

# ¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.  
 Ese apoyo que siempre quiso tener...

## 1. Desarrollo de componentes y proyectos a medida



## 2. Auditoría de código y recomendaciones de mejora

## 3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



## 4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,  
HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)  
Gestor de contenidos (Alfresco)  
Aplicaciones híbridas

Tareas programadas (Quartz)  
Gestor documental (Alfresco)  
Inversión de control (Spring)

Control de autenticación y  
acceso (Spring Security)  
UDDI  
Web Services  
Rest Services  
Social SSO  
SSO (Cas)

JPA-Hibernate, MyBatis  
Motor de búsqueda empresarial (Solr)  
ETL (Talend)

Dirección de Proyectos Informáticos.  
Metodologías ágiles  
Patrones de diseño  
TDD

BPM (jBPM o Bonita)  
Generación de informes (JasperReport)  
ESB (Open ESB)



Powered by **autentia** Hosting patrocinado por **enredados**

- [Inicio](#)
- [Quienes somos](#)
- [Tutoriales](#)
- [Formación](#)
- [Empleo](#)
- [Colabora](#)
- [Comunidad](#)
- [Libro de Visitas](#)
- [Comic](#)

## NUEVO ¿Quieres saber cuánto ganas en relación al mercado? pincha aquí...

[Ver cursos que ofrece Autentia](#)

[Descargar comics en PDF y alta resolución](#)



[NUEVO!] 2008-04-07



2008-04-01



2008-03-25



2008-03-17

Estamos escribiendo un libro sobre la profesión informática y estas viñetas formarán parte de él. Puedes opinar en la seccion [comic](#).

### Tutorial desarrollado por



**Daniel Hernandez del Peso**

Consultor tecnológico de desarrollo de proyectos informáticos. Constructor de Adictos Al Trabajo 2.0

Ingeniero en Informática

Puedes encontrarme en [Autentia](#)

Somos expertos en Java/J2EE

### Catálogo de servicios de Autentia

[Descargar \(6,2 MB\)](#)

[Descargar en versión comic \(17 MB\)](#)

[AdictosAlTrabajo.com](#) es el Web de difusión de conocimiento de [Autentia](#).



[Catálogo de cursos](#)

Descargar este documento en formato PDF: [elearningScorm.pdf](#)

Fecha de creación del tutorial: **2008-04-08**

## E-learning y SCORM ®

### ¿qué es e-learning?

Aunque actualmente tendamos a pensar en e-learning como aprendizaje "on line" a través de internet, lo cierto es que, según ciertos expertos, se puede considerar e-learning a cualquier proceso de formación que emplee como medio cualquier tipo de tecnología (Web, CD interactivo, MP3, etc.). Citamos aquí la definición que la "American Society of Training and Developing" hace: "término que cubre un amplio grupo de aplicaciones y procesos, tales como aprendizaje basado en web, aprendizaje basado en ordenadores, aulas virtuales y colaboración digital. Incluye entrega de contenidos vía Internet, intranet/extranet, audio y vídeo grabaciones, transmisiones satelitales, TV interactiva, CD-ROM y más"

En cualquier caso, parece que la red reúne características idóneas para la transmisión del conocimiento. Los contenidos son fáciles de actualizar y, como todos los usuarios acceden a la versión del contenido que hay en el servidor, estas actualizaciones se distribuyen muy fácilmente. Por tanto es en este medio en el que nos vamos a centrar.

En el ámbito Web, podemos distinguir entre dos elementos básicos a la hora de hablar de e-learning: la *plataforma* y los *contenidos*. Plataformas hay muchas, algunas de pago, algunas gratuitas y algunas que ofrecen las dos posibilidades...

En cuanto a los contenidos, cada plataforma puede decidir cómo gestionarlos, la estructura que deben tener, etc. Para evitar la anarquía que podría producirse si cada cual hiciera lo que quisiese, se han definido estándares para la creación y estructuración de contenidos, que veremos más adelante.

### Plataformas de e-learning

Se conoce como "plataforma" al conjunto de recursos hardware y software orientados a la formación. Puede incluir temario de los cursos, tests y otros mecanismos de calificación del alumno, informes...

Además de la plataforma y los contenidos, es importante disponer mecanismos para la comunicación entre los distintos agentes implicados en el proceso de aprendizaje (trabajos en grupo, interacciones profesor-alumno...). También pueden existir mecanismos de gestión de cuentas de usuario, para asegurar que sólo los estudiantes registrados tengan acceso a los cursos, o incluso que un curso sólo sea accesible a determinados estudiantes registrados. Este conjunto de plataforma y todos los elementos accesorios se conoce como LMS (*Learning Management System*, o sistema de gestión del aprendizaje). Utilizaremos este término durante todo el documento

Entre las más conocidas, está [Moodle](#), aunque existen otras posibilidades como [ATutor](#), [.LRN](#), o [MobileTest](#), desarrollado por nuestro compañero [Carlos García](#).

Este tutorial se centrará más en los contenidos y los estándares definidos para ellos, y dedicaremos otro más adelante a las plataformas.

### Estándares

Como en cualquier otro ámbito de la informática (y de lo que no es informática), la estandarización favorece la integración de elementos heterogéneos y evita en la misma medida dolores de cabeza a los usuarios. En el caso de e-learning la estandarización nos permite trabajar con distintos proveedores de contenidos y de herramientas, favorece la reutilización, etc. con el ahorro de costes y tiempo que ello supone, tanto para proveedores como para clientes de contenidos

**Catálogo de servicios Autentia (PDF 6,2MB)**



En formato comic...




Web

[www.adictosaltrabajo.com](#)

Buscar

### Últimos tutoriales

2008-04-08  
[E-learning y SCORM ®](#)

2008-04-08  
[Comparativa de antivirus freeware y opensource](#)

2008-04-05  
[JMX y monitorización de JBoss](#)

2008-04-05  
[Jersey: la implemetación de RESTFull de Sun](#)

2008-04-05  
[Metro: pila de webservices de Sun. Integración con Maven 2](#)

2008-04-05  
[Metro: pila de webservices de Sun.](#)

2008-04-04  
[Espectaculares efectos visuales en el escritorio de Linux, con Compiz Fusion](#)

2008-04-04  
[Monitorización de Web Services con Glassfish Wsmonitor](#)

2008-04-04  
[Axis2. Ejemplo de creación de un servicio Web](#)

2008-04-03  
[Servicios Web RESTful en Axis 2](#)

### Últimas ofertas de empleo

2008-04-04  
[Banca - Genérico - MADRID.](#)

Existen varios estándares de e-learning, desarrollados por distintas organizaciones:

- AICC desarrollado por la industria de aviación de EEUU,
- IEEE LTSC, del Instituto de Ingenieros Electrónicos e Informáticos
- IMS, del Global Learning Consortium
- SCORM ®, que es el más extendido. Por eso, el tutorial lo orientaremos a este estándar.

## SCORM ®

SCORM ® son las siglas de "**Shareable Content Object Reference Model**" (modelo de referencia para contenidos compartibles)

Es un estándar definido por **ADL (Advanced Distributed Learning)** que aglutina especificaciones de diversas organizaciones. Reúne los estándares citados anteriormente (AICC, LTSC, IMS), además de las especificaciones de otras organizaciones (ARIADNE, ASD TPSMG), aunque todas estas especificaciones se han modificado ligeramente para dar mayor coherencia al resultado final.

Existen varias versiones del estándar, desde SCORM ® 1.0 a SCORM ® 2004, que sería el equivalente a 1.3.

La versión más reciente es SCORM ® 2004 3rd edition (o 1.3.3). En la página de ADL podéis descargar toda la documentación sobre el estándar, de la que este tutorial es breve resumen y a la que haré referencia a lo largo de todo el documento. Concretamente, nuestro esfuerzo se centrará en describir lo referente a la creación de contenidos, dejando un poco más de lado todo lo que afecta directamente al LMS. Si alguien desea saber más sobre esta parte, le remito a la documentación

### Componentes de especificación SCORM ®

Dentro de SCORM ®, se pueden distinguir tres partes, cada una de la cual trata un problema concreto que afecta a la creación y distribución de contenidos. Estos apartados son el modelo de agregación de datos, el entorno de ejecución y aspectos de la secuenciación y la navegación por los distintos elementos de un contenido.

#### Modelo de agregación de datos

El modelo de agregación de datos se refiere a tres elementos básicos: la estructura de los datos dentro del contenido y su empaquetado para la distribución, la definición de metadatos (información sobre la información) y la descripción del contenido en un formato entendido por el LMS.

En cuanto al empaquetado de contenidos, simplemente se trata de un fichero ZIP en el que se incluirán todos los recursos (páginas html, imágenes, etc.) con su estructura correspondiente, y un fichero descriptor del contenido, que se encontrará en la raíz del ZIP y que se llamará **imsmanifest.xml**. En él se dice al LMS de qué recursos consta un contenido, la relación entre ellos, etc. veremos la estructura de este descriptor un poco más tarde

#### Entorno de ejecución

El entorno de ejecución consta también de dos partes: por un lado está todo lo necesario para la comunicación entre los contenidos y el LMS y por otro toda la lógica interna del LMS que permite decidir que contenido lanzar, cómo hacerlo, etc. Como ya hemos dicho antes, de momento esta segunda parte no nos va a interesar, nos centraremos especialmente en la comunicación entre LMS y Contenido

Esta comunicación se realiza a través de un API JavaScript definido por ADL y AICC, que permite un diálogo entre ambos actores independientemente de la plataforma que estemos usando y de la herramienta con la que hayamos creado el contenido

#### Secuenciamiento y navegación

Cuando el usuario (estudiante) está visualizando un contenido, no tiene control total sobre las acciones que puede hacer. Es el creador del contenido el que define a qué elementos se puede saltar desde uno concreto, y es el entorno de ejecución el que tiene que interpretar estas instrucciones del creador para generar la secuencia de contenidos que se presentarán al estudiante y los eventos que hacen que la navegación a través de esta secuencia sea posible.

En cualquier caso, en este tutorial vamos a ver cómo define el creador del contenido esta información, dejando el tratamiento por parte del LMS para que el que lo desee lo vea en la documentación

### El modelo de agregación de datos de SCORM ®

Cuando se está creando un contenido que se adapte al estándar, el autor debe tener en cuenta una serie de obligaciones que le impone SCORM ®:

1. Deben poder enviarse vía navegador web
2. Deben ser autocontenidos, esto es, deben poder empaquetarse con todas sus dependencias
3. Debe ser independiente de lenguajes de servidor (JSP, ASP...)
4. Independiente de recursos externos, URL's...
5. No debe usar componentes que deban descargarse e instalarse por un administrador

Además, una vez creado el contenido teniendo en cuenta esos criterios, el autor debe generar el descriptor **imsmanifest.xml** para que la plataforma sepa interpretarlo. A continuación damos una descripción de la estructura general del manifest, aunque no se han incluido ciertos elementos opcionales, que podéis encontrar en la documentación (ya dije que íbamos a hacer referencia a ella a menudo)

#### Estructura básica del imsmanifest.xml

Antes de empezar a describir el manifest, hay que tener en cuenta una consideración: siempre que se hable de rutas a recursos dentro del contenido (recordemos que SCORM ® prohíbe que el contenido haga referencia a recursos externos), la ruta será relativa (y con formato de URL, o sea, usando "/" como separador de directorios) a la raíz del fichero ZIP. Es el LMS el que debe preocuparse por conocer la ruta absoluta hasta un contenido.

La estructura se va a pintar como si fuera un documento XML. Para los hijos de un elemento, si no tiene formato de etiqueta (si el nombre no va entre "<" y ">"), es que se trata de un atribuido. Para cada elemento se da una descripción breve de su significado, para un conocimiento en profundidad... acudid a la documentación (seguro que ya sabáis lo que iba a poner... :-P )

- **<manifest>** : elemento raíz
  - **identifíer**: atributo obligatorio que contiene el identificador único del contenido
  - **version**: atributo opcional que contiene el número de versión del contenido
  - **<metadata>** : esquema y versión de SCORM ® utilizados. Obligatorio
    - **<schema>** Obligatorio. Dice que el contenido se ajusta al estándar SCORM ®
    - **<schemaversion>** Obligatorio. Indica la versión de SCORM ® a la que se ajusta el contenido
    - **<adlcp:location>** Opcional. Referencia a un XML donde se encuentra la definición de los metadatos. Pueden aparecer varios.
    - **<lom:lom>** Opcional. Metainformación del documento definida en el propio documento. Se definirá este elemento más adelante, cuando se expliquen los metadatos
  - **<organizations>** Obligatorio. Define un árbol de actividades, sin límite de profundidad. Las hojas del árbol (actividades que no tienen hijos) son las que se envían al estudiante y, por tanto será un recurso "lanzable" (ver "resources"). Al conjunto de una actividad y todas sus descendientes se le llama *cluster*. Para un contenido concreto,

2008-04-03  
Banca - Genérico - MADRID.

2008-04-02  
T. Información - Analista / Programador - MADRID.

2008-04-02  
T. Información - Analista / Programador - MADRID.

2008-03-29  
T. Información - Analista / Programador - MADRID.

Anuncios Google

pueden definirse varios árboles de actividad. Serán estos árboles de actividad los que se empleen para la parte de secuenciamiento y navegación. En cualquier caso, hay que tener en cuenta que esta organización de los contenidos es lógica. La organización física se define en el apartado "resources".

- **default** : Atributo obligatorio que contiene el identificador de la organización por defecto
- **<organization>** Al menos una obligatoriamente por *content package*.
  - **identifier** Atributo obligatorio, con el identificador único (dentro del manifest) de la organización
  - **structure**: Este atributo es opcional. Si no se especifica, su valor por defecto es "hierarchical"
  - **<title>** Obligatorio. Representa el título de la organización
  - **<item>** Obligatorio al menos uno. Define una actividad del árbol.
    - **identifier** identificador de la actividad (único en el manifest)
    - **identifierref** identificador del recurso asociado a la actividad (obligatorio sólo para las hojas)
    - **parameters** Cadena de parámetros de invocación del recurso (se usa en las hojas del árbol de actividad, es opcional)
    - **invisible** Valor por defecto "true". Indica si este elemento debe pintarse cuando la estructura del paquete sea mostrada
    - **<title>** Obligatorio. Título de la actividad
    - **<item>** Opcional. Actividad o actividades hijas (tienen la misma estructura que la que se está describiendo)
    - **<metadata>** Opcional. Si aparece, solo puede haber uno. Metainformación de la organización
      - **<adlcp:location>** Opcional. Referencia a un XML donde se encuentra la definición de los metadatos. Pueden aparecer varios.
      - **<lom:lom>** Opcional. Metainformación del documento definida en el propio documento. Se definirá este elemento más adelante, cuando se expliquen los metadatos
    - **<imsss:sequencing>** Opcional. Información sobre el secuenciamiento del elemento. Por su complejidad y extensión, este apartado se trata aparte
  - **<metadata>** Opcional. Metainformación de la organización
    - **<adlcp:location>** Opcional. Referencia a un XML donde se encuentra la definición de los metadatos. Pueden aparecer varios.
    - **<lom:lom>** Opcional. Metainformación del documento definida en el propio documento. Se definirá este elemento más adelante, cuando se expliquen los metadatos
- **<resources>** Obligatorio. Un contenido está compuesto por unidades de aprendizaje. Cada una de estas unidades es un *resource*. Una unidad está compuesta por una página o documento visualizable por un navegador Web que puede usar otros ficheros interpretables por el navegador (flash, JS, imágenes...), y no pueden ser páginas que necesiten procesamiento en el lado servidor (JSP, ASP, PHP). Pueden ser "lanzables" o "no-lanzables". Son "lanzables" si definen un fichero web como punto de lanzamiento (si tienen atributo "href")
  - **xml:base** : Atributo opcional. Ruta relativa para todos los recursos
  - **<resources>** Obligatorio al menos 1
    - **id** Atributo obligatorio. Identificador único del recurso dentro del manifest
    - **type** : Atributo obligatorio. tipo de recurso. Fijar a "webcontent"
    - **adlcp:SCORM @Type** Atributo obligatorio. Tiene dos valores posibles: "asset" si el recurso no necesita acceder al entorno de ejecución de SCORM ®. Si necesita acceder a dicho entorno de ejecución, el tipo es "sco" (SCO = *Shareable Content Objects*)
    - **href** : Atributo opcional. Indica la URL del punto de lanzamiento del recurso (por ejemplo, la página HTML que contiene el texto)
    - **xml:base** : Atributo opcional. Ruta relativa para todos los ficheros del recurso
    - **<metadata>** Tendrá uno o ninguno.
    - **<file>** uno o más (al menos uno). Ficheros incluidos en el recurso
      - **href**: Atributo obligatorio que indica el path del fichero que se está definiendo
    - **<dependency>**. En caso de haber dependencias, deben declararse debajo de todos los ficheros
      - **identifierref** : atributo obligatorio. Es el identificador de otro recurso (*resource*) del paquete

#### Elemento <sequencing> en el imsmanifest.xml

Ya hemos dicho que este es un elemento bastante complejo. Define toda la información que necesita el LMS para calcular la secuencia de contenidos que se presentará al usuario, etcétera.

Aquí podéis ver el aspecto que tiene la declaración del secuenciamiento en el imsmanifest.xml:

- **<imsss:sequencing>** Opcional. Información sobre las reglas de secuenciamiento del elemento
  - **ID**: Atributo Opcional, identificador único en todo el manifest
  - **IDRef** Atributo Opcional, identificador de un conjunto de secuenciamiento reutilizable definido en algún otro punto del XML
  - **<controlMode>** Especifica el modo de control para el elemento
    - **choice**: opcional, valor "true" por defecto. Si esta a "true", el usuario puede elegir a que actividad saltar
    - **choiceExit**: opcional, valor "true" por defecto. Cuando está activo, permite terminar una actividad si el usuario desea saltar a otra (usando "choice")
    - **flow**: opcional, valor "false" por defecto. Con valor "true", permite navegar secuencialmente (enlaces tipo "anterior-siguiente") entre las actividades hijas de la que declara el atributo
    - **forwardOnly**: opcional, valor "false" por defecto. Con valor "true", indica que no se puede acceder a actividades anteriores en el árbol de actividad
    - **useCurrentAttemptObjectiveInfo** opcional, valor "true" por defecto. Indica, si vale "true", que únicamente se tendrá en cuenta el progreso en la consecución del objetivo de las actividades hijas obtenido en el intento (attempt) actual.
    - **useCurrentAttemptProgressInfo** opcional, valor "true" por defecto. Con valor "true", significa que sólo se

tendrá en cuenta el progreso en las actividades hijas obtenido en el intento ("attempt") actual

- **<sequencingRules>** Define el comportamiento del secuenciamiento de una actividad. Cada actividad puede definir un número ilimitado de reglas de navegación
  - **<preConditionRule>** Se usan para decidir si una actividad se debe enviar al usuario. Pueden definirse n precondiciones
    - **<ruleConditions>**
      - **conditionCombination** Atributo opcional. Por defecto, el valor es "all". Especifica cómo deben combinarse las reglas definidas. Si toma el valor "any", equivale a un OR entre las reglas. Si vale "all", equivale a un AND
      - **<ruleCondition>**
        - **referencedObjective** (opcional) identificador de un objetivo local asociado a la actividad. Se usará su estado para la evaluación de las condiciones. Si no se declara este atributo, se toma por defecto el objetivo declarado como principal
        - **measureThreshold** (opcional) valor umbral para las condiciones basadas en medidas. Debe tomar un valor entre -1'0000 y 1'0000, con una precisión de al menos 4 decimales
        - **operator** (opcional, por defecto "noOp"). Puede tomar como valor "not" o "noOp". "not" niega la condición y "noOp" la deja como está
        - **condition** (obligatorio) Valores posibles:
          - **satisfied**
          - **objectiveStatusKnown**
          - **objectiveMeasureKnown**
          - **objectiveMeasureGreaterThan**
          - **objectiveMeasureLessThan**
          - **completed**
          - **activityProgressKnown**
          - **attempted**
          - **attemptLimitExceeded**
          - **timeLimitExceeded**
          - **outsideAvailableTimeRange**
          - **always**
      - **<ruleAction>** establece el comportamiento deseado si la condición se evalúa a "true"
        - **action** Obligatorio. Puede tomar uno de los valores siguientes
          - **skip**
          - **disabled**
          - **hiddenFromChoice**
        - **topForwardTraversal**
    - **<exitConditionRule>** reglas de secuenciamiento que se aplican cuando acaba el intento (attempt) de una actividad hija
      - **<ruleConditions>** Es igual que en el caso anterior
      - **<ruleAction>** valores posibles:
        - **exit**
    - **<postConditionRule>** descripción de acciones que controlan las decisiones de secuenciamiento cuando el intento (Attempt) termina
      - **<ruleConditions>** Es igual que en el caso anterior
      - **<ruleAction>** valores posibles
        - **exitParent**
        - **exitAll**
        - **retry**
        - **retryAll**
        - **continue**
        - **previous**
  - **<limitConditions>** Condición de finalización de la actividad
    - **attemptLimit** atributo opcional, indica el máximo número de intentos para la actividad. Debe ser un número entero positivo
    - **attemptAbsoluteDurationLimit** Tiempo máximo que puede emplear un estudiante en superar una actividad. Sólo cuenta el tiempo activo, no cuenta el tiempo que la actividad esté suspendida
  - **<rollupRules>** conjunto de reglas que controlan la relación entre el progreso en una actividad y el progreso de su actividad padre
    - **rollupObjectiveSatisfied** Atributo opcional, valor por defecto "true". Indica que el estado de objetivo satisfecho de la actividad está incluido en el progreso de la actividad padre
    - **rollupProgressCompletion** Atributo opcional, por defecto "true". Cuando está activo significa que el estado de completitud del intento ("attempt") en la actividad está incluido en el de su actividad padre
    - **objectiveMeasureWeight** Atributo opcional, valor por defecto 1.0000. Puede tomar valores entre 0.0000 y 1.0000, con al menos 4 decimales de precisión. Representa el peso que se asigna al progreso en esta actividad al contarla en la actividad padre
    - **<rollupRule>**
      - **childActivitySet** Atributo opcional, con valor por defecto "all". Representa qué valores de las reglas



restricciones sólo son validas para la actividad para la cual se definen

- **preventActivation** Atributo opcional (valor por defecto "false") que indica que no deben iniciarse intentos en actividades hijas hasta que la actividad actual sea su padre
- **constrainChoice** Opcional, con valor por defecto "false". Solo permite el acceso a actividades que puedan considerarse "siguientes" de la actividad con esta restricción
- **<adlseq:rollupConsiderations>** describe cuándo una actividad debe incluirse en el proceso de "rollup"
  - **requiredForSatisfied** Opcional, valor por defecto "always". Indica la condición bajo la cual la actividad es incluida en el cálculo de satisfacción de su actividad padre
  - **requiredForNotSatisfied** Opcional, valor por defecto "always". Indica la condición bajo la cual la actividad debe incluirse en la evaluación de la no-satisfacción de una regla de "rollup" de la actividad padre.
  - **requiredForCompleted** Opcional, valor por defecto "always". Indica la condición bajo la cual la actividad debe incluirse en la evaluación de la completitud de una regla de "rollup" de la actividad padre.
  - **requiredForIncomplete** Opcional, valor por defecto "always". Indica la condición bajo la cual la actividad debe incluirse en la evaluación de la incompletitud de una regla de "rollup" de la actividad padre.
  - **measureSatisfactionIfActive** Opcional, valor por defecto "true". Si está activo, quiere decir que la medida ("measure") debe usarse para determinar la satisfacción durante el "rollup" cuando la actividad está activa
  - Todos los atributos anteriores, a excepción del último, pueden tomar uno de estos valores:
    - Always
    - ifAttempted
    - ifNotSkipped
    - ifNotSuspended

### Metadatos en SCORM ®

Los metadatos son utilizados para, por ejemplo, dar al usuario información sobre el tipo de contenido que está visualizando (si es un curso, o una lección de un curso mayor...). Además, el sistema puede usar esa misma información para otras cosas como presentar al usuario el componente necesario para visualizar el tipo de contenido. SCORM ® reconoce LOM (Learning Object Metadata), definido por el IEEE, como estándar de facto en la definición de metadatos, y recomienda su uso por encima de otras opciones (aunque no prohíbe usarlas). Por tanto, siguiendo el consejo de SCORM ®, vamos a fijarnos en LOM a la hora de aprender a definir metadatos.

LOM organiza la información en nueve categorías:

1. General (General): Información general sobre el contenido
2. Ciclo de Vida (Life Cycle): Información sobre la historia y el estado actual del fichero, así como aquellos que han manipulado el contenido
3. Meta-metadato (Meta-metadata): Se utiliza para proporcionar información sobre los metadatos en sí, en lugar de sobre el componente que se está describiendo
4. Técnica (Technical): describe requerimientos técnicos y características del componente
5. Educativa (Educational): informa sobre las características educativas y pedagógicas del contenido
6. Derechos (Rights): Informa sobre los derechos de propiedad intelectual
7. Relación (Relation): Define las relaciones entre distintos componentes
8. Anotación (Annotation): comentarios e información sobre el uso pedagógico del componente, e incluso sobre los autores de los comentarios
9. Clasificación (Classification): Indica cuándo un componente se ajusta a un sistema de clasificación concreto

La estructura básica de un metadato es la siguiente:

- **<lom>**
  - **<general>**
    - **<identifier>** : Es un mecanismo para asignar un identificador único al componente
      - **<catalog>** : representa el esquema usado para crear y gestionar la entrada (entry). Algunos valores posibles: URI, URN, DOI (Digital Object Identifier) ISBN, ISSN (Internationa Standard Serial Number)
      - **<entry>** Valor del identificador
    - **<title>** título de la unidad de aprendizaje. Puede aparece como máximo una vez.
    - **<language>** representa el idioma en que está escrito el contenido. Se usará la notación "codLenguaje[-subcodLenguaje]". Por ejemplo, "es-ES" o "en-US". El código de lenguaje se define en la norma ISO 639:1988. El subcódigo se define en la norma ISO 3166-1997
    - **<description>** : descripción textual del componente que se está describiendo
    - **<keyword>** : Palabras o frases claves para describir la unidad de aprendizaje
    - **<coverage>** : época, región, cultura... sobre la que trata el artículo
    - **<structure>** : Aparece como máximo una vez. Describe la organización interna del componente. Puede tomar los valores "atomic" (un solo elemento indivisible), "hierarchical" (una serie de elementos que puede representarse como un árbol), "collection" (conjunto de elementos sin relación entre ellos), "networked" (conjunto de objetos relacionados, aunque la relación no se especifica) o "linear" (conjunto de elementos ordenados, con relaciones de tipo Anterior - Siguiente)
      - **<source>** : fuente de la que se extraen los valores. Valor fijo de "LOMv1.0"
      - **<value>** : un valor de los especificados anteriormente
    - **<aggregationLevel>** : Uno como máximo. Nivel de granularidad de la unidad de aprendizaje. Los valores posibles son 1 (datos sueltos), 2 (una lección de un curso), 3 (un curso) o 4 (conjunto de cursos para preparar una certificación)
      - **<source>** : fuente de la que se extraen los valores. Valor fijo de "LOMv1.0"
      - **<value>** : un valor de los especificados anteriormente
  - **<lifeCycle>**
    - **<version>** : aparece como mucho una vez. Indica el número de revisión del componente
    - **<status>** : Puede tomar los valores draft (borrador), final (es una versión cerrada), revised (el componente se ha revisado desde la última versión) o unavailable (información de estado no disponible). Aparece como máximo una vez
      - **<source>** : fuente de la que se extraen los valores. Valor fijo de "LOMv1.0"
      - **<value>** : un valor de los especificados anteriormente



Dicha interacción se efectúa a través de un API JavaScript, que se carga en el navegador Web, y que debe ser descubierto por los SCO, que son abiertos por el LMS en una ventana nueva (pop up) o en un frame. Para que el contenido pueda detectar el objeto JavaScript que expone las funciones del API, el estándar fija dos normas:

1. El objeto debe ser accesible por DOM
2. El nombre del objeto es fijo, "API\_1484\_11"

A parte de esas dos normas, el estándar sólo fija la interfaz. Los detalles de la implementación son responsabilidad del LMS

Para localizar el API, el SCO sólo tiene que recorrer la cadena de ventanas antecesoras hasta encontrar el objeto deseado. La propia documentación de SCORM ® propone un [script](#) que permite localizar el objeto que contiene la instancia de API:

```

view plain print ?
01. var nFindAPIAttempts = 0;
02. var API = null;
03. var maxTries = 500;
04. var APIVersion = "";
05. // The ScanForAPI() function searches for an object named API 1484 11
06. // in the window that is passed into the function. If the object is
07. // found a reference to the object is returned to the calling function.
08. // If the instance is found the SCO now has a handle to the LMS
09. // provided API Instance. The function searches a maximum number
10. // of parents of the current window. If no object is found the
11. // function returns a null reference. This function also reassigns a
12. // value to the win parameter passed in, based on the number of
13. // parents. At the end of the function call, the win variable will be
14. // set to the upper most parent in the chain of parents.
15. function ScanForAPI(win) {
16.     while ((win.API_1484_11 == null) && (win.parent != null) && (win.parent != win)) {
17.         nFindAPIAttempts++;
18.         if (nFindAPIAttempts > maxTries) {
19.             return null;
20.         }
21.         win = win.parent;
22.     }
23.     return win.API_1484_11;
24. }
25.
26. // The GetAPI() function begins the process of searching for the LMS
27. // provided API Instance. The function takes in a parameter that
28. // represents the current window. The function is built to search in a
29. // specific order and stop when the LMS provided API Instance is found.
30. // The function begins by searching the current window's parent, if the
31. // current window has a parent. If the API Instance is not found, the
32. // function then checks to see if there are any opener windows. If
33. // the window has an opener, the function begins to look for the
34. // API Instance in the opener window.
35. function GetAPI(win) {
36.     if ((win.parent != null) && (win.parent != win)) {
37.         API = ScanForAPI(win.parent);
38.     }
39.     if ((API == null) && (win.opener != null)) {
40.         API = ScanForAPI(win.opener);
41.     }
42.     if (API != null) {
43.         APIVersion = API.version;
44.     }
45. }

```

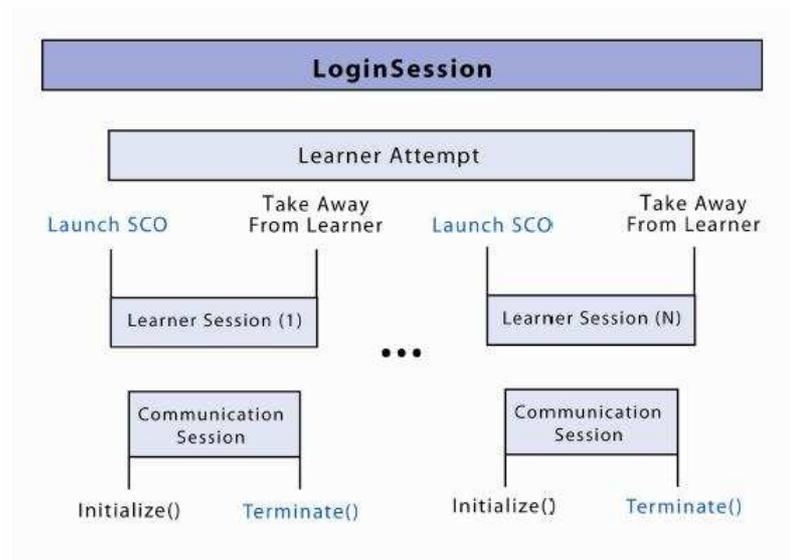
[Script provisto por ADL](#)

### Modelo Temporal

Al describir la relación temporal del usuario (estudiante) con el sistema (más concretamente con un contenido), SCORM ® define cuatro conceptos básicos para ayudar a seguir la pista del estudiante durante el aprendizaje

- Learner Attempt (Literalmente, intento del estudiante): es el esfuerzo del estudiante para satisfacer los requisitos de la unidad de aprendizaje a la que pertenece el contenido. Puede abarcar una o varias sesiones de aprendizaje (learner sessions) o interrumpirse entre sesiones
- Learner Session (sesión de aprendizaje): es el tiempo durante el cual un estudiante accede ininterrumpidamente al contenido
- Communication Session (sesión de comunicación): tiempo de conexión activa entre el contenido y el API
- Login Session: tiempo desde que el estudiante accede al sistema hasta que lo abandona

Veamos un gráfico extraído de la documentación de SCORM ® que facilita la comprensión de estos conceptos y sus relaciones



**Figure 2.1.1a: Temporal Model Relationships for a Specific SCO**

Como se puede ver, en la imagen se habla de SCO's... Si el contenido fuera un Asset (sin conexión con el API), sería válido quitando la parte de communication sessions, que no tienen sentido en este tipo de contenidos.

En cuanto a los datos que se manejan durante cada "Attempt", es responsabilidad del LMS decidir que hacer con ello una vez que ha acabado. Puede eliminarlos, persistirlos... el estándar no le obliga a nada. El único requisito que debe cumplir el LMS es mantener los datos en el caso de que el "attempt" se suspenda temporalmente

#### "Ejecución" de contenidos

Entendemos por ejecución el proceso por el que el LMS presenta un contenido en el navegador del usuario. Es responsabilidad del LMS, usando el descriptor visto anteriormente (**imsmanifest.xml**), determinar, tras un determinado evento, el siguiente contenido que debe mostrarse, y cómo hacerlo en función de su tipo. Las políticas para determinar el contenido adecuado son variadas. Puede ser el usuario el que determine la siguiente actividad que desea visualizar, o pueden seguir un orden secuencial, o calcularse en función de la "calificación" del usuario en actividades anteriores.

Finalmente, una actividad estará asociada a un contenido, que el LMS deberá ser capaz de localizar y presentar en el navegador. Recordemos que todas las rutas del descriptor son, según marca el estándar, relativas. Por tanto es también responsabilidad del sistema calcular la ruta completa de manera correcta (incluidos los parámetros en caso de que fueran necesarios).

En el caso de Assets, el LMS simplemente se tiene que encargar de localizar y enviar el contenido al cliente. Pero en el caso de SCO's, el proceso es más complejo. Por un lado, el modelo de lanzamiento de SCORM ® no permite la ejecución (y seguimiento) de más de un SCO por estudiante a la vez.

#### El API

Los métodos disponibles por el SCO para interactuar con el LMS están divididos en tres tipos

- Métodos de sesión. Inician o finalizan la sesión
  - Initialize("")
    - Inicia la sesión de comunicación (Communication session)
    - Parámetros: la cadena vacía
    - Devuelve "true" o "false", según la conexión con el LMS se haya realizado correctamente o no. En caso de error, fija el código de error al valor correspondiente
  - Terminate("")
    - Termina la sesión de comunicación. Al finalizar, hace automáticamente un "commit" de la información enviada por el SCO desde el último commit o desde la inicialización de sesión
    - Como parámetro recibe la cadena vacía
    - Devuelve "true" si el cierre de la conexión es correcto, o "false" en el caso de que hay algún error. Si esto se produce, fija el código de error al valor correspondiente
- Métodos de transferencia de datos. Sólo pueden llamarse mientras haya una sesión iniciada
  - GetValue(dato)
    - Solicita por parte del SCO información al LMS. La información solicitada puede ser valores de datos del modelo, versión del modelo utilizada, elementos soportados, ...
    - Recibe como parámetro el identificador del dato que se quiere recoger
    - Devuelve el valor solicitado o la cadena vacía en caso de error. En este último caso, el API se encarga de asignar el código de error apropiado
  - SetValue(dato, valor)
    - Envía al LMS el valor pasado como parámetro al dato que se indica en el primer parámetro. El API puede persistir al servidor el dato inmediatamente o almacenarlo en una caché local
    - Envía como parámetros el nombre del dato que se desea asignar, y su valor. El valor se envía como string, pero debe ser convertible al tipo definido para el dato
    - Devuelve "true" o "false", según haya ido todo correcto o no. En caso de error, el API asigna el código de error adecuado.

- Commit("")
  - Persiste en servidor cualquier dato que estuviera cacheado. Si no hubiera caché, retorna "true" y fija el código de error a 0 (cero), sin hacer más procesamiento
  - Recibe la cadena vacía como parámetro
  - Devuelve "true" (éxito) o "false" (error). En caso de error, asigna el código de error adecuado
- Métodos de soporte. Aportan soporte de errores, incluso fuera de sesión
  - GetLastError()
    - Solicita el código de error para el estado actual del API. Este método no afecta al estado del API al ser llamado
    - No acepta parámetros
    - Devuelve un string convertible a un número entero en el rango 0 (no error) – 65536, ambos inclusive. Los posibles códigos de error se encuentran definidos en la documentación
  - GetErrorString(error)
    - Obtiene una descripción textual de un error
    - Recibe como parámetro un string con el código de error
    - Devuelve la descripción del error (hasta 255 cars) o la cadena vacía si el código de error es desconocido. El estándar sólo fija los códigos de error, los mensajes textuales son responsabilidad del LMS
  - GetDiagnostic(parámetro)
    - Función de uso específico del LMS. Permite al LMS definir información adicional para diagnóstico a través del API
    - El parámetro puede ser un código de error, pero no tiene por qué serlo. Es un string definido por el desarrollador del API de hasta 255 caracteres
    - Devuelve la información de diagnóstico (255 caracteres) o la cadena vacía si el parámetro no se reconoce

Además, SCORM ® también define una serie de errores y excepciones que pueden generar las distintas llamadas al API (y que el propio API es capaz de devolver y consultar como ya hemos visto al hablar de los métodos de soporte), que pueden consultarse (como no) en la documentación de SCORM ®

### Responsabilidades

Es responsabilidad del LMS hacer que haya una instancia del API localizable a través de DOM, con el nombre API\_1484\_11. Ya se ha explicado anteriormente el mecanismo para localizar la instancia de API desde el SCO (buscando en las ventanas/frames padres o en los padres de la ventana que abrió la ventana que contiene el SCO, en el caso de popups).

Además, el estándar obliga a que DOM contenga un atributo llamado "version". Los valores de este atributo deben ser una serie de números enteros (indicando los números de versión/release) separados por punto (.). Para indicar conformidad con el estándar de IEEE, los tres primeros caracteres del atributo deben ser "1.0". Si se quiere dar información sobre la versión de la implementación, el cuarto carácter debe ser otro punto (.), por ejemplo (1.0.1)

Es responsabilidad del SCO ser capaz de encontrar la instancia del API con el mecanismo explicado anteriormente. Una vez localizada la instancia, el SCO debe realizar al menos un Initialize y un Terminate

### Modelo de datos (Data Model) del entorno de ejecución

El motivo para definir un modelo de datos es asegurar el correcto funcionamiento de los SCO en LMS distintos. Para diferenciarlos, todos los elementos del modelo de datos de SCORM ® empiezan "cmi.", y emplean el carácter "punto" (.) para separar unidades semánticas dentro del nombre del elemento. Por ejemplo:

#### cmi.interactions.3.learner\_response

significa que es un elemento del modelo de SCORM ® (empieza por "cmi."), concretamente el elemento "interactions", que es una colección de registros con varios campos. Se está accediendo al elemento de la colección con índice 3, y dentro de ese registro, al campo "learner\_response"

Además, el modelo de datos tiene tres palabras clave:

- `_version`: esta palabra clave se emplea para conocer la versión del modelo de datos soportada por el LMS. Para consultar dicha versión, habría que hacer la siguiente llamada: `GetValue("cmi._version")`
- `_count`: consulta el número de elementos en una colección
- `_children`: extrae todos los elementos de un modelo de datos dependiente de un elemento padre que son soportados por el LMS

La especificación del modelo de datos completo es demasiado extensa como para ponerla en un tutorial, por lo que remito a la documentación a todo el que esté interesado en ello.

## Conclusiones

Ya sabemos algo más sobre el amplio mundo del e-learning, y conocemos un estándar para que nuestros contenidos sean más fácilmente distribuibles... En otros tutoriales veremos cómo crear en la práctica contenidos que cumplan con SCORM ®, y veremos también distintas plataformas existentes gratuitas... Entre tanto, si tienes dudas sobre lo que hemos explicado, ya sabes que [puedes contratarnos](#) para ayudarte a resolverlas...

## Referencias

- Imagen y Script extraídos de la fuente: [Advanced Distributed Learning \(ADL\)](#), Sharable Content Object Reference Model (SCORM®) 2004 3rd Edition Run-Time Environment Version 1.0, 2006.
- Source (for image and script): [Advanced Distributed Learning \(ADL\)](#), Sharable Content Object Reference Model (SCORM®) 2004 3rd Edition Run-Time Environment Version 1.0, 2006.

- [Puedes opinar sobre este tutorial haciendo clic aquí.](#)
- [Puedes firmar en nuestro libro de visitas haciendo clic aquí.](#)
- [Puedes asociarte al grupo AdictosAlTrabajo en XING haciendo clic aquí.](#)
- [Añadir a favoritos Technorati.](#) 



Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

## Recuerda

**Autentia** te regala la mayoría del conocimiento aquí compartido ([Ver todos los tutoriales](#)). Somos expertos en: J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ... y muchas otras cosas.

**¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?, ¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?**

**Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos ...**

Autentia = Soporte a Desarrollo & Formación.

[info@autentia.com](mailto:info@autentia.com)



## Servicio de notificaciones:

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales.

Formulario de suscripción a novedades:

E-mail

## Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	pdf
<a href="#">Gestión documental</a>	Este tutorial nos va a hablar sobre la gestión documental que podemos incluir dentro de nuestras aplicaciones	2007-05-23	2441	<a href="#">pdf</a>
<a href="#">Introducción al software libre y/o de código abierto</a>	Este tutorial pretende ser una introducción al software libre (free software) y/o de código abierto (open source), y a algunas de las licencias más habituales.	2007-05-21	1555	<a href="#">pdf</a>
<a href="#">Documentar código Java con JavaDoc</a>	Os mostramos como utilizar los comentarios y etiquetas de JavaDoc para documentar programas Java.	2003-07-13	14932	<a href="#">pdf</a>
<a href="#">Fuentes de Documentación para OpenCms 7</a>	OpenCms es un gestor de contenidos potente y altamente configurable, y este tutorial muestra una selección de documentación útil para su aprendizaje	2008-03-17	357	<a href="#">pdf</a>
<a href="#">Extracción de texto de documentos Office desde Java</a>	En este tutorial vamos a ver como podemos extraer texto de los documentos de Office (DOC, XLS y PPT) desde Java	2007-05-22	5083	<a href="#">pdf</a>
<a href="#">Primeros pasos con Log 4 NET</a>	En este tutorial se muestra como almacenar las trazas de una aplicación con ayuda de Log 4 Net.	2005-12-15	7526	<a href="#">pdf</a>
<a href="#">Todo está en los libros</a>	Este es un atípico tutorial en nuestro Web donde, a través de la interpretación personal de obras de psicología y estrategia actuales, os invitamos a aprender a apreciar este tipo de libros, lo que seguro contribuirá a vuestra evolución profesional.	2005-01-16	9874	<a href="#">pdf</a>
<a href="#">Bibliografía básica recomendada</a>	Os comentamos algunos libros que creemos interesantes para aquellos que quieran avanzar (madurar ideas) en el mundo del desarrollo del Software, a todos los niveles.	2005-05-09	6545	<a href="#">pdf</a>

## Nota:

Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento. Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores. En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo. Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador [rcanales@adictosaltrabajo.com](mailto:rcanales@adictosaltrabajo.com) para su resolución.