

SCORM テキスト

第 1.1 版

20012 年 2 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
標準化推進委員会

改訂履歴

日付	バージョン	改訂内容
2009 年 9 月	0.1	ドラフト
2010 年 3 月	0.2	ドラフト原稿追加
2010 年 4 月	0.3	文言訂正
2010 年 6 月	0.5	文言訂正
2010 年 7 月	1.0	文言訂正
2012 年 2 月	1.1	文言訂正・クレジット表記の追加

目次

1. はじめに	1
2. SCORM 総論	2
2.1 SCORM 規格の目的	2
2.1.1 SCORM 規格制定の背景	2
2.1.2 SCORM 規格の目的	2
2.2 SCORM の構成	4
2.2.1 SCORM の基本構成	4
2.2.2 SCORM の技術仕様	4
2.3 SCORM バージョンの変遷	7
2.3.1 SCORM 1.0 から SCORM 1.2	7
2.3.2 SCORM 1.2 から SCORM 2004 への更新	8
2.3.3 SCORM 2004 のマイナー更新	10
2.4 e ラーニングにおける技術標準化	11
2.4.1 工業製品における技術標準化	11
2.4.2 e ラーニングにおける技術標準化	13
3. SCORM 1.2	15
3.1 SCORM 1.2 の概要	15
3.1.1 第 1 編 : SCORM 概要	15
3.1.2 第 2 編 : SCORM アグリゲーションモデル	15
3.1.3 第 3 編 : SCORM ランタイム環境	15
3.2 コンテンツアグリゲーションモデル	16
3.2.1 SCORM コンテンツモデル構成要素	16
3.2.2 メタデータ	18
3.2.3 SCORM コンテンツパッケージング	20
3.3 ランタイム環境	28
3.3.1 起動	30
3.3.2 API	31
3.3.3 データモデル	35

4. SCORM 2004	44
4.1 SCORM 2004 概要	44
4.1.1 第1編：SCORM 概要	44
4.1.2 第2編：コンテンツアグリゲーションモデル	44
4.1.3 第3編：ADL テンツモデル構成要素	44
4.1.4 第4編：SCORM	44
4.2 コンテンツモデル	45
4.2.1 コンテンツモデル構成要素	45
4.2.2 メタデータ	47
4.2.3 SCORM コンテンツパッケージング	47
4.2.4 マニフェストファイルの記述例	49
4.3 ランタイム環境	51
4.3.1 起動	51
4.3.2 API	51
4.3.3 データモデル	54
4.4 シーケンシング	57
4.4.1 シーケンシング規格	57
4.4.2 ナビゲーション規格	60
4.4.2 ナビゲーション規格	エラー! ブックマークが定義されていません。

1. はじめに

WBT (Web-based Training) 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約 10 年が経過した。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, 多く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 2004 年に SCORM 2004 規格が新たに公開された。SCORM 2004 では, 規格書は全部で 800 ページを越えており, その全体像を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 および SCORM 2004 規格の全体像, および, 各機能の解説を行った。ある程度コンテンツ作成に経験のある方が, 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。

2. SCORM 総論

2.1 SCORM 規格の目的

2.1.1 SCORM 規格制定の背景

SCORM (Sharable Content Object Reference Model) は、独習型 e ラーニングコンテンツと e ラーニングシステムの相互運用性に関する標準規格である。1995 年ごろから、インターネットや WWW が普及するとともに、この技術を教育や訓練に活用しようという動きが現れてきた。これが現在 e ラーニングと呼ばれている分野の始まりである。

WWW を用いた独習型の e ラーニング環境には大きく二つの流れがあった。ひとつは、WWW のハイパーテキスト機能を用いてマルチメディアの e ラーニング教材を実現しようとするもの、もうひとつは従来スタンドアロンのパソコンなどで使用されていた CAI (Computer Assisted Instruction) の仕組みを WWW に移植しようというものである。

前者は、単純な閲覧機能だけの教材を提供するには適していたが、テスト問題を組み込んだり、学習者の学習履歴を取得するためには、学習者の識別機能やテストの採点機能などを、個別に CGI (Common Gateway Interface) などの仕組みを用いてプログラムで実現する必要があった。このため、プログラミングの知識が無ければコンテンツ作成ができず、コンテンツごとにこれらのプログラムをその都度作成する必要があるという難点があった。

一方、後者の CAI システムは、スタンドアロンのパソコンで動作するマルチメディアコンテンツを比較的簡易に作成可能な編集ソフトウェア（オーサリングツール）を有している場合が一般的であった。これらのオーサリングツールでは、テストなども作成可能であり、教材作成の手間の点では優位性があった。しかし、できあがった教材は製品ごとに専用の CAI エンジンを用いないと実行できず、作成したコンテンツを他社のエンジンで動作させることはできなかった。このような CAI エンジンで WWW サーバに組み込んで、ブラウザでコンテンツの閲覧やテストの回答が実施できる製品も現れたが、各社の製品間でコンテンツを相互に流通させることはできなかった。このような問題点を解決するために策定されたのが SCORM 規格である。

2.1.2 SCORM 規格の目的

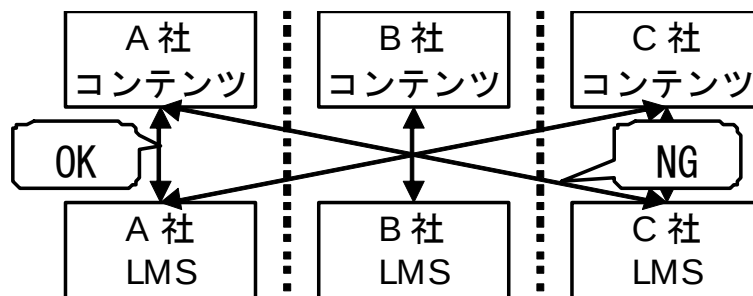
上記のように、WWW を用いた独習型の e ラーニング環境において、e ラーニングコンテンツと e ラーニングシステム (LMS: Learning Management System, 学習管理システム) の相互運用性を確立することが SCORM 規格の目的である。ここで、コンテンツには教育内容固有の解説やテスト問題が含まれ、LMS は学習者の識別を行い、学習履歴 (学習時間やテストの結果など) を記録する機能を有するものとする。SCORM 規格はこのようなコンテンツと LMS の間のインターフェースを規定することで、あるコンテンツが複数の異なるベンダが開発した LMS で実行でき、逆に、ある LMS で複数の異なるベンダが開発したコンテンツを実行できるような相互運用性を実現することを目指している。図 2.1 にこれを示す。

このような相互運用性が実現されると以下に挙げるような利点が得られる。

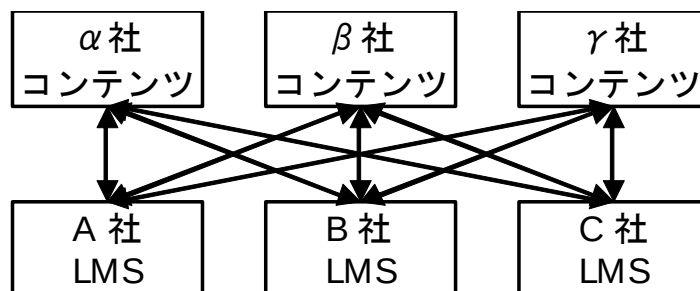
- ある利用者からみると、自分の保有する LMS で実行できるコンテンツの種類が飛躍的に増

加する。

- 自分で開発したコンテンツが、他の組織（例えば、社内の別部門や支社、他大学など）の LMS で（仮に LMS の種別が異なっても）簡単に実行できる。
- LMS が陳腐化して他の LMS に置き換えても、コンテンツはそのまま利用できる。
- 教材開発ツールや教材作成外注先を選択する際に、特定の LMS に依存した技術にしばられることが無く、ツールや外注先の本来の実力による選択が可能となる。



a) 標準規格無し



b) 標準規格有り

図 2.1 e ラーニングコンテンツの標準化

2.2 SCORM の構成

ここでは SCORM の概要として、どのような仕様書があり、何を定義しているのかを紹介する。

2.2.1 SCORM の基本構成

SCORM では、e-Learning 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-Learning 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大きな修正の必要がなく（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）ための規格と言える。

2.2.2 SCORM の技術仕様

SCORM1.2 の仕様書は以下の 3 編から構成されている。

- ・ **第 1 編：SCORM 概要**

(Book1: Sharable Content Object Reference Model(SCORM)Version 1. 2 The SCORM Overview)

ADL initiative に関する概要、SCORM に関する技術仕様、ガイドラインに関する概要が記述されている。

- ・ **第 2 編：SCORM アグリゲーションモデル**

(Book2: Sharable Content Object Reference Model(SCORM)Version 1. 2 The SCORM Content Aggregation Model)

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。また、これらのガイドラインは、LTSC (IEEE Learning Tchenology Standards Committee), IMS Global Learning Consortium, Inc, ARIADNE(Alliance of Remote Instructional Authoring and Distribution Network), AICC(Aviation CBT Committee)からの技術提供を受けまとめ上げられている。

- ・ **第 3 編：SCORM ランタイム環境**

(Book3: Sharable Content Object Reference Model(SCORM) Version 1. 2 The SCORM Run-Time Environment)

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。このガイドラインは、AICC の” CMI001 Guidelines for Interoperability” をその起源としている。

SCORM 2004 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、以下の仕様書が追加されている。

- ・ 第4編：シーケンシング&ナビゲーション

(Book4: Sharable Content Object Reference Model(SCORM) Version 1. 3 The SCORM Sequencing and Navigation(SN) Book)

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おり、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。

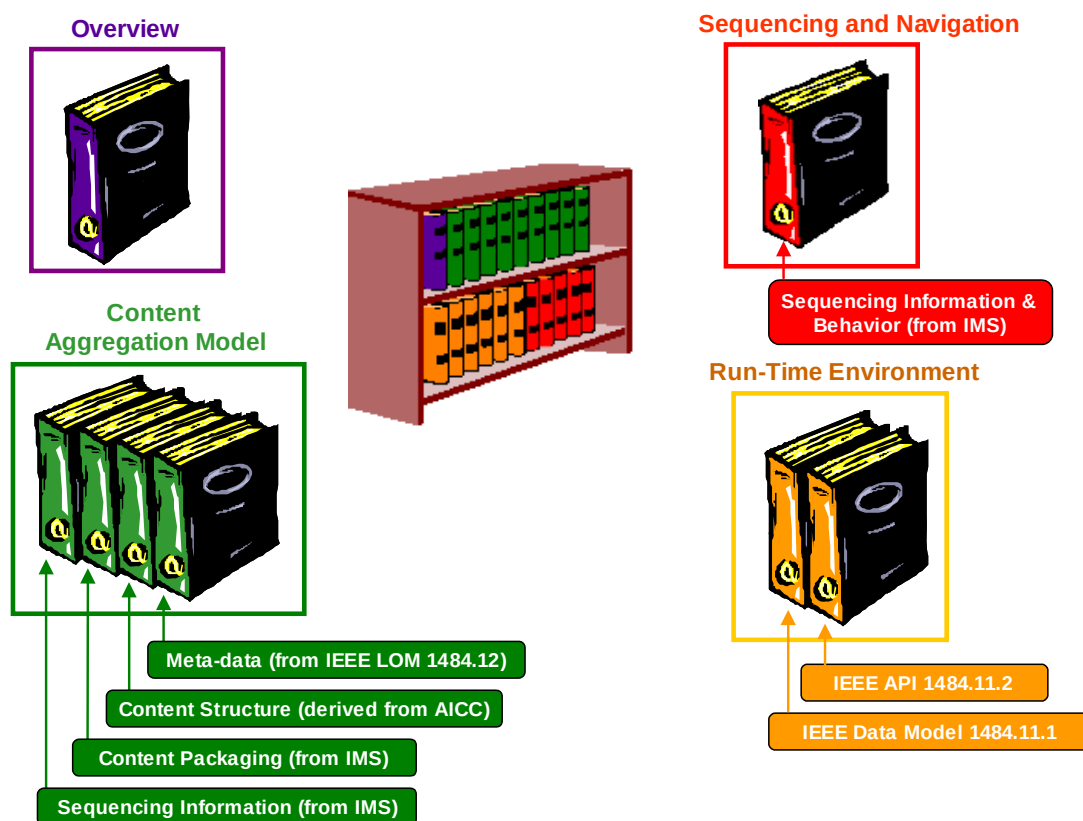


図 2.2 e ラーニングの SCORM の構成

2.2.2.1 第1編 SCORM 概要

第1編の SCORM 概要では、ADL の SCORM に対する取り組みやビジョン、SCORM の構成、第1編から第3編までの概要、SCORM を選択する理由(SCORM の必然性)についてなどが書かれており、はじめて SCORM コンテンツを作る場合、仕様書の全体構成や考え方を理解するのに有効である。

第1編の中には、SCORM のタイトルの変更についても触れられている。SCORM1.0 のときには、"Sharable Courseware Object Reference Model"であったタイトルが、SCORM1.1 では、"Sharable Content Object Reference Model"になったというもので、SCORM の技術仕様が適用される単位を、明確にあらわしているものといえる。

また、LMS とコンテンツの役割分担についても記述されている。SCORM では、LMS は学習資源の集合(コース)によって定義されたルールを解釈し、学習資源の提供方法を決定するというもので、SCORM ではコースが構成されてはじめて各学習資源の文脈が構成されるということを示し、学習資源をさまざまに組み合わせ、コースを構成することで、既存の学習資源を活用しながら新しい意味をもったコースが構成できる。このことが SCORM のタイトルを変更した理由である。

2.2.2.2 第2編 SCORM アグリゲーションモデル

第2編の SCORM アグリゲーションモデルでは、学習資源を集約してコースにするための過程について、

- ・ コンテンツモデル
- ・ メタデータ
- ・ コンテンツパッケージング

の説明で構成されてる。SCORM の構成要素であるアセット、SCO、メタデータなどの用語についても説明されており、教材づくりという観点から、SCORM コンテンツを設計する場合に理解しておくべき事項が盛り込まれている。

2.2.2.3 第3編 SCORM ランタイム環境

ランタイム環境とは、大雑把に言うと学習者に提供される学習環境のことで、学習者が LMS を通して学習資源を活用(起動)し、学習状況の送受信(進捗管理)、終了するまでの一連の流れを提供する環境のことである。

第3編の SCORM ランタイム環境では、これらの流れを実現する仕組みと、そこで送受信されるデータの種類について説明している。実際に SCORM API をインプリメントする場合にプログラマに必要となる情報であるとともに、教材づくりの上でも、どのような管理情報を有効に使えるのかを知ることができる。

2.2.2.4 第4編シーケンシング&ナビゲーション

SCORM2004 では、シーケンシング&ナビゲーション仕様書が追加された。

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。

また、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とにより、「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り、さらには、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ている。

2.3 SCORM バージョンの変遷

SCORM は過去のいくつかのバージョンに変更点を加えていく形で進歩してきた。それぞれの仕様書の説明にもあるように、SCORM 立案以前からあった、以下のような規格の影響を受けている。

- IEEE Data Model for Content Object Communication
- IEEE ECMAScript Application Programming Interface for Content to Runtime Services Communication
- IEEE Learning Object Metadata (LOM)
- IEEE Extensible Markup Language (XML) Schema Binding for Learning Object Metadata Data Model
- IMS Content Packaging
- IMS Simple Sequencing

2.3.1 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・評価段階に入り、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が、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ている。さらには、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ており、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ている。

2.3.2 SCORM 1.2 から SCORM 2004 への更新

SCORM 1.2 から SCORM 2004 への主な変更点を以下に述べる。

・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った。

2.3.2.1 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、以下のようなことが制御できるようになっている。

- ・ 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

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った。

2.3.2.2 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りました。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が可能となった。

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います。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。

2.3.2.3 SCORM ランタイム環境の変更

SCORM ランタイム環境については、API オブジェクト名、API 関数名、データモデルの3つの要素において、SCORM 1.2 から大きく変更がなされている。

2.3.2.4 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。

2.3.3 SCORM 2004 のマイナー更新

SCORM2004 ではエディション表記による小変更を行っており、それぞれ以下のような内容が更新されている。

SCORM Version 1.1
2001 年 1 月

コンテンツアグリゲーション
モデル (CAM)
メタデータ+バインディング

ランタイム環境 (RTE)
API+データモデル

SCORM Version 1.2
2001 年 10 月

コンテンツアグリゲーション
モデル (CAM)
メタデータ+バインディング
コンテンツ・パッケージング
コンテンツ構成

ランタイム環境 (RTE)
API+データモデル
仕様修正

SCORM 2004
2004 年 1 月

コンテンツアグリゲーション
モデル (CAM) Version 1.3
メタデータ+バインディング
コンテンツ・パッケージング
コンテンツ構成
シーケンシング&ナビゲーション
仕様修正

ランタイム環境 (RTE)
Version 1.3
API+データモデル
仕様修正

シーケンシング&ナビゲーション
(SN) Version 1.3
シーケンシングルール+ビヘイビア

SCORM 2004 3rd Edition
2006 年 10 月

コンテンツアグリゲーション
モデル (CAM) Version 1.0
メタデータ+バインディング
コンテンツ・パッケージング
コンテンツ構成
シーケンシング&ナビゲーション
仕様修正

ランタイム環境 (RTE)
Version 1.0
API+データモデル
仕様修正

シーケンシング&ナビゲーション
(SN) Version 1.0
シーケンシングルール+ビヘイビア
仕様修正

SCORM 2004 4th Edition
2009 年 3 月

コンテンツアグリゲーション
モデル (CAM) Version 1.0
メタデータ+バインディング
コンテンツ・パッケージング
コンテンツ構成
シーケンシング&ナビゲーション
仕様修正

ランタイム環境 (RTE)
Version 1.0
API+データモデル
仕様修正

シーケンシング&ナビゲーション
(SN) Version 1.0
シーケンシングルール+ビヘイビア
仕様修正

SCORM 2004 4th Edition
2009 年 8 月

コンテンツアグリゲーション
モデル (CAM) Version 1.1
メタデータ+バインディング
コンテンツ・パッケージング
コンテンツ構成
シーケンシング&ナビゲーション
仕様修正

ランタイム環境 (RTE)
Version 1.1
API+データモデル
仕様修正

シーケンシング&ナビゲーション
(SN) Version 1.1
シーケンシングルール+ビヘイビア
仕様修正

図 2.3 SCORM バージョンの変遷

2.4 e ラーニングにおける技術標準化

2.4.1 工業製品における技術標準化

工業製品における技術標準化には大きく二つの目的がある。ひとつは製品標準と呼ばれるもので、工業製品自体の形状、寸法、機能などを標準化するものである。もうひとつは方法標準と呼ばれるもので、製品自体ではなく、製品を製造したり検査したりする手順や方法を標準化するものである。

一般的に良く知られている標準規格は前者に属している。例えば、ネジの形状や寸法の標準規格は製品標準の例である。この標準規格のおかげで、世界中どのメーカーが作ったネジでも組み合わせて使用することが可能になる。同じような規格の例として VTR の VHS 規格が挙げられる。VTR 製品化の初期のころには、ビクターの提唱した VHS 規格とソニーの提唱したベータ規格に準拠した製品が市場でシェアを争っていた。このため、利用者は自分のビデオデッキに合致するビデオテープを注意して購入する必要があり、映画などのコンテンツも一方の規格でしか販売されていない場合があるなど、不便な状態が続いていた。近年では、VHS 規格のシェアが圧倒的に大きくなったため、このような不便は解消され、メーカー側も一種類の規格に準拠した製品だけを製造すればよかった。パソコンで良く用いられる USB 機器の規格も製品標準であり、インターネットの通信の規格である TCP/IP や HTTP もコンピュータ間でデータをやり取りする際のデータの順番や形式を定めたもので、製品標準の一種である。このような製品標準において重要な点は、標準規格はやりとりされるデータの順番や形式などを標準化するだけで、内容には関知しないという点である。

一方、方法標準の代表的な例として ISO 9000 を挙げることができる。ISO 9000 は具体的な製品自体の仕様などを決めていたわけではなく、製品を製造する工程における設計・評価の過程において必要となる手順を定めることで、間接的に製品の品質を一定水準に保つことを目的としている。同様な標準規格の例として、環境マネジメントに関する ISO 14000、情報セキュリティに関する ISO 27000 などを挙げることができる。

次に標準規格がどのように決まるのかを考えてみる。標準規格決定の形態に応じて、標準規格には大きく「デファクト (de facto) 標準」と「デジュリ (de jure) 標準」が存在する。この比較を表 2.1 に示す。デファクト標準は特定企業の製品が普及することで、その製品に用いられている技術仕様がそのまま市場の標準規格として受け入れられるものである。パソコンの OS の Windows などはその例である。その意味で、製品化が標準化よりも先に行われる。一方、デジュリ標準は正規の標準化団体が制定するものである。標準化団体の例として、国際的な団体としては ISO (International Organization for Standardization) や IEC (International Electrotechnical Commission)、国内レベルでは JISC (Japanese Industrial Standards Committee) などがある。

表 2.1 デファクト標準とデジュリ標準

	デファクト標準	デジュリ標準
標準制定過程	市場が決定 製品化→標準化	標準化機関が決定 標準化→製品化
標準化の期間とニーズ	短い 開発者が市場獲得に有利	長い 規格が使われない可能性
制定過程と参加者	制定過程はクローズド 参加者が限られ不透明	制定過程はオープン 参加者は制限なし

デファクト標準は、特定の企業・団体が定めるものであるから、標準化の期間は短く、また、その規格が普及すると技術の良く分かった開発者は市場獲得に有利となる。一方、規格の内容は、特定の企業・団体が決定するため、制定過程はクローズドで不透明である。これに対して、デジュリ標準は、基本的に誰でも参加可能な標準化団体で制定が進められる。このため、参加者の利害が対立した場合は規格が決まるまでに長い時間がかかる。また、規格がなかなか決まらなかったり、複数参加者の意見を入れることで規格が複雑化して、結局、使われない規格ができてしまう可能性がある。

デファクト標準とデジュリ標準の対極的な例として有名なのが、インターネットと OSI (Open Systems Interconnection) である。今日、コンピュータ通信の規格としては TCP/IP に代表されるインターネットの規格を用いるのが当然であるが、1980 年代にはこのような状態ではなく、複数の規格が存在していた。OSI は国際標準化機関である ISO/IEC JTC1 において標準化が進められたコンピュータ通信の標準規格であった。インターネットの開発では、大学などが中心となり、厳密な規格を定めるよりも、単純な仕様で実際に動くプログラムの実装が優先された。これに対し、OSI の開発には、世界各国の通信会社やコンピュータメーカーが参加したため、仕様の決定に時間がかかり、また、各国の意見を取り入れたため仕様が複雑で大規模なものになってしまった。このため、実際に仕様を実装することは非常に困難で、結局、OSI が普及する前にインターネットが市場のシェアを獲得してしまった。現在では、OSI の名前は OSI の 7 層参照モデルとして残っているだけである。

このように、デファクト標準とデジュリ標準は対立するもののよう捉えられており、かつ、デファクト標準の方が良いものである、と考えられている場合が多いが、実際にはそれは誤解である。標準規格には、デファクトとして普及したのち、デジュリになったものが多く存在する。例えば、C 言語は Unix の記述言語として開発され、多くのワークステーションやパソコンに移植されて普及した。しかし、その過程でデータ型や関数名などに多くの方言が生まれ、処理系間の互換性が無くなってしまった。このような状態を解消するためにアメリカの正規標準化団体である ANSI (American National Standards Institute) でデジュリ標準化が進められた。その後、ISO でも標準化が行われた。現在では、C 言語の処理系は ISO 準拠のものが普及している。

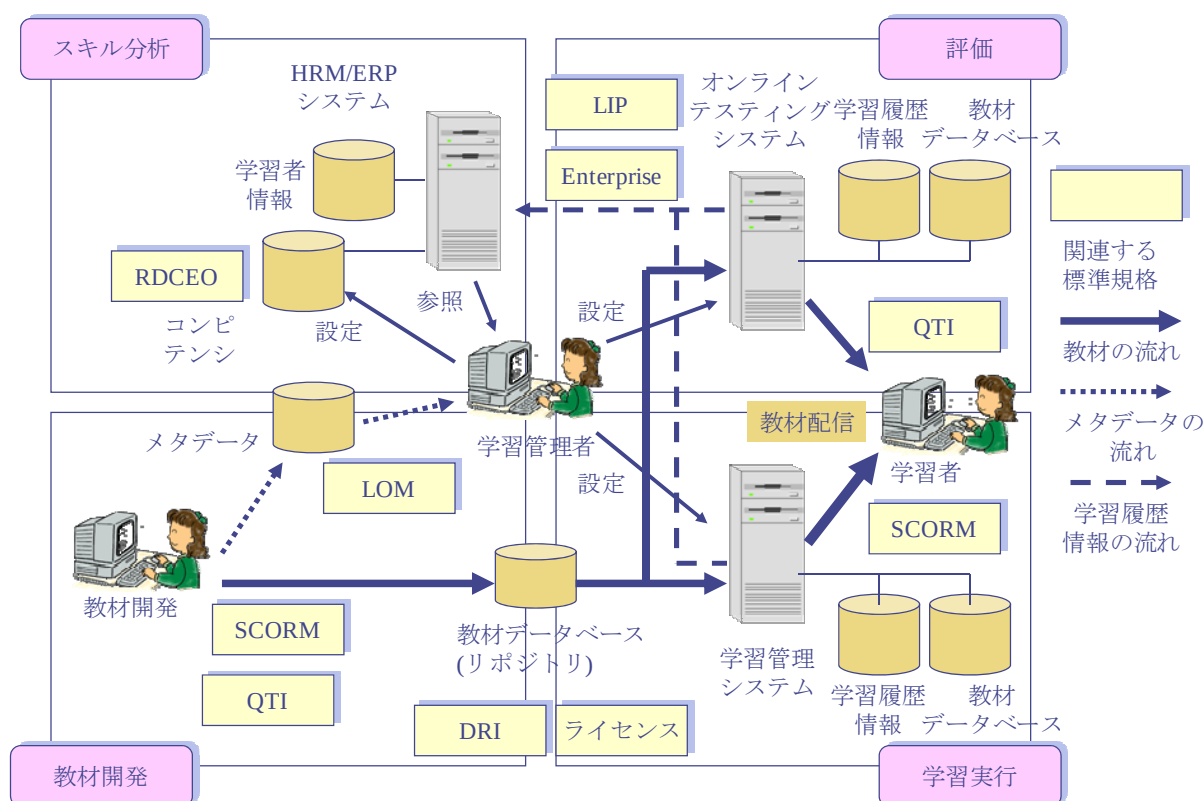
また、最近の標準化は、デファクト標準とデジュリ標準の良いところを併せ持つコンソーシアム標準の形態を取ることが多い。コンソーシアムはある業界の製造業者や利用者が集まって形成する団体で、基本的には誰でも自由に参加できるが、標準化の手続きは正規標準化団体よりも簡素化されている場合が多い。

2.4.2 e ラーニングにおける技術標準化

e ラーニングを進めるに当たって必要となる基本的な情報は、「学習者情報」、「学習体系情報」、「コンテンツ情報」である。これらには以下のように詳細化できる。

- 学習者情報
 - 氏名、学習者 ID、などの個人情報、所属組織などに関する情報、過去の学習履歴やポートフォリオ、保有資格、スキル、など
- 学習体系情報
 - カリキュラム、シラバス、スキル・コンピテンシー体系
- コンテンツ情報
 - 学習コンテンツ、テストコンテンツ、メタデータ、リポジトリ

これらの情報は、いずれも e ラーニングプラットフォームが変わっても永続的に利用可能である必要がある。したがって、2000 年ごろからこれらのデータを記述し、異なるプラットフォームでも利用可能とするための技術標準化が世界各国で進められてきた。図 2.4 に、e ラーニング



SCORM: Sharable Content Object Reference Model, LOM: Learning Object Metadata

RDCEO: Reusable Definition of Competency or Educational Objective

QTI: Question and Test Interoperability, DRI: Digital Repository Interface

LIP: Learner Information package

図 2.4 e ラーニングサイクルと標準規格

サイクルにおけるこれらの規格の位置付けを示す。

技術標準化はおおむね、技術の研究開発→規格作成→規格を実装したシステム開発→普及活動

→認定活動といった流れで進んでいく。図 2.4 に示した規格のうち、SCORM, LOM などは普及が進んでおり、SCORM については認定活動が行われている。

一方、近年では、より多様な規格の標準化にも目が向けられてきている。分散コンテンツリポジトリに関する CORDRA (Content Object. Repository Discovery and Registration/Resolution Architecture) や、グループ学習を含む多様な学習活動の記述の流通再利用を狙った LD (Learning Design) などはその一例である。

なお、e ラーニング標準化団体にも、デジュリ標準化団体とそれ以外の団体がある。前者には、ISO/IEC JTC1/SC36, IEEE LTSC (Learning Technology Standards Committee) などがある。それ以外の団体としては、アメリカ高等教育系の IMS Global Learning Consortium, アメリカ国防省系の ADL (Advanced Distributed Learning), 航空機産業関連団体の AICC (Aviation Industry CBT Committee) などがある。e ラーニング標準化でも、デファクト標準があとからデジュリ標準化される流れがあり、SCORM の一部である CMI 規格や CP 規格は、それぞれ、AICC, IMS で開発された規格が、IEEE LTSC, ISO/IEC JTC1/SC36 でデジュリ標準化された。

3. SCORM 1.2

3.1 SCORM 1.2 の概要

SCORM1.2 での規格化された仕様書は以下の 3 編から構成されている。

3.1.1 第 1 編：SCORM 概要

ADL の SCORM に対する取り組みやビジョン，SCORM の構成，第 1 編から第 3 編までの概要，SCORM を選択する理由(SCORM の必然性)についてなどが書かれている。

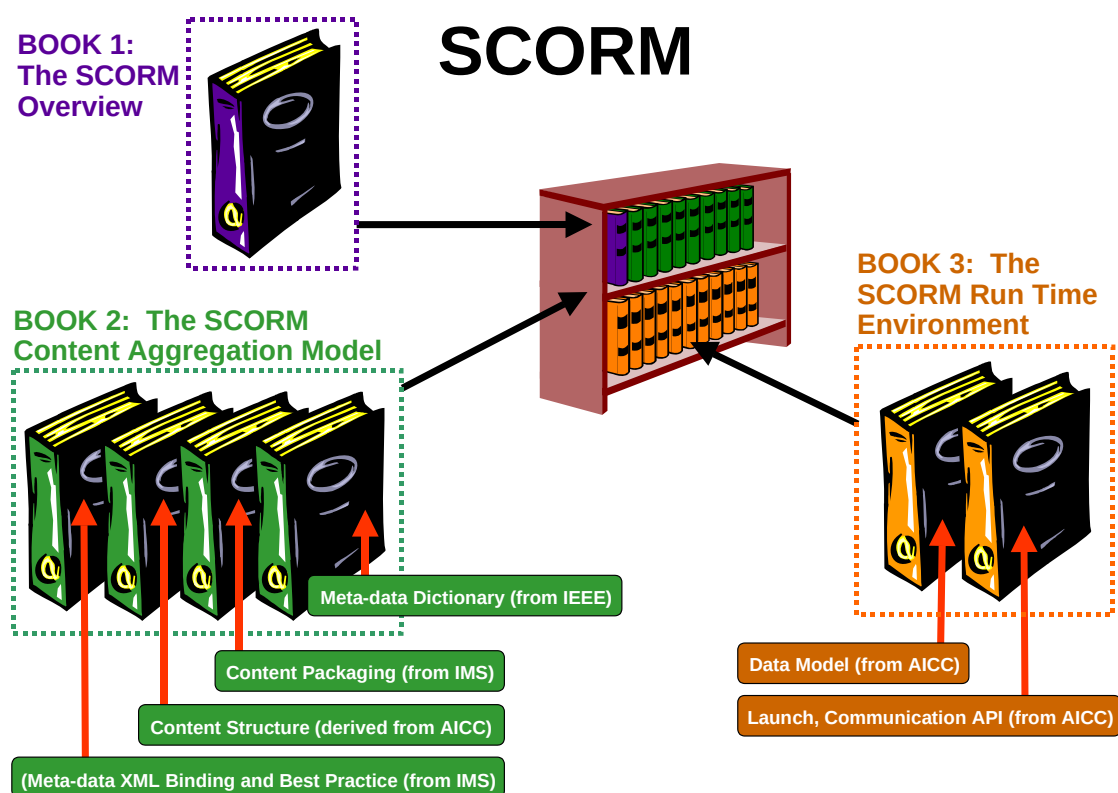


図 3.1 SCORM1.2 の構成

3.1.2 第 2 編：SCORM アグリゲーションモデル

SCORM アグリゲーションモデルでは，学習資源を集約してコースにするための過程について，

- ・ コンテンツモデル
- ・ メタデータ
- ・ コンテンツパッケージング

の説明で構成されている。

3.1.3 第 3 編：SCORM ランタイム環境

ランタイム環境では，学習者が LMS を通して学習資源を活用(起動)し，学習状況の送受信（進捗管理），終了するまでの一連の流れを実現する仕組みと，そこで送受信されるデータの種類について解説している。

3.2 コンテンツアグリゲーションモデル

SCORM1.2 のコンテンツアグリゲーションモデルは、階層型のコンテンツ構造を基本とし、SCO、アセット（3.2.1 で述べる）の Web コンテンツをどのような形で集約し、どのような順番で学習者に提示するかを決めるための規定を提供している。コンテンツアグリゲーションモデルに従って教材作成者が作成したコンテンツは、LMS で実行されて学習者に提供される。従って、コンテンツアグリゲーションモデルは、教材作成者がある設計意図に従って作成した学習経験を学習者に伝えるための重要な手段である。

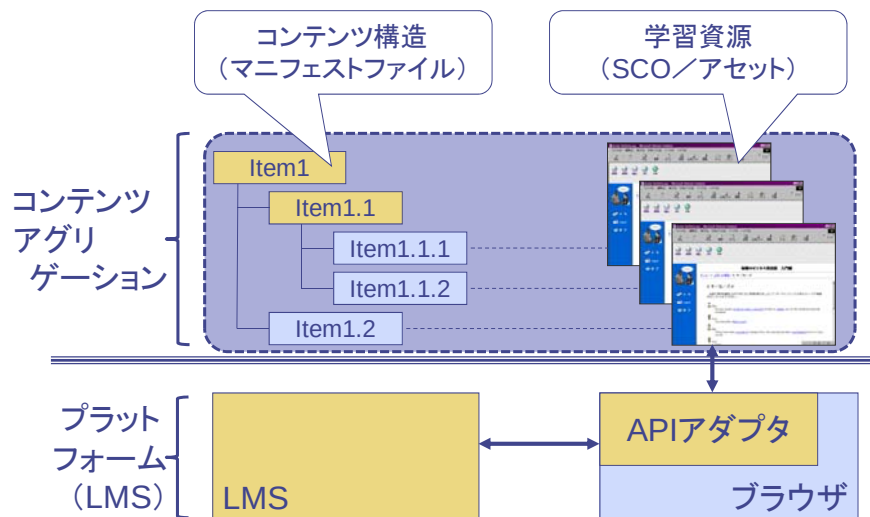


図 3.2 SCORM のコンテンツと LMS

3.2.1 SCORM コンテンツモデル構成要素

図 3.2 に SCORM1.2 のコンテンツとプラットフォーム（LMS）の関係をしめす。コンテンツと LMS は、それぞれ、サーバ側（Web サーバ側）とクライアント側（Web ブラウザ側）に分かれている。ブラウザで表示されるコンテンツには、アセットと SCO (Sharable Content Object) の二種類がある。これらを合わせて学習資源（または、学習リソース）と呼ぶ。サーバ側では、学習資源を教科書のような階層構造に集約して、一つの学習コンテンツの構造を定義するためのマニフェストファイルがある。これらのクライアント側とサーバ側のコンテンツを合わせてコンテンツアグリゲーションと呼ぶ。以下、それぞれについて説明する。

3.2.1.1 アセット (Asset)

アセットは SCORM コンテンツの最小単位で、テキスト、試験問題、音声、画像、動画、Flash コンテンツ、HTML ページなど、Web クライアントで表示再生可能なマルチメディアデータである。HTML ページは他のアセットを包含できるので、小さなアセットを集約して大きなアセットを構築できる。

3.2.1.2 SCO

SCO (Sharable Content Object) は、複数のアセットを集めたコンテンツである。アセットとの違いは、SCORM ランタイム環境 (Run-time Environment, RTE) による LMS との通信機能を持つことである。ランタイム環境を用いると、学習者の学習進捗状況や学習時間などを SCO

から LMS に送信することができるので、LMS はこれらの学習履歴を記録することができる。そのような意味で、SCO は LMS が学習履歴を記録可能なコンテンツの最小単位である。

SCO は SCORM におけるコンテンツの再利用の単位でもある。SCO をどの程度のサイズにすればよいかに関して SCORM では何も定めていないが、様々な教材のなかで SCO を再利用することを考えると、SCO のサイズはできるだけ小さい方がよい、ということができる。一方、サイズをあまり小さくしすぎると、まとまった教育的な意味が失われてしまうので、適切なバランスをとる必要がある。また、再利用性を高めるためには、SCO は他の SCO から独立である方がよい。SCO 同士の独立性を高めるためには、例えば、ある SCO の中で、他の SCO の内容に関して「～～を参照」というような表現は避けなければならない。

SCO は SCORM ランタイム環境規格に準拠する必要がある。すなわち、ブラウザ中で、LMS と通信するためのインターフェースである API アダプタを検索し、API アダプタの API 関数を用いて初期化 (LMSInitialize(" ")) 関数の呼び出し) と終了 (LMSFinish(" ")) 関数の呼び出し) を行う必要がある。他の関数をどのように用いるかは SCO の作成者に任されている。また、SCO がランタイム環境規格に準拠するということは、SCO は LMS から起動され、他の SCO を起動してはならないことを意味する (詳細は 3.3 ランタイム環境の説明を参照)。

SCO がランタイム環境規格に準拠することにより、以下のようなメリットが生ずる。

- ランタイム環境規格に準拠する LMS は、誰が作成した SCO でも、それを起動して学習履歴を記録することができる。
- LMS は、SCO が起動したこと、SCO が終了したことを知ることができる。
- LMS にとってどのような SCO でも同一の手順で起動することができる。

3.2.1.3 コンテンツアグリゲーション (Content Aggregation)

コンテンツアグリゲーションは、SCO やアセットなどの学習資源や他のコンテンツアグリゲーションを図 3.3 のように階層的に集約したものである。コンテンツアグリゲーションは、教科書の章節項や、コース・モジュールなどに相当するものと考えることができる。すなわち、このような階層構造を取り入れることで、さまざまなサイズの教育の単位の構造を表現することが可能となる。階層構造の末端のノードは SCO やアセットなどの学習資源に対応している。

また、コンテンツアグリゲーションは、学習資源を学習者に提示する順番 (シーケンス) を定義している、と考えることもできる。SCORM 1.2 では、学習資源の提示順序の制御 (シーケンシング) に関して、アグリゲーション間の前提条件を記述することができるが、この機能は十分なものではなく、本格的なシーケンシング機能は SCORM 2004 で実現された。

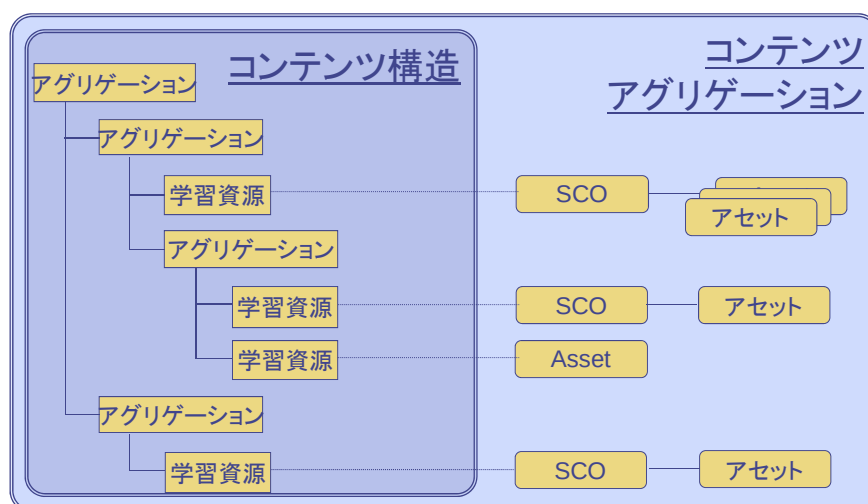


図 3.3 コンテンツアグリゲーション

このように、学習者に提示する学習資源と、学習コンテンツ全体の構造や提示順を制御するコンテンツアグリゲーションを明確に分離したことが SCORM の一つの特徴となっている。従来の CAI オーサリングツールでは、このような分離が必ずしも明確ではなく、学習資源とシーケンシング制御が独自の仕様で行われていた。このため、学習資源の再利用が難しく、コンテンツ作成にも独自の技術が必要であった。SCORM では、学習資源とコンテンツ構造・シーケンシング制御を明確に分離することで、学習資源の再利用性を向上させている。もちろん、SCO などの学習資源自体は、学習資源の内部でプログラミングが可能であり分岐などの制御を行うことができるが、これはあくまで学習資源内部に閉じたものであり、他の SCO などに影響を及ぼしてはならない。また、LMS は SCO 内部のシーケンシング制御とは無関係である。

3.2.2 メタデータ

メタデータは、データについてのデータである。もっとも典型的なメタデータは、図書館の図書カードに記載されている書誌情報であろう。書籍の題名、著者、発行年、出版社などは、対象となる書籍の各種の属性を表すメタデータの例である。近年では、書誌情報も電子化されて書籍の検索の容易化に役立っている。同様に e ラーニングコンテンツなどに付与するメタデータも、学習者や教材作成者がコンテンツの利用や再利用をする際のコンテンツの検索を容易化することをねらいとしている。

メタデータには、付与の対象とするコンテンツの種別に応じて各種の標準規格が存在するが、SCORM1.2 では、e ラーニングコンテンツや学習オブジェクト (Learning Object) を対象とするメタデータとして IEEE LTSC で標準規格化された LOM (Learning Object Metadata) 規格を用いることになっている。

メタデータは、SCORM コンテンツモデルを構成する各構成要素に付与することができる。すなわち、メタデータは、アセット、SCO、コンテンツアグリゲーションのそれぞれに付与され、それぞれの構成要素の検索を容易にし、再利用性を高めるために用いられる。

なお、SCORM1.2 ではメタデータを付与するか否かは任意である。メタデータがまったく付与されていなくても、LMS 上でのコンテンツの実行に影響を与えることは無い。しかし、組織内で

のコンテンツの再利用を促進したい場合などには、メタデータを活用することが望ましい。

3.2.2.1 SCORM メタデータ情報モデル

SCORM1.2 で用いられている LOM 規格では、題名、キーワード、作成者など約 60 のメタデータ要素を規定している。メタデータ要素は大きく以下の 9 つカテゴリに分類される。

1) 一般 (General)

対象の属性・特徴を総括して記述する一般的な情報。タイトル、キーワード、概要、使用言語、など。

2) ライフサイクル (Lifecycle)

対象の開発履歴や現在の状態、開発に関わった人などに関する情報。バージョン番号、開発者の役割、組織、など。

3) メタメタデータ (Meta-metadata)

メタデータそのもののメタデータ。メタデータを付与した人の情報、日付、など。

4) 技術的事項 (Technical)

対象の技術的要求条件や特徴。データのフォーマット、サイズ、実行環境、など。

5) 教育的事項 (Educational)

対象の教育的・教育学的特徴。利用者の種別、学年、難易度、利用学習形態や対話性、など。

6) 権利 (Rights)

対象の所有権や利用条件。使用料や著作権による制限、など

7) 他オブジェクトとの関連 (Relation)

対象と他の学習オブジェクトとの関係。参照、包含、前提、など。

8) 注釈 (Annotation)

対象に関するコメント。コメントの作成者、日付、内容。

9) 分類体系 (Classification)

対象がある分類体系 (例えば、図書館の書誌情報) のどこに属するかについての記述。

分類体系の目的、名称、分類体系における分類項目や ID、など

メタデータの各要素は、要素名称と要素値から構成される。要素値は要素の性質によって、使用できるデータタイプや繰り返し数が決められている。また、データタイプによっては、システムが最低限保証すべきデータの最大サイズが決められているものがある。繰り返し数についても、システムが最低限保証すべき繰り返し数の最大回数が決められている。なお、以下に例を挙げる要素名称の前の 1.2 などの数字は、上記の 9 つのカテゴリの中の要素番号である。

- 「1.2 タイトル」

- 繰り返し数: 1

- データタイプ: 文字列 (最低限保証すべき数: 1000 文字)

- 「1.6 キーワード」

- 繰り返し数: 0 以上 (最低限保証すべき数: 10 回)

- データタイプ: 文字列 (最低限保証すべき数: 1000 文字)

- 「5.8 難易度」
 - 繰り返し数：0 または 1
 - データタイプ：制限語彙
 - ✧ 大変易しい (very easy)
 - ✧ 易しい (easy)
 - ✧ 標準レベル (medium)
 - ✧ 難しい (difficult)
 - ✧ 大変難しい (very difficult)
- 「5.9 学習時間」
 - 繰り返し数：0 または 1
 - データタイプ：日付形式
- 「6.2 著作権および制約」
 - 繰り返し数：0 または 1
 - データタイプ：制限語彙
 - ✧ 有り (yes)
 - ✧ 無し (no)

以上のように、要素の性質によって異なるデータタイプが用いられる。特に、制限語彙のデータタイプは、要素に応じて必要な語彙が定められている。また、開発者などの人物や組織を表すためには vCard¹という形式が使用される。

メタデータ項目の詳細については、SCORM 1.2 規格書を参照のこと。

3.2.2.2 SCORM メタデータ構成要素

先に述べたように、SCORM ではコンテンツモデルを構成する各構成要素にメタデータを付与することができる。すなわち、メタデータは、アセット、SCO、コンテンツアグリゲーションのそれぞれに付与される。記述例については 3.2.3.3 を参照のこと。

3.2.3 SCORM コンテンツパッケージング

SCORM のコンテンツをオーサリングツールなどで作成してファイルとして保存したり、コンテンツプロバイダからコンテンツを購入して LMS に搭載する際に、コンテンツをやり取りするための物理的な形式を規定するのがコンテンツパッケージングである。すなわち、コンテンツパッケージングは、LMS やオーサリングツールの入出力ファイルとなる。コンテンツパッケージングでは以下のような項目が規定されている。

- パッケージの構成や内容を記述するマニフェストファイル
- マニフェストファイルの XML バインディング
- マニフェストファイルと関連する学習資源などのファイルを zip 形式などでパッケージングする方法

SCORM コンテンツパッケージングは、IMS コンテンツパッケージング規格に基づいており、

¹ <http://www.imc.org/pdi/>

SCORM 固有の事項（SCO、アセットの表現、など）のために拡張されている。

3.2.3.1 コンテンツパッケージの構成

図 3.3 にコンテンツパッケージの構成を示す。以下、各々の構成要素について説明する。

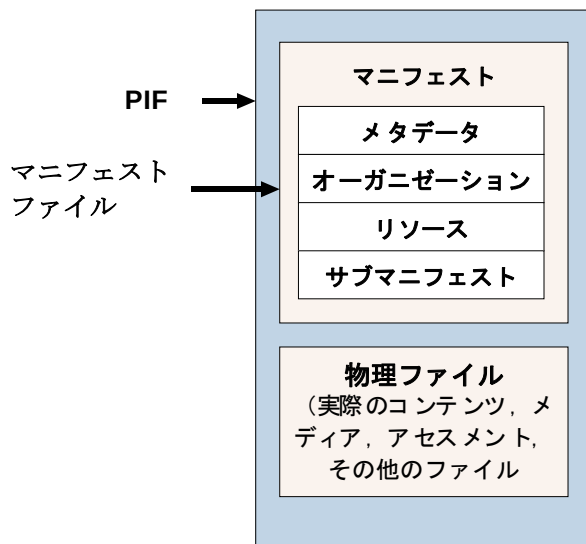


図 3.4 コンテンツパッケージ

1) マニフェスト (Manifest)

マニフェストはコンテンツパッケージ全体の構成を記述する XML ファイルである。マニフェストはさらに、図 3.4 に示すように、メタデータ、オーガニゼーション、リソース、サブマニフェストの要素を含んでいる。

2) メタデータ (Metadata)

メタデータはデータに関するデータであり、コンテンツパッケージ全体、および、その構成要素を記述するために用いられる。

3) オーガニゼーション (Organization)

オーガニゼーションは、コンテンツの階層構造を記述するために用いられる。SCORM のコンテンツアグリゲーションはオーガニゼーションによって表現される。オーガニゼーションは階層型のノードから構成されていて、各ノードからそれに対応するリソースを指すことができる。

4) リソース (Resource)

リソースは、学習資源を記述するために用いられる。実際の学習資源はコンテンツパッケージの内部にある場合もあるし、ネットワーク上など外部の学習資源を参照する場合もある。

5) 物理ファイル (Physical File)

リソースによって指定される実際の学習資源である。学習資源は上記のように、コンテンツパッケージの内部にある場合もあるし、ネットワーク上など外部の学習資源が参照されている場合もある。

6) PIF (Package Interchange File)

zip 形式でコンテンツパッケージ全体をアーカイブしたものを PIF (Package Interchange File, パッケージ交換ファイル) と呼ぶ.

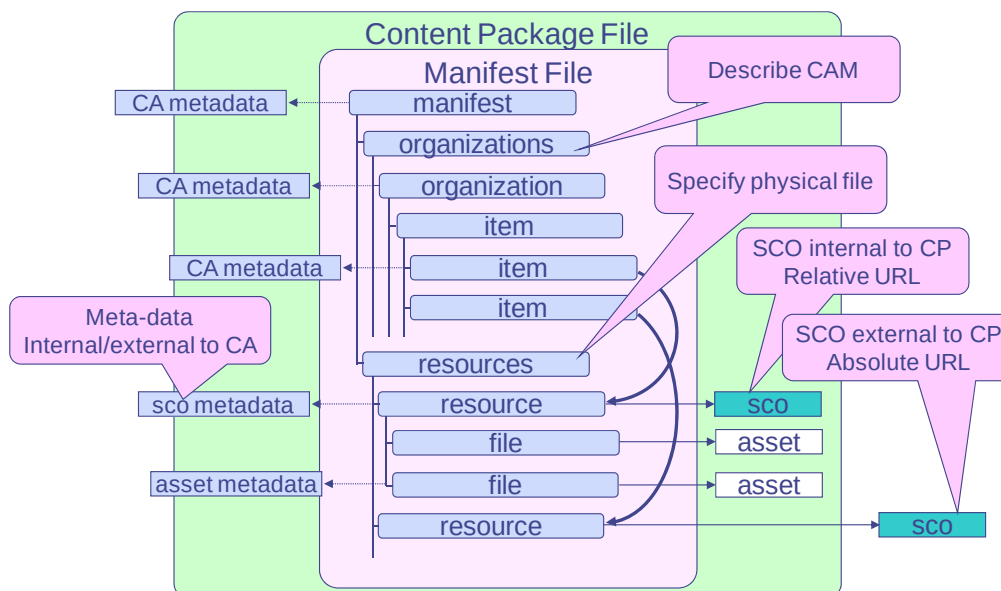


図 3.5 コンテンツパッケージの詳細

図 3.5 にコンテンツパッケージをやや詳細に記述した図を示す．PIF 中では，マニフェストファイルは **imsmanifest.xml** という名称で，ファイルディレクトリの最上位（ルートディレクトリ）に配置される．マニフェストは，大きくオーガニゼーションとリソースに分かれる．オーガニゼーションは階層構造になっていて，階層の各ノードはアイテム（**item**）というタグで表わされる．階層の末端レベルのアイテムからは，対応する学習資源を表すために，**Identifier Ref** 属性を用いてリソースへの参照が記述されている．リソースは，配下にそれを構成するファイルが記述される．さらにリソースやファイルからは，**SCO** やアセットの物理的な実体への参照が **URL** として記述されている．**SCO** やアセットはコンテンツパッケージ中に存在する場合もあるし，外部に存在する場合もある．コンテンツパッケージ中に存在する場合は，**URL** はマニフェストファイルからの相対パスとなる．外部に存在する場合は，**www** サイトの指定から始まる絶対パスとなる．以上の構成要素にはそれぞれメタデータを付与することができる．

3.2.3.2 SCORM コンテンツパッケージングデータモデル

SCORM コンテンツパッケージは、IMS コンテンツパッケージを SCORM 固有の事項に関して拡張したものである。ここではこれらの拡張事項に関して説明を行う。詳細については、SCORM 1.2 規格書を参照のこと。

1) メタデータに関する拡張

SCORM ではメタデータを、マニフェストファイルの内部および外部のいずれにも記述することができる。外部に記述する場合は、マニフェストファイル中にメタデータが記述されたファイルの位置を記述する。

2) item に関する拡張

階層型コンテンツ構造の各ノードを表す item に対して、以下の追加が行われている。

- **Prerequisite** の追加. 該当する item を実行する前提条件を表す. 階層構造の中間ノードにも末端ノードにも適用される. これにより、簡易な学習順序制御が行える. ただし、SCORM 1.2 の前提条件によるシーケンシング制御は機能的に十分でなく、SCORM 2004 で本格的なシーケンシング機能が追加されている。
- **Max Time Allowed** の追加. 該当する item の学習制限時間を表す. 階層構造の末端ノードにだけ適用される. この item に対応する SCO は、マニフェストに記述された学習制限時間の値を RTE の `cmi.student_data.max_time_allowed` データモデル要素を使って読みだすことができる。
- **Time Limit Action** の追加. 該当する item の学習制限時間が越えた場合の動作を表す. 階層構造の末端ノードにだけ適用される. この item に対応する SCO は、マニフェストに記述された動作の値を RTE の `cmi.student_data.time_limit_action` データモデル要素を使って読みだすことができる。
- **Data From LMS** の追加. 該当する item の起動パラメータを表す. 階層構造の末端ノードにだけ適用される. この item に対応する SCO は、マニフェストに記述されたパラメータの値を RTE の `cmi.launch_data` データモデル要素を使って読みだすことができる。
- **Mastery Score** の追加. 該当する item の合格点を表す. 階層構造の末端ノードにだけ適用される. この item に対応する SCO は、マニフェストに記述された合格点の値を RTE の `cmi.student_data.mastery_score` データモデル要素を使って読みだすことができる。

3) file に対する追加

実際の学習資源に対する参照を行う file に対して、以下の追加が行われている。

- **SCORM Type** の追加. 学習資源の種別が SCO かアセットかを表す。

3.2.3.3 コンテンツパッケージの例

図 3.6 にコンテンツパッケージの例を示す。左がコンテンツパッケージのルートディレクトリである。このディレクトリにマニフェストファイル `imsmanifest.xml` やスキーマ定義ファイルが置かれている。ルートディレクトリにこれらのファイルを置く以外は、コンテンツパッケージ内のファイルやディレクトリの構成は自由である。このパッケージの場合は **Resources** というディレクトリを設け、その中に右のように SCO やアセットなどの学習資源を置いている。

図 3.6 にこのコンテンツパッケージのマニフェストファイルをオーサリングツール²で表示した画面を示す。左側のペインはコンテンツパッケージの内容を示している。右側のペインはマニフェストファイルの内容を示している。マニフェストファイルはオーガニゼーションセクション（上半分）とリソースセクション（下半分）に分かれている。それぞれの構成要素にはメタデータを付与することができる。

² ここではオーサリングツールとして RELOAD エディタを用いている。
<http://www.reload.ac.uk/new/editor.html>

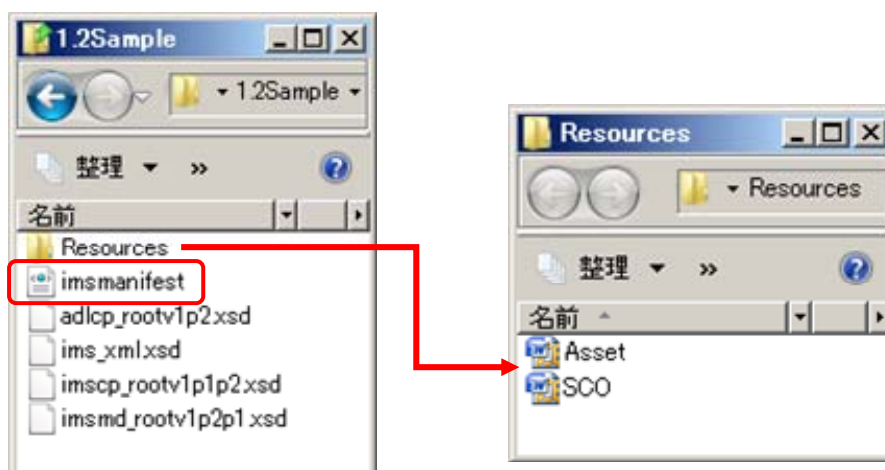


図 3.6 コンテンツパッケージの例

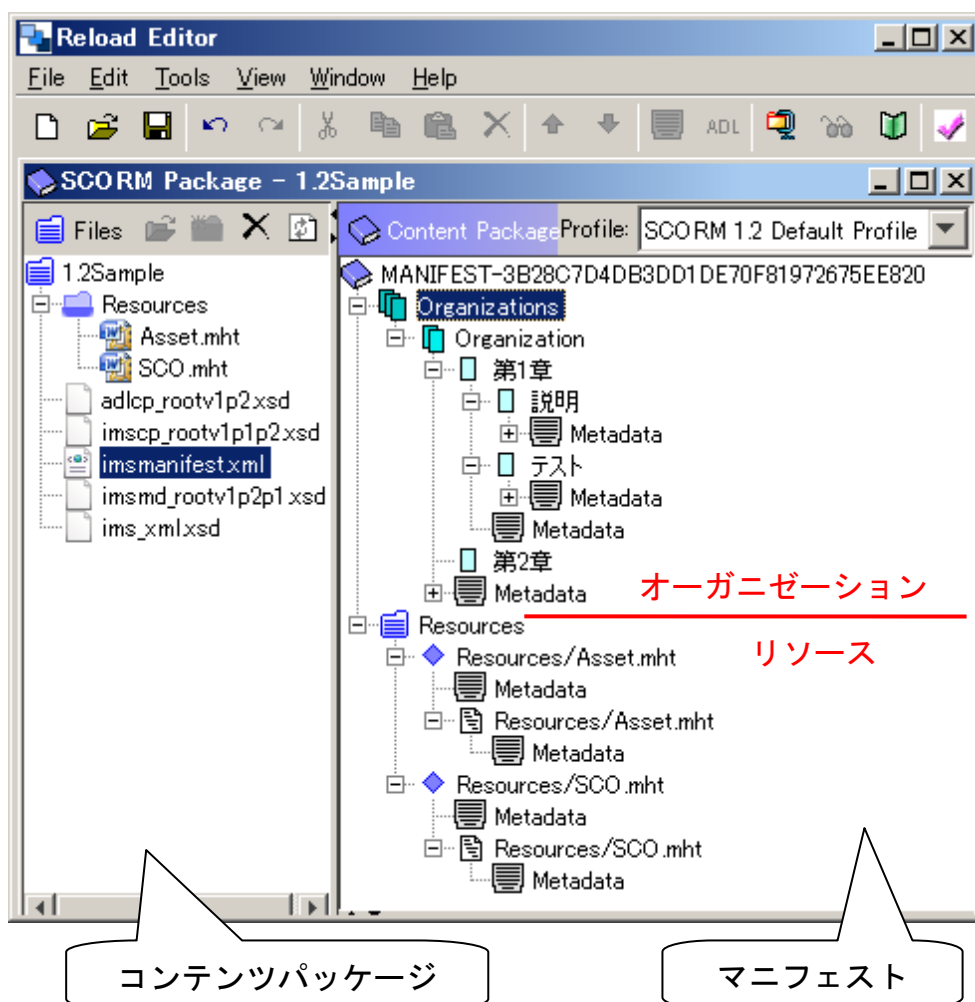


図 3.7 SCORM1.2 のマニフェストファイル編集画面

```

<?xml version="1.0" encoding="UTF-8"?>
<!--This is a Reload version 2.5.5 SCORM 1.2 Content Package document-->
<!--Spawned from the Reload Content Package Generator - http://www.reload.ac.uk-->
<manifest xmlns="http://www.imsglobal.org/xsd/imsdp_rootv1p2"
xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_rootv1p2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_rootv1p2"
identifier="MANIFEST-3B28C7D4DB3DD1DE70F81972675EE820"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imsdp_rootv1p2 imsgdp_rootv1p2.xsd
http://www.imsglobal.org/xsd/imsmd_rootv1p2p1 imsgmd_rootv1p2p1.xsd
http://www.adlnet.org/xsd/adlcp_rootv1p2 adlcp_rootv1p2.xsd">
  <organizations default="ORG-BFCEEDF05BB62BAC707ABDDA3B634EA1">
    <organization identifier="ORG-BFCEEDF05BB62BAC707ABDDA3B634EA1" structure="hierarchical">
      <title>Organization</title>
      <item identifier="ITEM-E6A557F12FF4B1411BFF60AD5117A77B" isvisible="true">
        <title>第1章</title>
        <item identifier="ITEM-932D1C8E5F573D98189D50BDA701CBE2" isvisible="true"
identifierref="RES-A5E22BB23FD2C7CB26CCA40116F767FD"> ——(1)
          <title>説明</title>
          <metadata>
            <schema>ADL SCORM</schema>
            <schemaversion>1.2</schemaversion>
            <imsmd:lom>
              <imsmd:general>
                <imsmd:title>
                  <imsmd:langstring xml:lang="en">SCORM1.2 サンプルアセット
                </imsmd:langstring>
              </imsmd:title>
              <imsmd:language>ja</imsmd:language>
            </imsmd:general>
          </imsmd:lom>
        </metadata>
      </item>
      <item identifier="ITEM-AAD0EB2F90307929273A01C8FAD768DA" isvisible="true"
identifierref="RES-04EC9CAE906477629D31D40E90055424"> ——(2)
        <title>テスト</title>
        <metadata>
          <schema>ADL SCORM</schema>
          <schemaversion>1.2</schemaversion>
          <imsmd:lom>
            <imsmd:general>
              <imsmd:title>
                <imsmd:langstring xml:lang="en">SCORM1.2 サンプル SCO</imsmd:langstring>
              </imsmd:title>
              <imsmd:language>ja</imsmd:language>
            </imsmd:general>
          </imsmd:lom>
        </metadata>
        <adlcp:maxtimeallowed>10:00</adlcp:maxtimeallowed> ——(3)
        <adlcp:timelimitaction>exit, no message</adlcp:timelimitaction>
        <adlcp:datafromlms>初期化データ</adlcp:datafromlms>
        <adlcp:masteryscore>80</adlcp:masteryscore>
      </item>
      <metadata>
        <imsmd:lom>
          <imsmd:general>
            <imsmd:title>
              <imsmd:langstring xml:lang="en">SCORM サンプル第1章</imsmd:langstring>
            </imsmd:title>
          </imsmd:general>
        </imsmd:lom>
      </metadata>
    </item>
    <item identifier="ITEM-6323C28B564E399605843BD3E8934946" isvisible="true">
      <title>第2章</title>
    </item>
    <metadata>
      <schema>ADL SCORM</schema>
      <schemaversion>1.2</schemaversion>
      <imsmd:lom>
        <imsmd:general>
          <imsmd:title>

```

```

    <imsmd:langstring xml:lang="en">SCORM1.2 サンプル</imsmd:langstring>
  </imsmd:title>
  <imsmd:language>ja</imsmd:language>
  <imsmd:keyword>
    <imsmd:langstring xml:lang="en">SCORM テキスト サンプル</imsmd:langstring>
  </imsmd:keyword>
  <imsmd:structure>
    <imsmd:source>
      <imsmd:langstring xml:lang="en">LOMv1.0</imsmd:langstring>
    </imsmd:source>
    <imsmd:value>
      <imsmd:langstring xml:lang="en">Hierarchical</imsmd:langstring>
    </imsmd:value>
  </imsmd:structure>
</imsmd:general>
<imsmd:lifecycle>
  <imsmd:status>
    <imsmd:source>
      <imsmd:langstring xml:lang="en">LOMv1.0</imsmd:langstring>
    </imsmd:source>
    <imsmd:value>
      <imsmd:langstring xml:lang="en">Final</imsmd:langstring>
    </imsmd:value>
  </imsmd:status>
  <imsmd:contribute>
    <imsmd:role>
      <imsmd:source>
        <imsmd:langstring xml:lang="en">LOMv1.0</imsmd:langstring>
      </imsmd:source>
      <imsmd:value>
        <imsmd:langstring xml:lang="en">Author</imsmd:langstring>
      </imsmd:value>
    </imsmd:role>
    <imsmd:centity>
      <imsmd:vcards>K.Nakabayashi</imsmd:vcards>
    </imsmd:centity>
  </imsmd:contribute>
</imsmd:lifecycle>
</imsmd:lom>
</metadata>
</organization>
</organizations>
<resources>
  <resource identifier="RES-A5E22BB23FD2C7CB26CCA40116F767FD" type="webcontent"
    href="Resources/Asset.mht" adlcp:scormtype="asset" > ——(1')
    <metadata />
    <file href="Resources/Asset.mht">
      <metadata />
    </file>
  </resource>
  <resource identifier="RES-04EC9CAE906477629D31D40E90055424" type="webcontent"
    href="Resources/SC0.mht" adlcp:scormtype="SC0" > ——(2')
    <metadata />
    <file href="Resources/SC0.mht">
      <metadata />
    </file>
  </resource>
</resources>
</manifest>

```

図 3.8 SCORM1.2 マニフェストファイルの例

図 3.8 に、図 3.7 のマニフェストファイルの XML ソースコードを示す。コンテンツの階層構造は、<organization>タグの配下の入れ子になった<item>タグで表わされている。階層の各ノードに関する情報は<item>から</item>の間に記述されている。例えば、そのノードの題名<title>やメタデータ<metadata>などである。ソースコード中の(3)の部分は、(2)の「テスト」という題名のノードに対して、SCORM1.2 固有の Max Time Allowed や Mastery Score の指定を行って

いる。

階層の末端のノードは対応する SCO やアセットなどの学習資源を有する。この関係は<item> タグの identifierref を用いて表現される。すなわち、identifierref で指定された識別子と等しい識別子を有する学習資源が<resources>から</resources>の間のリソースセクションに記述されている。例えば、(1)の<item>と(1')の<resource>、(2)の<item>と(2')の<resource>は、<item>の identifierref と<resource>の identifier が等しく、<item>で示したノードで<resource>で指定した学習資源を用いることを示している。

<resource>の中では、実際に用いる物理ファイルの指定が記述されている。例えば(1')の<resource>では、href="Resources/Asset.mht" adlcp:scormtype="asset" という記述により、コンテンツパッケージ中の Resources ディレクトリの Asset.mht をアセットとして使用することが分かる。この href 指定が上記のように相対パスの場合はコンテンツパッケージ中の物理ファイルが、http: で始まる絶対パスの場合はインターネット上の外部ファイルが学習資源として用いられる。

3.3 ランタイム環境

SCORM1.2 の RTE (Run-Time Environment : ランタイム環境) 規格は、図 3.9 に示すように、クライアント側のブラウザで動作する SCO(Sharable Content Object)とプラットフォームの間の規格である。RTE 規格は SCO とプラットフォームの間のインターフェースとして、

- 起動 (Launch)
- API (Application Programming Interface)
- データモデル (Data Model)

の 3 つを規定している。

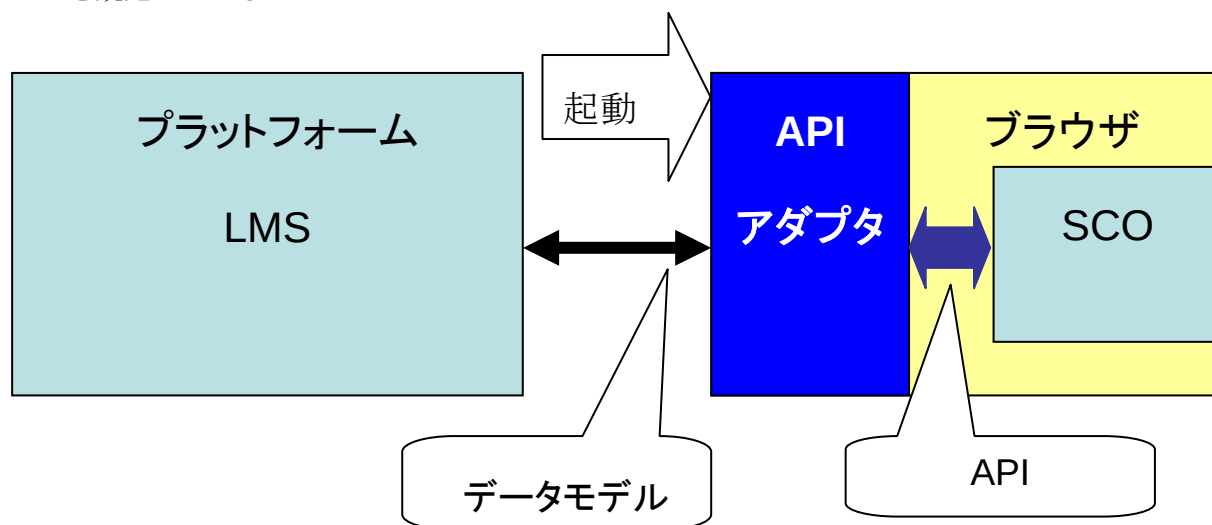


図 3.9 SCORM1.2 ランタイム環境

「起動」、「API」、「データモデル」とは具体的に何を指しているか、図 3.10 に SCORM1.2 コンテンツの表示例を示す。この例では、左フレームの目次や上フレームの前後ボタンをクリックすると、他の SCO やアセットが表示される。このように新しい SCO を表示する動作が「起動」である。また、演習問題の SCO の中にある「解答」ボタンをクリックすると、演習問題の解答の採点が行われ、結果が LMS に送信される。このとき、SCO が結果を LMS に送信するために使用する関数が「API」、送信することのできるデータの種類や値の範囲を定めたものが「データモデル」である。

以下、「起動」、「API」、「データモデル」のそれぞれについて説明する。

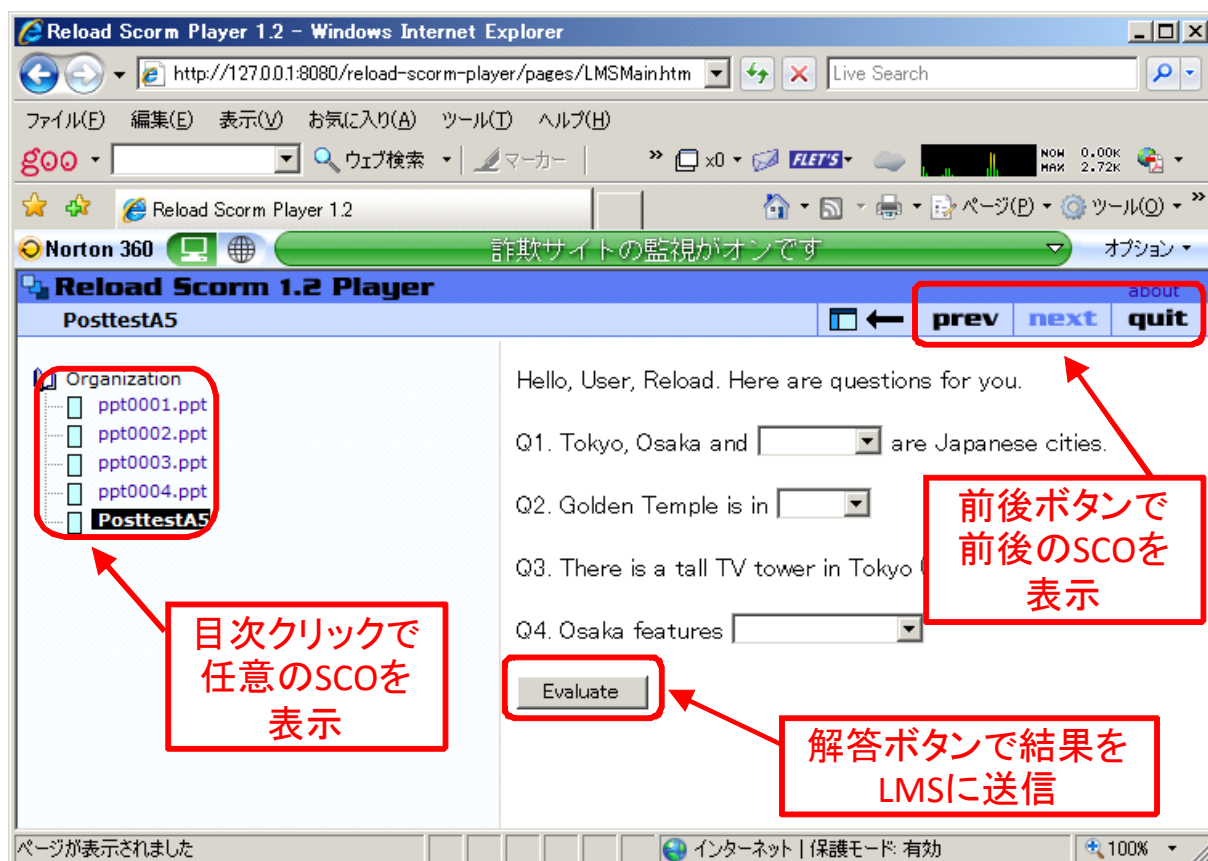


図 3.10 SCORM1.2 コンテンツ表示例

3.3.1 起動

図 3.10 で目次や前後ボタンをクリックすると、その要求は LMS に送られ、LMS はその要求に対応した SCO を起動する。SCORM1.2 においてどのような方法においても SCO を起動するのはすべて LMS の責任範囲である。つまり、LMS はマニフェストファイルに書かれた URL を使って SCO を起動しなくてはならない。

具体的な実現の仕組みをみてみよう。図 3.10 の画面構成において、右フレームの演習問題はコンテンツ作成者が作った SCO である。一方、左フレームの目次や上フレームの前後ボタンはコンテンツ作成者が作ったものではなく、LMS が表示したものである。つまり、図 3.10 の画面は以下のような HTML で構成されていると考えられる。

```
<HTML>
<HEAD> <TITLE>Sample SCORM Screen</TITLE> </HEAD>
<FRAMESET ROWS="10%,*">
  <!-- ボタンフレーム -->
  <FRAME NAME="Button" SRC="Button_URL">
  <FRAMESET COLS="30%,*">
    <!-- 目次フレーム -->
    <FRAME NAME="Index" SRC="Index_URL">
    <!-- SCO フレーム。ここに SCO が読み込まれて起動される -->
    <FRAME NAME="SCO" SRC="SCO_URL">
  </FRAMESET>
</FRAMESET>
</HTML>
```

図 3.11 SCORM1.2 コンテンツを表示する HTML のスケルトン

図 3.10 で目次や前後ボタンをクリックすると、その要求は LMS に送られ、その応答として LMS は再び図 3.11 のような HTML を生成してブラウザに送り返す。この中には、図 3.11 の網かけ部分の<FRAME>タグのように、マニフェストファイルに記述された SCO の URL がフレームのソースとして埋め込まれている。(このとき LMS はマニフェストファイルに記述された相対 URL を必要に応じて絶対 URL に変換する。)ブラウザは、図 3.11 の HTML を解釈し、<FRAME>タグの SRC 属性に記述された URL をアクセスして SCO を読み込んで表示する。これが SCO の「起動」と呼ばれる動作である。

LMS から起動できるのは SCO のほかに前述したアセットがある。アセットはそれ自身、LMS と通信しないため、起動するだけである。

注) 上記の画面の構成の細部は LMS によって異なる。

3.3.2 API

SCO は LMS からデータを取得したり、LMS にデータを送ることができる。LMS は受信したデータを学習履歴としてデータベースに格納する。また、SCO は LMS から取得したデータを使って動作を変えることができる。送受信できるデータの種別は次節に回し、ここではデータの送受信を行うための仕組みを見ていく。

3.3.2.1 API 関数

SCO と LMS の間の通信はすべて SCO の主導で行われる。つまり、SCO がデータを送受信するために RTE 規格で定められた JavaScript（標準規格では ECMAScript）の関数を呼び出すことによって通信が実行される。これらの関数を API 関数と呼ぶ。

表 3.1 API 関数一覧

SCORM 1.2	機能
LMSInitialize	RTE セッションを開始する。
LMSFinish	RTE セッションを終了する。
LMSGetValue	LMS からデータを取得する。
LMSSetValue	LMS に送るデータを設定する。
LMSCommit	LMS にデータを格納させる。
LMSGetLastError	直前の API 関数のエラーコードを取得する。
LMSGetErrorString	エラーメッセージを取得する。
LMSGetErrorDiagnostic	LMS 固有の診断メッセージを取得する。

API 関数の引数や戻り値、動作の詳細については、SCORM1.2 RTE 仕様書を参照のこと。
API 関数を使用する上で特に注意すべき点をいくつか挙げておく。

1) API 関数の引数と戻り値

API 関数の引数と戻り値はすべて JavaScript の文字列型となる。得点など意味的に数値の型を持つデータも、API 関数への引き渡しは文字列型に変換して行う。

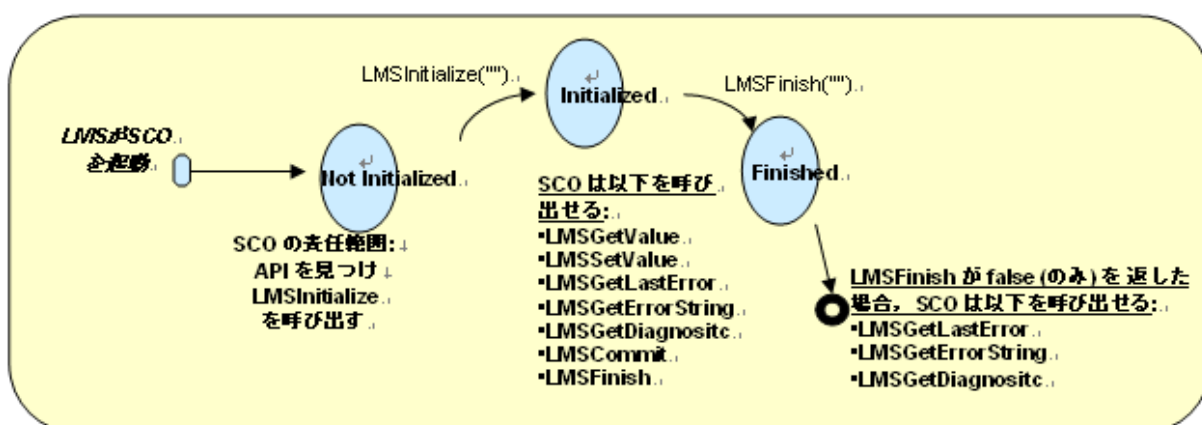
2) API 関数の呼び出し順序

RTE セッションは、SCO が起動されたのち、LMSInitialize()の呼び出しによって始まり、LMSFinish()の呼び出しで終了する。この間、どの時点でどの API 関数を呼び出せるかは厳密に定められている。たとえば、LMSInitialize()の前にいきなり LMSGetValue()を呼び出したり、一旦 LMSFinish()でセッションを終了した後、再度 LMSInitialize()を呼び出すようなことはできない。

3) SCO から LMS へのデータ送信

LMSSetValue()は SCO から LMS にデータを送信するために使用するが、呼び出した直後にデータが LMS に必ず送信されるとは限らない。SetValue()されたデータは、一旦ブ

ブラウザ側のバッファに蓄えられ、後で他のデータと一緒に LMS に送信されることもある。どのタイミングで実際にデータを送信するかは LMS の実装に依存しており、各ベンダによってまちまちである。ただし、LMSCommit()が実行された場合は、必ず実際に LMS にデータを送ることが規格で定められている。従って、SCO 実行中に確実にデータを格納したい場合は LMSCommit()を呼び出す。また、LMSFinish()では LMSCommit()と同じ動作を実行してからセッションを終了することになっている。C 言語をご存じの方であれば、LMSFinish()と LMSCommit()は、fclose()と fflush()の関係に対応することがわかる。



API アダプタの状態遷移

3.3.2.2 API アダプタと FindAPI

前節で、「SCO が API 関数を呼び出す」と述べてきた。SCORM1.2 規格では、API 関数を実装した「API アダプタ」を LMS が用意し、教材作成者が作成した SCO 中の JavaScript のプログラムから、API 関数を呼び出す。サーバー側の LMS がクライアント側のブラウザ上に API 関数を実装した API アダプタを用意するためには、SCO を起動する際、図 3.12 のように、あるフレームに SCO を、他のフレームに LMS 側で用意する「API アダプタ」を送り込む。

で、そうすることで SCO から API 関数を

`APIframe.LMSGetValue()`

のようにして呼び出すことができるようになる。

SCO の起動から、SCO が API アダプタを見つけて API 関数を用いた通信を開始するまでを、再度順を追って説明する。

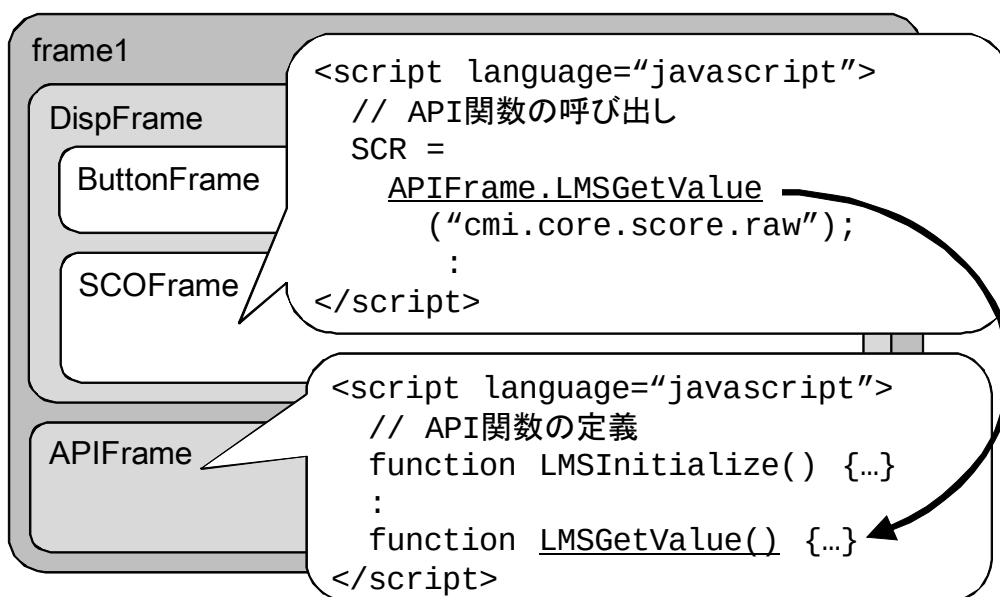


図 3.12 他のフレームにある関数の呼び出し

```

<HTML>
<HEAD> <TITLE>Sample SCORM Screen</TITLE> </HEAD>
<FRAMESET ROWS="100%,0%">
  <FRAMESET ROWS="10%,*">
    <!-- ボタンフレーム -->
    <FRAME NAME="Button" SRC="Button_URL">
    <FRAMESET COLS="30%,*">
      <!-- 目次フレーム -->
      <FRAME NAME="Index" SRC="Index_URL">
      <!-- SCO フレーム. ここに SCO が読み込まれて起動される -->
      <FRAME NAME="SCO" SRC="SCO_URL">
    </FRAMESET>
  <!-- API フレーム. ここに API アダプタを定義した Javascript を読み込む -->
  <FRAME NAME="API" SRC="API_Adaptor_URL">
</FRAMESET>
</HTML>

```

図 3.13 API アダプタの定義を含む SCORM コンテンツ表示用 HTML

- 1) LMS は起動の際に、図 3.12 に相当するようなフレームを定義した HTML をブラウザに送り込む。この例を図 3.13 に示す。この HTML は図 3.11 に示したものと同様であるが、網かけの API アダプタのフレームが追加されている。このとき、このフレームの名前は SCORM 1.2 の場合 "API" でなくてはならない。API アダプタのフレームには API 関数を定義した Javascript のプログラムが読み込まれる。
- 2) フレームに読み込まれた SCO は "API" という名前のフレームを見つける操作を行う。

これが Find API である。SCO と API アダプタのフレームのフレーム階層内での相対的な位置関係が固定的に決まっているか、あるいは、トップのフレームからの API アダプタのフレームの位置関係が決まっていればこのような操作は不要だが、固定的にしてしまうと LMS ベンダの設計の自由度を奪ってしまうため、このような操作が規定されている。SCO は起動されたら必ずこの操作を実行して、API アダプタのフレームを見つけなくてはならない。

- 3) API アダプタのフレームが見つかり、そのフレームを APIFrame という Javascript の変数で参照するようにしたとする。これ以降、API 関数として規定された LMSInitialize, LMSGetValue などの関数は、

```
APIframe.LMSInitialize()
```

```
APIframe.LMSGetValue()
```

という形で、SCO 中のプログラムから呼び出すことが可能になる。

なお、API アダプタの名称が SCORM 1.2 と SCORM 2004 で異なるのには意味がある。これは SCORM 1.2 用に作成したコンテンツを SCORM 2004 環境で使用する上で重要なポイントである。また、図 3.12 に示した HTML は動作環境によってタイミングの問題を生じることがある。これは LMS 側の問題で、コンテンツ作成上は気にする必要はないが、トラブルを解決する際にはこの知識が必要になることがある。

また、API アダプタは SCO に対して JavaScript の API 関数を提供するが、API アダプタ自体は LMS と通信を行うために Java Applet や Ajax で実装される場合が多い。これらの実装は SCO からは隠蔽されており、コンテンツを作成する際には気にする必要はない。しかし、Java Applet で作成された API アダプタを用いている LMS では、ブラウザ側に Java の実行環境が必要となるので、利用者パソコンの環境によっては制限が生じる可能性がある。

3.3.3 データモデル

ここまで API 関数を使って SCO と LMS がデータをやり取りする仕組みを見てきた。それでは実際にやりとりするデータにはどのようなものがあるのか？ SCORM1.2 では、SCO と LMS がやり取りすることのできるデータの種類や名称、書式を「データモデル」として規定している。ここでは、データを送受信する方法、主要なデータモデル要素、データモデルに関するいくつかのルールなどを説明する。

3.3.3.1 データの送信と受信

データの送受信には API 関数の `LMSGetValue` と `LMSSetValue` (SCORM 2004 では `GetValue` と `SetValue`) を用いる。

```
dataElementValue = LMSGetValue(dataElementName);
status = LMSSetValue(dataElementName, dataElementValue);
```

`dataElementName` がデータモデル要素の名前を表す文字列、`dataElementValue` がデータモデル要素の値である。例を以下に示す。

```
// lesson_status を取得
st = LMSGetValue("cmi.core.lesson_status");
if (st == "passed") { ...

// 得点を設定
status = LMSSetValue("cmi.core.score.raw", "75");
if (status == "ture") { ...
```

前節で述べたように API 関数の引数、戻り値は JavaScript の文字列である。

3.3.3.2 データの種別

表 3.2 にデータモデル要素の一覧を示す。データモデル要素の名前は、“cmi.core.lesson_status”のように“.”で区切ったキーワードを並べた形になっている。“cmi.core”というデータモデル要素のグループに複数のデータモデル要素が属する、といった階層構造になっている。表 3.2 に挙げたものの中にも、単一のデータモデル要素と、データモデル要素のグループに対応しているものがある。例えば、“cmi.core.student_id”は単一の要素であるが、“cmi.core.score”は要素のグループとなっていて、“cmi.core.score.raw”、“cmi.core.score.min”、“cmi.core.score.max”の3つのデータモデル要素が含まれる。

表 3.2 データモデル要素一覧

SCORM 1.2	意味
cmi.core.student_id	学習者 ID
cmi.core.student_name	学習者名
cmi.core.lesson_location	SCO 中の位置
cmi.core.credit	評価を行うか否か
cmi.core.lesson_status	進捗状態
cmi.core.entry	SCO 開始時の状態
cmi.core.score	得点
cmi.core.total_time	総学習時間
cmi.core.lesson_mode	SCO の動作モード
cmi.core.exit	SCO 終了時の状態
cmi.core.session_time	セッション学習時間
cmi.suspend_data	中断時保持データ
cmi.launch_data	起動時設定データ
cmi.comments	学習者のコメント
cmi.comments_from_lms	LMS からのコメント
cmi.objectives	学習目標
cmi.student_data.mastery_score	合格点
cmi.student_data.max_time_allowed	学習許容時間
cmi.student_data.time_limit_action	時間制限超過時動作
cmi.student_preference	学習者選好
cmi.interactions	演習問題等の記録

データモデル要素には、各々、送受信の方向（SCO→LMS, LMS→SCO, 双方向）、データ型、必須・任意の別（SCORM 1.2 の場合）という属性がある。このような情報は SCORM の規格書で調べることができる。図 3.14 は SCORM 1.2 の “cmi.core.student_id” の例である。“LMS Mandatry” が “Yes” であることから、これは必須要素で LMS 側ではこの要素を実装しなくてはならないことがわかる。この欄が “No” の場合はこの要素を実装していない LMS でも規格に準拠しているとみなされる。次の欄はデータ型でこの場合は “CMIIIdentifier”（識別子型）となっている。データ型には、文字列型、論理型、整数型などがあるが詳しくは参考文献[3]の 3.4.5 を見てほしい。次の欄はデータの送受信方向で、この場合は SCO から LMS に対して “Read Only”, つまり SCO がデータを読み出すだけの動作が許されていることがわかる。SCO から LMS への書き込みだけの場合はこの欄が “Write Only”, 読み書き双方向の場合は “Read Write” となる。

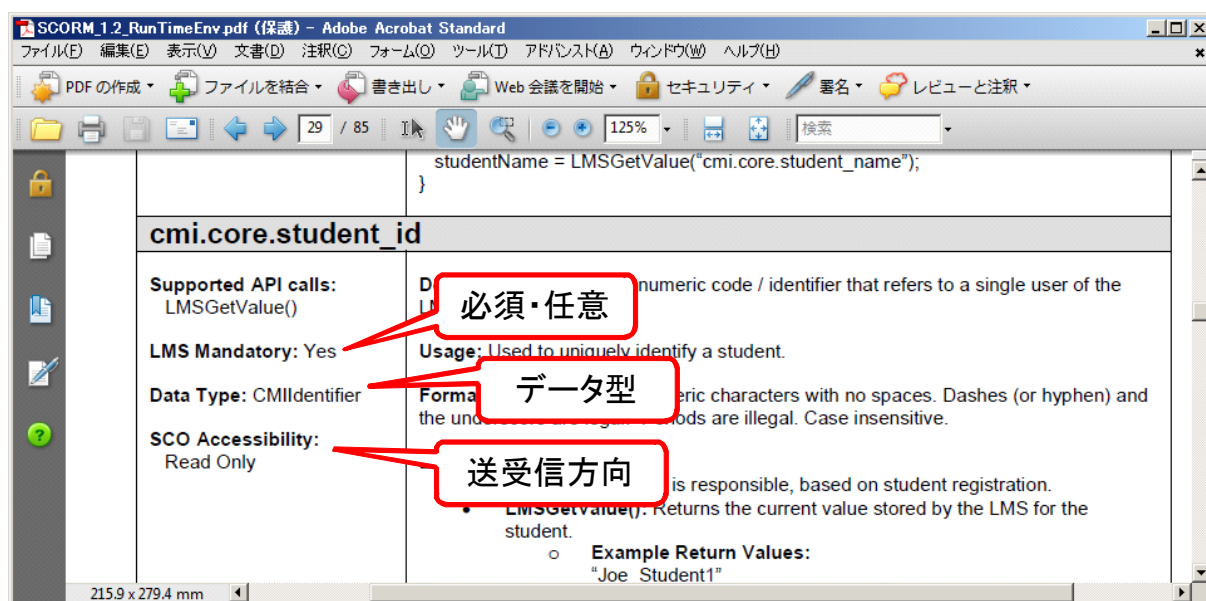


図 3.14 SCORM1.2 ランタイム環境仕様書のデータモデル要素の例

3.3.3.3 主要なデータモデル要素

1) 得点と学習状態

データモデル要素のうちで最も頻繁に使用されるのが、得点と学習状態に関するデータモデル要素であろう。例えば図 3.15 のような SCO で演習問題全体の得点や合否を記録する場合、これらのデータモデル要素を用いる。SCO で演習問題などを行った結果の得点は SCORM 1.2 では “cmi.core.score.raw”, SCORM 2004 では “cmi.score.scaled” を用いる (SCORM 2004 でも “cmi.core.score.raw” は利用可能であるが、シーケンシングの目的に使うためには “cmi.score.scaled” を使わなくてはならない)。これらはいずれも双方向に通信可能なデータモデル要素で、SCO は LMS に送信した値を後で読み出すことができる。

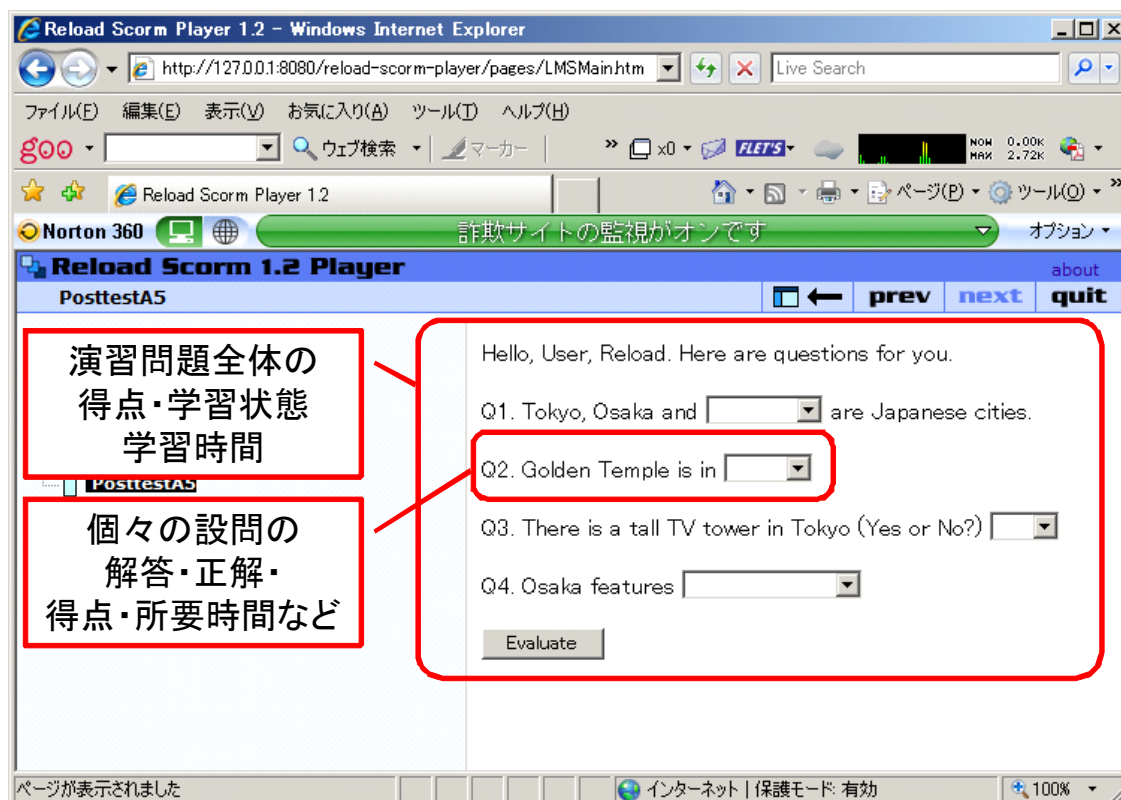


図 3.15 演習問題 SCO の例

学習状態は SCORM 1.2 では “cmi.core.lesson_status” のデータモデル要素を用いる。“cmi.core.lesson_status” は、進捗と習得に関する概念を双方向に通信可能なデータモデル要素で、SCO は LMS に送信した値を後で読み出すことができる。得点と学習状態の関係で注意しなくてはならないのは、マニフェストファイルに合格点が設定されている場合、LMS が合格点と得点の比較を行い、SCO が送信した学習状態を上書きする動作を行うことである。

2) 学習時間

図 3.15 のような SCO の学習時間を記録するためのデータモデル要素がある。これらは、セッション学習時間と総学習時間である。

ある教材の学習において、ひとつの SCO を学習してから、他の SCO を学習し、また最初の SCO を学習することができる。ひとつの SCO を起動してから終了までが 1 回のセッションである。例えば、図 3.15 の SCO を起動して、演習問題に解答し「Evaluate」を押すまでがこれに該当する。セッション学習時間はこの所要時間である。SCO はセッション学習時間を計測して LMS に送信する。セッション学習時間は、SCORM 1.2 では cmi.core.session_time, を用いる。(SCORM 2004 では cmi.session_time)

一方、LMS は複数のセッション学習時間を累積して総学習時間とする。SCO は LMS からこれまでの総学習時間を読み出すことができる。例えば、図 3.15 の SCO を再度起動したと

き、SCO はこれまでの総学習時間を知ることができる。総学習時間は、SCORM 1.2 では `cmi.core.total_time` である。(SCORM 2004 では `cmi.total_time`)

総学習時間とセッション学習時間に関して注意しなければならないのは、1 回のセッションでセッション学習時間を複数回送信した場合、総学習時間に累積されるのは、常に最後に送信した値だけだということである。

(参考文献[4]の 3.4.2 を参照)

3) 学習履歴の記録

図 3.15 のような演習問題の SCO で、個々の設問の解答・正解・得点などを履歴として記録したい場合がある。あるいは、シミュレーション機能を有する SCO で学習者のひとつひとつの操作手順を記録したい場合がある。このような場合、`cmi.interactions` を用いて学習履歴を LMS に送信して記録することができる。

`cmi.interactions` は `collection` (集合) 型³のデータモデル要素である。例えば、図 3.15 のような SCO の中のひとつの設問に関する解答や得点をまとめてひとつの「レコード」とし、このようなレコードを複数集めた集合を扱うことができる。図 3.15 に対しては、例えば以下のようなレコードの集合が考えられる。

レコード 1 = {正解 = “Kyoto”, 解答 = “Kyoto”, 正誤 = “correct”, 得点 = 20}

レコード 2 = {正解 = “Kyoto”, 解答 = “Oosaka”, 正誤 = “wrong”, 得点 = 0}

(`cmi.interactions` でレコードに記録することのできるデータモデル要素の詳細については、参考文献[2], [3]を参照)

4) 学習目標

SCO 全体としては先に述べた得点や学習状態データモデル要素を用いて、学習者の習得状況を記録することができる。さらに、ひとつの SCO が複数の学習目標と関連している場合も考えられる。例えば、図 3.15 の設問では、

学習目標 1 = 「日本の都市の名前を挙げることができる」

学習目標 2 = 「京都の名所を挙げることができる」

学習目標 3 = 「東京の名所を挙げることができる」

学習目標 4 = 「大阪の名所を挙げることができる」

という学習目標が設定されていて、それぞれの習得状況を確認するために設問 1 から 4 が設けられているかもしれない。あるいは、

学習目標 5 = 「日本の建築物を挙げることができる」

という学習目標が設定されていて、設問 2 と 3 がこの学習目標の習得状況の確認に用いられているかもしれない。演習問題と学習目標をどのように関連付けるかは教材設計者の設計意図次第である。

³ `collection` (集合) 型は SCORM 2004 での呼び方で、SCORM 1.2 では `array` (配列) ないし `list` (リスト) という呼び方をしていた。いずれも複数のレコードからなるデータの集まりの意味である。

このような学習目標の習得状況は、`cmi.objectives` を用いて、SCO から LMS に記録したり、SCO から読み出したりすることができる。`cmi.objectives` は `cmi.interactions` と同様に `collection` 型のデータモデル要素で、ひとつのレコードがひとつの学習目標に対応する。個々の学習目標に対しては、得点、習得状態などを記録することができる。

(詳細については、SCORM1.2 解説書を参照)

5) 中断状態の保持

SCO をいったん終了してセッションを終え、再度セッションを開始したとき、前回のセッションの状態を再現したい場合がある。例えば、複数の HTML ページを含む SCO で、最後に見ていた HTML ページを再開時に表示したり、学習者が回答欄に入力した情報を再開時に復元するといったような場合である。

このような目的のために、`cmi.core.lesson_location` (SCORM 2004 では `cmi.location`)、`cmi.suspend_data` を用いることができる。これらのデータモデル要素は SCO から LMS に記録したり、SCO から読み出したりすることができる。

6) 起動パラメータ

SCO は SCORM 教材における再利用の単位であり、ひとつの教材の中で繰り返し利用したり、複数の教材で使いまわすことができる。例えば、一つの演習問題の SCO を、ひとつの教材の中で、事前テストと事後テストに用いたり、同じ分野でも異なるレベルの教材で用いることができる。

このように SCO を再利用するとき、起動パラメータによって SCO の動作を変えたいという場合がある。例えば、SCO を事前テストと事後テストに用いるとき、事前テストでは合格点は低くてよいが、事後テストでは合格点を高くしたい、という場合がある。また、出題する設問の数や順番を実行時に変化可能な SCO であれば、事前テストと事後テストで出題する設問の難易度を変えたい、というような場合もある。

このような目的のために用いることのできるデータモデル要素が、`cmi.launch_data` と `cmi.student_data.mastery_score` (SCORM 2004 では `cmi.scaled_passing_score`) である。これらの値はいずれも教材作成者がマニフェストファイルの `item` タグの中に記述することができる。SCO は起動後、LMS からこれらの値を読み出し、その後の動作に反映することができる。

`cmi.launch_data` は文字列で、これをどのように利用するかは教材作成者が決めればよい。一方、`cmi.student_data.mastery_score` (`cmi.scaled_passing_score`) は合格点を表す数値であり、先に述べた LMS が学習状態を決定する動作とも関係するので注意が必要である。

3.3.3.4 特殊なデータモデル要素

1) 子のデータモデル要素

「データの種別」で述べたように、データモデル要素は階層的に決められており、その名前

は、“cmi.core.lesson_status”のように“.”で区切ったキーワードを並べた形になっている。“cmi.core”はデータモデル要素のグループを表しており、その中に複数の子のデータモデル要素が存在する。

データモデル要素のグループにどのようなデータモデル要素が含まれているかを知るために“_children”というキーワードが用意されている。例えば、SCORM 1.2で、

```
childElements = LMSGetValue("cmi.core.score._children");
```

とすると、必須データモデル要素のみをサポートしている LMS では、childElements は“raw”に、すべてのデータモデル要素をサポートしている LMS では“raw,min,max”となる。これを用いると、

```
childElements = LMSGetValue("cmi.core.score._children");
if (childElements.IndexOf("min") != -1) {
    // Use "cmi.core.score.min"
}
```

のように、LMS でサポートされているデータモデル要素に対応したプログラムを書くことができる。

2) 集合型データの扱い

データモデル要素の中には, `cmi.interactions` や `cmi.objectives` のように集合型のものがある. 集合型のデータモデル要素では, データモデル要素に番号を付けて個々のレコードを区別する. 番号は 0 から始まる整数値である.

さらに `cmi.interactions` や `cmi.objectives` はデータモデル要素のグループなので, 各レコードはさらに子のデータモデル要素を含んでいる. この関係は以下のように書くことができる.

cmi.interactions	.0	.learner_responce		0 番目の設問の解答
		.correct_responces	.0.pattern	0 番目の設問の 0 番目の正解
			.1.pattern	0 番目の設問の 1 番目の正解
		:		
		.result		0 番目の設問の判定結果
	:			
	.3	.learner_responce		3 番目の設問の解答
		.correct_responces	.0.pattern	3 番目の設問の 0 番目の正解
			:	
			.2.pattern	3 番目の設問の 2 番目の正解
		:		
		.result		3 番目の設問の判定結果
	:			

図 3.16 集合型データの要素の関係

`cmi.interactions.0` が 0 番目の設問に対するレコードを表している. そのレコードの中で, 0 番目の設問の解答は `cmi.interactions.0. learner_responce`, 0 番目の設問の判定結果は `cmi.interactions.0.result` で表す. 設問に対する正解は複数あり得るので, 正解のデータモデル要素 `cmi.interactions.0.correct_responces` はさらに集合型のデータモデル要素となっている. これを用いて, 例えば, 0 番目の設問の 1 番目の正解は `cmi.interactions.0.correct_responces.1.pattern` で表す.

集合型のデータモデル要素に何個のレコードが含まれているかを知るために “_count” というキーワードが用意されている. SCO は LMS からこの値を読み出すことで, 記録されているレコードの数を知ることができる. この使用例を以下に示す.

```
recordCount = LMSGetValue("cmi.interactions._count");
nextRecord = "cmi.interactions" + recordCount;
LMSSetValue(nextRecord + ".result", "correct");
LMSSetValue(nextRecord + ".weighting", "10");
```

この例では、新しいレコードを追加するために“_count”を用いて、新しいレコードの番号を設定している。レコードの番号が 0 から始まるので、例えば、0 番から 9 番までレコードが存在するとき“_count”の値は 10, つまり、新しく追加するレコードの番号に等しくなる。

4. SCORM 2004

4.1 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。

4.1.1 第1編：SCORM 概要

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。

4.1.2 第2編：コンテンツモデル

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。

4.1.3 第3編：ランタイム環境

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。

4.1.4 第4編：シーケンシング&ナビゲーション

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。

4.2 コンテンツモデル

SCORM 2004 のコンテンツ構造は SCORM 1.2 と同様、階層型となっている。SCORM 1.2 ではコンテンツアグリゲーションモデルという用語が使われていたが、SCORM 2004 ではコンテンツモデルという用語を用いている。実体の基本的な考え方は同一である。以下では、SCORM 1.2 との差分を中心に SCORM 2004 のコンテンツモデルの説明を行う。図 4.1 に SCORM 2004 コンテンツモデルの全体を示す。

4.2.1 コンテンツモデル構成要素

SCORM 2004 のコンテンツモデル構成要素は、アセット、SCO、アクティビティ、コンテンツオーガニゼーション、コンテンツアグリゲーションである。このうち、アセット、SCO は SCORM 1.2 と同様の概念を表している。SCORM 1.2 でコンテンツアグリゲーションと呼ばれていたものは、アクティビティ、コンテンツオーガニゼーション、コンテンツアグリゲーションの 3 つの概念として明確化された。

4.2.1.1 アセット (Asset)

アセットは SCORM コンテンツの最小単位で、テキスト、試験問題、音声、画像、動画、Flash コンテンツ、HTML ページなどである。SCORM 1.2 のアセットと同様の概念である。

4.2.1.2 SCO

SCO (Sharable Content Object) は、複数のアセットを集めたコンテンツである。アセットとの違いは、SCORM ランタイム環境 (Run-time Environment, RTE) による LMS との通信機能を持つことである。SCORM 1.2 の SCO と同様の概念である。RTE に関する各種の制約や必要条件も SCORM 1.2 と同様である。

4.2.1.3 アクティビティ (Activity)

アクティビティは、階層型コンテンツの各ノードに対応する。末端ノードの場合は、SCO やアセットなどの学習資源に対応付けられており、それ以外の場合は、子のアクティビティを集約したものがアクティビティである。アクティビティは、教科書の章節項や、コース・モジュールなどに相当するものと考えることができる。

4.2.1.4 コンテンツオーガニゼーション (Content Organization)

コンテンツオーガニゼーションは階層型コンテンツ全体に相当し、各アクティビティがお互いにどのように関係するかを示すものである。コンテンツオーガニゼーションの中では、シーケン

シング情報が各アクティビティに対して記述される。LMS は実行時にシーケンシング情報を解釈して、学習資源をどのような順番で提示するかを決定する。SCORM 1.2 では十分なシーケンシング機能が提供されていなかったが、SCORM 2004 ではシーケンシング&ナビゲーション規格で、新たなシーケンシング機能が定義されている。

4.2.1.5 コンテンツアグリゲーション (Content Aggregation)

コンテンツアグリゲーションは、学習コンテンツ全体に対応し、物理的にはコンテンツパッケージに該当する。従って、コンテンツアグリゲーションはコンテンツ全体の構造を規定し、オーサリングツールや LMS の間でのコンテンツのやり取りの対象となる。

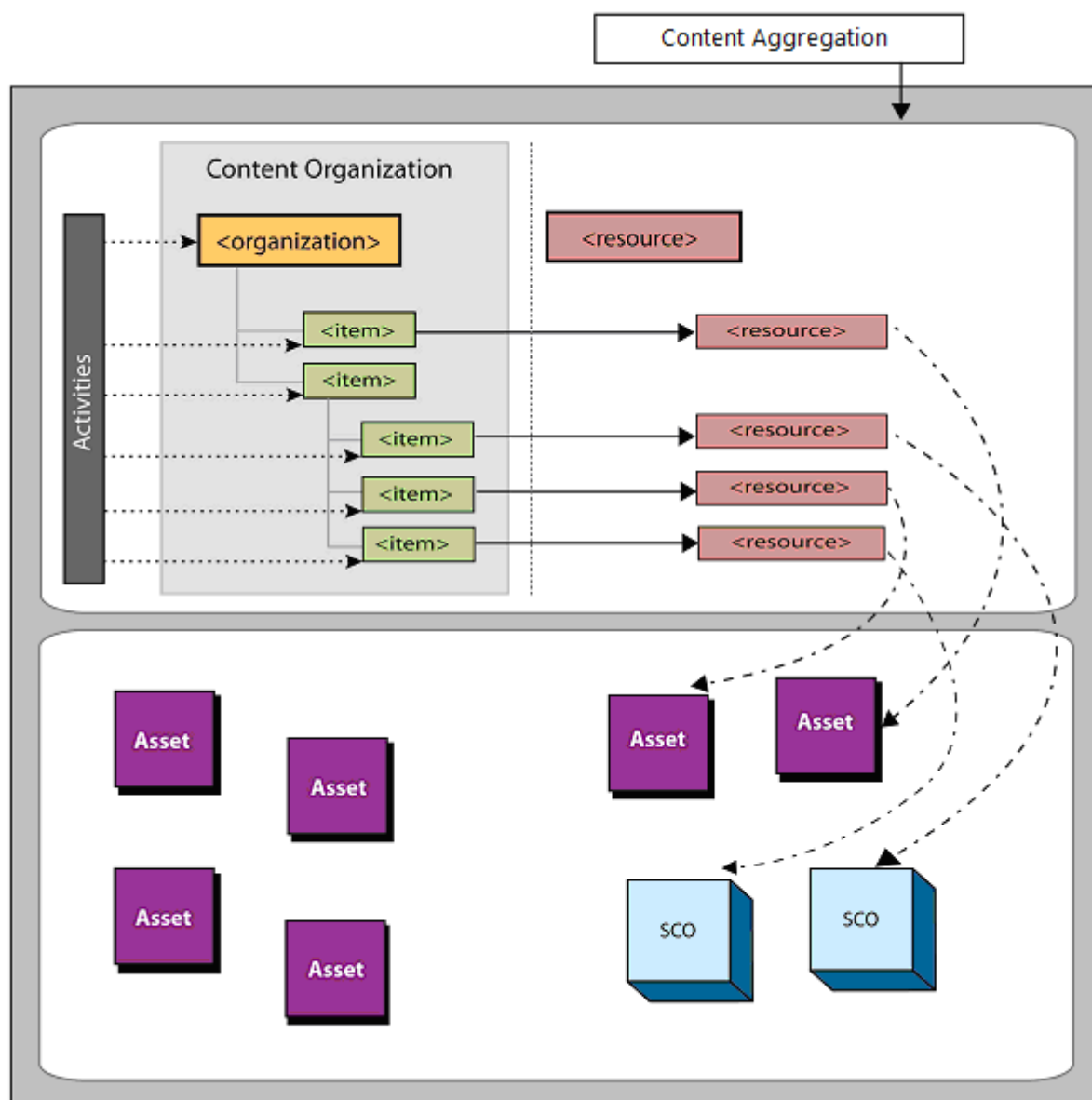


図 4.1 SCORM 2004 コンテンツモデル

4.2.2 メタデータ

メタデータの扱いは SCORM 1.2 と同様である。SCORM 2004 のメタデータは IEEE LTSC で規定された LOM 規格に基づいている。SCORM 2004 ではコンテンツモデル構成要素、すなわち、アセット、SCO、アクティビティ、コンテンツオーガニゼーション、コンテンツアグリゲーションのそれぞれに対してメタデータを付与することができる。メタデータは、マニフェストファイルの内部および外部のいずれにも記述することができる。

4.2.3 SCORM コンテンツパッケージング

SCORM コンテンツパッケージングは、IMS コンテンツパッケージング規格に基づいており、SCORM 固有の事項（SCO、アセットの表現、など）のために拡張されている。SCORM 2004 のコンテンツパッケージングは、基本的に SCORM1.2 と同一であり、マニフェストファイルにシーケンシング情報を記述するようになった点が最も大きな相違である。

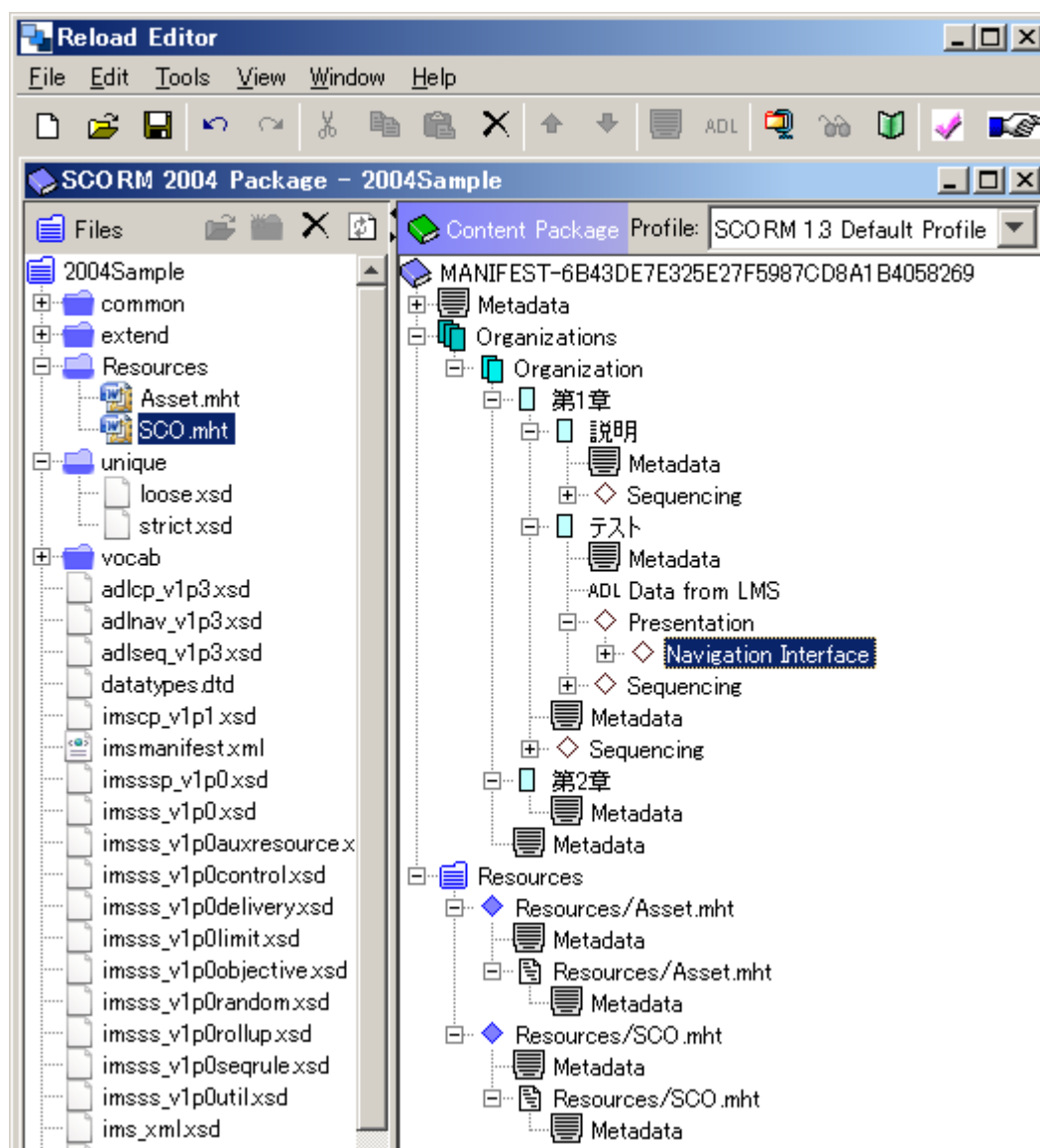


図 4.2 SCORM 2004 のマニフェストファイル編集画面

```

<?xml version="1.0" encoding="UTF-8"?>
<!--This is a Reload version 2.5.5 SCORM 2004 Content Package document-->
<!--Spawned from the Reload Content Package Generator - http://www.reload.ac.uk-->
<manifest xmlns="http://www.imsglobal.org/xsd/imsdp_v1p1"
xmlns:imsmd="http://ltsc.ieee.org/xsd/LOM"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_v1p3"
xmlns:imsss="http://www.imsglobal.org/xsd/imsss"
xmlns:adlseq="http://www.adlnet.org/xsd/adlseq_v1p3"
xmlns:adlnav="http://www.adlnet.org/xsd/adlnav_v1p3"
identifier="MANIFEST-6B43DE7E325E27F5987CD8A1B4058269"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imsdp_v1p1 imscp_v1p1.xsd
http://ltsc.ieee.org/xsd/LOM lom.xsd http://www.adlnet.org/xsd/adlcp_v1p3 adlcp_v1p3.xsd
http://www.imsglobal.org/xsd/imsss imsss_v1p0.xsd http://www.adlnet.org/xsd/adlseq_v1p3
adlseq_v1p3.xsd http://www.adlnet.org/xsd/adlnav_v1p3 adlnav_v1p3.xsd">
  <metadata>
    <schema>ADL SCORM</schema>
    <schemaversion>2004 3rd Edition</schemaversion>
  </metadata>
  <organizations default="ORG-09144932A7E9E59FE90A3BD9EE97B1E5">
    <organization identifier="ORG-09144932A7E9E59FE90A3BD9EE97B1E5" structure="hierarchical">
      <title>Organization</title>
      <item identifier="ITEM-9328F80287405C04120EBC4ACD47EC1E" isvisible="true">
        <title>第1章</title>
        <item identifier="ITEM-59A30516F9B637FE3D3FFB8032AA7491" isvisible="true"
identifierref="RES-6ADD175611D3CDD4845D1BA32D2C222E">
          <title>説明</title>
          <metadata>
            <imsmd:lom>
              <imsmd:general>
                <imsmd:language>ja</imsmd:language>
              </imsmd:general>
            </imsmd:lom>
          </metadata>
          <imsss:sequencing> ——(1)
            <imsss:sequencingRules>
              <imsss:preConditionRule>
                <imsss:ruleConditions conditionCombination="all">
                  <imsss:ruleCondition operator="noOp" condition="completed" />
                </imsss:ruleConditions>
                <imsss:ruleAction action="skip" />
              </imsss:preConditionRule>
            </imsss:sequencingRules>
            <imsss:rollupRules objectiveMeasureWeight="1.0000" />
          </imsss:sequencing>
        </item>
        <item identifier="ITEM-A1DCDCA63CA265A94AEE0490A7714A27" isvisible="true"
identifierref="RES-EB981004964E564CB9F94231FEC016B1">
          <title>テスト</title>
          <metadata>
            <imsmd:lom>
              <imsmd:general>
                <imsmd:language>ja</imsmd:language>
              </imsmd:general>
            </imsmd:lom>
          </metadata>
          <adlcp:dataFromLMS>データ初期値</adlcp:dataFromLMS>
          <adlnav:presentation> ——(A)
            <adlnav:navigationInterface>
              <adlnav:hideLMSUI>continue</adlnav:hideLMSUI>
              <adlnav:hideLMSUI>previous</adlnav:hideLMSUI>
            </adlnav:navigationInterface>
          </adlnav:presentation>
          <imsss:sequencing> ——(2)
            <imsss:rollupRules objectiveMeasureWeight="1.0000" />
            <imsss:objectives>
              <imsss:primaryObjective satisfiedByMeasure="true" objectiveID="">
                <imsss:minNormalizedMeasure>0.5</imsss:minNormalizedMeasure>
              </imsss:primaryObjective>
            </imsss:objectives>
          </imsss:sequencing>
        </item>

```

```

<metadata>
  <imsmd:lom>
    <imsmd:general>
      <imsmd:language>ja</imsmd:language>
    </imsmd:general>
  </imsmd:lom>
</metadata>
<imsss:sequencing> —(3)
  <imsss:sequencingRules>
    <imsss:preConditionRule>
      <imsss:ruleConditions conditionCombination="all">
        <imsss:ruleCondition operator="noOp" condition="satisfied" />
      </imsss:ruleConditions>
      <imsss:ruleAction action="skip" />
    </imsss:preConditionRule>
    <imsss:exitConditionRule>
      <imsss:ruleConditions conditionCombination="all">
        <imsss:ruleCondition operator="not" condition="satisfied" />
      </imsss:ruleConditions>
      <imsss:ruleAction action="exit" />
    </imsss:exitConditionRule>
    <imsss:postConditionRule>
      <imsss:ruleConditions conditionCombination="all">
        <imsss:ruleCondition operator="not" condition="satisfied" />
      </imsss:ruleConditions>
      <imsss:ruleAction action="retry" />
    </imsss:postConditionRule>
  </imsss:sequencingRules>
  <imsss:rollupRules objectiveMeasureWeight="1.0000" />
</imsss:sequencing>
</item>
<item identifier="ITEM-BEA296838AA083547FA6E1B7F77B09D9" isVisible="true">
  <title>第2章</title>
  <metadata />
</item>
<metadata />
</organization>
</organizations>
<resources>
  <resource identifier="RES-6ADD175611D3CDD4845D1BA32D2C222E" adlcp:scormType="asset">
    type="webcontent" href="Resources/Asset.mht">
    <metadata />
    <file href="Resources/Asset.mht">
    <metadata />
    </file>
  </resource>
  <resource identifier="RES-EB981004964E564CB9F94231FEC016B1" adlcp:scormType="SCO">
    type="webcontent" href="Resources/SCO.mht">
    <metadata />
    <file href="Resources/SCO.mht">
    <metadata />
    </file>
  </resource>
</resources>
</manifest>

```

図 4.3 SCORM2004 マニフェストファイルの例

4.2.4 マニフェストファイルの記述例

マニフェストファイルの記述例を図 4.2, 図 4.3 に示す。マニフェストファイルの基本的な構成は SCORM 1.2 の場合と同様である。マニフェストファイルは大きくオーガニゼーションとリソースから構成され、オーガニゼーションでコンテンツの階層構造が表現される。オーガニゼーションおよびリソース中の各構成要素はメタデータを持つことができる。また、階層構造の末端のノードから、リソース中の学習資源への対応付けが行われる。

SCORM 2004 のマニフェストファイルの特徴はシーケンシングやナビゲーションに関する記述が各<item>に付け加わることである。例えば、図 4.3 で、(1), (2), (3) はそれぞれシーケンシ

ングルールの記述である。(1)は「説明」という題名のアクティビティが完了していたら **Skip** すること, (2)は「テスト」という題名のアクティビティの習得度が 0.5 以上なら習得とみなすこと, (3)は「第 1 章」という題名のアクティビティが未取得なら **Retry** すること, などが記述されている。また, (A) には, 「テスト」という題名のアクティビティを表示したとき, **Continue, Previous** を発行するナビゲーション UI を LMS が表示しないことが記述されている。

4.3 ランタイム環境

SCORM 2004 の RTE(Run-Time Enviromet:ランタイム環境)では SCORM 1.2 から大きく変更がなされており，本節では特にその変更点について説明する．

4.3.1 起動

LMS より起動されるのは SCORM1.2 同様に SCO とアセットの 2 つである．SCORM2004 では，SCO を起動する前に，どの SCO を起動するのかを決めるためにシーケンシング動作が行われる．(図 4.4 シーケンシング参照)

4.3.2 API

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 ECMA Script API for Content to Runtime Service Communication”(IEEE 1484.11.2 で規格化が進め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．

SCORM1.2 では，API 関数を実装するものを API アダプタと呼び，LMS が用意するフレーム名を“API”と定義しているが，SCORM2004 の場合は API 関数を実装するものを API インスタンスと呼びそのフレーム名を“API_1484_11”と定義している．

SCORM2004 の LMS では SCORM1.2 と SCORM2004 の両方の SCO を扱えるように API を用意し，SCO は両方の API インスタンス (アダプタ) を意識しなくてもデータの送受信を API 関数を用いて実現できるようになっている．

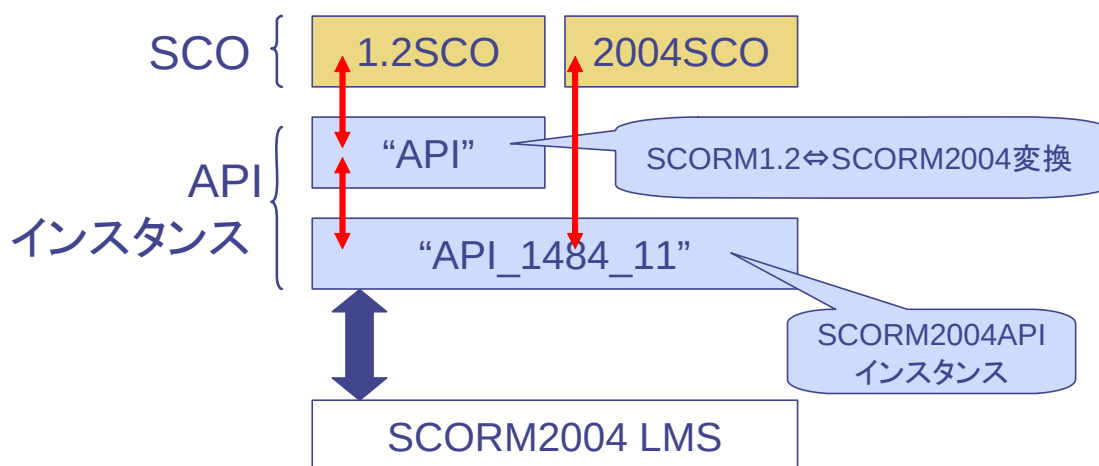


図 4.4 SCORM2004LMS における SCORM1.2 と SCORM2004 の混在利用

4.3.2.1 API 関数

SCORM2004 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 4.1 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

また、上記の API 関数を実装する API インスタンスの状態遷移を図 4.5 に示す。

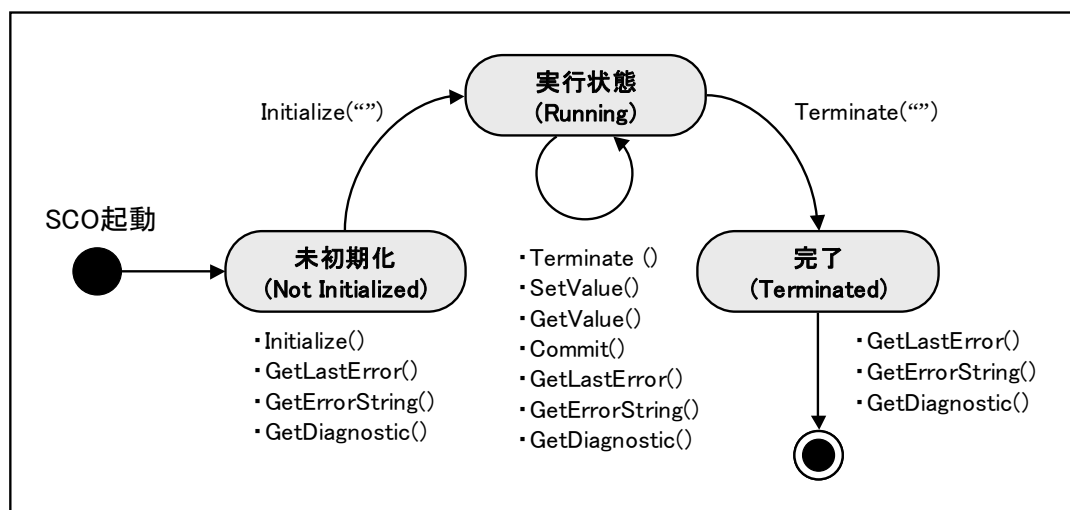


図 4.5 API インスタンスの状態遷移と SCORM API

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、より具体的な状態を確認できるようになっている。

表 4.2 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

4.3.3 データモデル

SCORM 2004 ランタイム環境におけるデータモデルは、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おり、

- ・ 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
- ・ データモデル要素の変更
 - －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)

4.3.3.1 データモデル要素

SCORM2004 では、前述したように全ての項目を必須として `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している。以下に SCORM1.2 と SCORM2004 のデータモデル要素の比較表を示す。

表 4.3 データモデル要素一覧

SCORM 1.2	SCORM 2004	意味
<code>cmi.core.student_id</code>	<code>cmi.learner_id</code>	学習者 ID
<code>cmi.core.student_name</code>	<code>cmi.learner_name</code>	学習者名
<code>cmi.core.lesson_location</code>	<code>cmi.location</code>	SCO 中の位置
<code>cmi.core.credit</code>	<code>cmi.credit</code>	評価を行うか否か
<code>cmi.core.lesson_status</code>	<code>cmi.completion_status</code> <code>cmi.success_status</code>	進捗状態 習得状態
<code>cmi.core.entry</code>	<code>cmi.entry</code>	SCO 開始時の状態
<code>cmi.core.score</code>	<code>cmi.score</code>	得点
<code>cmi.core.total_time</code>	<code>cmi.total_time</code>	総学習時間
<code>cmi.core.lesson_mode</code>	<code>cmi.mode</code>	SCO の動作モード
<code>cmi.core.exit</code>	<code>cmi.exit</code>	SCO 終了時の状態
<code>cmi.core.session_time</code>	<code>cmi.session_time</code>	セッション学習時間
<code>cmi.suspend_data</code>	<code>cmi.suspend_data</code>	中断時保持データ
<code>cmi.launch_data</code>	<code>cmi.launch_data</code>	起動時設定データ
<code>cmi.comments</code>	<code>cmi.comments_from_learner</code>	学習者のコメント
<code>cmi.comments_from_lms</code>	<code>cmi.comments_from_lms</code>	LMS からのコメント
<code>cmi.objectives</code>	<code>cmi.objectives</code>	学習目標
<code>cmi.student_data.mastery_score</code>	<code>cmi.scaled_passing_score</code>	合格点
<code>cmi.student_data.max_time_allowed</code>	<code>cmi.max_time_allowed</code>	学習許容時間
<code>cmi.student_data.time_limit_action</code>	<code>cmi.time_limit_action</code>	時間制限超過時動作
<code>cmi.student_preference</code>	<code>cmi.learner_preference</code>	学習者選好
<code>cmi.interactions</code>	<code>cmi.interactions</code>	演習問題等の記録
	<code>cmi.progress_measure</code>	進捗度
	<code>cmi.completion_threshold</code>	完了必須進捗度

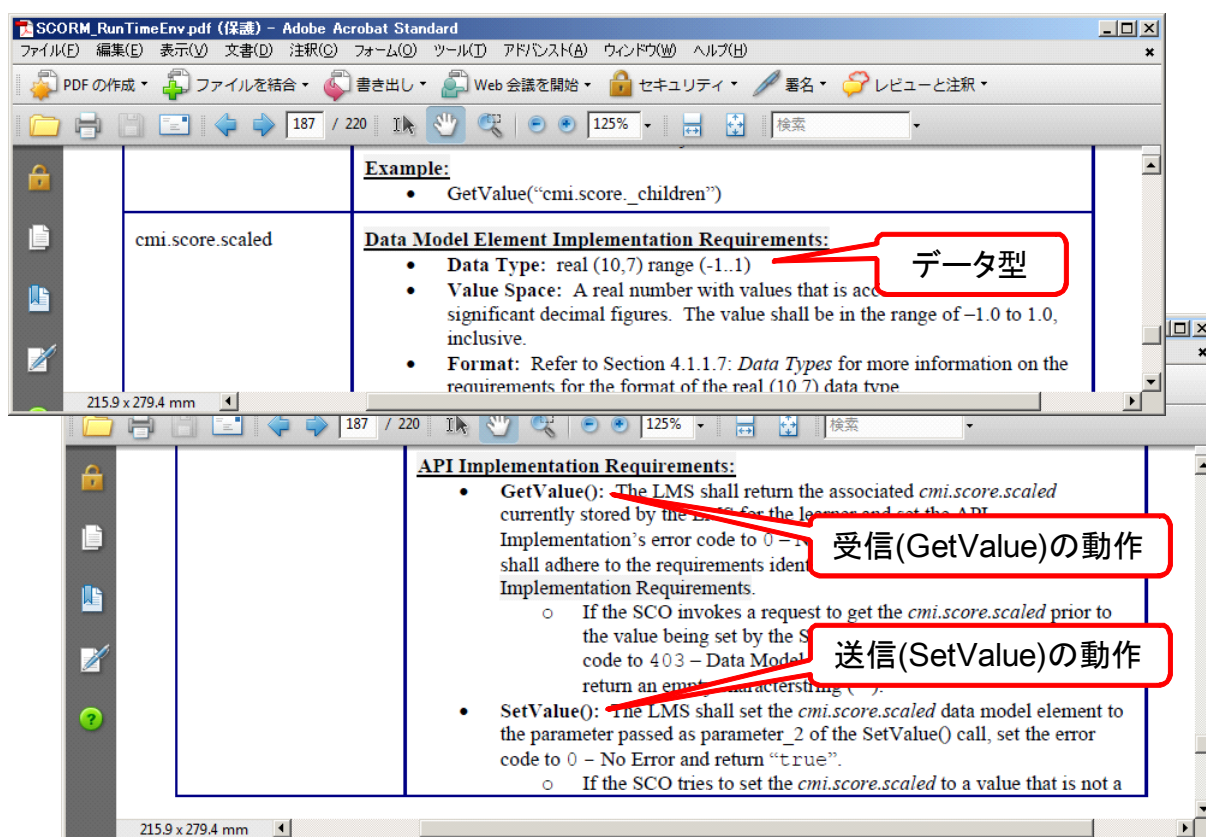


図 4.6 SCORM2004 ランタイム環境仕様書のデータモデル要素の例

図 4.6 は SCORM 2004 の “cmi.score.scaled” の例である。データ型は実数で-1 から 1 の範囲であることがわかる。また、SCO が受信を行う場合 LMS は値を返却し、送信の場合は値を設定する、と書かれていることから、このデータモデル要素は読み書き双方向が可能であることがわかる。

4.3.3.2 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“cmi.completion_status”を通して LMS に（トラッキング情報として）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。

4.4 シーケンシング

SCORM のコンテンツは階層構造となっていて、各ノードをアクティビティと呼ぶ。階層構造の末端のアクティビティはアセット、SCO など学習者に提示される学習資源と対応している。これらの学習資源を学習者に提示する順序を制御することをシーケンシングと呼ぶ。シーケンシングの最も単純な方法は、階層構造で決められた順序に従うこと、および、学習者が要求したものをそのまま提示することである。SCORM 1.2 ではこの 2 つがシーケンシングの主要な機能であった。

SCORM 2004 ではこれに加えて、教材作成者があらかじめ定めたルールに従ったシーケンシングが可能となっている。これは、学習者の進捗状況や習得状況に応じて、学習資源を提示する順序についてルールを用いて動的に決定するものである。これによって、演習問題の成績が低ければ復習を行う、成績が良ければより進んだ内容を学習する、といった動作を教材作成者が記述することが可能になった。

SCORM2004 では新たに以下の二つの規格が加わっている。

- (1) シーケンシング規格. アクティビティの提示順序を学習者の習得状況や学習経過時間などに応じて、サーバが動的に決定するための動作記述方法を規定する。例えば、「演習問題のアクティビティが不合格となったら復習のアクティビティを提示する」、「学習目標を習得するまで解説と問題を繰り返す」といった動作を記述する標準的な方法を規定する。
- (2) ナビゲーション規格. 「次に進む」、「前に戻る」など学習者がコンテンツ構造内を移動するためのナビゲーションコマンドの GUI を SCO が提供し、サーバにコマンドを通知する方法を規定する。

以下の節で、シーケンシング規格およびナビゲーション規格の概要を述べる。

4.4.1 シーケンシング規格

SCORM2004 のシーケンシング機能は IMS Simple Sequencing 規格[10]で規定されたものである。図 4.7 に概要を示す。コンテンツ作成者は、コース構造とそれに付随する動作ルールを記述することによってコンテンツの動作を制御する。実行時に、コース構造と動作ルールは LMS 内のシーケンシングエンジンに読み込まれる。シーケンシングエンジンは、学習者からの要求を受け取り、動作ルールを解釈し、学習者の学習状態を反映するための状態情報（トラッキング情報と呼ぶ）を更新し、次の提示画面を決定するという動作を繰り返す。

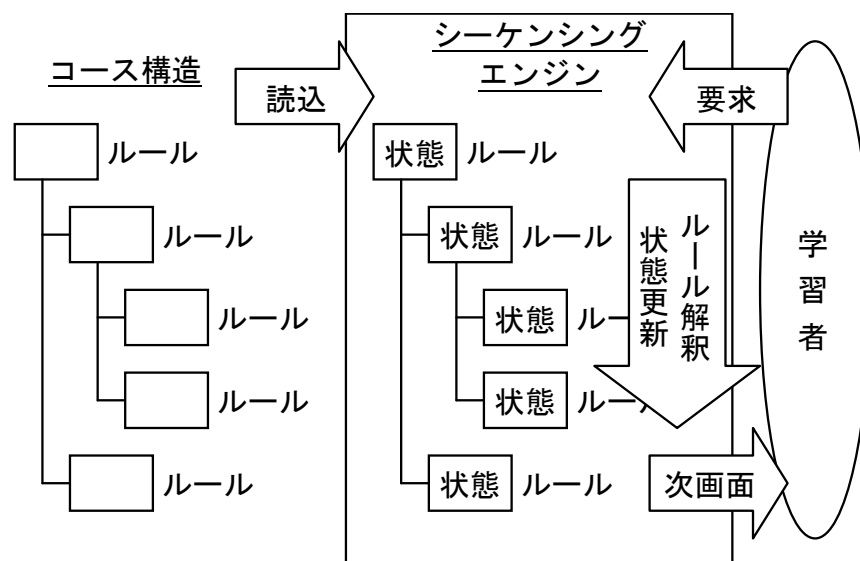


図 4.7 シーケンシング処理の概要

シーケンシングの主要な構成要素と外部機能は、

- 1) コース構造と学習目標
- 2) トラッキング情報
- 3) シーケンシング要求
- 4) 動作ルール

で規定される。実行時にこれらを用いてシーケンシングをどのように実行するかは擬似コードとして規格に明記されている。

4.4.1.1 コース構造と学習目標

コース構造は階層構造で各ノードはアクティビティ(Activity)と呼ばれる。各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。教材作成者は、それ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。アクティビティと共有学習目標の間には、読み書きの関係を定義するようになっている。ある共有学習目標に書き込みが行えるのはひとつのアクティビティからのみと規定されており、これによって共有学習目標の状態が不確定になるのを防いでいる。

4.4.1.2 トラッキング情報

トラッキング情報は、実行時の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。アクティビティのトラッキング情報には、進捗状態(アクティビティを完了したか未完了か)、アクティビティの実行回数や実行時間、などが含まれる。学習目標のトラッキング情報は、習得状態(学習目標を習得したか未修得か)および習得度(習得度合いを示す得点)からなる。トラッキング情報は学習実行時にシーケンシングエンジンが自動的に状態を更新していく。

4.4.1.3 シーケンシング要求

シーケンシング要求は、「Start (実行開始)」「Continue (次に進む)」、「Previous (前に戻る)」,

「Choice（選択アクティビティに移動）」、「Retry（再試行）」、「Exit（終了）」など、シーケンシングエンジンに対する動作コマンドである⁴。シーケンシング要求は、学習者が GUI を通して入力した要求に直接対応するものと、次に述べる動作ルールによって発生するものがある。シーケンシング要求により、シーケンシングエンジンは、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。このとき以下に述べる動作ルールのチェックが行われる。

4.4.1.4 動作ルール

動作ルールは、教材作成者の記述するシーケンシング動作の定義である。動作ルールは大きく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によるアクティビティ間遷移動作にある制限を加えるもの。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「アクティビティ配下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ナルルールや、「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ナルルールと呼ばれる。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。
- トラッキング情報の更新に関するもの。先に述べたように、アクティビティと学習目標はトラッキング情報を保持している。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。子アクティビティがロールアップに関与するか否か、また、ロールアップが生じる条件と結果を陽に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。

4.4.1.5 シーケンシングの具体例

以上のような機能によって、教材作成者が作成することのできるシーケンシング動作の一例を図 4.8 に示す。

この例は「解説」、「復習」、「テスト」のアクティビティからなる教材で、以下のような動作を意図している。

- 最初に「解説」を提示し「テスト」に合格したら終了。
- 不合格なら「復習」を提示し、再度「テスト」を実行。これを合格するまで繰り返す。
繰り返しの最大回数は 3 回。これを越えたら不合格でも強制終了。

⁴ 規格では、学習者が GUI から入力するナビゲーションイベント、ナビゲーションイベントをシーケンシングエンジンに対するコマンドにマップしたナビゲーション要求、エンジンの内部形式であるシーケンシング要求などが区別されているがここでは詳述しない。

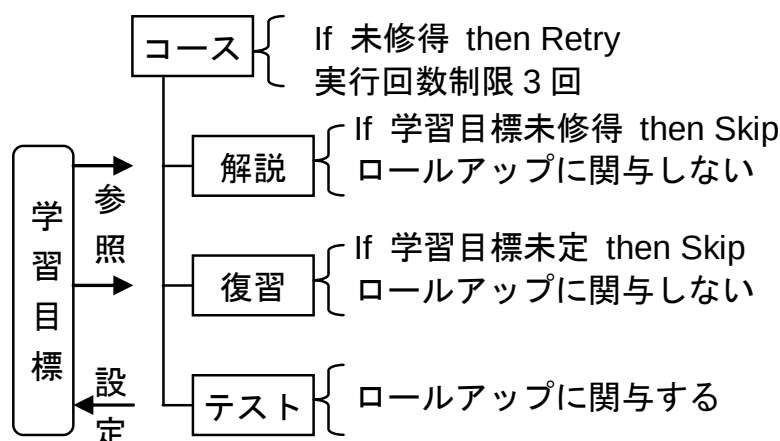


図 4.8 シーケンシングルール例

上記の動作を実現するため、図 4.8 では共有学習目標を用い、これを参照する動作ルールを各アクティビティに記述して動作を制御している。また、「テスト」アクティビティだけをロールアップに関与させ、「テスト」に合格（習得）すればコース全体が習得となるようにしている。

実行を開始するとシーケンシングエンジンは「解説」を提示しようとする。学習目標の初期値は「未定（未修得とは異なる）」であるので、「解説」のプリコンディションルールは成立せず、Skip せずに「解説」が提示される。次に「復習」を提示しようとするが、学習目標が「未定」であるので、「復習」のプリコンディションルールが成立し、「復習」を Skip して「テスト」が提示される。学習者が「テスト」を実行して不合格となれば、学習目標は「未修得」、ロールアップによって「コース」も未習得となり、「コース」のポストコンディションルールが成立して再試行が行われる。今度は学習目標が未修得となっているため、「解説」、「復習」それぞれのプリコンディションルールに従い、「解説」が提示されず、「復習」が提示される。その後「テスト」を実行して、不合格となれば同じ動作を繰り返す。但し、繰り返し回数が 3 回を越えたら「コース」の制限条件により実行を終了する。

4.4.2 ナビゲーション規格

従来の SCORM1.2 規格では、「次に進む」、「前に戻る」などのナビゲーションコマンドを SCO が発行することができず、コマンドはブラウザ中に SCO とは別に設けたコマンドフレームから発行していた。このフレームの GUI デザインは LMS に依存していて、コンテンツ作成者が変更することができないため、SCO の GUI デザインとコマンドフレームの GUI デザインを統一できない、という問題があった。また、SCO からナビゲーションコマンドが発行できないために、SCO 内の演習問題やシミュレーションコンテンツを操作中に任意のタイミングでナビゲーションコマンドを発行することもできなかった。

このような問題点を解決するため、SCORM2004 では、SCO からナビゲーションコマンドを発行するための仕様が規定された。実際には、SCO とサーバが通信を行うための RTE 規格を拡張し、SCO がナビゲーションコマンドをサーバに通知するためのデータモデルが追加されている。

本テキストについて

本書の執筆者は以下の通りである。

小林 裕司	株式会社 グラフィン
田口 邦彦	株式会社ビズバレー
仲林 清	千葉工業大学
星野 忠明	エスエイティーティー株式会社
宮内 浩	産業能率大学

本書（SCORM テキスト）は特定非営利活動法人日本イーラーニングコンソシアム（略称：eLC）の著作物である。

本書はSCORM技術者資格講習会の事前学習を目的として作成された。

eLC（ライセンス提供者）は、他者（ライセンス受領者）に本書のコピー，配布，表示，ハイパーリンクの生成を許可する。代わりに，ライセンス提供者のクレジット（上記3行の文章）を明記しなければならない。

また，ライセンス受領者はライセンス提供者の許可なく本書を商用利用してはならない。