SCORM 2004 解説書

第 1.0.6 版

2010 年 6 月

特定非営利活動法人日本イーラーニングコンソシアム

改訂履歴

日付	バージョン	改訂内容
2005 年 11 月	1.0	初版
	1.0.1	図表等一部修正
2006 年 2 月	1.0.2	誤植等修正
	1.0.3	同上
2006 年 3 月	1.0.4	同上
2006 年 3 月	1.0.5	終了ルール→ポストコンディションルール 学習目標，ステータスと Satisfied By Measure，閾値 ローカル→共有への反映 SN-4-18 複数学習目標を参照したルールの例 葉アクティビティに共有学習目標が割り当てられている場合の優先順位 Measure Satisfaction if Active
2010 年 6 月	1.0.6	共有学習目標への Write Map の記述．ある一つアクティビティの複数のローカル目標からの書込みは不可.

目次

1. はじめに

WBT(Web-based Tranining)のコンテンツに関する SCORM(Sharable Content Object Reference Model)規格が実用的に使用されるようになって数年が経つ．この間，SCORM 規格に準拠した LMS(Learning Management System)，コンテンツ，オーサリングツールが国内外で数多く現れ，幅広く使用されるようになった．現在，一般に使用されている規格は 2000 年に発表された SCORM 1.2 である．SCORM 1.2 は多くの製品で使用されているが，一方で，機能面での不足，規格のあいまいさ，などが指摘されていた．これらの問題点を解決するために ADL が 2004 年に新たに公開したのが，本書で解説する SCORM 2004 規格である．SCORM 2004 では，シーケンシング，ナビゲーションといった新しい機能が追加されるとともに，規格の記述の全面的に詳細化が図られている．このため，実用的にほぼ満足のいく内容となっているが，一方で規格書は全部で 800 ページを越えており，全体像を規格書から把握するのは簡単ではない．このような状況をふまえ，本書では SCORM 1.2 に関してはある程度の知識を有している方を対象に，SCORM 2004 規格の全体像，新たに追加された機能，SCORM 1.2 との差分，といった内容の解説を行った．本書を読んでから規格書に目を通すことで，規格の理解が促進されることをねらいとしている．

以下，2.で SCORM 2004 の概略を，これまでの規格との相違点を中心に示す．3.と 4.で SCORM 2004 で新たに追加された機能であるシーケンシングとナビゲーションについて，その概略，規格書では理解が難しい点を中心に述べる．5.ではランタイム環境（Runtime Environment: RTE）について，SCORM 1.2 からの変更点を中心に解説する．6.，7. では主に LMS を実装する観点から，シーケンシングと RTE の実現方法，注意点を示す．8. は，SCORM 1.2 から SCORM 2004 への移行に関する説明である．

なお，本書中で以下の記号は，SCORM 2004 規格書の各分冊を示す．

OV: SCORM 2004 Overview 2ndEdition
CAM: SCORM Content Aggregation Model Ver.1.3.1
RTE: SCORM Run-time Environment Ver.1.3.1
SN: SCORM Sequencing and Navigation Ver.1.3.1
CR: SCORM 2004 Conformance Requirements Ver.1.3
ADD: SCORM 2004 2nd Edition Addendum Ver1.2

2. SCORM 2004 概要

本節では，SCORM 1.2 から SCORM 2004 への追加規格や変更点を中心に，ｅラーニング標準規格の意義を SCORM 規格の変遷と共に解説する．

2.1 SCORM とは

SCORM（Sharable Content Object Reference Model）はｅラーニングにおける学習コンテンツの共有化を図るため，

- 耐久性
- 相互運用性
- アクセス可能性
- 再利用性

を実現する仕様の標準化を目指してアメリカの ADL（Advanced Distributed Learning）が提示している規格である．

ｅラーニングコンテンツの流通のため，システムやソフトウェアのバージョンアップなどでも修正の必要がなく（耐久性），多くの OS や Web ブラウザなどで学習可能で（相互運用性），必要なときに学習教材が検索でき（アクセス可能性），既存のコンテンツを容易に再利用して新規コンテンツの作成を可能にする（再利用性）ための規格といえる．

2.2 SCORM 規格の成り立ち

SCORM 規格は，IMS(IMS Global Learning Consortium inc.)や AICC(the Aviation Industry CBT Committee)，ARIADNE(Alliance of Remote Instructional Authoring & Distribution Network for Europe)，IEEE LTSC(Institute of Electrical and Electronics Engineers, Learning Technology Standards Committee)などの技術仕様やガイドラインを参照し，これらを固有のコンテンツモデルに適用し一貫性のある実装のための推奨基準を開発することを目指したものであり，主に以下のような規格の影響を受けている．

・SCORM 2004 CAM

IEEE Learning Object Metadata (LOM)

IMS Content Packaging

IEEE Extensible Markup Language (XML) Schema Binding for Learning Object Metadata Data Model

・SCORM 2004 RTE

IEEE Data Model For Content Object Communication

IEEE ECMAScript Application Programming Interface for Content to Runtime Services Communication

・SCORM 2004 SN

IMS Simple Sequencing

2.3 LMS モデル

次の図は一般的な LMS モデルである．LMS はこの図に示すとおり，学習者プロファイルサービスやコンテンツ管理サービスなどさまざまな機能を有しているが，SCORM ではこれらの具体的な実装方法については定義をしない．SCORM 規格が取り扱っているのは，コンテンツと LMS のインタフェースについてである．具体的にはコンテンツの LMS への登録，コンテンツの起動，LMS とコンテンツとのデータのやり取りなどについて SCORM では定義している．

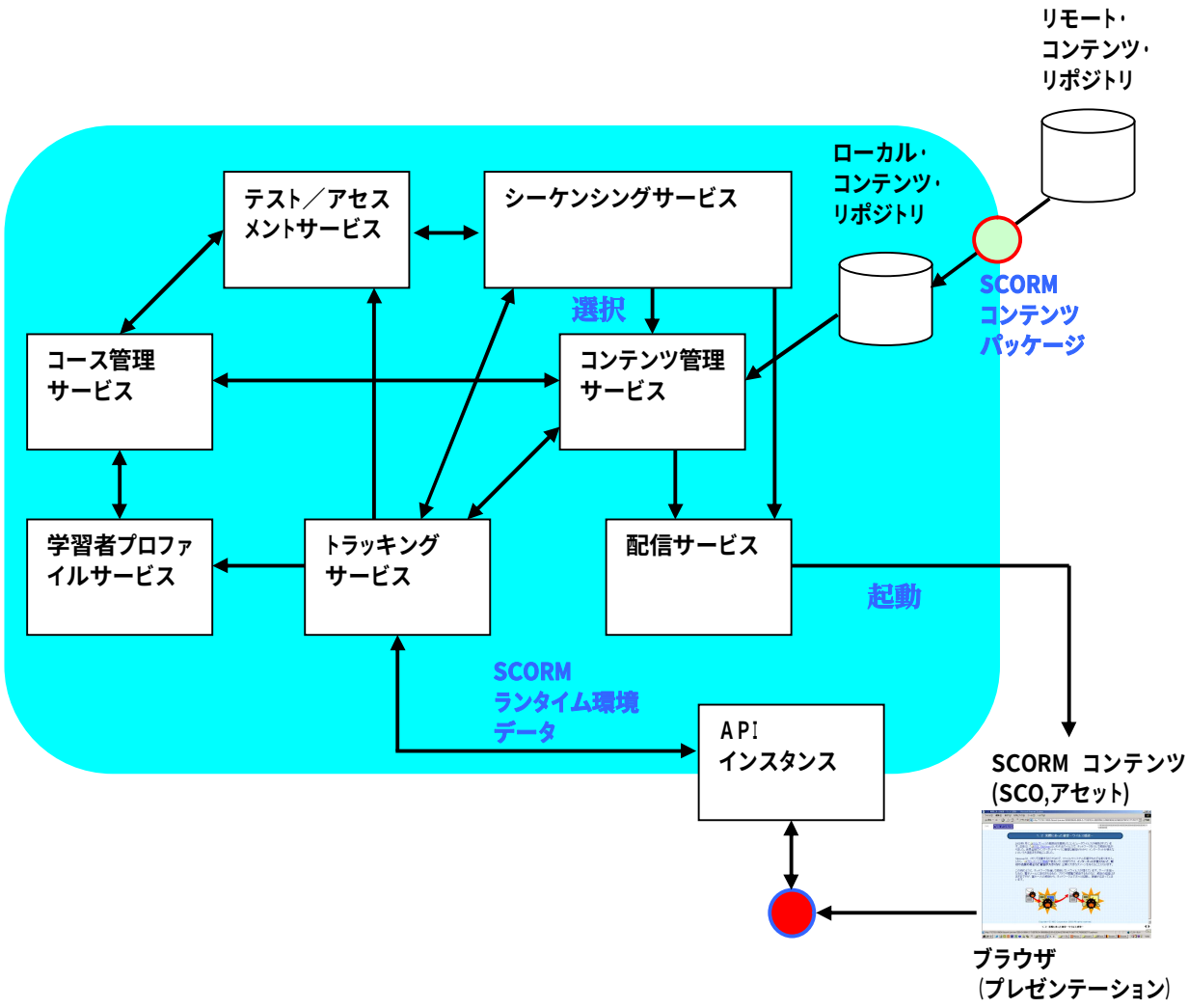

図 2.1 LMS モデル

2.4 SCORM 2004 概要

SCORM 2004 は ADL が SCORM 1.2 の後継規格として開発したものである．SCORM 2004 のもっとも大きな特徴は，シーケンシング＆ナビゲーションに関する仕様書が新たに追加されたことである．これにより，以前のバージョンでは記述することのできなかった，学習の順序だてや学習者の学習状況に応じた動的なコンテンツのふるまいをコンテンツ側で制御できるようになったり，「次へ進む」「前へ戻る」などのコマンドを LMS ではなくコンテンツ側で提供できるようになったりと，コンテンツ作成者の教材設計・開発の自由度が高くなった．

SCORM 1.2 では仕様書は，SCORM 概要，コンテンツアグリゲーションモデル，ランタイム環境の３編で構成されていたが，SCORM 2004 仕様書ではさらにシーケンシング＆ナビゲーションの規格が追加されたため，以下の４編から構成されている．

(1) SCORM 2004 **概要**（SCORM 2004 Overview Book）

ADL および SCORM の歴史や目的，SCORM が参照している仕様書および標準規格，各々の SCORM 仕様書の関連などについて記述されている．また，LMS とコンテンツの役割分担についてもこのブックに記述されている．

(2) **コンテンツアグリゲーションモデル**（The SCORM Content Aggregation Model(CAM) Book）

学習コンテンツを識別し，組み立てるためのガイドラインが記述されている．つまり SCORM コンテンツを設計する際に理解するべき事柄についてふれられている．このガイドラインは，IEEE LOM1484.12，AICC コンテンツ構造，IMS コンテンツパッケージング，IMS シーケンシング情報といった規格をもとに構成されている．

SCORM 技術の領域としては，SCO，アセット，コンテンツアグリゲーション，パッケージ，パッケージ交換ファイル（PIF），メタデータ，マニフェストファイル，シーケンシング＆ナビゲーションに関する情報が記述される．

(3) **ランタイム環境**（The SCORM Run-Time Environment(RTE) Book）

Web ベース環境でのコンテンツ起動，通信，受講履歴に関するガイドラインが記述されている．ここでは学習者が LMS を通じて学習資源を起動し，学習状況の送受信を行い，学習を終了するまでの一連の活動に必要な事柄についてふれられている．このガイドラインは，IEEE API 1484.11.2，IEEE Data Model 1484.11.1 といった規格をもとに構成されている．

SCORM 技術の領域としては，API，API インスタンス，起動，セッション・データ・サポートの方式，ランタイムデータモデルに関する情報が記述される．

(4) **シーケンシング＆ナビゲーション**（The SCORM Sequencing and Navigation(SN) Book）

SCORM 2004 で新たに追加された仕様書であり，SCORM 規格の最も大きな変更点である．ここでは学習コンテンツをどのように提示するかといった順序づけ（ふるまい）に関するガイドラインが記述されている．このガイドラインは，IMS シーケンシング情報＆ビヘイビア規格をもとに構成されている．また，ナビゲーション GUI に関してもこの仕様書でふれられている．

SCORM 技術の領域としては，アクティビティツリー，学習アクティビティ，シーケンシング情報，ナビゲーション情報，ナビゲーションデータモデルに関する情報が記述される．

これらの仕様書はそれぞれの領域に特化して記述されているが，一部相互に関連する領域も存在し，その際は相互に参照し合うよう記述がなされている．

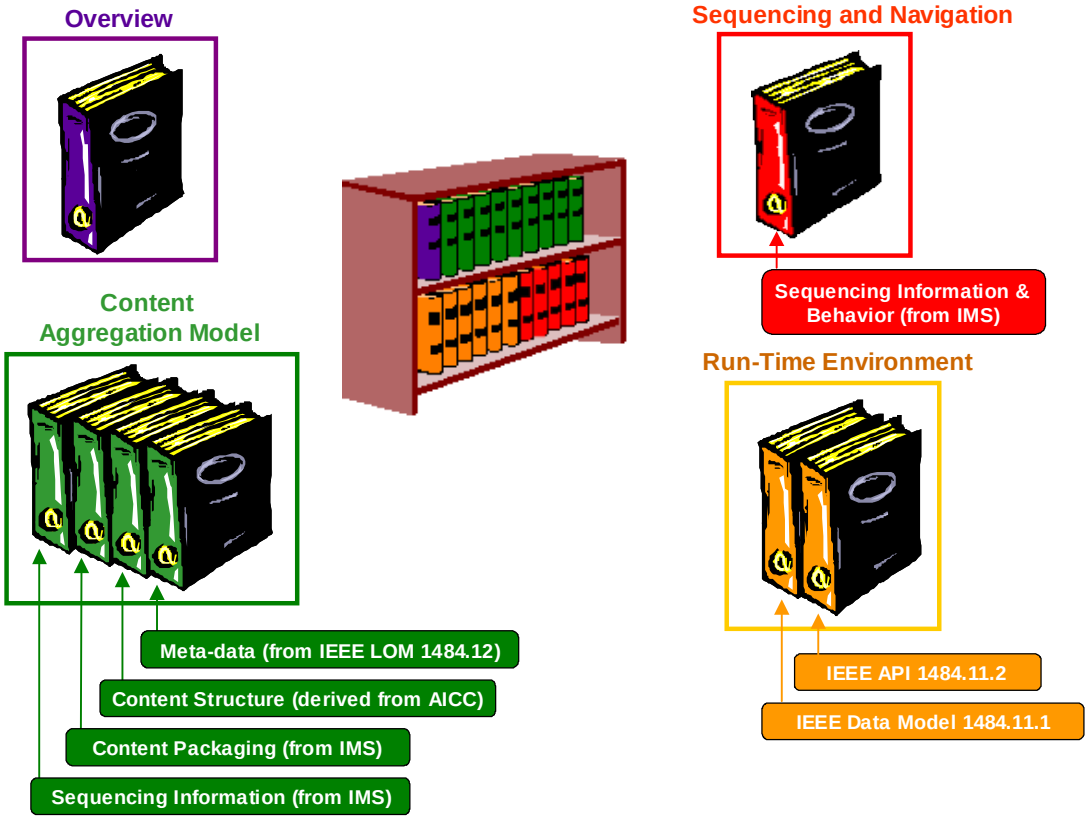

図 2.2　SCORM 2004 の構成（出典：『SCORM 2004 2nd Edition Overview』ADL）

2.5　SCORM 規格の変遷（SCORM 1.0 から SCORM 1.2）

SCORM は，過去の SCORM バージョンから，コンセプトや必要条件の明確化，標準化の推進，ADL コミュニティによるベストプラクティス，拡張，バグフィックスなどにより，さまざまな変更が加えられてきた．

SCORM は，まず SCORM 1.0 として試験・評価段階に入った．SCORM 1.0 では試験・評価の参加者から実装に際し多くの質問や課題が出された．

そこで，SCORM 1.1 では SCORM 1.0 の対象範囲を修正拡張せずに，これらの初期参加者からのフィードバックに基づく修正や改良が行われた．最も顕著な変更は名称の変更である．SCORM 1.0 では Sharable Courseware Object Reference Model としていたものが，SCORM 1.1 では Sharable Content Object Reference Model へと変更された．この変更は SCORM で参照している技術仕様がコース全体だけではなく各種レベルのコンテンツに適用されるという実態を反映している．また，SCORM 1.1 では，それぞれの仕様をサブセクションに分けつつ仕様を機能別グループに分けて使いやすい構成にした．さらに，ランタイム環境の API の重要な改良と変更が行われるとともに，SCORM が参照している AICC CMI 規格の簡素化が行われた関係で，SCORM ラインタイム環境のデータモデルのいくつかのデータ項目が削除された．

SCORM 1.2 では，IMS コンテンツパッケージング規格に基づく，SCORM コンテンツパッケージアプリケーションプロファイルが追加された．また，メタデータを IMS および IEEE LTSC

で開発された最新仕様を参照するように更新された．この更新は情報モデルおよび XML バインディングの変更を含んでいる．さらに，このバージョンではメタデータアプリケーションプロファイルの名称を変更し，SCORM コンテンツアグリゲーションモデルへの変更，および IMS コンテンツパッケージングの命名法により合致するようにした．

2.6 SCORM 1.2 から SCORM 2004 への変更点

SCORM 1.2 から SCORM 2004 への主な変更点を以下に示す．

2.6.1 仕様書バージョン表記の変更

SCORM 2004 では各仕様書の保守・独立性を高めるため SCORM のバージョン表記に関する記述が変更された．CAM や RTE といった各仕様書ごとに“Version1.3”のようなリリース番号をつけることとし，今後，更新があった仕様書のバージョンのみが変更されることになる．

2.6.2 シーケンシング機能の追加

SCORM 仕様書にシーケンシング＆ナビゲーション仕様書が新たに追加された．

これにより，SCORM 1.2 までの従来の規格の範囲内では設定することができなかった学習のシーケンス制御を行うことができるようになった．

例えば，学習コンテンツの提示順序を指定する，学習事前にプリテストを実施し，その結果により，学習するコンテンツの種類や順番を変更する，問題Ａと問題Ｂに合格するとコース修了とする，問題演習が不合格なら復習を繰り返す，学習目標を習得するまで解説と問題を繰り返す，といったことが制御できるようになった．

コンテンツ作成者は，コース構造とそれに付随するシーケンシングルールをマニフェストファイルに記述することによってコンテンツの動作を制御する．

学習の習得状態や進捗状態などさまざまな条件の組み合わせによって学習の経路や状態設定が可能になるので，学習者適応型のコンテンツやシミュレーション教材の作成などができるようになる．

2.6.3 SCO からのナビゲーションコマンド発行機能の追加

SCORM 2004 で新たに追加されたシーケンシング＆ナビゲーション仕様書では，SCO のナビゲーション方法に関する新たな仕様も追加された．

これにより，「次へ進む」「前へ戻る」といった SCO ナビゲーションコマンドの発行を SCO から行うことができるようになった．さらに，LMS が提供しているナビゲーションボタンの表示／非表示をコンテンツで制御できるようになった．

コンテンツ作成者は，新しく追加されたランタイムデータモデルを SCO に記述することで SCO ナビゲーション制御を行う．また，LMS ナビゲーションボタンの表示／非表示はマニフェストファイルに記述することで制御を行う．

この規格を用いることで，コンテンツ作成者は LMS の種類を気にすることなく，学習コンテンツの重要な要件であるナビゲーションの設計を行うことができる．

2.6.4 SCORM ランタイム環境の変更

SCORM ランタイム環境について，SCORM 2004 では SCORM 1.2 から大きく変更がなされた.

ここでは変更点の概略について述べる.

(1) API オブジェクト名の変更

API インプリメンテーションのオブジェクト名が API から API_1484_11 に変更された.

(2) API 関数名の変更

API 関数名が下記のとおり変更された.

表 2.1 API 関数名の変更

SCORM 1.2	SCORM 2004
LMSInitialize(“”)	Initialize(“”)
LMSFinish(“”)	Terminate(“”)
LMSGetValue(parameter)	GetValue(parameter)
LMSSetValue(parameter_1,parameter_2)	SetValue(parameter_1,parameter_2)
LMSCommit(“”)	Commit(“”)
LMSGetLastError()	GetLastError()
LMSGetErrorString(parameter)	GetErrorString(parameter)
LMSGetDiagnostic(parameter)	GetDiagnostic(parameter)

(3) データモデルの変更

主な変更点を下記の示す.

- すべてのデータモデルが LMS の必須（mandatory）要素となった.
- データモデルの平坦化が行われ, cmi.core および cmi.student_data エレメントが廃止された.
- 回答や正答情報記述フォーマットが精密化されるなど interaction が詳細化された.
- マルチバイト文字コードが全面採用（ISO 10646）され多言語化がなされた.
- lesson_status が廃止され，completion_sutatus と success_status に分離移行された. completion_status は完了状態に対応し，状態 completed，incomplete，not attempted，unknown を扱う．success_status は習得状態に対応し，状態 passed，failed，unknown を扱う．browsed 状態は廃止された.
- 習得度に対応する score.scaled が追加された. これに合わせ score.raw の 0～100 点までという値の制限は廃止された.
- データモデルの objectives とアクティビティの学習目標とに対応付けがなされ，共有グローバル学習目標の設定が可能になった.

- エラーコードが詳細化され，API インスタンスの状態やデータの一貫性のチェックなどが可能になった．

2.6.5 SCORM コンテンツアグリゲーションモデルの変更

SCORM コンテンツアグリゲーションモデルでは，シーケンシング＆ナビゲーション規格導入に伴う内容の追加や参照される XML スキーマ等の変更がなされた．

また，ADL コンテンツパッケージングの以下の要素は廃止され，関連するシーケンシングルールに変更された．

- <adlcp:prerequisites>
- <adlcp:masteryscore>
- <adlcp:maxtimeallowed>

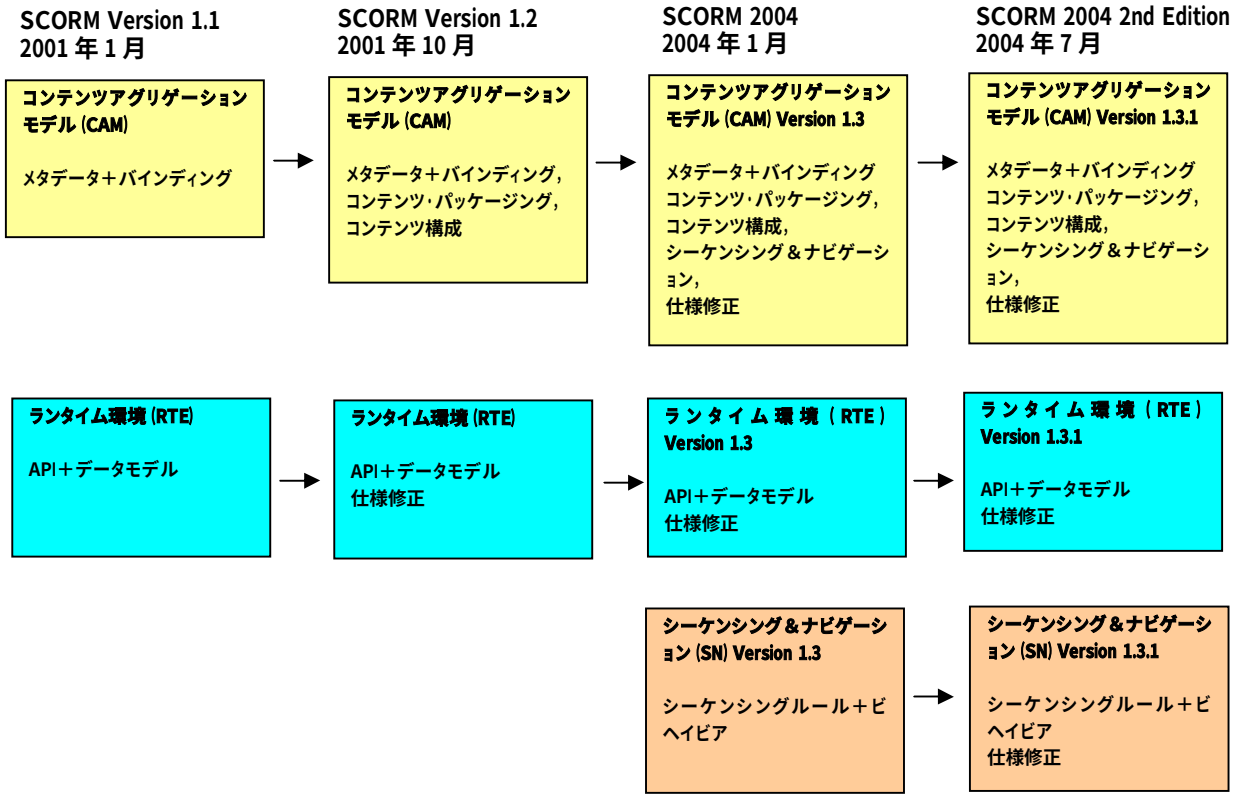

図 2.3 SCORM の進展

2.7 SCORM の今後

ADL は今後 Web ベースの学習基盤の新機能の候補として，

- 新しいランタイムおよびコンテンツデータモデルアーキテクチャの設計
- シミュレーションの組み込み
- 電子パフォーマンスサポートオブジェクトの組み込み

- SCORM ベースの知的学習支援機能の実装
- 新しいコンテンツモデルの設計
- ゲーム技術の組み込み

を挙げている.

なお，ADL では SCORM の次期バージョンの予定はないとしている.

3. シーケンシング

本節では，SCORM 2004 で新たに導入されたシーケンシング機能の基本となる概念とそれらの全体的な関連について説明し，シーケンシング動作がどのように規定されるかを説明する．図 3.1 にシーケンシング動作の概要を示す．コンテンツ作成者は，コンテンツ構造とそれに付随する動作ルール（シーケンシングルール）をマニフェストファイル（imsmanifest.xml）に記述することによってコンテンツの動作を制御する．マニフェストファイルは LMS に読み込まれて実行される．実行時に LMS は，学習者からの要求（ナビゲーション要求）を受け取り，学習者の学習状態を反映するための状態情報（トラッキング情報）を更新し，シーケンシングルールを解釈して，次の提示画面を決定して配信するという動作を繰り返す．

シーケンシングの主要な構成要素と外部機能は，

- コンテンツ構造と学習目標
- トラッキング情報
- ナビゲーション要求とシーケンシング要求
- シーケンシングルール

で規定される．実行時のシーケンシング動作は図 3.1 右側の「プロセス」の集合として規定され，各プロセスの動作は擬似コードとして規格で定義されている．プロセスと擬似コードの詳細な説明は 6. で行い，本節ではシーケンシングの外部機能を上の４つの要素によって説明する．

3.1 コンテンツ構造と学習目標

SCORM 2004 のコンテンツ構造は階層型（木構造型）である．各ノードはアクティビティ（Activity）と呼ばれる．コンテンツ構造全体をアクティビティツリー（アクティビティ木）と呼ぶ．階層構造の末端のアクティビティには，ブラウザに配信される学習資源（SCO ないしアセット）が付随する．

あるアクティビティとその直下の子アクティビティの集まりをクラスタと呼ぶ．例えば，図 3.1 で，アクティビティ 1.1.3, 1.1.3.1, 1.1.3.2 は 1.1.3 を親とするクラスタを構成する．また，1.1, 1.1.1, 1.1.2, 1.1.3 は 1.1 を親とするクラスタを構成する．クラスタはシーケンシング動作の単位であり，多くの場合，親アクティビティに記述されたルールがクラスタに対して適用される．

すべてのアクティビティにはデフォルトで学習目標（Objective）が必ず一つ付随する．この学習目標を主学習目標 (Primary Objective) と呼ぶ[1]．主学習目標の役割については，ロールアップの項で詳しく述べる．アクティビティと学習目標は 3.2 に述べるトラッキング情報を保持する．コンテンツ作成者は，デフォルトの学習目標以外に複数のアクティビティで共有される学習目標を定義することができる．すなわち，ひとつのアクティビティにはデフォルトの学習目標以外に

[1]主学習目標は，ロールアップ学習目標とも呼ばれる．CAM では，PrimaryObjective タグでこれを表すため，主学習目標の名称を用いている．SN では，アクティビティに関連付けられた学習目標のうち Objective Contributes to Rollup が True の学習目標と定義されているためロールアップ学習目標と呼ばれている．両者は同じものであり，本書では主学習目標という用語を用いる．

複数の共有グローバル学習目標を関連付けることができ，ひとつの共有グローバル学習目標は複数のアクティビティから共有される．アクティビティと共有グローバル学習目標の間には，読み書きの関係を定義するようになっている．すなわち共有グローバル学習目標のトラッキング情報の状態はアクティビティから書き込まれるトラッキング情報の値によって決まる．また，アクティビティは共有グローバル学習目標のトラッキング情報の値を読み出して，シーケンシングルールで参照することができる．共有グローバル学習目標については 3.4.6 で詳しく述べる．

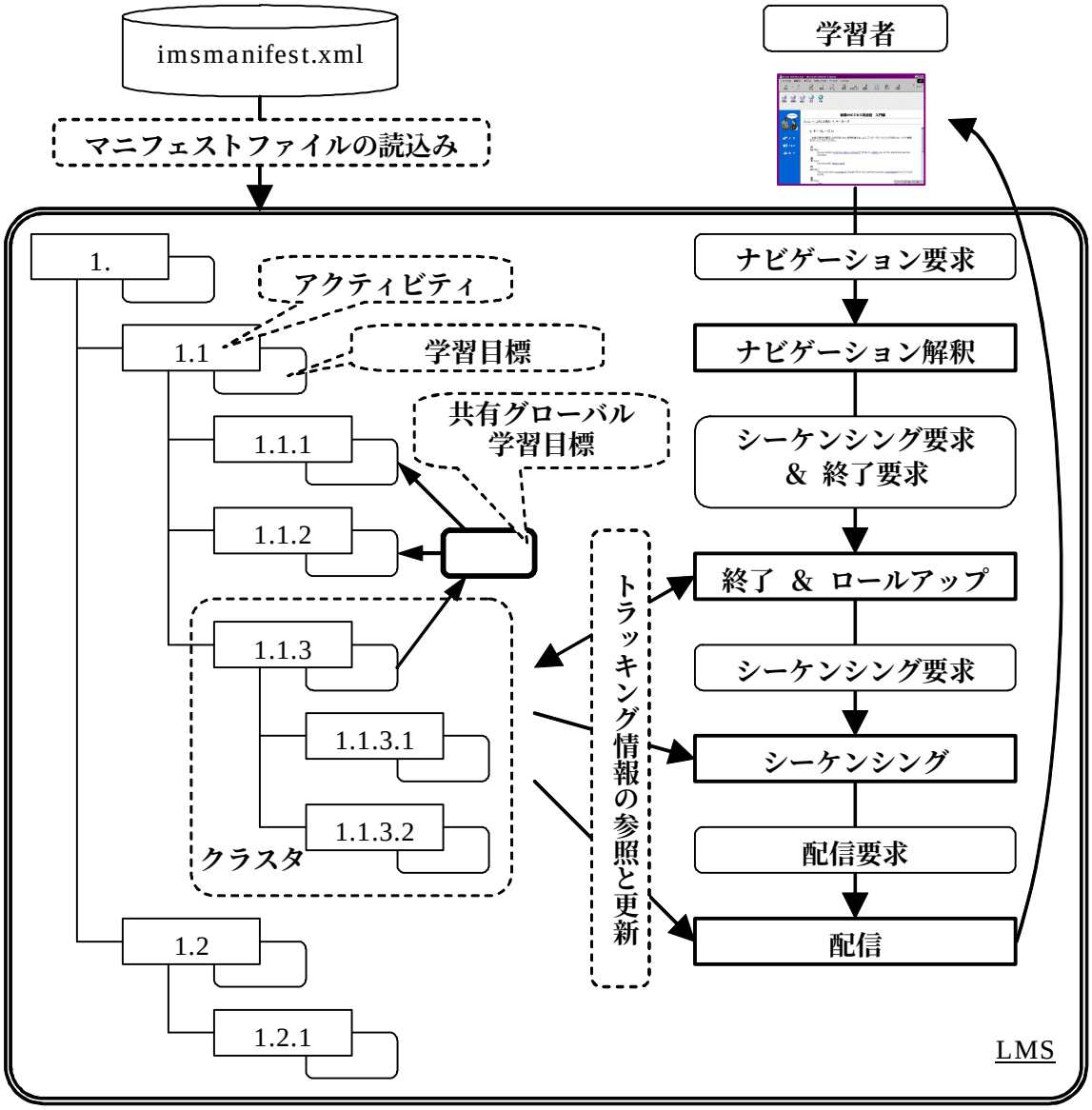

図 3.1 シーケンシング処理の概要

3.2 トラッキング情報

トラッキング情報は，学習者の学習状態を反映するための状態情報である．トラッキング情報は各アクティビティと学習目標に付随する．トラッキング情報は表 3.1 のように分類される．

トラッキング情報は，学習の習得，完了に関する情報と，学習時間，試行回数に関する情報に分けることができる．以下にそれぞれについて説明する．なお，ここで，あるアクティビティの一回の学習を開始してから終了するまでをアテンプトと呼ぶ．階層の末端のアクティビティで SCO を起動してから終了するまで, 階層の中間のアクティビティで配下のアクティビティの学習を開始してから，配下以外のアクティビティに移動するまでが 1 回のアテンプトとなる．アクティビティは複数回学習することができるから，アテンプトは複数存在する．アテンプトについては 3.5 で詳しく述べる．

表 3.1 トラッキング情報

学習目標進捗情報 Objective Progress Information	アクティビティ進捗情報 Activity Progress Information	アテンプト進捗情報 Attempt Progress Information
学習目標習得状態 Objective Satisfied Status		アテンプト完了状態 Attempt Completion Status
学習目標習得度 Objective Normalized Measure		アテンプト完了度 Attempt Completion Amount
	アクティビティ累積期間 Activity Absolute Duration	アテンプト累積期間 Attempt Absolute Duration
	アクティビティ経験期間 Activity Experienced Duration	アテンプト経験期間 Attempt Experienced Duration
	アクティビティ試行回数 Activity Attempt Count	

3.2.1 学習の習得，完了に関するトラッキング情報

SCORM 2004 では，学習の「習得」と「完了」を区別して管理する．これは，ある教材を最初から最後まで学習（完了）しても，内容が理解できていなければ不合格（未習得）であり，逆にすべて学習していなくても（未完了でも), 内容が理解できていれば合格（習得）とみなされる，ということに対応している．

「習得」は学習目標に関連した情報であり，ある学習目標を習得（合格）したか，未修得（不合格）か，習得の度合いはどの程度か，で表される．表 3.1 では，「学習目標習得状態」,「学習目標習得度」がこれに該当する．

一方,「完了」はアクティビティの一回の試行（アテンプト）に関連した情報であり，アテンプトにおいて, アクティビティを完了したか, 未完了か, 完了の度合いはどの程度か, で表される．表 3.1 では，「アテンプト完了状態」,「アテンプト完了度」がこれに該当する．

これらの情報は，階層の末端のアクティビティやそれに付随する学習目標では，対応する SCO からのランライム環境情報によって更新される．表 3.2 にトラッキング情報とランタイム環境情報の対応を示す．

一方，各クラスタでは，親アクティビティの情報は子アクティビティの情報を用いて更新される．すなわち，トラッキング情報は，教材全体では，SCO の情報に基づき，末端のアクティビティ→中間アクティビティ→ルートアクティビティと伝播されていく．この動作をロールアップと呼ぶ[2]．ロールアップによって，親アクティビティの情報をどのように更新するかはコンテンツ作成者が指定できる．ロールアップについては 3.4.5 で述べる．

[2] アテンプト完了度は，現在の規格ではロールアップの対象とならない．

表 3.2 トラッキング情報とランタイム環境情報の対応

トラッキング情報報		ランタイム環境情報
アテンプト完了状態 Attempt Completion Status		完了状態 cmi.completion.status
アテンプト完了度 Attempt Completion Amount		完了度 cmi.progress_measure
学習目標習得状態 Objective Satisfied Status	主学習目標	習得状態 cmi.success_status
	それ以外の学習目標	（学習目標の）習得状態 cmi.objectives.n.success_status
学習目標習得度 Objective Normalized Measure	主学習目標	正規化得点 cmi.score.scaled
	それ以外の学習目標	（学習目標の）正規化得点 cmi.objectives.n.score.scaled

3.2.2 学習時間，試行回数に関するトラッキング情報

学習時間は，アテンプト累積期間，アテンプト経験期間，アクティビティ累積期間，アクティビティ経験期間で表される．

アテンプト累積期間は一回のアテンプトの開始から終了までの総学習時間である．アテンプト経験期間は一回のアテンプトの開始から終了までの総学習時間で，途中の中断時間を除いたものである．中断がなければ両者は一致する．

アテンプト累積期間，アテンプト経験期間は，アクティビティの総学習時間に相当し，それぞれ，アテンプト累積期間，アテンプト経験期間を累計したものになる．

アクティビティ試行回数は，そのアクティビティのアテンプトの回数である．

これらの情報は，学習実行時に LMS が自動的に更新していく．

3.3 ナビゲーション要求，シーケンシング要求，終了要求

学習者からの「Continue（次へ進む)」,「Previous（前へ戻る)」，などの要求をナビゲーション要求と呼ぶ．ナビゲーション要求の種別を表 3.3 に示す．ナビゲーション要求は，学習者からブラウザを介して入力されるが，このとき，LMS の提供する GUI を用いる場合と，SCO からナビゲーション要求を発行する場合がある．SCO からナビゲーション要求を発行する方法については 4.で述べる．

ナビゲーション要求は，LMS の中で図 3.1 のナビゲーション解釈処理によって，シーケンシング要求と終了要求に分離される．ナビゲーション要求に対応するシーケンシング要求と終了要求を表 3.3 に示す．また，それぞれの説明を表 3.4，表 3.5 に示す．シーケンシング要求は，教材の開始，アクティビティ間遷移の契機となる．終了要求は，教材の中断，終了を行う．

シーケンシング要求，終了要求は，3.4.4 に述べるポストコンディションルールによって別のシーケンシング要求，終了要求に変換される場合がある．表 3.4 に示したもののうち Retry シーケンシング要求はナビゲーション要求から発行されることはなく，ポストコンディションルールによってのみ生成される．シーケンシング要求により，LMS は，現在提示しているアクティビティから他のアクティビティへの遷移を行い，学習者に提示する次のアクティビティを決定する．こ

のとき 3.4.2, 3.4.3 に述べる制限条件，プリコンディションルールのチェックが行われる．

表 3.3 ナビゲーション要求

名称	説明	シーケンシング要求	終了要求
Start	教材の学習を開始する．	Start	
Resume All	教材の学習を前回中断した状態から再開する．	Resume All	
Continue	前方に進む．	Continue	Exit
Previous	後方に進む．	Previous	Exit
Forward	現バージョンでは未使用．		
Backward	現バージョンでは未使用．		
Choice	指定されたアクティビティに進む．	Choice	Exit
Exit	現在のアクティビティを終了する．	Exit	Exit
Exit All	教材全体を終了する．	Exit	Exit All
Suspend All	教材全体の再開に必要な情報を記録して，中断する．	Exit	Suspend All
Abandon	現在のアクティビティを放棄する．	Exit	Abandon
Abandon All	教材全体を放棄する．	Exit	Abandon All

表 3.4 シーケンシング要求

名称	説明
Start	教材の学習を開始する．
Resume All	教材の学習を前回中断した状態から再開する．
Continue	前方に進む．
Previous	後方に進む．
Choice	指定されたアクティビティに進む．
Retry	アクティビティを再実行する．
Exit	現在のアクティビティを終了する．

表 3.5 終了要求

名称	説明
Exit	現在のアクティビティを終了する．
Exit All	教材全体を終了する．
Suspend All	教材全体の再開に必要な情報を記録して中断する．
Abandon	現在のアクティビティを放棄する．
Abandon All	教材全体を放棄する．

3.4 シーケンシングルール

シーケンシングルールは，コンテンツ作成者の記述するシーケンシング動作の定義である．シーケンシングルールは以下のように大別される．これらはいずれもアクティビティ毎に定義される．

- シーケンシング要求やアクティビティ間遷移動作に制限を加えるもの．固定的な制限と，トラッキング情報がある条件を満たすときに成立する制限がある．前者はシーケンシング制御モードと呼ばれ，例えば「クラスタ中の子アクティビティは順方向のみに提示し，後戻りを禁止する」のようなものである．後者の例として「もし学習目標の習得状態が修得済みならばアクティビティをスキップする」というような形のプリコンディションルールと,「アクティビティの総実行時間は 30 分以内」というような制限条件がある．
- トラッキング情報がある条件を満たすときに，特定のシーケンシング要求を発生するもの．この形式のルールはポストコンディションルールと呼ばれる．例えば,「もし学習目標の習得状態が未修得ならばアクティビティを再試行する」のようなものである．ポストコンディションルールは図 3.1 の「終了＆ロールアップ」の段階で評価実行される．
- トラッキング情報の更新に関するもの．3.2 に述べたように，アクティビティと学習目標のトラッキング情報は，SCO に対する学習者入力による状態変化を契機として，SCO に対応する末端アクティビティから階層の最上位アクティビティに向かって更新される．この更新動作をロールアップと呼ぶ．子アクティビティのロールアップへの関与，ロールアップが生じる条件と結果を記述することができる．例えば,「子アクティビティのうち 3 つ以上が完了ならば親アクティビティは完了」などのロールアップルールが記述可能である．ロールアップルールは図 3.1 の「終了＆ロールアップ」の段階で評価実行される．

以上のシーケンシングルールの分類は，シーケンシング動作という観点からは，以下のように整理できる．

(1) トラッキング情報の更新

図 3.1 の「終了＆ロールアップ」の段階でロールアップルールが評価され，アクティビティツリーの各アクティビティのトラッキング情報が更新される．

(2) シーケンシング要求の確定

次に，同じく図 3.1 の「終了＆ロールアップ」の段階でポストコンディションルールが評価される．ポストコンディションルールの条件が成立した場合は，学習者からのナビゲーション要求に基づくシーケンシング要求は，ポストコンディションルールによるシーケンシング要求で置き換えられる．

(3) 配信アクティビティの決定

確定したシーケンシング要求に基づき，図 3.1 の「シーケンシング」および「配信」の段階で，配信するアクティビティを決定する．このとき，シーケンシング制御モード，プリコンディションルール，制限条件を参照して，配信するアクティビティが選択される．

以下の節で，各々のシーケンシングルールについて詳しく述べる．

3.4.1 シーケンシング制御モード

シーケンシング制御モードは，クラスタにおけるシーケンシング動作の制御を行う．シーケンシング制御モードには大きく以下のタイプがある．

- 特定のナビゲーション要求を有効にするもの（Choice, Flow）
- アクティビティ間遷移動作に制限を加えるもの（Choice Exit, Forward Only）
- トラッキング情報の評価方法を制御するもの（Use Current Attempt Objective Information, Use Current Attempt Progress Information）

表 3.6 にこれらを示す．

表 3.6 シーケンシング制御モード

名称	説明
Choice	子アクティビティに対する Choice ナビゲーション要求を有効にする．
Choice Exit	自身および配下のアクティビティから Choice ナビゲーション要求で他のアクティビティに移動することを禁止する．
Flow	クラスタ内で Continue, Previous ナビゲーション要求を有効にする．
Forward Only	クラスタ内での後方への移動を禁止する．
Use Current Attempt Objective Information	ルールの評価において，現在のアテンプトの学習目標進捗情報を用いる．
Use Current Attempt Progress Information	ルールの評価において，現在のアテンプトのアテンプト進捗情報を用いる．

3.4.1.1 Choice と Flow

Choice シーケンシング制御モードは，移動するアクティビティを目次から学習者に自由に選らばさせるために使用する．親アクティビティの Choice シーケンシング制御モードが True の場合，学習者は Choice シーケンシング要求によって，クラスタ中のいずれかの子アクティビティに移動することができる．False の場合，Choice シーケンシング要求によって子アクティビティに移動することはできない．

Flow シーケンシング制御モードは，提示するアクティビティを Continue および Previous シーケンシング要求によって決定するために使用する．親アクティビティの Flow シーケンシング制御モードが True の場合，学習者は Continue および Previous シーケンシング要求によって，クラスタ中の子アクティビティ間を移動することができる．False の場合，Continue および Previous シーケンシング要求によってクラスタ中を移動することはできない．

3.4.1.2 Choice Exit

Choice Exit シーケンシング制御モードは，自身および配下のアクティビティからの Choice シーケンシング要求による移動を限定するために使用する．Choice Exit シーケンシング制御モー

ドが False のアクティビティおよびその配下のアクティビティから，Choice シーケンシング要求によって他のアクティビティに移動することはできない．Choice シーケンシング要求が有効になるためには，現在のアクティビティおよび祖先のアクティビティの Choice Exit シーケンシング制御モードがすべて True でなくてはならない．これによって，Choice シーケンシング要求を用いて，あるアクティビティの配下から配下外に移動することを禁止することができる．

3.4.1.3 Forward Only

Forward Only シーケンシング制御モードは，クラスタ中のアクティビティ間の移動方向を前方のみに限定し，後方への移動を禁止するために使用する．親アクティビティの Forward Only シーケンシング制御モードが True の場合，そのクラスタ中では，学習者は Previous シーケンシング要求および後方への Choice シーケンシング要求を使用できなくなる．False の場合は，前方にも後方にも移動することができる．

3.4.1.4 Use Current Attempt Objective Information と Use Current Attempt Progress Information

ルールの評価を行う際に，学習目標進捗情報およびアテンプト進捗情報として，クラスタにおける現在のアテンプトの情報だけを使うか，前回のアテンプトを含めた最新の情報を使うかを制御する．親アクティビティの Use Current Attempt Objective (Progress) Information が True の場合は現在のアテンプトの情報だけを使う．クラスタの現在のアテンプトで，まだ実行されていない子アクティビティの学習目標進捗情報およびアテンプト進捗情報は未知とみなされる．親アクティビティの Use Current Attempt Objective (Progress) Information が False の場合は前回のアテンプトを含めた最新の情報を使う．クラスタの現在のアテンプトで，まだ実行されていない子アクティビティについては，前回までのアテンプトの最新の学習目標進捗情報およびアテンプト進捗情報を使う．

この状況を図 3.2 に示す．図 3.2 で，a)では親アクティビティ 1.の Use Current Attempt Objective (Progress) Information が True に設定されており，b)では False に設定されている．

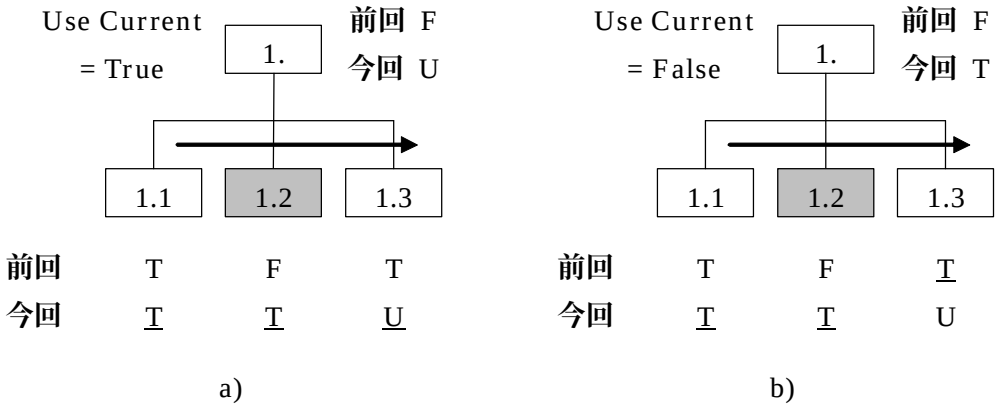

図 3.2 Use Current Attempt Objective / Progress Information

どちらの場合も，前回のアテンプトでの子アクティビティの学習目標習得（アテンプト完了）状

態は，1.1 が True，1.2 が False，1.3 が True で，1.は 3.4.5 に述べるデフォルトロールアップルールにより，1.1, 1.2, 1.3 の And で False になっている．今回のアテンプトでは，子アクティビティ 1.1 および 1.2 の学習が終了し，1.1 が True，1.2 が True になっている．

この状態で，親アクティビティ 1.の状態はどのような値を取るべきであろうか? 今回のアテンプトの状態値を基にする，と考えれば，今回のアテンプトではアクティビティ 1.3 がまだ実行されていないので，a)のように，親アクティビティ 1.の状態，すなわち，三つの子アクティビティの And は未定(Unknown)とするべきであろう．一方，過去の値も含めて考えるのであれば，今回だけでなく前回までのアテンプトも含め最新の状態値を用いることになる．その場合，アクティビティ 1.3 の最新の状態は前回のアテンプトにおける True なので, b)のように親アクティビティ 1.の状態は True となる．コンテンツ作成者は，どちらを用いるか，クラスタごとに決めることができる．

3.4.2 制限条件

制限条件によって，アクティビティの提示を禁止する条件を設定することができる．現在の規格ではアクティビティの実行回数（アテンプトの回数）を制限することができる．アクティビティに実行回数制限が設定されていると，制限回数以上のアテンプトの実行は禁止される．

3.4.3 プリコンディションルール

プリコンディションルールによって，アクティビティの提示を制限する条件を設定することができる．アクティビティの提示を制限する，という意味で，プリコンディションルールは制限条件と類似している．

プリコンディションルールは，各アクティビティに対して記述する．あるアクティビティに対して複数のプリコンディションルールを記述することができる．プリコンディションルールは

If [条件セット] Then [アクション]

の形をしている．条件セットは，アクティビティのトラッキング情報の値によって，真か偽になるような評価式である．アクションはアクティビティの提示を制限する内容である．プリコンディションルールの例を以下に示す．

If Satisfied Then Skip
　　アクティビティが習得済みなら，そのアクティビティを飛び越す
If Attempted Then Disabled
　　アクティビティが実行されていれば，提示を行わない
If Always Then Hidden from Choice
　　常に，Choice ナビゲーション要求の対象としない

3.4.3.1 プリコンディションルールの条件セット

プリコンディションルールの条件セットは以下の形式を取る．

条件結合子((条件演算子，条件要素),…)

すなわち，条件結合子で条件演算子と条件要素の対を一つ以上結びつけたものが条件セットである．条件結合子，条件演算子，条件要素はそれぞれ以下のような内容である．

- 条件結合子： All と Any の二種の結合子がある．All 結合子は後続するすべての条件要素が真の場合に真となる．Any 結合子は後続するいずれかの条件要素が真の場合に真となる．省略した場合，Any とみなされる．
- 条件演算子： NO-OP と Not の二種の演算子がある．NO-OP 演算子は対となる条件要素の真偽値を変えない．Not 演算子は条件要素の真偽値を否定する．
- 条件要素：アクティビティのトラッキング情報の値によって真か偽となる．条件要素を表 3.7 に示す．対象とするトラッキング情報が学習目標習得状態，学習目標習得度である場合，ルール条件参照学習目標 (Rule Condition Referenced Objective) で対象とする学習目標を指定する．また，学習目標習得度に対してはルール条件習得度しきい値 (Rule Condition Measure Threshold) で比較対象とするしきい値を指定する．

表 3.7 ルールの条件要素

条件要素	対象トラッキング情報	説明
Satisfied	学習目標習得状態	対象とする学習目標習得状態が習得の場合，真となる
Objective Status Known	学習目標習得状態	対象とする学習目標習得状態が未定でない場合，真となる
Objective Measure Known	学習目標習得度	対象とする学習目標習得度が未定でない場合，真となる
Objective Measure Greater Than	学習目標習得度	対象とする学習目標習得度がしきい値より大きい場合，真となる
Objective Measure Less Than	学習目標習得度	対象とする学習目標習得度がしきい値より小さい場合，真となる
Completed	アテンプト完了状態	アテンプト完了状態が完了の場合，真となる
Activity Progress Known	アテンプト完了状態	アテンプト完了状態が未定でない場合，真となる
Attempted	アクティビティ試行回数	アクティビティ試行回数が 1 以上の場合，真となる
Attempt Limit Exceeded	アクティビティ試行回数	アクティビティ試行回数が制限条件で定めた回数以上の場合，真となる
Always	なし	常に真となる

3.4.3.2 プリコンディションルールのアクション

プリコンディションルールのアクションは表 3.8 のいずれかである．これらのアクションは図 3.1 のシーケンシングの過程で，次に配信するアクティビティを決定する際に適用される．

表 3.8 プリコンディションルールのアクション

アクション	説明
Skip	Continue および Previous シーケンシング要求などによって，アクティビティツリーの中を移動して，提示するアクティビティを決定する際に用いられる．条件が真の場合，そのアクティビティを飛び越し，移動方向の次のアクティビティが配信可能かどうかをチェックする．
Disabled	アクティビティの提示を禁止する場合に用いる．条件が真の場合，アクティビティを選択しても，そのアクティビティは提示されない．
Hidden from Choice	Choice シーケンシング要求によるアクティビティの提示を禁止する場合に用いる．条件が真の場合，Choice シーケンシング要求によってアクティビティを選択しても，そのアクティビティは提示されない．
Stop Forward Traversal	アクティビティツリーの中を前方に移動して，提示するアクティビティを決定する際に用いる．条件が真の場合，そのアクティビティで移動は停止する．そのアクティビティは提示の対象とはならない．

3.4.4 ポストコンディションルール ／ 終了ルール

ポストコンディションルールおよび終了ルールによって，学習者が入力したナビゲーション要求を無視して，コンテンツ作成者が指定したシーケンシング要求や終了要求を発生することができる．

これらのルールは，各アクティビティに対して記述する．あるアクティビティに対して複数のルールを記述することができる．ポストコンディションルールおよび終了ルールは，プリコンディションルールと同様，

If [条件セット] Then [アクション]

の形をしている．条件セットは，プリコンディションルールと同様，アクティビティのトラッキング情報の値によって，真か偽になるような評価式である．アクションはシーケンシング要求および終了要求である．ポストコンディションルールの例を以下に示す．

If Not Satisfied Then Retry

アクティビティが未習得なら，そのアクティビティを再実行する

If All (Attempted, Satisfied) Then Exit All

アクティビティが実行されていて，習得済みならば，終了する

3.4.4.1 ポストコンディションルール ／ 終了ルールの条件セット

ポストコンディションルールおよび終了ルールの条件セットは，プリコンディションルールの条件セットと同様である．

3.4.4.2 ポストコンディションルール ／ 終了ルールのアクション

ポストコンディションルールのアクションは表 3.9 のいずれかである．これらのアクションは図 3.1 の終了＆ロールアップの過程で，学習者からのナビゲーション要求を置き換えて，新たなシーケンシング要求ないし終了要求を発生するために適用される．

表 3.9 ポストコンディションルール ／ 終了ルールのアクション

アクション	説明	シーケンシング要求	終了要求
Exit Parent	アクティビティの親アクティビティを終了する．		Exit Parent
Exit All	教材全体を終了する		Exit All
Exit	アクティビティを終了する		Exit
Retry	アクティビティを再実行する．もし，アクティビティが末端のアクティビティでない場合は，クラスタの子アクティビティの最初のものから実行を試みる	Retry	
Retry All	教材全体を終了して再実行する	Retry	Exit All
Continue	前方に進む	Continue	
Previous	後方に進む	Previous	

3.4.5 ロールアップルール

ロールアップによって，アクティビティツリーの各アクティビティのトラッキング情報は，SCO からツリーの根に向かって順次更新される．ロールアップにおいて子アクティビティのトラッキング情報から親アクティビティのトラッキング情報を決定するルールがロールアップルールである．

ロールアップルールには，学習目標習得度に関するもの，学習目標習得状態に関するもの，アテンプト完了状態に関するものがある．このときのトラッキング情報の関係を図 3.3 に示す．

習得度ロールアップでは，親アクティビティの主学習目標の習得度を子アクティビティの主学習目標の習得度から決定する．主学習目標は，3.1 に述べたようにアクティビティにひとつだけ存在する．

学習目標ロールアップでは，親アクティビティの主学習目標の習得状態を，習得度ロールアッ

プによって決定される自身の習得度，ないし，子アクティビティの主学習目標の習得状態，アテンプト完了状態，アクティビティ試行回数から決定する．

進捗状態ロールアップでは，親アクティビティのアテンプト完了状態を，子アクティビティの主学習目標の習得状態，アテンプト完了状態，アクティビティ試行回数から決定される．

以下，おのおのについて説明する．

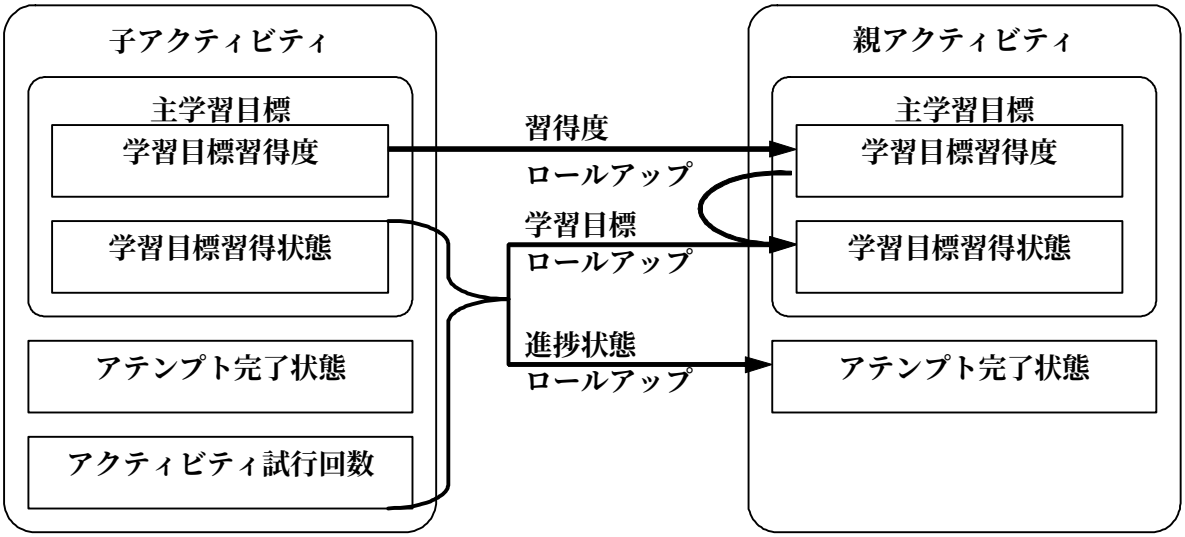

図 3.3 ロールアップにおけるトラッキング情報の関係

3.4.5.1 習得度ロールアップ

習得度ロールアップでは，子アクティビティの主学習目標の習得度の重み付き平均で，親アクティビティの主学習目標の習得度を決定する．学習目標習得度の重み付けは，Rollup Objective Measure Weight によってコンテンツ作成者が指定する．計算式を以下に示す．

$$\text{親アクティビティの学習目標習得度} = \frac{\sum_{\text{子アクティビティ}} (RollupObjectiveMeasureWeight \times \text{学習目標習得度})}{\sum_{\text{子アクティビティ}} RollupObjectiveMeasureWeight}$$

子アクティビティの主学習目標の習得度が未定の場合，その学習目標習得度を 0 として計算する．

3.4.5.2 学習目標ロールアップ

学習目標ロールアップでは，以下の手順で親アクティビティの主学習目標の習得状態を決定する．

(1) 習得度による決定．

親アクティビティの Objective Satisfied by Measure が真の場合，習得度ロールアップで

計算された親アクティビティの主学習目標の習得度と，親アクティビティの Objective Minimum Satisfied Normalized Measure を比較して，主学習目標の習得状態を決定する．学習目標習得度が Objective Minimum Satisfied Normalized Measure 以上なら学習目標習得状態を習得に設定し，そうでなければ未習得に設定する．学習目標ロールアップはここで終了する．

親アクティビティの Objective Satisfied by Measure が偽の場合，習得度による決定は行わず，(2)のロールアップルールによる決定に進む．

(2) ロールアップルールによる決定．

親アクティビティに，アクションが Satisfied か Not Satisfied のロールアップルールが設定されていれば，Not Satisfied ルールを先に評価し，次に Satisfied ルールを評価して，主学習目標の習得状態を設定する．つまり，Not Satisfied ルールの結果は Satisfied ルールの結果によって上書きされる場合がある．学習目標ロールアップはここで終了する．ルールの詳細についてはあとで述べる．

親アクティビティに，アクションが Satisfied か Not Satisfied のロールアップルールが設定されていなければ，(3)のデフォルトルールによる決定に進む．

(3) デフォルトルールによる決定．以下のデフォルトルールを(2)と同様の手順で実行する．

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

If all satisfied, Then satisfied

すなわち，

子アクティビティすべてがアテンプト済みか未修得なら，親アクティビティは未習得

子アクティビティすべてが習得なら，親アクティビティは習得

3.4.5.3 進捗状態ロールアップ

進捗状態ロールアップでは，以下の手順で親アクティビティのアテンプト完了状態を決定する．

(1) ロールアップルールによる決定．

親アクティビティに，アクションが Completed か Incomplete のロールアップルールが設定されていれば，Incomplete ルールを先に評価し，次に Completed ルールを評価して，アテンプト完了状態を設定する．つまり，Incomplete ルールの結果は Complete ルールの結果によって上書きされる場合がある．進捗状態ロールアップはここで終了する．ルールの詳細についてはあとで述べる．

親アクティビティに，アクションが Completed か Incomplete のロールアップルールが設定されていなければ，(2)のデフォルトルールによる決定に進む．

(2) デフォルトルールによる決定．以下のデフォルトルールを(1)と同様の手順で実行する．

If all (attempted or incomplete), Then incomplete

If all completed, Then completed

すなわち，

子アクティビティすべてがアテンプト済みか未完了なら，親アクティビティは未完了

子アクティビティすべてが完了なら，親アクティビティは完了

3.4.5.4 ロールアップルールの詳細

学習目標ロールアップにおける Satisfied/Not Satisfied のロールアップルール，および，進捗状態ロールアップにおける Completed/Incomplete のロールアップルールは，いずれも

If [条件セット] For [子アクティビティセット] Then [アクション]

という形式で記述される．条件セットは，子アクティビティのトラッキング情報の値によって，真か偽になるような評価式である．子アクティビティセットは，条件セットを個々の子アクティビティに適用し，その結果を集約して最終的に条件全体が真になるか偽になるかを決める際の集約の方法を指定する．アクションは親アクティビティのトラッキング情報の値を決める動作である．ロールアップルールの例を以下に示す．

If not satisfied For any Then not satisfied
　子アクティビティのいずれかが習得でないなら，親アクティビティは未習得
If satisfied For at least 3 Then satisfied
　子アクティビティのいずれか 3 つ以上が習得なら，親アクティビティは習得
If satisfied or completed For all Then completed
　子アクティビティのすべてが習得ないし完了なら，親アクティビティは完了
If satisfied and attempted For all Then satisfied
　子アクティビティのすべてが習得かつアテンプト済みなら，親アクティビティは習得
If not attempted For at least 50% Then incomplete
　子アクティビティの 50%以上がアテンプト済みでなければ，親アクティビティは未完了

► ロールアップルールの条件セット

ロールアップルールの条件セットは以下の形式を取る．

条件結合子((条件演算子，条件要素),…)

これは，プリコンディションルールで述べた形式と同様である（3.4.3 参照）．条件結合子（All, Any），条件演算子（Not, NO-OP）の種別も同様である．

条件要素については，プリコンディションルール，ポストコンディションルールとは異なる．表 3.10 にロールアップルールの条件要素を示す．プリコンディションルール，ポストコンディションルールと異なるのは，学習目標習得度の大小比較を行う条件要素が無いこと，対象とする学習目標が主学習目標に限られるため学習目標の指定が無いこと，である．

► ロールアップルールの子アクティビティセット

子アクティビティセットは，条件セットを個々の子アクティビティに適用した結果から，最終的な条件の真偽を決定する方法を指定する．例えば，条件セットを個々の子アクティビティに適用した結果，80%以上の子アクティビティが条件セットを満たせば，最終的な結果を真とする，

といった指定が可能である．子アクティビティセットの指定を表 3.11 に示す．

表 3.10 ロールアップルールの条件要素

条件要素	対象トラッキング情報	説明
Satisfied	学習目標習得状態	主学習目標の学習目標習得状態が習得の場合，真となる
Objective Status Known	学習目標習得状態	主学習目標の学習目標習得状態が未定でない場合真，となる
Objective Measure Known	学習目標習得度	主学習目標の学習目標習得度が未定でない場合，真となる
Completed	アテンプト完了状態	アテンプト完了状態が完了の場合, 真となる
Activity Progress Known	アテンプト完了状態	アテンプト完了状態が未定でない場合, 真となる
Attempted	アクティビティ試行回数	アクティビティ試行回数が 1 以上の場合，真となる
Attempt Limit Exceeded	アクティビティ試行回数	アクティビティ試行回数が制限条件で定めた回数以上の場合，真となる
Never	なし	常に偽となる

表 3.11 子アクティビティセット

名称	説明
All	条件セットを適用した結果，すべての子アクティビティの結果が真の場合，真となる
Any	条件セットを適用した結果，ある子アクティビティの結果が真の場合，真となる
None	条件セットを適用した結果，いずれの子アクティビティの結果も真でない場合，真となる
At Least Count	条件セットを適用した結果，一定数を越える子アクティビティの結果が真の場合，真となる
At Least Percent	条件セットを適用した結果，一定の割合を越える子アクティビティの結果が真の場合，真となる

次にロールアップの対象となる子アクティビティの指定について述べる．基本的にはクラスタ中のすべての子アクティビティがロールアップの対象となるが，以下のようなアクティビティはロールアップの対象とならず，上記の子アクティビティセットの評価に含まれない．例えば，At Least Count 子アクティビティセットの評価の際には，以下に該当するアクティビティを除いた子アクティビティの集合について，条件セットが成り立つ子アクティビティの割合を算出する．

- Tracked が False のアクティビティ．このようなアクティビティではトラッキング情報は記録されないので，ロールアップの対象としない．
- Rollup Objective Satisfied が False のアクティビティ．このようなアクティビティは，アクションが Satisfied または Not Satisfied のロールアップルールの対象とならない．
- Rollup Progress Completion が False のアクティビティ．このようなアクティビティは，アクションが Completed または Incomplete のロールアップルールの対象とならない．
- 各種の Required For 要素．これらの要素で指定される条件が成立しないとき，アクティビティはロールアップの対象とならない．Required For 要素を表 3.12 に示す．

表 3.12 Required For 要素

名称	説明	語彙（各要素に共通）
Required for Satisfied	どのような場合に，アクションが Satisfied のロールアップルールの対象とするかを示す．	・ Always － 常に対象とする ・ ifNotSuspended － Suspend で無い場合対象とする ・ ifAttempted － アテンプト済みの場合対象とする ・ ifNotSkipped － Skip されていない場合対象とする
Required for Not Satisfied	どのような場合に，アクションが Not Satisfied のロールアップルールの対象とするかを示す．	
Required for Completed	どのような場合に，アクションが Completed のロールアップルールの対象とするかを示す．	
Required for Incomplete	どのような場合に，アクションが Incomplete のロールアップルールの対象とするかを示す．	

► ロールアップルールのアクション

ロールアップルールのアクションは，Satisfied, Not Satsfied, Completed, Incomplete のいずれかである．表 3.13 にロールアップルールのアクションを示す．

表 3.13 ロールアップルールのアクション

アクション	説明
Satisfied	親アクティビティのロールアップ学習目標の習得状態を習得にする
Not Satisfied	親アクティビティのロールアップ学習目標の習得状態を未習得にする
Completed	親アクティビティのアテンプト完了状態を完了にする
Incomplete	親アクティビティのアテンプト完了状態を未完了にする

3.4.6 ローカル学習目標と共有グローバル学習目標

アクティビティはデフォルトで一つのローカル学習目標を有する．コンテンツ作成者は，さらに任意の数のローカル学習目標をアクティビティに関連付けることができる．

各ローカル学習目標は，共有グローバル学習目標に関連付けることができる．これによって，アクティビティ間でトラッキング情報を共有してシーケンシングを行うことが可能となる．例えば，プリテストアクティビティと解説アクティビティで学習目標を共有し，プリテストの結果によって解説アクティビティを提示するかしないかを切り替えることが可能となる．

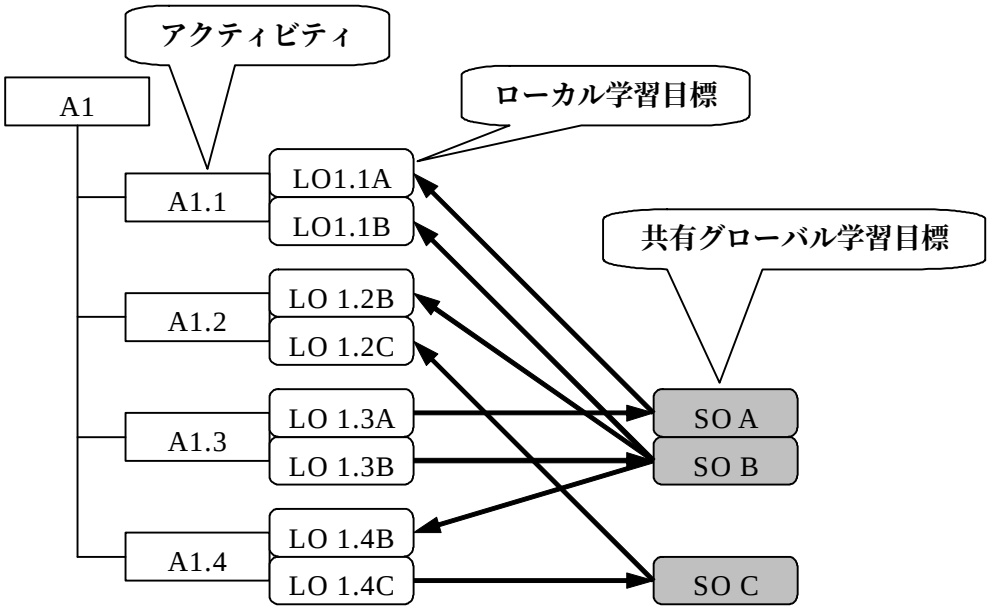

図 3.4 アクティビティ, ローカル学習目標, 共有グローバル学習目標の関係

アクティビティ，ローカル学習目標，共有グローバル学習目標の関係を図 3.4 に示す．これらの参照関係には以下のような制限がある．

- ひとつのアクティビティは，複数のローカル学習目標と関連付けることができる．例）A1.1 と LO1.1A, LO1.2B の関係.
- ひとつのローカル学習目標は，ひとつの共有グローバル学習目標とだけ関連付けることができる．複数の共有グローバル学習目標に関連付けることはできない．例）LO1.1A と SO A の関係.
- ひとつの共有グローバル学習目標は複数のローカル学習目標に関連付けることができる．例）SO B と LO1.1B, LO1.2B, LO1.4B の関係.
- 以上から，ひとつのアクティビティは複数のローカル学習目標を経由して，複数の共有グローバル学習目標を参照することができる．例）A1.1 と SO A, SO B の関係.
- 逆に，ひとつの共有グローバル学習目標は複数のローカル学習目標を経由して，複数のアクティビティから参照される．例）SO B と A1.1, A1.2, A1.4 の関係.

ローカル学習目標と共有グローバル学習目標を関連付ける際に，習得度および習得状態をどちらからどちらに転送するかを指定する．すなわち，ローカル学習目標から共有グローバル学習目

標にデータを書き込むか，共有グローバル学習目標からデータを読み出すか，を指定する．これについては以下のような制限がある．

- あるひとつのアクティビティから共有グローバル学習目標に書き込むことができるのはひとつのローカル学習目標だけである．複数のローカル学習目標から書き込むことはできない．
  複数のアクティビティのそれぞれひとつのローカル学習目標から書き込むことは可能である．
  例）SO B と LO1.1B, LO1.2B, LO1.4B の関係．

3.4.7 共有グローバル学習目標とルールの評価

3.4.7.1 共有グローバル学習目標とプリコンディションルール，ポストコンディションルール，終了ルール

3.4.3 プリコンディションルール, 3.4.4 ポストコンディションルール／終了ルールで述べたように，これらのルールの条件部では，当該のアクティビティに関連付けられたローカル学習目標を参照することができる．このとき，ローカル学習目標が共有グローバル学習目標に関連付けられていて，共有グローバル学習目標からローカル学習目標にデータを読み出すように設定されていれば，ルールの評価には共有グローバル学習目標の値が使われる．

3.4.7.2 共有グローバル学習目標とロールアップルール

3.4.5 ロールアップルールで述べたように，ロールアップの際に使用される学習目標は主学習目標だけである．主学習目標が共有グローバル学習目標に関連付けられている場合は，ロールアップによって共有グローバル学習目標の値も変化する．

ロールアップと共有グローバル学習目標の関係を図 3.5 に示す．ロールアップは，SCO の値の変化によってトラッキング情報が変化した末端アクティビティ，および，そのアクティビティから値を書き込まれる共有グローバル学習目標を読み出しているアクティビティを基点として実行される．これらのロールアップの基点となるアクティビティの集合をロールアップセットと呼ぶ．図 3.5 でアクティビティ A1.1.1 のトラッキング情報が変化したとすると，これが反映される SO B

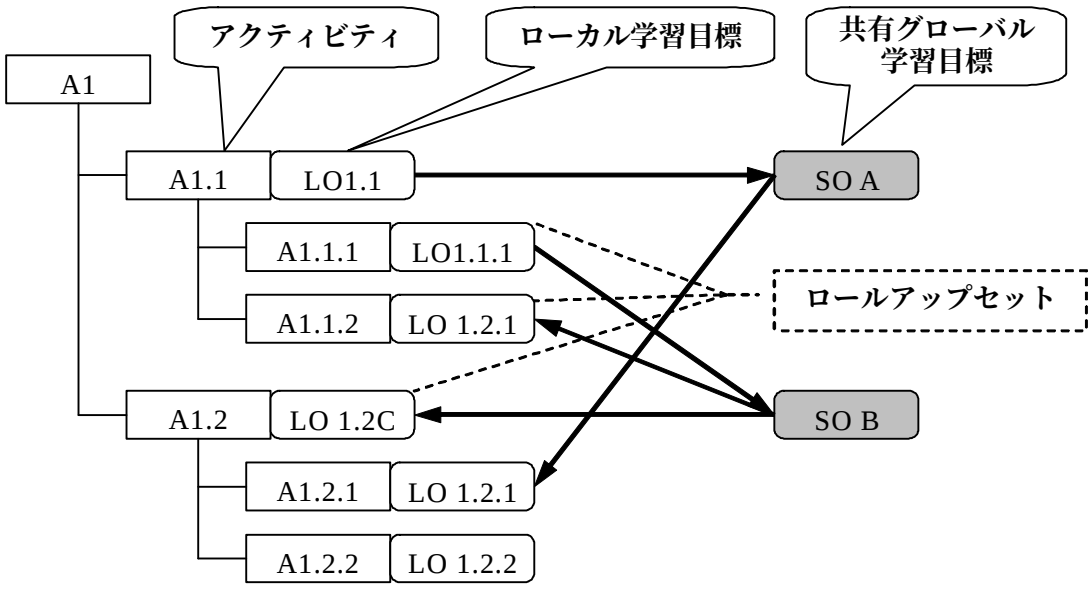

図 3.5 共有グローバル学習目標とロールアップの関係

を読み出す A1.1.2, A1.2 を加えた三つのアクティビティがロールアップセットになる．

ロールアップセット中のあるアクティビティからロールアップを開始して，ロールアップセットの他のアクティビティにロールアップが行われた場合，2 番目のアクティビティをロールアップセットから取り除く．また，ロールアップの途中で，共有グローバル学習目標に書き込みを行っている主学習目標の値が変化した場合は，共有グローバル学習目標の値は主学習目標の値に更新されるが，その共有グローバル学習目標を読み出しているアクティビティから別のロールアップが発生することは無い．図 3.5 の場合，A.1.1.1 からのロールアップで A1.1 のトラッキング情報が変化し，共有グローバル学習目標 SO A が変化するが，それが A1.2.1 に波及することはない．

3.5 アテンプト

あるアクティビティが学習者に配信されて学習が開始されてから，そのアクティビティの学習を終了して他のアクティビティが配信のために選択されるまでの間をアテンプトと呼ぶ．いったんアテンプトを終了したアクティビティが再度選択されて配信された場合は，別のアテンプトとして扱われる．

アテンプトはアクティビティ階層の親子関係の中で規定される．すなわち，図 3.6 で A1.1.2 が学習中だとすると，A1.1.2, A1.1, A1 がアテンプト中ということになる．このとき，A1.1.2 を終了して A1.1.1 を選択した場合，A1.1.2 のアテンプトは終了するが A1.1, A1 のアテンプトは継続する．

学習者が Suspend All ナビゲーション要求を発行して学習を一旦中断し，あとで Resume All ナビゲーション要求によって学習を再開した場合は，前回のアテンプトが継続するとみなされる．すなわち，学習再開の場合は，同一のアクティビティを前回中断した状態から継続して学習するため，新しいアテンプトが始まるのでなく，中断時のアテンプトが継続するものとする．

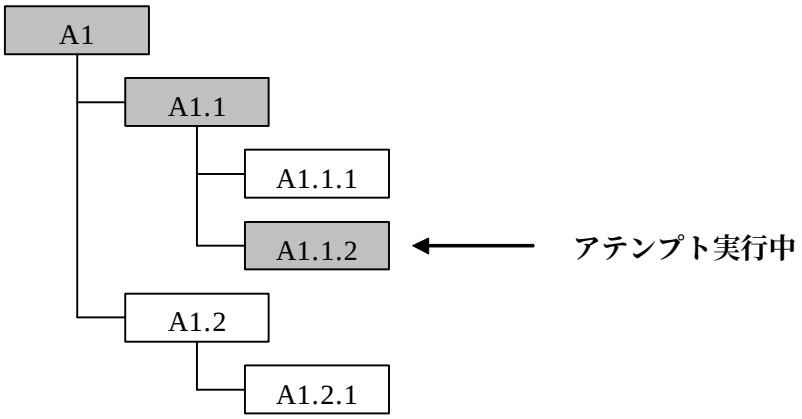

図 3.6 アテンプト

4. ナビゲーション

本節では，SCORM 2004 の新たに追加された機能であるナビゲーション GUI 機能の基本となる概念とそれらの全体的な関連について説明し，ナビゲーションにおける動作がどのように規定されるかを説明する．

4.1 ナビゲーションコントロール概要

4.1.1 SCORM 1.2 における SCO ナビゲーション

SCORM 1.2 では SCO のナビゲーション制御は LMS が行うよう規定されていた．例えば，ある SCO を表示したり，ある SCO から他の SCO に移動するためには，LMS が提供するインタフェースで制御しなければならない．逆に言うと，コンテンツ側では SCO 間のナビゲーションについては一切関与してはならず，SCO 側から他の SCO へ移動するためのナビゲーションボタンを提供してはならない．

また，SCORM 1.2 規格では，LMS が SCO をどのようにナビゲーションするかということについて定義されていないため，ボタンや目次の有無や表示位置，キャプション，ナビゲーションの仕方などのインタフェースは LMS によってまちまちであった．

そのため，複数の LMS で動作させることを念頭においたコンテンツを作成しようとする場合，コンテンツ作成者が，意図する画面設計をしたり，学習者に一貫した操作を提供したいと考えても，SCORM 1.2 規格でコンテンツを作成する限り，ナビゲーションの設計には制限があった．

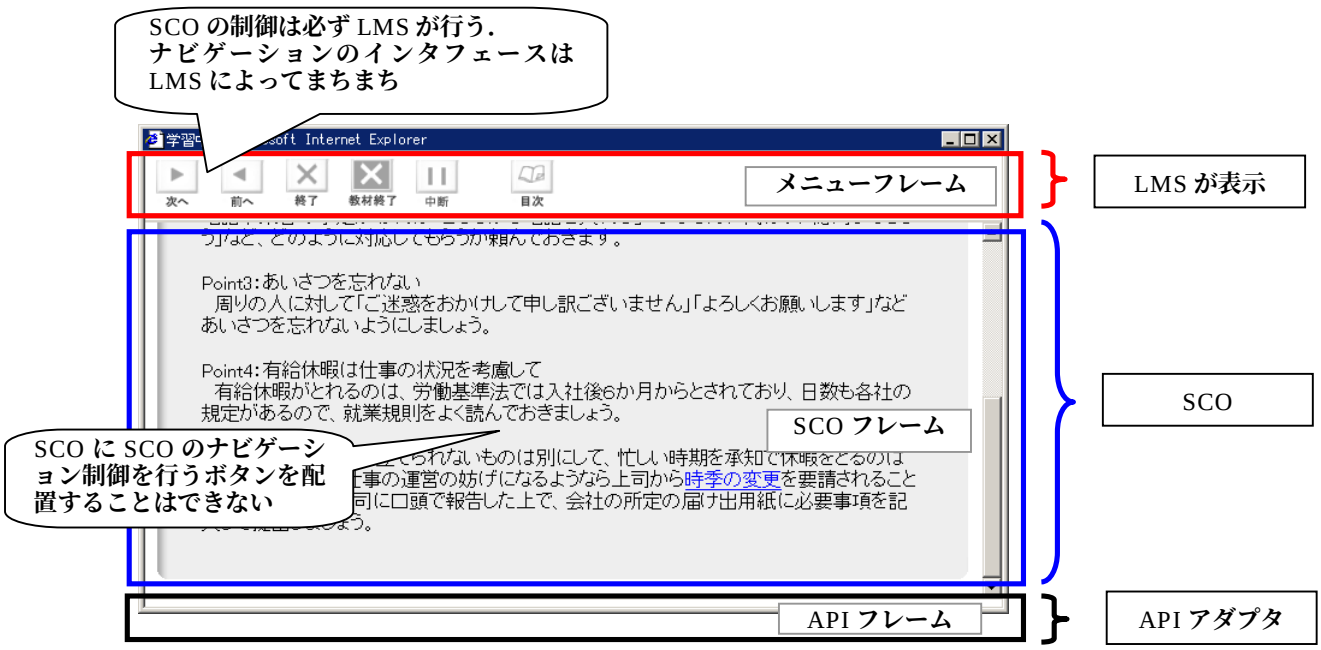

図 4.1 SCORM 1.2 ナビゲーション概観

4.1.2 SCORM 2004 における SCO ナビゲーション

SCORM 2004 では，SCO のナビゲーション方法についての規格が新たに追加され，コンテンツから SCO ナビゲーションの操作ができるようになった．具体的には，SCO から SCO ナビゲーションコマンドの発行ができるようになった．さらに，コンテンツから LMS のナビゲーションボタンの表示／非表示についても制御ができるようになった．

これにより，コンテンツ作成者は LMS の種類を気にすることなく，標準規格でコンテンツの学習コンテンツ作成の重要な要件であるナビゲーションの設計を行うことができる．

なお，SCO 内部のナビゲーション（アセットの制御）については，SCORM 2004 も SCORM 1.2 と同様，LMS は一切関与せず，コンテンツ側ですべてを制御しなければならない．

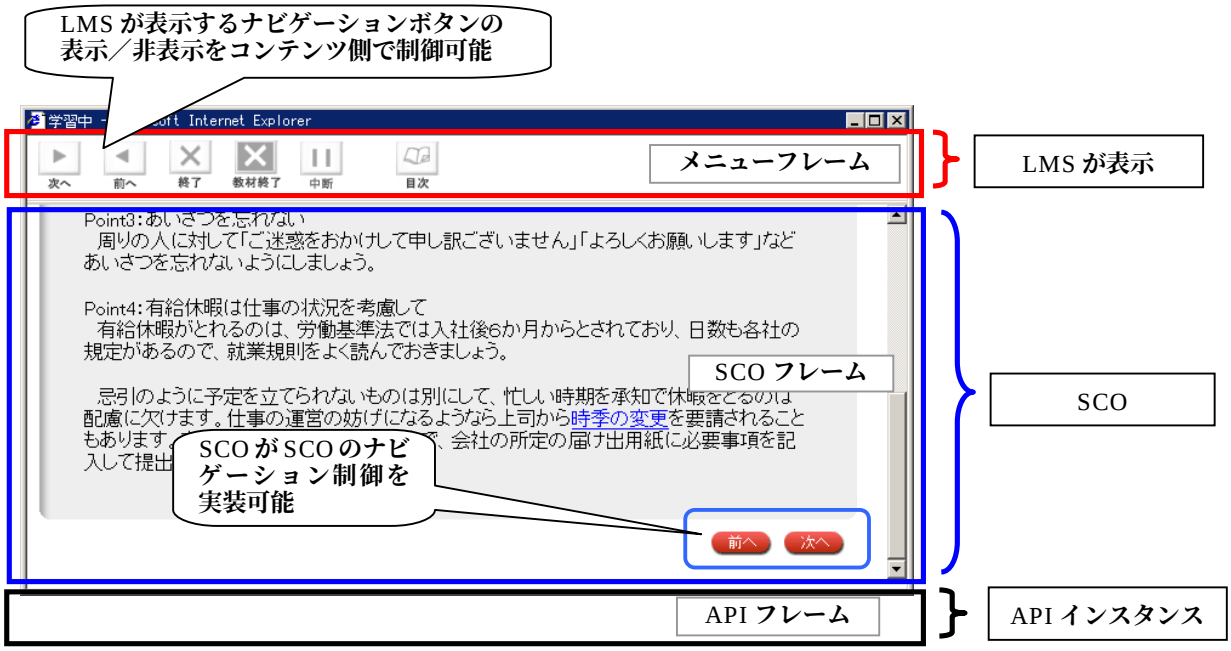

図 4.2 SCORM 2004 ナビゲーション概観

4.2 ナビゲーションコマンドの送信と SCO の終了

4.2.1 SCO での SCO ナビゲーションコマンド

SCORM 2004 では LMS による SCO ナビゲーションに加え，「次へ進む」「前へ戻る」といったナビゲーションコマンドを SCO からも発行することができるようになった．SCO から発行されたナビゲーション要求は LMS で発行したものと同様に処理される．

SCO で制御できるナビゲーションコマンドは次のとおりである．

表 4.1 SCO で発行できるナビゲーションコマンド一覧

コマンド名	説明
continue	現在の SCO を終了し，前方へのナビゲーション要求を発行する
previous	現在の SCO を終了し，後方へのナビゲーション要求を発行する
choice	現在の SCO を終了し，指定したアクティビティのナビゲーション要求を発行する
exit	現在の SCO を終了する
exitAll	教材を終了する
abandon	現在の SCO を放棄する
abandonAll	教材を放棄する
none	未処理のナビゲーション要求をクリアする

SCO が発行するナビゲーションイベントは，LMS が提供する SCO ナビゲーション制御と同様，シーケンシング制御モードに関連して有効化され発生する．学習アクティビティの親クラスタによって定義される制御モードは子クラスタに適用可能なナビゲーションイベントを定義する．

例えば choice 制御モードが有効なら，choice ナビゲーションイベントがクラスタの子アクティビティに適用可能となる．同様に flow 制御モードが有効であれば，continue および previous ナビゲーションイベントがクラスタの子アクティビティに対して実行可能となる．

4.2.2 ナビゲーション要求の発行と SCO の終了

SCO からナビゲーション要求を行うには，SCORM 2004 で追加された新しいランタイムデータモデル adl.nav.request を使用する．ナビゲーション要求の発行は，下記のようにデータモデル adl.nav.request に，continue や previous，choice，exit といった実行させたいナビゲーション要求をセットすることで行われる．

```
SetValue("adl.nav.request",<REQUEST>)
  ※<REQUEST> = continue,previous,choice,exit,exitAll,abandon,abandonAll
```

choice コマンドを使用する場合は，起動するアクティビティを指定する必要があり，次のように記述する．

```
SetValue("adl.nav.request","{target=<STRING>}choice")
  ※<STRING> = アクティビティの item identifier
```

SCO が Terminate API 関数を呼び出すと，LMS はそれまでに SCO が発行したナビゲーション要求イベントの処理を実行する．つまり，SCO からナビゲーション要求イベントが設定されても，その要求イベントは直ちに実行されず，LMS は Terminate 要求を受け取った時点で，はじめてナビゲーション要求イベントが実行される．

なお，Terminate を呼び出すまで，SCO はナビゲーション要求イベントを何回でも設定することができるが，新しいナビゲーション要求が設定されると以前に設定したナビゲーション要求は棄てられる．つまり，Terminate 処理を実行する際，最後に発行されたナビゲーション要求だけが実行される．

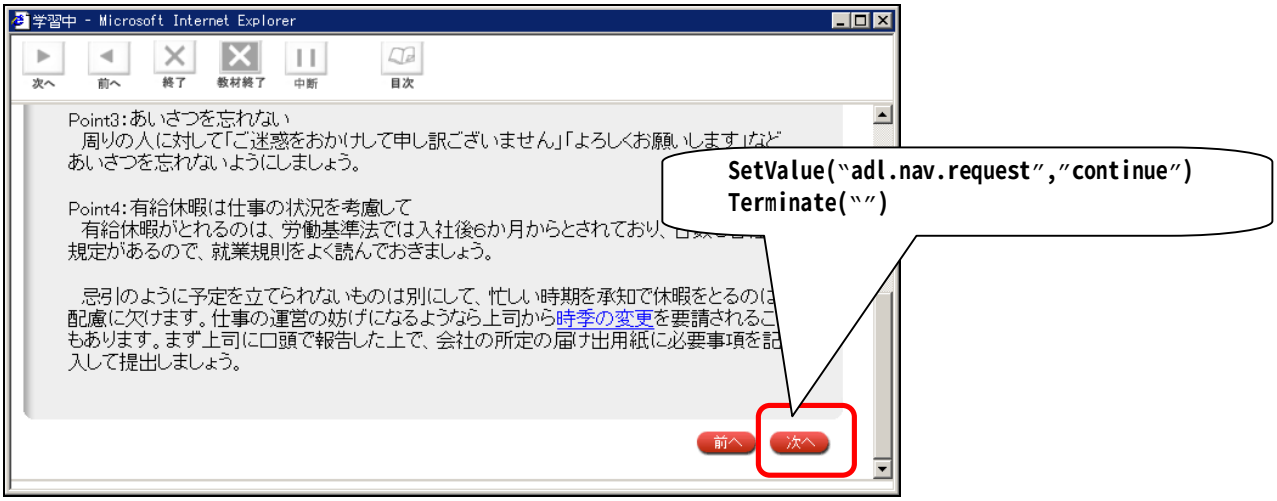

図 4.3 continue コマンドの使用例

4.2.3 ナビゲーション要求の使用可否の確認

ナビゲーション要求が実行可能か否かを確認するにはデータモデル"adl.nav.request_valid.REQUEST"を使用する．このデータモデルは SCO が continue や previous, choice といったナビゲーション要求が実行可能か否かを LMS に問い合わせるもので，LMS はこのコマンドを受け取ったら SCO に対し，REQUEST の実行可否を返却する．実行可能な場合は true，実行不可能な場合は false，不明な場合は unknown が返される．

```
GetValue("adl.nav.request_valid.<REQUEST>")
  ※<REQUEST> = continue,previous,choice
  ※戻り値 = true,false,unknown
```

choice コマンドの実行可否を確認する場合は，ターゲットアクティビティを指定する必要があり，次のように記述する．

```
GetValue(“adl.nav.request_valid.choice.{target=<STRING>}”)
  ※<STRING> = アクティビティの item identifier
```

なお，データモデル adl.nav.request に現在格納されている値は GetValue することで確認することができ，GetValue すると現在格納されている値が返される．値が何もセットされていない場合は，adl.nav.request の初期値である”_none_”が返される．

```
GetValue("adl.nav.request",<REQUEST>)
  ※<REQUEST> = continue,previous,choice,exit,exitAll,abandon,abandonAll
```

4.3 LMS のナビゲーション GUI 制御

SCORM 2004 では LMS のナビゲーションボタンの表示／非表示を指定できるようになった．SCO のナビゲーション要求コマンドと LMS のナビゲーションボタンの表示／非表示を制御することで，コンテンツ作成者はコンテンツのインタフェース設計や教材構成において多様な設計が可能になる．

LMS のナビゲーションボタンの表示／非表示の制御は，continue,previous,exit,abandon コマンドについて行うことができ，マニフェストファイル（imsmanifest.xml）に，hideLMSUI データ要素の指定をアクティビティごとに記述することで実現される．

LMS のナビゲーションボタンの非表示設定のパラメータは以下のとおりである．

表 4.2 LMS ナビゲーションボタンの表示

パラメータ	説明
previous	「前へ戻る」ボタンを非表示にする
continue	「次に進む」ボタンを非表示にする
exit	「終了」ボタンを非表示にする
abandon	「中止」ボタンを非表示にする

下記の例は，item1 アクティビティ実行時，LMS のメニューフレームの continue と previous ボタンを表示しないよう設定するものである．

```
<organization>
 <item identifier="item1" identifierref="Resource1" isvisible="true">
    <adlnav:presentation>
       <adlnav:navigationInterface>
          <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
          <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
       </adlnav:navigationInterface>
    </adlnav:presentation>
 </item>
</organization>
```

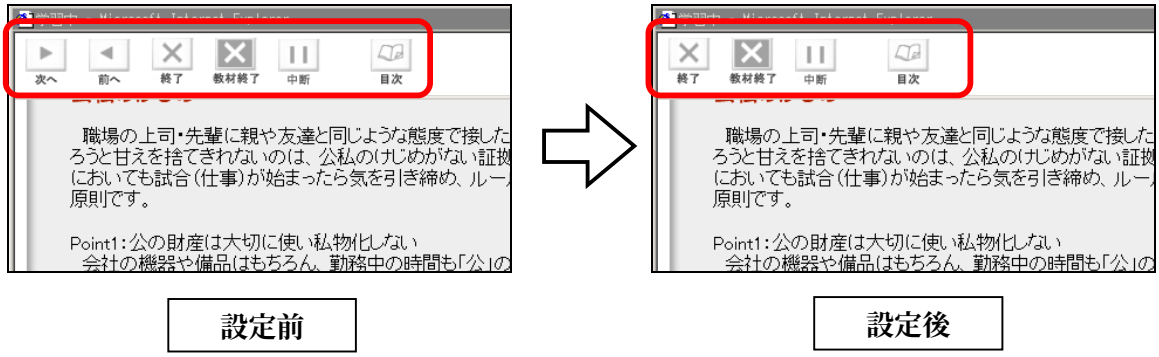

図 4.4 LMS ナビゲーションボタンの表示／非表示

5. RTE

本節では, LMS と SCO の間の実行環境を規定している SCORM ランタイム環境の概念について説明し，特に SCORM 1.2 から SCORM 2004 へのバージョンアップに伴う変更点について解説する.

5.1 SCORM ランタイム環境の概要

SCORM ランタイム環境（Run-Time Environment 以下 RTE）では，学習資源の起動メカニズム，学習資源と LMS とが通信するための共通のメカニズム，および学習資源上での学習者のふるまいをトラッキングする為の共通データモデルを規定している．下図に示すようにランタイム環境では，アプリケーション・インタフェース（API）を通じて，配信された SCO と LMS の間でデータ通信を行う.

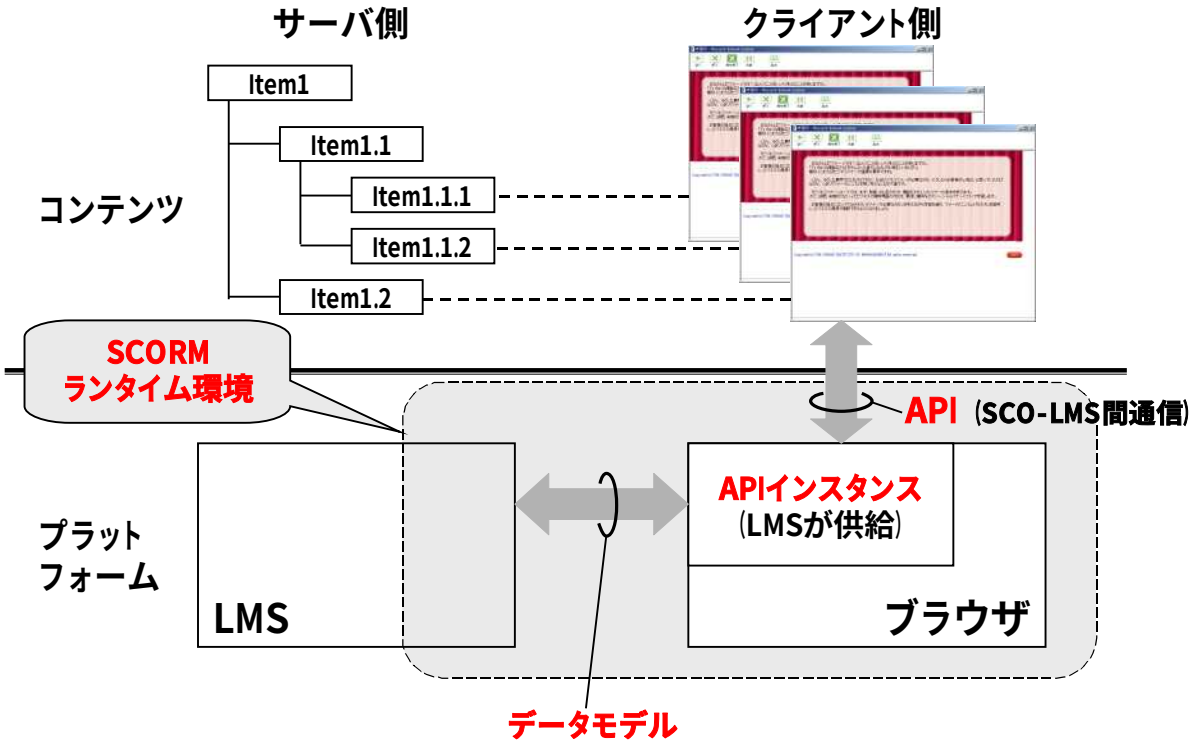

図 5.1 SCORM ランタイム環境

起動プロセスは，LMS が Web ベースの学習資源を起動するための共通な方法を定義する．このメカニズムは，配信された学習資源と LMS との間で通信を確立するための手続きと責任範囲を規定している．この通信メカニズムは，共通の API の使用を通じて標準化される.

API は，LMS に学習資源（SCO）の状態（例えば，初期化，終了，エラー状態）を伝える通信メカニズムであり，LMS と SCO との間でデータを取得したり，設定したりする為に使用される．図 5.1 の API インスタンス（API Instance）は，クライアントサイドで起動される実行プログラムであり，ECMAScript（Java スクリプト）で呼び出し可能なソフトウェア部品である.

データモデルは, SCO の完了状態やクイズやテストのような評価からの得点を記録するための

情報を定義するために使用されるデータ要素の標準セットである．LMS と SCO はお互いに，データモデルにどういう要素が含まれていて，それらがどういう意味を持つのかを予め「知っている」ものとして実装され，やりとりされる．

5.2 学習資源の起動

LMS はナビゲーション要求に基づいて，学習のアクティビティを決定し，配信すべき学習資源を決定する役割をもつ．学習資源を配信する際，LMS は学習資源の起動ロケーションとして定義された URL を使って起動（Launch）する．学習資源の起動には HTTP プロトコルが使われ，起動ロケーションとして指定された学習資源は，最終的にクライアント側ブラウザに表示される．

LMS によって起動される学習資源には，「アセット」と「SCO」の 2 つがある．

5.2.1 アセット

アセットは，テキスト，画像，アンケートなどの Web クライアントに配信可能なメディアで構成される学習資源であり，LMS はこのアセットを起動するだけで，それ自体は LMS と通信する必要がないので，LMS によって供給される API を実装する必要はない．

5.2.2 SCO

SCO（Sharable Content Object）は，複数のアセットの集合体として，SCORM ランタイム環境を利用して LMS と通信を行うものと規定されている．SCO は LMS が動作を記録できる最小単位の学習資源となる．

また，LMS では一度にひとつの SCO だけが起動され，ひとつの SCO だけがアクティブになるよう規定している．

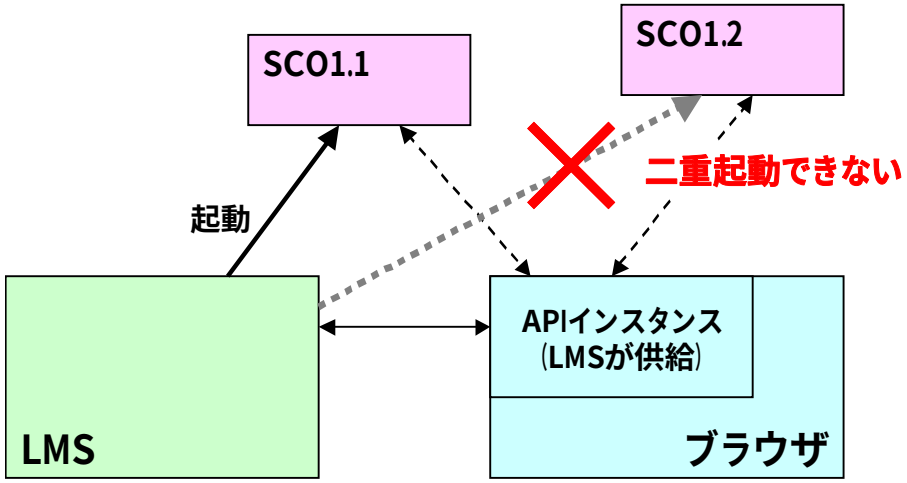

図 5.2 SCO の起動

5.3 API

5.3.1 API の概要

SCORM 2004 の RTE では，LMS と SCO とのやりとりにおいて“IEEE 1484.11.2 Standard for ECMAScript API for Content to Runtime Service Communication”（IEEE 1484.11.2 で規格化が進めてられているランタイムサービス（RTS）における学習資源に対するアプリケーション・インタフェース（API））を採用している．共通の API を利用することで，SCORM の相互運用性や再利用性といった高レベル要求事項の多くを満たす事ができる．この共通の API によって SCO が LMS と通信するための標準的な方法が提供される．LMS は，この API を実装し，クライアント SCO にそのインタフェースを提示する API インスタンスを提供しなければならない．

5.3.2 API インスタンスの概要

API インスタンス[3]は，API の機能を実現・提示する LMS のソフトウェア部品であり，SCO にそのインタフェースを提示するものである．コンテンツ作成者は，LMS 側で実装された API インスタンスを見つけることができるようコンテンツ（SCO）を開発しなければならない．

SCO は API を利用して LMS と通信を行う．SCO が起動されると，SCO は API インスタンスを通じて LMS が持つ情報をやりとりする（例：”get” or “set”）ことができる．API インスタンスと SCO との通信は，SCO が初期化して開始し，API インスタンス上の関数を呼び出すことによって実現される．

SCORM 2004 では，API インスタンスの名称は“API_1484_11[4]”である．

5.3.3 API インスタンスの実装方法

起動された SCO と LMS とで通信を確立するには，まず API インスタンスを呼び出す必要がある．SCO は API インスタンスが実装されている LMS ウィンドウ（API フレーム）が見つかるまで，親およびオープナ・ウィンドウの階層を再起的に探索することが必要となる．ここで LMS ウィンドウは，API インスタンスを（”API_1484_11”と命名される）DOM オブジェクトとしてアクセスできるように，LMS 側で提供しなければいけない．

---

[3] API インスタンス（API Instance）は，SCORM 1.2 以前では API アダプタ（API Adapter）と呼ばれていた．

[4] SCORM 2004 では，API インスタンスの名称が”API”から”API_1484_11”に変更された．

5.3.3.1 LMS の責任範囲

LMS は API インスタンスを提供しなければならない．API インスタンスを提供するための必要条件は以下のとおりである．

- API インスタンスは，“API_1484_11”と命名されるオブジェクトとしてアクセス可能でなければならない．
- API インスタンスは，SCO から ECMAScript（Java スクリプト）でアクセス可能でなくてはならない．
- LMS は，API インスタンスを含んでいる子ウィンドウあるいは，LMS ウィンドウの子フレームであるブラウザ・ウィンドウで SCO を起動しなければならない．

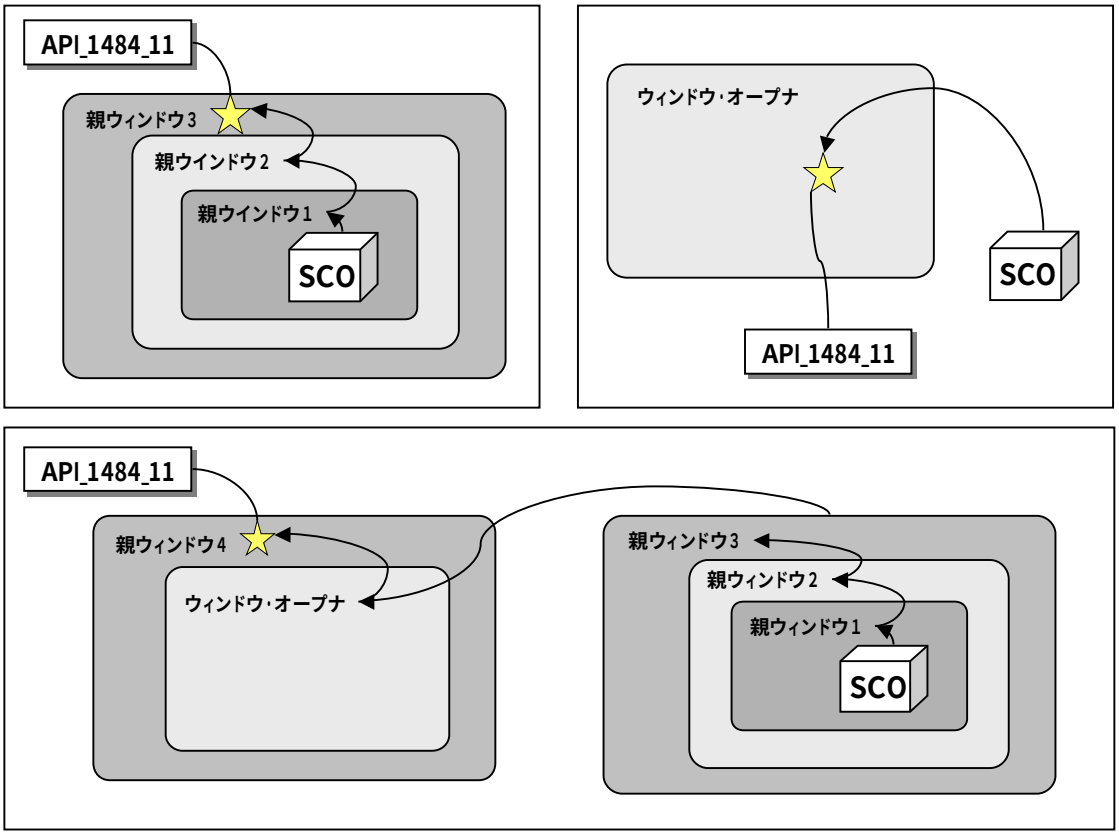

図 5.3 API の起動方法

5.3.3.2 SCO の責任範囲

SCO は, API インスタンスを探索して, LMS と通信を確立できるようにしなければならない. SCO が DOM ウィンドウ内に配置されている API インスタンスを見つける為には，以下の範囲を探索しなければならない.

- 現在のウィンドウに対する親ウィンドウの連鎖＝連鎖している親ウィンドウがトップ・ウィンドウになるまで探索する.
- ウィンドウ・オープナ（window.opener）＝SCO のウィンドウをオープンしたウィンドウ.
- ウィンドウ・オープナに対する親ウィンドウの連鎖.

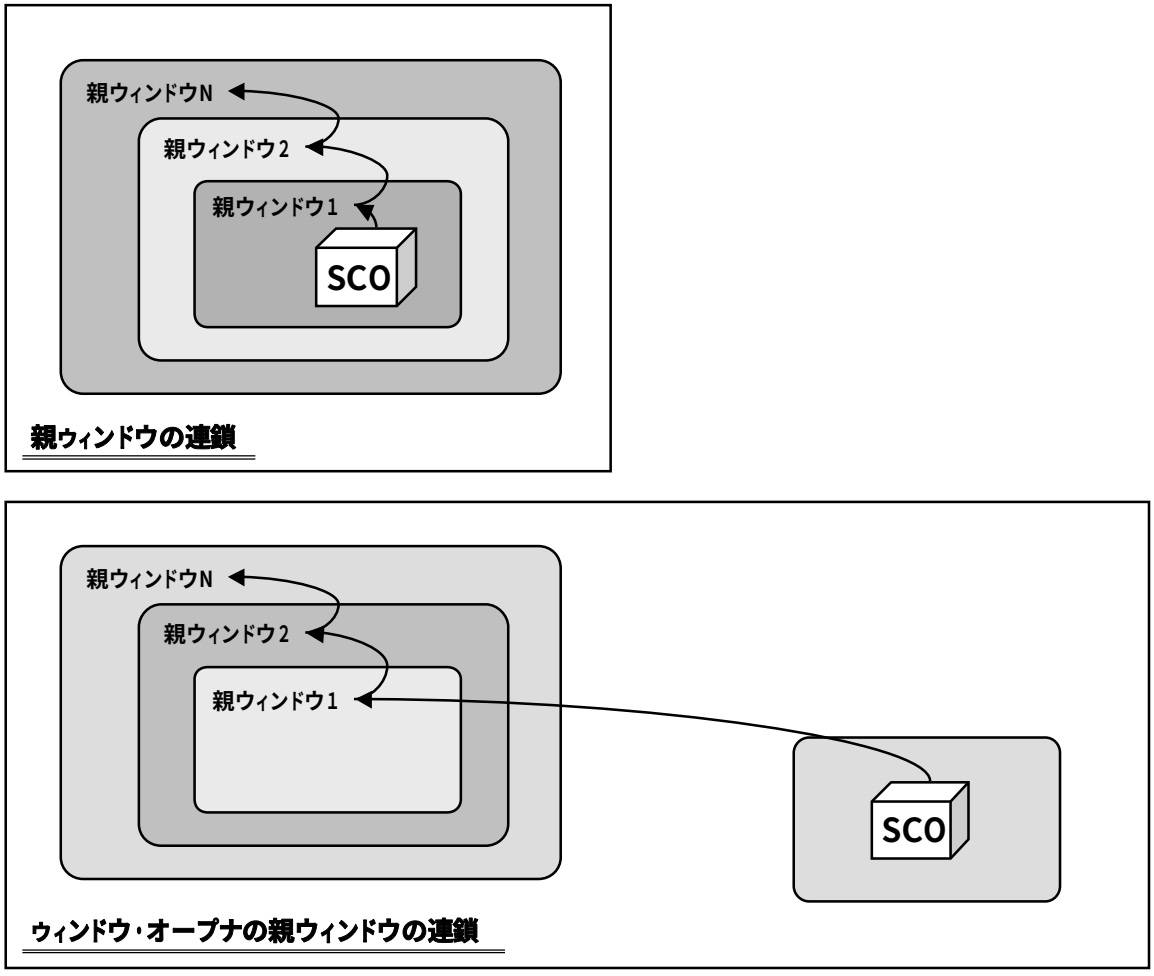

図 5.4 API インスタンスの探索

また，SCO が API インスタンスを見つけると，最低限，Initialize(“”)と Terminate(“”)の API を呼び出さなくてはならない．IEEE 標準規格では，ECMA スクリプト（Java スクリプト）の簡単なコードを利用して API インスタンスを探索する方法を提供している．しかし規格では ECMA スクリプトの利用を必要条件としていないので，他の方法を採用することもできる.

API インスタンスを再帰的に探索するためのサンプルコードは以下のとおりである.

```
<html>
<head>
<script type="text/javascript">
<! --
 // ----- API インスタンスを見つける -----
 var API = null;
 function FindAPI(win) {
    if ((typeof(win.API_1484_11) != "undefined") && (win.API_1484_11 != null)) {
       return win.API_1484_11;
    } else if (win.location == top.location) {
       return null;
    } else {
       return FindAPI(win.parent);
    }
 }
 function MyInit() {
    // API フレームを見つける
    if ((window.parent != null) && (window.parent != window)) {
       API = FindAPI(window.parent);
    }
    // ウィンドウ・オープナにある API フレームを見つける
    if ((API == null) && (window.opener != null)) {
       API = FindAPI(window.opener);
    }
    if (API != null) {
       // 初期化を行います
       API.Initialize("");
    } else {
       alert("API が見つかりません. ");
    }
 }
 funciton MyFin() {
    if (API != null) {
       // 終了処理を行います
       API.Terminate("");
    }
 }
// ->
</script>
</head>
<body onload="MyInit();" onunoad="MyFin()">
 <h1>SCORM サンプル</h1>
</body>
</html>
```

5.3.4 API 関数の概要

API 関数は，3 つのカテゴリに分けられる．3 つのカテゴリは下表のとおりである．

表 5.1 API 関数のカテゴリ

カテゴリ	説明	API 関数
セッション関数	API インスタンスを通じて SCO と LMS との間の通信セッションの開始と終了を指示する為に使用される関数	Initilize Terminate
データ転送関数	API インスタンスを通じて SCO と LMS との間でデータモデル要素の値をやりとりする為に使用される関数	GetValue SetValue Commit
サポート関数	エラー発生時に API インスタンスを通じて SCO と LMS との間の補助的な通信を行う為に使用される関数	GetLastError GetErrorString GetDiagnostic

SCORM 2004 では，LMS が提供する API 関数名が下記のとおり変更になった．
（“ LMS ”を削除し，直感的にわかりやすい名称に変更された）

表 5.2 API 関数名の変更

SCORM 1.2	SCORM 2004
LMSInitialize	Initialize
LMSFinish	Terminate
LMSGetValue	GetValue
LMSSetValue	SetValue
LMSCommit	Commit
LMSGetLastError	GetLastError
LMSGetErrorString	GetErrorString
LMSGetErrorDiagnostic	GetErrorDiagnostic

API 関数の詳細は，下表のとおりである．

表 5.3 API 関数の詳細一覧

セッション関数	
Initialize	構文： Initialize(parameter) 解説： 通信セッションを開始（初期化）する． パラメータ： (“”)－空文字列 返り値： 真理値（True／False）の文字列 “true”－LMS 側での初期化が成功したことを示す． “false”－LMS 側での初期化が失敗したことを示す． この場合，API インスタンスにエラーコードの値が設定されるので，エラー情報を解析する場合にはサポート関数を利用するとよい．
Terminate	構文： Terminate(parameter) 解説： 通信セッションを終了する．この終了処理は，API インスタンスに設定したデータを LMS に送信することも同時に行う．また，一度この終了処理を行うと，サポート関数しか呼び出すことができなくなる． パラメータ： (“”)－空文字列 返り値： 真理値（True／False）の文字列 “true”－LMS 側での終了処理が成功したことを示す． “false”－LMS 側での終了処理が失敗したことを示す． この場合，API インスタンスにエラーコードの値が設定されるので，エラー情報を解析する場合にはサポート関数を利用するとよい．
データ転送関数	
GetValue	構文： GetValue(parameter) 解説： LMS から情報を取得することができる． SCO が LMS から取得できる情報は以下のとおりである． ・LMS でサポートされているデータモデル要素の値 ・LMS でサポートされているデータモデルのバージョン ・サポートされている固有のデータモデル要素 パラメータ： データモデル要素名 返り値： 下記の２つのうち，どちらかの値，返り値はすべて文字列． ・パラメータに関係する値の文字列 ・エラーが発生した場合には，空文字列(“”)が返却される．この場合は API インスタンスにエラーコードの値が設定されるので，エラー情報を解析する場合にはサポート関数を利用するとよい．
SetValue	構文： SetValue(parameter_1,parameter_2) 解説： LMS へ情報を送信することができる． データ要素（parameter_1）に対する値（parameter_2）を指定する．API インスタンスは，設計によってデータを直ぐにサーバ側に送るか，一旦データをキャッシュして送るように実装されている． パラメータ： parameter_1－設定されるデータ要素名 parameter_2－データ要素する値（文字列） 返り値： 真理値（True／False）の文字列 “true”－LMS 側でのデータ設定が成功したことを示す． “false”－LMS 側でのデータ設定が失敗したことを示す． この場合，API インスタンスにエラーコードの値が設定されるので，エラー情報を解析する場合にはサポート関数を利用するとよい．

Commit	構文： Commit(parameter) 解説： SCO からのデータを LMS に送信する．前回 Initialize()か Commit() が実行された以降に API インスタンスによってキャッシュされたデータが送信される．LMS 側への送信が成功すると，エラー未発生のエラーコードが API インスタンスに設定され，“true”が返却される．また，API インスタンスにデータがキャッシュされていない場合でも，上記と同様に処理される． パラメータ： (“”)－空文字列 返り値：真理値（True／False）の文字列 “true”－LMS 側への送信が成功したことを示す． “false”－LMS 側への送信が失敗したことを示す． この場合，API インスタンスにエラーコードの値が設定されるので，エラー情報を解析する場合にはサポート関数を利用するとよい．
サポート関数	
GetLastError	構文： GetLastError () 解説： API インスタンスに設定された最新のエラー状態に対応するエラーコードを取得する．この関数を呼び出すと，API インスタンスのエラー状態が変化しないが，単純に要求した情報が返却される． パラメータ： 何も指定しない． 返り値： API インスタンスの最新のエラー状態に対応するエラーコードが文字列で返却される．
GetErrorString	構文： GetErrorString (prameter) 解説： 最新のエラー状態のテキストによる説明を取り出す．API インスタンスは，API に実装されているエラーコードを支援することを保証しなければならない．このエラーの呼び出しは，最新のエラー状態に影響なく，要求した情報を返却してくれる． パラメータ： エラーメッセージに対応するエラーコードの文字列． 返り値： パラメータで設定したエラーコードに対応するエラーメッセージが文字列で返却される． ・返却される文字列は，最大 255 文字まで ・エラーコードは規定されているが，エラーに関する記述は LMS 固有のものである． ・LMS が要求したエラーコードを識別できなけば，空文字列(“”)が返却される．
GetDiagnostic	構文： GetDiagnostic (prameter) 解説： LMS が個別利用するための出力関数．API インスタンスを通して，診断情報を付加して定義することができる． パラメータ： 診断の為に実装された個別の値．最大でも 255 文字の文字列に指定すべきである．パラメータの値にエラーコードを指定する場合もあるが，その制限はない． 返り値： パラメータで設定したエラーコードに対応するエラーメッセージが文字列で返却される．このエラーの呼び出しは，最新のエラー状態に影響なく，要求した情報を返却してくれる． 注：この characterstring (“”)関数のパラメータが空文字列(“”)ならば，この関数は，直前に発生したエラーについての診断情報の文字列が返却されるように推奨されている．

5.3.5 API インスタンス状態遷移

API インスタンスは，実行時に状態遷移があり，その概念的な状態モデルが SCORM 規格で定義されている．API インスタンスの状態は，規定のイベントによって状態遷移する．API インスタンスの状態の定義は下記のとおりである．

- Not Initialized（未初期化）
- Running（実行状態）
- Terminated（完了）

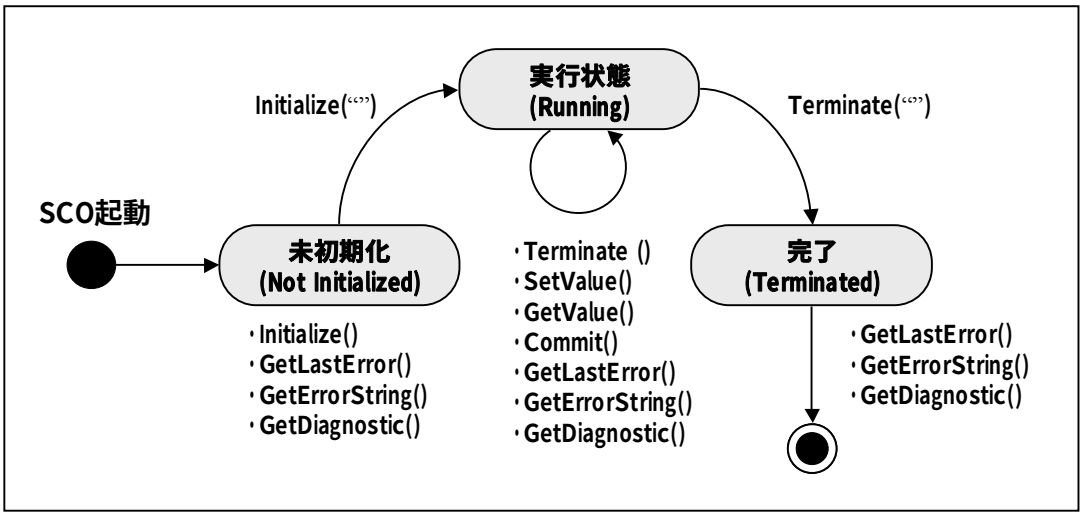

図 5.5 API インスタンスの状態遷移と SCORM API

(1) Not Initialized（未初期化）

セッション（＝通信）が確立されていない状態であり，データの読み出しや書き込みができない状態である．この状態での呼び出し可能な API 関数は以下のとおりである．

- Initialize()
- GetLastError()
- GetErrorString()
- GetDiagnostic()

(2) Running（実行状態）

セッションが確立された状態であり，SCO と LMS がデータ通信できる状態である．この状態での呼び出し可能な API 関数は以下のとおりである．

- Terminate ()
- SetValue()
- GetValue()
- Commit()
- GetLastError()
- GetErrorString()
- GetDiagnostic()

(3) Terminated（完了）

確立したセッションを終了した状態であり，以降の API 呼び出しをできない状態である．この状態での呼び出し可能な API 関数は以下のとおりである．

- GetLastError()
- GetErrorString()
- GetDiagnostic()

5.3.6 API エラーコードの概要

API 関数のサポート関数で取得されるエラーコードは，すべて int 型の数字を文字列（0～65535）で表現したものである．IEEE 規格では， 0 から 999 までを予約コードとし，追加のエラーコードは残りの範囲内（1000～65535）で実装することになっている．SCORM 2004 で定義されているエラーコードのカテゴリは以下のとおりである．

表 5.4 エラーコードカテゴリ

エラーコードカテゴリ	エラーコード範囲	説明
エラー無し (No Error)	0	エラーが発生しなかった時に API インスタンスにこのコードを返す.
一般エラー (General Errors)	100－199	API メソッド要求処理中に発生したエラー
文法エラー (Syntax Errors)	200－299	API メソッドの構文に関するエラー
ランタイムエラー (RTS Errors)	300－399	ランタイムサービスの実装に関するエラー
データモデルエラー (Data Model Errors)	400－499	LMS への送信データ, または LMS からの受信データに関するエラー
実行時エラー (Implementation-defined Errors)	1000－65535	拡張して実装したエラー

API エラーコードの詳細は，以下のとおりである．

表 5.5 API エラーコード詳細一覧

コード	記述		説明	API 関数
0	No error	エラー無し	エラーが発生しなかった時に API インスタンスにこのコードを返す.	すべて
101	General Exception	一般的な例外	API 要求処理中に存在していないエラーが発生するエラー状態である.	Initialize()
102	General Initialization Failure	初期化失敗	通信セッションの初期化が失敗した場合にエラーとなり，“Not Initialized”の状態のままになっているエラー状態を指す.	Initialize()
103	Already Initialized	初期化済み	既に通信セッションを初期化した後，すなわち“running”の状態で通信セッションを初期化しようとしたことを示すエラー状態である.	Initialize()

コード	記述		説明	API 関数
104	Content Instance Terminated	終了済み	既に通信セッションが終了している状態で, 通信セッションを初期化しようとしたことを示すエラー状態である.	Initialize()
111	General Termination Failure	一般的な終了失敗	通信セッションを終了が発生した失敗した場合に発生したことを示すエラー状態である.	Terminate ()
112	Termination Before Initialization	初期化前に終了	通信セッションを初期化する前に, SCO が通信セッションを終了しようとしたことを示すエラー状態である.	Terminate ()
113	Termination After Termination	終了後に終了	通信セッションを終了した後に, SCO が通信セッションを終了しようとしたことを示すエラー状態である.	Terminate ()
122	Retrieve Data Before Initialization	初期化前にデータ読み出し	通信セッションの初期化が成功する前に, SCO がデータの読み出しを行ったことを示すエラー状態である.	GetValue()
123	Retrieve Data After Termination	終了後にデータ読み出し	通信セッションの終了後に, SCO がデータの読み出しを行ったことを示すエラー状態である.	GetValue()
132	Store Data Before Initialization	初期化前にデータを格納	通信セッションの初期化が成功する前に, SCO が（API インスタンスに）データを格納しようとしたことを示すエラー状態である.	SetValue()
133	Store Data After Termination	終了後にデータを格納	通信セッションの終了後に, SCO が（API インスタンスに）データを格納しようとしたことを示すエラー状態である.	SetValue()
142	Commit Before Initialization	初期化前にデータを保存	通信セッションの初期化が成功する前に, SCO が LMS にデータを保存したことを示すエラー状態である.	Commit()
143	Commit After Termination	終了後にデータを保存	通信セッションの終了後に, SCO が LMS にデータを保存したことを示すエラー状態である.	Commit()
201	General Argument Error	不当な引数エラー	API 関数に不当な引数を渡そうとしたことを示すエラー状態である.	Initialize() Terminate () Commit()
301	General Get Failure	一般的なデータ読み出し失敗	データの読み出しを失敗し, 他のエラー情報が無い場合の状態を示すエラーである, このエラーが発生した場合でも, “Running”の状態である.	GetValue()
351	General Set Failure	一般的なデータ保存失敗	データの保存を失敗し, 他のエラー情報が無い場合の状態を示すエラーである, このエラーが発生した場合でも, “Running”の状態である.	SetValue()

コード	記述		説明	API 関数
391	General Commit Failure	一般的なデータ格納失敗	データの格納（コミット）を失敗し, 他のエラー情報が無い場合の状態を示すエラーである, このエラーが発生した場合でも, “Running”の状態である.	Commit()
401	Undefined Data Model Element	定義されていないデータモデル要素	API インスタンスで判別できないデータモデル要素をパラメータとして指定した場合のエラー状態である. このエラーが発生した場合でも, “Running”の状態である.	GetValue() SetValue()
402	Unimplemented Data Model Element	実装されていないデータモデル要素	LMS が実装していないデータモデル要素をパラメータとして指定した場合のエラー状態である. 拡張したデータモデルを指定した場合に起こるものである.	GetValue() SetValue()
403	Data Model Element Value Not Initialized	データモデル要素の値が初期化されていない	SCO が初期化されていないデータモデル要素の値のデータを読み出そうとしたことを示すエラー状態である. LMS によって初期化する値と, SCO によって初期化する値との２つの場合が存在する.	GetValue()
404	Data Model Element Is Read Only	データモデル要素は読出し専用	SCO が読み出し専用のデータモデル要素にデータを格納しようとしたことを示すエラー状態である.	SetValue()
405	Data Model Element Is Write Only	データモデル要素は書き込み専用	SCO が書き込み専用のデータモデル要素からデータを読み出そうとしたことを示すエラー状態である.	GetValue()
406	Data Model Element Type Mismatch	データモデル要素のデータタイプが不一致	SCO が不正なデータタイプの値をデータモデル要素に格納しようとしたことを示すエラー状態である.	SetValue()
407	Data Model Element Value Out Of Range	データモデル要素の値が範囲外	SCO が範囲外の値をデータモデル要素に格納しようとしたことを示すエラー状態である.	SetValue()
408	Data Model Dependency Not Established	データモデルに依存したデータが設定されていない	依存性のあるデータモデルの間で, 他のデータモデル要素が設定される前に設定されるべきデータモデルが設定されていないことを示すエラー状態である.	GetValue() SetValue()

SCORM 2004 では, API インスタンスの状態遷移に合わせてエラーコードが詳細化された. 変更前後のエラーコードの比較を以下に示す.

表 5.6 SCORM 1.2 と SCORM 2004 とのエラーコード比較

SCORM 1.2 Error Code	SCORM 2004 Error Code
0 – No error	0 – No error
101 – General Exception	101 – General Exception
	102 – General Initialization Failure
	103 – Already Initialized
	104 – Content Instance Terminated
	111 – General Termination Failure
	112 – Termination Before Initialization
	113 – Termination After Termination
	122 – Retrieve Data Before Initialization
	123 – Retrieve Data After Termination
	132 – Store Data Before Initialization
	133 – Store Data After Termination
	142 – Commit Before Initialization
	143 – Commit After Termination
201 - Invalid argument error	201 – General Argument Error
202 - Element cannot have children 203 - Element not an array. Cannot have count	301 – General Get Failure
	351 – General Set Failure
	391 – General Commit Failure
401 - Not implemented error	401 – Undefined Data Model Element
401 - Not implemented error	402 – Unimplemented Data Model Element
301 - Not initialized	403 – Data Model Element Value Not Initialized
403 - Element is read only	404 – Data Model Element Is Read Only
404 - Element is write only 402 - Invalid set value, element is a keyword	405 – Data Model Element Is Write Only
405 - Incorrect Data Type	406 – Data Model Element Type Mismatch
	407 – Data Model Element Value Out Of Range
	408 – Data Model Dependency Not Established

5.3.7 API エラーコードの実装例

API インスタンスの状態遷移中にエラーが発生した場合には，API インスタンスにエラーコードが格納される．すべての API 関数呼び出しで予めエラー処理を行うように SCO を実装すべきである．

エラーコードを取得するサンプルコードは以下のとおりである．

```
<html>
<head>
<script type="text/javascript">
<! --
 var API = null;
 function FindAPI(win) {
        (省略)
 }
 function MyInit() {
    (省略)
 }
 funciton MyFin() {
    (省略)
 }
// ----- エラーをチェックする -----
    function CheckError() {
        var errMsg = "";
        if (API != null) {
        if (parseInt(API.GetLastError()) > 0) {
                errMsg = API.GetErrorString() + ":" + API.GetDiagnostic();
                alert(errMsg);
        }
     }
    }
// ->
</script>
</head>
<body onload="MyInit();" onunoad="MyFin()">
 <h1>SCORM サンプル</h1>
</body>
</html>
```

5.4 データモデル

5.4.1 データモデルの概要

SCORM 2004 ランタイム環境におけるデータモデルは,“IEEE P1484.11.1 Draft Standard for Learning Technology Data Model for Content Object Communication”をベースに規定されている．この標準規格では，学習オブジェクト（SCO）から LMS への伝達情報として利用されるデータモデルの要素のまとまりを定義している．このデータモデルには,

- 学習者についての情報
- 学習者の SCO とのインタラクション
- 学習目標
- 合格状態や完了状態

などが含まれており，コンテンツの様々な目的に応じて利用できるように定義されている．このデータの主要な利用目的は以下のとおりである.

- 学習者の進捗や状態の記録
- シーケンシング決定の支援
- 学習者の SCO とのインタラクション全体の報告

前バージョンの SCORM 1.2 では,“AICC CMI001 Guideline for Interoperability”のデータモデルを採用していたが，SCORM 2004 では，AICC のデータモデルを国際標準規格として制定した IEEE 1484.11.1 標準規格書のドラフトをベースに作成されている．そのデータモデルの変更に伴い，SCORM 2004 では，主に以下のようにデータモデルの変更・追加が行われている.

- 全てのデータモデルが LMS の必須要素に
- データモデルの変更
  - －cmi.core, cmi.student_data データモデル階層を廃止し，データモデルを平坦化
  - －score.scaled の追加
  - －objectives と学習目標の対応付け
- Interaction の詳細化
- マルチバイトコードの全面採用（ISO-10646-1）

5.4.2 データモデルの基本事項

5.4.2.1 データモデル要素

データモデルを識別するために，すべてのデータモデル要素の名前は“cmi”から始まっている，このことは，LMS のデータモデル要素が IEEE P1484.11.1 規格の一部を採用していることを示している．これは，他のデータモデルが開発された場合，異なる指定で始まるモデル（例えば，cmi.elementName の代わりに adl.elementName）が導入されるということを示している.

LMS は，SCORM で記述されているすべてのデータモデル要素を実装し，動作保証することが要求される.

SCO は，すべてのデータモデルを任意で使用できる.

すべてのデータモデルの名前は，ドット表記を利用して ECMA スクリプト文字列でなければならない（例，cmi.success_status）．

5.4.2.2 シーケンシングに与える影響

SCO は SCORM ランタイムデータモデルを通して，LMS に学習者のインタラクションの結果を報告する．LMS は送られた情報を利用して，シーケンシング情報をもとに次のアクティビティを決定する．例えば，SCO が自身のアテンプト完了状態をデータモデル要素“cmi.completion_status”を通して LMS に（トラッキング情報として）“完了”と報告すると，LMS はその SCO に関するアクティビティが完了したとみなして次のアクティビティを決定する．RTE データモデルの一部はアクティビティのトラッキング情報と関連して，シーケンシングに影響を与える．トラッキング情報と RTE データモデルの対応については，7.4 で述べる．

5.4.2.3 コレクションの扱い

データモデル要素には，お互いに関連するデータの集合として規定されているものがある．このようなデータの集合は“レコード”と呼ばれる．各レコードは，配列のひとつの要素となる．レコードは配列の中のデータの順番を表すインデックス値によってアクセスされる．すべての配列のインデックスは 0 から開始される（0 基準の配列）．

データのレコードのコレクションとして定義されるデータモデル要素は下記のとおりである．

- Comments from learner (`cmi.comments_from_learner`)
- Comments from LMS (`cmi.comments_from_lms`)
- Objectives (`cmi.objectives`)
- Interactions (`cmi.interactions`)

上記のデータモデル要素は，SCO が複数のコメント，学習目標，インタラクションをトラッキングすることを意図している．学習目標（Objectives）とインタラクション（Interactions）のデータモデル要素は，SCO の学習目標やインタラクションの各々にユニークなインデックスを付与する識別子データモデル要素を含んでいる．

コレクション内のデータモデル要素を参照するには，ドット（“.”）＋番号を用いる．

cmi.objective.n.completion_status

例えば，SCO 内の最初の学習目標の完了状態のデータモデル要素の値は，"cmi.objective.0.completion_status"という表記になり，4 番目の学習目標の完了状態のデータモデル要素の値は，"cmi.objective.3.completion_status"という表記になる．

コレクション内のデータモデル要素の数を取得するには_count キーワードを利用する．例えば，SCO に対して格納されている学習目標の数を取得するためには，以下の API 呼び出しが使用される．

```
var numOfObjectives = GetValue("cmi.objectives._count");
```

5.4.2.4 最低限保証される最大値（SPM）

SCORM 2004 では，データモデル要素に最低限保証される最大値（smallest permitted

maximums (SPMs)）が定義されている．データモデル要素で SPM が定義されるケースは２つある. それは, 文字列の長さとコレクション内のデータモデル要素の数に対してである. SPM は, コレクションのエントリ数, あるいは文字列の長さの最小値として定義され, LMS はそれを受け取り, 処理できるように実装しなければならない. SPM で定義された以上の文字列やコレクションを扱えることには問題ではないが, SPM で定義された以上の長さを含んでいれば, 注意を促すように LMS は実装すべきである.

5.4.2.5 キーワードデータモデル要素

SCORM では，LMS の管理情報やデータモデル要素の状態を取得するためのデータモデル要素が定義されている．そのデータモデル要素をキーワードデータモデル要素といい，特定のデータモデル要素にしか適応されない．キーワードデータモデル要素は，読み込み専用である.

- _version：LMS によってサポートされているデータモデルのバージョンを取得するために利用される.
- _count：コレクション内のデータモデル要素を取得するために利用される.
- _children：LMS でサポートされている親のデータモデル要素内に含まれる子のデータモデルをすべて取得するために利用される．この_children の要求に対して返却された文字列が, すべてのデータモデル要素のリストをコンマで区切られた文字列となるように LMS はサポートするべきである．このキーワードデータモデル要素は, 子を持つデータモデル要素しか適用されない.

5.4.2.6 予約されている区切り文字

以下のようなケースの場合，特別な予約された区切り文字をドット表記に加えなければならない.

- 固有の文字列に対して言語タイプを与える場合（データタイプ: localized_string_type）
- インタラクションに対して学習者のレスポンスに順序性があるかどうかを表現する場合
- インタラクションに対して学習者のレスポンスが複数があるかどうかを表現する場合
- リスト内の値のセット，あるいは値のペアを表現する場合

上記のどの場合でも，該当する場合，デフォルト値が与えられる．特別な予約されている区切り文字が指定されない場合，このデフォルト値を使用する．どんな場合でも区切り文字は，SPM のカウントの対象とはならない.

表 5.7 予約されている区切り文字

予約された区切り文字	デフォルト値	例
{lang=<language_type>}	{lang=en}	{lang=en}
{case_matters=<boolean>}	{case_matters=false}	{case_matters=true} {case_matters=false}
{order_matters=<boolean>}	{order_matters=true}	{order_matters=true} {order_matters=false}
[.]	該当なし．値を与える必要がある．	インタラクションのコレクションに対する値のペアを区切るために利用される． 1[.]a
[,]	該当なし．値を与える必要がある．	インタラクションのコレクションに対する値のセットを区切るために利用される． 1[.]a[,]2[.]c[,]3[.]b
[:]	該当なし．値を与える必要がある．	数値の範囲の間を区切る為に利用される． 1[:]100 - 1から100までの数値の範囲

5.4.2.7 **データタイプ**

それぞれのデータモデル要素には，データタイプ（データ型）が指定されている．データモデル要素の値は，その指定されたデータ型の要件を忠実に守る必要がある．以下にデータ型とそれぞれのデータ型に関する要求事項について解説する．

(1) characterstring**（キャラクタ文字列）**

ISO 10646 **にて定義されているキャラクタの文字列．**ISO 10646 **は，**Unicode **標準に相当する．**

(2) localized string type**（ローカル文字列）**

言語指定を有する文字列．言語情報が重要とされるデータモデル要素がある．SCORM **では文字列を表すための予約された区切り文字：**{lang=<language_type>} **が適用される．このローカル文字列の表現はオプションである．特に指定がなければ，デフォルトの言語で指定される（英語（**lang=en**）が指定される）．**

この文字列のフォーマットは，次のように記述される．

“{lang=<language_type>}<actual characterstring>”

例：{lang=ja}**仲林 清**

(3) language type**（言語タイプ）**

言語を表すために使用されるデータタイプ．このデータタイプのフォーマットは，言語コード（lnagcode**）と補助的にハイフンでサブコード（**subcode**）から成る文字列である．**

language_type ::= langcode [“-” subcode]*

例：ja, en-GB

(4) long identifier type**（長い識別子タイプ）**

ラベル，あるいは識別子を表すデータタイプ．このラベル，あるいは識別子は，SCO **の文**

脈の中でユニークでなければならない．この識別子タイプは，URI として定義される構文に従う文字列でなければならない．SCORM では，URI が URN（Uniform Resource Name）の形式のグローバルにユニークな識別子となることを推奨している．この識別子タイプの値は，4000 文字の SPM で実装される．

<URN> ::= “urn:”<NID>“:”<NSS>

<NID>は，Namespace Identifier，<NSS> は，Namespace Specific String

例：urn:ADL:interaction-id-0001

(5) short identifier type（短い識別子タイプ）

このラベル，あるいは識別子は，SCO の文脈の中でユニークでなければならない．この識別子タイプは，URI として定義される構文に従う文字列でなければならない．この識別子タイプは，グローバルでユニークな識別として利用することを想定していない．この識別子タイプの値は，250 文字の SPM で実装される．

(6) integer（整数）

データモデルの要素が，正の整数（例：1, 2, 3），負の整数（例：-1, -2, -3）と 0 の値をとる場合に指定する．

(7) state（状態）

データモデル要素の値に状態のセットが定義されているものがある．これは次のような記述で定義される．

例：state (browse,normal,review)

(8) real (10,7)

このデータタイプは，有効数字 7 桁の実数を意味する．

(9) time (second, 10, 0)

時間を表すデータタイプ．このデータタイプは，1 秒単位までの正確性が必要とされる（0.01 秒はオプション）．

例：2003-07-25T03:00:00

(10) timeinterval (second, 10, 2)

データモデル要素の値に対して経過時間を表すためのデータタイプ．

例：P1Y3M2DT3H (＝１年 3 ヶ月と２日と３時間)

5.4.2.8 拡張されているデータモデル

SCORM ランタイム環境データモデルは，それ自身拡張されるべきではない．もし，LMS が定義されていないデータモデル要素名の API 要求を受け取れば，エラーとされるべきである．

5.4.3 SCORM ランタイム環境におけるデータモデル

5.4.3.1 データモデルの概要

SCORM ランタイム環境におけるデータモデルには，SCO 実行時に SCO によって LMS に記録されるデータモデル要素が含まれている．このデータモデル要素は，状態，得点，インタラクション, 学習目標などの記録項目として利用されたり, SCO と LMS 間で情報をやりとりしたり，

シーケンシングに影響を与えたりする．データモデル要素の概要は下記のとおりである．各データモデル要素の詳細は次項で示す．

表 5.8 SCORM ランタイム環境データモデル一覧

No	データモデル要素	データ内容	説明
1	cmi.comments_from_learner	学習者からのコメント	学習者からのテキストを記録.
2	cmi.comments_from_lms	LMSからのコメント	学習者に提供することを目的とするコメントや注釈を記録する
3	cmi.completion_status	完了状態	学習者がSCOを完了しているかどうかを示す
4	cmi.completion_threshold	完了状態のしきい値	SCOが学習者の進捗状況を完了とみなすための目安にする値を示す
5	cmi.credit	評価	学習者がSCOでのパフォーマンスに対して評価（記録更新）されているかどうかを示す.
6	cmi.entry	エントリ	学習者以前にアクセスしたかどうかを記す情報を記録する
7	cmi.exit	退出	SCOからどのように，なぜ退出したかを記録する
8	cmi.interactions	インタラクション	成績記録や評価の目的でインタラクション（学習者の応答）に関する情報を定義する
9	cmi.launch_data	起動データ	起動の際に利用するSCO独自のデータを提供する
10	cmi.learner_id	学習者ID	どの学習者に対してSCOが起動されたかを示す
11	cmi.learner_name	学習者名	学習者の名前
12	cmi.learner.preference	学習者のプリファレンス	SCOを利用する際の学習者の固有の設定情報
13	cmi.location	ロケーション	SCOの中のブックマークなどのSCO独自の内容
14	cmi.max_time_allowed	最大許容時間（タイムリミット）	学習者がSCOを利用して学習することを許されている累積時間
15	cmi.mode	動作モード	学習者に与えられるSCOの動作モードを示す
16	cmi.objectives	学習目標	SCOに関連する学習目標を設定
17	cmi.progress_measure	進捗状態の測定値	SCOの完了への進捗状態の測定値を示す
18	cmi.scaled_passing_score	合格点（正規化表現）	SCOに対する正規化された合格点を示す
19	cmi.score	得点	SCOに対する学習者の得点を示す
20	cmi.session_time	セッション時間	SCOに対して学習者が費やしたセッション時間を示す
21	cmi.success_status	合格状態	学習者がSCOを合格したどうかを示す
22	cmi.suspend_data	中断データ	SCO中断時に保存しておく情報
23	cmi.time_limit_action	タイムリミット超過後の動作	最大許容時間（タイムリミット）を超過した時にSCOが何をすべきかを示す
24	cmi.total_time	全学習時間	現在の学習セッションで学習者が試行した学習のセッション時間の合計を示す

5.4.3.2 データモデルの詳細

表 5.9 SCORM ランタイム環境データモデル詳細

No	データモデル要素	説明	データタイプ	値空間	SCO	備考
0.1	cmi._version	データモデルのバージョン	ｷｬﾗｸﾀ文字列 (characterstring)	ISO-10646-1	R	値はピリオドで区切る “1.0”
1.	cmi.comments_from_learner	SCO の学習体験についての学習者からコメント	コレクション (collection) SPM: 250 個まで		—	
1.0.1	cmi.comments_from_learner._children	学習者からのコメントのデータ要素のリスト	ｷｬﾗｸﾀ文字列 (characterstring)	ISO-10646-1	R	
1.0.2	cmi.comments_from_learner._count	学習者からのコメントのデータ要素の数	整数 (integer)	0 以上の整数	R	
1.1	cmi.comments_from_learner.n.comment	学習者からのコメント	ローカル文字列 (localized_string _type) SPM: 4000 文字まで	ローカル情報を持つ文字列 (ISO-10646-1)	R/W	初期値は設定されない
1.2	cmi.comments_from_learner.n.location	コメントを適用する SCO の位置	ｷｬﾗｸﾀ文字列 (characterstring) SPM: 250 文字まで	ISO-10646-1	R/W	
1.3	cmi.comments_from_learner.n.timestamp	コメントを作成・更新した時間	時間 time (second,10,0)		R/W	
2.	cmi.comments_from_lms	学習者に提供する SCO に関する情報	コレクション (collection) SPM: 100 個まで		—	
2.0.1	cmi.comments_from_lms._children	LMS からのコメントのデータ要素のリスト	ｷｬﾗｸﾀ文字列 (characterstring)	ISO-10646-1	R	
2.0.2	cmi.comments_from_lms._count	LMS からのコメントのデータ要素の数	整数 (integer)	0 以上の整数	R	
2.1	cmi.comments_from_lms.n.comment	LMS からのコメント	ローカル文字列 (localized _string_type) SPM: 4000 文字まで	ローカル情報を持つ文字列 (ISO-10646-1)	R	
2.2	cmi.comments_from_lms.n.location	コメントを適用する SCO の位置	ｷｬﾗｸﾀ文字列 (characterstring) SPM: 250 文字まで	ISO-10646-1	R	
2.3	cmi.comments_from_lms.n.timestamp	コメントを作成・更新した時間	時間 time (second,10,0)		R	
3.	cmi.completion_status	学習者が SCO を完了したかどうか	状態 (state)	完了 “completed” 未完了 “incomplete” 未試行 “not_attempted” 不明 “unknown”	R/W	デフォルト値は, “unknown” SCO が書き込むことを想定しており, シーケンシングの進捗状態に影響を与える.

＊SCO 欄の表記　R : read only(読み出しのみ),　W : write only(書き込みのみ)

R/W : read/write(読み書き可能)

No	データモデル要素	説明	データタイプ	値空間	SCO	備考
4.	cmi.completion_threshold	SCO を完了とみなすかどうかを決定するためのしきい値	0…1 までの数 real(10,7) range (0..1)		R	LMS は「17. cmi.progress_measure」の値と比較して完了状態を決定する（SCO からの状態設定「3. cmi.completion_status」よりも優先される） imsmanifest の <adlcp:competion Threshold> で定義された値で初期化される
5.	cmi.credit	SCO 内での学習者のパフォーマンスに対して（LMS が）評価するかどうか	状態 (state)	評価する “credit” 評価しない “no_credit”	R	デフォルト値は, “credit”
6.	cmi.entry	以前に SCO にアクセスしたかどうかの情報	状態 (state)	初回試行 “ab_initio” 中断再開 “resume” 情報なし “”(空文字列)	R	
7.	cmi.exit	SCO からどのように，なぜ終了したかを記録する	状態 (state)	時間切れ “time-out” 中断 “suspend” 中途であるが終了を希望した “logout” 通常終了 “normal” 情報なし “”(空文字列)	W	
8.	cmi.interactions	SCO に対する学習者の応答（インタラクション）を LMS に記録するために使用する	コレクション (collection) SPM: 250 個まで		—	
8.0.1	cmi.interactions._children	インタラクションのデータ要素のリスト	ｷｬﾗｸﾀ文字列 (characterstring)	ISO-10646-1	R	
8.0.2	cmi.interactions._count	インタラクションのデータ要素の数	整数 (integer)	0 以上の整数	R	
8.1	cmi.interactions.n.id	インタラクションのデータの識別子	長い識別子 (long_identifier_type) SPM: 4000 文字まで	URI (RFC 2396) で表す文字列 URN (RFC 2141) を推奨	R/W	SCO 内でユニークでなければならない.

＊SCO 欄の表記　R : read only(読み出しのみ),　W : write only(書き込みのみ)

R/W : read/write(読み書き可能)

No	データモデル要素	説明	データタイプ	値空間	SCO	備考
8.2	cmi.interactions.n.type	インタラクションのデータのタイプ	状態 (state)	○× “true-false” 選択 “choice” 穴埋め “fill-in” 論述式 “long-fill-in” アンケート “likert” 組合せ “matching” パフォーマンス測定 “performance” 並び替え “sequencing” 数値 “numeric” その他 “other”	R/W	このデータ要素は, “correct_response” “learner_response” に依存しているので, 上記データ要素を設定する前に設定しなければならない.
8.3	cmi.interactions.n.objectives	インタラクション内の学習目標	コレクション (collection) SPM: 10 個まで		_	
8.3.0.1	cmi.interactions.n.objectives._count	インタラクション内の学習目標の数	整数 (integer)	0 以上の整数	R	
8.3.1	cmi.interactions.n.objectives.n.id	インタラクション内の学習目標の識別子	長い識別子 (long_identifier_type) SPM: 4000 文字まで	URI (RFC 2396) で表す文字列 URN (RFC 2141) を推奨	R/W	
8.4	cmi.interactions.n.timestamp	インタラクションが発生した時間	時間 time(second,10,0)		R/W	
8.5	cmi.interactions.n.correct_responses	インタラクションの正答情報	コレクション (collection) SPM: 10 個まで		_	
8.5.0.1	cmi.interactions.n.correct_responses._count	インタラクションの正答情報の数	整数 (integer)	0 以上の整数	R	
8.5.1	cmi.interactions.n.correct_responses.n.pattern	インタラクションの各応答に対するパターン	8.2 cmi.interactions.n.type に依存		R/W	
8.6	cmi.interactions.n.weighting	インタラクションに与えられる得点計算の重み付け	実数型 real (10,7)	有効数字 7 桁の実数	R/W	
8.7	cmi.interactions.n.learner_response	インタラクション内の学習者の各応答	8.2 cmi.interactions.n.type に依存		R/W	
8.8	cmi.interactions.n.result	インタラクションの各結果	状態 (state)	正しい “correct” 間違い “incorrect” 予期しない結果 “unanticipated” どっちつかず “neutral” 数値 real(10,7)	R/W	

＊SCO 欄の表記　R : read only(読み出しのみ),　W : write only(書き込みのみ)

R/W : read/write(読み書き可能)

No	データモデル要素	説明	データタイプ	値空間	SCO	備考
8.9	cmi.interactions.n.latency	インタラクション中の学習者の反応時間	経過時間 timeinterval (second,10,2) 0.01 秒単位まで		R/W	
8.10	cmi.interactions.n.description	インタラクションについての記述	ローカル文字列 (localized _string_type) SPM: 250 文字まで	ローカル情報を持つ文字列	R/W	
9.	cmi.launch_data	SCO を初期化するための起動データを提供する	キャラクタ文字列 (characterstring) SPM: 4000 文字まで	ISO-10646-1	R	imsmanifest の <adlcp:dataFrom LMS>で定義された値で初期化される
10.	cmi.learner_id	SCO を起動した学習者を識別するための情報	長い識別子 (long_identifier_type) SPM: 4000 文字まで	URI (RFC 2396) で表す文字列 URN (RFC 2141) を推奨	R	LMS から提供される
11.	cmi.learner_name	SCO を起動した学習者の名前	ローカル文字列 (localized _string_type) SPM: 250 文字まで	ローカル情報を持つ文字列	R	LMS から提供される
12..	cmi.learner_preference	SCO の学習者利用に関連する個別情報			-	
12.0.1	cmi.learner_preference._children	上記データ要素のリスト	キャラクタ文字列 (characterstring)	ISO-10646-1	R	
12.1	cmi.learner_preference.audio_level	学習者のオーディオレベル関する個別情報	0 以上の実数 real(10,7), range (0..*)	有効数字 7 桁の実数	R/W	
12.2	cmi.learner_preference.language	学習者の使用言語に関する個別情報	言語タイプ (language_type) SPM: 250 文字まで	ISO-646	R/W	
12.3	cmi.learner_preference.delivery _speed	学習者の配信速度に関する個別情報	0 以上の実数 real(10,7), range (0..*)	有効数字 7 桁の実数	R/W	
12.4	cmi.learner_preference.audio _captioning	学習者の音声テキスト表示に関する個別情報	状態 (state)	テキスト OFF “-1” 状態の変化なし “0” テキスト ON “1”	R/W	各状態の語彙は, “off” “no_change” “ on” に相当する.
13.	cmi.location	SCO の格納場所	キャラクタ文字列 (characterstring) SPM: 1000 文字まで	ISO-10646-1	R/W	SCO によって提供される. 初期状態は, “” (空文字列) LMS はこのデータを解釈･変更してはいけない SCO 退出時の終了ポイントを保存するのに利用してもよい.
14.	cmi.max_time_allowed	SCO の学習試行時間	経過時間 timeinterval (second,10,2) 0.01 秒単位まで		R	imsmanifest の <imsss:attemptAbsoluteDurationLimit>で定義された値で初期化される

＊SCO 欄の表記　R : read only(読み出しのみ),　W : write only(書き込みのみ)

R/W : read/write(読み書き可能)

No	データモデル要素	説明	データタイプ	値空間	SCO	備考
15.	cmi.mode	SCO を学習者にどのように提供するかのモードを設定する 起動後の SCO の動作を示す.	状態 (state)	閲覧のみ “browse” 学習履歴を記録する(通常) “normal” 学習済み “review”	R	指定がない場合には, “normal” 関連項目: 「5. cmi.credit」
16.	cmi.objectives	SCO を通じた 学習活動に関する学習目標を記録するために利用する.	コレクション (collection) SPM: 100 個まで		—	
16.0.1	cmi.objectives._children	学習目標のデータ要素のリスト	キャラクタ文字列 (characterstring)	ISO-10646-1	R	
16.0.2	cmi.objectives._count	学習目標のデータ要素の数	整数 (integer)	0 以上の整数	R	
16.1	cmi.objectives.n.id	学習目標の識別子	長い識別子 (long_identifier_type) SPM: 4000 文字まで	URI (RFC 2396) で表す文字列 URN (RFC 2141) を推奨	R/W	少なくとも SCO 内ではユニークでなければならない. imsmanifest の <imsss:objectives> の属性 objective ID で定義された値で初期化される
16.2	cmi.objectives.n.score	学習目標の得点			—	
16.2.0.1	cmi.objectives.n.score._children	学習目標の得点のデータ要素のリスト	キャラクタ文字列 (characterstring)	ISO-10646-1	R	
16.2.1	cmi.objectives.n.score.scaled	学習者の学習に対するパフォーマンスを反映した数値. -1〜1 の範囲で正規化される.	-1〜1 までの実数 real (10,7) range (-1..1)	有効数字 7 桁の実数 -1.0〜1.0 の範囲の値	R/W	SCO に関する学習アクティビティの学習目標の値に影響を与える.
16.2.2	cmi.objectives.n.score.raw	学習者の学習に対するパフォーマンスを反映した数値.	実数 real (10,7)	有効数字 7 桁の実数	R/W	
16.2.3	cmi.objectives.n.score.min	学習目標に対する最低得点	実数 real (10,7)	有効数字 7 桁の実数	R/W	
16.2.4	cmi.objectives.n.score.max	学習目標に対する最高得点	実数 real (10,7)	有効数字 7 桁の実数	R/W	
16.2.5	cmi.objectives.n.success_status	学習目標を学習者が合格したかどうかの状態	状態 (state)	合格 “passed” 不合格 “failed” 不明 “unknown”	R/W	SCO に関する学習アクティビティの学習目標の値 (Objective Progress Status) に影響を与える.
16.2.6	cmi.objectives.n.completion_status	学習目標を学習者が完了したかどうかの状態	状態 (state)	完了 “completed” 未完了 “incomplete” 未試行 “not_attempted” 不明 “unknown”	R/W	
16.2.7	cmi.objectives.n.progress_measure	学習目標の完了に向けた学習者が進捗状態	0〜1 までの実数 real (10,7) range (0..1)	有効数字 7 桁の実数 0.0〜1.0 の範囲の値	R/W	

＊SCO 欄の表記 R : read only(読み出しのみ), W : write only(書き込みのみ)

R/W : read/write(読み書き可能)

No	データモデル要素	説明	データタイプ	値空間	SCO	備考
16.2.8	cmi.objectives.n.description	学習目標に関する記述	ローカル文字列 (localized _string_type) SPM: 250 文字まで	ローカル情報を持つ文字列 (ISO-10646-1)	R/W	
17.	cmi.progress_measure	SCO の完了に向けた学習者の進捗状態	0～1 までの実数 real (10,7) range (0..1)	有効数字 7 桁の実数 0.0～1.0 の範囲の値	R/W	「3. cmi.completion _status」の値とマッピングされる. 0 ⇒ “not attempted” 1 ⇒ “completed” 0 > value < 1 ⇒ “incomplete” 注) ただし, しきい値が設定されていない場合
18.	cmi.scaled_passing_score	SCO を合格するために必要とされる正規化された得点	-1～1 までの実数 real (10,7) range (-1..1)	有効数字 7 桁の実数 -1.0～1.0 の範囲の値	R	imsmanifest の <imsss:minNormal-izedMeasure>で定義された値で初期化される
19.	cmi.score	学習者の得点			-	主に SCO によって利用される
19.0.1	cmi.score._children	学習者の得点のデータ要素のリスト	キャラクタ文字列 (characterstring)	ISO-10646-1	R	
19.1	cmi.score.scaled	学習者の正規化した得点	-1～1 までの実数 real (10,7) range (-1..1)	有効数字 7 桁の実数 -1.0～1.0 の範囲の値	R/W	この値と SCO の学習目標の最初の値 (Objective Measure Status) は同期されるべきである.
19.2	cmi.score.raw	学習者の得点	実数 real (10,7)	有効数字 7 桁の実数	R/W	
19.3	cmi.score.max	学習者の最高得点	実数 real (10,7)	有効数字 7 桁の実数	R/W	
19.4	cmi.score.min	学習者の最低得点	実数 real (10,7)	有効数字 7 桁の実数	R/W	
20.	cmi.session_time	SCO に対して現在の学習者が費やした時間	経過時間 timeinterval (second,10,2) 0.01 秒単位まで		W	
21.	cmi.success_status	学習者が SCO を合格したかどうかの状態	状態 (state)	合格 “passed” 不合格 “failed” 不明 “unknown”	R/W	SCO によって初期化される. LMS ではこの値を制御できないが, 「18. cmi.scaled _passing_score」を設定することにより間接的に書き換えることができる. (この SCO から合格設定よりも優先される) この値と SCO の学習目標の最初の値 (Objective Measure Status) は同期されるべきである.

＊SCO 欄の表記　R : read only(読み出しのみ),　W : write only(書き込みのみ)

R/W : read/write(読み書き可能)

No	データモデル要素	説明	データタイプ	値空間	SCO	備考
22.	cmi.suspend_data	SCO が中断・再開する際に利用するデータ	キャラクタ文字列 (characterstring) SPM: 4000 文字まで	ISO-10646-1	R/W	学習者のセッション内で LMS はこのデータを解釈・変更してはいけない 関連項目:「13. cmi.location」
23.	cmi.time_limit_action	制限時間を超えた場合の SCO の処理を示す	状態 (state)	退出・ﾒｯｾｰｼﾞ有 “exit,message” 継続・ﾒｯｾｰｼﾞ有 “continue,message” 退出・ﾒｯｾｰｼﾞ無 “exit,no essage” 継続・ﾒｯｾｰｼﾞ無 “continue,no message”	R	imsmanifest の <adlcp:timeLimit Action>で定義された値で初期化される デフォルト値は, “continue,no message”
24.	cmi.total_time	すべての学習者のセッション時間の合計	経過時間 timeinterval (second,10,2) 0.01 秒単位まで		R	SCO 側でセッション時間（cmi.session_time）を書き込まないと, 最新の学習試行時間（cmi.total_time）が更新されない

＊SCO 欄の表記　R : read only(読み出しのみ),　W : write only(書き込みのみ)

R/W : read/write(読み書き可能)

5.4.4 データモデルの実装例

データモデルは, LMS と SCO の間でやりとりされるデータ要素の集まりである. LMS と SCO は各データ要素について「知っている」ものとして通信を行う．通信の主な動作は,

- 初期化
- データの読み出し
- データの書き込み
- 保存／格納

である.

5.4.4.1 読み出しのみの例

(1) 記述例

例１）学習者の ID

```
id = GetValue("cmi.learner_id");
```

例２）SCO の起動データ

```
Lprm = GetValue("cmi.launch_data");
```

例３）SCO の学習試行時間

```
attempt_time = GetValue("cmi.max_time_allowed");
```

(2) 動作

・LMS はセッション情報ないしコース構造の値で初期値設定

・SCO はこれらのデータを読み出して利用

・SCO はこのデータ要素を変更して，保存することはできない.

(3) 解説

・例１）の初期値は LMS が管理しているセッション情報より提供される.

・例２）および例３）の初期値は，それぞれ下記のマニフェストファイル（imsmanifest.xml）で定義された値により初期化される.

例２）SCO の起動データ：<adlcp:dataFromLMS>の値により初期化

例３）SCO の学習試行時間：<imsss:attemptAbsoluteDurationLimit>の値により初期化

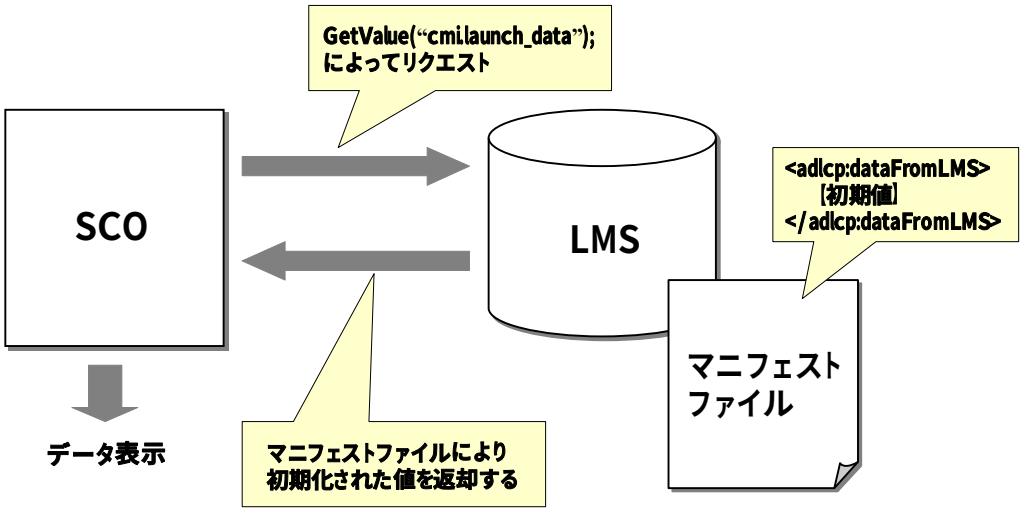

図 5.6 データモデルの読み出し

5.4.4.2 書き込みのみの例

(1) 記述例

例４）セッション時間

```
SetValue(“cmi.session_time”, “05:15:00”);
```

(2) 動作

・LMS による初期化なし

・SCO はデータを書き込む

・LMS はこれらのデータを処理・格納・保存する

(3) 解説

・SCO の学習結果の記録に利用されるデータが主である.

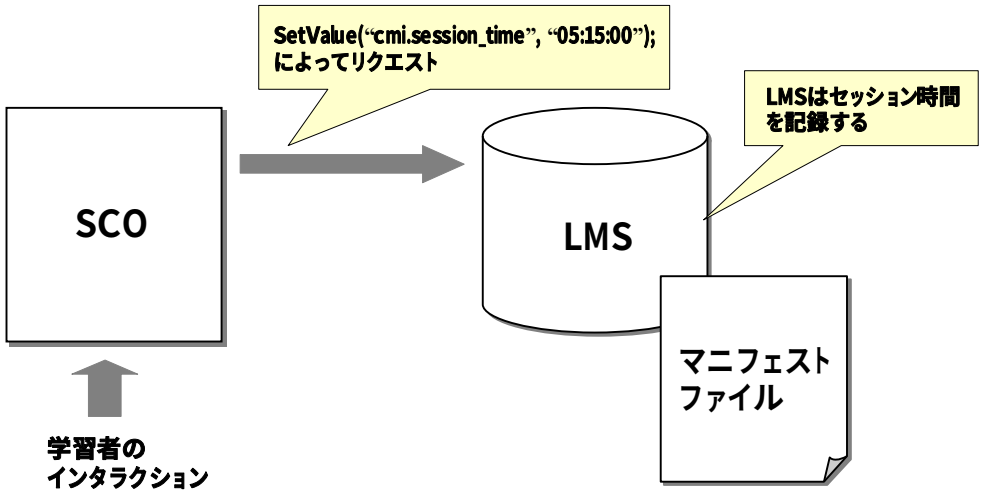

図 5.7 データモデルの書き込み

5.4.4.3 読み書きの例

(1) 記述例

例５）学習目標の読み出し

```
Loc = GetValue("cmi.success_status");
```

例６）学習目標の書き込み

```
SetValue("cmi.success_status", "passed");
```

(2) 動作

・LMS は適当な値で初期値設定

・SCO はこれらの値を読み出して利用・更新

・LMS はこれらの値を処理・格納・保存

・SCO は後に再度これらの値を読み出して利用

(3) 解説

・進捗状態や合格状態, 得点などセッション中で状態変化があるデータモデル要素に適応される.

・SCO によって，初期化すべきである.

・状態や数値を設定する為，キーワードや範囲外の数値を設定すると，エラーが発生する.

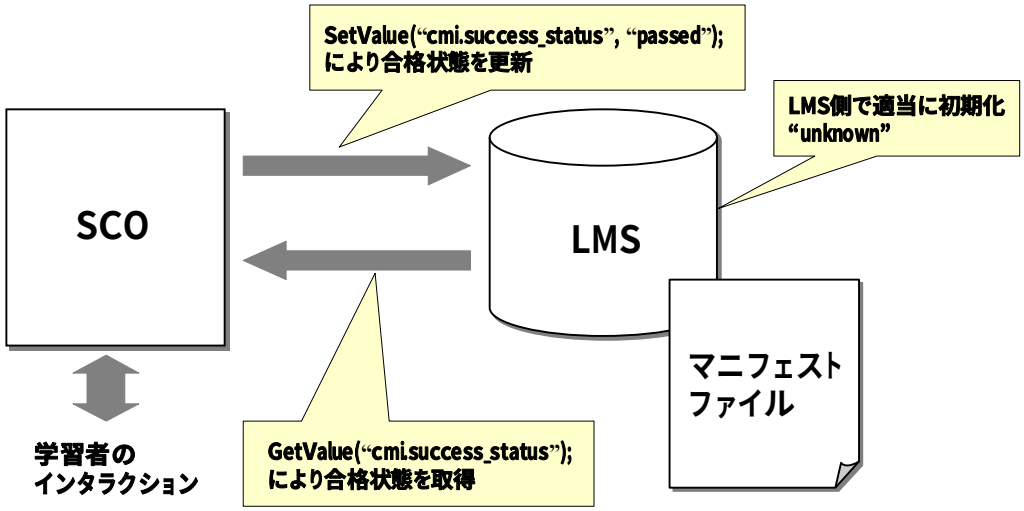

図 5.8 データモデルの読み書き

6. シーケンシングの実現

本節ではシーケンシング動作の実現方法について述べる．シーケンシング動作はシーケンシングプロセスと呼ばれる手順の集合によって実現される．シーケンシングプロセスの間では要求（リクエスト）が引き渡されていく．また，シーケンシングプロセスは擬似コードによって表現される．この擬似コードの処理の概要を示す．

6.1 シーケンシングプロセス

図 3.1 に示したようにシーケンシング動作は，いくつかの「プロセス」と呼ばれる手順の集合によって実現される．規格書ではこの手順を指すのに「動作（Behavior）」という用語も用いているが，ここではプロセスで統一する．これらのプロセス間の関係の詳細を図 6.1 に示す．

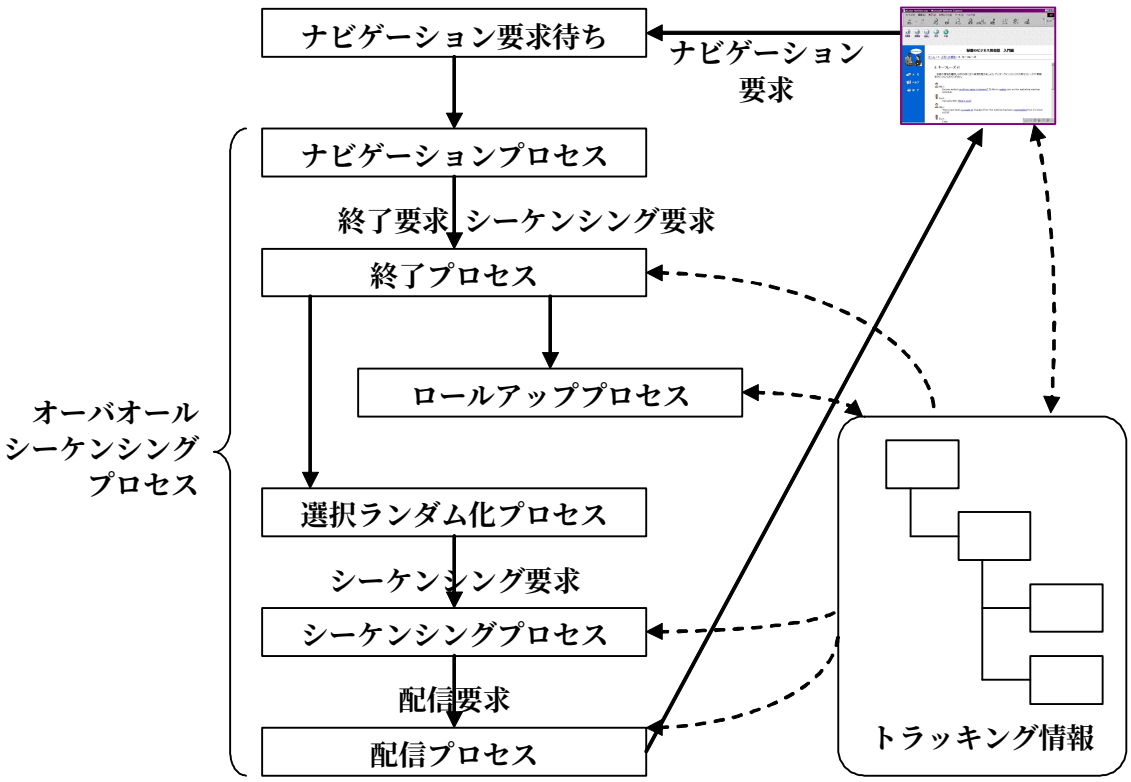

図 6.1 オーバオールシーケンシングプロセス

シーケンシング動作全体は「オーバオールシーケンシングプロセス」と呼ばれるプロセスによって実現される．オーバオールシーケンシングプロセスは，ブラウザからのナビゲーション要求によって処理を開始し，ブラウザへ配信するアクティビティを決定し，次のナビゲーション要求を待つ，という動作を繰り返す．

ブラウザからのナビゲーション要求はナビゲーションプロセスに送られる．ナビゲーションプロセスは，ナビゲーション要求の実行可否を判断し，3.3 に示したように，終了要求とシーケンシング要求を発生する．

終了要求は，終了プロセスで処理され，アクティビティのみの終了，教材全体に終了ないし中断，といった処理が行われる．アクティビティを終了してアクティビティのトラッキング情報を確定した後，終了プロセスはロールアッププロセスを呼び出し，ロールアップルールに従って，アクティビティツリー全体のトラッキング情報を更新する．またロールアップの後で，ポストコンディションルールの評価が行われ，ルールが成り立った場合，学習者からのシーケンシング要求が他のシーケンシング要求に変換される．

この後，選択ランダム化プロセスによって，必要に応じて，提示の対象となるアクティビティの並べ替えを行う．

次にシーケンシング要求はシーケンシングプロセスに送られ，提示するアクティビティが決定される．この決定のために，シーケンシングプロセスでは，送られてきたシーケンシング要求ごとに異なるプロセスを実行し，その中で，シーケンシング制御モード，制限条件，プリコンディションルールなどを評価しながら，提示するアクティビティを決定する．

最後に，決定されたアクティビティは配信要求として，配信要求プロセスに送られる．配信要求プロセスでは, そのアクティビティの制限条件, プリコンディションルールなどを再度確認し，そのアクティビティのトラッキング情報の取得を開始して，アクティビティの配信を行う．

オーバオールシーケンシングプロセスを構成する各プロセスはトラッキング情報の更新ないし参照を行う．末端のアクティビティのトラッキング情報は，ブラウザ中の SCO からランタイム環境を経由して更新される．ブラウザ中の SCO は逆に末端のアクティビティのトラッキング情報を参照することができる．ロールアッププロセスで，末端のアクティビティのトラッキング情報からアクティビティツリー全体のトラッキング情報が更新される．終了プロセス，シーケンシングプロセス，配信要求プロセスでは，ロールアッププロセスで更新されたトラッキング情報を参照して，シーケンシングルールや制限条件の評価を行う．

6.1.1 ナビゲーションプロセス

ナビゲーションプロセスでは，ブラウザからのナビゲーション要求を受け取り，3.3 に示したように，終了要求とシーケンシング要求を発生する．このとき，すでに学習が始まっているのに Start ナビゲーション要求が発行されたなど，明らかに無効なナビゲーション要求は無視され，システムはナビゲーション要求待ちの状態に戻る．それ以外の場合，ナビゲーションプロセスは表 3.3 に示したように，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．

6.1.2 終了プロセス

終了プロセスでは，ナビゲーションプロセスから終了要求を受け取り，現在のアクティビティを終了させる．終了要求の種別によって，現在のアクティビティは以下のいずれかの形で終了する．

- Exit, ExitAll の場合．SCO のランタイム環境情報の値が，アクティビティのトラッキング情報の値に反映される．現在のアクティビティのアテンプトが終了する．ExitAll の場合，シー

ケンシングセッションを終了する．

- Abandon, AbandonAll の場合．SCO のランタイム環境情報の値は，アクティビティのトラッキング情報の値に反映されない．現在のアクティビティのアテンプトが終了する．AbandonAll の場合，シーケンシングセッションを終了する．
- SupendAll の場合．現在のアクティビティおよびそのすべての祖先のアクティビティのアテンプトが中断される．中断したアテンプトは，以降の ResumeAll シーケンシング要求で再開できる．このアテンプトは新たなアテンプトではなく中断されたアテンプトの継続である．

終了要求が Exit の場合は，終了プロセスでは以下のような動作が実行される．

(1) 現在のアクティビティを終了し，実行中の SCO からのランタイム環境情報の値を，アクティビティのトラッキング情報の値に反映する．アクティビティに付随する学習リソースがアセットでランタイム環境情報の値を返さない場合, LMS がデフォルトのトラッキング情報の値を設定する．

(2) 次節に述べるロールアップを実行する．

(3) 終了ルール，ポストコンディションルールの評価を行う．Exit Parent，Exit All などのルールで，親のアクティビティが終了する場合は，親の終了ルール，ポストコンディションルールを再帰的に評価する．また，Continue，Previous，Retry などのルールでシーケンシング要求が発生する場合は，このシーケンシング要求でナビゲーションプロセスから得られたシーケンシング要求を置き換える．

6.1.3 ロールアッププロセス

ロールアッププロセスでは，習得度ロールアップ，学習目標ロールアップ，進捗状態ロールアップ，の三つの動作が行われる．個々の処理の詳細は 3.4.5 を参照のこと．

(1) 習得度ロールアップでは，親アクティビティの主学習目標の習得度を，子アクティビティの主学習目標の習得度の重みつき平均で決定する．

(2) 学習目標ロールアップでは，親アクティビティの主学習目標の習得状態を以下のいずれかの方法で決定する．1)を適用した場合は 2)以降は実行しない．また 2)を適用した場合は 3)は実行しない．

1) 習得度ロールアップで算出した学習目標習得度と，予め設定したしきい値を比較する．
2) ロールアップルールを適用する．
3) デフォルトロールアップルールを適用する．

(3) 進捗状態ロールアップでは，親アクティビティのアテンプト完了状態を以下のいずれかの方法で決定する．1)を適用した場合は 2)は実行しない．

1) ロールアップルールを適用する．
2) デフォルトロールアップルールを適用する．

6.1.4 選択ランダム化プロセス

選択ランダム化プロセスでは，クラスタ中の子アクティビティのうちのいくつの子アクティビティを学習者に提示するか，提示する順番をどうするか，という決定が行われる．

(1) 選択プロセスでは，クラスタ中の複数の子アクティビティから，指定した個数の子アクテ

ィビティを学習者に提示する子アクティビティとして選択する．選択を行うタイミングは，当該のクラスタのアテンプトを最初に実行する時点となる．

(2) ランダム化プロセスでは，クラスタ中の複数の子アクティビティの提示順序をランダムに並べ替える．並べ替えは，当該のクラスタのアテンプトを最初に実行する時点か，毎回アテンプトを実行するたびに行うかを選択することができる．

6.1.5 シーケンシングプロセス

シーケンシングプロセスでは，ナビゲーションプロセスからシーケンシング要求を受け取り，次に学習者に提示するアクティビティを決定する．シーケンシングプロセスは，シーケンシング要求に対応した複数のプロセスから構成される．各プロセスでは，シーケンシング制御モード，制限条件，プリコンディションルールを参照しながら，提示するアクティビティの決定が行われる．

6.1.5.1 アクティビティ決定処理の概要

提示するアクティビティの決定処理は，シーケンシング要求の内容によっていくつかに分類できる．

(1) 学習の開始：Start，Resume All，Choice

これらのシーケンシング要求は，現在のアクティビティが確定していない状態から，学習を始めるために実行される．Start では，後に述べる Flow サブプロセスによって，アクティビティツリーの根のアクティビティからトラバースを行い，提示するアクティビティを決定する．Resume All では，前回学習を中断したときのアクティビティから学習を再開する．Choice では，指定されたアクティビティから学習を開始する．

(2) ツリーのトラバース：Continue，Previous，Choice

これらのシーケンシング要求は，現在のアクティビティからツリー中を前後に移動して，提示するアクティビティを決定する．Continue ではツリーを前方に，Previous では後方に移動を行う．移動に際しては，後に述べる Flow サブプロセスを適用する．Choice では指定されたアクティビティに向かって移動する．もし，末端アクティビティでないアクティビティが指定された場合は，指定されたアクティビティからさらに Flow サブプロセスを適用して提示するアクティビティを決定する．

(3) 繰り返し：Retry

Retry シーケンシング要求は，現在のアクティビティを再度実行する．アクティビティが末端アクティビティで無い場合は，Flow サブプロセスを適用して提示するアクティビティを決定する．

(4) 終了：Exit

アクティビティのアテンプトを終了する．もしアクティビティがアクティビティツリーの根のアクティビティの場合，シーケンシングセッションを終了する．

6.1.5.2 Flow サブプロセスの概要

Flow サブプロセスでは，アクティビティツリーをあるアクティビティから，前方ないし後方

に移動して提示するアクティビティを決定する．提示対象となるのはアクティビティツリーの末端のアクティビティであり，末端のアクティビティに到達できなかった場合，Flow サブプロセスはエラーを返す．以下に Flow サブプロセスの動作を示す．

(1) 現アクティビティから指定された方向にひとつ移動し，これを候補アクティビティとする．
(2) 候補アクティビティの親アクティビティのシーケンシング制御モード Flow が偽ならエラー．処理を終了する．
(3) 候補アクティビティの skip プリコンディションルールが成り立てば，指定された方向にさらにひとつ移動し，これを候補アクティビティとする．(2)へ戻る．
(4) disable プリコンディションルールか，制限条件が成り立っていればエラー．処理を終了する．
(5) 候補アクティビティが末端アクティビティなら，これを配信対象と確定する．処理を終了する．
(6) 候補アクティビティが末端アクティビティでなければ，クラスタに入って候補アクティビティを決める．
    前方への移動の場合，クラスタの最初の子アクティビティを候補アクティビティとする．
    後方への移動でシーケンシング制御モード Forward Only が偽の場合，クラスタの最後の子アクティビティを候補アクティビティとする．
    後方への移動でシーケンシング制御モード Forward Only が真の場合，クラスタの最初の子アクティビティを候補アクティビティとする．

6.1.6 配信プロセス

配信プロセスでは，シーケンシングプロセスで決定された配信候補アクティビティ（配信要求）を受け取り，クライアントへの配信処理を行う．配信プロセスでは，配信候補アクティビティおよびそのすべての祖先のアクティビティが，制限条件を冒していないかどうかを確認する．そして，対象のアクティビティのアテンプトを開始し，アテンプトに関する実行時間などの記録を開始する．

なお，実際の SCO，アセットなどの学習資源のクライアントへの配信はランタイム環境の起動処理によって実行される．

6.2 擬似コード

SCORM 2004 では，擬似コードによってシーケンシング動作を構成する各プロセスの動作を厳密に規定している．ここでは，各プロセスの呼び出し関係と処理の概略を整理する．

表 6.1 と表 6.2 にプロセスの呼び出し関係を示す．表 6.1 は Overall Sequencing Process を主体として，そこから呼び出されるプロセスの関係を記述したものである．この中で，複数個所から呼び出されているプロセスは太字で示してあり，その詳細が表 6.2 に示されている．プロセス名のあとのカギカッコ中の記号は，規格書の擬似コードで用いられているプロセスの識別コードである．

以下に主要なプロセスの概略を説明する．

表 6.1 擬似コードプロセスの呼び出し関係

Overall Sequencing Process [OP.1]
Navigation Request Process [NB.2.1]
Termination Request Process [TB.2.3]
End Attempt Process [UP.4]
Sequencing Exit Action Rules Subprocess [TB.2.1]
Sequencing Rules Check Process [UP.2]
Terminate Descendent Attempts Process [UP.3]
End Attempt Process [UP.4]
Sequencing Post Condition Rules Subprocess [TB.2.2]
Sequencing Rules Check Process [UP.2]
Terminate Descendent Attempts Process [UP.3]
Select Children Process [SR.1]
Randomize Children Process [SR.2]
Sequencing Request Process [SB.2.12]
Start Sequencing Request Process [SB.2.5]
Flow Subprocess [SB.2.3]
Resume All Sequencing Request Process [SB.2.6]
Exit Sequencing Request Process [SB.2.11]
Retry Sequencing Request Process [SB.2.10]
Flow Subprocess [SB.2.3]
Continue Sequencing Request Process [SB.2.7]
Flow Subprocess [SB.2.3]
Previous Sequencing Request Process [SB.2.8]
Flow Subprocess [SB.2.3]
Choice Sequencing Request Process [SB.2.9]
Sequencing Rules Check Process [UP.2]
Choice Activity Traversal Subprocess [SB.2.4]
Sequencing Rules Check Process [UP.2]
Choice Flow Subprocess [SB.2.9.1]
Choice Flow Tree Traversal Subprocess [SB.2.9.2]
Choice Flow Tree Traversal Subprocess [SB.2.9.2] REC
Flow Subprocess [SB.2.3]
Terminate Descendent Attempts Process [UP.3]
Delivery Request Process [DB.1.1]
Check Activity Process [UP.5]
Content Delivery Environment Process [DB.2]
Clear Suspended Activity Subprocess [DB.2.1]
Terminate Descendent Attempts Process [UP.3]

6.2.1 Overall Sequensing Process

シーケンシング動作全体に相当するプロセスである．6.1 **に述べた各処理に該当するプロセスを順次呼び出す．すなわち，**Navigation Request Process, Termination Request Process, Select Children Process, Randomize Children Process, Sequencing Request Process, Delivery Request Process, Content Delivery Environment Process **を順次呼び出して，シーケンシングの一連の動作を実現する．**

表 6.2 擬似コードプロセスの呼び出し関係（補助プロセス）

Flow Subprocess [SB.2.3]
Flow Tree Traversal Subprocess [SB.2.1]
Flow Tree Traversal Subprocess [SB.2.1] REC
Flow Activity Traversal Subprocess [SB.2.2]
Sequencing Rules Check Process [UP.2]
Flow Tree Traversal Subprocess [SB.2.1]
Flow Activity Traversal Subprocess [SB.2.2] REC
Check Activity Process [UP.5]

Terminate Descendent Attempts Process [UP.3]
End Attempt Process [UP.4]

End Attempt Process [UP.4]
Overall Rollup Process [RB.1.5]
Measure Rollup Process [RB.1.1]
Objective Rollup Process [RB.1.2]
Objective Rollup Using Measure Process [RB.1.2 a]
Objective Rollup Using Rules Process [RB.1.2 b]
Rollup Rule Check Subprocess [RB.1.4]
Activity Progress Rollup Process [RB.1.3]
Rollup Rule Check Subprocess [RB.1.4]

Rollup Rule Check Subprocess [RB.1.4]
Check Child for Rollup Subprocess [RB.1.4.2]
Sequencing Rules Check Process [UP.2]
Evaluate Rollup Conditions Subprocess [RB.1.4.1]

Check Activity Process [UP.5]
Sequencing Rules Check Process [UP.2]
Limit Conditions Check Process [UP.1]

Sequencing Rules Check Process [UP.2]
Sequencing Rule Check Subprocess [UP.2.1]

6.2.2 Termination Request Process

終了要求の処理を行う．アクティビティの終了処理とロールアップを行う End Attempt Process，終了ルールの処理を行う Sequencing Exit Action Rules Subprocess，ポストコンディションルールの処理を行う Sequencing Post Condition Rules Subprocess，現在のアクティビティの祖先のアクティビティの終了処理を行う Terminate Descendent Attempts Process から構成される．

6.2.3 Sequencing Request Process

受け取ったシーケンシング要求に応じて，シーケンシング要求の種別に応じた各プロセス，すなわち，Start, Resume All, Exit, Retry, Continue, Previous, Choice のいずれかの Sequencing Request Process が呼び出される．これらのうち，Start, Retry, Continue, Previous のプロセスでは Flow Subprocess を用いてアクティビティツリーのトラバーサルを行って提示するアクティ

ビティを決定する．

Choice の場合は，現在のアクティビティと移動先として指定されたアクティビティのアクティビティツリー中の関係によって独自のトラバーサル処理を行って指定されたアクティビティが提示可能かどうかを決定する．

6.2.4 Flow Subprocess

現在のアクティビティから前方ないし後方にツリートラバーサルを試み，提示候補のアクティビティを決定する．内部は，Flow Tree Traversal Subprocess と Flow Activity Traersal Subprocess が再帰的に呼び出される構成になっている．Flow Tree Traersal Subprocess ではプリコンディションルールなどのチェックは行わず，現在のアクティビティと指定された移動方向から，次のアクティビティを決定する．Flow Activity Traersal Subprocess では，Flow Tree Traersal Subprocess で得られたアクティビティに対してプリコンディションルールなどのチェックを行い，実際に配信が可能かどうかをチェックする．また，Skip プリコンディションルールが成り立っているような場合は，Flow Tree Traersal Subprocess と Flow Activity Traersal Subprocess をさらに再帰的に適用してツリーをトラバースする．

6.2.5 End Attempt Process

アクティビティの終了処理を行う．具体的には，アクティビティが葉ノードで，アクティビティに関連付けられた学習資源が SCO で無い場合は，デフォルトのトラッキング情報を設定する．また，Overall Rollup Process を呼び出してロールアップを実行する．Overall Rollup Process では，Measure Rollup Process, Objective Rollup Process, Activity Progress Rollup Process によってそれぞれ，習得度ロールアップ，学習目標ロールアップ，進捗状態ロールアップを行う．

6.2.6 Check Activity Process

アクティビティの制限条件，Disable プリコンディションルールなどのチェックを行い，アクティビティが配信可能がどうかをチェックする．

6.2.7 Sequencing Rules Check Process

シーケンシングルールの条件節が成立するか否かのチェックを行う．

7. ランタイム環境の実現

5.で述べたように，SCORM 規格では，LMS と SCO が通信を行うためのランタイム環境を規定している．本節では，主に LMS を実装する観点から，ランタイム環境を実装する方法，および，その際の注意事項について述べる．初めに，SCO の起動と終了の実装の具体的な方法を示す．また，API アダプタの実装，SCORM 1.2 と SCORM 2004 の両方の SCO に対応するための方法を示す．さらに，RTE データモデルの各要素の実装上の注意，アテンプトとの関連，中断再開時のデータ保持などについて述べる．また，シーケンシングのトラッキング情報との関係について述べる．最後に，ナビゲーション機能の実装上の注意について述べる．

7.1 起動

起動処理において，LMS はクライアントに配信する学習資源を決定して起動する．配信される学習資源が SCO の場合，SCO は LMS が提供する API インスタンスを検索して，以降の LMS との通信を行う．ここでは，このような一連の流れを実行するための LMS の実装例を示す．

図 7.1 に SCORM 準拠 LMS で学習を実行している際の，ブラウザの画面例を示す．この例ではブラウザを 3 つのフレームに分割している．最上段のメニューフレームには，ナビゲーション要求を発生するためのコマンドボタンを配置している．このフレームの表示内容は LMS によって異なる．２つ目の SCO フレームには学習資源が表示される．このフレームの表示内容は学習資源そのものである．3 つ目の API フレームは API インスタンスを格納するためのもので，通常は大きさを 0 にして，学習者に見えないようにしてある．

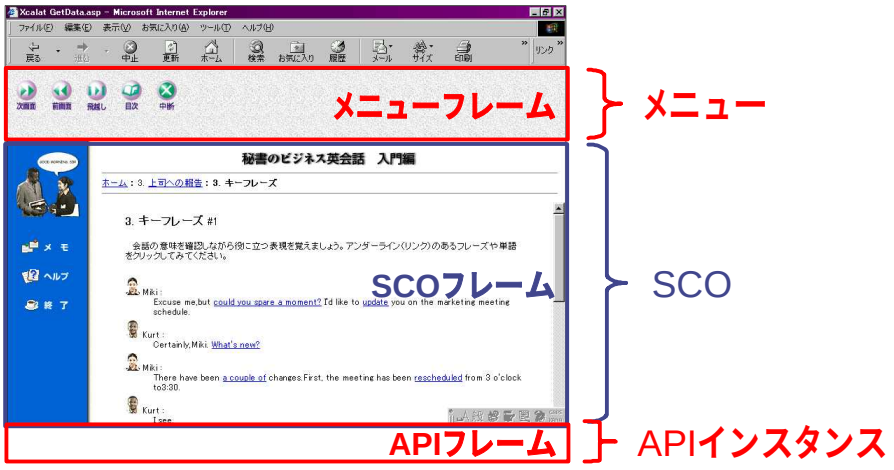

図 7.1 クライアントのフレーム構成

上述した起動処理では，LMS はこのような 3 つのフレームを定義した HTML ファイルをブラウザに送り込むことで，API インスタンスをブラウザに配置し，学習資源をブラウザに読み込ませる，という処理を実現する．

このような HTML ファイルの例を以下に示す．

```
<HTML>
<HEAD>
<TITLE>LMS system</TITLE>

<SCRIPT LANGUAGE ="JavaScript">
    function myOnLoad( url )      ------------------------------------------------------(1)
    { self.Main.location = url; }
</SCRIPT></HEAD>

<FRAMESET ROWS="100%,0%" onLoad = myOnLoad("SCO URL")> ------------------------------(2)
    <FRAMESET ROWS="43,*">        ------------------------------------------------------(3)
        <FRAME NAME="Menu" SRC="MENU GENERATION URL">
        <FRAME NAME="Main" SRC="about:blank"> ---------------------------------------(4)
    </FRAMESET>
    <FRAME NAME="API_1484_11" SRC="API ADAPTER URL API.html">   ---------------------(5)
</FRAMESET>

</HTML>
```

(2)と(3)でフレームセットを定義しており，メニューフレーム（名称 Menu），SCO フレーム（名称 Main），API インスタンスフレーム（名称 API_1484_11）を定義している．メニューフレームと SCO フレームを一段深いフレームセットに定義しているのは，5.3.3 で述べたように SCO フレームから親のフレームをたどって API インスタンスフレームが見つかるようにするためである．

このHTML定義のポイントは，(4)で最初にSCOフレームをブランクにしておき，(2)のOnLoadイベントを使って，(1)の myOnLoad 関数を呼び出して実際の SCO を読み込んでいるところである．このような定義をせずに，(4)に直接 SCO の URL を記述すると，(4)の SCO と(5)の API インスタンスのどちらが先にブラウザに読み込まれるかが保証されない．仮に SCO が先に読み込まれて API インスタンスの検索を開始してしまうと，API インスタンスがまだ読み込まれておら

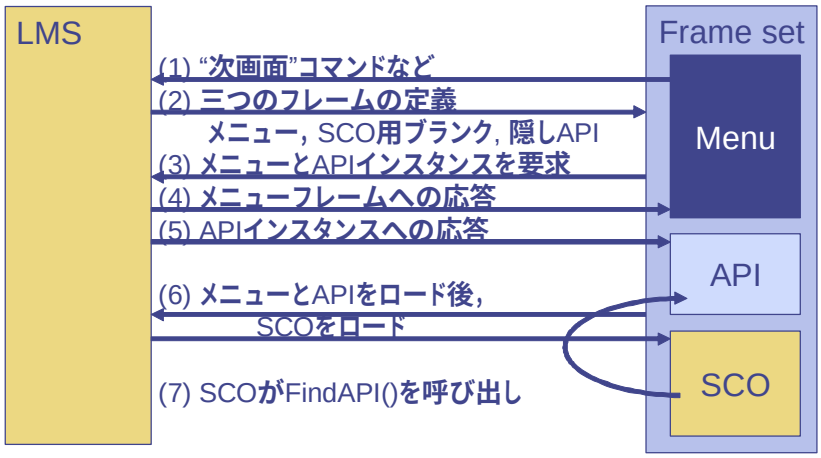

図 7.2 SCO 起動時のタイミング

ず，検索に失敗する可能性がある．このトラブルはタイミングの問題なので，ブラウザの種別など環境によって発生したりしなかったりする可能性があり，非常に見つけにくい．これを防ぐため，上記の HTML では，(2)で OnLoad イベントを使って(1)の myOnLoad 関数を呼び出し，これによって実際の SCO を読み込んでいる．OnLoad イベントは，フレームセットの読み込みが完了してから発生するので，必ず(5)の API インスタンスが読み込まれてから(1)で SCO が読み込まれることになる．時間的な関係を図 7.2 に示す．

7.2 API インスタンスの実装

API インスタンスは DOM オブジェクトとして実装される．この DOM オブジェクトでは JavaScript で API 関数の定義が行われている．API インスタンスに要求されるのは，規格で規定された API 関数によってデータモデル要素の通信を実行することだけで，それ以外の実装に関する事項，例えば，データモデル要素をクライアント側で保持するか，サーバ側で保持するか，どのタイミングでサーバ・クライアント間の通信を行うか，といったことはすべて LMS 設計者に任されている．一般に，処理を単純にするのであればクライアント側のモジュールでは処理を行わず，サーバ側ですべての処理を行うのが簡単であろう．一方，性能的な面に配慮するのであれば，クライアント側で極力データモデルの管理を行うほうが，通信量，サーバの処理量の面で有利になる．

SCORM 2004 では API インスタンスの名称は ”API_1484_11” とするように規格で決められている．一方，SCORM 1.2 では API インスタンス（API アダプタ）の名称は ”API” であった．この違いを利用すると，SCORM 2004 LMS 環境で，SCORM 1.2 と SCORM 2004 の両方の SCO を扱うことができるようになる．これを図 7.3 に示す．LMS 側では，”API_1484_11” と ”API” という二つの API インスタンスを用意しておく．SCORM 2004 SCO は前者の，SCORM 1.2 SCO は後者の API インスタンスを使用するので，SCO からは SCORM 2004 と SCORM 1.2 の API 関数名やデータモデルの相違を気にする必要はなくなる．SCORM 1.2 API インスタンスでは，

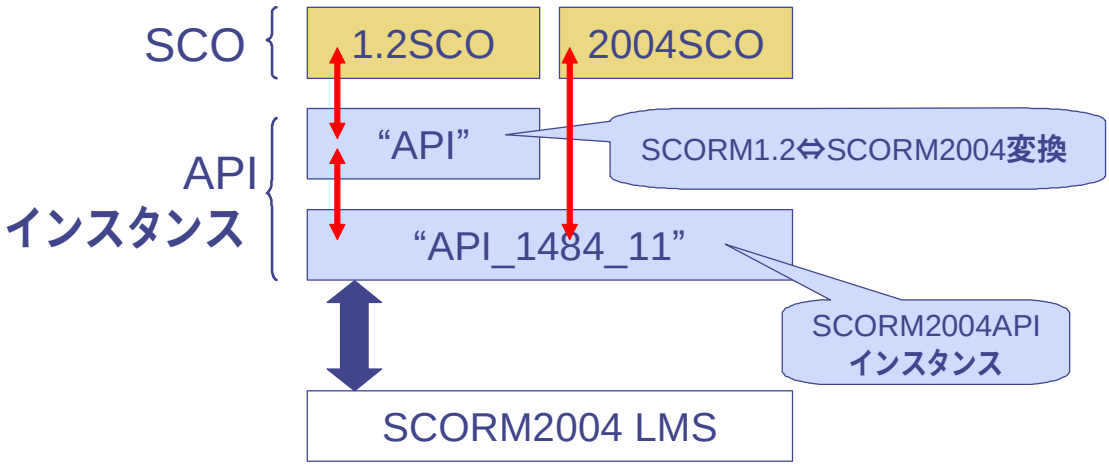

図 7.3 SCORM 1.2 SCO と SCORM 2004 SCO の混在

SCORM 1.2 のデータモデルを SCORM 2004 データモデルに変換するようにしておく．これについては 8.で詳しく述べる．

7.3 データモデルの実装

RTE で提供されているデータモデルの要素は，SCO から見て「読み出しのみ」,「書き込みのみ」,「読み書き」，という通信の方向を有する．このうち,「読み出しのみ」のデータ要素は，LMS が管理データやマニフェストファイルの設定値から初期設定を行い，それを SCO が利用するというものが多い．従って，LMS はこれらの初期値設定を行う必要がある．また，現在利用されておらず，値が固定されているデータモデル要素もある．「読み書き」のデータモデル要素については，LMS は SCO が書き込んだ値を保持しておき，読み出し時にそれを返却すればよいが，データモデル要素によってはシーケンシングのトラッキングモデルと関連しているものもあり，注意が必要である．ここではこのような実装に注意が必要なデータモデルについて解説する．

7.3.1 LMS の管理情報から設定するデータモデル要素

cmi.learner_id, cmi.learner_name は読み出しのみのデータモデル要素であり，LMS の管理情報から設定する．通常，LMS では学習者 ID，学習者名を管理しているので，それらの値を SCO が読み出せるように設定する．

7.3.2 Manfest ファイルから設定するデータモデル要素

cmi.completion_threshold, cmi.launch_data, cmi.max_time_allowed, cmi.scaled_passing_score, cmi.time_limit_action は読み出しのみのデータモデル要素であり，それぞれ対応するマニフェストファイルのデータから値を設定する．表 7.1 にこれらの対応を示す．マニフェストファイルに値が定義されていない場合は，SCO がデータを読み出すと“403 – Data Model Element Value Not Initialized”エラーとなる．

表 7.1 Manifest ファイルから設定するデータモデル要素

データモデル要素名	説明
Manifestfile 要素名	
cmi.completion_threshold	SCO を完了したとみなすために必要な進捗度
<adlcp:completionThreshold>	
cmi.launch_data	SCO 起動時の初期化パラメータ
<adlcp:dataFromLMS>	
cmi.max_time_allowed	SCO の最大許容実行時間
<imsss:attemptAbsoluteDurationLimit>	
cmi.scaled_passing_score	SCO を習得したとみなすために必要な正規化得点
<imsss:minNormalizedMeasure>	
cmi.time_limit.action	SCO の実行時間が Maximum Time Allowed に達した際の SCO の動作を規定する．
<adlcp: timeLimitAction>	

7.3.3 値が固定されているデータモデル要素

cmi.credit, cmi.mode については，読み出しのみのデータモデル要素として実装が義務付けられているが，用途は規格の範囲外となっている．値はそれぞれ “credit”, “normal” に固定しておけばよい．

7.3.4 読み出しのみと書き込みのみの要素が関連するデータモデル要素

cmi.session_time と cmi.total_time，cmi.exit と cmi.entry は，それぞれ書き込みのみと読み出しのみのデータモデル要素であるが，書き込みのみの要素の値によって読み出しのみの要素の値が影響を受ける．

cmi.session_time は SCO を起動してからのセッションの経過時間を示す．cmi.total_time は SCO の総実行時間で各起動セッションの実行時間の合計である．LMS は SCO を起動してから SCO が終了するまでの間で，最後に受け取った cmi.session_time を cmi.total_time に足しこむ．SCO 実行中に途中で受け取った cmi.session_time は破棄する．cmi.total_time の初期値は 0 である．

cmi.exit は SCO の終了時の状態を表し，cmi.entry は次回 SCO が起動したときに前回の終了状態を通知する．初期状態では，cmi.entry は “ab_initio” である．cmi.exit が “suspend”, “logout” だった場合，cmi.entry は “resume” となる．cmi.exit が “normal”, “time-out”, “” だった場合，cmi.entry は “” となる．

cmi.total_time と cmi.entry は中断 (Suspend All) 後の再開 (Resume All) 時に中断前の値を引き継ぐ必要がある．従って，LMS は，SCO ごとに，中断時にファイルなどに cmi.total_time と cmi.entry の値を保存し，再開時に値を復元する必要がある．

7.3.5 格納・再設定が必要なデータモデル要素

cmi.location, cmi.suspend_data, cmi.interactions, cmi.learner_preference, cmi.score, cmi.progress_measure, cmi.objectives, cmi.completion_status, cmi.success_status は読み書きのデータモデル要素である．これらは，SCO の書き込んだ値を，あとで読み出せるようになっていなくてはならない．学習を中断した場合は，中断前の値を再開後に読み出せなくてはならないので，LMS では中断時の値をファイルなどに保存しておく必要がある．

cmi.location, cmi.suspend_data, cmi.interactions については，SCO の書き込んだ値があとでそのまま読み出せるようになっている必要がある．LMS はこれらの値を一切加工しない．SCO が値を書き込む前に読み出しを行った場合，“403 – Data Model Element Value Not Initialized” エラーとなる．

cmi.learner_preference は，学習者の使用する言語，スピーカの音量，などを設定する読み書き可能なデータモデル要素である．これらの値は，LMS が管理情報から初期値を設定し，学習中に学習者が書き換えることができる．LMS がこのような情報を管理していない場合は，データモデル要素ごとのデフォルト値が設定される．

cmi.score, cmi.progress_measure, cmi.objectives についても，SCO の書き込んだ値があとでそのまま読み出せるようになっている必要がある．LMS はこれらの値を一切加工しない．SCO が値を書き込む前に読み出しを行った場合，“403 – Data Model Element Value Not Initialized”

エラーとなる.

cmi.completion_status, cmi.success_status は，SCO から送信される値をそのまま用いるのが基本で初期値は Unknown だが，LMS が対応する得点 (cmi.progress_measure, cmi.score.scaled) とマニフェストファイルに設定されたしきい値 (adlcp:completionThreshold, imsss:minNormalizedMeasure) との比較から決定する場合もある．表 7.2，表 7.3 に

表 7.2　Completion Status の決定

Manifest の adlcp:completionThreshold = cmi.completion_threshold	SCO が設定する cmi.progress_measure	SCO が設定する cmi.completion_status	LMS が設定する cmi.completion_status
未定	未定	未定	unknown
未定	未定	定義語彙（注）	SCO 設定値
未定	0.5	未定	unknown
未定	0.5	定義語彙	SCO 設定値
0.8	未定	未定	unknown
0.8	未定	定義語彙	SCO 設定値
0.8	0.5	未定	incomplete
0.8	0.5	定義語彙	incomplete
0.8	0.9	未定	completed
0.8	0.9	定義語彙	completed

（注）completed, incomplete, not attempted のいずれか

表 7.3　Success Status の決定

Manifest の imsss:minNormalizedMeasure = cmi.scaled_passing_score	SCO が設定する cmi.score.scaled	SCO が設定する cmi.success_status	LMS が設定する cmi.success_status
未定	未定	未定	unknown
未定	未定	定義語彙（注）	SCO 設定値
未定	0.5	未定	unknown
未定	0.5	定義語彙	SCO 設定値
0.8	未定	未定	unknown
0.8	未定	定義語彙	SCO 設定値
0.8	0.5	未定	failed
0.8	0.5	定義語彙	failed
0.8	0.9	未定	passed
0.8	0.9	定義語彙	passed

（注）passed, failed のいずれか

cmi.completion_status, cmi.success_status の決定ルールを示す． cmi.completion_status 設定の基本的な考え方は，以下の通りである．

- cmi.progress_measure が cmi.completion_threshold より大きいか等しければ completed. SCO の設定する cmi.completion_status の値は無視される．
- cmi.progress_measure が cmi.completion_threshold より小さければ incomplete. SCO の設定する cmi.completion_status の値は無視される．
- cmi.progress_measure か cmi.completion_threshold の一方でも未定の場合は，SCO の設定する cmi.completion_status の値を最終的な値とする．

cmi.success_status の決定の考え方も同様で，cmi.score.scaled と cmi.scaled_passing_score の比較が優先され，いずれか一方でも未定の場合は，SCO からの設定値が採用される．

これの動作を用いて，同一の SCO を複数のアクティビティに関連付け，マニフェストファイル中でアクティビティによって異なる合格点を設定することで，合格の条件を変えることができる．

なお，cmi.completion_status, cmi.success_status, cmi.score.scaled, cmi.progress_measure, cmi.objectives.n.success_status, cmi.objectives.n.score.scaled はシーケンシング機能のトラッキング情報と関係する．これについては，次節で説明する．

7.4 トラッキング情報と RTE データモデルの対応

3.2 で述べたトラッキング情報のうち，末端のアクティビティの完了に関する情報，および，学習目標の習得に関する情報は，SCO からの RTE のデータモデル要素の値を使って設定される．アクティビティに関連する学習資源が SCO でなくアセットの場合は LMS がデフォルト値を設定する．

7.4.1 アクティビティの完了に関する情報の設定

表 7.4 にアクティビティのアテンプトの完了に関するトラッキング情報と RTE データモデル要素の関係を示す．

アクティビティの完了を示すアテンプト完了状態は cmi.completion_status の値によって決まる．cmi.completion_status の値は，SCO から送信される値が用いられるのが基本だが，前節で述べたように cmi.progress_measure と cmi.completion_threshold の比較で LMS が決定する場合もあり，注意が必要である．

アクティビティに対応付けられた学習資源が，SCO でなくアセットの場合は，cmi.completion_status の値が設定されないので，LMS が自動的にアテンプト完了状態を「完了」にする場合がある．LMS がこの動作を行う条件は，以下の通りである．

- アクティビティの Tracked が True，かつ，
- アクティビティの Completion Set by Content が False，かつ，
- cmi.completion_status が設定されていない．

アテンプト完了度，および，cmi.progress_measure は，データ要素としては定義されているが，現在のところはシーケンシングで使用されておらず，両者の対応も規定されていない．

表 7.4 アテンプトの完了に関するトラッキング情報と RTE データモデル要素の関係

RTE データモデル要素		トラッキング情報	
cmi.completion_status		アテンプト完了状態	
	completed		完了（アセットの場合の設定値）
	incomplete		未完了
	not attempted		未完了
	unknown		未定（初期値）
注： cmi.completion_status の決定法については表 7.2 参照			
cmi.progress_measure		アテンプト完了度	
	[0..1]		[0..1]
	unknown		未定（初期値）

注： SCORM 2004 では Attempt Completion Amount はシーケンシングに使用されず，Progress Measure との対応も規定されていない

7.4.2 主学習目標の習得に関する情報の設定

表 7.5 に主学習目標の習得に関するトラッキング情報と RTE データモデル要素の関係を示す．

学習目標の習得に関する情報は，学習目標習得状態，学習目標習得度で管理される．主学習目標の場合，これらの値は，cmi.success_status, cmi.score.scaled で決定される．これらは SCO から送信される値をそのまま用いるのが基本だが，cmi.success_status については，表 7.3 に示すように cmi.score.scaled と cmi.scaled_passing_score の比較から LMS が決定する場合もある．

アクティビティに対応付けられた学習資源が，SCO でなくアセットの場合は，cmi.success_status の値が設定されないので，LMS が自動的に学習目標習得状態を「習得」にする場合がある．LMS がこの動作を行う条件は，以下の通りである．

- アクティビティの Tracked が True，かつ，
- アクティビティの Objective Set by Content が False，かつ，
- cmi.success_status が設定されていない．

表 7.5 主学習目標の習得に関するトラッキング情報と RTE データモデル要素の関係

RTE データモデル要素		トラッキング情報	
cmi.success_status		学習目標習得状態	
	passed		習得（アセットの場合の設定値）
	failed		未習得
	unknown		未定（初期値）
cmi.score.scaled		学習目標習得度	
	[-1..1]		[-1..1]
	unknown		未定（初期値）

注： cmi.success_status の決定法については表 7.3 参照

7.4.3 主学習目標の以外の学習目標の習得に関する情報

その他の学習目標については，学習目標習得状態は cmi.objectives.n.success_status，学習目標習得度は cmi.objectives.n.score.scaled から決定される．このとき，cmi.objectives.n.success_status と cmi.objectives.n.score.scaled の間には主学習目標に対する cmi.success_status, cmi.score.scaled のような依存関係はない．また，アセットの場合のような LMS による自動設定も行われない．SCO 側で読み出される cmi.objectives.n.id はアクティビティのローカル学習目標に付与された ID(Activity Objective ID)である．

ローカル学習目標が，共有グローバル学習目標からデータを読み出すように関連付けられている場合，SCO が読み出す cmi.objectives.n.success_status と cmi.objectives.n.score.scaled の値は，共有グローバル学習目標の学習目標習得状態と学習目標習得度の値となる．逆に，ローカル学習目標が，共有グローバル学習目標にデータを書き込むように関連付けられている場合，SCO が書き込む cmi.objectives.n.success_status と cmi.objectives.n.score.scaled の値は，共有グローバル学習目標の学習目標習得状態と学習目標習得度に設定される．

表 7.6 主学習目標以外の学習目標の習得に関する トラッキング情報と RTE データモデル要素の関係

RTE データモデル要素	トラッキング情報
cmi.objectives.n.success_status	学習目標習得状態
passed	習得
failed	未習得
unknown	未定（初期値）
cmi.objectives.n.score.scaled	学習目標習得度
[-1..1]	[-1..1]
unknown	未定（初期値）
cmi.objectives.n.id	Activity Objective ID
Universal Resource Identifier	Unique Identifier

7.5 ナビゲーション機能の実装

4.で述べたように SCORM 2004 では SCO がナビゲーションイベントを発行できるようになった．ナビゲーションイベントは SCO が RTE の API 関数を用いて SetValue(“adl.nav.request”, REQUEST) を実行することで設定される．このあと，SCO が Terminate(“”) を実行すると，LMS 側では REQUEST で指定したナビゲーションリクエストが実行される． SCO は SetValue(“adl.nav.request”, REQUEST) を何回も呼び出すことができる．LMS は，Terminate(“”) API 関数が実行される直前に設定された REQUEST を採用し，それ以前の REQUEST は廃棄する．REQUEST は “continue”, “previous”, “choice”, “exit”, “exitAll”, “abandon”, “abandonAll”, “_none_” のいずれかである．“_none_” が最後に設定された場合，LMS は何も実行しない．

ナビゲーションイベントは，SCO 以外に，SCORM 1.2 と同様 LMS が設けたナビゲーション GUI からも発生することができる．このため，SCO が発生するナビゲーションイベントと LMS が設けたナビゲーション GUI からのナビゲーションイベントが競合する可能性がある．このような場合は，LMS は常に LMS のナビゲーション GUI からのナビゲーションイベントを優先して採用する．

例えば，SCO が SetValue(“adl.nav.request”, “continue”) を実行したとする．その後，学習者が LMS の目次機能を使って “choice” を実行して別の SCO を選んだとする．この場合，新しい SCO がブラウザにロードされるので，古い SCO はアンロードされる．通常 SCO はアンロードイベントで Terminate(“”) API 関数を実行するようになっているので，先に SCO が設定した “continue” ナビゲーションイベントと学習者が発行した “choice” ナビゲーションイベントが競合する形になる．このような場合，LMS は学習者が発行したナビゲーションイベントを優先し，“continue” ナビゲーションイベントは廃棄する．

8. SCORM 1.2 から 2004 への移行

SCORM 2004 では，SCORM 1.2 と比較して，シーケンシング機能，ナビゲーション機能が追加になったほか，細かな変更がいくつか行われている．本節では，SCORM 1.2 と SCORM 2004 の相違点，コンテンツの移行のための具体的な方法について述べる．

8.1 マニフェストファイルの相違点と移行

SCORM 2004 のマニフェストファイルは，基本的に SCORM 1.2 のマニフェストファイルの上位互換となっている．SCORM 1.2 のマニフェストファイルはシーケンシングルールが全く設定されていない SCORM 2004 のマニフェストファイルとして扱われる．

しかし，SCORM 1.2 で規定されていた以下の要素は SCORM 2004 で廃止されており，移行にあたって注意が必要である．

- adlcp:prerequisites （前提条件）
- adlcp:masteryscore （マスタリスコア）
- adlcp:maxtimeallowed （許容時間）

前提条件は，あるアクティビティが学習可能となるための条件で，他のアクティビティの完了ないし習得状態を要素とする論理式で表現される．SCORM 1.2 では，このような前提条件の表現はできるものの，条件の成否によって LMS がどのように動作すればよいかが規定されておらず，相互運用性上の問題となっていた．このように動作が不明確であったため，SCORM 2004 に移行する際には，コンテンツ作成者が意図する動作をシーケンシングルールによって書き直す必要がある．

マスタリスコア，許容時間は，SCORM 2004 では，それぞれ，imsss:minNormalizedMeasure, imsss:attemptAbsoluteDurationLimit となった．これらの値は，SCORM 1.2 の場合と同様，実行時に RTE を通して SCO からも参照することができる．対応するデータモデル要素名は，cmi.max_time_allowed, cmi.scaled_passing_score である．

前提条件，マスタリスコア，許容時間，を使用していない SCORM 1.2 のマニフェストファイルはそのまま SCORM 2004 に対応した LMS で実行可能である．また，このようなマニフェストファイルに対して必要に応じてシーケンシングルールを追加して，SCORM 2004 の機能を活用することができる．このようにマニフェストファイルの移行には，それほど大きなコストは発生しない．

8.2 RTE の相違点と移行

5. に述べたように RTE については，API インスタンスの名称，API 関数の名称，データモデルの変更などが行われ，互換性は保たれていない．このため，SCORM 1.2 用に作成した SCO を SCORM 2004 環境で直接使用するためには，変更に対応した修正を SCO に施す必要がある．このような修正は非常に大幅なものとなる可能性があり，移行のためのネックとなる．

このようなネックを解消するための LMS 側での対応を以下に述べる．また，当面は

SCORM 1.2 対応の LMS と SCORM 2004 対応の LMS が並存する状況が続くと考えられる．このような状況に SCO 側で対応するための方策も述べる．

8.2.1 RTE 移行のための LMS の対応

SCORM 1.2 用に作成した SCO を SCORM 2004 環境で使用するためには，LMS 側で 7.2 に述べた API インスタンス名の違いを利用する．7.2 で述べたように，SCORM 1.2 の API インスタンスの DOM オブジェクト名は “API”，一方，SCORM 2004 の API インスタンスの DOM オブジェクト名は “API_1484_11” である．従って，SCORM 1.2 準拠の SCO は起動時に “API” という名前の API インスタンスを検索し，SCORM 2004 準拠の SCO は “API_1484_11” という名前の API インスタンスを検索するはずである．この性質を利用すると，SCORM 1.2 準拠の SCO を SCORM 2004 環境で利用可能となる．

このためには，既に図 7.3 に示したように，LMS 側で “API” と “API_1484_11” というふたつの API インスタンスを用意する．前者は SCORM 1.2 の RTE API 関数とデータモデルを提供し，後者は SCORM 2004 の RTE API 関数とデータモデルを提供する．上記のように各規格に準拠した SCO は，それぞれ該当する API インスタンスを検索して使用するので，SCO の修正は不要となる．

SCORM 1.2 対応の API インスタンスの内部では，SCORM 1.2 と SCORM 2004 との間のデータモデル要素やエラーコードの変換を行う．このとき，ほとんどのデータモデル要素は名称の差異以外は両規格で対応が取れているので，そのままマッピングが可能である．しかし，学習状態については，SCORM 1.2 では cmi.core.lesson_status というひとつのデータモデル要素で表現されていたものが，SCORM 2004 では cmi.completion_status と cmi.success_status に分離しているため変換のルールを決めておく必要がある．このようなルールの例を表 8.1，表 8.2 に示す．同様の変換は SCORM 1.2 の cmi.objective.n.status と SCORM 2004 の cmi.objective.n.completion_status, cmi.objective.n.success_status の間にも適用可能である．また，SCORM 2004 で新設された重要なデータモデル要素として，cmi.score.scaled, cmi.objective.n.score.scaled がある．これらはシーケンシングに影響を及ぼすため適切な値を SCO から設定する必要がある．SCORM 1.2 ではこれらに対応するデータモデル要素は存在しないが，cmi.core.score.raw, cmi.objective.n.score.raw を正規化して cmi.score.scaled, cmi.objective.n.score.scaled の値とするのが妥当と考えられる．なお，このような変換ルールは規格として定められているわけではないので，LMS で適宜決定して使用する必要がある．

8.2.2 RTE 移行のための SCO の対応

SCORM 2004 が発表されたといえ，すべての LMS がすぐに対応できるわけではなく，今後しばらくは，SCORM 1.2 対応の LMS と SCORM 2004 対応の LMS が並存する状況が続くと考えられる．従って，今後作成するコンテンツは両方の LMS に対応できるようになっていることが望ましい．

表 8.1 学習状態の変換 (SCORM 1.2→SCORM 2004)

SCORM 1.2		SCORM 2004	
cmi.core.lesson_status		cmi.completion_status	cmi.success_status
unknown	→	unknown	unknown
passed	→	completed	passed
failed	→	incomplete	failed
completed	→	completed	unknown
incomplete	→	incomplete	unknown
browsed	→	incomplete	unknown
not attempted	→	unknown	unknown

表 8.2 学習状態の変換 (SCORM 1.2←SCORM 2004)

SCORM 1.2		SCORM 2004	
cmi.core.lesson_status		cmi.completion_status	cmi.success_status
unknown	←	unknown	unknown
passed	←	unknown	passed
failed	←	unknown	failed
completed	←	completed	unknown
passed	←	completed	passed
failed	←	completed	failed
incomplete	←	incomplete	unknown
passed	←	incomplete	passed
failed	←	incomplete	failed

図 8.1 にこのような SCO を実現するための構成を示す. この SCO は内部に API インスタンスとインタフェースを取るための「仮想 API」を有している．この仮想 API は，現在の実行環境が SCORM 1.2 か SCORM 2004 かを判定し，SCO 本体との間で適切な API 関数呼び出し，データモデル要素の変換を行う.

実行環境の判定は，例えば，最初に “API_1484_11” という API インスタンスを，次に “API” という API インスタンスを検索し, 見つかった API インスタンスの名称で, 現在自身が SCORM 1.2 環境ないし SCORM 2004 環境のいずれかで実行されているのかを判定する.

API 関数呼び出し，データモデル要素の変換では，SCO 本体に対して RTE の API 関数に対応する関数を用意し，これらが呼び出されたら，必要なデータモデル要素の変換を行って，現在の環境の相当する API 関数を呼び出す，という操作を行う.

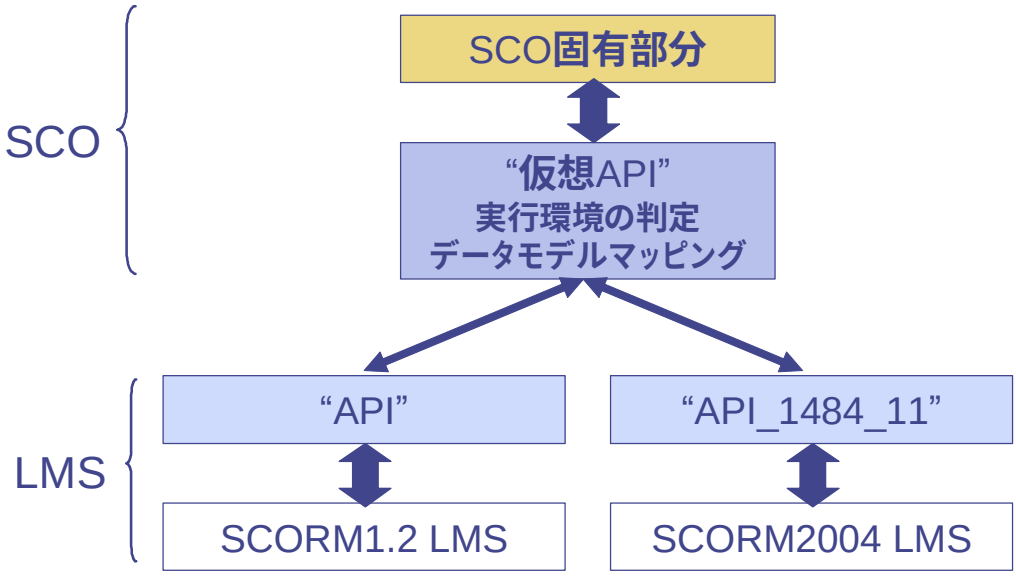

図 8.1 SCORM 1.2, SCORM 2004 両用 SCO

索引

T

U

あ

お

か

き

く

こ

さ

し

せ

て

と

な

は

ふ

ほ

ま

よ

ら

ろ

この文書について

本書の執筆者は以下の通りである.

1 章，3 章，6〜9 章　　仲林 清（NTT レゾナント株式会社）

2 章，4 章，5 章　　宮内 浩（学校法人産業能率大学），太田 衛（株式会社エネゲート）

本書は経済産業省がスポンサーとなり SCORM　2004 の普及促進を目的として作成された.