

# SCORM準拠のe-ラーニング・コンテンツ開発の実際

## Current conditions concerning the development of e-learning content in line with SCORM



日本アイ・ビー・エム株式会社  
ソフトウェア事業部 ロータス事業推進  
KM & eLソリューション  
主任ITスペシャリスト

篠原 竜雄

Tatsuo Shinohara

Advisory IT Specialist  
BU-KM & eL Solution  
IBM Japan, Ltd.

昨今、e-ラーニングという言葉を目にする機会がかなり増えてきました。筆者が、e-ラーニングの前身ともいえるWBT(Web Based Training)の世界にかかわり始めた1998年ごろに比べると、さまざまなメディアがe-ラーニングを取り上げ、また、数多くの企業がe-ラーニング市場に参入してきています。しかしながら、一般に入手できるe-ラーニングの情報のほとんどは「e-ラーニングの可能性・市場動向・活用事例・標準化動向」といった概論的な内容であり、e-ラーニングの世界で活躍するITエンジニア向けの純粋な技術資料は非常に少ないのが現状です。本論文は、e-ラーニングの重要な要素である教材コンテンツを開発するに当たり、e-ラーニングの標準化のポイントであるSCORM標準規格をどのように活用するかという点を中心に論述しています。簡単なJavaScriptのサンプル・コードを例に、e-ラーニング教材コンテンツの開発の実際について紹介します。

The term "e-learning" has been gaining increasing currency of late. In comparison with the period around 1998 when I myself first became involved in the world of WBT (Web-Based Training), which might be thought of as the precursor of e-learning, the idea of e-learning is now being featured more and more extensively in the mass media, and many companies have become involved in the e-learning market. But despite this, almost all the readily available information on e-learning offers little more than general summaries dealing with matters such as the possibilities of e-learning, trends in the market, examples of application, and moves toward standardization. Very little information is currently available in the form of purely technical materials aimed at IT engineers active in the world of e-learning.

This paper concentrates especially on how SCORM standards (the bases for the standardization of e-learning) are being used in connection with the development of the teaching materials that constitute important elements of e-learning. Using the example provided by simple JavaScript sample codes, we take a look at how e-learning teaching materials are actually being used.

## 1.はじめに

Webサイトやメールで配信されるIT関連のニュースで、ほとんど毎日のようにeラーニングに関する記事が紹介されています。しかしながら、現在、一般に入手することのできるeラーニングの情報は「eラーニングの可能性・市場動向・活用事例・標準化動向」といった概論的な内容がほとんどです。eラーニングの世界で活躍するITエンジニア向けの純粋な技術資料はまず見つかりません。技術者は、AICC、ADLといった標準化団体が提供する英文の標準仕様書や、ALICなど国内の標準化団体が提供する標準仕様書の概略書、学習管理システム(LMS: Learning Management System)や教材コンテンツ作成オーサリング・ツールのベンダーのマニュアルなどから、苦勞して技術的な情報を入手しています。しかしながら、これらの技術資料はあくまでも仕様書またはリファレンスであり、「いかにして教材コンテンツを開発するのか」「いかにしてeラーニングのシステムを構築・設計するのか」といったノウハウについての解説書は存在しません。技術者自らが試行錯誤しながらノウハウを培っているというのが現状です。

本論文は、eラーニングの重要な要素である教材コンテンツの開発に当たり、eラーニング標準化のポイントであるSCORM標準規格をどのように活用するのかという点を中心に論述しました。

最初にeラーニングの標準化の意義と動向について簡単に説明します。次に、解説型コンテンツ・演習問題型コンテンツの二つの代表的なeラーニング教材コンテンツについて、簡単なJavaScript™のサンプル・コードを例に開発の実際を紹介します。最後に、現在(2002年7月)入手可能な主な教材作成オーサリング・ツールについて簡単に紹介します。

## 2. eラーニングの標準化

### 2.1. eラーニング標準化の意義

ITの世界では、オープンな標準技術を多くのベンダーが製品に採用するという傾向があります。オープンな標準仕様を複数の製品で採用することが、ユーザーのみならずベンダーにもメリットがあることは、ここ数年のXML、J2EE、EJBなどに代表される標準化技術の広がりを見れば明らかです。

一方、1990年代後半より、Webのインフラストラクチャーを使用してトレーニングや学習を提供するソリューションとしてWBT(Web Based Training)が注目され始めました。学習者は、Webブラウザを用いてWBTサーバーに「いつでも好きなときに、どこでも好きな場所から」アクセスし、学習を「自分のペース」で行えます。一方、管理者はサーバーに蓄積された学習者の学

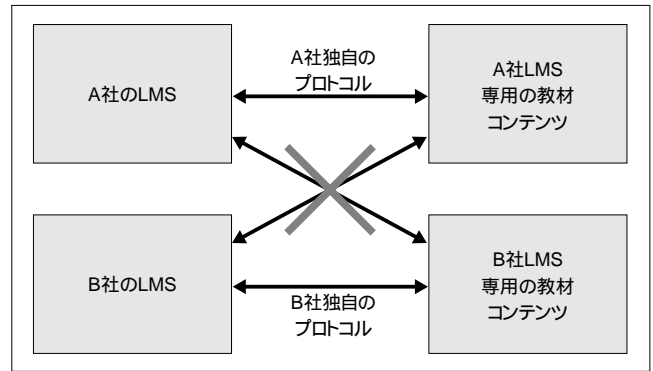


図1. LMSの標準化以前

習履歴や成績を効率的に管理できます。Webのインフラストラクチャーを活用することで、多数の学習者への教育を一斉に、しかも低コストで実現できることが注目を集めました。当時のWBTは、学習者やコンテンツを管理するサーバーと、教材コンテンツが一体となっていたケースがほとんどです。各ベンダーはそれぞれ独自の管理システムと、その管理システム上のみで動作する教材コンテンツを提供していました。コンテンツ自体は管理システムに大きく依存しているため、ほかのシステムに移植することは不可能であり、コンテンツ自体を単独で流通させるという概念はまだ存在しなかったのです(図1)。

1999年ごろより、米国で先行していたeラーニングの標準化が、国内にも紹介され始めました。米国の航空業界が主導となって推進しているAICC(Aviation Industry CBT Committee)のeラーニング標準仕様であるCMI(Computer Managed Instruction)規格を採用した製品が、国内でも少しずつ登場し始めました。AICCの規格は、LMSと教材コンテンツのための相互運用性と、互換性のための仕様(ガイドライン)を示したものです。これにより、ユーザー、LMSベンダー、教材コンテンツ提供者にそれぞれ下記のようなメリットが生じます(図2参照)。

《ユーザーからみたLMS標準化のメリット》

- 複数のベンダーが提供する標準規格対応製品(教材コンテンツ、LMS)の中から、自由に製品を選択し、組み合わせて使用できます。

- 特定のLMSベンダーに拘束されません。

- 教材コンテンツの継承・再利用が約束されます。

《教材コンテンツ・プロバイダーからみたLMS標準化のメリット》

- 特定のLMSベンダーへの依存がなくなります。

- 独自の教材の企画・制作が可能となります。

《LMSシステム・ベンダーからみたLMS標準化のメリット》

- 汎用教材の再利用が可能となります。

- 互換性のある教材の流通により市場が拡大します。

AICCのCMI規格では主に以下の2点の仕様が定義されました。

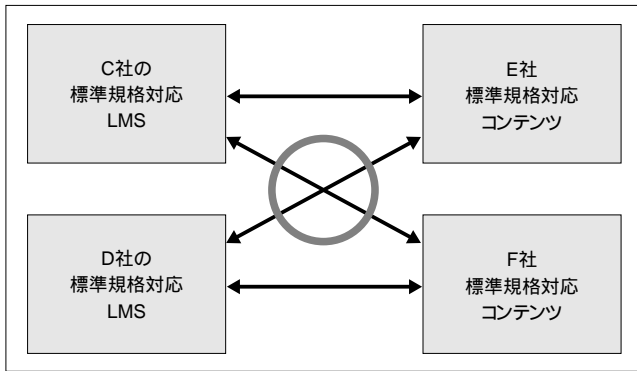


図2. LMSの標準化のメリット

- LMSと教材コンテンツ間で受け渡されるデータについての詳細な仕様の規定。
- 異なるLMS間でコース構造を共有するための仕様の規定（CSV、ini形式）。

AICCの普及と並行して、米国ではADL(Advanced Distributed Learning Initiative)という団体によって、AICCおよびほかの幾つかの標準化団体の仕様が統合され、SCORM(Shareable Content Object Reference Model)という規格が策定されました。SCORM規格は、AICC規格に比べて主に以下の点が強化されています。

- メタデータの仕様LOM(Learning Object Metadata)を統合。
- コース構造の記述をCSV、ini形式からXML形式へ変更。
- LMSと教材コンテンツ間のデータの受け渡しにJavaScriptによるAPIを導入。
- コース構造と教材コンテンツをアーカイブ・ファイル化して、移植性を高めたコンテンツ・パッケージング規格を導入。

国内においては、先進学習基盤協議会(ALIC)、日本イーラーニングコンソーシアムなどが中心となって、SCORMやAICC普及のための活動を進めています。その結果、数多くの国内のLMSベンダーからSCORMに準拠したLMS製品が発表され、また、SCORMに準拠した教材作成オーサリング・ツールや教材コンテンツ自体も数多く発表されています。また、米国の主要なeラーニング・ベンダーによる日本市場への参入も続いています。

## 2.2. 学習管理システムと教材コンテンツの役割分担

SCORMやAICCなどの標準仕様をベースとしたeラーニング・システムにおいては、教材コンテンツとLMSとを明確に分離することが大きなポイントとなります。その上で、教材コンテンツとLMSとの間のインターフェースを規定し、それぞれの役割分担を明確にすることが求められます(図3)。

LMSに求められる役割は以下の通りです。

- 教材のコース構造(章立て・目次に相当)を表示します(コース構造は、後述するように、教材コンテンツ側によって提供さ

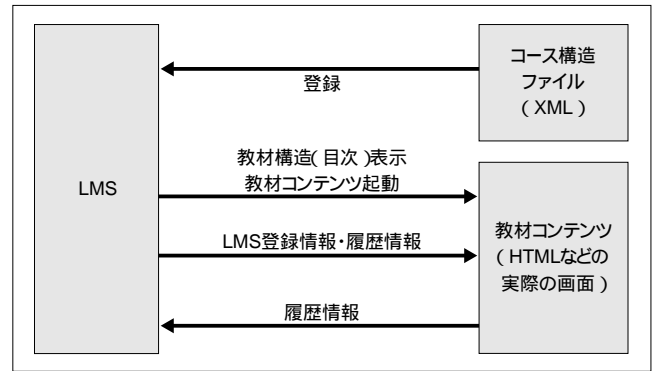


図3. LMSと教材コンテンツの役割

- れたコース構造ファイルを事前に読み込んでおきます)。
  - コース構造から選択された項目の教材コンテンツの画面を表示します。
  - 教材コンテンツからの履歴情報送受信のリクエストを待ちます。
  - 教材コンテンツから送られた履歴情報を受信し、格納・管理します。
- 一方、教材コンテンツ側の役割は以下の通りです。
- 教材のコース構造(目次)ファイルを提供します。
  - LMSによって起動される教材画面や演習問題画面の実体、すなわちHTML、PDF、アニメーション、動画など、実際にWebブラウザに表示される画面です。演習問題コンテンツの場合、各問題の正誤判定や採点は、LMSではなく教材コンテンツ側で行います。演習問題の得点や学習進捗などの履歴情報は、教材コンテンツがLMSへの送信を行います。

## 2.3. SCORM概要

SCORMとは2.1節で記述した通りLMS標準規格の一つであり、現在最も有力な規格です。SCORM規格では、前節で示したLMSと教材コンテンツの役割分担をベースとしています。2002年7月現在の最新版はSCORM1.2です(2002年後半にはSCORM1.3が発表される予定)。

コース構造ファイルについては、以下のようにXML形式のファイルについての仕様が既定されています(図4)。

図5は上記のXML形式のコース構造ファイルをLMS(LearningSpace® 5.01)に読み込んだ画面です。コース構造ファイルでは、教材コンテンツの目次構成・章立て・前提条件などを定義します。コース構造ファイルは一般的には教科書のような「章・節・項」から成る階層構造を採っています。

コース構造の末端のページには、ページごとにHTML、PDF、アニメーション、動画など、実際にWebブラウザに表示される画面の実体のURLが対応付けられており、これをSCORMではSCO(Shareable Content Object)と呼びます。

SCOは、学習時間、演習問題の解答、正誤、得点などの履歴

```

<?xml version="1.0" ?>
-<course>
  -<block id="ROOT">
    -<identification>
      <title>N001「ノーツ入門コース」</title>
      <description>ノーツ入門コース</description>
    </identification>
    -<block id="B1">
      -<identification>
        -<labels>
          <developer>D_B1</developer>
        </labels>
        <title>はじめに</title>
        <description>はじめに</description>
      </identification>
      <au id="A1">

```

図4. コース構造ファイルの例

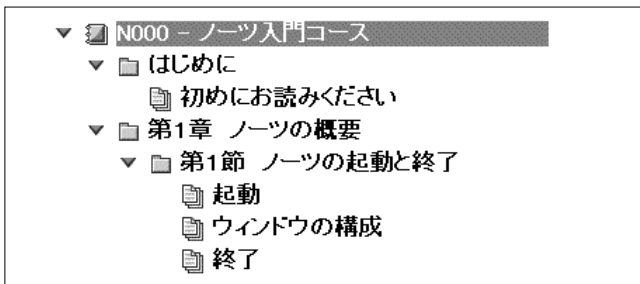


図5. 画面例

情報をLMSに送信する役割を持ちます。SCOは、LMSが提供するAPIをJavaScriptから呼び出してLMSと通信します。以下に、主なSCORM APIメソッドを示します。

- LMSInitialize  
初期化処理 (LMS側から情報を取得)
- LMSGetValue  
LMSの取得情報から必要な情報の取り出し。
- LMSSetValue  
LMSに送信する履歴情報をセット。
- LMSCommit  
LMSに履歴情報を一括送信。
- LMSFinish  
終了処理。
- LMSGetLastError, LMSGetErrorString  
エラー情報の取得。  
以下にLMSSetValue、LMSGetValueによって、SCOとLMS間でやり取りされる主なデータを示します。
- 学習者ID (cmi.core.student\_id)
- 学習者氏名 (cmi.core.student\_name)
- 前回終了した位置 (cmi.core.lesson\_location)
- ステータス (cmi.core.lesson\_status)  
未開始 (not attempt)、未修了 (incomplete)、修了 (completed)、合格 (passed)、不合格 (failed)

- 得点 (cmi.core.score.raw)
- 実行時間 (cmi.core.session\_time、cmi.core.total\_time)
- 起動時指定データ (cmi.launch\_data)
- 前回終了時一時保存データ (cmi.suspend\_data)
- 演習問題の問題名 (cmi.interactions.n.id)
- 演習問題の問題タイプ (cmi.interactions.n.type)
- 演習問題の正解  
(cmi.interactions.n.correct\_responses.0.pattern)
- 演習問題の学習者の回答  
(cmi.interactions.n.student\_response)
- 演習問題の正誤 (cmi.interactions.n.result)
- 演習問題の配点 (cmi.interactions.n.weighting)
- 演習問題の回答時間 (cmi.interactions.n.latency)

### 3. 教材コンテンツ開発の実際

#### 3.1. 教材コンテンツのコース構造

前章で説明したように、コース構造は「教科書」の目次に相当します。ここではコース構造の実装方法について、三つの方法を示しました。

《一つのコースにSCOが一つだけのコース構造》

一つのコースに含まれるSCOが一つだけのコース構造ファイルを構築します。つまり章立ての定義や目次の構成は、コース構造ファイルではなく、SCO自体で行います (図6)。

HTMLベースで作成している独自教材コンテンツが存在し、既に章立てや目次はその中で独自に定義しているような場合に、コンテンツをSCORM規格に迅速に対応させたいときによく採られる手法です。

問題点として、教材の中でどこまで進捗があったかという履歴が細かく取れなかったり、途中で章立ての構造を修正したいようなときの対応が面倒といったことが挙げられます。

《複数のSCOから構成されるコース構造》

一つのコースに複数のSCOを含むコース構造ファイルを構築することで、大きな分類による章立ての定義をSCO内部に持たせる必要がなくなります (図7)。また、SCO単位での前提条件の設定も可能になり、例えば、2章を学習するには1章の学習を完了しなければならない、といった構成を採ることができます。

進捗の履歴については、SCO単位で取ることが可能です。しかし、コース構造がSCOだけの場合、階層構造ではなくフラットな構造になるため、章・節・項などから成る大規模な教材コンテンツには不向きです。1教材当たり10章以下で、各章の規模もそれほど大きくない場合に適しています。

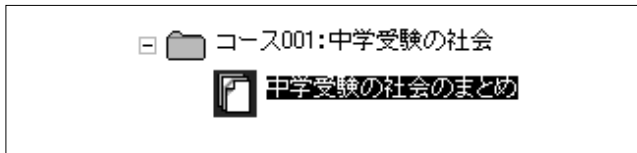


図6. 一つのコースにSCOが一つだけのコース構造の例

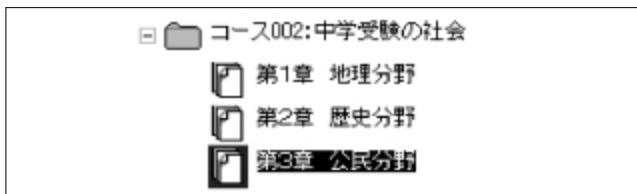


図7. 複数のSCOから構成されるコース構造の例



図8. 階層化された複数のSCOから構成されるコース構造の例

### 《階層化された複数のSCOから構成されるコース構造》

一つのコースに複数のSCOによる階層構造のコース構造ファイルを構築することで、より柔軟で管理しやすい教材を構築できます(図8)。

SCO内部には章立てや目次についての定義を持たせる必要が一切なく、SCO自体の開発も容易です。SCO単位での前提条件の設定も可能になり、より細かい履修管理ができます。進捗の履歴についても、SCO単位で取ることができ、詳細な履歴管理が可能です。しかし、教材構造がかなり複雑になるため、コース構造ファイルの管理は煩雑です。各LMSにコース構造を管理する機能がない場合はかなり面倒になります。

## 3.2. 解説型コンテンツ

### 3.2.1. 解説型コンテンツの概要

解説型のコンテンツは、解説的な説明をテキスト・画像・音声・動画などを使用して表示・再生し、学習者はその内容を見たり聞いて学習を進めるという形式のコンテンツです。俗に「紙芝居型」「ページめくり型」などとも呼ばれ、e-ラーニングの教材コンテンツの基本形です。教材は複数のSCOから構成され、

各SCOはさらに複数のページから構成されるのが一般的です。LMSからSCOを起動すると、まずSCOの最初のページが表示され、SCO内で各ページ間の移動を行い、学習を進めていきます。また、ブックマーク機能を追加することで、任意のページに直接移動したり、LMSに記録されていた前回アクセス時のページに移動するといったことも実装可能です。

各ページの内容については、ブラウザで表示・再生できるものであれば何でも利用できます。単純なHTMLによるページに加え、JavaScript、Java アプレット、Directorムービー、Flashアニメーション、PDF文書、ActiveXコントロール、Authorware、動画(MPEG、RealVideo、Windows® Media、QuickTimeなど)などさまざまな形式で記述されたページを使用することが可能です。

解説型SCOでは、SCORM規格に未対応のコンテンツをそのままSCOとして使用することも可能です。しかしながら、より完成度の高い教材を作成するには、LMSとの通信を効率的に行う仕組みを組み込んでおくことが好ましいでしょう。LMSと通信する仕組みを実装するには、2.3節で示したSCORM APIを使用し、JavaScriptで記述するのが一般的です。

ここでは解説型SCOのコンテンツ記述方法を解説します。

### 3.2.2. 解説型コンテンツの典型的なロジック

ここでは、実際にSCOコンテンツを使って学習者が操作を進めるに当たり、どのような操作をしたときにどのような情報のやり取りをするべきかを、以下の場面ごとに説明します。

```
var mAPIAdapter = null;
var kNotInitialized = "301";
function findAPIAdapter(win){
    var        retVal = win.API;
    if (retVal == null){
        var        parentWindow = win.parent;
        if (parentWindow == null || win == parentWindow)
            retVal = null;
        else
            retVal = findAPIAdapter(parentWindow);}
    return retVal;
}

function getAPIAdapter(){
    var        retVal = mAPIAdapter;

    if (retVal == null){
        var        onMouseEvent = window.onerror;
        window.onerror = handleError;
        retVal = findAPIAdapter(window);
        if (retVal == null){
            var openerWindow = window.parent.opener;
            if (openerWindow != null)
                retVal = findAPIAdapter(openerWindow);}
        window.onerror = onMouseEvent;}

    return retVal;
}
```

図9. SCORM APIフレームの検索のためのスクリプト例

- 各ページ共通の初期化処理。
- SCO起動時にやり取りすべき情報。
- SCO内でページを移動するときにやり取りすべき情報。
- SCOを中断するときにやり取りすべき情報。
- SCOを終了するときにやり取りすべき情報。

以降の解説では、上記の各場面でSCOとLMSとの間でやり取りされるSCORM APIデータ・モデルについて具体的に説明します。

《各ページ共通の初期化処理》

SCO内のHTMLページの先頭では以下のような初期化処理を行う必要があります。

- LMSが提供するSCORM APIフレームの検索。
- SCORM APIをJavaScript関数として定義。
- LMSInitialize( )の呼び出し。

図9はLearningSpace5.01に添付されているSCORM APIフレームの検索のためのスクリプト例です。

《SCO起動時にLMSから情報を取得》

SCOはLMSに登録されている情報を取得し、コンテンツ内に表示したり、コンテンツの分岐の条件として使用できます。以下にSCOがLMSから取得できる主な情報を示します。

取得できる情報	CBT-APIモデル名
学習者の氏名	cmi.core.student_name
得点	cmi.core.score.raw
進捗ステータス	cmi.core.lesson_status
コンテンツ滞在時間	cmi.core.total_time
SCO固有情報	cmi.launch_data
SCO内ページ情報	cmi.core.lesson_location
その他の履歴情報の取得	cmi.suspend_data

学習者の氏名はLMS側の登録情報です。コンテンツ上に学習者名を表示したり、コンテンツの保護(正しい名前が取得できなければコンテンツの起動を中止するなど)といった用途に使えます。

得点・進捗ステータス・滞在時間は、SCOへのアクセスの履歴情報です。コンテンツ上にアクセス状況を表示したり、コンテンツのページ遷移の条件(例えば得点が80点以上でステータスが「合格」の場合は、合格者用のページに遷移するなど)といった用途に使えます。

SCO固有情報(ベンダー固有情報)は、あらかじめSCOの設定項目、SCOの説明、SCOの教材名など、SCOに固有の情報をLMS側で設定し(コース構造ファイルで定義)SCO側でこの情報を取得することにより、SCO独自の処理や表示を行うため

のもです。使用方法は特に限定されていないため、SCO開発者が任意に使うことが可能です。

SCO内ページ情報は、主に各SCO内部でのページ情報を保存するためのものです。例えばSCOを途中で終了したときにSCO内ページ情報を保存しておくことにより、次にアクセスしたときには前回終了時のページから再開するといった用途に使います。

そのほかの履歴情報は、得点・進捗ステータス・SCO内ページ情報などに保存できないISCO固有の履歴情報などを保存するためのものです。例えば、SCO固有情報(ベンダー固有情報)からデフォルトの設定値を取得し、学習者がその値を変更した場合、その情報を基に次にアクセスしたときは前回の設定内容でSCOが動作するといった用途に使えます。使用方法は特に限定されているわけではなく、SCO開発者が任意に使用することが可能です。

以下のサンプルは、LMSから学習者の氏名を取得し、それがNullであればエラー・メッセージを表示して終了するページに強制的に遷移するという処理を行います。LMSからSCOとして起動するのではなく、ブラウザからコンテンツのURLを直接入力したときに、コンテンツが表示されるのを回避するためのものです。

```
if (LMSGetValue("cmi.core.student_name")=="){
location.href='error.htm';}
```

以下のサンプルは、前回中断したときのページのURLをLMSから取得し、SCO起動時に直接中断したページに遷移するという処理を行います(中断したときのページの保存は、後述の「ページを変えるタイミングで途中進捗情報をLMSに送信」を参照)。

```
if (LMSGetValue("cmi.core.lesson_location")!="){
location.href=LMSGetValue("cmi.core.lesson_location");}
```

LMSInitializeの呼び出しや、上記のような先頭ページ必須の処理については、以下のようにHTMLの<body>タグのonloadイベントに記述しておくのが好ましいでしょう。

```
<html><head>
<Script Language='JavaScript'
function SCORMInitialize(){
LMSInitialize();
if (LMSGetValue("cmi.core.student_name")=="){
location.href='error.htm';}
if (LMSGetValue("cmi.core.lesson_location")!="){
location.href=LMSGetValue("cmi.core.lesson_location");}
}
</Script>
</head>
<body onload='SCORMInitialize()' onunload='LMSFinish()'>
</body></html>
```

《SCO起動時に初期情報をLMSに送信》

SCO起動時に、初期情報をLMSに送信できます。以下に送信することの多い主な情報を示します。

送信する情報	CBT-APIモデル名
進捗ステータス = 「開始済み」	cmi.core.lesson_status

進捗ステータス = 開始済み (lesson\_status = incomplete) を送信することにより、LMS側に「SCOを既に開始して学習している」という履歴を残すことができます。

lesson\_status = incompleteを送信するのは、通常は最初の1回だけにすべきです。ステータスを送信する前に、LMSから取得した履歴情報のlesson\_statusを確認し、開始も修了もしていない状態 (lesson\_status = not attempted) の場合のみ、lesson\_status = incompleteを送信するようにしておく必要があります。

以下にステータス「開始済み」を送信する例を示します。

```
if (LMSGetValue("cmi.core.lesson_status")=="not attempted")
{
    LMSSetValue( "cmi.core.lesson_status", "incomplete" );
    LMSCommit();
}
```

《ページを変えるタイミングで途中進捗情報をLMSに送信》

SCO内でページを変える(別のURLのhtmlファイルに遷移する)タイミングでSCO内ページ情報(現在のURL)を保存することで、次にアクセスしたときに、前回終了時のページから再開するといったことが可能となります(「SCO起動時にLMSから情報を取得」参照)。

送信する情報	CBT-APIモデル名
CBT内ページ情報	cmi.core.lesson_location

改ページ時のLMSへの送信例を示すと次のようになります。

```
LMSSetValue( "cmi.core.lesson_location", location.pathname );
LMSCommit();
```

《SCOを中断するときに進捗情報をLMSに送信》

SCO中断時にSCO開始時間からの学習時間を送信することで、正確な学習時間を履歴として残すことができます。

また、SCO固有の途中の履歴情報についてはcore.suspend\_dataデータ・モデルを使用できます。任意の情報の保存・再取り出しのために自由に使用することが可能です。

送信する情報	CBT-APIモデル名
そのほかの履歴情報の送信	cmi.suspend_data
コンテンツ滞在時間	cmi.core.time

以下に、中断時の滞在時間の送信例を示します。

```
function padNumber(num){
    if (num < 10)
        return "0" + num;
    else
        return num;}

now = new Date();
//startTimeはSCO開始時刻が保存されている変数
diff = Math.floor(now.getTime()/1000)-Math.floor(startTime.getTime()/1000);
minutes = Math.floor(diff / 60);
latency = padNumber(Math.floor(minutes / 60))+":"+padNumber(minutes % 60)+":"+ padNumber(diff % 60);
LMSSetValue( "cmi.core.session_time", latency );
LMSCommit();
LMSFinish();
```

SCO終了時には、LMSFinish( )を必ず呼び出す必要があるため、HTMLの<body>タグのonunloadイベントに記述しておくといいいでしょう。これにより、学習者がブラウザを強制終了したような場合でもLMSFinish( )が呼び出され履歴が正しく保存されます。

《SCOを終了するときに進捗情報をLMSに送信》

SCOを最後まで学習した場合、終了時には所要時間・ステータスを送信します。簡単なテストやクイズがあり、得点や回答(インタラクション)を送信する場合は3.3節で説明します。

進捗ステータス = 修了 (lesson\_status = completed) を送信することで、LMS側に「SCOを最後まで学習して修了した」という履歴を残すことができます。

送信する情報	CBT-APIモデル名
進捗ステータス = 「修了」	cmi.core.lesson_status
コンテンツ滞在時間	cmi.core.time

以下に、終了時のLMSへの送信例を示します。

```
function padNumber(num){
    if (num < 10)
        return "0" + num;
    else
        return num;}

now = new Date();
//startTimeはSCO開始時刻が保存されている変数
diff = Math.floor(now.getTime()/1000)-
Math.floor(startTime.getTime()/1000);
minutes = Math.floor(diff / 60);
latency = padNumber(Math.floor(minutes / 60))+":"+padNumber(minutes % 60)+":"+ padNumber(diff % 60);
LMSSetValue( "cmi.core.session_time", latency );
LMSSetValue( "cmi.core.lesson_status", "completed");
LMSCommit();
LMSFinish();
```

### 3.3. 演習問題型コンテンツ

演習問題型のコンテンツは、解説型コンテンツで学習した内容が定着しているかどうかを確認するための確認テストや、学習を始める前のスキル・チェックなどによく用いられます。コース教材は、複数の解説型SCOと演習問題型SCOから構成され、各SCOはさらに複数のページから構成されるのが一般的です。LMSからSCOを起動すると、まずSCOの1画面に1問、または1画面に複数問の演習問題が表示され、学習者はそれに回答していくことになります。

演習問題型コンテンツで使用される演習問題は、

- 正誤( × )問題
- 択一問題
- 複数選択問題
- 結び付け問題
- 穴埋め問題
- 記述問題

などのさまざまな形式で出題可能ですが、択一問題や複数選択問題が一般的です。また、演習問題コンテンツの位置付けにより、

- 各問題に回答後、正誤判定・正解・解説文を表示
- 全問回答後、回答のレビュー・やり直し
- 全問回答後、得点の表示

などの機能が求められる場合もあります。

回答方法は、HTMLのフォームのチェック・ボックス、ラジオ・ボタン、テキスト・ボックスを使用するのが一般的ですが、アイコンや画像をドラッグしたり、アイコン同士を線で結ぶといった方法もよく見受けられます。

#### 3.3.1. 演習問題型コンテンツの典型的なロジック

ここでは演習問題型コンテンツのSCOを作成するに当たり、SCOがLMSとの間でどのような情報のやり取りをするべきかについて説明します。

実際にSCOコンテンツを使って学習者が操作を進める際に、どのような操作をしたときにどのような情報のやり取りをするべきか、以下の場面ごとに説明します。

- 各ページ共通の初期化処理。
- SCO起動時にやり取りすべき情報。
- SCO内でページを移動するときにやり取りすべき情報。
- SCOを中断するときにやり取りすべき情報。
- SCOを終了するときにやり取りすべき情報。

上記の各場面のほとんどで、解説型コンテンツと共通な処理をするため、以降の解説では、演習問題型特有の処理のみを具体的に説明します。

《SCO起動時にLMSから情報を取得》

そのほかの履歴情報( suspend\_data )を使用して、これまでの学習者の回答を一時的に保持しておくことが可能です。演習問題を中断したときに、それまでの回答をcmi.suspend\_dataに保存し、再開時に前回までの回答を復元するといった使用方法となります。最後にまとめて集計するような場合や、回答のレビューややり直しを行う場合に有効です。

《ページを変えるタイミングで途中進捗情報をLMSに送信》

SCO内でページを変える(別のURLのhtmlファイルに遷移する)タイミングで、そのページでの問題への回答をセットするのが一般的です。

送信する情報	CBT-APIモデル名
演習問題の問題名	cmi.interactions.n.id
演習問題の問題タイプ	cmi.interactions.n.type
演習問題の正解	cmi.interactions.n.correct_responses.0.pattern
演習問題の学習者の回答	cmi.interactions.n.student_response
演習問題の正誤	cmi.interactions.n.result
演習問題の配点	cmi.interactions.n.weighting
演習問題の回答時間	cmi.interactions.n.latency

以下に改ページ時のLMSへの送信例を示します。

```
LMSSetValue( "cmi.interactions.1.id", InteractionID );
LMSSetValue( "cmi.interactions.1.type", "choice" );
LMSSetValue( "cmi.interactions.1.student_response",
QuizAnswer[StudentAnswer] );
LMSSetValue("cmi.interactions.1.correct_responses.0.pattern",
QuizAnswer[CorrectAnswer] );
LMSSetValue( "cmi.interactions.1.result", Judge );
LMSSetValue( "cmi.interactions.1.weighting", Weighting );
LMSSetValue( "cmi.interactions.1.latency", latency );
LMSCommit();
```

ページを変えるタイミングでLMSCommit( )も呼び出すかどうかは演習問題の位置付けに依存します。

全問回答後にレビューややり直しを許可したり、前の問題に戻って再回答を許可するようなコンテンツの場合は、ページを変えるタイミングでLMSCommit( )を呼び出すべきではありません。最終ページでLMSCommit( )を呼び出すことにより、回答情報は一括送信されます。

回答後に正誤判定や解説画面を表示し、再回答を許可しないコンテンツであれば、ページを変えるタイミングでLMSCommit( )を呼び出してもかまいません。この場合、各画面遷移ごとに回答情報が送信されます。

《SCOを中断するときに進捗情報をLMSに送信》

「SCO起動時にLMSから情報を取得」の説明を参照してください。

《SCOを終了するときに進捗情報をLMSに送信》

SCOを最後まで学習した場合、終了時には所要時間・ステータス・得点を送信します。各画面で回答情報を送信していない場合は、LMSCommit( )呼び出しのタイミングで回答情報も併せて送信します。

LMSに登録されている合格点によって、進捗ステータスには合格( passed )または不合格( failed )のいずれかを送信することになります。

送信する情報	CBT-APIモデル名
進捗ステータス = 「合格または不合格」	cmi.core.lesson_status
得点	cmi.core.score.raw
コンテンツ滞在時間	cmi.core.time

以下に、終了時のLMSへの送信例を示します(コンテンツ滞在時間については解説型のサンプル参照)。

```
LMSSetValue( "cmi.core.score.raw", TotalScore );
if (LMSGetValue("cmi.student_data.mastery_score") > TotalScore)
{
    LMSSetValue( "cmi.core.lesson_status", "failed" );
}
else
{
    LMSSetValue("cmi.core.lesson_status", "passed");
}
LMSCommit( );
LMSFinish( );
```

### 3.4. 教材コンテンツの作成方法

3.2.1節で触れたように、SCOコンテンツは、ブラウザで表示・再生できるものならどんな形式で作成してもかまいません。しかしながら、LMSとの通信の実装を考えた場合、SCORM APIを3.2節、3.3節で示したようにコンテンツに埋め込む必要があります。

コンテンツをSCORMに対応させるには、JavaScriptでAPIを直接記述する方法と、市販のSCORM対応オーサリング・ソフトウェアを使用する方法とがあります。ここではそれぞれの開発環境についての長所 / 短所を簡単に説明します。

#### • JavaScriptでAPIを直接記述

HTMLコンテンツの中にJavaScriptでAPIを直接記述する方法です。既存のHTMLコンテンツを再利用する場合は、この手法を用いることが多くあります。スクリプトによる記述となるため、自由度のかなり高いコンテンツ設計が可能となります。

ただ、基本的にはJavaScriptの高度な知識、およびAPIの使用法、データ・モデルの意味の十分な理解が必要となるため、すべての教材開発者がすぐ簡単に使えるというものではありません。

#### • Authorware( Macromedia )

Authorwareはマクロメディア社が提供するマルチメディア教材開発用のオーサリング・ツールです。CD-ROM教材作成用

として広く使われてきましたが、AICCと互換性のあるCMI関数をサポートしたことで、現時点ではe-ラーニング用の教材開発オーサリング・ツールとして最も広く使用されています。CMI関数と呼ばれるAICC互換の関数・変数を標準で数多く備えています。オーサリング方法自体はアイコンをフローチャート風に配置し、その中に簡単なスクリプトを記述するというもので、プログラミング経験の少ない教材作成者でも比較的簡単に修得することができます。

欠点としては、コンテンツを表示するためにブラウザに再生用プラグインをインストールしておかなければならないことと、また、SCORM APIを呼び出すにはgetUrl関数から直接SCORMのAPIを呼び出す形となるため、CMI関数が使用できない点です。

#### • Toolbook Assistant( Click2Learn )

Toolbook Assistantは、Click2Learn社が提供するe-ラーニングに特化したオーサリング・ツールです。PowerPointでプレゼンテーションを作るような感覚で容易に解説型 / 演習問題型コンテンツを作成できます。演習問題を作成する豊富な部品やウィザードを備えており、SCORMのAPIを意識することなくコンテンツを開発できます。どちらかというエンド・ユーザー向けのオーサリング・ツールといえます。プラグインが不要であることも長所といえます。

欠点としては、関数やAPIによるスクリプト作成機能がないため、自由度の高いコンテンツ作成には不向きなことです(スクリプト作成機能のあるToolbook Instructorという製品が予定されています)。

#### • LiveCreator( レイル )

レイル社が提供するe-ラーニングに特化したオーサリング・ツールです。コース構造の設計からSCOの作成までをトータルに行えます。スクリプト機能がなく、これもどちらかというエンド・ユーザー向けのオーサリング・ツールです。

欠点としては、Toolbook同様に、関数やAPIによるスクリプト作成機能がないため、自由度の高いコンテンツ作成には不向きなことです。また、コンテンツ再生にはActiveXコントロールを使用しているためNetscape環境では動作しません。

#### • FlashMX( Macromedia )

Flashは、マクロメディア社が提供するデファクト・スタンダードとなったアニメーション作成用オーサリング・ツールです。今日、数多くのWebサイトでFlashが使用されています。FlashMXという最新版では、SCORM対応のクイズを作成するためのテンプレートを提供しています。また、ActionScriptというスクリプト言語からSCORM APIの呼び出しも可能です。数多くのFlash技術者が存在している点や、Webの世界で市民権を得た環境ということで、今後はe-ラーニングのオーサリング・ツールとしても注目されます。

## 4. おわりに

冒頭で説明したように、e-ラーニングという言葉はすっかり市民権を得ました。検索エンジンで「e-ラーニング」と入力すると1万以上のサイトがヒットします。数多くの国内のLMSベンダーが、SCORMやAICCに準拠した製品を発表しています。『日経オープンシステム』誌2002年6月号のe-ラーニング特集記事で紹介されている国内13製品のうち、実に9製品がAICCまたはSCORMに対応しています。AICCのサイトには、AICCの規格に準拠した製品が紹介されていますが、80以上の製品がワールドワイドでAICC準拠をうたっています。

その一方で、e-ラーニングの市場にはまだ不透明感があることも否めません。ワールドワイドでも国内市場でも、LMSトップ・シェア5製品ほどを集めても全LMS市場の20～30%程度というのが実情です。残りの80%近くは標準仕様に対応していない独自仕様の製品であったり、エンド・ユーザーが自社内で開発した独自のシステムであったりします。トップ・シェアの数製品もすべてが標準仕様に対応しているわけではなく、LMS全体の市場から見た標準仕様採用LMSの割合は1割にも満たない可能性があります。

また、SCORMやAICCに対応した製品についても、各ベンダーでの標準規格の解釈の仕方の違いや、教材コンテンツ側制作サイドでの解釈の違いにより、必ずしも「すべての教材コンテンツがすべてのLMSで動作する」という標準仕様の目的が果たされているとはいえません。しかしながら、e-ラーニングのより一層の普及には、標準規格に対応したLMS、オーサリング・ソフトウェア、教材コンテンツが数多く出回ることが必要条件です。前述のような各ベンダー間での互換性の問題など、技術的な問題は速やかに解決されなければなりません。そのためには、SCORM/AICCなどの標準規格を熟知した技術者を数多く育成することが、e-ラーニングにかかわるベンダーに与えられた課題ではないかと筆者は考えます。

また、教材作成者の育成も重要な課題です。現状はAuthorwareなどの専用性の高いツールを使いこなせるごく一部の企業によって多くの教材コンテンツが作成されています。その一方で、一般のWebコンテンツの制作を行う「コンテンツ・クリエーター」と呼ばれる人たちは無数にいます。より使いやすく、技術情報が入手しやすく、安価なオーサリング・ツールがSCORMなどに対応することにより、彼らの力を借りることができます。結果として教材の供給能力も大幅に上がり、e-ラーニング市場も現状の問題点を打破できるのではと筆者は考えます。

本論文が、e-ラーニングに携わるIT技術者、コンテンツ・クリエーターの一助になれば幸いです。

(ページ数および表記上の観点から、著者の了解を得て編集部にて手を入れてあります)

### [参考文献]

- [1] 仲林 清「e-ラーニング技術の最新標準化動向 特集 進化するe-ラーニングの標準技術を知る」  
<http://www.atmarkit.co.jp/fengineer/special/newelestd/newelestd01.html>
- [2] Advanced Distributed Learning Initiative, Sharable Content Object Reference Model Version 1.2, The SCORM Run-Time Environment
- [3] Advanced Distributed Learning Initiative, Sharable Content Object Reference Model Version 1.2, The SCORM Content Aggregation Model