *Selection Timing* 要素の定義された値と一貫性があることを保証しなくてはならない.

第二の *Selection Count* 要素はクラスタの子がいくつ選択されるかを示す．この要素は負でない整数値を含む．アクティビティに対して明示的に定義されないならば，*Selection Count* の既定値はゼロである．*Selection Count* がクラスタの子の数を超える場合，全ての子が選択される．

第三の *Selection Count Status* 要素は *Selection Count* 値が有効であるかどうかを示す．この要素は論理型 (True/False) の値を含む．アクティビティに対して明示的に定義されないならば，*Selection Count Status* の既定値は False である．

ADL ノート: *Selection Timing* 要素の値に関わらず，クラスタの子の選択が発生するには，*Selection Count Status* 要素が明示的に True と定義され，かつ *Selection Count* が明示的にゼロ以外の整数でなければならない．それ以外は，アクティビティツリーで定義された クラスタの全ての子がデフォルトで対象となる．

選択コントロール要素は葉アクティビティに定義されても効果がない．

3.12. ランダム化コントロール

ランダム化コントロール (表 3.12a) は, 様々なシーケンシング動作 (セクション 4: シーケンシング動作参照) を実行する際に, 対象となるクラスタアクティビティの子を順序付けし直すのに, LMS がいつどんなアクションを取るべきかを表している. コンテンツ開発者は, アクティビティツリーのいずれのクラスタにもランダム化コントロールを適用することが可能である.

表 3.12a: ランダム化コントロール

No.	名称	説明	値空間	既定値
1	*Randomization Timing*	子アクティビティの順序付けがいつ起こるべきかを示す. • *Never* – ランダム化が適用されない. • *Once* – ランダム化がアクティビティの最初の試行の前に適用される. • *On Each New Attempt* – ランダム化がアクティビティの新しい試行の前に必ず適用される.	語彙	Never
2	*Randomize Children*	子アクティビティの順番がランダム化されることを示す (真か偽).	論理型	False

二つのランダム化コントロール要素は強く関連し合っている. 第一の *Randomization Timing* 要素は いつ (もし発生すれば) クラスタの子が順序付けし直されるべきかを示す. *Randomization Timing* 要素は以下の値を持つ語彙である:

- **Never**(既定値): クラスタにランダム化が適用されることは決してない.
- **Once**: クラスタへの最初の試行の前にランダム化が適用される.
- **On Each New Attempt**: クラスタへ新たな試行が起こされる前に必ずランダム化が適用される

標準シーケンシング動作擬似コード (付録 C 参照)は, いつ子ランダム化プロセスが呼び出されるか明示的に定義していない. LMS は, 選択ランダム化動作(セクション 4.7 参照)に記述されている通り, 子ランダム化プロセスの適用が *Randomization Timing* 要素の定義された値と一貫性があることを保証しなくてはならない.

第二の *Randomize Children* 要素はクラスタの子が順序付けし直されるかどうかを示す. この要素は論理型 (True/False) の値を含む. アクティビティに対して明示的に定義されないならば, *Randomize Children* の既定値は False である.

ADL ノート: *Randomization Timing* 要素の値に関わらず, クラスタの子の再順序付けを発生させるには *Randomize Children* 要素が明示的に True と定義されなければならない.

ランダム化コントロール要素は葉アクティビティに定義されても効果がない.

3.13. 配信コントロール

配信コントロール (表 3.13a) は, アクティビティへの試行が開始する前と試行が終了した後に LMS が取るべきアクションを表す. 配信コントロールは, LMS がアクティビティのトラッキング状態情報の管理を補助するために使用される. この要素は, アクティビティに対応した SCO が特定のタイプのトラッキング情報を通信すると LMS が予期してよいかどうかを示す.

表 3.13a: 配信コントロール

No.	名称	説明	値空間	既定値
1	*Tracked*	試行における学習目標進捗情報およびアクティビティ／試行進捗情報を記録し (真または偽), 他のシーケンシング情報がそれを妨げない限り, データがアクティビティの親アクティビティのロールアップに含まれることを示す. トラッキング状態情報がどのように追跡記録されるかは定義されない	論理型	True
2	*Completion Set by Content*	アクティビティの *Attempt Completion Status* が, アクティビティに対応付けられたコンテンツオブジェクトによって設定されることを示す.	論理型	False
3	*Objective Set by Content*	アクティビティに付随する, *Objective Contributes to Rollup* が True の学習目標の *Objective Satisfied Status* が, アクティビティに対応付けられたコンテンツオブジェクトによって設定されることを示す.	論理型	False

3.13.1. Tracked

Tracked 要素は, アクティビティのトラッキング状態情報(セクション 4.2: トラッキングモデル参照)が管理されているかどうかを示す. この要素は論理型(True/False)の値を持つ. アクティビティに対して明示的に定義されていないならば, *Tracked,*の既定値は True である.

アクティビティの *Tracked* 要素が False の場合, LMS はアクティビティのトラッキング状態情報の初期化, 管理, アクセスのいずれも行わない. トラッキング状態情報を用いる全ての評価でデフォルトの「unknown」値が用いられる. *Tracked* 要素が False のアクティビティは, 親に対するロールアップ評価に含まれない.

ADL ノート: コンテンツ開発者は, アクティビティを「追跡しない」(*Tracked* を False) と宣言した場合, LMS は「追跡しない」アクティビティのいかなる状態も間知りないので, 「読み出し」学習目標マップはシーケンシング評価には用いられないことに中止しなくてはならない. LMS は「追跡しない」アクティビティのすべての評価に対して「未定」を返す. しかし, 「読み出し」学習目標マップが存在すると, その情報は SCO のランタイム環境の `cmi.objectives` 集合を初期化するのに用いられる (RTE 4.2.17: 学習目標参照).

ADL ノート: アクティビティツリーへシーケンシング戦略を適用する際, 要求された条件付き動作のために, しばしば大量のトラッキング情報が必要となる. アクティビティの *Tracked* 要素を False に設定するとアクティビティとその祖先に適用されるシーケンシング戦略が制限されてしまうことに, コンテンツ開発者は気をつける必要がある.

3.13.2. Completion Set by Content

Completion Set by Content 要素は，アクティビティに対応付けられたコンテンツオブジェクトが，アクティビティが完了したかどうかを通知する義務があることを示す．この情報は，アクティビティの試行進捗情報(セクション 4.2: トラッキングモデル参照)に影響を与える．この要素は論理型(True/False)の値を持つ．アクティビティに対して明示的に定義されていないならば，*Completion Set by Content*,の既定値は False である．

葉アクティビティの *Completion Set by Content* 要素が True の場合，LMS はアクティビティの試行完了状態に関して仮定を行わない．つまり，アクティビティに対応したコンテンツオブジェクトが完了情報を通知しないと，アクティビティの完了ステイタスは「unknown」となる－ *Attempt Progress Status* が False となる．

葉アクティビティの *Completion Set by Content* 要素が False で，アクティビティに対応付けられたコンテンツオブジェクトが完了情報を通知しない場合，アクティビティへの現在の試行が終了したら，LMS はアクティビティが完了したと仮定する－ *Attempt Progress Status* が True となり *Attempt Completion Status* が True となる．

ADL ノート: *Completion Set by Content* 要素の既定値は False であり，これにより非通信型コンテンツオブジェクト(Assets)をシーケンシング戦略に用いることができる．既定値が使用される場合においても，コンテンツオブジェクトから通信された情報が存在するなら，それが常に使用され，LMS はその情報を変えることはない．一般的に，アクティビティに関連付けられた特定のシーケンシング情報の中で，コンテンツ開発者がこの要素を明示的に取り入れて True に設定する必要はない．

Completion Set by Content 要素は，クラスタアクティビティに定義されても効果がない．

3.13.3. Objective Set by Content

Objective Set by Content 要素は，アクティビティに対応付けられたコンテンツオブジェクトが，アクティビティのロールアップ学習目標が達成されたかどうかを通知する義務があることを示す．この情報は，アクティビティのロールアップ学習目標の学習目標進捗情報(セクション 4.2: トラッキングモデル参照)に影響する．この要素は論理型(True/False)の値を持つ．アクティビティに対して明示的に定義されていない場合，*Objective Set by Content*,の既定値は False である．

葉アクティビティの *Objective Set by Content* 要素が True の場合，LMS はアクティビティのロールアップ学習目標の学習目標習得状態に関して仮定を行わない．つまり，アクティビティに対応したコンテンツオブジェクトが習得情報を通知しない場合，ロールアップ学習目標は「unknown」となる－ *Objective Progress Status* は False となる．

葉アクティビティの *Objective Set by Content* 要素が False で，アクティビティに対応付けられたコンテンツオブジェクトが習得情報を通知しない場合，現在の試行が終了すると，LMS はアクティビティのロールアップ学習目標が習得されたと仮定する－ *Objective Progress Status* が True となり *Objective Satisfied Status* が True となる．

ADL ノート: *Objective Set by Content* 要素の既定値は False であり，非通信型コンテンツオブジェクト(Assets)をシーケンシング戦略に用いることができる．既定値が使用される場合においても，コンテンツオブジェクトから通信された情報が存在するなら，それが常に使用され，LMS はその情報を変えることはない．一般的に，アクティビティに関連付けられた特定のシーケンシング情報の中で，コンテンツ開発者がこの要素を明示的に取り入れて True に設定する必要はない．

Objective Set by Content 要素は，クラスタアクティビティに定義されても効果がない．

セクション4
シーケンシング動作

このページは空白である.

4.1. シーケンシング動作概要

このセクションでは, 様々なシーケンシングプロセスの動作を記述する. SCORM シーケンシングプロセスは IMS SS 仕様に記述されたプロセスから導出されている. 以下の説明は IMS SS 仕様の詳細説明を置き換えることを意図するものではなく, 主要な機能とプロセスの特性の抽出を促進することを意図している. いくつかのケースでは, SCORM シーケンシングは IMS SS プロセスを拡張あるいは変更している. このため, 付録Cに詳細に記述しているシーケンシング動作擬似コードは, IMS SS 仕様の擬似コードを置き換えており, これは SCORM 対応 LMS の規範である.

IMS SS 仕様には, アクティビティツリーの各アクティビティに適用される二つのデータモデルがある. ひとつは, アクティビティの状態を保持するデータモデル, そして, アクティビティが処理される際にコンテンツ開発者のシーケンシング意図を記述するためのデータモデルである. さらに個々のアクティビティとアクティビティツリー全体の状態を保持するための状態モデルが定義されている. シーケンシングプロセスは, シーケンシング動作擬似コード (付録C参照) に示すように 3 つのモデル全ての情報を使用する. データモデルとアクティビティとの関係は以下のようにまとめられる:

- **トラッキングモデル** (セクション 4.2:トラッキングモデル参照) ―学習者アクティビティに対応するコンテンツオブジェクトのインタラクションから得られた情報を取り込む. これは動的なランタイム時 (学習者がコンテンツオブジェクトおよび LMS と対話(interact)している間) のデータモデルである.
- **アクティビティ状態モデル** (セクション 4.2.1.5:アクティビティ状態情報参照) ―アクティビティツリーの各アクティビティのシーケンシング状態とアクティビティツリーのグローバル状態を管理する. これは, LMS シーケンシング実装がシーケンシングセッション中にアクティビティの状態を管理するために使用する動的ランタイムデータモデルである.
- **シーケンシング定義モデル** (セクション 3:シーケンシング定義モデル参照) ―様々なシーケンシングプロセスが, 定義されたシーケンシング動作を提供するために, アクティビティをシーケンスするトラッキングモデル情報をどのように使用し, 解釈するかについて記述する. 通常, これは (SCORM コンテンツパッケージで定義されている) 静的データモデルで, コンテンツオーガニゼーションの作成されたシーケンシング意図を表わす.

様々なシーケンシング動作は互いに独立しているが, 上記の 3 組のデータを用いて実行される. 各シーケンシング動作は, いくつかのプロセスおよびサブプロセスから成り立ち, これらのプロセスおよびサブプロセスは, 十分に定義されている動作を実現するが, 他のシーケンシング動作にも直接依存することはない. すなわち, ひとつのシーケンシング動作が他のシーケンシング動作を直接呼び出すことはない. オーバーオールシーケンシングプロセスは, *sequencing session* および *sequencing loop* (セクション 4.3.1 シーケンシングループ参照)の中で, 全てのシーケシング動作がお互いにどのように関連するかについて記述している.

4.2. トラッキングモデル

アクティビティの条件付きシーケンシングを実現するために，配信されるアクティビティに対応して起動されたコンテンツオブジェクトと学習者とのインタラクションに関する情報が，保持，管理されなければならない．IMS SS 仕様では，アクティビティツリーの各アクティビティに対して保持しなければならないトラッキング情報が記述されている．トラッキング情報を記述するデータモデル要素はトラッキングモデルと呼ばれる．

SCORM は，実装において，トラッキングモデルをどのように表現，および，管理するか，なにも要求していない．さらに，ある時点で各アクティビティに対して一組のトラッキング情報だけが存在するという要求もなければ，トラッキングモデルが下記に記述された要素から成り立つ，もしくはこれらの要素に限定されるという要求もない．実装に対して要求されていることは，このセクションで記述されるトラッキングモデルに対して動作しているとみなされるように，シーケンシング擬似コード（付録 C）に記述されている動作を実現することである．アクティビティツリー状態の評価を実行する際に，実装はトラッキング情報を自由に管理，最適化することができる．しかし，オーバーオールシーケンシングプロセス（セクション 4.3 参照）の中で様々なシーケンシングプロセスが適用されるとき，全てのプロセスは同一の有効なトラッキング情報を使用して，シーケンシング動作の一貫性のある適用を確保しなければならない．

4.2.1. トラッキングモデル概要

前バージョンの SCORM では，規定されたデータモデルは SCORM ランタイム環境データモデルだけであった．この情報は学習者の SCO とのインタラクションをトラッキングするために使用されていた．シーケンシング規格の追加により，LMS は，もう一つのデータモデルを管理するように規定された．それがトラッキングモデルである．トラッキングモデルは，各学習者に対するアクティビティツリー内で，各アクティビティに関連付けられた動的シーケンシングステイタス情報の集合体である．全てのトラッキングモデル要素に対して既定値が定義されている．学習行為中，トラッキングモデル要素は，現在実行中のコンテンツオブジェクトと学習者のインタラクションが反映されるよう更新される．

SCORM RTE ブックで定義された SCORM ランタイム環境データモデルを用いて，SCO は学習者とコンテンツオブジェクト（例：ステイタス，スコア）とのインタラクションに関する情報を通信する．SCORM ランタイム環境データモデル要素のいくつかは，トラッキングモデル要素に直接対応する．図 4.2.1a は，アクティビティツリー，ある特定のアクティビティのトラッキング情報，そのアクティビティに対応付けられたコンテンツオブジェクトおよびコンテンツオブジェクトのランタイムデータの間の概念的な関係を図示している．二つのデータモデルの要素間の各々の関係は以降のセクションで記述される．SCORM ランタイム環境データモデル要素が，いつどのようにトラッキングモデル要素にマッピングされるかについての詳細は，SCORM ランタイム環境データモデル[4]（セクション 4 参照）を参照のこと．

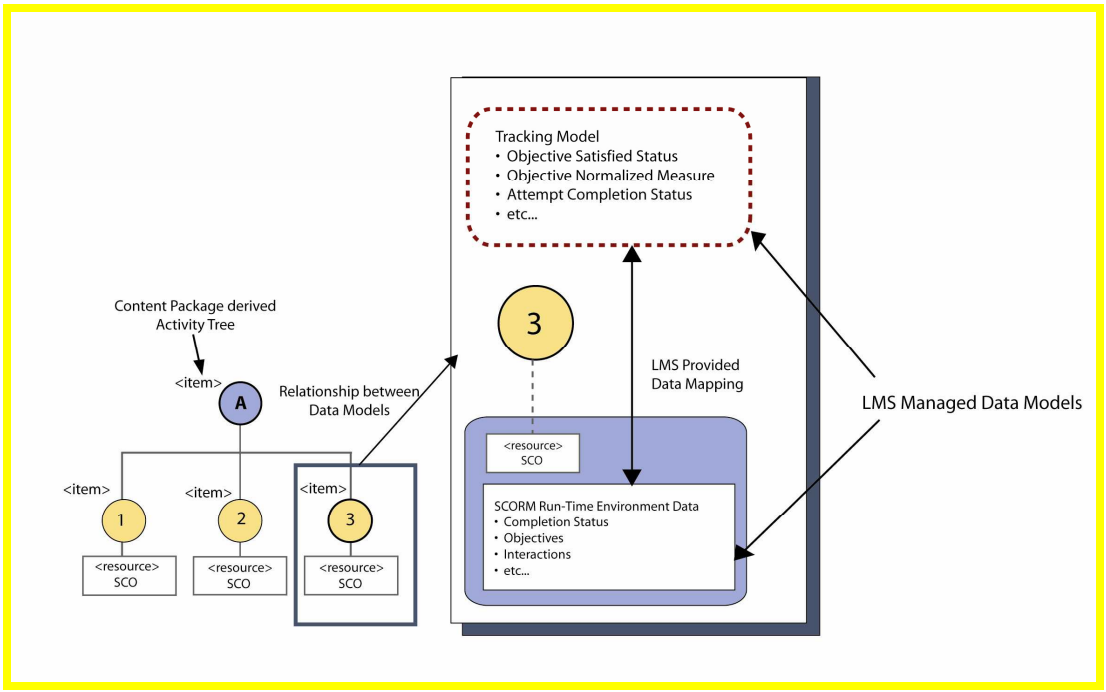

図 4.2.1a:ランタイム環境データモデルとトラッキングモデルとの関係

4.2.1.1 トラッキングモデル

全てのアクティビティは，アクティビティを経験する各学習者にそれぞれ固有のトラッキング状態情報を対応付ける．図 4.2.1.1a は，アクティビティツリーと各アクティビティに対応付けられたトラッキング情報の例を示す．LMSは，起動されたコンテンツオブジェクトと学習者とのインタラクションによって，実行時にトラッキング情報を更新すると仮定している．SCO と対応付けられたアクティビティについて，LMS は SCO と通信した情報に基づいてトラッキングモデルを管理する．アセットについては，通信は行なわないが，アセットに対応するアクティビティのトラッキング情報を管理するために LMS を補助する要求が以降のセクションにいくつか定義されている．

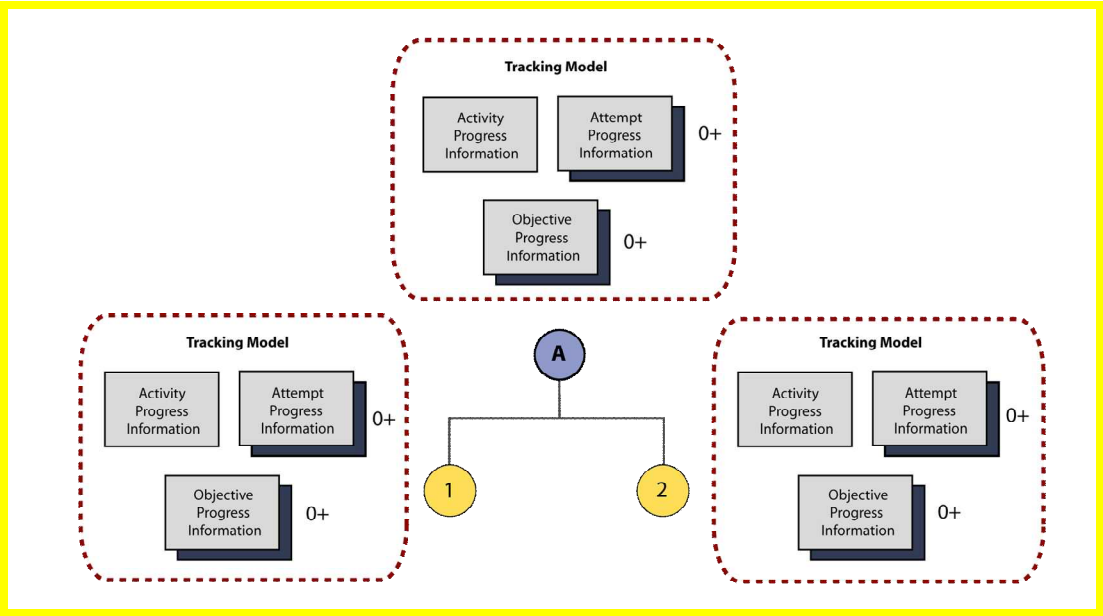

図 4.2.1.1a: トラッキングモデル

アクティビティのトラッキングモデル要素の値の変化は，アクティビティの親のトラッキング状態情報の値に影響を与える．子のトラッキング状態の変化に基づいて親アクティビティのトラッキング状態を評価するプロセスをロールアップと呼ぶ．ロールアップ動作（セクション 4.6 参照）はトラッキング状態情報が更新される必要があるとき LMS によって引き起こされる．

トラッキングモデルは，追跡された *Tracked* が真のシーケンスされた学習アクティビティを配信するために，システムが保持すべき情報を記述している．LMS は，定義された各アクティビティのトラッキング状態情報を保持しなければならない．また，LMS は SCO のランタイムデータを適切なトラッキングモデル要素にマッピングできなければならない[4]（セクション 4: SCORM ランタイム環境データモデル）．追跡されない（*Tracked* が偽の）アクティビティについては，トラッキング状態情報に対するすべての要求について，LMS はデフォルト値「未定」を提供する．

トラッキングモデルは以下のトラッキング状態情報を定義する：

- **学習目標進捗情報**：学習目標に関する学習者の進捗を記述する
- **アクティビティ進捗情報**：アクティビティにおける学習者の進捗を記述する．この情報は，アクティビティにおける全ての試行にわたる累積的な進捗状況について記述する
- **試行進捗情報**：アクティビティにおける学習者の進捗について記述する．
- **アクティビティ状態情報**：学習者毎，アクティビティツリー毎のアクティビティの状態を記述する

4.2.1.2 学習目標進捗情報

アクティビティは一つないしは複数の学習目標と対応付けられる．SCORM は，学習目標がどのように定義され，使用され，もしくは解釈されるかについては述べない．アクティビティに対応付けられた各学習目標は，シーケンシングのために，学習者の学習目的に対する進捗状況がトラッキングできる一組のトラッキング情報を持ち，これによって条件付きのシーケンシング判断が可能となる．

トラッキングされた各学習目標のシーケンシング特性は，シーケンシング定義要素 *Objective Description* によって記述される．アクティビティへの各試行において，学習者はアクティビティに対応付けられた各学習目標に対する一組の学習目標進捗情報（表 4.2.1.2a）を得る．学習目標進捗情報の要素は，様々なシーケンシングプロセス中に，シーケンシング動作擬似コード（付録C参照）から参照される．学習目標進捗

情報の表で各要素の既定値が定義されている. LMS のシーケンシング実装によって明示的に設定されるまで, LMS は既定値を使用する.

表 4.2.1.2a: 目標進捗情報

No.	要素	説明	値空間	既定値
1	*Objective Progress Status*	現在, 学習目標が有効な習得度を持つこと示す.	論理型	False
2	*Objective Satisfied Status*	学習目標が習得されたことを示す (真または偽) . 習得か未習得かの判断基準や意味についてはこのモデルの中では定義されない. *Objective Progress Status* が True でない場合, この値は意味を持たない.	論理型	False
3	*Objective Measure Status*	学習目標が習得度を持つことを示す (真または偽) .	論理型	False
4	*Objective Normalized Measure*	-1..1 の間 (境界値を含む) に正規化された学習目標の習得度 (例えば標準化された得点) . 習得度を正規化するメカニズムについてはこのモデルでは定義されない. *Objective Measure Status* が True でない場合, この値は意味を持たない.	実数型 [-1.0..1.0]の有効数字 4 桁の実数	0.0

学習目標進捗情報の記述はデータモデルペアを使用する. つまり, 一つの要素はトラッキングされたデータを記述し, もう片方はトラッキングされたデータが有効かどうかを記述する. 例えば, *Objective Satisfied Status* は学習目標が達成されたかどうかを記述し, *Objective Progress Status* は *Objective Satisfied Status* の値が有効かどうかを記述している. 詳細なシーケンシング動作はデータモデルペアの両方の値を参照する. トラッキングモデルはランタイムデータモデルなので, 実装者はこれらの値を自由に表現してシステムを最適化することができる. しかし, システムは標準シーケンシング動作擬似コード (付録C参照) で表された動作を実現しなければならない.

本書では, シーケンシング動作の記述をより読みやすくするために, 各ペアを説明するのに一つの要素だけを使用し, 「unknown」を設定可能な値として追加する. 例えば, *Objective Satisfied Status* は以下の語彙を使って記述される:

- **satisfied** – *Objective Progress Status* が True で, *Objective Satisfied Status* が True
- **not satisfied** – *Objective Progress Status* が True で, *Objective Satisfied Status* が False
- **unknown** – *Objective Progress Status* が False

同様に, 通常の浮動小数点実数の範囲の値に加えて, *Objective Normalized Measure* の値を「unknown」と記述することにより, *Objective Measure Status* が False に設定されていることを示す.

IMS SS 仕様はローカル学習目標と共有グローバル学習目標の学習目標進捗情報を区別している. デフォルトでは, アクティビティの各試行毎に初期化される各学習目標の学習目標進捗情報は, そのアクティビティに対してローカルである. しかし, ローカル学習目標進捗情報と共有グローバル学習目標進捗情報は, シーケンシング定義モデル要素 *Objective Map* によって対応付けられる. シーケンシングプロセスが

学習目標の状態にアクセスするときどんな情報が引き出されるかは，どの学習目標進捗情報を使うかによって定義される．

1. アクティビティが追跡されていない（*Tracked*が偽の）場合，LMS はトラッキング情報を管理しない．そのため，学習目標進捗情報を要求すると「未知」（デフォルトトラッキング状態）が呼び出される．
2. *Objective Satisfied Status*が要求され，ローカル学習目標と共有グローバル学習目標をリンクする *Read Objective Satisfied Status* 学習目標マップが定義されていて、共有グローバル学習目標の *Objective Progress Status*が True に設定されている場合，共有グローバル *Objective Satisfied Status* が呼び出される．
3. *Objective Satisfied Status*が要求され，ローカル学習目標と共有グローバル学習目標をリンクする *Read Objective Satisfied Status* 学習目標マップが定義されていないか、リンクされた共有グローバル学習目標の *Objective Progress Status*が False に設定されている場合，ローカル *Objective Satisfied Status*が呼び出される．
4. *Objective Normalized Measure*が要求され，ローカル学習目標と共有グローバル学習目標をリンクする *Read Objective Normalized Measure* 学習目標マップが定義されていて、共有グローバル学習目標の *Objective Measure Status*が True に設定されている場合，共有グローバル *Objective Normalized Measure*が呼び出される．
5. *Objective Normalized Measure*が要求され，ローカル学習目標と共有グローバル学習目標をリンクする *Read Objective Normalized Measure* 学習目標マップが定義されていないか、リンクされた共有グローバル学習目標の *Objective Measure Status*が False に設定されている場合，ローカル *Objective Normalized Measure*が呼び出される．

ADL ノート: アクティビティが追跡されている（*Tracked*が真）場合，SCO が起動される前に，LMS はアクティビティの学習目標進捗情報に格納されている情報を使って SCO ランタイム環境データ (cmi.objectives)を初期化する．SCORM RTE ブック[4]の 4.2.17 学習目標セクションは，これがどのように行われるかについて記述している．

シーケンシングプロセスにおいて，学習目標進捗情報が共有グローバル学習目標から取り出された場合，ローカル学習目標は変更されない．

アクティビティへの試行が終了すると，「write」目標マップが考慮される．*Write Objective Satisfied Status* かつ/もしくは *Write Objective Normalized Measure*が True の場合 (既定値 False)，学習目標進捗情報の該当する部分（*Objective Progress Status*と *Objective Satisfied Status* かつ/もしくは *Objective Measure Status*と *Objective Normalized Measure*）が，ローカル学習目標から対応付けられた共有グローバル学習目標へコピーされる．共有グローバル学習目標のすべての学習目標進捗情報は無条件で上書きされる．

4.2.1.3 アクティビティ進捗情報

各アクティビティは，そのアクティビティのすべての試行にまたがったトラッキング状態情報を持つ．この情報は，表 4.2.1.3a で定義されるアクティビティ進捗情報である．アクティビティ進捗情報の表に記述される要素は，様々なシーケンシングプロセスにおいて，シーケンシング動作擬似コード（付録C参照）から参照される．アクティビティ進捗情報の表では各要素の既定値が定義されている．LMS のシーケンシング実装により要素が明示的に設定されるまで既定値が使用される．

表 4.2.1.3a: アクティビティ進捗情報

No.	要素	記述	値空間	既定値
1	*Activity Progress Status*	アクティビティのアクティビティ進捗情報が意味をもつことを示す．	論理型	False

2	*Activity Absolute Duration*	アクティビティに対するすべての試行の累計期間，すなわち，アクティビティの開始から終わりまでの時間. 期間を決めるメカニズムはこのモデルでは定義されない. *Activity Progress Status* が True でない場合，この値は意味を持たない	精度 0.1 秒の期間	0.0
3	*Activity Experienced Duration*	アクティビティに対するすべての試行の累計学習期間，すなわち，アクティビティが中断されているとき（アクティビティが学習されていないかアクティブでないとき）は除いたもの. 期間を決めるメカニズムはこのモデルでは定義されない. *Activity Progress Status* が True でなくかつアクティビティが葉でない場合，この値は意味を持たない	精度 0.1 秒の期間	0.0
4	*Activity Attempt Count*	アクティビティの試行回数. 回数は現在の試行を含む. すなわちゼロはアクティビティが試行されなかったことを意味し，1 以上は，試行が継続中もしくは終了したことを意味する. *Activity Progress Status* が True でない場合，この値は意味を持たない.	非負整数型	0

アクティビティ進捗情報要素は以下のように管理される：

- アクティビティへの最初の試行が開始されるとき *Activity Progress Status* が True に設定される
- *Activity Absolute Duration* はアクティビティのすべての試行の合計の絶対期間である.

  ADL ノート: SCORM シーケンシングは期間に基づくシーケンシング情報（例：ほとんどの制限コンディションといくつかのルールアクション）の評価を要求しない. 従って，LMS はこの要素を管理することを要求されていない. 要素が値を持っている場合，その値がシーケンシング動作に影響を与えることはない.
- *Activity Experienced Duration* はアクティビティへのすべての試行の総学習期間である. この値は葉でないアクティビティに対してはトラッキングされない.

  ADL ノート: SCORM シーケンシングは期間に基づくシーケンシング情報の評価を要求しない（例：ほとんどの制限コンディションやいくつかのルールアクション）. 従って，LMS はこの要素を管理することを要求されていない. 要素が値を持っている場合，その値がシーケンシング動作に影響を与えることはない.
- アクティビティの新たな試行が開始されると *Activity Attempt Count* は増加する.

4.2.1.4 試行進捗情報

アクティビティの各試行において，学習者に，表 4.2.1.4a に定義されている試行進捗情報が割り当てられる. 試行進捗情報の表に記述されているこの要素は，様々なシーケンシングプロセス中に，シーケンシン

グ動作擬似コード(付録C参照)で参照される. 試行進捗情報の表は各要素に対して既定値を定義する. LMS のシーケンシング実装により要素が明示的に設定されるまで, 既定値が使用される.

表 4.2.1.4.a: - 試行進捗情報

No.	要素	説明	値空間	既定値
1	*Attempt Progress Status*	アクティビティの試行に対して試行進捗情報に意味があることを示す(真または偽).	論理型	False
2	*Attempt Completion Amount*	アクティビティの試行完了についての度合い. 0 から 1 (境界値を含む) の間に正規化され, 1 は試行が完了していることを意味し, それより小さい値は試行が終了していないことを意味する. 完了度合を定義するメカニズムはこのモデルでは定義されない. *Attempt Progress Status* が True でない場合, この値は意味を持たない.	実数型[0..1]有効数字4桁の実数	0.0
3	*Attempt Completion Status*	試行が完了していることを示す(真または偽). 完了または非完了の判断基準や意味はこのモデルでは定義されない. *Attempt Progress Status* が True でない場合, この値は意味を持たない.	論理型	False
4	*Attempt Absolute Duration*	アクティビティの試行期間, すなわち, 試行開始から終了までの時間. 期間を決定するメカニズムは本モデルでは定義されない. *Attempt Progress Status* が True でない場合, この値は意味を持たない.	精度 0.1 秒の期間	0.0
5	*Attempt Experienced Duration*	アクティビティの試行における学習期間すなわち, 試行開始から終了までの時間で, アクティビティが中断されているとき(アクティビティが学習されていないかアクティブでないとき)は除いたもの. 期間および中断時間を決定するメカニズムは本モデルでは定義されない. *Attempt Progress Status* が True でない場合, この値は意味を持たない.	精度 0.1 秒の期間	0.0

試行進捗情報要素は以下の通り管理される:

- *Attempt Progress Status* はアクティビティの現在の試行において，他のトラッキング情報が記録される場合 true に設定される．
- *Attempt Completion Status* は，アクティビティの現在の試行が完了したかどうかを示す．
- *Attempt Completion Amount* は，アクティビティの現在の試行の完了度合を示す．現バージョンの IMS SS 仕様は *Attempt Completion Amount* を使用しない．

  ADL ノート: SCORM はこの要素に対して他のいかなる動作も定義しない．定義されたシーケンシング動作(付録 *C: シーケンシング動作擬似コード*参照)に準拠する限り，実装においてこの要素を自由に使用することができる．
- *Attempt Absolute Duration* は，アクティビティの現在の試行の総絶対時間である．

  ADL ノート: SCORM シーケンシングは期間に基づくシーケンシング情報の評価を要求しない．従って，LMS はこの要素を管理することを要求されていない．要素が値を持った場合，その値がシーケンシング動作に影響を与えることはない．
- *Attempt Experienced Duration* は，アクティビティの現在の試行の総学習期間である．

  ADL ノート: SCORM シーケンシングは期間に基づくシーケンシング情報の評価を要求しない．従って，LMS はこの要素を管理することを要求されていない．要素が値を持った場合，その値がシーケンシング動作に影響を与えることはない．

試行進捗情報の記述はデータモデルペアを使用する．つまり，一つの要素はトラッキングされたデータを記述し，そしてもう片方はトラッキングされたデータが有効かどうかを記述する．例えば，*Attempt Completion Status* は試行が完了したか否かを記述し，*Attempt Progress Statu* は *Attempt Completion Status* の値が有効かどうかを記述している．詳細なシーケンシング動作はデータモデルペアの両方の値を参照する．トラッキングモデルはランタイムデータモデルなので，実装者はこれらの値を自由に表現してシステムを最適化することができる．しかし，システムは標準シーケンシング動作擬似コード（付録C参照）で表された動作を実現しなければならない．

本書では，シーケンシング動作の記述をより読みやすくするために，各ペアを説明するのに一つの要素だけを使用し，「unknown」を設定可能な値として追加する．例えば，*Attempt Completion Status* は以下の語彙を使って記述される：

- **completed** – *Attempt Progress Status* が True で，*Attempt Completion Status* が True
- **incomplete** – *Attempt Progress Status* が True で，*Attempt Completion Status* が False
- **unknown** – *Attempt Progress Status* が False

同様に，定義された期間タイプに加えて，*Attempt Progress Status* が False に設定されていることを示すために，*Attempt Absolute Duration* と *Attempt Experienced Duration* の値を「unknown」と記述する．

4.2.1.5 アクティビティ状態情報

規定されたシーケンシング動作を行なうために，LMS は，アクティビティツリーの各アクティビティに対して，学習者毎に追加の状態情報を保持しなければならない．この情報は，図 4.2.1.5a で定義されているアクティビティ状態情報と呼ばれ，この情報はアクティビティが追跡されているか否かに依らず存在する．アクティビティ状態情報の表で記述された要素は，様々なシーケンシングセッション中に，シーケンシング動作擬似コード（付録C参照）で参照される．アクティビティ状態情報の表は各要素の既定値を定義する．既定値は，LMS のシーケンシング実装により要素が明示的に設定されるまで使用される．

表 4.2.1.5a: アクティビティ状態情報

No.	要素	説明	値空間	既定値
1	*Activity is Active*	アクティビティの試行が現在進行中であることを示す．すなわち，アクティビティが学習者に配信されて終了されていない，または，アクティビティが *Current Activity* の祖先である，のいずれかであることを示す（真または偽）．	論理型	False
2	*Activity is Suspended*	アクティビティが現在中断されていることを示す（真または偽）．	論理型	False
3	*Available Children*	アクティビティの実行可能な子アクティビティの順序付けを示すリスト．	アクティビティの順序付きリスト	All children

アクティビティ状態情報要素は以下のように管理される：

- アクティビティへの試行が開始されると *Activity is Active* が True に設定され，終了すると False に設定される．ある学習者と特定のアクティビティツリーに対して，この要素は様々なシーケンシングプロセスにおいていくつかの特性と効果を持つ：
    - どのタイミングでも，アクティビティツリーには，一つの「active path」しか存在しない．つまり，「active path」上のアクティビティだけが True の *Activity is Active* を持てる．「active path」はツリーのルートから始まり，*Current Activity* で終わる（4.2.1.6:グローバル状態情報参照）．
    - どのタイミングでも，一つの（もしくはゼロの）葉アクティビティだけが True の *Activity is Active* を持つことができる．葉アクティビティが True の *Activity is Active* をもつ場合，そのアクティビティは *Current Activity* でなければならない．
    - まだ終了していない場合，*Current Activity* は True の *Activity is Active* を持つ．つまり，現試行が終了していない．
- アクティビティへの現在の試行が終了する際，*Activity is Suspended* が True に設定されることがある．これはアクティビティの種類により二つのうち一つの方法で行われる．
    - アクティビティが葉の場合，アクティビティに関連付けられたコンテンツオブジェクトもしくは LMS が，アクティビティのコンテンツオブジェクトが中断状態で終了したことを示すことがある．
    - アクティビティが親クラスタの場合，子のいずれかが中断されたら，LMS のシーケンシング実行がそのクラスタを中断に設定する．

    アクティビティの中断状態は，そのアクティビティへ次の試行がどのように行われるかについて記述する．アクティビティが中断された場合，アクティビティへの次の試行は前回の試行を再開し前回のトラッキングモデル状態を使用する．新たなトラッキングモデルは初期化されない．関連付けられたアクティビティの中断状態に SCO がどのように影響するかについての詳細は，SCORM RTE ブック[4]のセクション 4.2.8：*Exit* を参照のこと．
- *Available Children* は，シーケンシング動作実行中に，LMS のシーケンシング実装が使用可能なアクティビティの子の順序付きリストを保持する．この要素は，クラスタのアクティビティにだけ適用される．この要素は，ナビゲーション動作（Navigation Behavior），終了動作（Termination Behavior），ロールアップ動作（Rollup Behavior），シーケンシング動作（Sequencing Behavior）および配信動作（Delivery Behavior）において，シーケンシングの対象となる子の集合として使用される．

ADL ノート: LMS のシーケンシング実装は, 作成時に定義されたアクティビティの順序付きリストをクラスタに対して保持しなくてはならない. このリストは選択ランダム化動作中に *Available Children* 要素を決定するために使用される.

4.2.1.6 グローバル状態情報

LMS のシーケンシング実装は, アクティビティツリーに関する追加の状態情報を保持する. この情報は, 図 4.2.1.6a で定義されており, グローバル状態情報と呼ばれる. グローバル状態情報の表で記述された要素は, 様々なシーケンシングセッション中に, シーケンシング動作擬似コード (付録C 参照) で参照される. グローバル状態情報の表は各要素の既定値を定義する. LMS シーケンシング実装により要素が明示的に設定されるまで既定値が使用される.

表 4.2.1.6a: グローバル状態情報

No.	要素	説明	値空間	既定値
1	*Current Activity*	*Current Activity* を示す. 学習者がアクティビティを実行している場合, *Current Activity* は最後に完了した *Content Delivery Environment Process* によって配信されたアクティビティである. 学習者がアクティビティを実行していない場合, *Current Activity* は最後に完了した *Terminate Request Process* によって終了するよう指定されたアクティビティ, ないし, 最後に成功した *Choice Sequencing Request Process* で配信できなかった対象アクティビティである.	アクティビティ	None
2	*Suspended Activity*	*Suspend All* ナビゲーション要求を引き起こす元になったアクティビティを示す.	アクティビティ	None

グローバル状態モデル要素は以下のように管理される:

- *Current Activity* は, アクティビティツリーで LMS のシーケンシング実装によってトラッキングされる固有のアクティビティである. このアクティビティは, ナビゲーション動作, 終了動作, およびシーケンシング動作中, 最初に検討されるアクティビティである. このアクティビティの全ての祖先は True に設定された *Activity is Active* を持たなければならない. *Current Activity* は全てのシーケンシング要求が処理される位置を定義している.

  ADL ノート: 様々なシーケンシング要求プロセス (付録 C: シーケンシング動作擬似コード参照)で, シーケンシング要求処理が単一の *Current Activity* 要素から開始されると定義されているが, 実装は唯一の要素だけを使用するよう要求されていない. 実装はシーケンシング動作擬似コードで記述された標準動作に準拠すればよい, つまり, 実装は一つの要素だけを使用しているかのように見えればよい. 例えば, 実装は, 直近に配信されたアクティビティを *Current Activity* としてトラッキングしようとし, 直近に終了したアクティビティを *First Candidate Activity* (最初の候補アクティビティ) としてトラッキングしてもよい. そしてシーケンシング要求プロセスとして *First Candidate Activity* を開始アクティビティとして使用してもよい.

- *Suspended Activity* は，前回のシーケンシングセッション（例：*Suspend All* 要求のために終了したシーケンシングセッション）で *Current Activity* であった固有のアクティビティを示す．アクティビティツリーのルートから *Suspended Activity* へ到る経路の全てのアクティビティの *Activity is Suspended*（セクション 4.2.1.5：アクティビティ状態情報参照）は True に設定される．後続のシーケンシングセッションは *Resume All* ナビゲーション要求で開始され，経路上の *Suspended Activity* を含む全てのアクティビティを再開する．これは単純な形の「bookmarking」と考えられる．

図 4.2.1.6a は，*Current Activity* の状態モデルを図示し，様々なシーケンシングプロセスの *Current Activity* および *Current Activity* の状態に対する影響をまとめている．

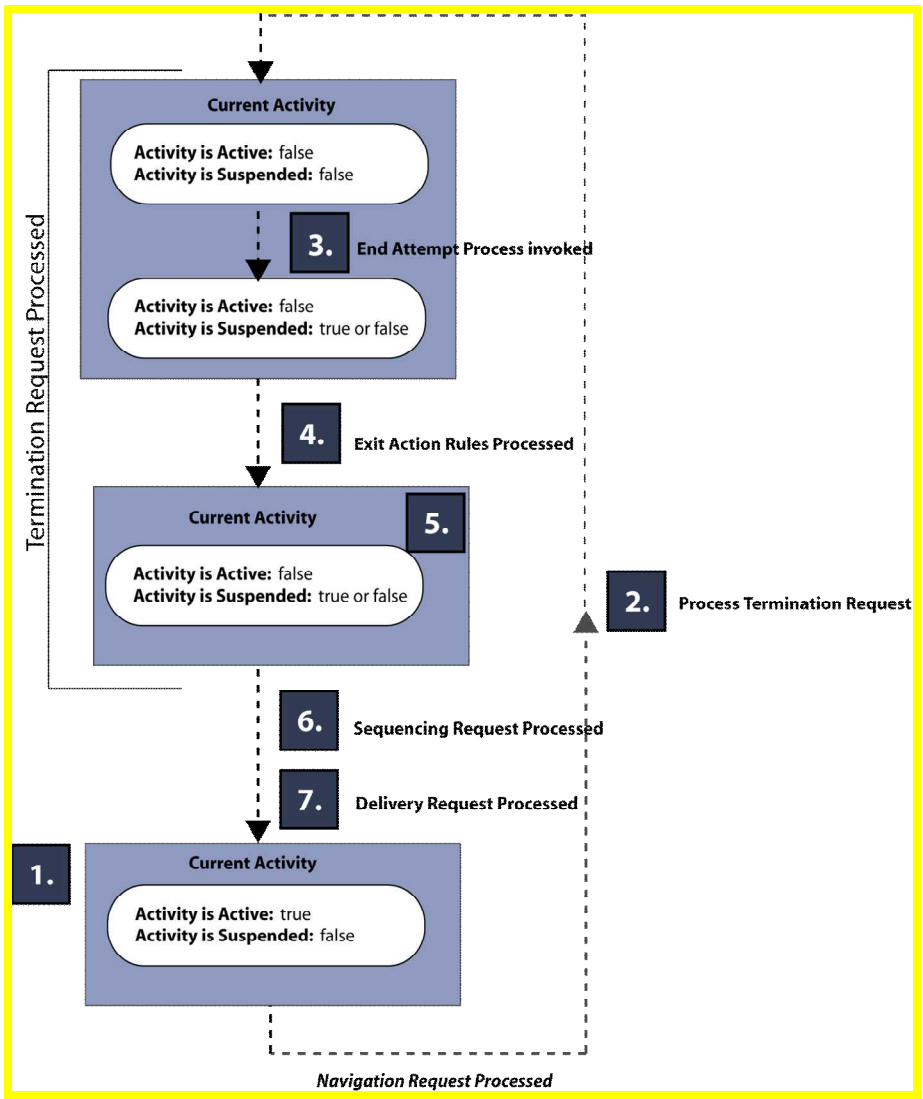

図 4.2.6.1a Current Activity 状態モデル

1. LMS のシーケンシング実装は，直近に（配信用に）指定されたアクティビティに対応付けられたコンテンツオブジェクトが学習者に対して起動されたと仮定する．配信されたアクティビティは *Current Activity* である．これはアクティビティツリーの葉でなければならず，現在アクティブであ

る. アクティビティの試行はアクティビティの祖先の試行の中で発生するので, *Current Activity* の全ての (「active path」上の) 祖先は同様にアクティブである.

2. 学習者もしくはコンテンツから始動されるナビゲーション要求によって LMS がオーバーオールシーケンシングプロセス (*Overall Sequencing Process*) を呼び出すまでの間, アクティビティの状態はこのように留まる. ナビゲーション要求が有効であれば, ナビゲーション要求動作により, *Current Activity* への試行を終了する終了要求(*Exit*)が得られる. トラッキング情報は更新され, アクティビティ状態情報要素 *Activity is Active* が False となり, オーバーオールロールアッププロセス (*Overall Rollup Process*) が呼び出される. *Current Activity* はまだ変わらない.
3. 葉アクティビティへの試行が終了するとき, コンテンツは学習者セッションが中断状態で終了したと示すことがある. これは, 試行終了プロセス (*End Attempt Process*) で行われる. システムがこれを示せば, *Current Activity* のアクティビティ状態情報要素 *Activity is Suspended* が True となる. コンテンツオブジェクトの学習者試行状態は対応付けられた学習アクティビティの状態と同期する.
4. 終了動作中, *Current Activity* の全ての祖先で終了アクションシーケンシング (*Sequencing Exit Action*) ルールが評価される. これは, 終了アクションルールシーケンシングサブプロセス (*Sequencing Exit Action Rule Subprocess*) で行われる. このサブプロセスの結果, 終了したばかりの葉アクティビティが *Current Activity* にままとなるか, もしくは, 葉アクティビティの先祖が *Current Activity* になる. 祖先が *Current Activity* になれば, その祖先への現在の試行は試行終了プロセス (*End Attempt Process*) によって終了する. 祖先の *Activity is Active* は False となり, オーバーオールロールアッププロセス (*Overall Rollup Process*) が呼び出される.
5. 終了動作中, *Current Activity* が中断していなければ, *Current Activity* のポストコンディションアクションルール (*Post Condition Action Rules*) だけが評価される. ここで *Current Activity* は葉アクティビティか終了アクションルールサブプロセス (*Exit Action Rule Subprocess*) (Step #4 参照)中で指定されたアクティビティである.
6. シーケンシング動作は, 全ての保留中のシーケンシング要求を処理し, これによって配信要求が得られることがある.
7. 配信動作プロセスはすべての保留している配信要求を有効にする. 配信要求が有効になると, コンテンツ配信環境プロセス (*Content Delivery Environment Process*) が呼び出される. このプロセス中, 指定されたアクティビティが *Current Activity* となり, 試行が開始 (再開) される. *Current Activity* のアクティビティ状態属性の *Activity is Active* が true となり, *Activity is Suspended* が false となる
8. Step #1 へ戻る

4.2.1.7 トラッキング動作

このセクションの情報は, IMS SS 仕様のトラッキングモデル動作セクションを (置き換えるのではなく) 補完するものである. より詳細は IMS SS 仕様を参照のこと.

SCORM LMS は, 学習目標進捗情報を管理するとき, 以下の要求に従わなければならない:

1. シーケンシング定義モデルの学習目標記述要素を使用してアクティビティに対して定義された全ての学習目標は, 各学習者について, アクティビティへの各試行のたびにローカル学習目標進捗情報の組を割り当てられる.
2. 共有グローバル学習目標は, 各学習者について, 定義された範囲を割り振られた学習目標進捗情報の組を持つ.
3. 「read」学習目標マップが定義され, 共有グローバル学習の状態が既知の時, 共有グローバル学習目標進捗情報がシーケンシングおよびロールアップルールを評価するために使用される. 「read」学習目標マップが定義ていないとき, ローカル学習目標進捗情報がシーケンシングおよびロールアップルールを評価するために使用される.

4. 複数のアクティビティが同一の学習目標進捗情報を参照できるようにするためには，情報にアクセスしようとするすべてのアクティビティの学習目標マップ（*Objective Map*）が定義されなければならない．参照される学習目標進捗情報は「共有グローバル」学習目標と呼ばれる．各学習目標マップは，ローカル学習目標と共有グローバル学習目標との関係を定義する．

   学習目標マップには 2 種類ある：

   - **write** 学習目標マップ（***write*** *Objective Map*）は，共有グローバル学習目標要素を対応するローカル学習目標の値に設定する．ローカル学習目標進捗情報が変更されるごとに **write** 学習目標マップが適用される．**write** 学習目標マップは，アクティビティがアクティブな間に複数回適用されることがあるが，最低一回，試行が終了する際には適用されなければならない．

     ADL ノート: 同一アクティビティに関連する複数のローカル学習目標は，情報を同じ共有グローバル学習目標に **write** することができない．これは予期せぬ動作を生じさせる．

   - LMS のシーケンシング実装が，ローカル学習目標要素を要求したとき，read 学習目標マップにより共有グローバル学習目標要素が読みだされる．

     ADL ノート: ローカル学習目標は共有グローバル学習目標の一つからのみ **read** 可能である．そうでなければ，予期せぬ動作を生じさせる．

5. ローカル学習目標の *Objective Satisfied by Measure* が True の場合，学習目標の Objective Satisfied Status にアクセスすると，Objective Normalized Measure の *Objective Minimum Satisfied Normalized Measure* 閾値に対する比較結果だけが使用される．この評価は，すべての設定されたローカルもしくは共有グローバル Objective Satisfied Status を使用する代わりに実行される．

ADL ノート: 定義された習得度しきい値（*Objective Satisfied by Measure*）の評価の評価は，アクティビティの状態およびアクティビティの *Measure Satisfaction If Activey* 要素の値に依存する．アクティビティがアクティブ（*Activity is Active* が真）で *Measure Satisfaction If Activey* が偽の場合，習得度しきい値の評価を行ってはならない．アクティビティの習得状態は未定としなくてはならない．

このケースでは，LMS は評価結果の Objective Satisfied Status をアクティビティのローカル学習目標進捗情報に保存することがある．それによってローカル Objective Measure は変わることはない．

4.3. オーバーオールシーケンシングプロセス

このセクションの情報は, IMS SS 仕様のトラッキングモデル動作セクションを置き換えるのではなく, 補完するものである. 詳細について IMS SS 仕様を参照のこと.

オーバーオールシーケンシングプロセス (*Overall Sequencing Process*) は, LMS のシーケンシング実装の全般的なコントロールプロセスを提供し, シーケンシングセッションの中で, 様々なシーケンシング動作がどのように適用されるかについて記述している. オーバーオールシーケンシングプロセスは, 以下のシーケンシング動作を含む:

- **ナビゲーション動作** –ナビゲーション要求の有効性をどのように確かめ, それを終了要求およびシーケンシング要求に変換するかについて記述している.
- **終了動作** –アクティビティへの現在の試行がどのように終了するか, アクティビティツリーの状態がどのように更新されるか, 試行終了によってどのようなアクションが実行されるかについて記述している.
- **ロールアップ動作** –クラスタアクティビティのトラッキング情報がどのように子アクティビティから導出されるかについて記述している.
- **選択ランダム化動作** – シーケンシング要求処理時に, クラスタのアクティビティがどのように検討されるのかについて記述している.
- **シーケンシング動作** – 次に配信されるアクティビティを特定するために, シーケンシング要求がどのようにアクティビティツリーで処理されるかについて記述している.
- **配信動作** – 配信するよう特定されたアクティビティの有効性をどのように確認し, LMS が確認されたアクティビティの配信をどのように行うのかについて記述している.

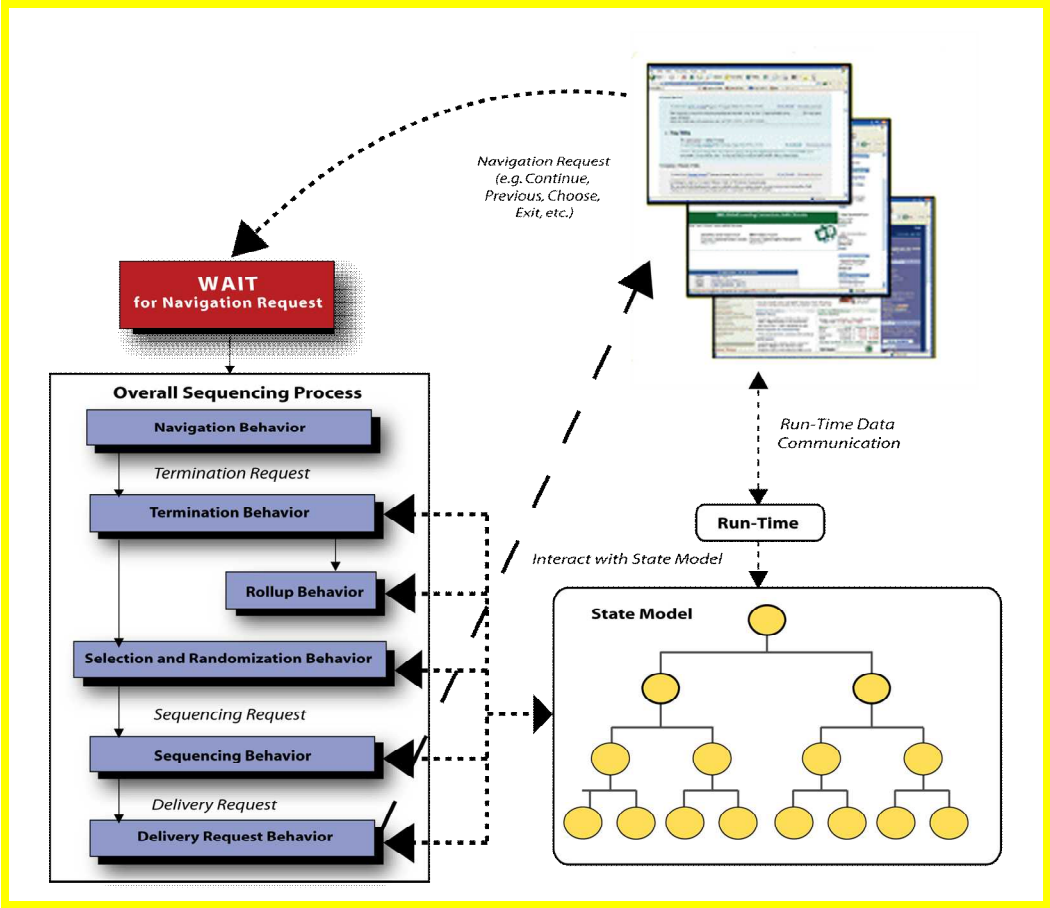

図 4.3a - オーバーオールシーケンシングプロセスの概念モデル

IMS SS 仕様は上記の動作を，同一データモデル（トラッキングモデル（セクション 4.2 参照））の上で動作する独立したスタンドアロンプロセスの集合として記述している．オーバーオールシーケンシングプロセスの概念モデル（図 4.3a）は，様々なシーケンシング動作，アクティビティツリー，トラッキングモデルの相互の関係を示している．オーバーオールシーケンシング動作への入口は LMS が発行するナビゲーション要求である．典型的には，ナビゲーション要求は，学習者によるナビゲーションイベントないしはコンテンツが発行するナビゲーション要求（セクション 4.4.3：ナビゲーション要求参照）から発行される．オーバーオールシーケンシング動作からの出口は，次に配信されるアクティビティの指定（何も配信しないも含む）か，例外処理である．アクティビティが配信のために指定された場合，LMS はアクティビティと関連付けられたコンテンツオブジェクトを起動する．SCORM は，配信されるアクティビティが指定されないケース，もしくは例外が発生したケースの動作については定義していない．オーバーオールシーケンシング動作結果について何らかの情報を LMS が学習者に提供し，これによってシーケンシングセッションを継続させるようにすることを推奨している．

オーバーオールシーケンシングプロセスは，シーケンシングセッションが開始もしくは終了する際，シーケンシングループの動作を実現する．シーケンシングループのステップの詳細を以下に記述する．

4.3.1. シーケンシングループ

シーケンシングセッション開始

(1) 学習者がLMSへのアクセスを開始し(例:システムへアクセス, ログイン等), ある学習ユニット内の文脈を確立する(例:コース, コンテンツオーガニゼーションなどを選択する)

(2) LMSが, *Start*, *Resume All*もしくは*Choice*ナビゲーション要求を発行してシーケンシングプロセスを開始する. *
前のシーケンシングセッションが*Resume All*ナビゲーション要求で終了した場合, LMSは*Resume All*ナビゲーション要求でシーケンシングセッションを開始しなくてはならない.
*Choice*ナビゲーション要求が利用可能であっても, LMSは*Start*ないし*Resume All*ナビゲーション要求でシーケンシングセッションを開始しなくてはならない.
*シーケンシングセッションを開始する前に, カレントアクティビティは*None*(未定義)とみなさなくてはならない.

(3) ナビゲーション動作は, *Start*, *Resume All*, もしくは*Choice*ナビゲーション要求を適切なシーケンシング要求に変換して処理する. 配信されるアクティビティが特定されたとき, シーケンシングセッションが正式に開始される. すなわち, 以下のシーケンシングループに通じる成功するパスである.
シーケンシングセッションが開始されたら, 中断アクティビティは*None*(未定義)とみなさなくてはならない.

シーケンシングループ開始

(4) シーケンシング動作は, シーケンシング要求に基づき, トラッキング状態モデルとシーケンシング定義モデルの情報を使用して, 学習者へ配信すべき適切なアクティビティを定めるようにアクティビティツリーを探索する. 配信されるアクティビティが特定できない場合, オーバーオールシーケンシングプロセスは停止し, 他のナビゲーション要求を待つ. Step #9へ飛ぶ.

(5) 配信動作は, 特定されたアクティビティが配信できるか否かを決定し, 配信できる場合, アクティビティに関連付けられたコンテンツオブジェクトを学習者に対して起動する準備をする. 指定されたアクティビティが配信できない場合, オーバーオールシーケンシングプロセスは停止し, 他のナビゲーション要求を待つ. Step #9へ飛ぶ.

(6) 学習者がコンテンツオブジェクトとインタラクションを行う. インタラクション中, シーケンシングプロセスはアイドリング状態で要求を待っている.

(7) インタラクション中にコンテンツオブジェクトが, 様々なトラッキングモデル要素を更新する値を通信することがある.

(8) 学習者, コンテンツオブジェクトもしくはシステムが, *Continue*, *Previous*, *Choose* activity X, *Abandon*, *Exit*などのナビゲーションイベントを呼び出す.

(9) LMSは, ナビゲーション要求を発行することによりシーケンシング実装にナビゲーションイベントを知らせる.

(10) ナビゲーション動作は, ナビゲーション要求を終了要求とシーケンシング要求に変換する. 学習者がアクティビティツリーのルートアクティビティへの試行を終了したいというナビゲーション要求を示した場合, シーケンシングセッションは終了する. (シーケンシングセッションを終了する動作とアクティビティ状態モデルの保存方法については定義されず, LMSの実装に任されている)

(11) コンテンツオブジェクトの終了に伴うナビゲーション要求が発行された場合, トラッキングモデルを更新する値が通信されることがある. アクティビティへの試行が終了する. 学習者とコンテンツオブジェクトとのインタラクションにより発生した状態変化の影響を決定するためにロールアップ動作が呼び出される. ロールアップ動作はアクティビティとアクティビティツリー内の祖先に対してトラッキング状態モデルを更新する.

(12) シーケンシングループは, シーケンシングセッションが終わるまで, ステップ 4から繰り返す**.
**シーケンシングセッションが終わったあとは, カレントアクティビティは*None*(未定義)とみなさなくてはならない.

オーバーオールシーケンシング動作は，様々なシーケンシングプロセスがどのように関係するのかを示すが，実装は，オーバーオールシーケンシングプロセスの文脈外であれば個々のシーケンシングプロセスを，いつでも自由に呼び出すことができる．その場合，アクティビティとアクティビティツリーのトラッキング状態モデルが，ナビゲーション要求を処理する際のオーバーオールシーケンシングプロセスで記述された動作を確実に実現するため，充分な状態管理を提供する必要がある．オーバーオールシーケンシングプロセス外でシーケンシングプロセスを呼び出す通常のシナリオは，学習リソースを起動する結果になるナビゲーション要求だけを確実に実行し，有効なナビゲーションコントロールだけを含む知的ユーザインターフェースを提供することである．

実装において，様々なナビゲーションイベントに対する「what if」シナリオを評価するために，仮のデータ上のオーバーオールシーケンシングプロセスの文脈外で（複数のトラッキング情報のセットを管理する，追加のトラッキング状態モデル要素を提供する，ロールバックを使用する等々によって），シーケンシング動作を呼び出すことができる．例えば，学習者に *Continue* ナビゲーションイベントを発行させる何らかのナビゲーションコントロールを提供するように実装することが可能である．学習者がそのコントロールを実行した場合，LMS は，何が起こるか確認するため，自由に仮の *Continue* ナビゲーション要求を処理することができる．仮の要求がエラーに終わるか，もしくは何も配信しない結果になった場合，LMS はナビゲーションイベントを無効とし（オーバーオールシーケンシングプロセスを呼び出さず），学習者に提示し，仮の状態データを破棄することができる．仮の要求がうまく行った場合，LMS はオーバーオールシーケンシングプロセスを呼び出し（あるいはその何らかのサブセットを最適化し），うまく行ったコンテンツオブジェクトを配信し，仮状態データを生かすことができる．

4.4. ナビゲーション動作

ナビゲーション動作はオーバーオールシーケンシングプロセスへの主要な入口である．学習者とシステムの意図を LMS のシーケンシング実装に反映する手段を提供する．ナビゲーションの意図を表わす外部イベントはナビゲーションイベントと呼ばれる．これらのイベントを発行する手段はナビゲーションコントロールと呼ばれる．LMS はナビゲーションイベントを処理し，ナビゲーションイベントに対応するナビゲーション要求でシーケンシング実装を呼び出す役割がある．

4.4.1. ナビゲーションイベント

ナビゲーションイベントは，学習者もしくはシステムが何らかの方法でコンテンツをナビゲートする意図を示す（LMS のシーケンシング実装の）外部のイベントである．これらのイベントは，通常，学習者によってユーザインターフェースコントロールを通して発行されるが，LMS も自由にナビゲーションイベントを発行ことができる．SCORM はどのようにナビゲーションイベントが発行されるかについては何も制限しない．

ナビゲーションイベントが検出されると，LMS は以下の二つのうち一つの応答を行なう：

1. イベントを無視する－LMS は，（オーバーオールシーケンシングプロセスを通じて）処理した結果，配信するものがないナビゲーションイベントについては無視しなければならない．これは，学習者には望ましくないシステム状態（経験）である．ナビゲーションイベントが配信するものがないという結果に終わることを，LMS がどのように決定するのかについて，SCORM はどのような要求もしていない．例えば，学習者が現在のアクティビティツリーの最後の葉に関連付けられたコンテンツオブジェクトを経験している場合，次の項目に進みたいと要望しても，配信するものがないという結果になる．この場合，LMS は *Continue* ナビゲーションイベントを無視しなければならない．

2. ナビゲーション要求を出す－LMS は，ナビゲーションイベントを対応するナビゲーション要求に転換し，オーバーオールシーケンシングプロセスを呼び出す．

4.4.2. ナビゲーションコントロール

ナビゲーションコントロールは，学習者がある特定の方法で *Current Activity* から抜け出すための要求を送るためのユーザインターフェース装置である．SCORM は，*Continue*, *Previous* および *Choice* ナビゲーション要求によって学習者に対して配信されるコンテンツが得られる場合，少なくともこれらの要求を発生するナビゲーションコントロールを LMS が提供することを要求する．さらに SCORM は，*Continue*, *Previous* および *Choice* ナビゲーション要求によって疑似コード例外が発生する場合，学習者がナビゲーションコントロールによって活動を中断するナビゲーション要求を発生することができるのであれば，これらの要求を発生するナビゲーションコントロールを LMS が提供しないことを要求する．SCORM は，ナビゲーションコントロールがどのように表示されるか，どのように発行されるか，どのナビゲーションイベントが発行されるか，について定義していない．

SCORM は，コンテンツがコンテンツ中でナビゲーションコントロールを提供していることをコンテンツ開発者が特定する方法（`<adlnav:presentation>`）を提供している．この場合，LMS はコンテンツの要求を尊重し，冗長で混乱を招く可能性のあるユーザインターフェースを提供しないよう求められる．

ナビゲーションコントロールに関する追加の情報は SCORM ナビゲーションモデル (セクション 5:SCORM ナビゲーションモデル参照) に記述されている。

4.4.3. ナビゲーション要求

ナビゲーション要求が LMS シーケンシング実装に発行されると, オーバーオールシーケンシングプロセスが始まる. 一旦ナビゲーション要求が出されたなら, シーケンシング擬似コード (付録C参照) で定義されている動作を適用しなければならない. オーバーオールシーケンシングプロセスが開始される.

SCORM 対応 LMS は, 表 4.4.3a で定義されている以下のナビゲーション要求を受け入れ, 対応する動作を実現しなければならない.

表 4.4.3a: SCORM 2004 ナビゲーション要求

ナビゲーション要求	アクション
Start	*Current Activity* が未定義なら, *Start* シーケンシング要求を発行する.
Resume All	*Current Activity* が未定義で *Suspended Activity* が定義されていれば, *Resume All* シーケンシング要求を発行する.
Continue	*Current Activity* の *Activity is Active* が *True* ならば, *Exit* 終了要求を発行する. *Continue* シーケンシング要求を発行する.
Previous	*Current Activity* の *Activity is Active* が *True* ならば, *Exit* 終了要求を発行する. *Previous* シーケンシング要求を発行する.
Forward	このバージョンの SCORM では定義されない.
Backward	このバージョンの SCORM では定義されない.
Choice	*Current Activity* の *Activity is Active* が *True* ならば, *Exit* 終了要求を発行する. *Choice* シーケンシング要求を発行する. 要求はターゲットアクティビティの指定を伴う.
Exit	*Exit* 終了要求を発行する. *Exit* シーケンシング要求を発行する. *Current Activity* の現在の試行は正常に終了する. アクティビティの終了は他の外部ナビゲーションイベント(*Continue, Previous, Choice*)によるものではない.
Exit All	*Exit All* 終了要求を発行する. *Exit* シーケンシング要求を発行する.
Suspend All	*Suspend All* 終了要求を発行する. *Exit* シーケンシング要求を発行する. *Current Activity* とその全ての先祖の現在の試行は正常に終了する. 試行は終了せず, アクティビティは完結しない. アクティビティは将来のある時点で再開さ

	れる可能性がある（再開は新しい試行ではない）. LMS のシーケンシング実装は, アクティビティを将来再開させるために, 必要な状態・トラッキング情報を記録しなければならない.
Abandon	*Abandon* 終了要求を発行する. *Exit* シーケンシング要求を発行する. *Current Activity* の現在の試行は異常終了し, アクティビティは完了しない. アクティビティ試行は再開されない. トラッキングデータはロールバックされない.
Abandon All	*Abandon All* 終了要求を発行する. *Exit* シーケンシング要求を発行する. *Current Activity* とその全ての先祖の現在の試行は異常終了し, アクティビティは完了しない. アクティビティ試行は再開されない. トラッキングデータはロールバックされない.

4.4.4. ナビゲーション要求プロセス

このセクションの情報は, IMS SS 仕様のトラッキングモデル動作セクションを置き換えるのではなく, 補完するものである. 詳細については IMS SS 仕様を参照のこと. 実装は, IMS SS 仕様に記述されている擬似コードではなく, シーケンシング動作擬似コード（付録C参照）に記述されている標準動作を実現するように要求される.

ナビゲーション要求プロセス（*Navigation Request Process*）は, オーバーオールシーケンシングプロセス中に起動されるが, LMS によって直接起動される場合もある. ナビゲーション要求プロセスは, ナビゲーション要求を受け入れ, 例外, シーケンシング要求, ないし, 終了要求とシーケンシング要求を返す.

例外が返されるのは以下の場合である:

- 定義されていない(表 4.4.3a にない)ナビゲーション要求が発行される.
- *Start* ナビゲーション要求が発行されたが, シーケンシングセッションが既に始まっている.
- *Resume All* ナビゲーション要求が発行されたが, シーケンシングセッションが既に始まっている. シーケンシングセッションは始まっていないが, *Suspended Activity* が存在しない（おそらく前のセッションが *Suspend All* ナビゲーション要求によって終了していない）.
- *Continue* ナビゲーション要求が発行されたが, *Current Activity* の親のシーケンシングコントロールモード *Flow* が True でない.
- *Previous* ナビゲーション要求が発行されたが, *Current Activity* の親のシーケンシングコントロールモード *Flow* が True でないか, *Current Activity* の親のシーケンシングコントロールモード *Forwad Only* が False でない.
- *Choice* ナビゲーション要求が発行されたが:
    - *Choice* ナビゲーション要求のターゲットがアクティビティツリーに存在しない. これは, ターゲットアクティビティがツリーに存在しないか, ターゲットアクティビティがターゲットの親の *Available Children* の一員でない場合である(セクション 4.7: 選択ランダム化動作参照).
    - *Choice* ナビゲーション要求のターゲットの親のシーケンシングコントロールモード *Choice* が False である.
    - *Choice* ナビゲーション要求のターゲットに対する *Choice* シーケンシング要求を処理する際,（アクティビティパスに沿った）アクティブなアクティビティが終了する必要があり,

かつ, そのアクティビティのシーケンシングコントロールモード *Choice Exit* が False である.

有効な *Continue*, *Previous*, *Choice* ナビゲーション要求は, 対応するシーケンシング要求に帰着する. さらに, 有効な *Continue*, *Previous*, *Choice* ナビゲーション要求が発行され, *Current Activity* がアクティブである場合, *Current Activity* の現在の試行を終了するために *Exit* 終了要求が発行される.

Exit, *Exit All*, *Suspend*, *Abandon*, *Abandon All* ナビゲーション要求は, 対応する終了要求と *Exit* シーケンシング要求に帰着する. これらのナビゲーション要求は, *Current Activity* の試行を終了し, 場合によってはシーケンシングセッションを終了する.

4.5. 終了動作

終了動作は二つの目的を持っている：*Current Activity* の現在の試行を終了すること，そして，アクティビティツリーの状態を，最新の有効な状態しておくことの2つである．終了動作は終了要求に基づいて動作する．終了動作は *Current Activity* を動かし，シーケンシング要求を返すことがある．

アクティビティが終了することと，アクティビティに関連付けられたコンテンツオブジェクトが終了することを区別するのは重要である．いつ，どのようにアクティビティに関連付けられたコンテンツオブジェクトが終了するかについては SCORM の対象外である．SCORM は，SCO が終了する前に(Terminate()を要求することによって)通信を終了することだけを要求している．コンテンツオブジェクトの終了の詳細については SCORM RTE ブックのセクション 2.1：ランタイム環境管理(RTE) [4]を参照のこと．アクティビティの終了は内部のシーケンシング表現および動作の一部であり，コンテンツオブジェクトの終了に影響されたり影響を与えたりすることはない．

より具体的には，*Current Activity* は，アクティブな場合，終了要求により終了する．シーケンシング要求の処理をする際にアクティビティツリーが最新の有効な状態にあるように，LMS のシーケンシング実装は *Current Activity* の終了を確実に行わなければならない．しかし，LMS のシーケンシング実装が最新の有効な状態の情報を確実に取得するため，アクティビティの終了によって，関連付けられたコンテンツオブジェクトを強制的に終了しなくてはならないことが（LMS の実装によっては）ありうる．

4.5.1. 終了要求

通常，終了要求は *Current Activity* の現在の試行が終了しなければならない，つまり，*Current Activity* が非アクティブにならなければならないことを示している．IMS SS 仕様は数種類の終了要求を定義し，それぞれの終了要求は異なった動作に帰着する．SCORM 対応 LMS はこれらの動作（表 4.5.1a）を実現する．

表 4.5.1a: SCORM 2004 終了要求

終了要求	アクション
Exit	*Current Activity* の現在の試行が正常に終了する．試行は終了する．
Exit Parent	*Current Activity* の親の現在の試行が正常に終了する．試行は終了する．
Exit All	すべてのアクティブなアクティビティ（ルートから *Current Activity* までの全アクティビティ）の現在の試行が正常に終了する．試行は終了する．
Suspend All	すべてのアクティブなアクティビティ（ルートから *Current Activity* までの全アクティビティ）の現在の試行が中断する．*Current Activity* の試行は再開される可能性がある．
Abandon	*Current Activity* の現在の試行が異常終了する．アクティビティは完了しない．試行は再開されない．トラッキングデータはロールバックされない．
Abandon All	すべてのアクティブなアクティビティ（ルートから *Current Activity* までの全アクティビティ）の現在の試行が異常終了する．アクティビティは完了しない．放棄されたアクティビティへの試行は再開されない．トラッキングデータはロールバックされない．

4.5.2. ポストコンディションと終了アクションルールの評価

アクティビティは, 一つ以上のポストコンディションおよび終了アクションシーケンシングルールをもつことができる. シーケンシングルールの構造は以下のとおり:

If [*condition set*] **Then** [*action*]

[*condition set*] は, アクティビティのトラッキング情報に対して個別に評価されるコンディションの集合である. 各コンディションは, 否定される (Rule Condition 演算子が「Not」である) 場合もある一つの値を [*condition set results*]に提供する. ルール評価の単一の結果 (true / false / unknown)を決めるために, コンディションコンビネーションが[*condition set results*]の中の値に適用される. そのルール評価の結果が true であれば, ルール [*action*]が適用される.

いくつかのトラッキングモデル要素はペアで記述される. つまり, 一つは状態データを記述し, もう一つはその状態データの有効性を記述する. これらの要素の評価を含むシーケンシングルールは, 基になるトラッキング情報が無効な場合, [*condition set results*]で「unknown」値を返すことがある. 「unknown」値を含むセットにルールコンディション演算子とコンディションコンビネーション (付録C : UP.2.1) を適用する方法は以下の表で定義される:

表 4.5.2a: NOT Truth Table

NOT	True	False	Unknown
	False	True	Unknown

表 4.5.2b: AND Truth Table

AND	True	False	Unknown
True	True	False	Unknown
False	False	False	False
Unknown	Unkown	False	Unknown

表 4.5.2c: OR Truth Table

OR	True	False	Unknown
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

シーケンシングルールアクションは, そのルールの評価のタイミング, そのルールをどのシーケンシングプロセスを適用するか, そして, そのルールのプロセスへの影響, に, 通常対応する 3 つのセットに分類される. 2 種類のシーケンシングルールアクション, Post Condition および Exit は終了動作中に適用される. 終了アクションシーケンシングルールは, 終了動作の終了アクションルールシーケンシングサブプロセス

(Sequencing Exit Action Rule Subprocess) の中でのみ評価される. ポストコンディションシーケンシングルールは, 終了動作のポストコンディションルールシーケンシングサブプロセス (Sequencing Post Condition Rules Subprocess) の中でのみ評価される.

例:

- **If not satisfied Then retry** –アクティビティの学習目標状態が not satisfied であれば, アクティビティで Retry シーケンシング要求を処理する.
- **If attempted Then exit parent** – アクティビティが試行されていれば, このアクティビティの親を Exit する.
- **If attempted Then exit all** – アクティビティが試行されていれば, アクティビティツリーを Exit し現在のシーケンシングセッションを終了する.

上記の例は, アクティビティに定義されているシーケンシングルールの種類の一部に過ぎない.

ADL ノート: コンテンツ開発者は, ポストコンディションルールが *Current Activity* でだけ評価されるとことを忘れてはいけない. 意図したシーケンシング戦略において, ポストコンディションアクションをクラスタアクティビティに適用する必要がある場合, クラスタアクティビティはポストコンディションルールが評価される前に, 終了アクションルールによって明示的に終了されていなければならない.

4.5.3. 終了要求プロセス

このセクションの情報は, IMS SS 仕様のトラッキングモデル動作セクションを置き換えるのではなく, 補完するものである. 詳細については IMS SS 仕様を参照のこと. 実装は, IMS SS 仕様に記述されている擬似コードではなく, シーケンシング動作擬似コード (付録C参照) に記述されている標準動作を実現するように要求される.

終了要求プロセス (*Termination Request Process*) は, シーケンシング要求の処理に先立って, *Current Activity* の試行を終了するために, オーバーオールシーケンシングプロセス (*Overall Sequencing Process*) によって呼び出される. *Current Activity* の現在の試行は以下の 3 つのうち1つの方法で終了する:

- Normal Termination – *Exit* もしくは *Exit All* 終了要求によって生じる. *Current Activity* に対応付けられたコンテンツオブジェクトはアクティビティのトラッキング情報に影響を与える. 終了要求が *Exit All* の場合, シーケンシングセッションが終了する.
- Abnormal Termination – *Abandon* もしくは *Abandon All* 終了要求によって生じる. *Current Activity* に対応付けられた学習アクティビティはアクティビティのトラッキング情報に影響を与えない. 終了要求が *Abandon All* の場合, シーケンシングセッションが終了する.
- Suspended – *Suspend All* 終了要求によって生じる. *Current Activity* とその全ての祖先の試行は中断され, シーケンシングセッションが終了する. これは, 中断した試行を *Resume All* ナビゲーション要求によって再開し, シーケンシングセッションを後から開始することを意図している. 学習者はシーケンシングセッションを *Current Activity* の学習で開始する.

シーケンシングセッション中, 最も一般的な終了要求は *Exit* である. 終了要求プロセス (*Termination Request Process*) は *Exit* 終了要求中に以下のアクションを実行する:

試行終了プロセス (End Attempt Process) 中

(1) *Current Activity* の現在の試行を終了する(*Current Activity* の *Activity is Active* を False に設定する).

(2) アクティビティに対応付けられたコンテンツオブジェクトがアクティビティのトラッキング情報に影響を与える状態情報を送信する.
(3) アクティビティに対応付けられたコンテンツオブジェクトが状態情報を送信しない場合, LMS のシーケンシング実装はアクティビティのトラッキング情報を satisfied および completed に適宜設定する.
(4) *Current Activity* の現在の試行が正常に終了した場合, 試行は「suspended」(*Activity is Suspended* が True)状態で終了する可能性がある. アクティビティに対応付けられたコンテンツオブジェクトはこの状態を通知する.
(5) ロールアップが実行される – *Current Activity* のトラッキング情報が *Current Activity* の祖先を通してアクティビティツリーを伝達される.

終了アクションルールシーケンシングサブプロセス (Sequencing Exit Action Rules Subprocess) 中

(6) *Current Activity* の祖先の一つに終了アクションルールを定義することによって, 祖先の現在の試行を終了し, ロールアップを実行し, そして, その祖先を *Current Activity* にすることができる.

ポストコンディションルールシーケンシングサブプロセス (Sequencing Post Condition Rules Subprocess) 中

(7) *Current Activity* が中断されていない場合, *Current Activity* のポストコンディションルールが評価される. これらのルールは *Current Activity* の祖先を終了させることがある(*Exit Parent* および *Exit All* ルール), もしくはシーケンシング要求(*Continue*, *Previous* および *Retry* ルール)を発生することがある. *Current Activity* の祖先が終了すると, 祖先は *Current Activity* になり, そのポストコンディションルールが評価される (これは再帰的なオペレーションである). シーケンシング要求が発行されたら, 要求はオーバーオールシーケンシングプロセス (*Overall Sequencing Process*) へ返され, それによって保留されているすべてのシーケンシング要求が上書きされる.

IMS SS 仕様は, *Current Activity* の開始点に関する情報を維持せずに *Current Activity* を動かす「絶対」オペレーションとして, 終了動作を定義している. シーケンシング動作 (セクション 4.8 参照) は全ての処理を *Current Activity* から始めるので, この終了動作が要求される. 様々なシーケンシングプロセスは, *Current Activity* の現在の試行が既に終了しており, アクティビティツリーが最新状態にあると仮定している. さらに, 保留中の配信要求を処理してコンテンツオブジェクトを起動し, それに対応するアクティビティの試行を開始する前に, *Current Activity* の現在の試行は終了していることを, 配信動作 (セクション 4.9 参照) は前提としている.

実装は, *Current Activity* の情報を保持し *Current Activity* の「what-if」終了を実行するための終了動作を (もしくは他のどんなシーケンシング動作も) 自由に拡張することができる. 実装は, 追加のトラッキングモデル要素, 追加の (サブ) プロセス, および, 拡張もしくは変更したサブプロセスを自由に使用することができる. SCORM 対応 LMS に対する唯一の要求は, オーバーオールシーケンシングプロセスが呼び出されたとき, IMS SS 仕様に記述された擬似コードではなく, 標準シーケンシング動作擬似コード (付録C 参照) に記述された通りに LMS が動作することである. つまり, LMS がナビゲーション要求を実行すると決定し, オーバーオールシーケンシングプロセスが呼び出されるとき, 終了動作に記述された通りに, 保留中のシーケンシング要求の処理に先立って *Current Activity* の現在の試行が終了することである.

4.5.4. 試行終了プロセス

試行終了プロセス (*End Attempt Process*) は, アクティビティが正常に終了するときに呼び出されるユーティリティプロセスである. このプロセスにより, 終了するアクティビティの状態が更新, 情報のアクティビティツリー全体への伝達が行なわれる. 試行終了プロセスは *Current Activity* を変えることはない.

試行終了プロセス (*End Attempt Process*) 中に以下の動作が発生する:

- 終了するアクティビティが葉で, 対応付けられたコンテンツオブジェクトが SCO の場合, SCO は最新の学習者試行が「suspended」状態で終了したと通知することがある. つまり, SCO は cmi.exit を suspend に設定する. LMS のシーケンシング実装は, アクティビティの現在の試行を「suspend」するためにこの情報を使用する.
- アクティビティがクラスタの場合, その子のいずれかが「suspended」であれば, クラスタ自身も「suspended」となる.

  ADL ノート: アクティビティへの試行は, 親 (先祖) アクティビティの試行の中で起こるので, アクティビティツリーの葉のどれかが中断されると, アクティビティツリーのルートも中断される. この状況では, *start* ナビゲーション要求はアクティビティツリーのルートへの新しい試行ではなく, 前回の試行を再開する結果になる.

- 終了するアクティビティが葉で, 対応付けられたコンテンツオブジェクトが SCO の場合, 表 4.5.4a に詳述されたデータマッピングが, End Attempt Process 疑似コードの 1.1 行 (付録 C: UP.4) の後で, 記述された順序に即座に実行される. データマッピングに関する詳細情報は, SCORM RTE ブックの該当するセクションを参照のこと[4].

  ADL ノート: カレントアクティビティの試行が, Abandon ないし Abandon All ナビゲーション要求で終了した場合, 表 4.5.4a のデータマッピングは生じない. 放棄された試行は End Attempt Process を起動せず, アクティビティの状態に影響しない.

  ADL ノート: SCO が終了するとき, ロールアップに寄与する学習目標 (主学習目標) は, 二つの可能な情報源を有する. SCO の学習目標集合(`cmi.objectives.xxx`)と, `cmi.success_status` および `cmi.score.scaled` データ要素である. SCO が (`SetValue()`呼び出しで) 一方のデータ要素のみ提供した場合, そのデータがアクティビティのロールアップに寄与する学習目標にマップされる. SCO が双方の情報を提供する場合, `cmi.success_status` および `cmi.score.scaled` データ要素のデータがアクティビティのロールアップに寄与する学習目標にマップされる.

表 4.5.4a ランタイムデータのシーケンシングトラッキングデータへのマッピングの要約

SCORM ランタイム環境データモデル要素		シーケンシングトラッキングデータモデル要素
1.	`cmi.objectives.n.success_status`	cmi.objectives.n.id と同じ ID を有するアクティビティの学習目標の Objective Progress Status および Objective Satisfied Status
	`unknown`	Objective Progress Status = false Objective Satisfied Status = false
	`failed`	Objective Progress Status = true Objective Satisfied Status = false
	`passed`	Objective Progress Status = true Objective Satisfied Status = true
2.	`cmi.objectives.n.score.scaled`	cmi.objectives.n.id と同じ ID を有するアクティビティの学習目標の Objective Normalized Measure
	`unknown`	Objective Measure Status = false Objective Normalized Measure = 0.0
	`Defined between -1.0 to 1.0`	Objective Measure Status = true Objective Normalized Measure = the defined value

3.	cmi.success_status	アクティビティのロールアップに寄与する学習目標 (主学習目標) の Objective Progress Status および Objective Satisfied Status
	unknown	Objective Progress Status = false Objective Satisfied Status = false
	failed	Objective Progress Status = true Objective Satisfied Status = false
	passed	Objective Progress Status = true Objective Satisfied Status = true
4.	cmi.score.scaled	アクティビティのロールアップに寄与する学習目標 (主学習目標) の Objective Normalized Measure
	unknown	Objective Measure Status = false Objective Normalized Measure = 0.0
	Defined between -1.0 to 1.0	Objective Measure Status = true Objective Normalized Measure = the defined value
5.	cmi.completion_status	Attempt Completion Status
	unknown	Attempt Progress Status = false Attempt Completion Status = false
	incomplete	Attempt Progress Status = true Attempt Completion Status = false
	completed	Attempt Progress Status = true Attempt Completion Status = true
	not attempted	Attempt Progress Status = true Attempt Completion Status = false

- 終了するアクティビティが葉で, コンテンツオブジェクトが以下の対応する値を提供しない場合, LMS のシーケンシング実装は, ロールアップ学習目標を satisfied, 完了進捗 (completion progress) を completed に設定することがある. これは LMS のシーケンシング実装によって, アクティビティの配信コントロールの値に基づいて自動的に行われる.
- 終了するアクティビティがアクティブではなくなる(*Activity is Active* が False に設定される).
- オーバーオールロールアッププロセスが呼び出される.

4.6. ロールアップ動作

トラッキングモデル（セクション 4.2 参照）で定義された通り，トラッキング状態情報が各アクティビティの各試行に対応付けられている．アクティビティツリーの各葉アクティビティは，アクティビティに対応付けられたコンテンツオブジェクトと学習者とのインタラクションをトラッキングする．SCO は，アクティビティのトラッキング状態情報に影響を与える状態情報を伝達することがある．アセットは状態情報を伝達しない．このようなコンテンツオブジェクトに対応付けられたアクティビティにシーケンシング情報を適用するには，*Objective Set by Content* を False，*Completion Set by Content* を False に設定する．この場合，LMS のシーケンシング実装は，対応するアクティビティのトラッキング状態情報を直接設定する．

クラスタアクティビティはコンテンツオブジェクトを提供できず，自身の状態情報を直接設定する手段がない．クラスタアクティビティの状態は，子アクティビティの状態に基づく．クラスタの状態情報を評価するプロセスを「ロールアップ」と呼ぶ．図 4.6a は 3 つの子アクティビティを持つクラスタを示している．この図は，このセクションで，様々なロールアッププロセスを説明するために使用される．クラスタアクティビティ（A）の状態情報は，ロールアップにより，クラスタの子アクティビティ（1，2 および 3）の状態情報から決定される．

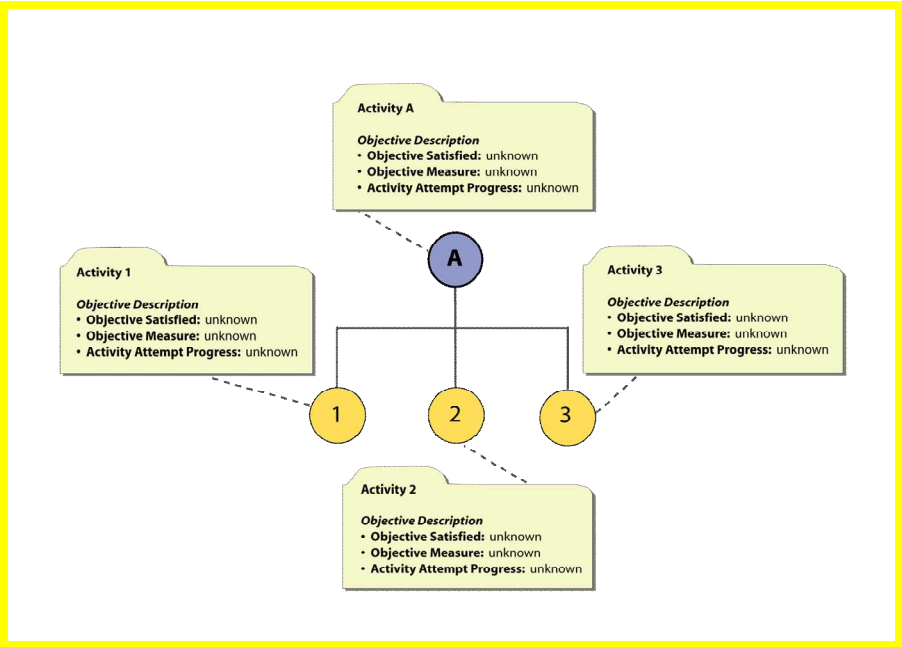

図 4.6a: ロールアップ時に使用されるアクティビティ状態情報

このセクションを通して使用される用語「ロールアップ」は，「子の状態情報に基づいてクラスタアクティビティの状態情報を決定するプロセス」を意味する．この用語は「オーバーオールロールアッププロセス（Overall Rollup Process）を適用する」ことと同義である．

4.6.1. オーバーオールロールアッププロセス

このセクションの情報は, IMS SS 仕様のロールアップ動作セクションを置き換えるのではなく, 補完するものである. 詳細については IMS SS 仕様を参照のこと. 実装においては, IMS SS 仕様に記述されている擬似コードではなく, シーケンシング動作擬似コード (付録C参照) で記述されている標準動作を実現することが要求されている.

オーバーオールロールアッププロセス (*Overall Rollup Process*) は, ロールアップが始まるアクティビティから, どのようにロールアップがアクティビティツリーに適用されるかについて記述している. 制御プロセスは, すべてのロールアッププロセスの適切な適用を確実に行なう.

オーバーオールロールアッププロセス (*Overall Rollup Process*) は, 以下の二つの異なるイベントに対応して適用しなくてはならない:

1. 試行終了プロセス (*End Attempt Process*) を通してアクティビティが終了したとき – シーケンシング動作擬似コード(付録 *C* 参照)で定義されているとおりである
2. 共有グローバル学習目標の状態が変わるとき – アクティビティが終了するときに起こる可能性がある(イベント #1 参照).

両方のケースにおいて, ロールアップを評価するプロセスは以下の通り拡張される:

A 状態の変化に影響されるすべてのアクティビティを特定する. これは, *Current Activity* および, *Current Activity*(**write** *Objective Map*)と共有グローバル学習目標を共有するいずれのアクティビティ (**read** *Objective Map*)を含む. これを「ロールアップセット」という.

B アクティビティツリーで最もレベルの深いアクティビティからオーバーオールロールアッププロセス (*Overall Rollup Process*) を始めるよう適用する.

C オーバーオールロールアッププロセス (*Overall Rollup Process*) 中, 行き当たったアクティビティをロールアップセットから取り除く.

D ロールアップセットが空になるまでステップ B および C を繰り返す

拡張ロールアッププロセス中に出現したアクティビティで共有グローバル学習目標に影響を与えるものは, ロールアップセットに対して新たなアクティビティを追加しない. ロールアップセットが決定されるのは一度だけであり, それはオーバーオールロールアッププロセス (*Overall Rollup Process*) が呼び出される前に行われる.

上記に記述された二つのイベント以外に, LMS はいつでも自由にロールアップを呼び出すことができる. LMS がオーバーオールロールアッププロセス (*Overall Rollup Process*) を呼び出した場合, LMS はアクティビティツリーのすべてのアクティビティの状態情報をトラッキングし, 対応付けられた共有グローバル学習目標が定義されたシーケンシング動作と一貫性があることを保証しなければならない. すなわち, ロールアップによって得られたトラッキング状態情報は, ロールアップが上記に定義された場合に行われた場合にのみ, "committed" (確定) されなければならないということである. LMS による仮のロールアップがアクティビティツリーのトラッキング状態情報に悪影響を及ぼさないために, 実装に際しては, トラッキング状態情報の予備セット, 「ダーティー」フラグ, ロールバック, もしくは, ロールアップ評価の原因を特定する他の方法を使用することができる.

アクティビティと他のシーケンシング動作に関連付けられたシーケンシング情報においては:

- トラッキングされた子だけがロールアップの対象となる.
- ロールアップコントロール (*Rollup Controls*) で定義されている, ロールアップに貢献する子だけがロールアップの対象となる.

- ロールアップコンシダレーションルール (Rollup Consideration rules) を満たす子だけがロールアップの対象となる.
- 対象となる子にロールアップルールの *Rollup Child Activity Set* が適用される. その結果の評価で, 状態変化があるかどうかが決定される.
- ルールの *Rollup Child Activity Set* の対象となる子の数がゼロ(評価セットが空)の場合, 状態変化は起こらない.
- ロールアップは, ロールアップの引き金となる葉アクティビティ (の状態変化) から始まり, アクティビティツリーのルートに向かって順番に行われる.
- 習得度ロールアップは常に最初に実施され, その後, 学習目標ロールアップとアクティビティ進捗ロールアップが, どのような順序でもよいが, 行なわれる.
- 習得度ロールアッププロセス (*Measure Rollup Processes*) と学習目標ロールアッププロセス (*Objective Rollup Processes*) は, 各々の子アクティビティのロールアップに関与する特定の学習目標の学習目標進捗情報だけを対象とする.
- 学習目標ロールアッププロセス (*Objective Rollup Process*) の結果は, クラスタのロールアップに関与する特定の学習目標だけに影響を与える.
- クラスタアクティビティの状態が変わらないとき, オーバーオールロールアッププロセス (*Overall Rollup Process*) は停止することができる.
- ロールアップルールは, クラスタアクティビティに対してどのようにロールアップが評価されるかを定義する.
- クラスタアクティビティの現在の状態は (もし既知であれば), *Overall Rollup Process* を起動することによって変更されない. ロールアップルールか *Measure and Objective Rollup Proces* の評価に成功した場合, アクティビティの現在の状態が変化する.
- ロールアップルールは, 葉アクティビティに定義されても効果がない. つまりロールアップするものがない.
- 習得度ロールアップは葉アクティビティには適用されない.
- ロールアップは, クラスタアクティビティのトラッキング状態値だけに影響を与える. つまりロールアップはどんなシーケンシングルール評価も起こさなければ, どんな副作用アクションも起こさない.

4.6.2. ロールアップルールの評価

クラスタアクティビティは, 一つ以上のロールアップルールを持つことができる. ロールアップルールの構造は以下のとおり:

If [*child-activity set*], [*condition set*] **Then** [*action*]

アクティビティのロールアップを評価しているとき, アクティビティの全ての子が対象となるが, アクティビティのロールアップ状態に影響を与えるのは, トラッキングされ関与する子だけである. [*condition set*]は一組のコンディションを定義しており, 各コンディションはロールアップに関与する各々の子アクティビティのトラッキング状態情報に対して評価される (付録 C: RB.1.4.2C 参照). 各コンディションは, 否定 (ロールアップコンディション演算子が「Not」である) される場合もある一つの値を提供し, この値が[*condition set results*]に反映される. [*condition set results*]に含まれる値に対して, コンディションコンビネーションが適用され, 各子アクティビティのルール評価に対する一つの回答(true / false / unknown)が決定される. [*child-activity set*]は, アクティビティの状態を変えるべきか, またどのように変えるか ([*action*] を決めるために, (あるロールアップルールの関与するひとつの子アクティビティに対して一つの) すべてのルール評価結果が, どのように使用されるのかについて記述している.

いくつかのトラッキングモデル要素はペアで記述される. つまり, 一つは状態データを記述し, もう一つはその状態データの有効性を記述する. これらの要素の評価を含むシーケンシングルールは, 基になるトラッキング情報が無効なときには, [*condition set results*]に「unknown」値を与える. ロールアップコンディシ

ョン演算子とコンディションコンビネーション (付録 C: RB.1.4.1 参照) を「unknown」値を含む値の組へ適用する方法については, 以下の表で定義されている:

表 4.6.2a: NOT Truth Table

NOT	True	False	Unknown
	False	True	Unknown

表 4.6.2b: AND Truth Table

AND	True	False	Unknown
True	True	False	Unknown
False	False	False	False
Unknown	Unkown	False	Unknown

表 4.6.2c: OR Truth Table

OR	True	False	Unknown
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

例えば:

- **If any not satisfied Then not satisfied** –クラスタの子で, トラッキングされロールアップに関与するものいずれかの学習目標の状態が *not satisfied* の場合, クラスタの学習目標の状態は *not satisfied* に設定される.
- **If 3 satisfied Then satisfied** – トラッキングされロールアップに関与する子のうち 3 つ以上の子の学習目標の状態が *satisfied* の場合, クラスタの学習目標の状態は *satisfied* に設定される.
- **If all satisfied or completed Then completed** – トラッキングされロールアップに関与するすべての子の学習目標の状態が *satisfied*, もしくはアクティビティ試行進捗状態が *completed* の場合, クラスタのアクティビティ試行進捗状態は *completed* に設定される.
- **If all satisfied and attempted Then satisfied** – トラッキングされロールアップに関与するすべての子の学習目標の状態が *satisfied* かつ試行されている場合, クラスタの学習目標の状態は *satisfied* に設定される.
- **If 50% not attempted Then incomplete** – トラッキングされロールアップに関与する子のうち 50%以上の子が試行されていない場合, アクティビティ試行進捗状態は *incomplete* に設定される.

上記の例は, 定義されているロールアップルールの種類のうちの一部に過ぎない. ロールアップルール記述に関する完全な定義はセクション 3.7 を参照のこと.

4.6.3. 習得度ロールアッププロセス

習得度ロールアッププロセス (*Measure Rollup Process*) は, クラスタの習得度を子の習得度の加重平均に設定する. 習得度ロールアッププロセスは, トラッキングされ, *Rollup Objective Satisfied* が true の子アクティビティだけを対象とする. 習得度ロールアッププロセスは, クラスタの学習目標もしくは進捗状態に直接は影響を与えないが, 学習目標ロールアッププロセス (*Objective Rollup Process*) 中に, ロールアップされた習得度に *Objective Minimum Satisfied Normalized Measure* が適用され, 学習目標の習得度状態が設定されることがある.

トラッキングされている子のいずれかのロールアップ学習目標の習得度が定義されている場合, クラスタの習得度は必ず定義される. アクティビティの習得度は, そのアクティビティの *Rollup Objective Measure Weight* を 0.0 に設定することで, 習得度ロールアップから除外することができる. 図 4.6.3a に, 習得度ロールアッププロセスの例を図示する. 破線の四角の中の情報はアクティビティに関連付けられたシーケンシング情報から得られる.

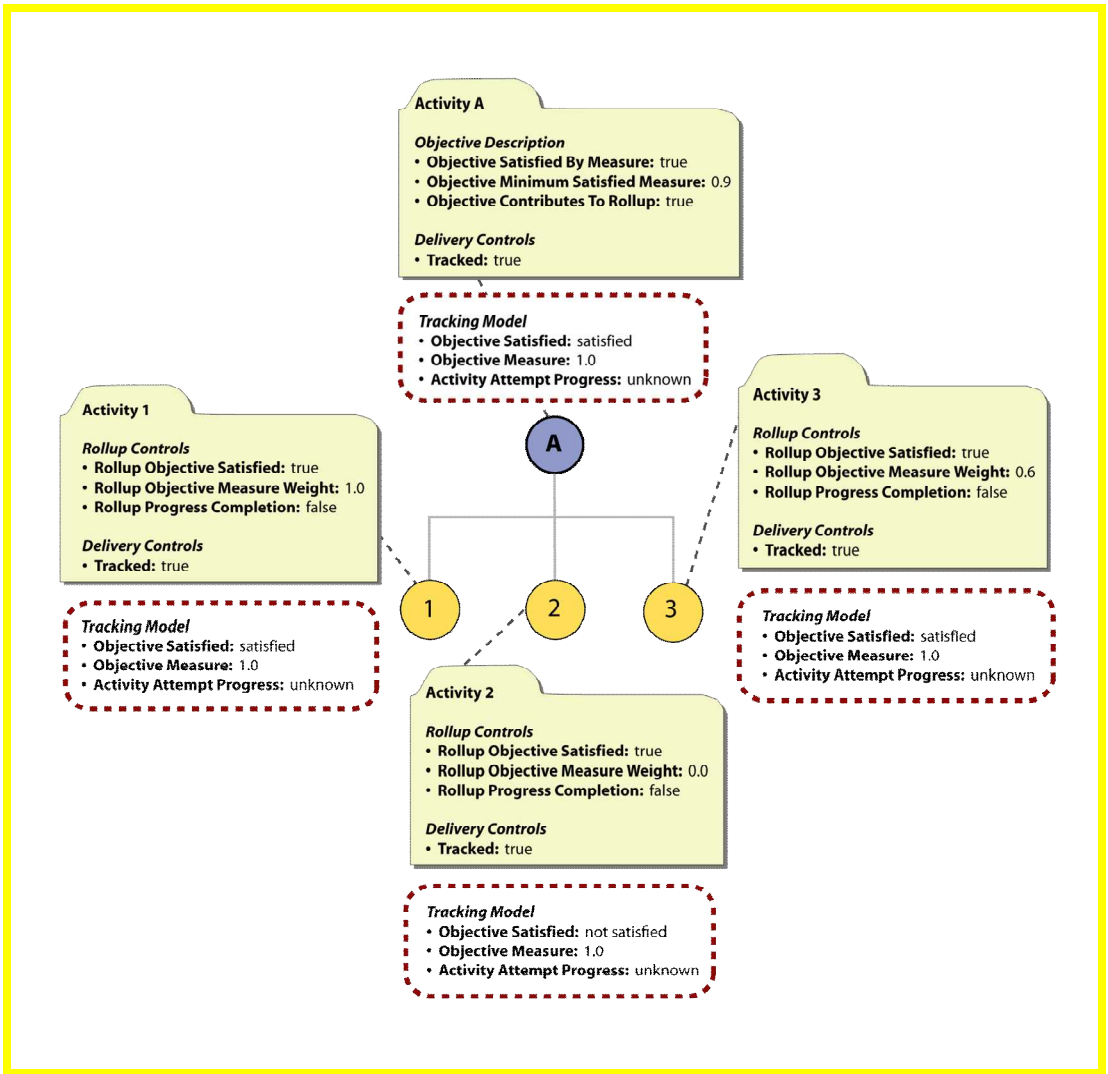

図 4.6.3a: 習得度ロールアッププロセスの例

4.6.4. 学習目標ロールアッププロセス

学習目標ロールアッププロセス (*Objective Rollup Process*) は, クラスタのロールアップ学習目標 (*Objective Contributes to Rollup* が True の学習目標) の状態を *unknown*, *satisfied* もしくは *not satisfied* に設定する. 学習目標ロールアッププロセスは, トラッキングされ, *Rollup Objective Satisfied* が true の子アクティビティだけを対象とする. 学習目標のロールアップには 3 つの方法がある. 最初に適用された方法だけがクラスタの学習目標の状態を評価するために使われる.

1. 習得度の使用 – ロールアップ学習目標の *Objective Satisfied by Measure* が true の場合, ロールアップされた習得度は *Objective Minimum Satisfied Measure* に対して比較される:

   - アクティビティがアクティブで *Measure Satisfaction if Active* が false の場合, アクティビティの状態は変化しない. そうでない場合は以下のとおり:
     - ロールアップされた習得度が unknown の場合, 学習目標状態は unknown になる.
     - ロールアップされた習得度が *Objective Minimum Satisfied Measure* と等しいかそれ以上の場合, 学習目標状態は satisfied になる.
     - ロールアップされた習得度が *Objective Minimum Satisfied Measure* より小さい場合, 学習目標状態は not satisfied になる.

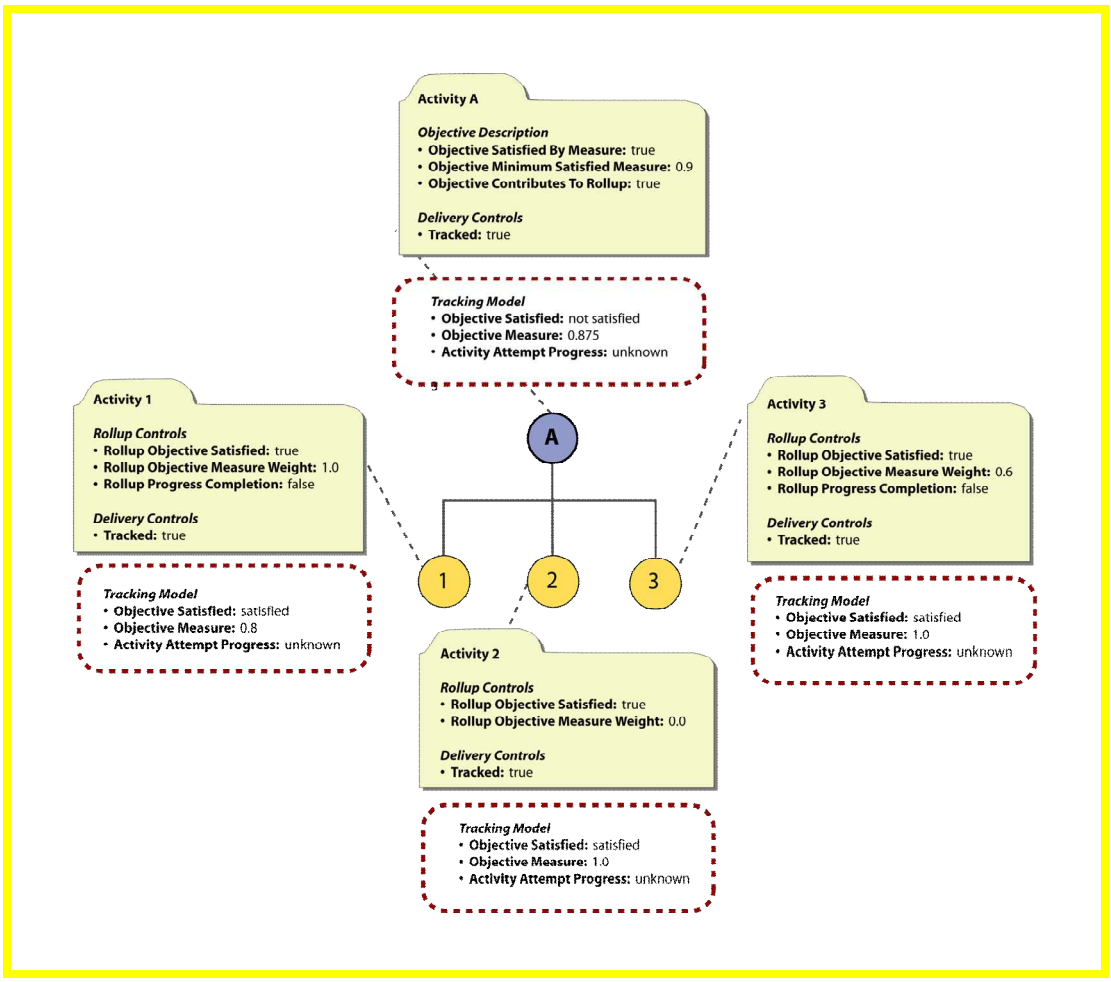

図 4.6.4a: 習得度を使用した学習目標ロールアップ

2. ルールの使用 – *satisfied* もしくは *not satisfied* アクションを持つロールアップルールがアクティビティで定義されている場合，クラスタの学習目標状態を決定するためにそれらのルールが評価される．*not satisfied* ルールが先に評価される．

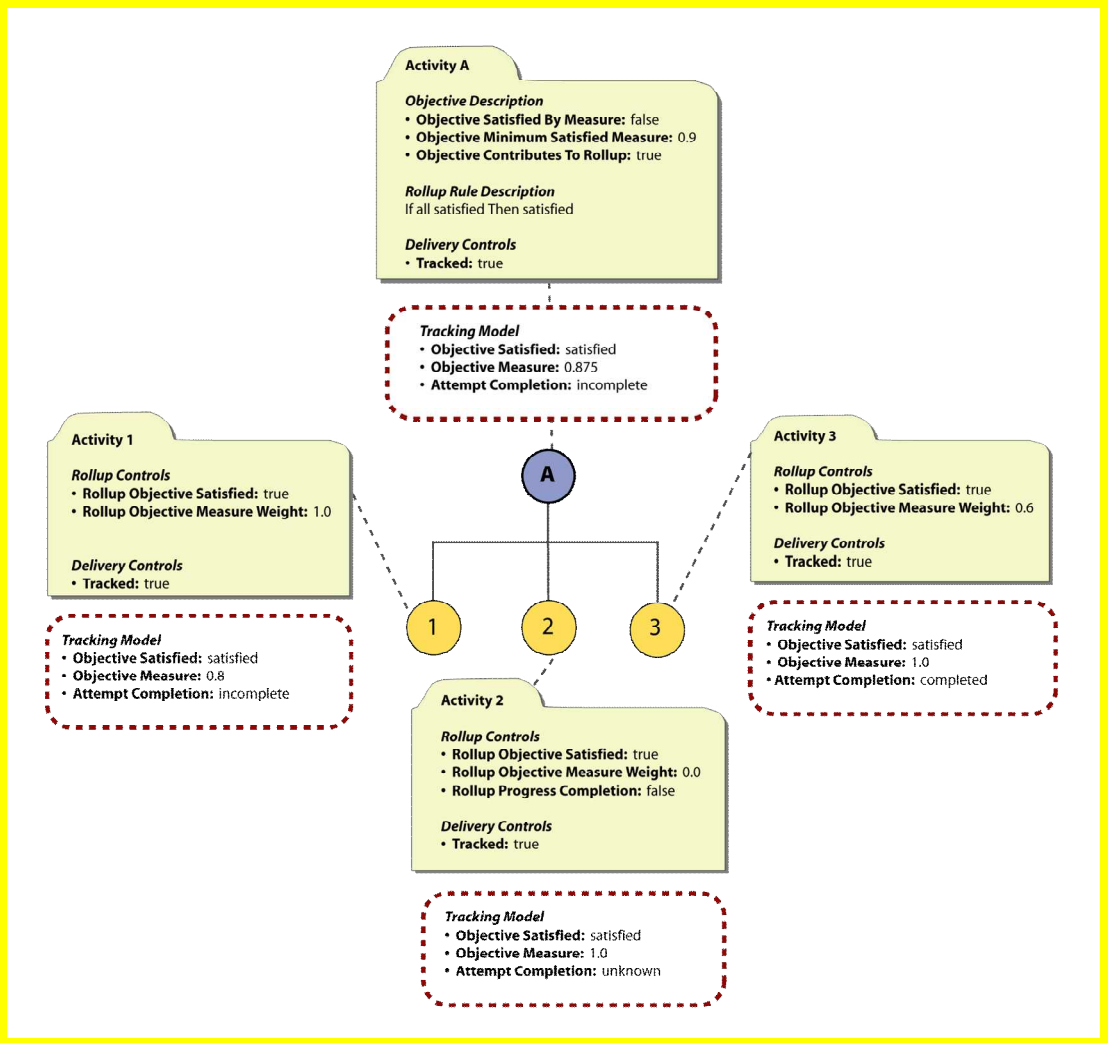

図 4.6.4b: ルールを使用した学習目標ロールアップ

3. デフォルトルール – *satisfied* もしくは *not satisfied* アクションを持つロールアップルールがアクティビティで定義されていない場合，デフォルトロールアップルールは以下のようになる：

   - If all satisfied, Then satisfied
   - If all (attempted or not satisfied), Then not satisfied

   デフォルトルールは，定義されたロールアップルールと同じ順序で評価される．つまり *not satisfied* ルールが先に評価される．

ADL ノート: ロールアップ学習目標の *Objective Satisfied by Measure* が false の場合，ロールアップされた習得度，*Objective Minimum Satisfied Measure* 要素および *Measure Satisfaction* 要素はロールアップ学習目標に影響を与えない．(デフォルト)ロールアップルールだけが適用される．

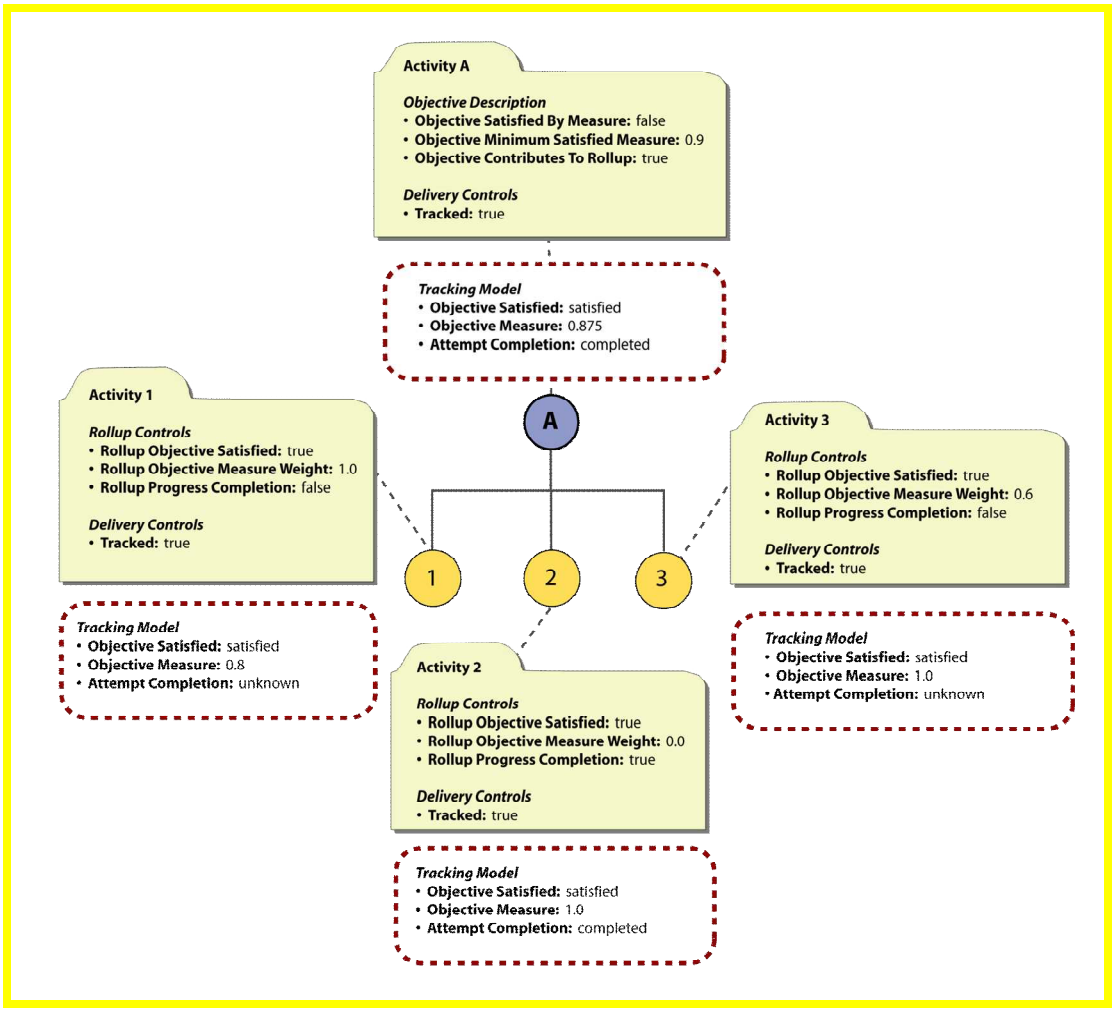

図 4.6.4c: デフォルトルールを使用した学習目標ロールアップ

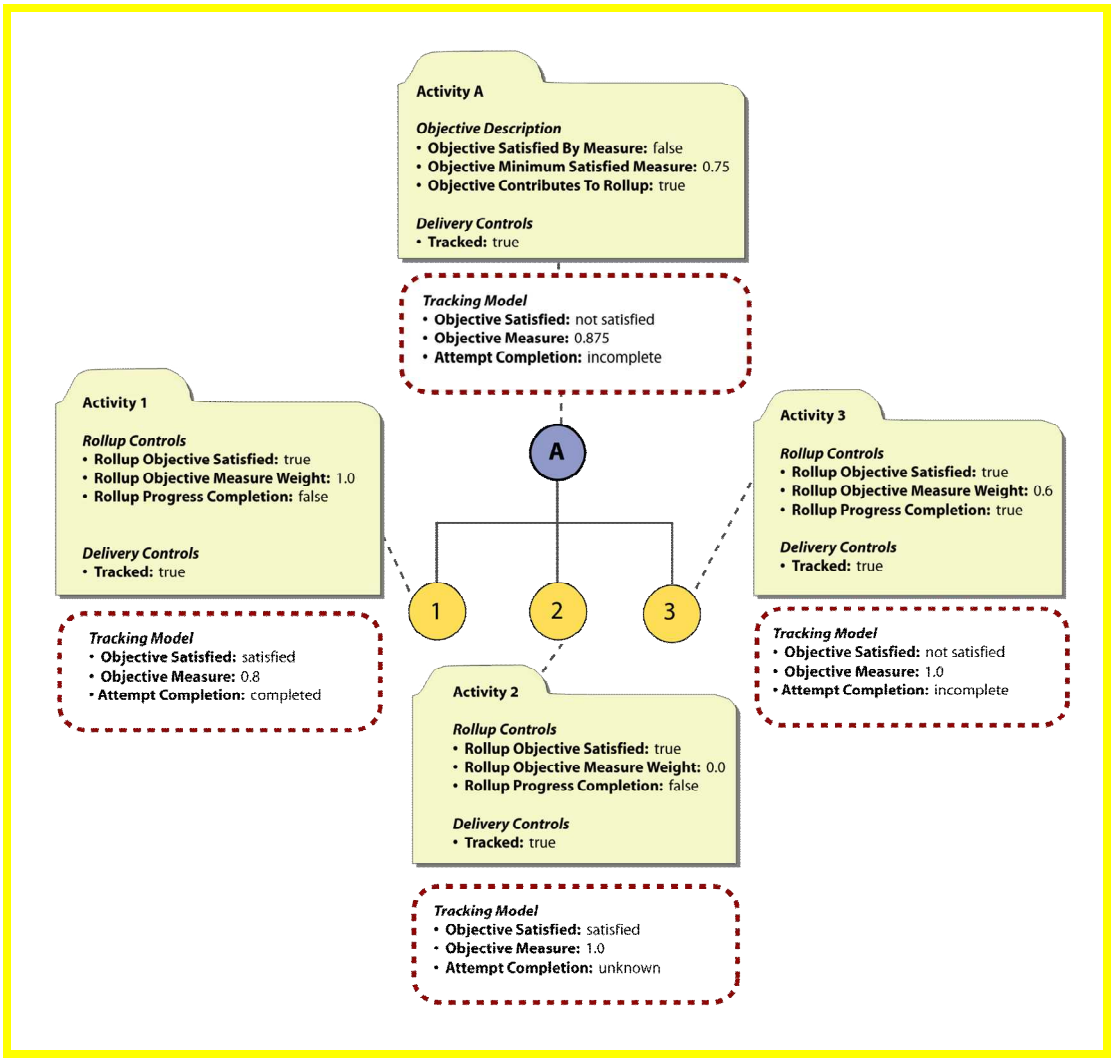

図 4.6.4d: デフォルトルールを使用して習得度を使用しない学習目標ロールアップ

4.6.5. アクティビティ進捗ロールアッププロセス

アクティビティ進捗ロールアッププロセス (*Activity Progress Rollup Process*) は, クラスタのアクティビティ試行進捗状態を *unknown*, *complete* もしくは *incomplete* に設定する. アクティビティ進捗ロールアッププロセスは, トラッキングされ, *Rollup Progress Completion* が True の子アクティビティだけを対象とする. 進捗情報をロールアップする方法には二つある. 最初に適用された方法だけがクラスタの進捗状態を評価するために使われる.

1. ルールの使用 – *complete* もしくは *incomplete* アクションのロールアップルールがアクティビティで定義されている場合, これらのルールがクラスタの進捗状態を決定するために評価される. *incomplete* ルールがまず最初に評価される.

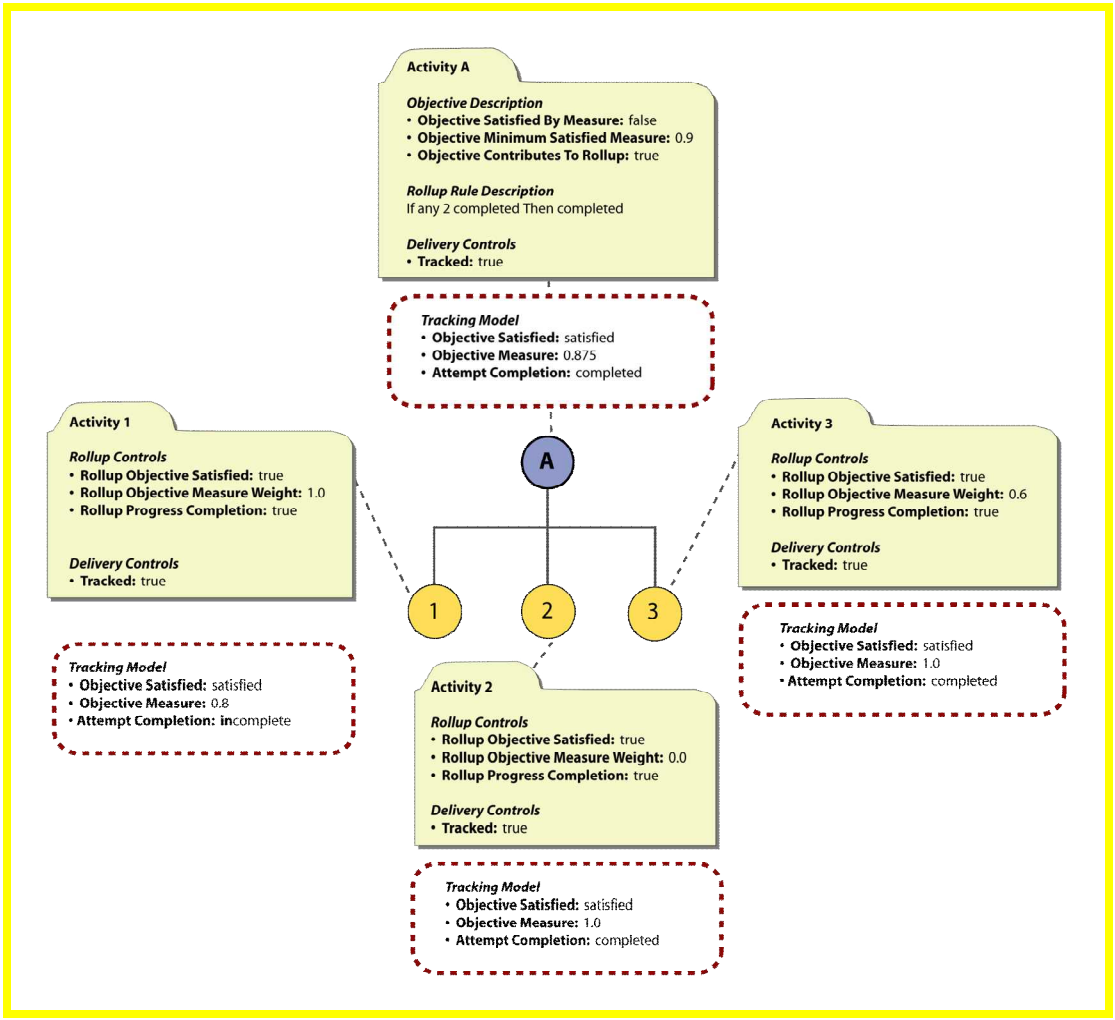

図 4.6.5a: ルールを使用したアクティビティ進捗状態ロールアップ

2. デフォルトルール – *complete* もしくは *incomplete* アクションのロールアップルールがアクティビティで定義されていない場合，デフォルトロールアップルールは以下のようになる：

   - If all completed, Then completed
   - If all (attempted or incomplete), Then incomplete

   デフォルトルールは定義されているロールアップルールと同一の順序で評価される．*incomplete* が先に評価される．

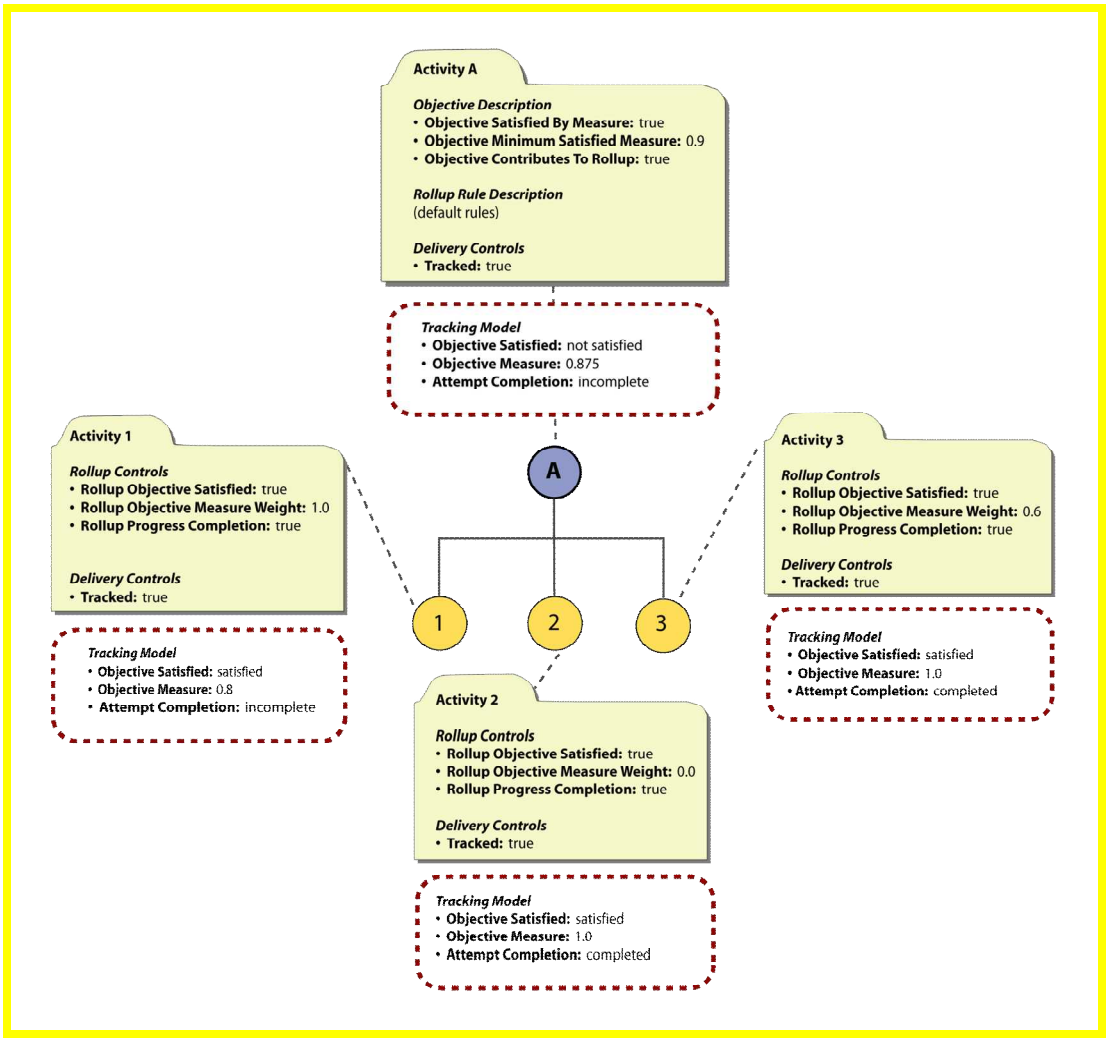

図 4.6.5b: デフォルトルールを使用したアクティビティ進捗ロールアップ

ADL ノート: アクティビティ進捗ロールアッププロセス評価は, アクティビティの *Attempt Completion Amount* の値に影響を与えない. *Attempt Completion Amount* の値は, LMS のシーケンシング実装では使用ないし保持されない. LMS は, 動作拡張が定義されたオブジェクトおよびアクティビティ進捗ロールアップ動作を変えない限り, *Attempt Completion Amount* に対して拡張ロールアップ動作を自由に定義, 実行することが可能である.

4.7. 選択ランダム化動作

このセクションの情報は, IMS SS 仕様の選択ランダム化動作セクションを置き換えるのではなく, 補完するものである. 詳細は IMS SS 仕様[1]を参照のこと. 実装に際しては, IMS SS 仕様に記述されている擬似コードではなく, シーケンシング動作擬似コード (付録C参照) で記述されている標準動作を実現するように要求されている.

選択ランダム化動作セクションは, クラスタの子のいくつかのサブセット (場合によっては全て) がいつ選択され, そしてそのサブセットがいつ並べ替えられるのかについて記述している. これらのプロセスは, 様々なシーケンシングプロセス中に, 使用可能なターゲットアクティビティに影響を与える.

子選択プロセス (*Select Children Process*) とランダム化プロセス (*Randomize Children Process*) は, アクティビティの選択ランダム化コントロール(セクション 3.11: 選択コントロールおよび 3.12: ランダム化コントロール参照)で定義されているように, LMS のシーケンシング実装によって適切に呼び出されることを意図している. これは, 終了動作中や終了動作後, シーケンシング動作中, または配信動作中に起こることがある. さらに, LMS がナビゲーション要求の仮評価を実施すると, オーバーオールシーケンシングプロセス (*Overall Sequencing Process*) 外で選択ランダム化プロセス (*Selection and Randomization Processes*) が呼び出されることがある. SCORM 対応 LMS へに対する唯一の要求は, 選択ランダム化プロセスが, 関連するシーケンシング定義モデル要素の時間属性と一貫性をもって適用されることである.

- **Never** –選択プロセス (*Selection Processes*) もしくはランダム化プロセス (*Randomization Processes*) は決して適用されない. 全ての子アクティビティは, 常に作成時に定義した順序でなされる.
- **Once** –選択プロセス (*Selection Processes*) もしくはランダム化プロセス (*Randomization Processes*) を現在のシーケンシングセッション中に一回適用する. これは, シーケンシング動作プロセス中にクラスタの子が対象となる前に発生しなければならない. LMS は通常, シーケンシングセッションが開始する前にこの決められたタイミングで, 選択ランダム化を全てのアクティビティに適用する
- **On Each New Attempt** –選択ランダム化プロセス (*Selection and Randomization Processes*) をアクティビティへの新しい試行を行っている最中, もしくはその前に適用する. ロールアップや様々なシーケンシング動作プロセス中に, 正確で一貫性のある子のセットが確実に使用されるために, LMS は通常, アクティビティへの最初の試行が始まる前および試行が終了した直後 (試行終了プロセス (*End Attempt Process*) 中) にこの定義されたタイミングでアクティビティに選択ランダム化を適用する.

4.7.1. 子選択プロセス

子選択プロセス (*Selection Children Process*) は, コンテンツ開発者が, 学習戦略を満たすために, クラスタに要求されるよりも多い子を含むことができるようにする. これは, 異なった学習者に異なった学習アクティビティを学習させることができるようにするためである. コンテンツ開発者は, クラスタのアクティビティのサブセットを学習者に提示するよう定義することができる. 選択プロセスは定義された数の子アクティビティを選択し, 子の相対的な順序は維持する. 様々なシーケンシングプロセス中では選択されたアクティビティだけが対象となる.

4.7.2. 子ランダム化プロセス

子ランダム化プロセス（*Randomize Children Process*）はコンテンツ開発者が，学習戦略を満たすために，学習者がアクティビティを学習する順序をかえることができるようにする．これは，異なった学習者が同じ学習リソースを異なった順番で学習することができるようにするものである．コンテンツ開発者は，クラスタの使用可能アクティビティ（コンテンツ開発者によって定義されたアクティビティや子選択プロセス（*Select Children Process*）中に選択されたアクティビティ）をランダムに並べ替えるよう定義できる．ランダム化プロセスは順序をかえるだけで，使用可能なアクティビティは変えない．様々なシーケンシングプロセスは，子ランダム化プロセスで定義された順序で子アクティビティを扱う．

4.8. シーケンシング動作

このセクションの情報は，IMS SS 仕様のシーケンシング動作セクションを置き換えるのではなく，補完するものである．詳細については IMS SS 仕様を参照のこと．実装は，IMS SS 仕様に記述されている擬似コードではなく，シーケンシング動作擬似コード（付録C参照）に記述されている標準動作を実現するように要求される．

このセクションで記述される動作は，SCORM シーケンシングの基礎となるものである．シーケンシング動作の目的は，アクティビティツリーの現在の状態において，ある定義された形で *Current Activity* からアクティビティツリーを探索し，次に配信するアクティビティを決定すること，もしくは，学習者に配信する最初のアクティビティを特定することにより新しいシーケンシングセッションを開始することである．

シーケンシングプロセスはアクティビティツリーの状態を変更しない．*Current Activity* を変えることもなければアクティビティのトラッキング状態情報に影響を与えることもない．シーケンシング動作は，シーケンシング要求プロセスが呼び出された時点で，アクティビティツリーの状態が現行化されていると仮定している．

シーケンシング動作がオーバーオールシーケンシングプロセスの一部として呼び出された場合，シーケンシング動作が配信するアクティビティを特定しないことがある．このコンディションにおいて，どのように学習者に適切な学習行為を提供するかについては LMS に任されている．

4.8.1.1 シーケンシング要求

SCORM 対応 LMS は以下のシーケンシング要求を処理し，表 4.8.1.1a で定義されている動作を実現できなければならない：

表 4.8.1.1a: SCORM 2004 シーケンシング要求

シーケンシング要求	シーケンシング要求サブプロセス
Start	*Start* シーケンシング要求サブプロセス
Resume All	*Resume All* シーケンシング要求サブプロセス
Continue	*Continue* シーケンシング要求サブプロセス
Previous	*Previous* シーケンシング要求サブプロセス
Choice	*Choice* シーケンシング要求サブプロセス
Retry	*Retry* シーケンシング要求サブプロセス
Exit	*Exit* シーケンシング要求サブプロセス

シーケンシング要求は全般的な動作により4つのカテゴリーに分類される:

シーケンシングセッションの開始 – *Start, Resume All, Choice* (シーケンシングセッションが始まる前) – これらの要求では *Current Activity* が未定義でなければならない(シーケンシングセッションはまだ始まっていない). これらの要求は学習者が新しいシーケンシングセッションで学習する最初のアクティビティを特定しようとする.

ADL ノート: 配信するアクティビティの特定に成功しても, アクティビティが配信されることは保証されない(セクション 4.9: 配信動作参照). シーケンシングセッションは, 学習者に最初のアクティビティが配信されるまでは, 開始されない.

- **アクティビティツリーの「次」アクティビティに向けての探索** – *Continue, Previous, Choice* (シーケンシングセッションが始まった後) – これらの要求では *Current Activity* が定義されていなければならない(シーケンシングセッションがすでに始まっている). これらの要求は *Current Activity* から始まり, アクティビティツリーを定義されたとおり探索し, 次に配信するアクティビティを見つける.
- **Current Activity の繰り返し** – *Retry* – この要求では *Current Activity* が定義されていなければならない(シーケンシングセッションがすでに始まっている). *Current Activity* を配信するか, *Current Activity* がクラスタの場合は使用可能な最初の子を配信しようとする.
- **シーケンシングセッションの終了** – *Exit* – この要求では *Current Activity* が定義されていなければならない(シーケンシングセッションがすでに始まっている). *Current Activity* がアクティビティツリーのルートの場合, シーケンシングセッションは終わる– これはオーバーオールシーケンシングプロセスを終了し, コントロールを LMS へ返すものである. *Current Activity* がアクティビティツリーのルートでない場合, この要求は配信するアクティビティを特定せず, LMS のシーケンシング実装は, 他のナビゲーション要求が出されるまで待機しなければならない.

4.8.2. シーケンシング要求プロセス

シーケンシング要求プロセス (*Sequencing Request Process*) は, ナビゲーション動作もしくは終了動作から発行されるシーケンシング要求によって, オーバーオールシーケンシングプロセス (*Overall Sequencing Process*) (セクション 4.3 参照)から呼び出される. シーケンシング要求プロセスの結果は, 次に学習者に配信するアクティビティの指定で, これを配信要求と呼ぶ. シーケンシング要求プロセスは, 保留中のシーケンシング要求に基づいて, 適切なシーケンシングサブプロセスを呼び出す. *Current Activity* が未定義の場合だけに実行されるもの(*Start* and *Resume All*)も含めて, 全てのシーケンシングサブプロセスは *Current Activity* で処理される.

実装は, オーバーオールシーケンシングプロセスの外で, シーケンシング要求プロセスを自由に呼び出すことができる. さらに実装は, 様々なシーケンシングサブプロセスを *Current Activity* 以外のアクティビティから自由に呼び出し, 追加の例外情報をトラッキングし, 知的UIコントロールやシーケンシング例外の文脈を有効にすることができる. しかし, 実装された SCORM 対応 LMS は, オーバーオールシーケンシングプロセスの中でシーケンシング動作が呼び出されたとき, シーケンシング動作擬似コード(付録 C 参照)に記述された標準動作を実現しなければならない.

4.8.3. 制限条件の評価

コンテンツ開発者は，アクティビティの使用可能範囲を定義することができる．制限条件は，シーケンシング定義モデル(セクション 3 参照)で定義される．SCORM は，*Max Attempt Limit* 制限条件だけをサポートする．この制限条件の評価は制限条件チェックプロセス中に実施される．

4.8.4. プリコンディションシーケンシングルールの評価

アクティビティは一つ以上のプリコンディションシーケンシングルールを持つことができる．シーケンシングルールの構造は以下のとおりである．

If [*condition set*] **Then** [*action*]

[*condition set*] は，アクティビティのトラッキング情報に対して個々に評価されるコンディションの集合を定義する．各コンディションは，否定（ルールコンディション演算子が「Not」）される場合もある一つの値を[*condition set results*]へ提供する．コンディションコンビネーションは[*condition set results*]に含まれる値の組に適用され，ルール評価のひとつの結果(true / false / unknown)を決める．ルール評価結果が true であればルール[*action*]が適用される．

いくつかのトラッキングモデル要素はペアで記述される．つまり，一つは状態データを記述し，もう一つはその状態データの有効性を記述する．これらの要素の評価を含むシーケンシングルールは，基になるトラッキング情報が無効な場合，[*condition set results*]で「unknown」値を返すことがある．ルールコンディション演算子とコンディションコンビネーション（付録C：UP.2.1）を「unknown」値を含む値の組へ適用する方法については以下の表で定義される：

表 4.8.4a: NOT Truth Table

NOT	True	False	Unknown
	False	True	Unknown

表 4.8.4b: AND Truth Table

AND	True	False	Unknown
True	True	False	Unknown
False	False	False	False
Unknown	Unkown	False	Unknown

表 4.8.4c: OR Truth Table

OR	True	False	Unknown
True	True	True	True
False	True	False	Unknown
Unknown	True	Unknown	Unknown

シーケンシングルールアクションは 3 つに分類される. この分類は, 通常, ルールの評価のタイミング, ルールが適用されるシーケンシングプロセス, それらのプロセスにおけるルールの影響に対応する. プリコンディションシーケンシングルールは, 様々なシーケンス要求プロセスの中の様々な時点で評価される

プリコンディションシーケンシングルールはシーケンシングルールチェックプロセス (*Sequencing Rules Check Process*) によって評価される.

例えば:

- **If satisfied Then skip** – アクティビティが satisfied なら, フローサブプロセスを実施中にアクティビティをスキップする.
- **If attempted Then disable** –アクティビティが試行されていれば, アクティビティを無効にする.
- **If always Then hidden from choice** –このアクティビティを *Choice* シーケンシング要求で選択しない.

上記の例は, アクティビティに定義されるシーケンシングルールの種類のうちの一部に過ぎない. シーケンシングルール記述のより詳細な定義についてはセクション 3.4 を参照のこと.

4.8.5. フローサブプロセス

フローサブプロセス (*Flow Subprocess*) は, LMS シーケンシング実装が, あるアクティビティからある方向性にアクティビティツリー内探索する方法について定義している. フローサブプロセスは, 多くのシーケンシングプロセス(*Start*, *Retry*, *Choice*, *Continue*, *Previous*)で, LMS シーケンシング実装がアクティビティツリーの探索を制御しなければならないときに使用される. フローサブプロセスが停止して(配信しようとするのは)葉アクティビティだけである. フローサブプロセスは *Sequencing Control Mode Flow* が False であるアクティビティに行き当たると停止する. 図 4.8.5a は, フローがツリー全体に対して有効であるとして, アクティビティツリーの葉の相対的な順序を図示している.

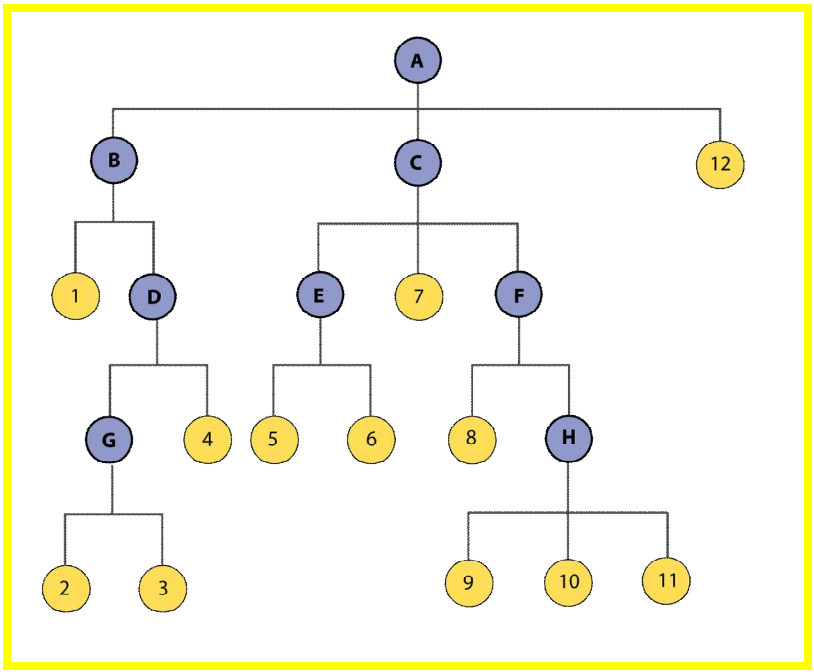

図 4.8.5a: アクティビティツリーを通る"Flowing"の相対的順序

フロープロセスは以下のように要約できる:

1. 指定されたアクティビティから指定方向に 1 アクティビティ移動し, 候補アクティビティの特定を試みる(シーケンシングツリートラバーサルサブプロセス (*Sequencing Tree Traversal Subprocess*) を呼び出す)

ループ

2. 候補アクティビティの親の *Sequencing Control Mode Flow* が False の場合, フローサブプロセスを終了する. 配信するものはない.
3. 候補アクティビティがスキップされた場合, 指定されたアクティビティからさらに指定方向に 1 アクティビティの移動を試みる(シーケンシングツリートラバーサルサブプロセス (*Sequencing Tree Traversal Subprocess*) を呼び出す) – Step 2 を繰り返す.
4. アクティビティ候補が無効ではないことを確認する. 無効の場合, フローサブプロセスを終了する. 配信するものはない.
5. アクティビティ候補が制限条件に違反しないことを確認する. 制限条件に違反している場合, フローサブプロセスを終了する. 配信するものはない.
6. アクティビティ候補が葉の場合, 配信要求でアクティビティが指定される. – フローサブプロセスを終了する.
7. アクティビティ候補がクラスタの場合, 適切な方向にクラスタに入る:
   - 前方に探索する場合, 次のアクティビティは最初の子.
   - 後方に探索し, クラスタの Forward Only が False の場合, 次のアクティビティは最後の子.
   - 後方に探索し, クラスタの Forward Only が True の場合, 次のアクティビティは最初の子 – 一時的に(クラスタの子を検証している間)前方へフロー.
8. アクティビティが指定されない場合, フローサブプロセスを終了する. 配信するものはない.
9. Step 2 を**繰り返す**

ADL ノート: アクティビティツリーの最後の利用可能な葉アクティビティを越えて前方への移動を試みた場合, アクティビティツリーから離脱することができる. シーケンシングセッションの間, アクティビティツリーの任意のアクティビティが (カレントアクティビティであっても), 様々なシーケンシング情報およびアクティビティツリーの状態の組み合わせにより, カレントアクティビティに対して相対的に最後の葉アクティビティとなる. シーケンシング要求によりツリーから離脱する移動が生じる場合の LMS の相互運用性を保った動作を保証するため, LMS はアクティビティツリーのルートの現在の試行とすべてのアクティブな子孫を終了して, シーケンシングセッションを終えなくてはならない.

4.8.6. オーバーオールシーケンシングプロセス

このセクションの情報は, IMS SS 仕様のシーケンシング要求サブプロセスセクションを置き換えるのではなく, 補完するものである. 詳細に関しては IMS SS 仕様を参照のこと. 実装においては, IMS SS 仕様に記述されている擬似コードではなく, シーケンシング動作擬似コード (付録C参照) で記述されている標準動作を実現するように要求されている.

4.8.6.1 Start シーケンシング要求サブプロセス

Start シーケンシング要求サブプロセス (*Start Sequencing Request Subprocess*) は, シーケンシングセッションがまだ始まっていないことを要求する. *Start* シーケンシング要求サブプロセスは, アクティビティツリーのルートへフローすることにより新しいシーケンシングセッションを始めようとする. このプロセスはフローサブプロセス (*Flow Subprocess*) (セクション 4.8.5 参照)を使用する.

Sequencing Control Mode Flow が False のアクティビティに遭遇してフローサブプロセスが終了した場合, 配信するものは指定されずシーケンシングセッションは始まらない. このケースでは, シーケンシングセッションを開始するのに, 学習者が望むアクティビティを指定するように, LMS が何らかのメカニズム (例えば, 選択ナビゲーションユーザインターフェース) を提供することが推奨されている.

アクティビティが一つだけのアクティビティツリーの場合, そのアクティビティは葉である. フローサブプロセスは呼び出されず, アクティビティツリーのルートが配信対象に指定される.

4.8.6.2 Resume All シーケンシング要求サブプロセス

Resume All シーケンシング要求サブプロセス (*Resume All Sequencing Request Subprocess*) は, シーケンシングセッションがまだ始まっていないことを要求する. *Resume All* シーケンシング要求サブプロセスは, 最後のシーケンシングセッションが *Suspend All* ナビゲーション要求によって終了したかどうかを決定するために, *Suspended Activity* のアクティビティ状態情報要素を確認する. *Suspend All* ナビゲーション要求によって終了した場合, *Suspended Activity* は前回のシーケンシングセッションを再開するアクティビティを指定する.

Suspended Activity が定義されていない場合, サブプロセスは終了し, シーケンシングセッションは始まらない. *Suspended Activity* が定義されていた場合, *Suspended Activity* が配信対象に指定される.

4.8.6.3 Retry シーケンシング要求サブプロセス

Retry シーケンシング要求サブプロセス (*Retry Sequencing Request Subprocess*) は, 終了動作中に評価されたポストコンディションシーケンシングルールによって呼び出される. サブプロセスは, シーケンシングセッションは既に始まっていること, そして *Current Activity* がリトライの対象であることを前提としている. *Current Activity* がクラスタの場合, retry プロセスは, 学習者が次にどのアクティビティを学習すべきか決定するために, フローサブプロセス(セクション 4.8.5 参照)を呼び出す.

ADL ノート: このシーケンシング要求の目的は, あるアクティビティとそのアクティビティの子孫に新しい試行を始めることである. シーケンシング要求の処理中に, LMS は, アクティビティツリー探索中に遭遇した全アクティビティの評価に対して, デフォルトトラッキング情報を適用しなければならない.

4.8.6.4 Exit シーケンシング要求サブプロセス

Exit シーケンシング要求サブプロセス (*Exit Sequencing Request Subprocess*) は, シーケンシングセッションが既に始まっていること, そして *Current Activity* が exit する対象であることを前提としている. このサブプロセスは配信するアクティビティを特定しない. *Current Activity* がアクティビティツリーのルートの場合, *Exit* シーケンシング要求サブプロセスは, シーケンシングセッションを終了し, コントロールを LMS へ返すことを示している.

4.8.6.5 Continue シーケンシング要求サブプロセス

Continue シーケンシング要求サブプロセス (*Continue Sequencing Request Subprocess*) は, 既存のシーケンシングセッションが既に始まっていることを前提としている. *Current Activity* に対する *Sequencing Control Mode Flow* が True の場合, フローサブプロセス(セクション 4.8.5 参照)が *Current Activity* から前方へ呼び出される. フローサブプロセスがアクティビティを特定すると, そのアクティビティが配信対象として特定される.

4.8.6.6 Previous シーケンシング要求サブプロセス

Previous シーケンシング要求サブプロセス (*Previous Sequencing Request Subprocess*) は, 既存のシーケンシングセッションが既に始まっていることを前提としている. *Current Activity* に対する *Sequencing Control Mode Flow* が True の場合, フローサブプロセス(セクション 4.8.5 参照)が *Current Activity* から後方へ呼び出される. フローサブプロセスがアクティビティを特定すると, そのアクティビティが配信対象として特定される.

4.8.6.7 Choice シーケンシング要求サブプロセス

Choise シーケンシング要求サブプロセス (*Choice Sequencing Request Subprocess*) は, 学習者 (もしくはシステム) が配信対象に指定したアクティビティを指定する. シーケンシングセッションが既に始まっている場合, *Choice* シーケンシング要求サブプロセスは, *Current Activity* から対象アクティビティへとアクティビティツリーを探索する. シーケンシングセッションが始まっていない場合, *Choice* シーケンシング要求サブプロセスは, ルートから対象アクティビティへとアクティビティツリーを探索する. Choice プロセスがアクティビティを特定し, そのアクティビティが葉でない場合, フローサブプロセス(セクション 4.8.5 参照)がそのアクティビティから前方へ起動される. *Choice* シーケンシング要求サブプロセスが葉アクティビティを特定すると, そのアクティビティが配信対象に特定される.

4.9. 配信動作

このセクションの情報は，IMS SS 仕様の配信動作セクションを置き換えるのではなく，補完するものである．詳細については IMS SS 仕様を参照のこと．実装は，IMS SS 仕様に記述されている擬似コードではなく，シーケンシング動作擬似コード（付録C参照）に記述されている標準動作を実現するように要求される．

配信動作は，オーバーオールシーケンシングプロセス（*Overall Sequencing Process*）の最後のステップについて定義している．配信動作の目的は，指定された配信要求をうけて，その要求の有効性を確認し，有効な場合，適切なコンテンツオブジェクトを配信することである．LMS は，コンテンツパッケージを使って，指定されたアクティビティに対して配信するコンテンツオブジェクトを決定しなければならない．配信動作がオーバーオールシーケンシングプロセスの一部として呼び出された場合，配信要求が有効とされないことがある．この場合，学習者に適切な学習行為を提供する方法は LMS に任されている．

配信動作(配信要求プロセス(*Delivery Request Process*))は，オーバーオールシーケンシングプロセスの外で，LMS から呼び出されることがある．これは，配信要求の「what-if」評価を実施するために行われる．配信要求プロセスはトラッキング情報に影響を与えない．従って，副作用の心配なく呼び出すことができる．しかし，結果を適切に管理するのは実装の責任である．

SCORM のゴールの一つは，コンテンツオブジェクトが，複数の LMS 間で再利用でき相互運用できることである．これを可能にするため，コンテンツオブジェクトが試行を開始する共通の方法がなければならない．コンテンツ配信環境プロセス（*Content Delivery Environment Process*）は，LMS シーケンシング実装と SCORM 配信メカニズムの間を取り持つものを定義している．予想されるコンテンツオブジェクトの配信を待つようにアクティビティツリーの状態を管理し，SCORM 配信メカニズムにその学習リソースを指定する．

SCORM 配信メカニズムは，LMS が Web ベースのコンテンツオブジェクトの試行を開始する共通の方法を定義する．このメカニズムは，配信されたコンテンツオブジェクトと LMS の通信の確立に対する手順と役割を定義する．通信プロトコルは，共通 API を用いて標準化されている．この共通配信スキームは，基になる LMS 実装に依らず，複数の LMS にわたる一貫したコンテンツオブジェクト配信動作を保証する．

ADL ノート: この文脈では，用語「LMS」は学習リソースの配信を管理する機能を含むシステムを示すのに使用される．この配信スキームは，学習行為の中で，SCO および起動可能なアセットという Web ベース学習リソースの配信を扱う．

4.9.1. 配信要求プロセス

配信要求プロセス（*Delivery Request Process*）は，配信要求で指定されたアクティビティが配信可能か否かを決定する．すなわち，保留中の配信要求の有効性を確認する．このプロセスは，アクティビティツリーをルートから特定されたアクティビティまで進み，途中のアクティビティが無効でないこともしくは制限条件に違反していないことを確認する．アクティビティが無効もしくは制限条件に違反していれば，何も配信されず，*Current Activity* は変わらない．LMS シーケンシング実装は制御を LMS へ返し，他のナビゲーション要求を待つ．

4.9.2. コンテンツ配信環境プロセス

コンテンツ配信環境プロセス (*Content Delivery Environment Process*) は, オーバーオールシーケンシングプロセス (*Overall Sequencing Process*) に呼び出される最後のプロセスである. 配信要求を受け, 指定されたアクティビティの配信に対して, アクティビティツリーの準備をする. このプロセスは以下を含む:

1. 現在アクティブで, 特定されたアクティビティが配信される際にアクティブでなくなるアクティビティ全ての現在の試行を終了する
2. 現在非アクティブで, 特定されたアクティビティが配信されるとアクティブになるアクティビティ全ての試行を開始 (再開) する
3. 新たにアクティブになったアクティビティ全てに対して適切なトラッキング情報を初期化する
4. 配信に特定されたアクティビティを LMS へ特定する

コンテンツ配信環境プロセスの結果, LMS シーケンシング実装は制御を LMS へ返し, 他のナビゲーション要求を待つ.

コンテンツ配信環境プロセスは, オーバーオールシーケンシングプロセスの外で呼び出してはならない. 呼び出すと矛盾した一貫性の無い動作を起こす可能性がある.

ADL ノート: SCO が起動されるアクティビティに関連付けられているとき, LMS は SCO の cmi.objectives データモデル要素をアクティビティのトラッキングデータおよび関連する Read Objective Maps の現在の情報で初期化する (更新する) 責任がある. 表 4.9.2a に必要なランタイムデータの更新が要約されている. 前回の学習者のセッションの間に SCO が生成した追加の学習目標は初期化の影響を受けない.

表 4.9.2a: シーケンシングトラッキングデータの SCO ランタイムデータへのマッピングのまとめ

SCORM ランタイム環境データモデル要素		シーケンシングトラッキングデータモデル要素
1.	アクティビティにおいて定義されているすべての学習目標に対して.	`cmi.objectives` 要素は学習目標の `ID` と同じ `ID` を持つ学習目標で初期化される.
2.	アクティビティにおいて定義されているすべての学習目標に対して, それらの学習目標の Objective Progress Status と Objective Satisfied Status が適切な値の決定に用いられる.	アクティビティの学習目標と同じ ID を有する SCO の学習目標に対して, `cmi.objectives.n.success_status` は Objective Progress Status と Objective Satisfied Status を用いて以下のように初期化される.
	Objective Progress Status = true Objective Satisfied Status = false	`Failed`
	Objective Progress Status = true Objective Satisfied Status = true	`Passed`
3.	アクティビティにおいて定義されているすべての学習目標に対して, それらの学習目標の Objective Measure Status と Objective Normalized Measure が適切な値の決定に用いられる.	アクティビティの学習目標と同じ ID を有する SCO の学習目標に対して, `cmi.objectives.n.score.scaled` は Objective Measure Status と Objective Normalized Measure を用いて以下のように初期化される.
	Objective Measure Status = true Objective Normalized Measure = 定義された値	-1.0 から 1.0 の間の定義された値

4.9.3. コンテンツオブジェクトの起動

LMS は，オーバーオールシーケンシングプロセス（*Overall Sequencing Process*）によって配信対象に特定されたアクティビティと対応付けられたコンテンツオブジェクトを準備し起動する役割を持つ．この動作は SCORM RTE ブック[4]で定義されている．

セクション5
SCORM ナビゲーションモデル

このページは空白である．

5.1. ナビゲーションモデル概要

SCORM において, 学習者に提供される学習行為は, あるアクティビティツリーに対して学習者が学習した一連の学習アクティビティである. つまり, シーケンサによって配信対象として特定され最終的に起動された一連のアクティビティである. シーケンシング動作セクション(セクション 4.3: オーバーオールシーケンシングプロセス (*Overall Sequencing Process*) 参照)で説明されているように, LMS のシーケンシング実装は, LMS の受動的な部分であり, LMS が発行したナビゲーション要求に応えて動作するだけである. ナビゲーションは, 学習行為を実現するために, 学習者と LMS が協力してナビゲーション要求を特定するプロセスである.

通常, LMS は, 学習者が望むナビゲーション要求を発行するのに使用するユーザーインターフェース装置を提供する. 場合によっては, LMS ではなく, コンテンツがこれらのインターフェース装置を提供すべきであるとコンテンツ開発者が望むことがある. LMS が提供するインターフェース装置に追加して, コンテンツがインターフェース装置を提供する事もたびたびある. いずれの場合も, ナビゲーション要求はアクティビティツリー内の学習者もしくはコンテンツ主導の移動に対応する.

SCORM は, 実行時に学習者に提示されるユーザーインタフェースの種類もしくはスタイルにどのような要求も強要しない. ユーザーインターフェースの特性, および, 学習者と LMS の間のやりとりを入力するメカニズムは意図的に規定されていない. ルックアンドフィール, 提示スタイル, ユーザーインターフェース装置もしくはコントロールの配置といった事項は SCORM の対象外である.

5.2. ナビゲーション要求の発行

SCORM ナビゲーションモデルは，学習アクティビティ間のナビゲーションにのみ適用される．現時点で，SCORM は SCO 内でシーケンシングもしくはナビゲーションを定義する方法については直接取り上げない．SCORM は，SCO 内のナビゲーションを排除するわけではない（このナビゲーションは完全に SCO に制御されている）．例えば，SCORM ナビゲーションは，複数ページある SCO 内の個別のページ間のナビゲーションには適用されない．

SCORM ナビゲーションモデルは，学習者が LMS およびコンテンツが提供するユーザーインターフェース装置を通して発行するナビゲーションイベント，および SCO が直接発行するナビゲーションイベントを定義している．そのようなイベントが SCO 内もしくは LMS を通じて発行される方法については SCORM では定義しない．さらに，SCORM は，ランタイム時に学習者に提示されるユーザーインタフェースの種類ないし形式にどのような要求もしない．ユーザーインターフェースの特性および学習者と LMS 間のやりとりを入力するメカニズムは意図的に規定されていない．ルックアンドフィール，提示スタイル，ユーザーインターフェース装置もしくはコントロールの配置といった事項は SCORM の対象外である．

ナビゲーション要求は，SCORM シーケンシング動作（セクション 4.4：ナビゲーション動作（*Navigation Behavior*）参照）によって定義されたように処理される．ナビゲーション要求は，ある特定の学習アクティビティを選択する，次のアクティビティへ継続する，前回のアクティビティへ戻るなど，アクティビティツリーを移動するのに望まれる方法を表わす相互運用可能な手段を学習者およびコンテンツに提供する．

表 5.2a は，ナビゲーションイベントの一覧と，これらのナビゲーションイベントのナビゲーションとの対応を定義する．更に各々のナビゲーション要求の発行元を定義する．

表 5.2a: ナビゲーションイベントおよび記述

ナビゲーションイベント	動作説明	ソース
Start	このイベントは，アクティビティツリーで使用可能な最初の，つまり「開始」アクティビティを特定する要求を示す．一般的にこのイベントは，学習者がアクティビティツリーのルートへの新たな試行を開始するとき，LMS によって自動的に生成される． このイベントは *Start* ナビゲーション要求を発行する．	LMS のみ
Resume All	このイベントは，前回中断されたアクティビティツリーのルートへの試行を再開したいという要求を示す．一般的にこのイベントは，学習者が前回中断されたアクティビティツリーの試行を再開するとき，LMS によって自動的に生成される． このイベントは，*Resume all* ナビゲーション要求を発行する．	LMS のみ
Continue	このイベントは，アクティビティツリーで（*Current Activity* に対して）論理的に次に使用可能な学習アクティビティを特定したいという要求を示す． このイベントは，*Continue* ナビゲーション要求を発行する．	LMS ないし SCO
Previous	このイベントは，アクティビティツリーで（*Current Activity* に対して）論理的に前で使用可能な学習アクティビティを特定したいという要求を示す．	LMS ないし SCO

	このイベントは, *Previous* ナビゲーション要求を発行する.	
Choose	このイベントは, アクティビティツリーの特定の学習アクティビティへ直接飛びたいという要求を示す. このイベントは, 指定された対象アクティビティに対する *Choice* ナビゲーション要求を発行する.	LMS ないし SCO
Abandon	このイベントは, 現在配信されているコンテンツオブジェクトへの現在の試行を, 後で再開することを意図せずに, 早期に異常終了したいという要求を示す. このイベントは *Current Activity* における現在の試行を終了する. *Current Activity* に親がある場合, 親アクティビティの試行は終了しない. さらに, *Abandon* は *Current Activity* の祖先に直接の影響を与えない. 放棄された試行は1試行として数えられる. *Abandon* は, 既に記録されたトラッキング情報をロールバックすることを意味しない. 例えば, アクティビティが一旦 passed もしくは completed と記録されたら, failed もしくは incomplete となることはない. このイベントは, *Abandon* ナビゲーション要求を発行する.	LMS ないし SCO
Abandon All	このイベントは, アクティビティツリーのルートアクティビティへの現在の試行を, 後で再開することを意図せずに, 異常終了したいという要求を示す. このイベントは, アクティビティツリーのルートアクティビティおよびすべてのアクティブな学習アクティビティの現在の試行を終了する. 全ての放棄された試行は1試行として数えられる. *Abandon All* は, 既に記録されたトラッキング情報をロールバックすることを意味しない. 例えば, アクティビティが一旦 passed もしくは completed と記録されたら, failed もしくは incomplete となることはない. このイベントは, *Abandon All* ナビゲーション要求を発行する.	LMS ないし SCO
Suspend All	このイベントは, アクティビティツリーのルートアクティビティへの現在の試行を, 休止したいという要求を示す. このイベントは, アクティビティツリーのルートアクティビティおよびすべてのアクティブな学習アクティビティの現在の試行を中断する. 中断されたアクティビティへの試行が終了する事はない. アクティビティツリーのルートアクティビティへの次の試行が *Resume All* イベントで開始されたら, 中断されたすべてのアクティビティへの試行は再開する. *Suspend All* は, 既に記録されたトラッキング情報をロールバックすることを意味しない. 例えば, アクティビティが一旦 passed もしくは completed と記録されたら, failed もしくは incomplete となることはない.	LMS のみ

	このイベントは, *Suspend All* ナビゲーション要求を発行する.	
Unqualified Exit	このイベントは, 現在配信されているアクティビティへの現在の試行が正常に終了したこと, および終了が *Continue*, *Previous* もしくは *Choose* という他のナビゲーションイベントによって発行されなかったことを示す. このイベントは *Current Activity* への現在の試行を終了する. このイベントは, *Exit* ナビゲーション要求を発行する.	LMS ないし SCO
Exit All	このイベントは, アクティビティツリーのルートアクティビティへの現在の試行が正常に終了したことを示す. このイベントはアクティビティツリーのルートアクティビティおよびすべてのアクティブな学習アクティビティの現在の試行を終了する. このイベントは *Exit All* ナビゲーション要求を発行する.	LMS ないし SCO

5.3. ナビゲーション要求の処理

学習者もしくはコンテンツがなんらかの方法でナビゲーションイベントを発行すると，LMS はシーケンシングシステムを起動して対応するナビゲーション要求を処理する．ナビゲーション要求の処理結果は常に以下の一つとなる：

1. ナビゲーション要求の内容がアクティビティツリーの現在の試行を終了することであれば，LMS は，Exit All ナビゲーション要求を処理して，試行を終了させ制御を LMS へ戻す．
2. アクティビティツリーの現在のトラッキング状態および適用可能なシーケンシング情報を評価した結果，LMS は意図されたナビゲーション要求を処理する事を放棄する．この場合，LMS はナビゲーション要求を無視する．LMS は他のナビゲーション要求が発行されるまで，シーケンシングアクションを取らない．

   例えば：

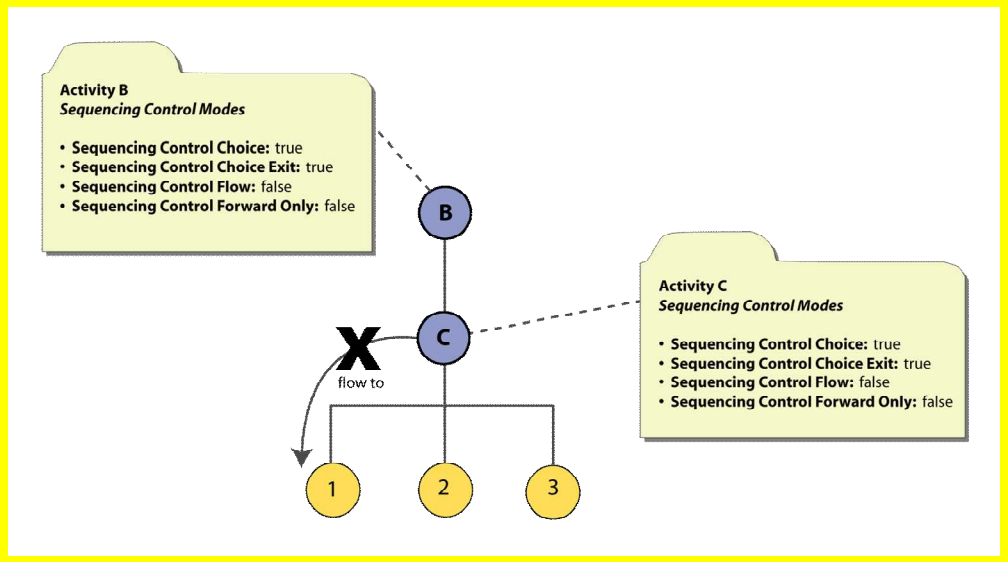

図 5.3a: 無効なフローとクラスタアクティビティの選択

図 5.3a に記されたアクティビティツリーの一部で，学習者は現在アクティビティ AAA(図示されていない)を学習しているとする．もし，アクティビティ BA に対する choice ナビゲーション要求が発行されたとすると，LMS はその要求を評価（有効性を確認）し，配信するアクティビティが特定されないと決定する．この情報を使用して，LMS は要求を無視し，学習者にアクティビティ AAA を継続させる．

3. アクティビティツリーの現在のトラッキング状態および適用可能なシーケンシング情報を評価した結果、LMS は意図されたナビゲーション要求の処理を実行すべきであると決定する。LMS は意図されたナビゲーション要求に基づきオーバーオールシーケンシングプロセス（セクション 4.3 参照）を呼び出す。オーバーオールシーケンシングプロセス の結果は以下の一つである：

   a. **配信するアクティビティが特定される** – LMS は特定された学習アクティビティに関連付けられたコンテンツオブジェクトを準備し起動する(SCORM RTE ブック[4]参照).

b. **配信するアクティビティが特定されない** – このケースでは, SCORM は, LMS 動作にどんな要求もしない. しかし, LMS が学習者への影響を最小に抑えることを推奨する.

   例えば, 上記の例で説明されたように, LMS がアクティビティ BA に対する choice ナビゲーション要求を実行すると, 配信するアクティビティは特定されない. アクティビティ AAA の現在の学習セッションは終了するが, 以降の LMS 動作は定義されていない.

c. **シーケンシングセッションの間に例外が発生する** – このケースでは, SCORM は, LMS 動作になんの要求もしない. しかし, LMS が例外をとり扱って学習者への影響を最小に抑えることを推奨する.

5.4. ナビゲーションによるコンテンツオブジェクトの終了

LMS がナビゲーションイベントを発生するユーザーインターフェース装置を提供する場合, 学習者は一つ以上のこれらの装置を操作してナビゲーション要求を示すことができる. 学習者がナビゲーション要求を発行するとき, SCORM は, 学習者が現在起動されているコンテンツオブジェクトを終了したことを示唆していると仮定する. LMS が学習者からのナビゲーションイベントを受け取ると, LMS はまず現在起動されているコンテンツオブジェクトを取り除き (アンロードし), その後適切なナビゲーション要求を処理する. シーケンシングに影響する学習者トラッキング情報をコンテンツオブジェクトが確実に記録するために, コンテンツオブジェクトはナビゲーション要求を処理する前に終了されていなければならない.

SCO は SCORM ナビゲーションデータモデルを介してナビゲーション要求を直接 LMS と通信できる. SCO は, その意図に基づいて LMS がいつ動作すべきかを, LMS に通知しなければならない. これは, Terminate() (SCORM RTE book [4]参照)を呼び出すことにより行われる. Terminate() API メソッドは, SCO が LMS との通信を完了したことを示す. 従って, SCO が通信を完了しナビゲーションの意図を示したら, LMS はそれに基づいて動作しなければならない.

一旦 Terminate()要求が処理されたら, LMS はまず学習者が選択した保留中のナビゲーションイベントを処理する. 学習者からの保留中のナビゲーションイベントがなければ, LMS は SCO と通信した最後のナビゲーション要求を処理する(セクション 5.6.4:ナビゲーション要求のランタイム通信参照). 学習者, SCO 共にナビゲーション要求を示さない場合は, LMS は学習者がナビゲーションイベントを示すまで待たなければならない.

SCO A が起動され Initialize()を呼び出すシナリオを考えてみよう. SCO A が実行されている間に, 学習者は LMS が提供したナビゲーションユーザーインターフェースコントロールを使用して他のアクティビティを選択する. 対応する SCO である SCO B が同じブラウザー画面で起動され, それにより, 元の SCO である SCO A を終了する. このケースでは, 自身が終了されると認識するとき, SCO A は Terminate()を呼び出さなければならない. たとえ SCO A が Terminate()を呼び出すことに失敗しても, LMS は SCO A と通信していたセッションを終了するよう実装しなければならない.

SCO A は, onUnload イベントに対して, LMS と必要な通信を実行し, さらに Terminate()を呼び出すハンドラーを実装しなければならない. SCORM は将来, 強制的に SCO が終了させられる前に, LMS が SCO に通知を行なう通信メカニズムを導入するかもしれない.

学習者が発行したナビゲーションイベントは, SCO が通信したナビゲーション要求より常に優先される. 例えば, 上記のシナリオのように, SCO が終了される前にナビゲーションイベントを LMS へ通信しても, LMS が提供するユーザーインターフェースを通じて学習者から他のナビゲーションイベントが発行され次第, LMS は SCO からのナビゲーションイベントを無効にする. 上記のシナリオでは, SCO A が *Continue* ナビゲーション要求を LMS へ通信し, 次に *Choose* ナビゲーションイベントにより Terminate されたとすると, *Choose* イベントが学習者から発行されたために実行され, SCO A が送信した *Continue* ナビゲーション要求は無視される.

5.5. ナビゲーションおよび補助リソース

IMS SS 仕様は，学習者に提供される学習アクティビティに関連した支援サービスとして補助リソースの概念を定義している．補助リソースは，用語集，参照マニュアル，チャットルーム，掲示板等々を含む．IMS SS 仕様は，補助リソースに対する最小限のフックしか提供せず，現時点で SCORM は補助リソースの相互運用可能な使用方法を定義する充分なコミュニティ要求を持たない．SCORM は補助リソースの使用を禁止しないが，将来的な相互運用性を確保するため，コンテンツ開発者および LMS ベンダは補助リソースの使用に充分注意することが推奨される．

5.6. ナビゲーションに対するユーザーインターフェース (UI) 装置

5.6.1. ナビゲーションに対するUI装置の提供

LMS は, 最低限, 学習者が UI 装置を通してナビゲーションイベントを発行する機能を提供しなければならない. SCORM は, 実行時に学習者に提示されるユーザーインターフェース (ナビゲーションのための UI 装置を含む) のタイプや形式に要求をしない. UI の特徴と学習者と LMS の間のやりとりを入力する仕組みは意図的に規定されていない. ルックアンドフィール, 提示スタイル, ユーザーインターフェース装置もしくはコントロールの配置は SCORM の規定外である.

SCORM は, 学習者に提供される UI 装置の種類やスタイルにどのような要求もしないが, LMS が提供するユーザーインターフェース装置は, ナビゲーションイベントを起こすことができるものだけとすることを推奨している. すなわち LMS が有効なナビゲーション要求を生成できるもののみである. また, LMS は, 全てのコンテンツオブジェクトを通して, 提供された UI 装置の一貫性のあるルックアンドフィールを維持することが推奨されている. さらに, LMS が提供する UI 装置を通して発行された動作は, 全ての学習行為を通して一貫性があることが推奨される.

SCO は, オプションとしてナビゲーションイベントを発行するユーザーインターフェース装置を実装することがある. SCORM は, SCO がナビゲーション要求をどのように発行するのか, および SCO があるナビゲーション要求が有効か否かをどのように LMS に問い合わせるのか, について規定している. コンテンツ開発者は, コンテンツオブジェクト毎に, UI 装置を提供するか否かを選択できる. さらに LMS が, 同じナビゲーションイベントに対する重複する UI 装置を提供しないようにすることを選択できる. この目的は, SCO および LMS が提供する UI 両方に表れる“previous” および “continue” ボタンなど, 重複する UI 装置で学習者が混乱するのを避けることである. この機能の他の使用方法は SCORM の対象範囲外である.

5.6.2. isvisible 属性の使用

SCORM CAM ブック[3]は, isvisible 属性の使用について説明している. isvisible 属性は, パッケージの構造が表示されたとき, それに関連付けられた項目が表示されるかどうかを示す. その値は, それが定義された項目だけに影響し, 項目の子もしくは項目と関連付けられたリソースには影響しない. invisible 項目の値が非表示にされた場合(isvisible=false), Choose ナビゲーションイベントにために LMS が提供する UI 装置で項目を表示しないようにすることが推奨される.

しかし, isvisible の効果は, UI 装置での提示に限られており, *Choice* ナビゲーション要求に関連した SCORM シーケンシング動作に影響しない. 非表示の学習アクティビティ (SCORM コンテンツパッケージの isvisible 属性が false に設定されたもの) は, *choice* ナビゲーション要求の対象となる. *Choice* ナビゲーション要求の対象となる学習アクティビティは, 配信対象として特定され起動されることがある. 従って, コンテンツ開発者は, ある特定の学習アクティビティを *choice* ナビゲーション要求によって配信されないようにしたい場合, “If always then hide from choice”プリコンディションアクションシーケンシングルールを学習アクティビティに適用しなければならない.

5.6.3. プレゼンテーション情報モデル

SCORM ナビゲーションモデルは, コンテンツオブジェクトの, あるプレゼンテーション特性をコンテンツ開発者が設定するための最小限のプレゼンテーションモデルを定義している. コンテンツ開発者は, コンテンツオブジェクト毎に, コンテンツオブジェクトがある UI 装置を提供するように設定できる. さらに LMS が同じナビゲーションイベントに対する重複した UI 装置を提供しないように設定できる. 表 5.6.3a では, コンテンツのプレゼンテーションの意図を示すのに使用する構造を定義する. プレゼンテーションモデルは, 将来, 他のコンテンツオブジェクトプレゼンテーション特性を示すのに使用することがある.

表 5.6.3.a: プレゼンテーション情報モデル

Nr	名称	説明	値空間	データタイプ	デフォルト
1	Presentation	プレゼンテーションに関する情報	-	-	
1.1	Navigation Interface	ユーザーインターフェースコントロールに関する特徴	-	-	
1.1.1	Hide LMS UI	対応するナビゲーションイベントを学習者が発行する特定のユーザーインターフェース装置を LMS が提供すべきでないことを示す	ゼロ以上の語彙トークン	既定トークンのあるオープン, 拡張可能語彙 (表 5.6.3b 参照)	(空)

表 5.6.3b はコンテンツが提供できる UI devices のリストである. LMS は, 重複して混乱を招きかねない UI ナビゲーション装置を提供しないように, UI ナビゲーション装置を隠す要求をすべて実行することが推奨される.

表 5.6.3b: ランタイムユーザーインターフェース装置語彙

トークン	定義	説明
previous	Previous navigation device	このトークンが定義された場合, 対応するコンテンツオブジェクトが起動している間, LMS は Previous ナビゲーションイベントを発行できる操作可能な UI 装置を表示してはならない
continue	Continue navigation device	このトークンが定義された場合, 対応するコンテンツオブジェクトが起動している間, LMS は *Continue* ナビゲーションイベントを発行できる操作可能な UI 装置を表示してはならない
exit	Exit navigation device	このトークンが定義された場合, 対応するコンテンツオブジェクトが起動している間, LMS は Exit ナビゲーションイベントを発行できる操作可能な UI 装置を表示してはならない
exitAll	Exit All navigation device	このトークンが定義された場合, 対応するコンテンツオブジェクトが起動している間, LMS は Exit All ナビゲーションイベントを発行できる操作可能な UI 装置を表示してはならない
abandon	Abandon navigation device	このトークンが定義された場合, 対応するコンテンツオブジェクトが起動している間, LMS は Abandon ナビゲーションイベントを発行できる操作可能な UI 装置を表示してはならない
abandonAll	Abandon All navigation device	このトークンが定義された場合, 対応するコンテンツオブジェクトが起動している間, LMS は Abandon All ナビゲーションイベントを発行できる操作可能な UI 装置を表示してはならない
suspendAll	Suspend All navigation device	このトークンが定義された場合, 対応するコンテンツオブジェクトが起動している間, LMS は Suspend All ナビゲーションイベントを発行できる操作可能な UI 装置を表示してはならない

プレゼンテーションモデルで記述された情報は，そのコンテンツオブジェクトだけに適用される．プレゼンテーションモデルの効果は，コンテンツオブジェクトが起動してから取り除かれるまでの間だけ発生する．プレゼンテーションモデルはコンテンツオブジェクトが起動していないときは LMS に影響を与えない．このとき，LMS は自由に自身で選択する UI 装置を提供する事ができる．SCORM CAM ブック[3] (セクション 5.2: プレゼンテーション/ナビゲーション情報 (*Presentation/Navigation Information*) 参照)では，SCORM コンテンツパッケージに含まれるコンテンツオブジェクトにプレゼンテーションモデルがどのように適用されるかについて説明している．

5.6.4. ナビゲーション要求のランタイム通信

SCO は学習者がナビゲーション要求を発行するための UI 装置を含むことも含まないこともある．SCO は，あるナビゲーション要求が配信対象となる学習アクティビティの特定に到るか否か，つまり，そのナビゲーション要求は有効か否かを知りたいことがある．SCO は，LMS に様々なナビゲーション要求の有効性を問い合わせることができる．この情報は，より正確に使用可能な UI 装置を提供するために使用することができる．

SCO が UI 装置を提供するかどうかに関わらず，SCO は直接 LMS にナビゲーションの意図を通信することができる．SCO は終了する際，LMS が処理するナビゲーション要求を一つだけ伝えることができる．例えば，SCO は *Previous*, *Exit* および *Choose* といったナビゲーション要求を LMS と通信できる．SCO が終了したあと，LMS は指示されたナビゲーション要求を処理し，特定された学習アクティビティを配信する．

SCORM ナビゲーションデータモデルのすべての通信は，SCORM ランタイム API (SCORM RTE ブック[4]参照)を使用して行う．ナビゲーション要求に関する SCO と LMS 間の通信は以下の表のとおり定義される．

表 5.6.4a: SCORM ナビゲーションデータモデル

No.	名称	説明	値空間	データタイプ
1	Navigation	ナビゲーション要求に関する情報	-	-
1.1	Request	SCO が終了時に LMS に処理を求めるナビゲーション要求の情報	“_none_” “continue” “previous” “choice” {target} “abandon” “abandonAll” “exit” “exitAll”	語彙(制限された)
1.2	Valid Request	ナビゲーション要求が有効かどうか示す情報	-	-
1.2.1	Continue	この要素は，*Continue* ナビゲーション要求が配信対象となるアクティビティを特定する結果になるかどうかを決めるのに使用される． **ADL :** この要素は，*Continue* ナビゲーションイベントを発行するナビゲーションコントロール用の UI 装置を SCO が学習者に提供すべきか判断するために使用される．	“true” “false” “unknown”	語彙(制限された)
1.2.2	Previous	この要素は，*Previous* ナビゲーション要求が配信対象となるアクティビティを特定する結果	“true” “false”	語彙(制限された)

		になるかどうかを決めるのに使用される. **ADL :** この要素は, *Previous* ナビゲーション イベントを発行するナビゲーションコントロール用の UI 装置を SCO が学習者に提供すべきか判断するために使用される.	"unknown"	
1.2.3	Choice {target}	この要素は, 特定のアクティビティに対する *Choice* ナビゲーション要求が配信対象となるアクティビティを特定する結果になるかどうかを決めるのに使用される. **ADL :** この要素は, 特定のアクティビティに対する *Choice* ナビゲーションイベントを発行するナビゲーションコントロール用の UI 装置を SCO が学習者に提供すべきか判断するために使用される.	"true" "false" "unknown"	語彙(制限された)

5.6.5. SCORM ランタイムナビゲーションデータモデル

以下のセクションでは, SCORM ナビゲーションデータモデルの実装に対する要求を定義する. 各データモデル要素は新たなセクションで提示される (例:5.6.6, 5.6.7 等々). 各セクションは特定のデータモデル要素に対する要求を表す表を含む. これらの要求は LMS および SCO 両方の実装に適用される. ある要求は LMS の実装に, あるものは SCO の実装に, あるものは両方に影響がある.

表 5.6.5a: データモデル要素テーブル説明

Dot-Notation Binding	詳細
<dot-notation characterstring representation of the data model element>	**データ要素実装要件:** テーブルのこのセクションは, データモデル要素実装要件を定義する. このセクションは, LMS と SCO 双方が従うべき要件の概要を記述する. このセクションはデータ型, 値空間, フォーマットの 3 つのサブセクションに別れる. • **データ型:** データモデル要素に対する特定のデータ型を記述する. これらのデータ型は, SCORM RTE ブックのデータ型セクションで定義される (セクション 4.1.1.7: SCORM RTE ブック[4]参照) • **値空間:**データ型が保持する値空間を表す • **フォーマット:** データ型の値に関するフォーマット制約を記述する **LMS 動作要件:** • このセクションは, LMS が従わなければならない要件を記述する **SCO 動作要件:** • このセクションは, SCO が従わなければならない要件を記述する **API 実装要件:** • **GetValue():** このセクションは, LMS が指定されたデータモデル要素に対する GetValue()要求を処理するとき従うべき固有の動作を記述する. また, このセクションは, GetValue()要求で指定されたデータモデル要素を使用したときに発生するエラー条件も記述する • **SetValue():** このセクションは, LMS が指定されたデータモデル要素に対する SetValue()要求を処理するとき従う

	べき固有の動作を記述する. また, このセクションは, SetValue()要求で指定されたデータモデル要素を使用したときに発生するエラー条件も記述する **追加動作要件:** • このセクションは, データモデル要素に固有の追加動作要件を記述する **例:** • このセクションは, データモデル要素を使用する有効な API メソッドの呼び出し例を提供する

5.6.6. 要求

SCO は終了時に, LMS が処理するナビゲーション要求を一つだけ提示ことができる. 例えば, SCO は *Previous*, *Exit* および *Choose* といったナビゲーション要求を LMS に送信できる. SCO が終了したあと, LMS は提示されたナビゲーション要求を処理し, 特定された学習アクティビティを配信する.

Terminate()の呼び出しに成功して LMS との通信が終了するまでは, adl.nav.request 要素を通じて LMS に送信されるナビゲーション要求は効力を持たない. SCORM ナビゲーションデータモデルは, SCO への学習セッション中にのみ有効である. これは SCO が終了するまで LMS に管理されるが, 終了した状態では維持されない.

以下のシナリオを考えてみよう.

A. 学習行為中, ユーザーがアクティビティ A に遭遇し, アクティビティ A に対応する SCO A を提示される (SCO A への学習者セッション 1):

- SCO A は adl.nav.request 要素を“continue”に設定する
- 学習者がナビゲーションイベントを発行する前に, SCO が Terminate()を呼び出す. この結果, SCO と API インスタンスの間の通信が終了する

シナリオ A の結果, *Continue* ナビゲーション要求が LMS に処理される.

B. 後に同一の学習行為中, ユーザーがアクティビティ A に再び遭遇し, SCO A が学習者に再び提示される (SCO A への学習者セッション 2):

- SCO A が直ちに GetValue(adl.nav.request)を呼び出した場合, この学習セッションでは SCO はナビゲーション要求を通信していないので “_none_”が返される
- この学習セッション中, SCO は adl.nav.request 要素を設定しない
- 学習者がナビゲーションイベントを発行する前に SCO が Terminate()を呼び出す. この結果, SCO と API インスタンスの間の通信が終了する

シナリオ B では, adl.nav.request データモデル要素は学習者セッション 1 から持続されない. つまり前回からの値“continue”を含まない. このケースでは Terminate()ではナビゲーション要求は発行されない. LMS は, ナビゲーション要求を処理する前に, 学習者によるナビゲーションイベントを待つ.

ADL ノート: コンテンツ開発者は “exit All”, “abandon All”ないし“suspend All”を発生する SCO は再利用性が限定されることに注意しなくてはならない.

表 5.6.6a: 要求データモデル要素に対する Dot-notation Binding

Dot-Notation Binding	詳細
adl.nav.request	このデータモデル要素は, SCO が, `Terminate()`の呼び出しに成功した直後に処理される望まれるナビゲーション要求を提示するために使用される **データ要素実装要件:** • **データ型:** (制限された) キャラクタ文字列 (continue, previous, choice, exit, exitAll, abandon, abandonAll, and _none_), および文字列で表されるターゲットとなる区切り文字列 (デリミタ) • **フォーマット:** キャラクタ文字列のフォーマットは以下の通り: o {target=<*STRING*>}<navigation request> デリミタ文字列`{target=<STRING>}` は, *Choice* ナビゲーション要求のターゲットを示す. もしナビゲーション要求が"choice"なら, デリミタ文字列は必須で, `SetValue()`呼び出しの parameter_2 の始めの部分の文字列でなくてはならない. (SCORM RTE Book [4]のセクション 3.1.4.2 参照). <*STRING*>の値は, 通常, アクティビティツリーが抽出されるコンテンツパッケージの`<item>`要素の識別子属性を参照する. 他のすべてのナビゲーション要求は, このデリミタ文字列を含むとエラーになる. • **値空間:** SCORM では, キャラクタ文字列に許された値は, 以下の制約された語彙トークンにバインドされている: o "continue": SCO 終了直後に *Continue* ナビゲーション要求を処理するように, コンテンツが LMS に指示したことを示す. o "previous": SCO 終了直後に *Previous* ナビゲーション要求を処理するように, コンテンツが LMS に指示したことを示す. o "choice": SCO 終了直後に *Choice* ナビゲーション要求を処理するように, コンテンツが LMS に指示したことを示す. o "exit": SCO 終了直後に *Exit* ナビゲーション要求を処理するように, コンテンツが LMS に指示したことを示す. o "exitAll": SCO 終了直後に *Exit All* ナビゲーション要求を処理するように, コンテンツが LMS に指示したことを示す. o "abandon": SCO 終了直後に *Abandon* ナビゲーション要求を処理するように, コンテンツが LMS に指示したことを示す. o "abandonAll": SCO 終了直後に *Abandon All* ナビゲーション要求を処理するように, コンテンツが LMS に指示したことを示す. o "suspendAll": SCO 終了直後に *Supend All* ナビゲーション要求を処理するように, コンテンツが LMS に指示したことを示す. o "_none_": SCO 終了直後に, SCO がこれまで提示したどのナビゲーション要求も処理しないように, コンテンツが LMS に指示したことを示す. この値を設定することですべての保留中のナビゲーション要求が消去される.

	LMS 動作要件: • このデータモデル要素は必須で, LMS は読み書き可能(read/write)に実装しなくてはならない. • 通常, 学習者は LMS が提供するナビゲーションユーザーインターフェースコントロールによりナビゲーションの要求を示す. しかし, 場合によっては, ユーザーインターフェースコントロールは SCO に組み込まれているか, もしくは, SCO が学習者に代わってナビゲーション要求を提供しようとする. 学習者が LMS の提供する UI コントロールを用いてナビゲーション要求を発行しないとき, LMS は SCO がこの要素で指定したナビゲーション要求を処理する. • SCO によって設定されていない場合, デフォルトのナビゲーション要求は "_none_"である • SCO が正常に終了(SCO が *cmi.exit* を "" もしくは "normal"へ設定する)し, 学習者が LMS の提供する UI コントロールを用いてナビゲーション要求を発行しないとき, LMS はこの要素で指定されたナビゲーション要求を, 学習者のために管理されているアクティビティツリー上で処理する. • SCO が中断状態で終了する場合(SCO が *cmi.exit* を"suspend" もしくは "logout"へ設定する), LMS はこの要素で指定されたナビゲーション要求を処理せず, 代わりに *Suspend* もしくは *SuspendAll* 要求を適宜処理する (セクション 4.2.8 *Exit* [4]参照). **SCO 動作要件:** • この要素は LMS によって読み書き可能なものとして実装される. SCO は, *adl.nav.request* データモデル要素の値を読み出し, 書き込むことができる. **API 実装要件:** • **GetValue():** o LMS は SCO に対してその時点で LMS に保存されている対応するナビゲーション要求を返し, エラーコードは"0" – No error を示す. 返された文字列は, データ要素実装要件で特定された要件に従う. o SCO が値を設定するまで, adl.*nav.request* の既定値は "_none_"である. o SCO に対して LMS がその時点で保存するナビゲーション要求が "choice"の場合, 返される文字列のフォーマットは: `{target=<STRING>}choice` ここで<*STRING*>は保留中の"choice"ナビゲーション要求のターゲットを表す. **ADL ノート**: 一般的なデリミタ文字列の文法は SCORM RTE ブック (セクション 4.1.1.6: 予約されたデリミタ文字列 (*Reserved Delimiters*) [4]参照)で定義されている. • **SetValue():** o SCO がナビゲーション要求を設定するための要求を発行し, 値が上記に示された語彙トークンの一つでないと, LMS は "false"を返し, API インスタンスのエラーコードは "406"– データモデル要素タイプミスマッチ– になる. LMS はこの要求によって要素の状態を変えてはならない. o "choice" ナビゲーション要求の際, デリミタ文字列

	{target=<*STRING*>}が指定されていない，もしくは不正確なフォーマットで記述されていると，LMS は“false”を返し，エラーコード“406”－データモデル要素タイプミスマッチ–を示す．要素の現在の状態は変えない． o “choice”以外のナビゲーション要求で，デリミタ文字列 {target=<*STRING*>}が指定されると，LMS は“false”を返し，エラーコード“406”－データモデル要素タイプミスマッチ–を示す．要素の現在の状態は変えない． **例:** • GetValue(“adl.nav.request”) • SetValue(“adl.nav.request”，“{target=intro}choice”); • SetValue(“adl.nav.request”,”continue”)

5.6.7. 要求の有効性

SCO が，学習者がナビゲーションイベントを発行するするためのユーザーインターフェース機能を提供しようとするとき，SCO はその機能をいつ有効または無効にすべきか把握できることが望ましい．この判断は，ナビゲーション要求の処理によって配信対象となるアクティビティが特定できるか否かに基づくべきである．例えば，コンテンツ設計者は，SCO が論理的な順序で存在しているときに限って"Continue"や"Next"ボタンを表示するように SCO を設計することができる．SCO 自体はナビゲーション要求の有効性に関して正確な決定を下すことはできないが，LMS はこの情報をシーケンシング機能を通して持っている．SCO は，ナビゲーション要求の有効性を確認するために SCORM ナビゲーションデータモデルを呼び出すことができる．

ADL ノート: LMS はあるナビゲーション要求が有効である事を提示できるが，これは LMS が入手できる直近の情報に基づいている．SCO が学習進捗(習得状態，スコア，等)を設定する度に，SCO は LMS に有効なナビゲーション要求を問い合わせることが推奨される．

表 5.67a: 要求有効データモデル要素に対する Dot-notation Binding

Dot-Notation Binding	詳細
adl.nav.request_valid.continue	このデータモデル要素は，*Continue* ナビゲーション要求を現在のアクティビティツリーの状態に適用した場合，配信対象となるアクティビティが特定されるかどうかを SCO が問い合わせるために使用される． **データ要素実装要件:** • **データ型:** 状態 (true, false, unknown) • **値空間:** SCORM はこれらの状態の値を以下の制限付き語彙トークンへバインドする： o “true”: LMS が，*Continue* ナビゲーション要求を現在のアクティビティツリーの状態に適用した場合，配信対象となるアクティビティが特定されると判断したことを示す． o “false”: LMS が，*Continue* ナビゲーション要求を現在のアクティビティツリーの状態に適用した場合，配信対象となるアクティビティが特定**されない**と判断したことを示す． o “unknown”: LMS が，*Continue* ナビゲーション要求を現在のアクティビティツリーの状態に適用した結果を現

	時点で評価できないことを示す. SCO は, LMS もしくは *Continue* ナビゲーション要求の処理結果にいかなる仮定もしてはならない. • **フォーマット:** データモデル値のフォーマットは, 上にあげた 3 つのトークン("true", "false", "unknown")のうち一つでなければならない. **LMS 動作要件:** • この要素は必須で, LMS は読み出し専用(read-only)として実装しなくてはならない. • LMS は, 現在のコンテンツオブジェクトを起動する前に *Continue* ナビゲーション要求の有効性を確認することが推奨される. これにより LMS は, 正確で意味のあるナビゲーションコントロールをユーザインタフェースで提供すること, および, SCO からのナビゲーション要求の有効性の確認に応えることが可能となる. • LMS は, SCO が `Commit()`要求を処理するたび, および, シーケンシングに関連したトラッキング情報 (進捗情報, 学習目標状態, 習得度, 学習目標) が更新されるたびに, *Continue* ナビゲーション要求の有効性を確認をすることが推奨される. • LMS が評価するまで, デフォルトステイタスは "unknown"でなければならない. **SCO 動作要件:** • この要素は LMS が読み出し専用(read-only) として実装する. • SCO は *adl.nav.request_valid.continue* データモデル要素の値を読み出すことができる. • `GetValue()`要求で"unknown" が返却されたら, SCO はしばらく待ち, 再度要求することが推奨される. **API 実装要件:** • **GetValue():** LMS は, 学習者に対して保持されたアクティビティツリーの現在の状態に対して, *Continue* ナビゲーション要求の有効性を確認した結果を返却し, エラーコードは"`0`" – No error を示す. 返却値は, データ要素実装要件に指定された要求に従う. • **SetValue():** SCO が `SetValue()`要求を呼び出して *adl.nav.request_valid.continue* を設定しようとしたら, LMS はエラーコードを "`404`"－データモデル要素読み出し専用－を設定し, "`false`"を返す. LMS はこの要求によって要素の状態を変えてはならない. **例:** • `GetValue("adl.nav.request_valid.continue")`
adl.nav.request_valid.previous	このデータモデル要素は, *Previous* ナビゲーション要求を現在のアクティビティツリーの状態に適用した場合, 配信対象となるアクティビティが特定されるかどうかを SCO が問い合わせるために使用される. **データ要素実装要件:** • **データ型:** 状態 (true, false, unknown) • **値空間:** SCORM はこれらの状態の値を以下の制限付き語彙トークンへバインドする: o "true": LMS が, *Previous* ナビゲーション要求を現在のアクティビティツリーの状態に適用した場合, 配信対象となるアクティビティが特定されると判断したことを示す. o "false": LMS が, *Previous* ナビゲーション要求を現在のアクティビティツリーの状態に適用した場合, 配信対

	象となるアクティビティが特定**されない**と判断したことを示す. o “unknown”: LMS が, *Previous* ナビゲーション要求を現在のアクティビティツリーの状態に適用した結果を現時点で評価できないことを示す. SCO は, LMS もしくは *Previous* ナビゲーション要求の処理結果にいかなる仮定もしてはならない. • **フォーマット:** データモデル値のフォーマットは, 上にあげた 3 つのトークン(“true”, “false”, “unknown”)のうち一つでなければならない. **LMS 動作要件:** • この要素は必須で, LMS は読み出し専用(read-only)として実装しなくてはならない. • LMS は, 現在のコンテンツオブジェクトを起動する前に *Previous* ナビゲーション要求の有効性を確認することが推奨される. これにより LMS は, 正確で意味のあるナビゲーションコントロールをユーザインタフェースで提供すること, および, SCO からのナビゲーション要求の有効性の確認に応えることが可能となる. • LMS は, SCO が `Commit()` 要求を処理するたび, および, シーケンシングに関連したトラッキング情報 (進捗情報, 学習目標状態, 習得度, 学習目標) が更新されるたびに, *Previous* ナビゲーション要求の有効性を確認をすることが推奨される. • LMS が評価するまで, デフォルトステイタスは “unknown”でなければならない. **SCO 動作要件:** • この要素は LMS が読み出し専用(read-only)として実装する. • SCO は *adl.nav.request_valid. previous* データモデル要素の値を読み出すことができる. • `GetValue()` 要求で“unknown” が返却されたら, SCO はしばらく待ち, 再度要求することが推奨される. **API 実装要件:** • **GetValue():** LMS は, 学習者に対して保持されたアクティビティツリーの現在の状態に対して, *Previous* ナビゲーション要求の有効性を確認した結果を返却し, エラーコードは“0” – No error を示す. 返却値は, データ要素実装要件に指定された要求に従う. • **SetValue():** SCO が `SetValue()` 要求を呼び出して *adl.nav.request_valid.previous* を設定しようとしたら, LMS はエラーコードを “404”－データモデル要素読み出し専用－を設定し, “false”を返す. LMS はこの要求によって要素の状態を変えてはならない. **例:** • `GetValue(“adl.nav.request_valid.previous”)`
adl.nav.request_valid.choice.{target=*STRING*}	このデータモデル要素は, *Choice* ナビゲーション要求を現在のアクティビティツリーの状態に適用した場合, 配信対象となるアクティビティが特定されるかどうかを SCO が問い合わせるために使用される. この要求のターゲットアクティビティは, ドット表記に引数として含まれるターゲットアクティビティのデリミタ文字列によって表わされる: `adl.nav.request_valid.choice.{target=<STRING>}` この引数デリミタ文字列`{target=<STRING>}`は *Choice* 有効性確認要求のターゲットを示す. この引数デリミタ文字列は必須であり,

	GetValue()を呼ぶ際のパラメータの最後の“.”の直後になくてはならない. <*STRING*>はキャラクタ文字列として表される. <*STRING*>の値は, アクティビティツリーが抽出されるコンテンツパッケージの<item>要素の識別子属性を参照する. **データ要素実装要件:** • **データ型:** 状態 (true, false, unknown) • **値空間:** SCORM はこれらの状態の値を以下の制限付き語彙トークンへバインドする: o “true”: LMS が, *Choice* ナビゲーション要求を現在のアクティビティツリーの状態に適用した場合, 配信対象となるアクティビティが特定されると判断したことを示す. o “false”: LMS が, *Choice* ナビゲーション要求を現在のアクティビティツリーの状態に適用した場合, 配信対象となるアクティビティが特定**されない**と判断したことを示す. o “unknown”: LMS が, *Choice* ナビゲーション要求を現在のアクティビティツリーの状態に適用した結果を現時点で評価できないことを示す. SCO は, LMS もしくは *Choice* ナビゲーション要求の処理結果にいかなる仮定もしてはならない. • **フォーマット:** データモデル値のフォーマットは, 上にあげた 3 つのトークン(“true”, “false”, “unknown”)のうち一つでなければならない. **LMS 動作要件:** • この要素は必須で, LMS は読み出し専用(read-only) として実装しなくてはならない. SCO は *adl.nav.request_valid. choice* データモデル要素の値を読み出すことができる. • LMS はアクティビティツリーの各アクティビティに対してこの要素を保持および管理する必要はない. LMS は, データ要素実装要件で定義された要求に対する応答を提供すればよい. LMS は, 完了した有効性確認要求を, これらの要求へのレスポンスタイムを向上するために, できるだけ長く (ツリーの状態変化がキャッシュされた結果に影響するまで) キャッシュすることが推奨されている. • LMS が評価するまで, デフォルトステイタスは “unknown”でなければならない. **SCO 動作要件:** • この要素は LMS が読み出し専用(read-only) として実装する. • SCO は *adl.nav.request_valid. choice* データモデル要素の値を読み出すことができる. • GetValue()要求で“unknown” が返却されたら, SCO はしばらく待ち, 再度要求することが推奨される. **API 実装要件:** **GetValue():** o ターゲットデリミタ文字列{target=<*STRING*>}が提供されたら, LMS は, 学習者に対して保持されたアクティビティツリーの現在の状態に対して, *Choice* ナビゲーション要求の有効性を確認した結果を返却し, エラーコードは“0” – No error を示す. 返却値は, データ要素実装要件に指定された要求に従う.

	o ターゲットデリミタ文字列{target=<*STRING*>}が提供されなければ，LMS は“false”を返却し，エラーコード“301” ―一般 Get エラー―を示す． • **SetValue():** SCO が SetValue()要求を呼び出して *adl.nav.request_valid.previous* を設定しようとしたら，LMS はエラーコードを “404”―データモデル要素読み出し専用―を設定し，“false”を返す．LMS はこの要求によって要素の状態を変えてはならない． **例:** • GetValue(“adl.nav.request_valid.choice.{target=intro}”)

付録A
略語表

このページは空白である．

略語表

ADL	Advanced Distributed Learning
AICC	Aviation Industry CBT Committee
API	Application Program Interface
ARIADNE	Alliance of Remote Instructional Authoring & Distribution Networks for Europe
CAM	Content Aggregation Model
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IEEE	International Electrical and Electronics Engineers
IMS	IMS Global Learning Consortium, Inc.
LMS	Learning Management System
OP	Overall Sequencing Process
RTE	Run-Time Environment
SCO	Sharable Content Object
SCORM	Sharable Content Object Reference Model
SN	Sequencing and Navigation
SS	Simple Sequencing
UI	User Interface
URL	Universal Resource Locator
XML	Extensible Markup Language
XSD	XML Schema Definition

このページは空白である.

付録B
参考文献

このページは空白である．

参考文献

1. *IMS Simple Sequencing Behavior and Information Model* v1.0 Final Specification, IMS Global Learning Consortium, Inc., March 2003
   Available at: http://www.imsproject.org/.

2. *SCORM 2004 3rd Edition Overview Version 1.0*, Advanced Distributed Learning, October 20, 2006
   Available at: http://www.adlnet.gov/

3. *SCORM 2004 3rd Edition Content Aggregation Model Version 1.0*, Advanced Distributed Learning, October 20, 2006
   Available at: http://www.adlnet.gov/

4. *SCORM 2004 3rd Edition Run-Time Environment Model Version 1.0*, Advanced Distributed Learning, October 20, 2006
   Available at: http://www.adlnet.gov/

このページは空白である．

付録C
シーケンシング動作擬似コード

このページは空白である．

シーケンシング動作擬似コード

この付録は，更新されたバージョンの全ての IMS SS 1.0 擬似コード[1]を含む．SCORM 対応 LMS は，シーケンシング動作を以下の擬似コードの記述のとおり実装しなければならない．コンテンツ開発者は，シーケンシング定義モデル要素および ADL 名前空間シーケンシング拡張を使用する際，実行時にはこの擬似コードに記述されたとおりに動作するとみなすべきである．

疑似コード目次

オーバーオールシーケンシングプロセス（Overall Sequencing Process）[OP.1]:		
参照: Content Delivery Environment Process DB.2; Delivery Request Process DB.1.3; Navigation Request Process NB.2.1; Sequencing Request Process SB.2.12; Termination Request Process TB.2.3		
1.	**Loop** – ナビゲーション要求を待つ	
1.1.	*Navigation Request Process*をナビゲーション要求に適用する	
1.2.	**If** *Navigation Request Process*が*Not Valid* という結果を返す**Then**	
1.2.1.	ナビゲーション要求例外を処理する	動作は規定されない
1.2.2.	**Continue Loop** – 次のナビゲーション要求を待つ	
	End If	
1.3.	**If** 終了要求がある **Then**	現アクティビティがアクティブであれば，現アクティビティの試行を終了する
1.3.1.	*Termination Request Process*を終了要求に適用する	
1.3.2.	**If** *Termination Request Process*が*Not Valid*という結果を返す **Then**	
1.3.2.1.	終了要求例外を処理する	動作は規定されない
1.3.2.2.	**Continue Loop** – 次のナビゲーション要求を待つ	
	End If	
1.3.3.	**If** *Termination Request Process*がシーケンシング要求を返す **Then**	
1.3.3.1.	保留中のシーケンシング要求を*Termination Request Process*が返したシーケンシング要求で置き換える	保留中のシーケンシング要求はひとつだけ保持される．もし*Termination Request Process*がシーケンシング要求を返せばそれを使用する
	End If	
	End If	
1.4.	**If** シーケンシング要求がある**Then**	
1.4.1.	シーケンシング要求に*Sequencing Request Process*を適用する	
1.4.2.	**If** *Sequencing Request Process*が*Not Valid*という結果を返す **Then**	
1.4.2.1.	シーケンシング要求例外を処理する	動作は規定されない
1.4.2.2.	**Continue Loop** – 次のナビゲーション要求を待つ	
	End If	
1.4.3.	**If** *Sequencing Request Process*がシーケンシングセッションを終了するという結果を返す**Then**	
1.4.3.1.	**Exit** *Overall Sequencing Process* – シーケンシングセッションを終了し，LTSへ制御を戻す	アクティビティツリーのルートから出ることでシーケ

		ンシングセッションを終え，制御をLTSに戻す
	End If	
1.4.4.	**If** *Sequencing Request Process*が配信するアクティビティを特定しない**Then**	
1.4.4.1.	**Continue Loop** – 次のナビゲーション要求を待つ	
	End If	
1.4.5.	配信要求は*Sequencing Request Process*が特定したアクティビティである	
	End If	
1.5.	**If** 配信要求があれば**Then**	
1.5.1.	*Delivery Request Process*を配信要求に適用する	
1.5.2.	**If** *Delivery Request Proces*が*Not Valid*という配信要求の結果を返す **Then**	
1.5.2.1.	配信要求例外を処理する	動作は規定されない
1.5.2.2.	**Continue Loop** – 次のナビゲーション要求を待つ	
	End If	
1.5.3.	*Content Delivery Environment Process*を配信要求に適用する	
	End If	
2.	**End Loop** – 次のナビゲーション要求を待つ	

pseudo code for overall sequencing process

ナビゲーション要求プロセス（Navigation Request Process）[NB.2.1] (ナビゲーション要求および場合によって指定されたアクティビティに対して，ナビゲーション要求の有効性を返す．終了要求，シーケンシング要求およびターゲットアクティビティを返す．例外コードを返すことがある．):		
参照: Current Activity AM.1.2; Sequencing Control Choice SM.1; Sequencing Control Choice Exit SM.1; Sequencing Control Flow SM.1; Sequencing Control Forward Only SM.1; Suspended Activity AM.1.2		
1.	**Case:**ナビゲーション要求が*Start*	
1.1.	**If** *Current Activity*が定義されていない **Then**	シーケンシングセッションがまだ始まっていない事を確認する
1.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *Start*; **ターゲットアクティビティ**: *n/a; **例外**: n/a*）	
1.2.	**Else**	
1.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-1*）	
	End If	
	End Case	
2.	**Case:**ナビゲーション要求が*Resume All*	
2.1.	**If** *Current Activity*が定義されていない **Then**	シーケンシングセッションがまだ始まっていない事を確認する
2.1.1.	**If** *Suspended Activity*が定義されている **Then**	前回のシーケンシングセッションが suspend all 要求で終了したことを確認する
2.1.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *Resume All*; **ターゲットアクティビティ**: *n/a; **例外**: n/a*）	
2.1.2.	**Else**	
2.1.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-3*）	
	End If	
2.2.	**Else**	
2.2.1.	**Exit** *Navigation Request Process*（(**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-1*）	
	End If	
	End Case	
3.	**Case:**ナビゲーション要求が*Continue*	
3.1.	**If** *Current Activity*が定義されていない**Then**	シーケンシングセッションが既に始まっている

		ことを事を確認する
3.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*;**シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; 例外: NB.2.1-2*）	
	End If	
3.2.	**If** *Current Activity*がアクティビティツリーのルートでなく**And** *Current Activity*の親の*Sequencing Control Flow*が*True* **Then**	'flow'シーケンシング要求が現アクティビティから処理することができることを確認する
3.2.1.	**If** *Current Activity*に対する*Activity is Active*が*True* **Then**	現アクティビティが終了していなければ，現アクティビティを終了する
3.2.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *Exit*; **シーケンシング要求**: *Continue*; **ターゲットアクティビティ**: *n/a; 例外: n/a*）	
3.2.2.	**Else**	
3.2.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *Continue*; **ターゲットアクティビティ**: *n/a; 例外: n/a*）	
	End If	
3.3.	**Else**	
3.3.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; 例外: NB.2.1-4*）	Flow は許されていないか現アクティビティがアクティビティツリーのルートである
	End If	
	End Case	
4.	**Case:ナビゲーション要求が***Previous*	
4.1.	**If** *Current Activity*が定義されていない **Then**	シーケンシングセッションが既に始まっている事を確認する
4.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; 例外: NB.2.1-2*）	
	End If	
4.2.	**If** *Current Activity*がアクティビティツリーのルートでない **Then**	論理的にアクティビティツリーのルートに“前の”アクティビティは存在しない

4.2.1.	**If** *Current Activity* **の親の***Sequencing Control Flow***が***True***で** **And** *Current Activity***の親の***Sequencing Control Forward Only* **が***False* **ならThen**	‘flow’シーケンシング要求が現アクティビティから処理することができることを確認する
4.2.1.1.	**If** *Current Activity***に対する***Activity is Active***が***True* **Then**	現アクティビティが終了していなければ，現アクティビティを終了する
4.2.1.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *Exit*; **シーケンシング要求**: *Previous*; **ターゲットアクティビティ**: *n/a; **例外**: n/a*）	
4.2.1.2.	**Else**	
4.2.1.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *Previous*; **ターゲットアクティビティ**: *n/a; **例外**: n/a*）	
	End If	
4.2.2.	**Else**	
4.2.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-5*）	Control モードに反する
	End If	
4.3.	**Else**	
4.3.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-6*）	アクティビティツリーのルートから後方へ移動できない
	End If	
	End Case	
5.	**Case:ナビゲーション要求が***Forward*	動作は定義されていない
5.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-7*）	
	End Case	
6.	**Case: ナビゲーション要求が***Backward*	動作は定義されていない
6.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-7*）	
	End Case	
7.	**Case:ナビゲーション要求が***Choice*	
7.1.	**If** *Choice***ナビゲーション要求が指定したアクティビティがアクティビティツリー内に存在する Then**	アクティビティツリーにターゲットアクティビ

		ティが存在する事を確認する
7.1.1.	**If** *Choice*ナビゲーション要求が指定したアクティビティがアクティビティツリーのルートである**Or** *Choice*ナビゲーション要求が指定したアクティビティの親の*Sequencing Control Choice*が*True* **Then**	'choice' シーケンシング要求がターゲットアクティビティで処理できることを確認する
7.1.1.1.	**If** *Current Activity* が定義されていない **Then**	choice を通してシーケンシングセッションが始まる
7.1.1.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *Choice*; **ターゲットアクティビティ**: *Choice* ナビゲーション要求が指定したアクティビティ; **例外**: *n/a*）	
	End If	
7.1.1.2.	**If** *Choice* ナビゲーション要求が指定したアクティビティが *Current Activity* の兄弟でない **Then**	現アクティビティの兄弟を選択することは常に許されている
7.1.1.2.1.	*Current Activity*と*Choice*ナビゲーション要求が指定したアクティビティの共通の祖先を探す	
7.1.1.2.2.	*Current Activity*から共通祖先へアクティビティの順序つき系列であるアクティビティパスを形成する	共通の祖先が現在のアクティビティでない限り，共通の祖先は choice シーケンシング要求を処理した結果終了することはない．現アクティビティは常にアクティビティパスに含まれるべきである
7.1.1.2.3.	**If** アクティビティパスが空でない **Then**	
7.1.1.2.3.1.	**For** アクティビティパスの各アクティビティ	あるアクティビティが終了する選択を許されない場合，ターゲットの選択がそのアクティビティを終了させないことを確認する
7.1.1.2.3.1.1.	**If** アクティビティの*Activity is Active* が *True*で**And** アクティビティに対する	

	*Sequencing Control Choice Exit*が*False* **Then**	
7.1.1.2.3.1.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a;* **例外**: *NB.2.1-8*）	コントロールモードに反する
	End If	
	End For	
7.1.1.2.4.	**Else**	
7.1.1.2.4.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a;* **例外**: *NB.2.1-9*）	
	End If	
	End If	
7.1.1.3.	**If** *Current Activity*の*Activity is Active*が *True* **And** *Current Activity* のシーケンシング制御*Choice Exit*が *False*なら **Then**	Choice の対象がカレントアクティビティの兄弟で，カレントアクティビティを終了できるか確かめる
7.1.1.3.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *Exit*; **シーケンシング要求**: *Choice*; **ターゲットアクティビティ**: *Choice*ナビゲーション要求が指定したアクティビティ*;* **例外**: *n/a*）	制御モード違反
	End If	
7.1.1.4.	**If** *Current Activity* の *Activity is Active* が *True* なら **Then**	現アクティビティが終了していなければ、現アクティビティを終了する
7.1.1.4.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *Choice*; **ターゲットアクティビティ**: *Choice*ナビゲーション要求が指定したアクティビティ*;* **例外**: *n/a*）	
7.1.1.5.	**Else**	
7.1.1.5.1.	**Exit** ナビゲーション要求プロセス(**ナビゲーション要求**: *Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *Choice*; **ターゲットアクティビティ**: *Choice*ナビゲーション要求が指定したアクティビティ*;* **例外**: *n/a*)	
	End If	
7.1.2.	**Else**	
7.1.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **終了要求**: *n/a*; **シーケンシング要求**: *n/a*; **ターゲットアクティビティ**: *n/a;* **例外**: *NB.2.1-10*）	コントロールモードに反する

	End If	
7.2.	**Else**	
7.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **シーケンシング要求**: *n/a*; **終了要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-11*）	**ターゲットアクティビティが存在しない**
	End If	
	End Case	
8.	**Case: ナビゲーション要求が***Exit*	
8.1.	**If** *Current Activity***が定義されている Then**	**シーケンシングセッションが始まっている事を確認する**
8.1.1.	**If** *Current Activity***に対する***Activity is Active***が***True***なら Then**	**現アクティビティがまだ終了していない事を確認する**
8.1.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *Exit*; **シーケンシング要求**: *Exit*; **ターゲットアクティビティ**: *n/a*) ; **例外**: *n/a*）	
8.1.2.	**Else**	
8.1.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **シーケンシング要求**: *n/a*; **終了要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-12*）	**アクティビティはすでに終了している**
	End If	
8.2.	**Else**	
8.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **シーケンシング要求**: *n/a*; **終了要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-2*）	
	End If	
	End Case	
9.	**Case: ナビゲーション要求が***Exit All*	
9.1.	**If** *Current Activity***が定義されている Then**	**シーケンシングセッションがすでに始まっていれば，無条件で全てのアクティブなアクティビティを終了する**
9.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *Exit All*; **シーケンシング要求**: *Exit*; **ターゲットアクティビティ**: *n/a; **例外**: n/a*）	
9.2.	**Else**	
9.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **シーケンシング要求**: *n/a*; **終了要求**: *n/a*; **ターゲットアクティビティ**: *n/a; **例外**: NB.2.1-2*）	
	End If	
	End Case	
10.	**Case:ナビゲーション要求が***Abandon*	
10.1.	**If** *Current Activity***が定義されている Then**	**シーケンシング**

		セッションがすでに始まっている事を確認する
10.1.1.	**If** *Current Activity*に対する*Activity is Active* が*True* **Then**	現アクティビティが終了していないことを確認する
10.1.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *Abandon*; **シーケンシング要求**: *Exit*; **ターゲットアクティビティ**: *n/a; 例外: n/a*）	
10.1.2.	**Else**	
10.1.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **シーケンシング要求**: *n/a*; **終了要求**: *n/a*; **ターゲットアクティビティ**: *n/a; 例外: NB.2.1-12*）	
	End If	
10.2.	**Else**	
10.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **シーケンシング要求**: *n/a*; **終了要求**: *n/a*; **ターゲットアクティビティ**: *n/a; 例外: NB.2.1-2*）	
	End If	
	End Case	
11.	**Case:** ナビゲーション要求は*Abandon All*である	
11.1.	**If** *Current Activity* が定義されている **Then**	シーケンシングセッションが始まっていれば, 無条件で全てのアクティブなアクティビティを終了する
11.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *Abandon All*; **シーケンシング要求**: *Exit*; **ターゲットアクティビティ**: *n/a; 例外: n/a*）	
11.2.	**Else**	
11.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **シーケンシング要求**: *n/a*; **終了要求**: *n/a*; **ターゲットアクティビティ**: *n/a; 例外: NB.2.1-2*）	
	End If	
	End Case	
12.	**Case:** ナビゲーション要求が*Suspend All*	
12.1.	**If** *Current Activity*が*Defined* **Then**	シーケンシングセッションがすでに始まっていたら
12.1.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Valid*; **終了要求**: *Suspend All*; **シーケンシング要求**: *Exit*; **ターゲットアクティビティ**: *n/a; 例外: n/a*）	
12.2.	**Else**	
12.2.1.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **シーケンシング要求**: *n/a*; **終了要求**: *n/a*; **ターゲット**	

	アクティビティ: *n/a;* **例外**: *NB.2.1-2*）	
	End If	
	End Case	
13.	**Exit** *Navigation Request Process*（**ナビゲーション要求**: *Not Valid*; **シーケンシング要求**: *n/a*; **終了要求**: *n/a*; **ターゲットアクティビティ**: *n/a;* **例外**: *NB.2.1-13*）	**未定義なナビゲーション要求**

pseudo code for navigation request process

終了アクションルールサブプロセス（Sequencing Exit Action Rules Subprocess）[TB.2.1] (C*urrent Activity*に対して，*Current Activity*を変更する場合がある):		
参照: Current Activity AM.1.2; End Attempt Process UP.4; Sequencing Rules Check Process UP.2; Sequencing Rule Description SM.2; Terminate Descendent Attempts Process UP.3		
1.	アクティビティツリーのルートから*Current Activity*の親アクティビティまで，両端のアクティビティを含む，順序つき系列のアクティビティパスを作る	
2.	終了ターゲットを*Null*とする	
3.	**For** アクティビティパスの各アクティビティ	アクティビティツリーのルートから始め，アクティビティパスに沿って全ての終了ルールを評価する
3.1.	アクティビティと*Exit*動作の集合に対して*Sequencing Rules Check Process*を適用する	
3.2.	**If** *Sequencing Rules Check Process*が*Nil*を返さない **Then**	
3.2.1.	終了ターゲットをアクティビティとする	終了ルールの評価がTrueである最初のアクティビティで止める
3.2.2.	**Break For**	
	End If	
	End For	
4.	**If** 終了ターゲットが*Null*でない **Then**	
4.1.	終了ターゲットに*Terminate Descendent Attempts Process*を適用する	全てのアクティブな下位アクティビティの現在の試行を終了する
4.2.	終了ターゲットに*End Attempt Process*を適用する	'終了'アクティビティの現在の試行を終了する
4.3.	*Current Activity*を終了ターゲットに設定する	現アクティビティを終了し，指定されたアクティビティへ移動する
	End If	
5.	**Exit** *Sequencing Exit Action Rules Subprocess*	

ポストコンディションルールシーケンシングサブプロセス（Sequencing Post Condition Rules Subprocess）[TB.2.2] (*Current Activity*に対して，終了要求とシーケンシング要求を返す):		
参照: Activity is Suspended AM.1.1; Current Activity AM.1.2; Sequencing Rules Check Process UP.2; Sequencing Rule Description SM.2		
1.	**If** C*urrent Activity*の*Activity is Suspended*が*True* **Then**	中断したアクティビティにはポストコンディションルールを適用しない
1.1.	**Exit** *Sequencing Post Condition Rules Subprocess*	
	End If	
2.	*Current Activity*と*Post Condition*動作の集合に対して*Sequencing Rules Check Process*を適用する	現アクティビティにポストコンディションルールを適用する
3.	**If** *Sequencing Rules Check Process*が*Nil*を返さない **Then**	
3.1.	**If** *Sequencing Rules Check Process*が*Retry, Continue,*ないし *Previous* を返す **Then**	
3.1.1.	**Exit** *Sequencing Post Condition Rules Subprocess*（**シーケンシング要求**: *Sequencing Rules Check Process*の返却値; **終了要求**: *n/a*）	保留中のシーケンシング要求をこれで上書きする
	End If	
3.2.	**If** *Sequencing Rules Check Process*が*Exit Parent* ないし *Exit All*を返す **Then**	
3.2.1.	**Exit** *Sequencing Post Condition Rules Subprocess*（**シーケンシング要求**: *n/a*; **終了要求**: *Sequencing Rules Check Process*の返却値）	適切なアクティビティを終了する
	End If	
3.3.	**If** *Sequencing Rules Check Process*が*Retry All*を返せば **Then**	
3.3.1.	**Exit** *Sequencing Post Condition Rules Subprocess*（**終了要求**: *Exit All*; **シーケンシング要求**: *Retry*）	全てのアクティブなアクティビティを終了し，現アクティビティをアクティビティツリーのルートへ移動する．そして”実行中の”startを行う
	End If	
	End If	
4.	**Exit** *Sequencing Post Condition Rules Subprocess*（**シーケンシング要求**: *n/a*; **終了要求**: *n/a*）	

終了要求プロセス（Termination Request Process）[TB.2.3] (終了要求に対して，*Current Activity*の現在の試行を終了，終了要求の有効性を返却し，場合によりシーケンシング要求を返却する，例外コードを返すことがある):		
参照: Activity is Active AM.1.1; Activity is Suspended AM.1.1; Current Activity AM.1.2; End Attempt Process UP.4; Sequencing Exit Action Rules Subprocess TB.2.1; Sequencing Post Condition Rules Subprocess TB.2.2; Terminate Descendent Attempts Process UP.3		
1.	**If** *Current Activity*が定義されていない **Then**	シーケンシングセッションが始まっていない場合，終了するものはない
1.1.	**Exit** *Termination Request Process*（**終了要求**: *Not Valid;* **シーケンシング要求**: *n/a;* **例外**: *TB.2.3-1*）	
	End If	
2.	**If** (終了要求が*Exit* ないし *Abandon*なら) **And** *Current Activit*の *Activity is Active*が*False* **Then**	現アクティビティがすでに終了している場合，終了するものはない
2.1.	**Exit** *Termination Request Process*（**終了要求**: *Not Valid;* **シーケンシング要求**: *n/a;* **例外**: *TB.2.3-2*）	
	End If	
3.	**Case:** 終了要求が*Exit*	
3.1.	*Current Activity*へ*End Attempt Process*を適用する	現アクティビティの状態更新を確実にする
3.2.	*Current Activity*へ*Sequencing Exit Action Rules Subprocess*を適用する	現アクティビティの祖先を終了させる必要があるか確認する
3.3.	**Repeat**	
3.3.1.	終了処理済みを*False*に設定する	
3.3.2.	*Current Activity*へ*Sequencing Post Condition Rules Subprocess*を設定する	
3.3.3.	**If** *Sequencing Post Condition Rule Subprocess*が*Exit All*終了要求を返す **Then**	
3.3.3.1.	終了要求を*Exit All*へ変更する	
3.3.3.2.	**Break** 次の**Case**へ	Exit All 終了要求を処理する
	End If	
3.3.4.	**If** *Sequencing Post Condition Rule Subprocess*が*Exit Parent*終了要求を返す **Then**	現アクティビティの親を終了する場合，現アクティビティを現アクティビティの親へ移動する
3.3.4.1.	**If** *Current Activity*がアクティビティツリーのルートでない **Then**	アクティビティツリーのルート

		は終了する親を持たない
3.3.4.1.1.	*Current Activity*を*Current Activity*の親に設定する	
3.3.4.1.2.	*End Attempt Process*を*Current Activity*に適用する	
3.3.4.1.3.	終了処理済みを*True*に設定する	新しい現アクティビティのポストコンディションを評価する必要がある
3.3.4.2.	**Else**	
3.3.4.2.1.	**Exit** *Termination Request Process*（**終了要求**: *Not Valid;* **シーケンシング要求**: *n/a;* **例外**: *TB.2.3-4*）	
	End If	
3.3.5.	**Else**	
3.3.5.1.	**If** *Current Activity* がアクティビティツリーのルートで **And** 事後条件シーケンシングルールサブプロセスが返却したシーケンシング要求が *Retry* でなければ **Then**	アクティビティツリーのルートへの試行がRetryで終了したのでなければシーケンシングセッションを終了する
3.3.5.1.1.	**Exit** 終了要求プロセス (**終了要求**: *Valid;* **シーケンシング要求**: *Exit;* **例外**: *n/a*)	
	End If	
	End If	
3.4.	**Until** 終了処理済みが*False*である	
3.5.	**Exit** *Termination Request Process*（**終了要求**: *Valid*; **シーケンシング要求**:もし存在すれば*Sequencing Post Condition Rule Subprocess* の返却値．そうでなければ*n/a;* **例外**: *n/a*）	
	End Case	
4.	**Case:** 終了要求が*Exit All*	
4.1.	**If** *Current Activity*の*Activity is Active*が*True* **Then**	完了サブプロセスおよびロールアップは既に現在のアクティビティに適用されているか？
4.1.1.	*Current Activity*へ*End Attempt Process*を適用する	
	End If	
4.2.	アクティビティツリーのルートへ*Terminate Descendent Attempts Process*を適用する	
4.3.	アクティビティツリーのルートへ*End Attempt Process*を適用する	
4.4.	*Current Activity*をアクティビティツリーのルートへ設定する	現在のアクティビティをアクティビティツリーのルートに移動する
4.5.	**Exit** 終了要求プロセス (**終了要求**: *Valid*; **シーケンシング要求**: *Exit;* **例外**: *n/a*)	シーケンシングセッションが終

		了したことをシーケンサーに通知する
	End Case	
5.	**Case: 終了要求が***Suspend All*	
5.1.	**If** (*Current Activity*の*Activity is Active*が*True*) **ないし** (*Current Activit*の*Activity is Suspended*が*True*) **Then**	**現アクティビティがアクティブもしくは既に中断されている場合，アクティビティとその後続要素全てを中断する**
5.1.1.	**オーバオールロールアッププロセスをアクティビティに適用する**	**このアクティビティへのいかなる状態変化もアクティビティツリー全体に確実に伝搬させる**
5.1.2.	*Suspended Activity*を*Current Activity*に設定する	
5.2.	**Else**	
5.2.1.	**If** *Current Activity*がアクティビティツリーのルートでない **Then**	**現在のアクティビティはアクティビティツリーのルートではないことを確認する**
5.2.1.1.	*Suspended Activity*を*Current Activity*の親へ設定する	
5.2.2.	**Else**	
5.2.2.1.	**Exit** *Termination Request Process*（**終了要求**: *Not Valid;* **シーケンシング要求**: *n/a;* **例外**: *TB.2.3-3*）	**中断するものはない**
	End If	
	End If	
5.3.	*Suspended Activity*からアクティビティツリーのルートまで，両端のアクティビティを含む，順序つき系列のアクティビティパスを形成する	
5.4.	**If** アクティビティパスが空 **Then**	
5.4.1.	**Exit** *Termination Request Process*（**終了要求**: *Not Valid;* **シーケンシング要求**: *n/a;* **例外**: *TB.2.3-5*）	**中断するものはない**
	End If	
5.5.	**For** アクティビティパスの各アクティビティ	
5.5.1.	アクティビティの*Activity is Active*を*False*へ設定する	
5.5.2.	アクティビティの*Activity is Suspended*を*True*へ設定する	
	End For	
5.6.	*Current Activity*をアクティビティツリーのルートに設定する	**現在のアクティビティをアクティビティツリーのルートへ移動する**

5.7.	**Exit** *Termination Request Process*（**終了要求**: *Valid*; **シーケンシング要求**: *Exit;* **例外***: n/a*）	シーケンシングセッションが終了した事をシーケンサーに知らせる
	End Case	
6.	**Case: 終了要求が***Abandon*	
6.1.	*Current Activity*の*Activity is Active*を*False*へ設定する	
6.2.	**Exit** *Termination Request Process*（**終了要求**: *Valid;* **シーケンシング要求**: *n/a;* **例外***: n/a*）	
	End Case	
7.	**Case: 終了要求が***Abandon All*	
7.1.	*Current Activity*からアクティブティツリーのルートまで，両端のアクティビティを含む，順序つき系列のアクティビティパスを形成する	
7.2.	**If** アクティビティパスが空 **Then**	
7.2.1.	**Exit** *Termination Request Process*（**終了要求**: *Not Valid;* **シーケンシング要求**: *n/a;* **例外***: TB.2.3-6*）	破棄するものはない
	End If	
7.3.	**For** アクティビティパスの各アクティビティ	
7.3.1.	アクティビティの*Activity is Active*を*False*へ設定する	
	End For	
7.4.	*Current Activity*をアクティビティツリーのルートへ設定する	現在のアクティビティをアクティビティツリーのルートへ移動する
7.5.	**Exit** *Termination Request Process*（**終了要求**: *Valid*; **シーケンシング要求**: *Exit;* **例外***: n/a*）	シーケンシングセッションが終了したことをシーケンサーに通知する
	End Case	
8.	**Exit** *Termination Request Process*（**終了要求**: *Not Valid;* **シーケンシング要求**: *n/a;* **例外***: TB.2.3-7*）	未定義の終了要求

pseudo-code for termination request process

習得度ロールアッププロセス（Measure Rollup Process）[RB.1.1] (アクティビティに対して; アクティビティの*Objective Information*を変更することがある):		
Reference: Objective Contributes to Rollup SM.6; Objective Description SM.6; Objective Measure Status TM.1.1; Objective Normalized Measure TM.1.1; Rollup Objective Measure Weight SM.8; Tracked SM.11		
1.	**重み付き習得度総和を** 0.0 **に設定する**	
2.	**有効データを***False***に設定する**	
3.	**計算対象習得度を** 0.0 **に設定する**	
4.	**ターゲット学習目標を未定義に設定する**	
5.	**For アクティビティに付随する各学習目標**	
5.1.	**If 学習目標の***Objective Contributes to Rollup***が***True***ならThen**	**習得度ロールアップのターゲット学習目標を見つける**
5.1.1.	**ターゲット学習目標を学習目標に設定する**	
5.1.2.	**Break For**	
	End If	
	End For	
6.	**Ifターゲット学習目標が定義されているならThen**	
6.1.	**For各子アクティビティ**	
6.1.1.	**If 子アクティビティの***Tracked* **が***True***ならThen**	**トラッキングされた子だけを対象とする**
6.1.1.1.	**ロールアップ学習目標を未定義に設定する**	
6.1.1.2.	**For 子アクティビティに付随する各学習目標**	
6.1.1.2.1.	**If 学習目標の***Objective Contributes to Rollup***が***True***ならThen**	
6.1.1.2.1.1.	**ロールアップ学習目標を学習目標へ設定する**	
6.1.1.2.1.2.	**Break For**	
	End If	
	End For	
6.1.1.3.	**Ifロールアップ学習目標が定義されているならThen**	
6.1.1.3.1.	**計算対象習得度を子の***Rollup Objective Measure Weight***だけ増加する**	
6.1.1.3.2.	**If ロールアップされた学習目標に対する***Objective Measure Status***が***True***ならThen**	
6.1.1.3.2.1.	**ロールアップ学習目標の***Objective Normalized Measure***と子アクティビティの***Rollup Objective Measure Weight***の積を重み付き習得度総和に加える**	
6.1.1.3.2.2.	**有効データを***True***に設定する**	
	End If	
6.1.1.4.	**Else**	
6.1.1.4.1.	**Exit 習得度ロールアッププロセス**	**ロールアップ学習目標をもたない子アクティビティがある**
	End If	
	End If	

	End For	
6.2.	**If 有効データが *False* なら Then**	
6.2.1.	**対象学習目標の *Objective Measure Status* を *False* に設定する**	**トラッキング状態がロールアップされず，ロールアップ習得度を決定できない**
6.3.	**Else**	
6.3.1.	**If 計算対象習得度が 0.0 より大きければThen**	**対象学習目標のロールアップ習得度を設定する**
6.3.1.1.	**ターゲット学習目標の*Objective Measure Status*を*True*へ設定する**	
6.3.1.2.	**ターゲット学習目標の*Objective Normalized Measure*を重み付き習得度総和を計算対象習得度で割った値に設定する**	
6.3.2.	**Else**	
6.3.2.1.	**ターゲット学習目標の*Objective Measure Status*を*False*へ設定する**	**重みに寄与する子が無い**
	End If	
	End If	
	End If	
7.	**Exit 習得度ロールアッププロセス**	**ロールアップに貢献する学習目標がないので何も設定できない**

pseudo-code for measure rollup process

学習目標習得度ロールアッププロセス（Objective Rollup Using Measure Process）[RB.1.2 a] (アクティビティに対して;アクティビティの*Objective Information*を変更する場合がある):		
参照: Objective Contributes to Rollup SM.6; Objective Description SM.6; Objective Satisfied by Measure SM.6; Objective Measure Status TM.1.1; Objective Normalized Measure TM.1.1; Objective Progress Status TM.1.1; Objective Satisfied Status TM.1.1; Activity is Active AM.1.1; *adlseq:measureSatisfactionIfActive SCORM SN.*		
1.	**ターゲット学習目標を未定義に設定する**	
2.	**For アクティビティに付随する各学習目標**	
2.1.	**If 学習目標の***Objective Contributes to Rollup***が***True* **Then**	**アクティビティの子のロールアップ習得度によって変更される可能性のある学習目標を特定する**
2.1.1.	**ターゲット学習目標を学習目標へ設定する**	
2.1.2.	**Break For**	
	End If	
	End For	
3.	**If ターゲット学習目標が定義されている Then**	
3.1.	**If ターゲット学習目標の***Objective Satisfied by Measure***が***True* **Then**	**学習目標が習得度で習得となる場合，定義された閾値を比較する**
3.1.1.	**If ターゲット学習目標の***Objective Measure Status***が***False* **Then**	**習得度が不明なので，学習目標ステイタスは信頼できない**
3.1.1.1.	**ターゲット学習目標の***Objective Progress Status* **を***False***へ設定する**	
3.1.2.	**Else**	
3.1.2.1.	**If アクティビティの***Activity is Active***が***False* **Or （アクティビティの***Activity is Active***が***True* **And アクティビティの***adlseq:measureSatisfactionIfActive***が***True*）**Then**	
3.1.2.1.1.	**If ターゲット学習目標の***Objective Normalized Measure***がターゲット学習目標の***Objective Minimum Satisfied Normalized Measure***に等しいか大きい Then**	
3.1.2.1.1.1.	**ターゲット学習目標の***Objective Progress Status***を***True***へ設定する**	
3.1.2.1.1.2.	**ターゲット学習目標の***Objective Satisfied Status***を***True***へ設定する**	
3.1.2.1.2.	**Else**	
3.1.2.1.2.1.	**ターゲット学習目標の***Objective Progress Status* **を***True***へ設定する**	
3.1.2.1.2.2.	**ターゲット学習目標の***Objective Satisfied Status* **を***False***へ設定する**	
	End If	

3.1.2.2.	**Else**	
3.1.2.2.1.	**ターゲット学習目標の** *Objective Progress Status* **を** *False* **へ設定する**	**不完全な情報. 学習目標のステイタスを評価できない**
	End If	
	End If	
	End If	
3.2.	**Exit** *Objective Rollup Using Measure Process*	
4.	**Else**	
4.1.	**Exit** *Objective Rollup Using Measure Process*	**ロールアップに関与する学習目標がないので何も設定できない**
	End If	

pseudocode for objective rollup using measure process

学習目標ルールロールアッププロセス（Objective Rollup Using Rules Process）[RB.1.2 b] (アクティビティに対して;アクティビティの*Objective Information*を変更する場合がある):		
参照: Section: Objective Contributes to Rollup SM.6; Objective Description SM.6; Objective Progress Status TM.1.1; Objective Satisfied Status TM.1.1; Rollup Rule Check Subprocess RB.1.4; Rollup Action SM.5		
1.	**ターゲット学習目標を未定義に設定する**	
2.	**For アクティビティに付随する各学習目標**	
2.1.	**If 学習目標の***Objective Contributes to Rollup***が***True* **Then**	**アクティビティの子のロールアップ状態に基づいて変更される可能性のある学習目標を特定する**
2.1.1.	**ターゲット学習目標を学習目標へ設定する**	
2.1.2.	**Break For**	
	End If	
	End For	
3.	**If ターゲット学習目標が定義されている Then**	
3.1.	***Rollup Rule Check Subprocess*をアクティビティの*Not Satisfied*ロールアップアクションに対して適用する**	**全ての**Not Satisfied**ルールを最初に処理する**
3.2.	**If** *Rollup Rule Check Subprocess***が***True***を返す Then**	
3.2.1.	**ターゲット学習目標の***Objective Progress Status***を***True***に設定する**	
3.2.2.	**ターゲット学習目標の***Objective Satisfied Status***を***False***に設定する**	
	End If	
3.3.	***Rollup Rule Check Subprocess*をアクティビティの*Satisfied* ロールアップアクションに対して適用する**	**全ての**Satisfied **ルールを最後に処理する**
3.4.	**If** *Rollup Rule Check Subprocess***が***True***を返す Then**	
3.4.1.	**ターゲット学習目標の***Objective Progress Status***を***True***に設定する**	
3.4.2.	**ターゲット目標に対する***Objective Satisfied Status***を***True***に設定する**	
	End If	
3.5.	**Exit** *Objective Rollup Using Rules Process*	
4.	**Else**	
4.1.	**Exit** *Objective Rollup Using Rules Process*	**ロールアップに関与する学習目標がないので何も設定できない**
	End If	

アクティビティ進捗ロールアッププロセス（Activity Progress Rollup Process）[RB.1.3] (アクティビティに対して;アクティビティの*Attempt Information*を変更する場合がある):		
参照: Attempt Completion Status TM.1.2.2; Attempt Progress Status TM.1.2.2; Rollup Rule Check Subprocess RB.1.4; Rollup Action SM.5		
1.	アクティビティと*Incomplete*ロールアップアクションに対して*Rollup Rule Check Subprocess*を適用する	全ての Incomplete ルールを最初に処理する
2.	**If** *Rollup Rule Check Subprocess*が*True*を返す **Then**	
2.1.	アクティビティの*Attempt Progress Status*を*True*に設定する	
2.2.	アクティビティの*Attempt Completion Status*を*False*に設定する	
	End If	
3.	アクティビティと*Completed* ロールアップアクションに対して*Rollup Rule Check Subprocess*を適用する	全ての Completedルールを最後に処理する
4.	**If** *Rollup Rule Check Subprocess*が*True*を返す **Then**	
4.1.	アクティビティの*Attempt Progress Status*を*True*に設定する	
4.2.	アクティビティの*Attempt Completion Status*を*True*に設定する	
	End If	
5.	**Exit** *Activity Progress Rollup Process*	

ロールアップルールチェックサブプロセス（Rollup Rule Check Subprocess）[RB.1.4] (アクティビティと*Rollup Action*に対して; アクションが適用する場合Trueを返す):		
参照: Check Child for Rollup Subprocess RB.1.4.2; Evaluate Rollup Conditions Subprocess RB.1.4.1; Rollup Action SM.5; Rollup Child Activity Set SM.5; Rollup Minimum Count SM.5; Rollup Minimum Percent SM.5; Rollup Rule Description SM.5; Tracked SM.11; Tracking Model TM		
1.	**If アクティビティが指定された*Rollup Action*の*Rollup Rules*を持つ Then**	**アクティビティが評価対象ルールを有することを確認する**
1.1.	**指定された*Rollup Action*を持つアクティビティの*Rollup Rules*を選び，ルールの順番を保って，ルールリストを初期化する**	
1.2.	**For リスト中の各ルール**	
1.2.1.	**関与する子の集合を空集合に初期化する**	
1.2.2.	**For アクティビティの各子アクティビティ**	
1.2.2.1.	**If 子の*Tracked* が*True* Then**	
1.2.2.1.1.	**子とロールアップアクションに対してロールアップ子チェックサブプロセスを適用する**	**この子が親の状態に関与する事を確認する**
1.2.2.1.2.	**If ロールアップ子チェックサブプロセスが*True*を返せばThen**	
1.2.2.1.2.1.	**子とルールの*Condition Combination*と*Rollup Conditions*に対してロールアップ条件評価サブプロセスを適用する**	**子アクティビティに対してヘロールアップ条件を評価する**
1.2.2.1.2.2.	**If ロールアップ条件評価サブプロセスが *Unknown* を返せば Then**	‘unknown’ **条件評価を説明する**
1.2.2.1.2.2.1.	**関与する子の集合に*Unknown*値を追加する**	
1.2.2.1.2.3.	**Else**	
1.2.2.1.2.3.1.	**If ロールアップ条件評価サブプロセス が *True*を返せばThen**	
1.2.2.1.2.3.1.1.	**関与する子の集合に*True*値 を追加する**	
1.2.2.1.2.3.2.	**Else**	
1.2.2.1.2.3.2.1.	**関与する子の集合に*False*値を追加する**	
	End If	
	End If	
	End If	
	End If	
	End For	
1.2.3.	**状態変更を*False*へ初期化する**	**適切な子がロールアップに関与するか否かを決定する; そうなら，アクティビティの状態を変**

		更する
1.2.4.	**Case:** *Rollup Child Activity Set* は*All*である	
1.2.4.1.	**If** 関与する子の集合が*False* **Or** *Unknown* を含まない **Then**	
1.2.4.1.1.	状態変更を*True*に変更する	
	End If	
	End Case	
1.2.5.	**Case:** *Rollup Child Activity Set* は*Any*である	
1.2.5.1.	**If** 関与する子の集合が*True*を含む **Then**	
1.2.5.1.1.	状態変更を*True*へ変更する	
	End If	
	End Case	
1.2.6.	**Case:** *Rollup Child Activity Set*は *None*である	
1.2.6.1.	**If** 関与する子の集合が*True* **Or** *Unknown*を含まない **Then**	
1.2.6.1.1.	状態変更を*True*へ変更する	
	End If	
	End Case	
1.2.7.	**Case:** *Rollup Child Activity Set*は*At Least Count*である	
1.2.7.1.	**If** 貢献関与する子の集合の*True*の個数がルールの*Rollup Minimum Count*と等しいか大きい **Then**	
1.2.7.1.1.	状態変更をTrueへ変更する	
	End If	
	End Case	
1.2.8.	**Case:** *Rollup Child Activity Set*は*At Least Percent*である	
1.2.8.1.	**If** 関与する子の集合に含まれる*True*の個数の（0と1の間に正規化された）割合がルールの*Rollup Minimum Percent*と等しいか大きい **Then**	
1.2.8.1.1.	状態変更を*True*へ変更する	
	End If	
	End Case	
1.2.9.	**If** 状態変更が*True* **Then**	
1.2.9.1.	**Exit** *RollupRule Check Subprocess*（**評価値**: *True*）	真と評価された最初のルールで停止 – 関連するアクションを実行する
	End If	
	End For	
	End If	
2.	**Exit** *Rollup Rule Check Subprocess*（**評価値**: *False*）	trueと評価されるルールはない – アクションを実行しない

ロールアップコンディション評価サブプロセス（Evaluate Rollup Conditions Subprocess） **[RB.1.4.1]** (アクティビティとロールアップコンディション集合に対して;コンディションが真と評価されれば*True*を返し，コンディションが偽と評価されさば*False*を返し，コンディションが評価できなければ*Unknown*を返す):		
参照: Section: Condition Combination SM.5; Rollup Condition SM.5; Rollup Condition Operator SM.5; Tracking Model TM		
1.	**ロールアップコンディション集合を空集合に初期化する**	**ルールのコンディションの評価を記録するために使用される**
2.	**For ロールアップコンディション集合中の各***Rollup Condition*	
2.1.	**アクティビティの適切なトラッキング情報を***Rollup Condition***に適用してロールアップコンディションを評価する**	**アクティビティのトラッキング情報に対して各コンディションを評価する．この評価は**‘unknown’**に終わることがある**
2.2.	**If** *Rollup Condition***の***Rollup Condition Operator***が***Not* **Then**	‘unknown’**を否定しても**‘unknown’**に終わる**
2.2.1.	**Negate ロールアップコンディション**	
	End If	
2.3.	**ロールアップコンディションの値をロールアップコンディション集合に追加する**	**評価されたコンディションの集合にこのコンディションの評価を追加する**
	End For	
3.	**If ロールアップコンディション集合が空 Then**	**ルールに対して定義されたコンディションがなければ，ルールが適用されない**
3.1.	**Exit** *Evaluate Rollup Conditions Subprocess*（**評価**: *Unknown*）	
	End If	
4.	*Condition Combination***をロールアップコンディション集合に適用し単一のルール結合評価を生成する**	**ロールアップルール定義に基づいて，評価済みコンディションの**‘And’**もしくは**‘Or’**集合**
5.	**Exit** *Evaluate Rollup Conditions Subprocess*（**評価**:**ルール結合評価の値**）	

pseudo-code for evaluate rollup conditions subprocess

ロールアップ子チェックサブプロセス（Check Child for Rollup Subprocess）[RB.1.4.2] (アクティビティと*Rollup Action*に対して; アクティビティがロールアップに含まれれば Trueを返す):		
参照: *Rollup Action* SM.5; Rollup Objective Satisfied SM.8; Rollup Progress Completion SM.8; Activity Attempt Count TM.1.2.1; Sequencing Rules Check Process UP.2; *adlseq:requiredForSatisfied SCORM SN*; *adlseq:requiredForNotSatisfied SCORM SN*; *adlseq:requiredForCompleted SCORM SN*; *adlseq:requiredForIncomplete SCORM SN*		
1.	*False*に設定する	
2.	**If** *Rollup Action*が*Satisfied* **Or** *Not Satisfied* **Then**	
2.1.	**If** アクティビティの*Rollup Objective Satisfied*値が*True* **Then**	**学習目標ロールアップコントロールをテストする**
2.1.1.	*True*に設定する	**デフォルト動作** – adlseq:requiredFor[xxx] == always
2.1.2.	**If** (*Rollup Action*が*Satisfied* **And** *adlseq:requiredForSatisfied*が*ifNotSuspended*) **Or** (*Rollup Action*が*Not Satisfied* **And** *adlseq:requiredForNotSatisfied*が*ifNotSuspended*) **Then**	
2.1.2.1.	**If** アクティビティの*Activity Attempt Count*がゼロより大きい（*(>) Zero (0)*）**And** アクティビティの*Activity is Suspended*が*True* **Then**	
2.1.2.1.1.	*False*に設定する	
	End If	
2.1.3.	**Else**	
2.1.3.1.	**If** (*Rollup Action*が*Satisfied* **And** *adlseq:requiredForSatisfied*が*ifAttempted*) **Or** (*Rollup Action*が*Not Satisfied* **And** *adlseq:requiredForNotSatisfied*が*ifAttempted*) **Then**	
2.1.3.1.1.	**If** アクティビティの*Activity Progress Status*が*False* **Or**アクティビティの*Activity Attempt Count*がゼロ*(0)*ならThen	
2.1.3.1.1.1.	*False*に設定する	
	End If	
2.1.3.2.	**Else**	
2.1.3.2.1.	**If** (*Rollup Action*が*Satisfied* **And** *adlseq:requiredForSatisfied*が*ifNotSkipped*) **Or** (*Rollup Action*が*Not Satisfied* **And** *adlseq:requiredForNotSatisfied*が*ifNotSkipped*) **Then**	
2.1.3.2.1.1.	アクティビティとその*Skipped*シーケンシングルールに*Sequencing Rules Check Process*を適用する	
2.1.3.2.1.2.	**If** *Sequencing Rules Check Process*が*Nil*を返さない Then	
2.1.3.2.1.2.1.	*False*に設定する	
	End If	
	End If	
	End If	
	End If	
	End If	
	End If	

3.	**If** *Rollup Action*が*Completed* **Or** *Incomplete* **Then**	
3.1.	**If** アクティビティの*Rollup Progress Completion*値が*True* **Then**	**進捗ロールアップコントロールをテストする**
3.1.1.	*True*に設定する	**デフォルト動作** – adlseq:requiredFor[xxx] == always
3.1.2.	**If** (*Rollup Action*が*Completed* **And** *adlseq:requiredForCompleted*が*ifNotSuspended*) **Or** (*Rollup Action*が*Incomplete* **And** *adlseq:requiredForIncomplete*が*ifNotSuspended*)**ならThen**	
3.1.2.1.	**If** アクティビティの*Activity Attempt Count*がゼロより大きい（*(>) Zero (0)*） **And** アクティビティの*Activity is Suspended*が*True* **Then**	
	*False*に設定する	
	End If	
3.1.3.	**Else**	
3.1.3.1.	**If** (*Rollup Action*が*Completed* **And** *adlseq:requiredForCompleted*が*ifAttempted*) **Or** (*Rollup Action*が*Incomplete* **And** *adlseq:requiredForIncomplete*が*ifAttempted*) **Then**	
3.1.3.1.1.	**If** アクティビティの*Activity Progress Status*が*False* **Or**アクティビティの*Activity Attempt Count*がゼロ*(0)* **ならThen**	
3.1.3.1.1.1.	*False*に設定する	
	End If	
3.1.3.2.	**Else**	
3.1.3.2.1.	**If** (*Rollup Action*が*Completed* **And** *adlseq:requiredForCompleted*が*ifNotSkipped*) **Or** (*Rollup Action*が*Incomplete* **And** *adlseq:requiredForIncomplete*が*ifNotSkipped*) **Then**	
3.1.3.2.1.1.	アクティビティとその*Skipped* シーケンシングルールに*Sequencing Rules Check Process*を適用する	
3.1.3.2.1.2.	**If** *Sequencing Rules Check Process*が*Nil*を返さない Then	
3.1.3.2.1.2.1.	*False*にセット	
	End If	
	End If	
	End If	
	End If	
	End If	
	End If	
4.	**Exit** *Check Child for Rollup Subprocess*（**子をロールアップに含める**: 含める）	

pseudo code for check child for rollup subprocess

オーバーオールロールアッププロセス（Overall Rollup Process）[RB.1.5] (アクティビティに対して; アクティビティとその祖先のトラッキング情報を変更する場合がある):		
Reference: Activity Progress Rollup Process RB.1.3; Measure Rollup Process RB.1.1; Objective Rollup Process RB.1.2; Tracked SM.11; Tracking Model TM		
1.	**アクティビティツリーのルートからアクティビティまで，両端のアクティビティを含む逆順に順序付き系列のアクティビティパスを作る**	
2.	**If アクティビティパスが空 Then**	
2.1.	**Exit** *Overall Rollup Process*	**ロールアップするものがない**
	End If	
3.	**For アクティビティパスの各アクティビティ**	
3.1.	**Ifアクティビティに子があれば Then**	**習得度ロールアッププロセスを葉でないアクティビティにのみ適用する**
3.1.1.	**アクティビティに対して習得度ロールアッププロセスを適用する**	
	End If	
3.2.	**アクティビティに対して適切な*Objective Rollup Process*を適用する**	**アクティビティに定義されたシーケンシング情報に応じて，セクション RB.1.2に記述された適切な動作を適用する**
3.3.	**アクティビティに対して*Activity Progress Rollup Process*を適用する**	**アクティビティに定義されたシーケンシング情報に応じて，セクション RB.1.3に記述された適切な動作を適用する**
	End For	
4.	**Exit** *Overall Rollup Process*	

pseudo-code for overall rollup process

子選択プロセス（Select Children Process）[SR.1] (アクティビティに対して;アクティビティの *Available Children*を変更することがある):		
参照: Activity is Active AM.1.1; Activity is Suspended AM.1.1; Available Children AM.1.1; Activity Progress Status TM.1.2.1; Selection Count SM.9; Selection Count Status SM.9; Selection Timing SM.9		
1.	**If** アクティビティが子を持たない **Then**	葉アクティビティに選択は適用できない
1.1.	**Exit** *Select Children Process*	
	End If	
2.	**If** アクティビティの*Activity is Suspended* が*True* **Or** アクティビティの*Activity is Active*が*True* **Then**	中断しているかアクティブなアクティビティに選択は適用できない
2.1.	**Exit** *Select Children Process*	
	End If	
3.	**Case:** アクティビティの*Selection Timing*が*Never*である	
3.1.	**Exit** *Select Children Process*	
	End Case	
4.	**Case:** アクティビティの*Selection Timing*が*Once*である	
4.1.	**If** アクティビティの*Activity Progress Status*が*False* **Then**	アクティビティがまだ試行されていない場合
4.1.1.	**If** アクティビティの*Selection Count Status*が*True* **Then**	
4.1.1.1.	子リストを空の順序付きリストに初期化する	
4.1.1.2.	**Iterate**アクティビティの*Selection Count*回繰り返す	
4.1.1.2.1.	アクティビティの子からアクティビティをランダムに選択する	
4.1.1.2.2.	選択したアクティビティを，元のアクティビティの順序を守って，子リストに追加する	
	End Iterate	
4.1.1.3.	子リストをアクティビティの*Available Children*に設定する	
	End If	
	End If	
4.2.	**Exit** *Select Children Process*	
	End Case	
5.	**Case:** アクティビティの*Selection Timing*が*On Each New Attempt*である	
5.1.	**Exit** *Select Children Process*	未定義な動作
	End Case	
6.	**Exit** *Select Children Process*	未定義なタイミング属性

pseudo-code for select children process

子ランダム化プロセス（Randomize Children Process）[SR.2] (アクティビティに対して; アクティビティの*Available Children*を変更する場合がある):		
Reference: Activity is Active AM.1.1; Activity is Suspended AM.1.1; Available Children AM.1.1; Activity Progress Status TM.1.2.1; Randomize Children SM.10; Randomization Timing SM.10		
1.	**If** アクティビティが子を持たない **Then**	葉アクティビティにランダム化は適用できない
1.1.	**Exit** *Randomize Children Process*	
	End If	
2.	**If** アクティビティの*Activity is Suspended*が*True* **Or** アクティビティの*Activity is Active*が*True* **Then**	中断しているかアクティブなアクティビティにランダム化は適用できない
2.1.	**Exit** *Randomize Children Process*	
	End If	
3.	**Case:** アクティビティの*Randomization Timing*が*Never*である	
3.1.	**Exit** *Randomize Children Process*	
	End Case	
4.	**Case:** アクティビティの*Randomization Timing*が*Once*である	
4.1.	**If** アクティビティの*Activity Progress Status*が*False* **Then**	アクティビティがまだ試行されていない場合
4.1.1.	**If** アクティビティの*Randomize Children*が*True* **Then**	
4.1.1.1.	アクティビティの*Available Children*に含まれるアクティビティをランダムに並べ替える	
	End If	
	End If	
4.2.	**Exit** 子*Randomize Children Process*	
	End Case	
5.	**Case:** アクティビティの*Randomization Timing*が*On Each New Attempt*である	
5.1.	**If** アクティビティに対する*Randomize Children*が*True* **Then**	
5.1.1.	アクティビティの*Available Children*に含まれるアクティビティをランダムに並べ替える	
	End If	
5.2.	**Exit** *Randomize Children Process*	
	End Case	
6.	**Exit** *Randomize Children Process*	未定義なタイムング属性

pseudocode for randomize children process

フローツリートラバーサルサブプロセス（Flow Tree Traversal Subprocess）[SB.2.1] (アクティビティ，トラバース方向，子検討フラグ，直前トラバース方向に対して; アクティビティツリー中の指定トラバース方向の'次'のアクティビティを返す，トラバース方向を返す場合がある，および例外コードを返す場合がある):		
参照: Available Children AM.1.1; Sequencing Control Forward Only SM.1; **下位試行終了プロセス** [UP.3];**シーケンシングルールチェックプロセス** UP.2		
1.	**トラバース方向を***False***に設定する**	
2.	**If (直前トラバース方向が定義され And 直前トラバース方向が後方) And アクティビティがアクティビティの親の***Available Children***のリストの最後のアクティビティ Then**	Forward only**クラスタを後方から移動して子の全てを飛ばしたかテストする**
2.1.	**トラバース方向は後方である**	
2.2.	**アクティビティはアクティビティの親の***Available Children***のリストの最初のアクティビティである**	
2.3.	**トラバース方向を***True***へ設定する**	
	End If	
3.	**If トラバース方向が前方 Then**	
3.1.	**If アクティビティがアクティビティツリーの前方順序付きツリートラバーサルの最後のアクティビティOr （アクティビティがアクティビティツリーのルートAnd** *consider children***が***False***）であれば Then**	**ツリーからの離脱でシーケンシングセッションが終了**
3.1.1.	**下位試行終了プロセスをアクティビティツリーのルートに適用する**	
3.1.2.	**Exit フローツリートラバーサルサブプロセス (次アクティビティ**: *n/a*;**シーケンシングセッション終了**: True;**例外**: *n/a*)	
	End If	
3.2.	**If アクティビティが葉である Or 子検討フラグが***False* **Then**	
3.2.1.	**If アクティビティがアクティビティの親の***Available Children* **のリストの最後のアクティビティ Then**	
3.2.1.1.	***Flow Tree Traversal Subprocess*****をアクティビティの親に対して，トラバース方向前方，直前トラバース方向** *n/a*，**子検討フラグ** *False***で適用する**	**再帰 - アクティビティの親の次の前方の兄弟へ移動する**
3.2.1.2.	**Exit** *Flow Tree Traversal Subprocess***（再帰フローツリートラバーサルサブプロセス（**the recursive *Flow Tree Traversal Subprocess***）の結果を返す）**	**再帰の結果を返す**
3.2.2.	**Else**	
3.2.2.1.	**ツリーを前方に，アクティビティの親の***Available Children***のリスト中を次アクティビティに一つ遡る**	
3.2.2.2.	**Exit** *Flow Tree Traversal Subprocess***（次アクティビティ**: **トラバーサルで特定されたアクティビティ**;**トラバース方向**: **トラバース方向**; **例外**: *n/a*）**	
	End If	
3.3.	**Else**	**クラスタに入る – 前方**
3.3.1.	**If アクティビティの***Available Children***のリストがNot 空 Then**	**このアクティビティが子アクテ**

		ィビティを持つことを確認する
3.3.1.1.	**Exit** *Flow Tree Traversal Subprocess*（**次アクティビティ**: アクティビティの*Available Children*リスト中の最初のアクティビティ; **トラバース方向**:トラバース方向; **例外**: *n/a* ）	
3.3.2.	**Else**	
3.3.2.1.	**Exit** *Flow Tree Traversal Subprocess*（**次アクティビティ** *Nil*; **トラバース方向**: *n/a*; **例外**: *SB.2.1-2*）	
	End If	
	End If	
	End If	
4.	**If トラバース方向が後方 Then**	
4.1.	**If アクティビティがツリーのルートアクティビティ Then**	アクティビティツリーのルートから移動できない
4.1.1.	**Exit** *Flow Tree Traversal Subprocess*（**次アクティビティ**: *Nil*; **トラバース方向**: *n/a*; **例外**: *SB.2.1-3*）	
	End If	
4.2.	**If アクティビティが葉 Or 子検討フラグが***False* **Then**	
4.2.1.	**If トラバーサル反対方向が***False* **Then**	forward onlyのクラスタを出ない場合forward onlyだけをテストする
4.2.1.1.	**If アクティビティの親の***Sequencing Control Forward Only* **が***True* **Then**	トラバース前にコントロールモードをテストする
4.2.1.1.1.	**Exit** *Flow Tree Traversal Subprocess*（**次アクティビティ**: *Nil*; **トラバース方向**: *n/a*; **例外**: *SB.2.1-4*）	
	End If	
	End If	
4.2.2.	**If アクティビティがアクティビティの親の***Available Children* **のリスト中で最初のアクティビティ Then**	
4.2.2.1.	*Flow Tree Traversal Subprocess***をアクティビティの親に対して適用する，トラバース方向後方，直前 トラバース方向** *n/a* **，子検討フラグ***False***で適用する**	再帰 - アクティビティの親の次の後方の兄弟へ移動する
4.2.2.2.	**Exit** *Flow Tree Traversal Subprocess*（**再帰フローツリートラバーサルサブプロセス**（the recursive *Flow Tree Traversal Subprocess*）**の結果**）	再帰の結果を返す
4.2.3.	**Else**	
4.2.3.1.	**ツリーを後方にアクティビティの親の***Available Children* **のリスト中を次アクティビティに一つ遡る**	
4.2.3.2.	**Exit** *Flow Tree Traversal Subprocess*（**次アクティビティ**: トラバーサルで特定されたアクティビティ; **トラバース方向**: トラバース方向; **例外**: *n/a*）	

	End If	
4.3.	**Else**	**クラスタへ入る – 後方**
4.3.1.	**If** アクティビティの*Available Children*のリスト中が**Not** 空 **Then**	**このアクティビティが子アクティビティを持つことを確認する**
4.3.1.1.	**If** アクティビティの*Sequencing Control Forward Only*が *True* **Then**	
4.3.1.1.1.	**Exit** *Flow Tree Traversal Subprocess*（**次アクティビティ**: アクティビティの*Available Children*リスト中の最初のアクティビティ; **トラバース方向**: *Forward*; **例外**: *n/a*）	forward onlyのクラスタの開始点から始める
4.3.1.2.	**Else**	
4.3.1.2.1.	**Exit** *Flow Tree Traversal Subprocess*（**次アクティビティ**: アクティビティの*Available Children*リスト中の最後のアクティビティ; **トラバース方向**: *Backward*; **例外**: *n/a*）	**クラスタの後方からは入る場合は最後から始める**
	End If	
4.3.2.	**Else**	
4.3.2.1.	**Exit** *Flow Tree Traversal Subprocess*（**次アクティビティ**: *Nil*; **トラバース方向**: *n/a*; **例外**: *SB.2.1-2*）	
	End If	
	End If	
	End If	

pseudo code for flow tree traversal subprocess

フローアクティビティトラバーサルサブプロセス（Flow Activity Traversal Subprocess）[SB.2.2] (アクティビティ，トラバース方向，直前トラバース方向に対して; アクティビティツリー中の指定トラバース方向の‘次’のアクティビティおよびアクティビティが配信可能な場合Trueを返す; 例外コードを返す場合がある):		
参照: Check Activity Process UP.5; Flow Activity Traversal Subprocess SB.2.2; Flow Tree Traversal Subprocess SB.2.1; Sequencing Control Flow SM.1; Sequencing Rules Check Process UP.2		
1.	**If** アクティビティの親の*Sequencing Control Flow*が*False* **Then**	‘flow’が実行可能なことを確認する
1.1.	**Exit** *Flow Activity Traversal Subprocess*（**配信可能**: *False*; **次アクティビティ**: アクティビティ; **例外**: *SB.2.2-1*）	
	End If	
2.	アクティビティに対して*Skipped* シーケンシングルールに関して*Sequencing Rules Check Process*を適用する	
3.	**If** *Sequencing Rules Check Process*が*Nil*を返さない **Then**	アクティビティをスキップし，‘next’アクティビティへ進むよう試みる
3.1.	アクティビティに対して*Flow Tree Traversal Subprocess*をトラバース方向および直前トラバース方向に子検討フラグ*False*で適用する	
3.2.	**If** *Flow Tree Traversal Subprocess*がアクティビティを特定しない **Then**	
3.2.1.	**Exit** フローアクティビティトラバーサルサブプロセス (**配信可能**: *False*; **次のアクティビティ**:アクティビティ; **シーケンシングセッション終了**: フローツリートラバーサルサブプロセスで特定された通り; **例外**: フローツリートラバーサルサブプロセスで特定された例外)	
3.3.	**Else**	
3.3.1.	**If** トラバース方向が後方 **And** *Flow Tree Traversal Subprocess* によって返されたトラバース方向が後方 **Then**	再帰コールが正しい方向を考慮することを確認する
3.3.1.1.	*Flow Tree Traversal Subprocess*が特定したアクティビティに対して*Flow Activity Traversal Subprocess*をトラバース方向に直前トラバース方向 *n/a*で適用する	再帰的に呼び出す – ‘next’アクティビティがOKかを確認する
3.3.2.	**Else**	
3.3.2.1.	*Flow Tree Traversal Subprocess*が特定したアクティビティに対して*Flow Activity Traversal Subprocess*をトラバース方向に直前トラバース方向の直前トラバース方向を適用する	再帰的に呼び出す– ‘next’アクティビティがOKかを確認する
	End If	
3.3.3.	**Exit** *Flow Activity Traversal Subprocess* - （再帰フローアクティビティトラバーサルサブプロセス（the recursive *Flow Activity Traversal Subprocess*）の結果を返す）	再帰から抜け出すことが可能
	End If	

	End If	
4.	**アクティビティに***Check Activity Process***を適用する**	**アクティビティが許可されるかを確認する**
5.	**If** *Check Activity Process***が***True* **を返す Then**	
5.1.	**Exit** *Flow Activity Traversal Subprocess*（**配信可能**: *False*; **次アクティビティ**: アクティビティ; **例外**: *SB.2.2-2*）	
	End If	
6.	**If アクティビティがアクティビティツリーの葉でない Then**	**葉アクティビティでないので配信できない; クラスタに入って葉をさがす**
6.1.	**アクティビティに対して***Flow Tree Traversal Subprocess***をトラバース方向に，直前トラバース方向** *n/a*，**子検討フラグ***True***で適用する**	
6.2.	**If** *Flow Tree Traversal Subprocess***がアクティビティを特定しない Then**	
6.2.1.	**Exit フローアクティビティトラバーサルサブプロセス (配信可能**: *False*; **次のアクティビティ**:アクティビティ; **シーケンシングセッション終了**: フローツリートラバーサルサブプロセスで特定された通り; **例外**: フローツリートラバーサルサブプロセスで特定された例外)	
6.3.	**Else**	
6.3.1.	**If トラバース方向が後方 And** *Flow Tree Traversal Subprocess* **によって返されたトラバース方向が前方 Then**	forward only**のクラスタで後方に進まないか確認する – 前方に移動しなければならない**
6.3.1.1.	*Flow Tree Traversal Subprocess***で特定されたアクティビティに対して***Flow Activity Traversal Subprocess***を，トラバース方向前方，直前トラバース方向後方で適用する**	**再帰的に呼び出す – 識別されたアクティビティ**がOK**であることを確認する**
6.3.2.	**Else**	
6.3.2.1.	*Flow Tree Traversal Subprocess***で特定されたアクティビティに対して***Flow Activity Traversal Subprocess***をトラバース方向に，直前トラバース方向***n/a***で適用する**	**再帰的に呼び出す– 識別されたアクティビティ**がOK**であることを確認する**
	End If	
6.3.3.	**Exit フ** *Flow Activity Traversal Subprocess* **- （再帰的フローアクティビティトラバーサルサブプロセス（**the recursive *Flow Activity Traversal Subprocess*）**の結果を返す）**	**再帰から抜け出すことが可能**
	End If	
	End If	
7.	**Exit** *Flow Activity Traversal Subprocess*（**配信可能**: *True*; **次アクティビティ**: アクティビティ; **例外**: *n/a* ）	**葉を見つけた**

pseudo code for フローアクティビティトラバーサルサブプロセス

フローサブプロセス（Flow Subprocess）[SB.2.3] (アクティビティ，トラバース方向，子検討フラグに対して; 移動が成功した否か，および，移動が停止したアクティビティを返す; 例外コードを返す場合がある):		
参照: Flow Activity Traversal Subprocess SB.2.2; Flow Tree Traversal Subprocess SB.2.1		
1.	アクティビティを候補アクティビティとする	候補アクティビティは移動を開始するところである
2.	候補アクティビティに対して*Flow Tree Traversal Subprocess*をトラバース方向に，直前トラバース方向*n/a* ，子検討フラグで適用する	指定方向に１アクティビティの移動を試みる
3.	**If** *Flow Tree Traversal Subprocess*がアクティビティを特定しない **Then**	移動するアクティビティがない
3.1.	**Exit** フローサブプロセス (**特定されたアクティビティ**: 候補アクティビティ; **配信可能**: *False*; **シーケンシングセッション終了**: フローツリートラバーサルサブプロセスで特定された通り; **例外**: フローツリートラバーサルサブプロセスで特定された例外)	
4.	**Else**	
4.1.	*Flow Tree Traversal Subprocess*が特定したアクティビティを候補アクティビティとする	
4.2.	候補アクティビティに対して*Flow Activity Traversal Subprocess*をトラバース方向に，直前トラバース方向 *n/a*で適用する	アクティビティの有効性を確認し有効な葉アクティビティが見つかるまで遡る
4.3.	**Exit** フローサブプロセス (**特定されたアクティビティ**: フローアクティビティトラバーサルサブプロセスで特定されたアクティビティ; **配信可能**: フローアクティビティトラバーサルサブプロセスで特定された通り; **シーケンシングセッション終了**: フローアクティビティトラバーサルサブプロセスで特定された値; **例外**: フローアクティビティトラバーサルサブプロセスで特定された例外)	
	End If	

pseudo code for flowsubprocess

Choiceアクティビティトラバーサルサブプロセス（Choice Activity Traversal Subprocess）[SB.2.4] (アクティビティとトラバース方向に対して;アクティビティに到達できればTrueを返す; 例外コートを返す場合がある):		
参照: シーケンシングコントロール Forward Only SM.1; シーケンシングルールチェックプロセス UP.2		
1.	**If** トラバース方向が前方 **Then**	
1.1.	アクティビティに*Sequencing Rules Check Process*を*Stop Forward Traversal sequencing rules*で適用する	
1.2.	**If** *Sequencing Rules Check Process*が*Nil*を返さない **Then**	
1.2.1.	**Exit** *Choice Activity Traversal Subprocess*（**到達可能**: *False*; **例外**: *SB.2.4-1*）	
	End If	
1.3.	**Exit** *Choice Activity Traversal Subprocess*（**到達可能**: *True*; **例外**: *n/a*）	
	End If	
2.	**If** トラバース方向が後方 **Then**	
2.1.	**If** アクティビティが親を持つ **Then**	
2.1.1.	**If** アクティビティの親の*Sequencing Control Forward Only*が *True* **Then**	
2.1.1.1.	**Exit** *Choice Activity Traversal Subprocess*（**到達可能**: *False*; **例外**: *SB.2.4-2*）	
	End If	
2.1.2.	**Else**	
2.1.2.1.	**Exit** *Choice Activity Traversal Subprocess*（**到達可能**: *False*; **例外**: *SB.2.4-3*）	アクティビティツリーのルートから後方へ動くことはできない
	End If	
2.2.	**Exit** *Choice Activity Traversal Subprocess*（**到達可能**: *True*; **例外**: *n/a* ）	
	End If	

pseudo code for choice activity traversal subprocess

Startシーケンシング要求プロセス（Start Sequencing Request Process）[SB.2.5] (配信要求を返す場合がある; 例外コードを返す場合がある):		
参照: Current Activity AM.1.2; Flow Subprocess SB.2.3		
1.	**If** *Current Activity*が定義されている **Then**	シーケンシングセッションがまだ始まっていないことを確認する
1.1.	**Exit** *Start Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *SB.2.5-1*）	配信するものがない
	End If	
2.	**If** アクティビティツリーのルートが葉 **Then**	開始前に，アクティビティツリーが一つ以上のアクティビティを含むことを確認する
2.1.	**Exit** *Start Sequencing Request Process*（**配信要求**:アクティビティツリーのルート; **例外**: *n/a*）	アクティビティが一つだけ，それは葉でなければならない
3.	**Else**	
3.1.	アクティビティツリーのルートに対して*Flow Subprocess*を*Forward*方向に，子検討フラグ*True*で適用する	アクティビティツリーへのフロー試みる
3.2.	**If** *Flow Subprocess*が*False*を返す **Then**	
3.2.1.	**Exit** *Start*シーケンシング要求プロセス (**配信要求**: *n/a*; **シーケンシングセッション終了**: フローサブプロセスで特定された通り; **例外**: フローサブプロセスで特定された例外)	配信するものがない
3.3.	**Else**	
3.3.1.	**Exit** *Start Sequencing Request Process*（**配信要求**: フローサブプロセスで特定されたアクティビティ; **例外**: *n/a*）	
	End If	
	End If	

pseudocode for start sequencing request process

Resume All シーケンシング要求プロセス（Resume All Sequencing Request Process）[SB.2.6] (配信要求を返す場合がある; 例外コードを返す場合がある):		
参照: Current Activity AM.1.2; Suspended Activity AM.1.2		
1.	**If** *Current Activity*が定義されている **Then**	シーケンシングセッションがまだ開始していないことを確認する
1.1.	**Exit** *Resume All Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *SB.2.6-1*）	配信するものがない
	End If	
2.	**If** *Suspended Activity*が定義されていない **Then**	再開する対象があることを確認する
2.1.	**Exit** *Resume All Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *SB.2.6-2*）	配信するものがない
	End If	
3.	**Exit** *Resume All Sequencing Request Process*（**配信要求**: *Suspended Activity*によって特定されたアクティビティ; **例外**: *n/a*）	配信要求プロセスが中断アクティビティが配信可能か否かを検証する

Continue シーケンシング要求プロセス（Continue Sequencing Request Process）[SB.2.7] (配信要求を返す場合がある; 例外コードを返す場合がある):		
参照: Current Activity AM.1.2; Flow Subprocess SB.2.3		
1.	**If** *Current Activity*が未定義 **Then**	シーケンシングセッションがすでに始まっている事を確認する
1.1.	**Exit** *Continue Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *SB.2.7-1*）	配信するものがない
	End If	
2.	**If** アクティビティがアクティビティツリーのルートアクティビティでない **Then**	
2.1.	**If** アクティビティの親の*Sequencing Control Flow*が*False* **Then**	フロー探索がアクティビティから許されることを確認する
2.1.1.	**Exit** *Flow Tree Traversal Subprocess*（**次のアクティビティ**: *Nil*; **例外**: *SB.2.7-2*）	
	End If	
	End If	
3.	*Current Activity*に対して*Flow Subprocess*を前方に子検討フラグを*False*で適用する	次に許可されたアクティビティへ前方へ移動
4.	**If** *Flow Subprocess*が *False* を返す **Then**	
4.1.	**Exit** *Continue*シーケンシング要求プロセス (**配信要求**: *n/a*; **シーケンシングセッション終了**: フローサブプロセスで特定された通り; **例外**: フローサブプロセスで特定された例外)	配信するものがない
5.	**Else**	
5.1.	**Exit** *Continue Sequencing Request Process*（**配信要求**: *Flow Subprocess*で特定されたアクティビティ; **例外**: *n/a* ）	
	End If	

pseudo-code for continue シーケンシング要求 process

Previous シーケンシング要求プロセス（Previous Sequencing Request Process）[SB.2.8] (配信要求を返す場合がある; 例外コードを返す場合がある):		
参照: Current Activity AM.1.2; Flow Subprocess SB.2.3		
1.	**If** *Current Activity*が未定義 **Then**	シーケンシングセッションがすでに始まっている事を確認する
1.1.	**Exit** *Previous Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *SB.2.8-1*）	配信するものがない
	End If	
2.	**If** アクティビティがアクティビティツリーのルートアクティビティでない **Then**	
2.1.	**If** アクティビティの親のシーケンシングコントロール *Flow*が *False*なら **Then**	アクティビティからのフロー探索が許されていることを確認する
2.1.1.	**Exit** *Previous* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.8-2*)	
	End If	
	End If	
3.	*Current Activity*に対して*Flow Subprocess*を後方に子検討フラグを *False*で適用する	次に許可されたアクティビティへ後方に移動
4.	**If** *Flow Subprocess*が*False*を返す **Then**	
4.1.	**Exit** *Previous Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *Flow Subprocess*で特定された例外）	配信するものがない
5.	**Else**	
5.1.	**Exit** *Previous Sequencing Request Process*（**配信要求**: *Flow Subprocess*で特定されたアクティビティ; **例外**: *n/a*）	
	End If	

pseudocode for previous シーケンシング要求プロセス

Choice シーケンシング要求プロセス（Choice Sequencing Request Process）[SB.2.9] (ターゲットアクティビティに対して; 配信要求を返す場合がある; *Current Activity*を変更する場合がある; 例外コードを返す場合がある):		
参照: Activity is Active AM.1.1; Activity is Suspended AM.1.1; Available Children AM.1.1; **チェックアクティビティプロセス** UP.5; Choice**フローサブプロセス**SB.2.9.1, Choice **アクティビティトラバーサルサブプロセス** SB.2.4; Current Activity AM.1.2; **試行終了プロセス** UP.4; **フローサブプロセス** SB.2.3; **シーケンシングコントロールモード** Choice SM.1; **シーケンシングコントロール** Choice Exit SM.1; **シーケンシングルールチェックプロセス** UP.2; **下位試行終了プロセス** UP.3; *adlseq:constrainedChoice SCORM SN; adlseq:preventActivation SCORM SN*		
1.	**If ターゲットアクティビティがない Then**	**選択対象のターゲットアクティビティがなければならない**
1.1.	**Exit** *Choice Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *SB.2.9-1*）	**配信するものがない**
	End If	
2.	**アクティビティツリーのルートからターゲットアクティビティまで，両端のアクティビティを含む順序付き系列のアクティビティパスを作る**	
3.	**For アクティビティパスの各アクティビティ**	
3.1.	**If アクティビティがアクティビティツリーのルートでなければ Then**	
3.1.1.	**If アクティビティの親の***Available Children***がアクティビティを含まなければThen**	**アクティビティが現在のところ使用不可能**
3.1.1.1.	**Exit** *Choice* **シーケンシング要求プロセス (配信要求**: *n/a*; **例外**: *SB.2.9-2*)	**配信するものがない**
	End If	
	End If	
3.2.	**シーケンシングルールチェックプロセスをアクティビティに対して***Hide from Choice* **シーケンシングルールについて適用する**	**隠蔽されたものを選択することはできない**
3.3.	**If シーケンシングルールチェックプロセス が***Nil***を返さなければばThen**	
3.3.1.	**Exit** *Choice* **シーケンシング要求プロセス (配信要求**: *n/a*; **例外**: *SB.2.9-3*)	**配信するものがない**
	End If	**隠蔽されたものを選択することはできない**
	End For	
4.	**If ターゲットアクティビティがアクティビティツリーのルートでなければThen**	
4.1.	**If ターゲットアクティビティの親のシーケンシングコントロールモード** *Choice***が***False***ならThen**	**コントロールモードが対象の‘choice’を許可しているか確認する**
4.1.1.	**Exit** *Choice* **シーケンシング要求プロセス (配信要求**: *n/a*; **例**	**配信するものが**

	外: *SB.2.9-3*)	**ない**
	End If	
	End If	
5.	**If** *Current Activity***が定義されているなら Then**	**シーケンシングセッションがすでに始まっているか？**
5.1.	*Current Activity***とターゲットアクティビティの共通の祖先を探す**	
6.	**Else**	
6.1.	**共通の祖先はアクティビティツリーのルートである**	**いえ、ターゲットの選択でシーケンシングセッションが始まる**
	End If	
7.	**Case:** *Current Activity***とターゲットアクティビティが同一である**	**ケース** #1 - **現在のアクティビティを選択する**
7.1.	**Break All** Cases	**このケースではすることがない**
	End Case	
8.	**Case:** *Current Activity***とターゲットアクティビティが兄弟である**	**ケース**#2 – **同じクラスタ**; **ターゲットアクティビティに移動する**
8.1.	*Current Activity***からターゲットアクティビティまで、両端のアクティビティを含む、順序付き系列のアクティビティリストを作る**	**ターゲットアクティビティに移動するように試行する。一旦ターゲットアクティビティに到達した場合それをテストする必要はない**
8.2.	**If アクティビティリストが空なら Then**	**選択するものがない**
8.2.1.	**Exit** *Choice* **シーケンシング要求プロセス (配信要求**: *n/a*; **例外**: *SB.2.9-5*)	**配信するものがない**
	End If	
8.3.	**If** the **ターゲットアクティビティがアクティビティツリーの順序付きトラバーサルに関して***Current Activity***の前方にある Then**	
8.3.1.	**移動方向は前方である**	
8.4.	**Else**	
8.4.1.	**移動方向は後方である**	
	End If	
8.5.	**For アクティビティリストの各アクティビティ**	
8.5.1.	*Choice* **アクティビティトラバーサルサブプロセスをアクティビティに移動方向に適用する**	
8.5.2.	**If** *Choice* **アクティビティトラバーサルサブプロセスが***False* **を返せば Then**	

8.5.2.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *Choice* アクティビティトラバーサルサブプロセスによって特定された例外)	配信するものがない
	End If	
	End For	
8.6.	**Break All** Cases	
	End Case	
9.	**Case:** *Current Activity*と共通の祖先が同じ **Or** *Current Activity*が未定義である	ケース#3 – 対象へのパスはアクティビティツリー内の前方である
9.1.	共通の祖先からターゲットアクティビティまで、ターゲットアクティビティを除く順序付き系列のアクティビティパスを作る	
9.2.	**If** アクティビティパスが空なら**Then**	
9.2.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-5*)	配信するものがない
	End If	
9.3.	**For**アクティビティパスの各アクティビティ	
9.3.1.	*Choice* アクティビティトラバーサルサブプロセスをアクティビティに前方に適用する	
9.3.2.	**If** *Choice* アクティビティトラバーサルサブプロセスが*False* を返せば**Then**	
9.3.2.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *Choice* アクティビティトラバーサルサブプロセスによって特定された例外)	配信するものがない
	End If	
9.3.3.	**If** アクティビティの*Activity is Active*が*False* **And** (アクティビティが共通の祖先ではない **And** アクティビティの *adlseq:preventActivation*が*True*)なら **Then**	確認中のアクティビティがアクティブでない場合、アクティブにすることを許可されていることを確認する
9.3.3.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-6*)	配信するものがない
	End If	
	End For	
9.4.	**Break All** Cases	
	End Case	
10.	**Case:**ターゲットアクティビティは*Current Activity*の共通の祖先である	ケース#4 – 対象へのパスはアクティビティツリーの後方である
10.1.	*Current Activity*からターゲットアクティビティまで，両端のアクティビティを含む，順序付き系列のアクティビティパスを作る	
10.2.	**If** アクティビティパスが空なら**Then**	
10.2.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-5)*	配信するものがない

	End If	
10.3.	**For** アクティビティパスの各アクティビティ	
10.3.1.	**If** アクティビティがアクティビティパスの最後のアクティビティでなければ**Then**	
10.3.1.1.	**If** アクティビティに対するシーケンシングコントロール *Choice Exit* が*False* なら**Then**	ターゲットが配信されたら、終了してはならないアクティビティが終了してしまうことを確認する
10.3.1.1.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-7*)	配信するものがない
	End If	
	End If	
	End For	
10.4.	**Break All** Cases	
	End Case	
11.	**Case:** ターゲットアクティビティが共通の祖先のアクティビティより前方である	ケース#5 – 対象は共通の祖先の下位アクティビティである
11.1.	*Current Activity*から共通の祖先まで、共通の祖先を含まない順序付き系列のアクティビティパスを作る	
11.2.	**If** アクティビティパスが空なら**Then**	
11.2.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-5*)	配信するものがない
	End If	
11.3.	制限付きアクティビティを未定義に設定する	
11.4.	**For** アクティビティパスの各アクティビティ	ツリーを共通の祖先に向けて移動する
11.4.1.	**If** アクティビティに対するシーケンシングコントロール *Choice Exit* が*False*なら**Then**	ターゲットが配信されたら、終了してはならないアクティビティが終了してしまうことを確認する
11.4.1.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-7*)	配信するものがない
	End If	
11.4.2.	**If** 制限付きアクティビティが未定義なら **Then**	現在のアクティビティに最も近い制限付きアクティビティを探す
11.4.2.1.	**If** アクティビティに対する *adlseq:constrainedChoice* が *True* なら **Then**	

11.4.2.1.1.	制限付きアクティビティをアクティビティに設定する	
	End If	
	End If	
	End For	
11.5.	**If** 制限付きアクティビティが定義されているなら **Then**	
11.5.1.	**If** ターゲットアクティビティがアクティビティツリー内で制限付きアクティビティより前方なら **Then**	
11.5.1.1.	移動方向は前方である	次にどんなアクティビティがあるのか確認するために前方へ移動する
11.5.2.	**Else**	
11.5.2.1.	移動方向は後方である	次にどんなアクティビティがあるのか確認するために後方へ移動する
	End If	
11.5.3.	*Choice* フローサブプロセス を移動方向に制限付きアクティビティに適用する	
11.5.4.	確認対象アクティビティを *Choice* フローサブプロセスで特定されたアクティビティに設定する	
11.5.5.	**If** ターゲットアクティビティが確認対象アクティビティの利用可能な子孫でない **And** ターゲットアクティビティが確認対象アクティビティではない **And** ターゲットアクティビティが制限付きアクティビティではない **Then**	ターゲットアクティビティがフローの制限付き選択肢内であることを確認する
11.5.5.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-8*)	
	End If	
	End If	
11.6.	共通の祖先からターゲットアクティビティまで、ターゲットアクティビティを含まない、順序付き系列のアクティビティパスを作る	
11.7.	**If** アクティビティパスが空なら**Then**	
11.7.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-5*)	配信するものがない
	End If	
11.8.	**If** ターゲットアクティビティがアクティビティツリー内で *Current Activity*より前方であるなら **Then**	ターゲットアクティビティに向かって移動する
11.8.1.	**For** アクティビティパスの各アクティビティ	
11.8.1.1.	アクティビティに対して*Choice* アクティビティトラバーサルサブプロセスを前方に適用する	
11.8.1.2.	**If** *Choice* アクティビティトラバーサルサブプロセスが *False*を返せば**Then**	
11.8.1.2.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**:	配信するものが

	n/a; **例外**: *Choice* アクティビティトラバーサルサブプロセスによって特定された例外)	ない
	End If	
11.8.1.3.	**If** アクティビティに対する*Activity is Active*が*False* **And** (アクティビティが共通の祖先ではない **And** アクティビティの*adlseq:preventActivation*が*True*)なら**Then**	確認中のアクティビティがアクティブでないなら、それをアクティブにすることが許可される事を確認する
11.8.1.3.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-6*)	配信するものがない
	End If	
	End For	
11.9.	**Else**	
11.9.1.	**For** アクティビティパスの各アクティビティ	
11.9.1.1.	**If** アクティビティの*Activity is Active*が*False* **And** (アクティビティが共通の祖先ではない **And** アクティビティの*adlseq:preventActivation*が*True*)なら **Then**	確認中のアクティビティがアクティブでないなら、それをアクティブにすることが許可される事を確認する
11.9.1.1.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-6*)	配信するものがない
	End If	
	End For	
	End If	
11.10.	**Break All** Cases	
	End Case	
12.	**If** ターゲットアクティビティが葉アクティビティなら**Then**	
12.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: the ターゲットアクティビティ; **例外**: *n/a)*	
	End If	
13.	フローサブプロセスをターゲットアクティビティへ前方に子検討フラグ*True*で適用する	特定されたアクティビティはクラスタである。クラスタに入り配信のため下位葉の発見を試みる
14.	**If** フローサブプロセスが*False*を返せば **Then**	配信するものがないがターゲットアクティビティへ到着した – 現在のアクティビティを移動する
14.1.	下位試行終了プロセスを共通の祖先に適用する	

14.2.	**試行終了プロセスを共通の祖先に適用する**	
14.3.	*Current Activity*をターゲットアクティビティに設定する	
14.4.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: *n/a*; **例外**: *SB.2.9-9*)	配信するものがない
15.	**Else**	
15.1.	**Exit** *Choice* シーケンシング要求プロセス (**配信要求**: フローサブプロセスで特定されたアクティビティ; **例外**: *n/a*)	
	End If	

pseudocode for choice sequencing request process

Choice フローサブプロセス（Choice Flow Subprocess）[SB.2.9.1] (アクティビティとトラバース方向に対して; フローがどのアクティビティで止まるかを示す):		
参照: Choice Flow Tree Traversal Subprocess SB.2.9.2		
1.	アクティビティに*Choice Flow Tree Traversal Subprocess*を移動方向に適用する	アクティビティから離れようとする，特定の方向へ 1 アクティビティ
2.	**If** *Choice Flow Tree Traversal Subprocess*が*Nil*を返す **Then**	
2.1.	**Exit** *Choice Flow Subprocess*（**特定されたアクティビティ** :アクティビティ）	
3.	**Else**	
3.1.	**Exit** *Choice Flow Subprocess*（**特定されたアクティビティ;フローツリートラバーサルサブプロセスで特定されたアクティビティ**）	
	End If	

Choice フローツリートラバーサルサブプロセス（Choice Flow Tree Traversal Subprocess） **[SB.2.9.2]** (アクティビティ，移動方向に対して; アクティビティツリーの移動方向へ次のアクティビティを返す):		
Reference: Available Children AM.1.1		
1.	**If トラバース方向が前方 Then**	
1.1.	**If アクティビティがアクティビティツリーの前方順序付きツリートラバーサルの最後のアクティビティOrアクティビティがアクティビティツリーのルートであればThen**	**アクティビティツリーを移動できない**
1.1.1.	**Exit** *Choice Flow Tree Traversal Subprocess*（**次のアクティビティ**: *Nil*）	
	End If	
1.2.	**If アクティビティがアクティビティの親の***Available Children***リストの最後のアクティビティ Then**	
1.2.1.	*Choice Flow Tree Traversal Subprocess***をアクティビティの親に対して，トラバース方向前方に適用する**	**再帰 - アクティビティの親の次の前方の兄弟へ移動する**
1.2.2.	**Exit** *Choice Flow Tree Traversal Subprocess*（**次のアクティビティ**: **再帰フローツリートラバーサルサブプロセス**（the recursive *Choice Flow Tree Traversal Subprocess*）**の結果を返す**）	**再帰の結果を返す**
1.3.	**Else**	
1.3.1.	**ツリーを前方に，アクティビティの親の***Available Children***のリスト中を次アクティビティに一つ遡る**	
1.3.2.	**Exit** *Choice Flow Tree Traversal Subprocess*（**次のアクティビティ**: **トラバーサルで特定されたアクティビティ**）	
	End If	
	End If	
2.	**If トラバース方向が後方なら Then**	
2.1.	**If アクティビティがアクティビティツリーのルート Then**	**アクティビティツリーのルートから移動できない**
2.1.1.	**Exit** *Choice Flow Tree Traversal Subprocess*（**次のアクティビティ**）	
	End If	
2.2.	**If アクティビティがアクティビティの親の***Available Children***リストの最初のアクティビティ Then**	
2.2.1.	*Choice Flow Tree Traversal Subprocess***をアクティビティの親へトラバース方向後方に適用する**	**再帰 – アクティビティの親の次の後方の兄弟へ移動する**
2.2.1.1.	**Exit** *Choice Flow Tree Traversal Subprocess*（**次のアクティビティ**: **再帰フローツリートラバーサルサブプロセス**（the recursive *Choice Flow Tree Traversal Subprocess*）**の結果を返す**）	**再帰の結果を返す**
2.3.	**Else**	
2.3.1.	**ツリーを後方にアクティビティの親の***Available Children***のリ**	

	スト中を次アクティビティに一つ遡る	
2.3.2.	**Exit** *Choice Flow Tree Traversal Subprocess*（**次のアクティビティ**: トラバーサルで特定されたアクティビティ）	
	End If	
	End If	

pseudo code for choice flow tree traversal subprocess

Retry シーケンシング要求プロセス（Retry Sequencing Request Process）[SB.2.10] (配信要求を返す場合がある; 例外コードを返す場合がある):		
参照: Activity is Active AM.1.1; Activity is Suspended AM.1.1; Current Activity AM.1.2; Flow Subprocess SB.2.3		
1.	**If** *Current Activity*が未定義 **Then**	シーケンシングセッションがすでに始まっていることを確認する
1.1.	**Exit** *Retry Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *SB.2.10-1*）	配信するものがない
	End If	
2.	**If** *Current Activity*の*Activity is Active*が*True* **Or** *Current Activity*の*Activity is Suspended*が*True* **Then**	まだアクティブもしくは中断されたアクティビティはリトライできない
2.1.	**Exit** *Retry Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *SB.2.10-2*）	配信するものがない
	End If	
3.	**If** *Current Activity*が葉でない **Then**	
3.1.	*Current Activity*に対して*Flow Subprocess*を前方に子検討フラグ*True*で適用する	
3.2.	**If** *Flow Subprocess*が*False*を返す **Then**	
3.2.1.	**Exit** *Retry Sequencing Request Process*（**配信要求**: *n/a*; **例外**: *SB.2.10-3*）	配信するものがない
3.3.	**Else**	
3.3.1.	**Exit** *Retry Sequencing Request Process*（**配信要求**: *Flow Subprocess*で特定されたアクティビティ; **例外**: *n/a*）	
	End If	
4.	**Else**	
4.1.	**Exit** *Retry Sequencing Request Process*（**配信要求**: *Current Activity*; **例外**: *n/a*）	
	End If	

pseudocode for retry シーケンシング要求 process

Exitシーケンシング要求プロセス（Exit Sequencing Request Process）[SB.2.11] (シーケンシングセッションが終了したか否かを提示する; 例外コードを返す):		
参照: Activity is Active AM.1.1; Current Activity AM.1.2		
1.	**If** *Current Activity*が未定義 **Then**	シーケンシングセッションがすでに始まっていることを確認する
1.1.	**Exit** *Exit Sequencing Request Process*（**シーケンシングセッション終了**: *False*; **例外**: *SB.2.11-1*）	
	End If	
2.	**If** *Current Activity*の*Activity is Active*が*True* **Then**	現在のアクティビティがすでに終了しているかを確認する
2.1.	**Exit** *Exit Sequencing Request Process*（**シーケンシングセッション終了**: *False*; **例外**: *SB.2.11-2*）	
	End If	
3.	**If** *Current Activity*がアクティビティツリーのルート **Then**	
3.1.	**Exit** *Exit Sequencing Request Process*（**シーケンシングセッション終了**: *True*; **例外**: *n/a*）	シーケンシングセッション終了，コントロールをLTSに返す
	End If	
4.	**Exit** *Exit Sequencing Request Process*（**シーケンシングセッション終了**: *False*; **例外**: *n/a*）	

pseudo code for exit sequencing request process

シーケンシング要求プロセス（Sequencing Request Process）[SB.2.12] (シーケンシング要求に対し; シーケンシング要求を検証し; 配信要求を返す場合がある; LTSに制御を返すことを示す場合がある; 例外コードを返す場合がある):		
参照: Choice Sequencing Request Process SB.2.9; Continue Sequencing Request Process SB.2.7; Exit Sequencing Request Process SB.2.11; Previous Sequencing Request Process SB.2.8; Resume All Sequencing Request Process SB.2.6; Retry Sequencing Request Process SB.2.10; Start Sequencing Request Process SB.2.5		
1.	**Case:** シーケンシング要求が*Start*である	
1.1.	*Start Sequencing Request Process*を適用する	
1.2.	**If** *Start Sequencing Request Process*が例外を返す **Then**	
1.2.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Not Valid*; **配信要求**: *n/a*; **シーケンシングセッション終了**: *n/a*; **例外**: *Start Sequencing Request Process*)で特定された例外）	
1.3.	**Else**	
1.3.1.	**Exit** シーケンシング要求プロセス (**シーケンシング要求**: *Valid*; **配信要求**: *Start* シーケンシング要求プロセスの結果; **シーケンシングセッション終了**: *Start* シーケンシング要求プロセスの結果; **例外**: *n/a*)	
	End If	
	End Case	
2.	**Case:** シーケンシング要求が*Resume All*である	
2.1.	*Resume All Sequencing Request Process*を適用する	
2.2.	**If** *Resume All Sequencing Request Process*が例外を返す **Then**	
2.2.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Not Valid*; **配信要求**: *n/a*; **シーケンシングセッション終了**: *n/a*; **例外**: *Sequencing Request Process*で特定された例外）	
2.3.	**Else**	
2.3.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Valid*; **配信要求**: *Resume All Sequencing Request Process*の結果; **シーケンシングセッション終了**: *n/a*; **例外**: *n/a*）	
	End If	
	End Case	
3.	**Case:** シーケンシング要求が*Exit*である	
3.1.	*Exit Sequencing Request Process*を適用する	
3.2.	**If** *Exit Sequencing Request Process*が例外を返す **Then**	
3.2.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Not Valid*; **配信要求**: *n/a*; **シーケンシングセッション終了**: *n/a*; **例外**: *Exit Sequencing Request Process*で特定された例外）	
3.3.	**Else**	
3.3.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Valid*; **配信要求**: *n/a*; **シーケンシングセッション終了**: *Exit Sequencing Request Process*の結果; **例外**: *n/a*）	
	End If	
	End Case	
4.	**Case:** シーケンシング要求が*Retry*である	
4.1.	*Retry Sequencing Request Process*を適用する	
4.2.	**If** *Retry Sequencing Request Process*が例外を返す **Then**	
4.2.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Not*	

	Valid; **配信要求**: *n/a*; **シーケンシングセッション終了**: *n/a*; **例外**: *Retry Sequencing Request Process*で特定された例外）	
4.3.	**Else**	
4.3.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Valid*; **配信要求**: *Retry Sequencing Request Process*の結果; **シーケンシングセッション終了**: *n/a*) ; **例外**: *n/a*）	
	End If	
	End Case	
5.	**Case:** シーケンシング要求が*Continue*である	
5.1.	*Continue Sequencing Request Process*を適用する	
5.2.	**If** *Continue Sequencing Request Process*が例外を返す **Then**	
5.2.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Not Valid*; **配信要求**: *n/a*; **シーケンシングセッション終了**: *n/a*; **例外**: *Continue Sequencing Request Process*で特定された例外）	
5.3.	**Else**	
5.3.1.	**Exit** シーケンシング要求プロセス (**シーケンシング要求**: *Valid*; **配信要求**: *Continue* シーケンシング要求プロセスの結果; **シーケンシングセッション終了**: *Continue* シーケンシング要求プロセスの結果; **例外**: *n/a*)	
	End If	
	End Case	
6.	**Case:** シーケンシング要求が*Previous*である	
6.1.	*Previous Sequencing Request Process*を適用する	
6.2.	**If** *Previous Sequencing Request Process*が例外を返す **Then**	
6.2.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Not Valid*; **配信要求**: *n/a*; **シーケンシングセッション終了**: *n/a*; **例外**: *Previous Sequencing Request Process*で特定された例外）	
6.3.	**Else**	
6.3.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Valid*; **配信要求**: *Previous Sequencing Request Process*の結果; **シーケンシングセッション終了**: *n/a*; **例外**: *n/a*）	
	End If	
	End Case	
7.	**Case:** シーケンシング要求が*Choice*である	
7.1.	*Choice Sequencing Request Process*を適用する	
7.2.	**If** *Choice Sequencing Request Process*が例外を返す **Then**	
7.2.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Not Valid*; **配信要求**: *n/a*; **シーケンシングセッション終了**: *n/a*; **例外**: *Choice Sequencing Request Process*で特定された例外）	
7.3.	**Else**	
7.3.1.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Valid*; **配信要求**: *Choice Sequencing Request Process*の結果; **シーケンシングセッション終了**: *n/a*; **例外**: *n/a*）	
	End If	
	End Case	
8.	**Exit** *Sequencing Request Process*（**シーケンシング要求**: *Not Valid*; **配信要求**: *n/a*; **シーケンシングセッション終了**: *n/a*; **例外**: *SB.2.12-1*）	無効なシーケンシング要求

配信要求プロセス（Delivery Request Process）[DB.1.1] (配信要求に対して; 配信要求有効性を返す; 例外コードを返す場合がある):		
参照: Check Activity Process UP.5		
1.	**If** 配信要求で指定されたアクティビティが葉でない **Then**	葉アクティビティだけを配信することができる
1.1.	**Exit** *Delivery Request Process*（**配信要求**: *Not Valid*; **例外**: *DB.1.1-1*）	
	End If	
2.	アクティビティツリーのルートから配信要求で指定されたアクティビティまで，両端のアクティビティを含む，順序付き系列のアクティビティパスを作る	
3.	**If** アクティビティパスが空 **Then**	配信するものがない
3.1.	**Exit** *Delivery Request Process*（**配信要求**: *Not Valid*; **例外**: *DB.1.1-2*）	
	End If	
4.	**For** アクティビティパスの各アクティビティ	パスに沿った各アクティビティが許可されているか確認する
4.1.	*Check Activity Process*をアクティビティに適用する	
4.2.	**If** *Check Activity Process*が*True* を返す **Then**	
4.2.1.	**Exit** *Delivery Request Process*（**配信要求**: *Not Valid*; **例外**: *DB.1.1-3*）	
	End If	
	End For	
5.	**Exit** *Delivery Request Process*（**配信要求**: *Valid*; **例外**: *n/a*）	

pseudo code for delivery request process

コンテンツ配信環境プロセス（Content Delivery Environment Process）[DB.2] (配信要求に対する; 例外コードを返すことがある):		
参照: Activity Progress Status TM.1.2.1; Activity Attempt Count TM.1.2.1; Activity is Active AM.1.1; Activity is Suspended AM.1.1; Attempt Absolute Duration TM.1.2.2; Attempt Experienced Duration TM.1.2.2; Attempt Progress Information TM.1.2.2; Clear Suspended Activity Subprocess DB.2.1; Current Activity AM.1.2; Objective Progress Information TM.1.1; Suspended Activity AM.1.2; Terminate Descendent Attempts Process UP.4; Tracked SM.11		
1.	**If** *Current Activity*の*Activity is Active*が*True* **Then**	現在のアクティビティの試行が終了していなければ，新しいコンテンツを配信することはできない
1.1.	**Exit** *Content Delivery Environment Process*（**例外**: *DB.2-1*）	配信要求は無効 - *Current Activity* は終了していない
	End If	
2.	**If** 配信対象アクティビティが*Suspended Activity*でない **Then**	コンテンツは配信される. suspend all状態をクリアする
2.1.	*Clear Suspended Activity Subprocess*を配信対象アクティビティに適用する	
	End If	
3.	*Terminate Descendent Attempts Process*を配信対象アクティビティに適用する	全ての試行が終了している事を確認する
4.	アクティビティツリーのルートから配信対象アクティビティまで，両端のアクティビティを含む，順序付き系列のアクティビティパスを作る	対象アクティビティを配信するため，全ての試行を開始する
5.	**For** アクティビティパスの各アクティビティ	
5.1.	**If** アクティビティの*Activity is Active*が*False* **Then**	
5.1.1.	**If** アクティビティの*Tracked* が*True* **Then**	
5.1.1.1.	**If** アクティビティの*Activity is Suspended*が*True* **Then**	中断によってアクティビティの前の試行が終了したなら中断状態をクリアする; 新しい試行を始めない
5.1.1.1.1.	アクティビティの*Activity is Suspended*を*False*に設定する	
5.1.1.2.	**Else**	
5.1.1.2.1.	アクティビティの*Activity Attempt Count*を 1 増やす	アクティビティの新しい試行を開始する

5.1.1.2.2.	**If アクティビティの*Activity Attempt Count*が *(1)* Then**	**これがアクティビティへの最初の試行？**
5.1.1.2.2.1.	**アクティビティの*Activity Progress Status*を*True*に設定する**	
	End If	
5.1.1.2.3.	**新しい試行に必要な*Objective Progress Information*と*Attempt Progress Information* を初期化する**	**新しい試行のためトラッキング情報を初期化する**
	End If	
	End If	
5.1.2.	**アクティビティの*Activity is Active*を*True*に設定する**	
	End If	
	End For	
6.	***Current Activity*を配信対象アクティビティに設定する**	**配信対象アクティビティが現在のアクティビティになる**
7.	**中断アクティビティを未定義にする**	
8.	**アクティビティのコンテンツリソースと補助リソースの配信が開始される**	**配信環境は対象アクティビティに関連付けられたコンテンツリソースを配信することを仮定する。アクティビティがアクティブであると仮定される間、シーケンサは学習者の状態を記録する**
8.1.	**If 配信対象アクティビティの*Tracked* が*False*なら Then**	
8.1.1.	**アクティビティの学習目標と試行進捗情報は配信中に記録しない**	
8.1.2.	**配信環境は*Attempt Absolute Duration* と*Attempt Experienced Duration*の記録を開始する**	
	End If	
9.	**Exit コンテンツ配信環境プロセス (例外: *n/a*)**	

pseudocode for content delivery environment process

中断アクティビティクリアサブプロセス（Clear Suspended Activity Subprocess）[DB.2.1] (アクティビティに対して; *Suspended Activity*を変更する場合がある):		
参照: Activity is Suspended AM.1.1; Suspended Activity AM.1.2		
1.	**If** *Suspended Activity*が定義されている **Then**	クリアするものがあるか確認する
1.1.	指定されたアクティビティと*Suspended Activity*の共通の祖先を見つける	
1.2.	*Suspended Activity*の親から共通の祖先まで，両端のアクティビティを含む，順序付き系列のアクティビティパスを作る	
1.3.	**If** アクティビティパスが空でない **Then**	
1.3.1.	**For** アクティビティパスの各アクティビティ	指定された各アクティビティを'not suspended'に設定しながらツリーを下りる
1.3.1.1.	**If** アクティビティが葉 **Then**	
1.3.1.1.1.	アクティビティの*Activity is Suspended*を*False*に設定する	
1.3.1.2.	**Else**	
1.3.1.2.1.	**If** アクティビティが*Activity is Suspended*が*True*の子アクティビティを持たない **Then**	
1.3.1.2.1.1.	アクティビティの*Activity is Suspended* を*False*に設定する	
	End If	
	End If	
	End For	
	End If	
1.4.	*Suspended Activity*を未定義に設定する	Suspended Activity をクリアする
	End If	
2.	**Exit** *Clear Suspended Activity Subprocess*	

制限条件チェックプロセス（Limit Conditions Check Process）[UP.1] (アクティビティに対して; アクティビティのいずれかの制限条件が違反されている場合*True*を返す):		
参照: Activity Attempt Count TM.1.2.1; Activity Progress Status TM.1.2.1; Activity Absolute Duration TM.1.2.1; Activity Experienced Duration TM.1.2.1; Attempt Progress Status TM.1.2.2; Attempt Absolute Duration TM.1.2.2; Attempt Experienced Duration TM.1.2.2; Limit Condition Activity Absolute Duration Control SM.3; Limit Condition Activity Absolute Duration Limit SM.3; Limit Condition Activity Experienced Duration Control SM.3; Limit Condition Activity Experienced Duration Limit SM.3; Limit Condition Attempt Absolute Duration Control SM.3; Limit Condition Attempt Absolute Duration Limit SM.3; Limit Condition Attempt Experienced Duration Control SM.3; Limit Condition Attempt Experienced Duration Limit SM.3; Limit Condition Attempt Control SM.3; Limit Condition Attempt Limit SM.3; Limit Condition Begin Time Limit SM.3; Limit Condition Begin Time Limit Control SM.3; Limit Condition End Time Limit SM.3; Limit Condition End Time Limit Control SM.3; Tracked SM.11		
1.	**If** アクティビティの*Tracked* が*False* **Then**	アクティビティが記録されていなければ，制限条件に反することはない
1.1.	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *False*）	アクティビティが記録されていないので，制限条件に反していない
	End If	
2.	**If** アクティビティの*Activity is Active*が*True* **Or** アクティビティの*Activity is Suspended*が*True*なら**Then**	新しい試行を開始するアクティビティのみチェックする必要がある
2.1	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *False*）	
	End If	
3.	**If** アクティビティの*Limit Condition Attempt Control*が*True* **Then**	
3.1.	**If** アクティビティの*Activity Progress Status*が*True* **And** アクティビティの*Activity Attempt Count*がアクティビティの*Limit Condition Attempt Limit* より大きいか等しい **Then**	
3.1.1.	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *True*）	制限条件に反している
	End If	
	End If	
4.	**If** アクティビティの*Limit Condition Activity Absolute Duration Control* が*True* **Then**	
4.1.	**If** アクティビティの*Activity Progress Status*が*True* **And** アクティビティの*Activity Absolute Duration*がアクティビティの*Limit Condition Activity Absolute Duration Limit* より大きいか等しい **Then**	
4.1.1.	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *True*）	制限条件に反している
	End If	
	End If	
5.	**If** アクティビティの*Limit Condition Activity Experienced Duration Control*が*True* **Then**	
5.1.	**If** アクティビティの*Activity Progress Status*が*True* **And** アクティビティの*Activity Experienced Duration*がアクティビティの*Limit*	

	*Condition Activity Experienced Duration Limit*より大きいか等しい **Then**	
5.1.1.	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *True*）	制限条件に反している
	End If	
	End If	
6.	**If** アクティビティの*Limit Condition Attempt Absolute Duration Control*が*True* **Then**	
6.1.	**If** アクティビティの*Activity Progress Status*が*True* **And** アクティビティの*Attempt Progress Status*が*True* **And** アクティビティの*Attempt Absolute Duration*がアクティビティの*Limit Condition Attempt Absolute Duration Limit*より大きいか等しい **Then**	
6.1.1.	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *True*）	制限条件に反している
	End If	
	End If	
7.	**If** アクティビティの*Limit Condition Attempt Experienced Duration Control*が*True* **Then**	
7.1.	**If** アクティビティの*Activity Progress Status*が*True* **And** アクティビティの*Attempt Progress Status*が*True* **And** アクティビティの*Attempt Experienced Duration*がアクティビティの*Limit Condition Attempt Experienced Duration Limit*より大きいか等しい **Then**	
7.1.1.	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *True*）	制限条件に反している
	End If	
	End If	
8.	**If** アクティビティの*Limit Condition Begin Time Limit Control*が*True* **Then**	
8.1.	**If** 現時刻がアクティビティの*Limit Condition Begin Time Limit*より前 **Then**	
8.1.1.	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *True*）	制限条件に反している
	End If	
	End If	
9.	**If** アクティビティの*Limit Condition End Time Limit Control*が*True* **Then**	
9.1.	**If** 現時刻がアクティビティの*Limit Condition End Time Limit*より後 **Then**	
9.1.1.	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *True*）	制限条件に反している
	End If	
	End If	
10..	**Exit** *Limit Conditions Check Process*（**制限条件違反**: *False*）	制限条件に反したものはない

pseudo code for limit conditions check process

ADL ノート: SCORM SN バージョン 1.3.1 の実装においてオプションでサポートされる制限条件チェックプロセス(UP.1)の擬似コードは, 灰色の強調箇所の部分である – 例: this is optional code.

シーケンシングルールチェックプロセス（Sequencing Rules Check Process）[UP.2] (アクティビティと*Rule Actions*の集合に対し; 適用する動作か*Nil*を返す):		
参照: Rule Action SM.2; Sequencing Rule Check Subprocess UP.2.1; Sequencing Rule Description SM.2		
1.	**If** アクティビティが指定されたいずれかの*Rule Actions*を持つ*Sequencing Rules*を含む **Then**	アクティビティは評価するルールを持っているかを確認する
1.1.	アクティビティに対して指定されたいずれかの*Rule Actions*を有する*Sequencing Rules*の集合を，元のルールの順序を保ちながら，選択してルールリストを初期化する	
1.2.	**For** ルールリストの各ルール	
1.2.1.	アクティビティとルールに対し*Sequencing Rule Check Subprocess*を適用する	ひとつずつ各ルールを評価する
1.2.2.	**If** *Sequencing Rule Check Subprocess*が*True*を返す **Then**	
1.2.2.1.	**Exit** *Sequencing Rules Check Process*（**アクション**: ルールの*Rule Action*）	trueと評価される最初のルールで終了する – 関連した動作を行う
	End If	
	End For	
	End If	
2.	**Exit** *Sequencing Rules Check Process*（**アクション**: *Nil*）	trueとして評価するルールがない – どんな動作も行わない

シーケンシングルールチェックサブプロセス（Sequencing Rule Check Subprocess）[UP.2.1] (アクティビティと*Sequencing Rule*に対し; ルールが適用されれば*True*を返し，ルールが適用されなければ*False*を返し，コンディションが評価できなければ*Unknown*を返す):		
参照: Rule Combination SM.2; Rule Condition SM.2; Rule Condition Operator SM.2; Sequencing Rule Description SM.2; Tracking Model TM		
1.	ルールコンディション集合を空に初期化する	ルールコンディション評価の記録をとるために使う
2.	**For** アクティビティの*Sequencing Rule*の各*Rule Condition*	
2.1.	アクティビティの適切なトラッキング情報を*Rule Condition* に適用してルールコンディションを評価する	アクティビティのトラッキング情報に対して各コンディションを評価する
2.2.	**If** *Rule Condition*の*Rule Condition Operator*が *Not* **Then**	
2.2.1.	**Negate** ルールコンディション	‘unknown’を否定すると‘unknown’に終わる
	End If	
2.3.	ルールコンディションの値をルールコンディション集合へ追加する	ルールに定義されたコンディションの集合に，このコンディションの評価を追加する
	End For	
3.	**If** ルールコンディション集合が空なら **Then**	ルールに定義されたコンディションがなければ，ルールは適用されない
3.1.	**Exit** *Sequencing Rule Check Subprocess*（**結果**: *Unknown*）	ルールコンディションはない
	End If	
4.	*Sequencing Rule*の*Rule Combination*をルールコンディション集合に適用して，単一の適合されたルール評価を得る	評価されたコンディションの集合をシーケンシングルール定義に基づき ‘And’もしくは‘Or’ にする
5.	**Exit** *Sequencing Rule Check Subprocess*（**結果**: ルール評価の値）	

pseudo code for sequencing rule check subprocess

下位試行終了プロセス（Terminate Descendent Attempts Process）[UP.3] (アクティビティに対して):		
参照: Current Activity AM.1.2; End Attempt Process UP.4		
1.	*Current Activity*と指定されたアクティビティの共通の祖先を見つける	
2.	*Current Activity*から共通の祖先まで，*Current Activity*と共通の祖先を除く，順序付き系列のアクティビティパスを作る	現在のアクティビティは終了さしていなければならない
3.	**If** アクティビティパスが空でない **Then**	終了する必要があるアクティビティがいくつかのある
3.1.	**For** アクティビティパスの各アクティビティ	
3.1.1.	アクティビティに*End Attempt Process*を適用する	各アクティビティでの現在の試行を終了する
	End For	
	End If	
4.	**Exit** *Terminate Descendent Attempts Process*	

pseudocode for terminate descendent attempts process

試行終了プロセス（End Attempt Process）[UP.4] (アクティビティに対して):		
参照: Activity is Active AM.1.1; Activity is Suspended AM.1.1; Attempt Completion Status TM.1.2.2; Attempt Progress Status TM.1.2.2; Completion Set by Content SM.11; Objective Contributes to Rollup SM.6; Objective Progress Status TM.1.1; Objective Satisfied Status TM.1.1; Objective Set by Content SM.11; Tracked SM.11; **オーバーオールロールアッププロセス** RB.1.5		
1.	**If アクティビティが葉 Then**	
1.1.	**If アクティビティの***Tracked* **が***True* **Then**	
1.1.1.	**If アクティビティの***Activity is Suspended***が***False* **Then**	**シーケンサは中断アクティビティの状態に影響を与えない**
1.1.1.1.	**If アクティビティの***Completion Set by Content***が***False* **Then**	**シーケンサがアクティビティの完了状態を設定すべきか？**
1.1.1.1.1.	**If アクティビティの***Attempt Progress Status False* **Then**	**コンテンツはアクティビティの完了状態をシーケンサに知らせたか？**
1.1.1.1.1.1.	**アクティビティの***Attempt Progress Status* **を***True* **に設定する**	
1.1.1.1.1.2.	**アクティビティの***Attempt Completion Status***を** ***True*****に設定する**	
	End If	
	End If	
1.1.1.2.	**If アクティビティの***Objective Set by Content***が***False* **Then**	**シーケンサはアクティビティの学習目標状態を設定すべきか？**
1.1.1.2.1.	**For アクティビティに関連する全学習目標**	
1.1.1.2.1.1.	**If 学習目標の***Objective Contributes to Rollup* **が** ***True*** **Then**	
1.1.1.2.1.1.1.	**If 学習目標の***Objective Progress Status***が** ***False*** **Then**	**コンテンツはアクティビティのロールアップ学習目標状態をシーケンサに知らせたか？**
1.1.1.2.1.1.1.1.	**学習目標の***Objective Progress Status* **を** ***True*****に設定する**	
1.1.1.2.1.1.1.2.	**学習目標の***Objective Satisfied Status* **を** ***True*****に設定する**	
	End If	
	End If	
	End For	
	End If	
	End If	

	End If	
2.	**Else**	**アクティビティは子を持つ**
2.1.	**If アクティビティが*Activity is Suspended*属性が*True*の子アクティビティを持っている Then**	**親の中断状態はその子の中断状態に依存する**
2.1.1.	**アクティビティの*Activity is Suspended*を*True*に設定する**	
2.2.	**Else**	
2.2.1.	**アクティビティの*Activity is Suspended*を*False*に設定する**	
	End If	
	End If	
3.	**アクティビティの*Activity is Active*を*False*に設定する**	**アクティビティの現在の試行は終了した**
4.	***Overall Rollup Process*をアクティビティへ適用する**	**アクティビティへのどのような状態変更もアクティビティツリー全体を通して伝達されることを確認する**
5.	**Exit** *End Attempt Subprocess*	

pseudocode for end attemptprocess

チェックアクティビティプロセス（Check Activity Process）[UP.5] (アクティビティに対して; アクティビティが無効ないし制限条件に反していたら*True*を返す):		
参照: Disabled Rules SM.2; Limit Conditions Check Process UP.1; Sequencing Rules Check Process UP.2		
1.	アクティビティに*Disabled* シーケンシングルールで*Sequencing Rules Check Process*を適用する	アクティビティが無効でないことを確認する
2.	**If** *Sequencing Rules Check Process*が*Nil*を返さない **Then**	
2.1.	**Exit** *Check Activity Process*（**結果**: *True*）	
	End If	
3.	アクティビティに*Limit Conditions Check Process*を適用する	アクティビティが制限条件に反していないことを確認する
4.	**If** *Limit Conditions Check Process*が*True*を返す **Then**	
4.1.	**Exit** *Check Activity Process*（**結果**: *True*）	
	End If	
5.	**Exit** *Check Activity Process*（**結果**: *False*）	アクティビティは許可されている

付録D
シーケンシング例外コード

このページは空白である.

シーケンシング例外コード

この付録では，シーケンシング擬似コード更新（付録 C 参照）で定義された様々なシーケンシング処理中に発生する例外について定義する．例外は，LMS を通じオーバーオールシーケンシングプロセス(OP)へ報告される．各々の例外は，どのシーケンシングプロセスで発生したかを示すコードによって，特定される．これは，“-”文字の直前にある例外コードの最初の部分である．

以下に示す例外は，標準擬似コードの処理中に発生する可能性のあるイベントをリストすることだけを意図している．また，これは包括的なセットではない．特に，このリストは，シーケンシングが次に配信するアクティビティを特定した後に発生する配信，起動もしくは破棄などの例外を含まない．

SCORM シーケンシングの実装は，以下に示す全てもしくは何れかの例外を実装することを強要するものではない．また，適宜追加の例外を実装する事も自由である．しかし，LMS は以下の例外を（そして，実装に追加されたどんなものも）使用し，学習行為への混乱を最小限にすることが推奨されている．

表 Appendix D – シーケンシング動作擬似コード例外

#	Code	記述
1	NB.2.1-1	*Current Activity* is already defined / Sequencing session has already begun
2	NB.2.1-2	*Current Activity* is not defined / Sequencing session has not begun
3	NB.2.1-3	*Suspended Activity* is not defined
4	NB.2.1-4	Flow Sequencing Control Mode violation
5	NB.2.1-5	Flow or Forward Only Sequencing Control Mode violation
6	NB.2.1-6	No activity is “previous” to the root
7	NB.2.1-7	Unsupported navigation request
8	NB.2.1-8	Choice Exit Sequencing Control Mode violation
9	NB.2.1-9	No activities to consider
10	NB.2.1-10	Choice Sequencing Control Mode violation
11	NB.2.1-11	Target activity does not exist
12	NB.2.1-12	*Current Activity* already terminated
13	NB.2.1-13	Undefined navigation request
14	TB.2.3-1	*Current Activity* is not defined / Sequencing session has not begun
15	TB.2.3-2	*Current Activity* already terminated
16	TB.2.3-3	Cannot suspend an inactive root
17	TB.2.3-4	Activity tree root has no parent
18	TB.2.3-5	Nothing to suspend; No active activities
19	TB.2.3-6	Nothing to abandon; No active activities
20	TB.2.3-7	Undefined termination request
21	SB.2.1-1	Last activity in the tree
22	SB.2.1-2	Cluster has no available children
23	SB.2.1-3	No activity is “previous” to the root
24	SB.2.1-4	Forward Only Sequencing Control Mode violation
25	SB.2.2-1	Flow Sequencing Control Mode violation
26	SB.2.2-2	Activity unavailable
27	SB.2.4-1	Forward Traversal Blocked
28	SB.2.4-2	Forward Only Sequencing Control Mode violation
29	SB.2.4-3	No activity is “previous” to the root
30	SB.2.5-1	*Current Activity* is defined / Sequencing session already begun
31	SB.2.6-1	*Current Activity* is defined / Sequencing session already begun
32	SB.2.6-2	No Suspended Activity defined
33	SB.2.7-1	*Current Activity* is not defined / Sequencing session has not begun
34	SB.2.7-2	Flow Sequencing Control Mode violation

35	SB.2.8-1	*Current Activity* is not defined / Sequencing session has not begun
36	SB.2.8-2	Flow Sequencing Control Mode violation
37	SB.2.9-1	No target for Choice
38	SB.2.9-2	Target activity does not exist or is unavailable
39	SB.2.9.3	Target activity hidden from choice
40	SB.2.9-4	Choice Sequencing Control Mode violation
41	SB.2.9-5	No activities to consider
42	SB.2.9-6	Unable to activate target; target is not a child of the *Current Activity*
43	SB.2.9-7	Choice Exit Sequencing Control Mode violation
44	SB.2.9-8	Unable to choice target activity – constrained choice
45	SB.2.9-9	Choice request prevented by Flow-only activity
46	SB.2.10-1	*Current Activity* is not defined / Sequencing session has not begun
47	SB.2.10-2	*Current Activity* is active or suspended
48	SB.2.10-3	Flow Sequencing Control Mode violation
49	SB.2.11-1	*Current Activity* is not defined / Sequencing session has not begun
50	SB.2.11-2	*Current Activity* has not been terminated
51	SB.2.12-1	Undefined sequencing request
52	DB.1.1-1	Cannot deliver a non-leaf activity
53	DB.1.1-2	Nothing to deliver
54	DB.1.1-3	Activity unavailable
55	DB.2-1	Identified activity is already active

付録E
ドキュメント改訂履歴

このページは空白である.

ドキュメント改訂履歴

SCORM バージョン	リリース日付	変更箇所
1.3 Working Draft 1	22-Oct-2003	イニシャルドラフト. 変更: • IMS シンプルシーケンシング仕様バージョン 1.0 の SCORM への紹介
SN Version 1.3	30-Jan-2004	変更: • SCORM 2004 の更新 • (サブ) マニフェストに関するガイドラインおよび要件の追加 • Measure Satisfaction If Active 要素に関する要件およびガイダンスの追加 • 全般的な構造および文法に関する変更
SN Version 1.3.1	22-Jul-2004	変更: • Constrained Choice の例(図 3.3.1a)のテキストの更新 • 表 3.4.5a: *Rule Condition Operator* のテキストをより明確に更新 • 表 3.7.2a: Fixed invalid element reference のコンディションに関する記述をより明確に更新 • 表 3.7.3a: 記述をより明確に更新 • セクション 3.10 の *Objective ID* 要素に使用の説明を追加 • セクション 3.10 の *Satisfied by Measure* 要素の使用の制約を追加 • 表 3.10.3a の更新. Fixed Objective Mapping error – 'read maps' は以前ステイタス情報を使用しなかった • 共有グローバル学習目標情報をローカル学習目標へマッピングする記述(セクション 4.2.1.2)の更新. • ルール評価動作の更新および 3 つの値の表をセクション 4.5.2, 4.6.2 および 4.8.4 へ追加 • データマッピング記述と要件を試行終了プロセスの記述(セクション 4.5.4)へ追加 • ロールアップの記述を更新. セクション 4.6.4 のロールアップ例を修正および追加 • target delimiter のタイプを URI から STRING へ変更 • IMS SS 1.0 から擬似コード変更履歴の削除 • 擬似コードバグ修正. 様々なマイナー修正.
SCORM 2004 3rd Edition: SN	20-Oct-2006	変更: • 文書で使用されるすべての画像の更新. 使用される色と画像のルックアンドフィールを共通化することを含

Version 1.0		む. • メタ-データをメタデータに変更. ハイフンの除去. • シーケンシングコレクション情報をアクティビティに定義されたシーケンシング情報のマージの扱いに関する情報の追加. • (サブ) マニフェストに関する説明・要件の除去. IMS が新しい IMS コンテンツパッケージの新しいバージョンの作業を終えるまで, ADL は (サブ) マニフェストの使用を推奨しないという説明を追加. 文書全体に注記を追加. • セクション 2.2 シーケンシングセッションの開始と終了を更新. シーケンシングセッションの外部でカレントアクティビティが未定義であることを明確化 • IMS シンプルシーケンシング仕様に基づくシーケンシング情報モデルの記述を更新. IMS 仕様中のデータを反映するよう種々のテーブルを更新. • セクション 3.9.1 Measure Satisfaction If Active とセクション 4.2.1.7 トラッキング動作の ADL ノートを更新 • 表 3.10a 学習目標の記述を更新. 学習目標の習得度が最小習得度を超えた場合と下回った場合を説明するため Objective Minimum Satisfied Normalized Measure の記述を追加. • セクション 3.13.1 Tracked を更新し, アクティビティがトラックされていない場合の動作の記述を追加. • セクション 4.3.1 シーケンシングループを更新し, いつ LMS がグローバル状態情報 Current Activity を None (未定義) に設定するかを明確化. • セクション 4.5.4 End Attempt プロセスを更新し, 関連する SCO ランタイム環境データモデル要素をシーケンシングトラッキングデータモデル要素に対応させる表を追加. • 表 5.2a を更新し, Suspend All ナビゲーション要求の発生元を変更. LMS のみから LMS ないし SCO に変更. また表 5.6.3b ランタイムユーザインターフェース装置語彙, 表 5.6.4a SCORM ナビゲーションデータモデル, 表 5.6.6a 要求データモデル要素のドットノーテーションバインディングを更新し, suspendAll, exitAll, abandonAll token を含めた. • すべてのシーケンシングプロセスのアルファベット順リストを追加. • ナビゲーション要求プロセスの行 7.1.1.2.2 を更新し, 「共通祖先を除く」という記述を除去. • 終了要求プロセス[TB.2.3]を更新. • 習得度ロールアッププロセス [RB.1.1]を更新し, 誤った変数への参照を記述 • ロールアップ子チェックサブプロセス [RB.1.4.2]を更新し, もれていた End If を追加. • オーバーオールロールアッププロセス [RB.1.5]を更新し, 習得度ロールアップが葉でないアクティビティのみ

		に適用される If を追加. • フローアクティビティトラバーサルサブプロセス [SB.2.2]を更新し, 探索方向の誤評価を記述. 3.3.1 行を更新し, Previous を追加. • Choice シーケンシング要求プロセス [SB.2.9]を更新し, Constrained Choise をどのように扱うかを更新 • Choice フローツリートラバーサルサブプロセス [SB.2.9.2]を更新し, もれていた Nil を追加. • 制限条件チェックプロセス [UP.1]を更新し, Active ないし Suspended のアクティビティのみを評価するように 2 行を更新. • 文書のバージョンを更新し SCORM 2004 3rd エディションとした. また, 文書の内部バージョンを 1.0 とした. • 文書全体を通じて, 必要な場合, SCORM 2004 3rd エディションへの変更に関わる情報を更新した. • アレキサンドリアの ADL Co-Lab の連絡先を更新した. • 文書全体を通じて編集上の更新を行った (例えば SCORM の登録商標の適用など). • 他の SCORM 2004 文書 (すなわち, ランタイム環境やコンテンツアグリゲーションモデル) で行われた変更を反映する全般的な更新を行った. • 関連するすべての adlnet.org への参照を adlnet.gov に変更した • ユーザインターフェース装置を提供する要件に関する情報を更新した. 配信可能なコンテンツオブジェクトの特定につながるナビゲーションイベントを発生するユーザインターフェース装置を LMS が提供する要件の説明を追加した. • 現時点で (サブ) マニフェストの使用が推奨されないことを強調する注釈を追加した. IMS グローバルコンソシアムが (サブ) マニフェストに関する IMS コンテンツパッケージ仕様の更新作業を行っている. • 明確化のために, 制御モードがアクティビティにだけ影響し, アクティビティツリーの配下には継承されないことを示す文章を追加した. • 図 3.2.3a を更新し, Continue ナビゲーションイベントがアクティビティツリーの最後のアクティビティで許されることを示した. また, このような状況で LMS がどのように動作すべきかを記述した. • 表 3.2.5a Use Current Attempt Objective Information に基づくトラッキング情報の評価, を追加した. この表は Use Current Attempt Objective Information に関連する動作を記述している. Use Current Attempt Objective Information はシーケンシングルール評価には適用されず, ロールアップルール評価にだけ適用される. • 表 3.2.6a Use Current Attempt Progress Information に基づくトラッキング情報の評価, を追加した. この表

		は Use Current Attempt Progress Information に関連する動作を記述している. Use Current Attempt Progress Information はシーケンシングルール評価には適用されず, ロールアップルール評価にだけ適用される. • より相互運用性の高い LMS UI 要件を達成するため, Constrained Choice と Prevent Activation が適用される際の目次の表示に関して, LMS に対してより厳密な記述と要件を追加した. • 表 4.9.2a シーケンシングトラッキングデータの SCO ランタイムデータへの対応の要約を追加し, アクティビティに関連する SCO が起動されるとき, SCO の cmi.objectives 要素がどのように初期化 (更新) されるかを記述した. • 疑似コードのいくつかのセクションを, アクティビティツリー状態の境界条件を考慮し, 相互運用性のある動作を行うよう更新した. • Constrained Choice がユーザインターフェースナビゲーション装置の提供に関連していることから, 関連する情報を更新した. LMS は Constrained Choice を確認して, 影響を受けるアクティビティの UI 装置を使用できないようにしなくてはならない. • 読み出しマップとグローバル学習目標に関する記述を更新した. グローバル学習目標からの読み出しマップの処理は, 学習目標のローカル状態が未定か否かによらず行わなくてはならない. • 学習目標システムグローバルに関する記述を更新した. もしこの値が False の場合, アクティビティツリーで定義された共有グローバル学習目標の状態はアクティビティツリーの新しい試行ごとにリセット (未定に設定) される. • オーバーオールロールアッププロセスに関する記述を変更した. オーバーオールロールアッププロセスを起動しても, クラスタアクティビティの現在のローカルトラッキング情報は変更されない. • 表 4.5.1a SCORM 2004 終了要求を更新し, Exit Parent 要求を追加した. これは表からもれていたものである.