Usando XML para metaanotar recursos educativos

Facultad de Informática
Universidad Nacional de La Plata

Director: Lic. Javier Diaz
Alumna: Andrea Deluchi
Legajo: 1658/6

Índice de Contenidos

Introducción

Uno de los grandes problemas aún sin resolver de las nuevas tecnologías de la información y la comunicación aplicadas a la educación es la falta de una metodología común que garantice los objetivos de accesibilidad, interoperabilidad, durabilidad y reutilización de los materiales didácticos basados en Web.

En las actuales soluciones e-learning, generalmente los contenidos preparados para un sistema no pueden ser fácilmente transferidos a otro. Los estándares e-learning son el vehículo a través del cual será posible dotar de flexibilidad a las soluciones e-learning, tanto en contenido como en infraestructura. Ellos han abierto una puerta hacia una manera más coherente de empaquetar los recursos y contenidos, tanto para los estudiantes como para los desarrolladores.

Esta convergencia de tecnologías e-learning es muy importante para los consumidores de estas tecnologías, debido a que los productos que se adhieran a estos estándares no quedarán obsoletos a corto plazo, protegiendo así las inversiones realizadas en este tipo de productos. Además, estándares comunes para asuntos tales como metadata de contenidos, empaquetamiento de contenidos, secuencia de contenidos, interoperabilidad de preguntas y test, perfil de alumnos, interacción en tiempo de ejecución, etc., son requisitos indispensables para el éxito de la economía del conocimiento y para el futuro del e-learning.

Estas problemáticas, así como el estudio de los estándares que intentan resolverlas no han sido parte de lo estudiado durante la carrera. Sin embargo, a través de la materia Herramientas Tecnológicas aplicadas a la Educación, así como también el desarrollo de un Campus Virtual en mi desempeño laboral, han generado en mí un amplio interés por el área del e-learning, lo cual me ha llevado a desarrollar esta tesis.

Durante el desarrollo del Campus Virtual antes mencionado me he encontrado con muchos de los problemas planteados anteriormente, especialmente en lo que concierne a la transferencia de material educativo de una plataforma a otra.

Creo que el experimentar las problemáticas planteadas generan un mayor interés en el estudio de sus soluciones, así como la posibilidad de aplicar lo estudiado, no solo en mis tareas laborales sino también en los ámbitos de la facultad que lo requieran, como la posibilidad de generar material educativo portable para las cátedras de la misma.

El objetivo de esta tesis es analizar en profundidad los principales estándares que existen en al ámbito de e-learning, comparar las ventajas y desventajas de cada uno de ellos, y evaluar su aplicabilidad.

DUBLÍN CORE (1997)

Introducción

Dublín Core es un conjunto de elementos de metadatos encaminado a facilitar la recuperación de recursos electrónicos. El estándar de la base de Dublín abarca quince elementos, la semántica de los cuales ha sido establecida con consenso por un grupo internacional e interdisciplinario de profesionales.

Otra manera de mirar la base de Dublín es como "lengua pequeña para hacer una clase particular de declaraciones sobre recursos". En esta lengua, hay dos clases de términos -- elementos (sustantivos) y calificadores (adjetivos).

DCMES (Dublín Core Metadata Element Set) es el conjunto de elementos de metadatos Dublin Core.

Características Principales

La base de Dublín tiene como metas las características siguientes:

Simplicidad de la creación y del mantenimiento

El sistema de elementos de la base de Dublín se ha mantenido tan pequeño y simple como ha sido posible a fin de permitir que un no especialista cree los elementos descriptivos simples para los recursos de la información de un modo fácil y económico, mientras que prevé la recuperación eficaz de dichos recursos.

Semántica comúnmente entendida

La base de Dublín puede ayudar al 'turista digital' -- un investigador no especialista – a navegar a través de un sistema común de elementos, ya que la semántica de los mismos es entendida universalmente. Por ejemplo, los científicos que desean localizar los artículos según un autor particular, y los eruditos del arte interesados en trabajos de un artista particular, pueden convenir en la importancia de un elemento "creador". Tal convergencia en un campo común, si levemente es más genérica, aumenta la visibilidad y la accesibilidad de todos los recursos, ambos dentro de una disciplina dada y más allá.

Alcance internacional

El sistema de elementos de la base de Dublín fue desarrollado originalmente en inglés, pero las versiones se están creando en muchos otros idiomas, incluyendo finlandés, noruego, tailandés, japonés, francés, portugués, alemán, griego, indonesio, y español. El grupo de interés especial en la base de Dublín en idiomas múltiples está coordinando esfuerzos para ligar estas versiones en un registro distribuido usando la tecnología del marco de la descripción del recurso que es convertida por el consorcio del World Wide Web (W3C).

Aunque los desafíos técnicos de la internacionalización en el World Wide Web no han sido tratados directamente por la comunidad de desarrollo de la base de Dublín, la implicación de representantes de casi cada continente ha asegurado que el desarrollo del estándar considere la naturaleza multilingüe y multicultural del universo de la información electrónica.

Extensibilidad

Mientras que balancean las necesidades de la simplicidad en describir recursos digitales con la necesidad de la recuperación exacta, los reveladores de la base de Dublín han reconocido la importancia de proporcionar un mecanismo para ampliar el sistema de elemento de D.C. para las necesidades adicionales del descubrimiento del recurso. Se espera que otras comunidades de expertos de metadatos creen y administren sistemas adicionales de metadatos. Los elementos de Metadatos de estos sistemas se podrían ligar a metadatos de la base de Dublín para resolver la necesidad de extensibilidad. Este modelo permite que diversas comunidades utilicen los elementos de D.C. para la información descriptiva de la base que será usable a través de Internet, mientras que no prohíbe las adiciones específicas que tienen sentido dentro de una área más limitada.

Alcance

El alcance de Dublin Core se diseñó específicamente para facilitar un vocabulario de metadatos de propiedades centrales o básicas *["core"]* capaz de proporcionar información descriptiva básica sobre cualquier tipo de recurso, independientemente del formato del medio, el área de especialización o el origen cultural. Es importante que un modelo semántico utilizado para la recuperación de recursos no dependa del medio del recurso que pretende describir.

El vocabulario de metadatos Dublin Core es el resultado de muchos años de investigación cooperativa para determinar un conjunto común de propiedades, universal para describir cualquier tipo de recurso. El uso de un sistema de clasificación general normalizado, facilita también metadatos en aquellas colecciones que tienen que mezclarse y para el conocimiento que, albergado dentro de cada colección, se puede compartir.

Dado que la mayor parte de las implementaciones Dublin Core sólo necesitan procesar unos metadatos descriptivos de un recurso, el medio de dicho recurso no se convierte en un problema. Esto hace posible que los metadatos Dublin Core se usen en museos y en otras organizaciones interesadas en la catalogación de tipos de materiales especializados o en colecciones heterogéneas de medios, al mismo tiempo que mantienen una estructura *[framework]* abierta que preserva su capacidad para compartir metadatos con otros implementadores de DC.

Propósito

El objetivo original de Dublin Core fue definir un conjunto de elementos que puedan ser utilizados por autores para describir sus propios recursos en la Web. Enfrentando el hecho de la proliferación de recursos electrónicos y la incapacidad de los profesionales de las bibliotecas de catalogar todos estos recursos, la meta era definir

unos pocos elementos y algunas reglas simples que pudieran ser aplicadas por autores inexpertos en catalogación. Los 13 elementos base fueron luego incrementados a 15. El estándar Dublin Core fue desarrollado para ser simple y conciso. Sin embargo, Dublin Core ha sido utilizado con otros tipos de materiales y en aplicaciones que demandan alguna complejidad. Ha existido históricamente una tensión entre quienes apoyan una vista "minimalista", los que enfatizan en la necesidad de mantener los elementos en un mínimo y la semántica y sintaxis simple, y quienes apoyan una vista "estructural" , que discuten por una fina distinción semántica y más extensibilidad para comunidades particulares.

Conjunto de Elementos

Los elementos poseen nombres descriptivos que pretenden transmitir un significado semántico a los mismos. Para promover una interoperabilidad global, una descripción del valor de algunos elementos podrá ser asociada a vocabularios controlados.

Cada elemento de la base de Dublín es opcional y puede ser repetido. Además, los elementos pueden aparecer en cualquier orden.

Podemos clasificar estos elementos en tres grupos que indican la clase o el ámbito de la información que se guarda en ellos:

1- Elementos relacionados principalmente con el contenido del recurso.
2- Elementos relacionados principalmente con el recurso cuando es visto como una propiedad intelectual.
3- Elementos relacionados principalmente con la instanciación del recurso.

Versión 1.0 (http://purl.org/metadata/dublin_core_elements)

Contenido	***Propiedad Intelectual***	***Instanciación***
Title (título)	Creador (autor o creador)	Date (fecha)
Subject (claves)	Publisher (editor)	Type (tipo del recurso)
Description (descripción)	Contributor (otros colaboradores)	Format (formato)
Source (fuente)	Rights (derechos)	Identifier (identificador del recurso)
Language (lengua)		
Relation (relación)		
Coverage (cobertura)		

Descripción de los elementos

1. **Título:** El nombre dado a un recurso, usualmente por el autor.
2. **Autor o creador:** La persona u organización responsable de la creación del contenido intelectual del recurso. Por ejemplo, los autores en el caso de documentos escritos, artistas, fotógrafos e ilustradores en el caso de recursos visuales.

3. **Claves:** Los tópicos del recurso. Típicamente, Subject expresará las claves o frases que describen el título o el contenido del recurso. Se fomentará el uso de vocabularios controlados y de sistemas de clasificación formales.
4. **Descripción:** Una descripción textual del recurso, tal como un resumen en el caso de un documento o una descripción del contenido en el caso de un documento visual.
5. **Editor:** La entidad responsable de hacer que el recurso se encuentre disponible en la red en su formato actual, por ejemplo la empresa editora, un departamento universitario u otro tipo de organización.
6. **Otros colaboradores:** Una persona u organización que haya tenido una contribución intelectual significativa en la creación del recurso pero cuyas contribuciones son secundarias en comparación a las de las personas u organizaciones especificadas en el elemento Creator (por ejemplo, editor, ilustrador y traductor).
7. **Fecha:** Una fecha en la que el recurso se puso a disposición del usuario en su forma actual. Esta fecha no debe confundirse con la que pertenece al elemento Cobertura, que sería asociada con el recurso sólo en la medida en que el contenido intelectual está de algún modo relacionado con esa fecha.
8. **Tipo del recurso:** La categoría del recurso, por ejemplo página personal, romance, poema, minuta, diccionario. Para asegurar la interoperabilidad.
9. **Formato:** El formato de datos de un recurso, usado para identificar el software y posiblemente, el hardware que se necesitaría para mostrar el recurso.
10. **identificador del recurso:** Secuencia de caracteres usados para identificar unívocamente un recurso. Ejemplo para recursos en línea pueden ser URLs
11. **Fuente:** Secuencia de caracteres utilizado para identificar unívocamente un trabajo a partir del cual proviene el recurso actual.
12. **Lengua:** Lengua/s del contenido intelectual del recurso.
13. **Relación:** Un identificador de un segundo recurso y su relación con el recurso actual. Este elemento permite enlazar los recursos relacionados y las descripciones de los recursos.
14. **Cobertura**: La característica de cobertura espacial y/o temporal del contenido intelectual del recurso. La cobertura espacial se refiere a una región, uso de coordenadas o nombres de lugares extraídos de una lista controlada. La cobertura temporal se refiere al contenido del recurso en vez de a cuando fue creado o puesto accesible ya que este último pertenece al elemento fecha.
15. **Derechos**: Una referencia (URL, por ejemplo) para una nota sobre derechos de autor, para un servicio de gestión de derechos o para un servicio que dará información sobre términos y condiciones de acceso a un recurso.

Versión 1.1 (1999) (http://dublincore.org/documents/dces/)

La versión 1.1 del metadato de Dublín Core reemplaza a la versión 1.0 presentada anteriormente.

Las definiciones utilizan un estándar formal para la descripción de los elementos del metadato. Esta formalización ayuda a mejorar la consistencia con otras comunidades

de metadatos y realza la claridad, el alcance y la consistencia interna de las definiciones del elemento del metadato de la base de Dublín.

Se define cada elemento de la base de Dublín usando un sistema de diez cualidades del estándar de ISO/IEC 11179 para la descripción de los elementos de datos. Estos incluyen:

- ***Nombre***: etiqueta asignada al elemento de datos
- ***Identificador***: identificador único asignado al elemento de datos
- ***Versión***: versión del elemento de datos
- ***Autoridad del Registro***: entidad autorizada para colocar el elemento de datos
- ***Lengua***: lengua en la cual se especifica el elemento de datos
- ***Definición***: declaración que representa claramente el concepto y la naturaleza esencial del elemento de datos
- ***Obligación***: indica si el elemento de datos es siempre requerido o a veces (es decir, que contenga un valor)
- ***Tipo de dato***: indica el tipo de dato para el valor del elemento de datos
- ***Ocurrencia Máxima***: indica el límite de repetición del elemento de datos
- ***Comentario***: observación referente al uso del elemento de datos.

Afortunadamente, seis de las diez cualidades antedichas son comunes a todos los elementos de la base de Dublín. Éstos están, con sus valores respectivos:

Versión: 1.1

Autoridad del Registro: DCMI

Lengua: en

Obligación: opcional

Tipo de Dato: Char String

Ocurrencia Máxima: ilimitado

Las definiciones proporcionadas incluyen la forma conceptual y de representación de los elementos de la base de Dublín. La definición captura el concepto semántico y el tipo de dato y el comentario capturan la representación del dato.

Cada definición de la base de Dublín se refiere al recurso que describe. Un recurso se define como “cualquier objeto que tenga identidad”. Para los propósitos del metadato de la base de Dublín, un recurso será típicamente un recurso de información o de servicio, pero se puede aplicar más ampliamente.

Clases de Calificadores

Cada elemento también tiene un sistema limitado de calificadores, que son las cualidades que se pueden utilizar para refinar (no extender) el significado del elemento. La iniciativa de Metadato de la base de Dublín (DCMI) ha definido maneras estándares para "calificar" elementos con varios tipos de calificadores.

En julio de 2000, la iniciativa de Metadato de la Base de Dublín publicó su lista de Calificadores recomendados de la base de Dublín. A la hora de la ratificación de estos calificadores, el DCMI reconoció dos amplias clases de calificadores:

Refinamiento de elemento: estos calificadores hacen que el significado de un elemento sea más estrecho o más específico. Un elemento refinado comparte el significado del elemento más amplio, pero con un alcance más restricto. Un cliente que no entiende un término específico del refinamiento del elemento debe poder no hacer caso del calificador y tratar el valor del metadato como si fuera un elemento más amplio.

Esquema de codificación: estos calificadores identifican los esquemas que ayudan en la interpretación de un valor del elemento. Estos esquemas incluyen vocabularios controlados y notaciones formales. Un valor expresado usando un esquema de codificación será así un símbolo seleccionado de un vocabulario controlado (por ejemplo, un término de un sistema de clasificación) o de una secuencia ajustada a un formato de acuerdo con una notación formal (por ejemplo, “2000-01-01” como la expresión estándar de una fecha).

Conclusión

Podemos concluir que la base de Dublín Core presenta como mayor ventaja la gran simplicidad en la creación y mantenimiento de metadatos para cualquier recurso.

Por otro lado, debe garantizar la posibilidad de que diversas comunidades hagan adiciones específicas que tengan sentido dentro de un área más limitada. Esto es lo que le dará durabilidad en el tiempo a la base de Dublín Core, dado que cada vez hay más comunidades que requieren de metadatos específicos.

IEEE LTSC LOM (Learning Object Metadata) (1997)

Introducción

LOM es un estándar que especifica la sintaxis y la semántica de la "metainformación de objetos educacionales".

Los estándares del LOM se centran en el conjunto mínimo de propiedades que permiten que los objetos educacionales sean gestionados, ubicados y evaluados.

Alcance

Esta norma especifica un esquema de datos conceptual que define la estructura de un caso de metadato para un objeto de aprendizaje. Para esta norma, un objeto de aprendizaje se define como cualquier entidad, digital o no-digital, que puede usarse para aprender, educar o entrenar.

Para esta norma, un caso del metadato para un objeto de aprendizaje describe características pertinentes del objeto de aprendizaje al que se aplica. Pueden reagruparse tales características en general, educativo, técnico y categorías de la clasificación.

El esquema de datos conceptual especificado en esta norma permitirá diversidad lingüística de objetos de aprendizaje y los metadatos que los describen.

El esquema de datos conceptual definido en esta norma especifica los elementos de los datos de un metadato para un objeto de aprendizaje.

Esta norma será referenciada por otras normas que definirán las descripciones de aplicación del esquema de los datos para que un caso de metadato para un objeto de aprendizaje pueda usarse por un sistema de tecnología de aprendizaje para manejar, localizar, evaluar o intercambiar objetos de aprendizaje.

Esta norma no define cómo un sistema de tecnología de aprendizaje representará o usará un caso de metadato para un objeto de aprendizaje.

Propósito

El propósito de esta norma es facilitar la búsqueda, evaluación, adquisición, y uso de objetos de aprendizaje, por ejemplo por aprendices o instructores. El propósito también es proveer la posibilidad de compartir e intercambiar objetos de aprendizaje, habilitando el desarrollo de catálogos e inventarios teniendo en cuenta la diversidad de contextos culturales y linguales en los que el aprendizaje crea sus objetos y explota los metadatos incluidos.

Especificando un esquema de datos conceptual común, esta norma asegura que los objetos metaanotados con LOM tendrán un alto grado de interoperabilidad semántico.

Definiciones

Categoría (LTSC LOM): Un grupo de elementos de datos relacionados.

Elementos de datos del LOM: Un elemento de datos cuyo nombre, explicación, tamaño, orden, espacio de valores y tipo de datos están definidos en este estándar.

Tipo de Datos: Una variedad de valores indicada por sus características comunes y las operaciones sobre ellos.

Elemento de datos extendido: Un elemento de una estructura de datos que está definido fuera de un estándar y permitido dentro de una instancia de una estructura de datos

Langstring: Un tipo de datos que representa una o más ristras de caracteres. Un valor langstring puede incluir varias cadenas de caracteres equivalentes semánticamente, tales como traducciones o descripciones alternativas. Ver también tipo de datos

Objeto educativo: Para este estándar un objeto educativo se define como cualquier entidad digital o no que pueda ser usada para el aprendizaje, la enseñanza y el entrenamiento.

Máximo más pequeño permitido: Para aquellos valores definidos para la implementación, indica el valor más pequeño permitido cuando se define un máximo para un rango de valores.

Espacio de valores: El conjunto de los valores posibles de un determinado tipo de datos. [ISO/IEC 11404:1996].

NOTA:-- En LOM, un espacio de valores es generalmente enumerado al margen, o definido con referencia a otro estándar u otra especificación.

Estructura Básica de los Metadatos

Los elementos de datos describen un objeto educativo y están agrupados en categorías. El esquema base LOM 1.0 consiste en 6 de éstas categorías.

- *La categoría General* (1) agrupa la información general que describe un objeto educativo de manera global.
- *La categoría Ciclo de Vida* (2) agrupa las características relacionadas con la historia y el estado actual del objeto educativo, y aquellas que han afectado a este objeto educativo durante su evolución
- *La categoría Meta-Metadatos* (3) agrupa la información sobre este registro de Metadatos, (diferente del objeto educativo que describe este registro)
- *La categoría Técnica* (4) agrupa los requerimientos técnicos y las características del objeto educativo
- *La categoría Uso Educativo* (5) agrupa las características educativas y pedagógicas del objeto educativo
- *La categoría de Derechos* (6) agrupa los derechos de propiedad intelectual y las condiciones para el uso del objeto educativo.
- *La categoría Relación* (7) agrupa características que definen la relación entre este objeto educativo y otros objetos educativos referenciados

- *La categoría Anotación* (8) proporciona comentarios sobre el uso del objeto educativo e información sobre cuándo y por quién fueron creados dichos comentarios
- *La categoría de Clasificación* (9) describe dónde puede ubicarse este objeto educativo en un determinado sistema de clasificación.

De forma colectiva, estas categorías forman el Esquema Base. La última categoría: Clasificación, permite a un usuario final clasificar un objeto educativo de acuerdo con una estructura de clasificación arbitraria. Como puede referenciarse cualquier clasificación, esta categoría se proporciona como un simple mecanismo de extensión.

Conjunto de Elementos

Los elementos de datos de LOM son:

No	Nombre
1	General
1.1	Identificador
1.1.1	Catálogo
1.1.2	Entrada
1.2	Título
1.3	Idioma
1.4	Descripción
1.5	Palabras Clave
1.6	Ámbito
1.7	Estructura
1.8	Nivel de Agregación
2	Ciclo de Vida
2.1	Versión
2.2	Estado
2.3	Contribución
2.3.1	Tipo
2.3.2	Entidad
2.3.3	Fecha
3	Meta-Metadatos
3.1	Identificador
3.1.1	Catálogo
3.1.2	Entrada
3.2	Contribución
3.2.1	Tipo
3.2.2	Entidad
3.2.3	Fecha
3.3	Esquema de Metadatos
3.4	Idioma

4	Técnica
4.1	Formato
4.2	Tamaño
4.3	Localización
4.4	Requisitos
4.4.1	AgregadorOR
4.4.1.1	Tipo
4.4.1.2	Nombre
4.4.1.3	Versión Mínima
4.4.1.4	Versión Máxima
4.5	Pautas de Instalación
4.6	Otros Requisitos de Plataforma
4.7	Duración
5	Uso Educativo
5.1	Tipo de Interactividad
5.2	Tipo de Recurso Educativo
5.3	Nivel de Interactividad
5.4	Densidad Semántica
5.5	Destinatario
5.6	Contexto
5.7	Rango Típico de Edades
5.8	Dificultad
5.9	Tiempo Típico de Aprendizaje
5.10	Descripción
5.11	Idioma
6	Derechos
6.1	Coste
6.2	Derechos de Autor y otras Restricciones
6.3	Descripción
7	Relación
7.1	Tipo
7.2	Recurso
7.2.1	Identificador
7.2.1.1	Catálogo
7.2.1.2	Entrada
7.2.2	Descripción
8	Anotación
8.1	Entidad
8.2	Fecha
8.3	Descripción
9	Clasificación
9.1	Propósito
9.2	Ruta Taxonómica

9.2.1	Fuente
9.2.2	Taxon
9.2.2.1	Id
9.2.2.2	Entrada
9.3	Descripción
9.4	Palabras clave

Las categorías agrupan elementos de datos. Por cada elemento de datos el esquema define:

- *Nombre*: el nombre de referencia del elemento de datos
- *Explicación*: la definición del elemento de datos;
- *Tamaño*: el número de valores permitido;
- *Orden*: relevancia de la ordenación de los valores (solo aplicable en el caso de elementos de datos de valores múltiples)
- *Ejemplo*: un ejemplo ilustrativo.

Para elementos de datos simples, el LOM v1.0 también define:

- *Espacio de valores*: el conjunto de valores permitido para el elemento de datos. Típicamente en forma de un vocabulario o una referencia a otro estándar
- *Tipo de datos*: un conjunto de valores distintivos

Tanto el tamaño como la información del tipo de datos pueden incluir un dato que defina tamaños mínimos del máximo valor permitido.

Extensiones al esquema base del LOMv1.0 deberán conservar el espacio de valores y el tipo de datos de los elementos del esquema base LOMv1.0. Las extensiones no definirán tipos de datos o espacios de valores para agregar elementos de datos en el esquema base LOMv1.0

El esquema de numeración de los elementos de datos representa una jerarquía de agregación de los elementos de datos y sus componentes. Como ejemplo, el elemento de datos agregado 7:2:Relación:Recurso tiene dos componentes 7:2:1:Relación:Recurso.Identificador y 7.2.2:Relación.Recurso.Descripción. El primero de ellos a su vez, es un elemento agregado, y contiene los componentes: 7.2.1.2:Relación.Recurso.Identificador.Catálogo y 7.2.1.2:Relación.Recurso.Identificador.Entrada.

Todos los elementos de datos son opcionales: esto significa que una instancia conforme al LOM puede incluir valores para cualquier elemento de datos definido.

Como el Esquema Base del LOMv1.0 impone una relación de agregación, los componentes podrán -por definición- estar presentes únicamente en una instancia del LOM como componente del elemento agregado al cual pertenece. Como ejemplo, 7.2.1:Relación.Recurso.Identificador aparece por definición como un componente de 7.2:Relación.Recurso. En este sentido, la presencia del componente implica automáticamente la presencia del elemento agregado al que pertenece.

Lista de Valores

En algunas instancias, un elemento de datos contiene una lista de valores, más que un valor único. Esta lista debe ser de uno de los siguientes tipos:

- *Ordenada*: el orden de los valores en la lista es relevante. Por ejemplo, en una lista de autores de una publicación, el primer autor es comúnmente considerado el autor más importante. Como otro ejemplo, en una estructura de clasificación, el orden es de más general a más especifico.
- *No ordenada*: el orden de los valores no tiene relevancia ni significado. Por ejemplo, si la descripción de una simulación incluye tres textos cortos que describen el uso educativo previsto en tres idiomas diferentes, entonces el orden de estos textos no es significativo. Pueden aparecer en cualquier orden sin pérdida de información.

Si un elemento de datos con sub-elementos contiene una lista de valores, entonces cada uno de estos valores debe ser una tupla de sub-elementos. Por ejemplo, en el Esquema Base LOMv1.0 se especifica que el elemento de datos 1.1:General.Identificador contiene una lista no ordenada de valores. Esto significa que el valor del elemento de datos 1.1:General.Identificador es una lista no ordenada de tuplas (1.1.1:General.Identificador.Catálogo, 1.1.2:General.Identificador.Entrada). En este caso, cada valor 1.1:General.Identificador.Catálogo determina el catálogo de donde viene el valor 1.3.2:General.Catálogo.Entrada.

Vocabularios

Hay algunos elementos de datos definidos como vocabularios. Un vocabulario es una lista recomendada de valores apropiados. Se pueden usar también otros valores no incluidos en la lista. Sin embargo, los metadatos que se ajustan a los valores recomendados tendrán el máximo grado de interoperabilidad semántica, esto significa que la probabilidad de que esos metadatos sean entendidos por otros usuarios será la máxima posible.

El valor del elemento de datos con vocabulario asociado se representará como un par (fuente, valor). Si la fuente es LOMv1.0 entonces el espacio de valores se describe en esta edición del estándar.

NOTA: Si la fuente no es LOMv1.0 entonces usuarios e implementadores deben en lo posible crear vocabularios que no entren en conflicto con este estándar.

Ejemplo: como ilustración damos ejemplos de diferentes casos para los elementos de datos 5.2:UsoEducativo.Tipo de Recurso Educativo:

- Si el valor está en el vocabulario, por ejemplo “cuestionario”, entonces se representaría como ("LOMv1.0", “cuestionario”). Esta opción es recomendable siempre que los valores del vocabulario pueden expresar adecuadamente el significado pretendido.

- Si el usuario quiere asignar un valor que no es parte de la lista dada para 5.2:UsoEducativo.Tipo de Recurso Educativo, entonces el usuario puede designar el valor como, por ejemplo, ("http://www.vocabularies.org/LearningResourceType", "MotivatingExample"). Esta opción proporciona mayor flexibilidad al indexador de un objeto educativo a expensas de la interoperabilidad semántica. Los valores definidos por el usuario no serán usados de forma consistente en otras comunidades más extensas. En el ejemplo anterior, se usó una URI para indicar la fuente del vocabulario. Este enfoque es ciertamente una buena práctica pero el uso de una URI no es un requerimiento.

El significado asociado con un valor de tipo vocabulario es definido por el término correspondiente en el Oxford English Dictionary, 2nd Ed 1989 salvo mención contraria explícitamente hecha en el Esquema Base LOM 1.0.

Máximo más pequeño permitido

En el esquema base del LOMv1.0 se define un mínimo para los valores máximos permitidos en los siguientes elementos:

- *elementos de datos de tipo lista*: Todas las aplicaciones deberán soportar al menos dicho número de elementos en la lista. En otras palabras, una aplicación puede imponer un máximo en el número de entradas que permite para el valor lista de dicho elemento de datos, pero dicho máximo no puede ser inferior que el valor asignado al máximo más pequeño permitido.
- *elementos de datos de tipo CharacterString* o *LangString*: Todas las aplicaciones que procesen instancias del LOM deberán permitir procesar al menos dicha longitud en el valor del CharacterString (bien directamente o en el contenido de LangString) del elemento de datos. En otras palabras, una aplicación puede imponer un máximo en el número de caracteres que permite para el valor CharacterString del elemento de datos, pero dicho valor no podrá ser inferior al menor máximo permitido para ese tipo de datos en el elemento de datos

NOTA 1:--: Se pretende que el valor del máximo más pequeño permitido cubra la mayoría de los casos

NOTA 2:-- El significado de procesar en la definición anterior dependerá de la naturaleza de la aplicación

Juegos de caracteres

Este estándar define una estructura conceptual para los metadatos de los objetos educativos. No trata aspectos de representación, que están especificados en otras partes del Learning Objects Metadata Standard. El esquema base LOMv1.0 específica estándares externos a los que se ajustarán los campos del tipo CharacterString (En el caso de valores de tipo cadena de caracteres no restringidos, se ajustarán a la referencia

ISO/IEC10646-1:2000). En cualquier caso, las decisiones que se tomen en relación con los formatos de representación deberán tener en cuenta el soporte multilingüe.

Representación

Para cada uno de los elementos de datos, la especificación incluye el tipo de datos del que derivan sus valores, tales como LangString, Date, etc.

Este estándar no define tokens para los nombres de los elementos o los valores de los vocabularios. Se espera que tales tokens sean definidos por las implementaciones de este estándar.

Dentro del esquema base LOMv1.0, el orden de las categorías y de las subcategorías es a título informativo. Una instancia del Esquema Base LOMv1.0 deberá preservar el anidamiento de las categorías, pero la instancia no precisa ordenar las categorías o los subitems dentro de una categoría o subcategoría. Por ejemplo la categoría 5: Educativo puede aparecer antes de la categoría 1: General y dentro de la categoría general el Item 1.3:General.Lenguaje puede aparecer antes que 1.2:General.Título.

Conformidad

- Una instancia con conformidad estricta a la especificación de metadatos de LOM deberá consistir únicamente de elementos de datos de LOM.
- Una instancia conforme al LOM puede contener extensiones a los elementos de datos.
- Una instancia que no contenga valores asignados a ninguno de los elementos de datos del LOM es una instancia conforme al estándar

Con objeto de maximizar la interoperabilidad semántica, los elementos de datos extendidos no deberán reemplazar elementos de datos en la estructura del LOM. Esto significa que una organización no debería introducir elementos de datos adicionales que reemplacen elementos de datos del LOM. Como ejemplo, una organización no debería introducir un elemento nuevo de datos “nombre” para remplazar 1.2:General.Título.

NOTA: Para garantizar la interoperabilidad semántica, los usuarios de este estándar deben procurar asignar la meta información acorde con el elemento de datos correspondiente en el estándar. Por ejemplo un usuario no debería asignar un elemento describiendo las fuentes usadas en el documento para el elemento 1.2:General.Título

Conclusión

LOM representa una buena solución para la descripción de recursos educativos en general. Esto hace que diferentes comunidades educativas puedan intercambiar material independientemente de las características específicas de su comunidad.

Como aspecto positivo de la especificación, se resalta la posibilidad que ofrece de realizar una descripción muy detallada de cualquier recurso. Gracias a la cantidad de campos que brinda, obviamente todos opcionales por tratarse de información adicional, se puede añadir una gran cantidad de datos que pueden facilitar las búsquedas, las

clasificaciones y la comprensión de todos los elementos implicados en el proceso de enseñanza.

A pesar de las buenas intenciones de la especificación, la flexibilidad que propone conlleva una falta de rigor en todos sus campos que ha acarreado los siguientes problemas:

- Hay campos que no son adaptables a otros tipos de educación, y sobre todo, resulta necesario contar con más campos orientados a la información pedagógica. Por ejemplo, los campos que muestra el elemento <educational>, teóricamente encargado de estos aspectos, resultan muy vagos. Como ejemplo nos podemos fijar en el elemento <semanticdensity>, hijo de <educational> y referido a la complejidad semántica del recurso educacional al que se refiere. Este elemento admite campos del tipo –low, high, very high- que resultan, subjetivos, y por tanto difíciles de usar una vez intercambiado el objeto.
- Esta falta de rigidez en los valores que pueden tomar los elementos y atributos dificulta enormemente el procesamiento automático de los elementos. Por supuesto que un aumento de rigidez en los valores posibles iría en detrimento de la facilidad de intercambio de los metadatos. Aún así, creo que muchos de los campos resultan comunes a todos los sistemas de enseñanza, por lo que se podría realizar un trabajo futuro en el que se ofreciera un esqueleto común para todos y se realizase una personalización posterior de la especificación a los principales armazones educativos.

En relación a esto IMS considera, como veremos más adelante, que el número de elementos definidos en LOM es demasiado grande

Muchas organizaciones dentro de la comunidad IMS recomendaron que se identificase un conjunto más pequeño de elementos LOM para simplificar los esfuerzos iniciales de implementación. IMS se ha apoyado para reducir LOM en la especificación Dublín Core, que como dijimos antes una de sus mayores ventajas es la simplicidad en la creación y mantenimiento de los metadatos, dada, entre otras cosas por el número reducido de elementos.

Correspondencia completa con Dublín Core

Como ya hemos visto el Dublín Core define 15 elementos de datos. Estos elementos de datos se corresponden con elementos de datos definidos en este estándar de la forma en que se muestra en la tabla adjunta.

Tabla 1. Correspondencia con Dublin Core

DC.Identifier	1.1:General.Identifier actualmente es un término reservado, pues no hay método especificado para la creación de un identificador global único.
DC.Title (título)	1.2:General.Title (título)
DC.Language (lengua)	1.3:General.Language (lengua)
DC.Description (descripción)	1.4:General.Description (descripción)
DC.Subject (claves)	1.5:General.Keywords (palabra clave) o 9:Classification (clasificación) con 9.1:Classification.Purpose (propósito) igual a "Discipline" o "Idea".
DC.Coverage (estructura)	1.7:General.Coverage (estructura)
DC.Type (tipo del recurso)	5.2:Educational.LearningResourceType (tipo de recurso educativo)
DC.Date (fecha)	2.3.3:LifeCycle.Contribute.Date (fecha) cuando 2.3.1:LifeCycle.Contribute.Role (tipo) tiene el valor "Publisher".
DC.Creator (autor)	2.3.2:LifeCycle.Contribute.Entity (entidad) cuando 2.3.1:LifeCycle.Contribute.Role (tipo) tiene el valor "Author".
DC.OtherContributor (otros colaboradores)	2.3.2:LifeCycle.Contribute.Entity (entidad) con el tipo de contribución en 2.3.1:LifeCycle.Contribute.Role (tipo).
DC.Publisher (editor)	2.3.2:LifeCycle.Contribute.Entity (entidad) cuando 2.3.1:LifeCycle.Contribute.Role (tipo) tiene el valor "Publisher".
DC.Format (formato)	4.1:Technical.Format (formato)
DC.Rights (derechos)	6:Rights (derechos)
DC.Relation (relación)	7:Relation (relación)
DC.Source (fuente)	7.2:Relation.Resource (recurso) cuando el valor de 7.1:Relation.Kind (tipo) es "IsBasedOn".

Colaboración entre DCMI y LOM

DCMI y LOM acuerdan colaborar en el desarrollo de especificaciones de metadatos para la web.

- Acuerdan evitar el solape semántico entre ambas propuestas de estándares (un mismo concepto o definición no debe estar presente en dos grupos de elementos diferentes).

- Independencia de cualquier tecnología o sintaxis para expresar metadatos. Sin embargo, prevé usar diversas tecnologías:
    - HTML, dado el segmento significativo que ocupa en la web.
    - XML, estándar de la industria para la codificación de datos y documentos, y
    - RDF, que proporciona un enfoque útil para satisfacer los requisitos de extensibilidad y modularidad, y puede así conducir a la interoperatividad entre los requisitos de cada comunidad de metadatos (DCMI, 2000:1).

"Los accesos disponibles a los depósitos globales de metadatos, se aprecian cada vez en mayor medida como un factor crítico en el punto de inflexión de la próxima generación de aprendizaje y creación de conocimiento. Sin embargo, esto requeriría que los metadatos sean muy interoperables y universalmente reutilizables. Este acuerdo marca un paso importante hacia la realización de esta visión". [Wayne Hodgins, Presidente del Grupo de Trabajo IEEE LTSC LOM, en DCMI, 1998:2]

"LOM y DCMI están construyendo el camino que permitirá que la recuperación e intercambio de información sea un proceso mucho más gratificante". [Stuart Weibel, Director de DCMI, en DCMI, 1998:1]

EML (Educational Modelling Language)

Introducción

EML es la abreviatura de Educational Modelling Language y es un método notacional para la educación en entornos de aprendizaje electrónicos.

Fue creado en la Open University de Holanda y actualmente su máxima implementación es el sistema Edubox que es la primera versión completamente operacional de EML. Actualmente Edubox está siendo usado en diversos sitios como: el "Maastricht School Of Hotel Management" o el instituto de educación a distancia de Sudáfrica, así como en la propia Open University que lo creó.

El Educational Modelling Language tiene pretensiones de formar un estándar para entornos de aprendizaje y unidades de aprendizaje. EML permite intercambiar unidades de aprendizaje entre distintos entornos de aprendizaje.

Unidades de estudio

Hemos dicho que EML es un sistema notacional. Pero para poder comprender correctamente lo que es un sistema notacional antes tenemos que ver las unidades de estudio.
Unidad de estudio es un término mucho más genérico que entorno de aprendizaje. Una unidad de estudio puede ser capaz de describir un entorno de aprendizaje y todos los subsistemas que puede haber en él, como cursos, etc. Así mismo, una unidad de estudio puede contener más unidades de estudio.

En la figura 1.1 podemos ver como una unidad de estudio puede describir un entorno de aprendizaje.

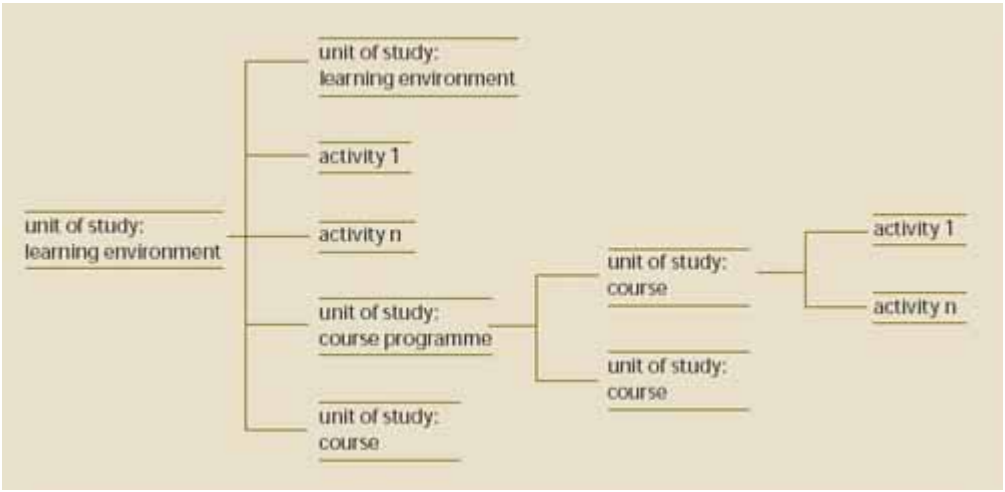


Figura 1.1

Las unidades de estudio serán la estructura principal a la hora de dar forma al Entorno de aprendizaje de EML.

Métodos notacionales

¿Que es un método notacional?

Para describir un método notacional es mucho más fácil decir cuáles son las implementaciones más conocidas. El lenguaje XML está siendo cada vez más usado como estándar de marcado para crear estandarizaciones entre sistemas heterogéneos. XML es la implementación elegida por los creadores de EML para definir su estándar. Es importante destacar que los métodos notacionales son muy potentes a la vez que simples ya que dividen y organizan la información jerárquicamente a través de etiquetas definidas por el propio usuario.

¿Qué requisitos tiene que cumplir?

El titulo de está sección es la pregunta que se realizan los autores de EML sobre los requisitos que debe de cumplir un método notacional. Para ello describen 11 requisitos que son los siguientes:

1. Los sistemas notacionales deben describir formalmente las unidades de estudio, por lo tanto el procesamiento automático es posible.
2. Los sistemas notacionales deben ser capaces de describir unidades de estudio que estén basadas en diferentes filosofías educacionales.
3. Los sistemas notacionales deben registrar explícitamente la estructura de los componentes de instrucción.
4. Los sistemas notacionales deben ser capaces de describir totalmente una unidad de estudio, incluyendo todos los contenidos y actividades de los estudiantes y miembros.
5. Los sistemas notacionales deben describir las unidades de estudio de modo que la ejecución repetida sea posible. La consecuencia de esto es que no se pueden hacer referencias a instancias concretas tales como información sobre el tiempo, personas o lugar. Todo esto es descrito de modo abstracto. La cooperación entre la gente y la interacción entre las personas y los objetos debe también ser posible describirse de modo abstracto.
6. Los sistemas notacionales deben poder describir aspectos personales dentro de la unidad de estudio, para que los contenidos y actividades dentro de la unidad de estudio puedan ser adaptados según las preferencias, anteriores conocimientos, y situaciones circunstanciales de los usuarios.
7. La notación de los componentes de contenido, donde sea posible, debe ser medianamente neutral, para que los usuarios puedan elegir el medio de presentación (de un modo neutral).
8. Donde sea posible, una "pared" debería ser colocada entre los estándar que son usados para metaanotar las unidades de estudio y la técnica usada para "ejecutar" la unidad de estudio. Gracias a esto, las inversiones en desarrollo educacional llegarán resistentes a los cambios tecnológicos y los problemas de conversión (interoperabilidad).
9. El sistema notacional debe ser adecuado con los estándares disponibles.
10. El sistema notacional debe hacer posible la identificación, aislamiento, descontextualización e intercambio de componentes de utilidad, y reutilizarlos en otros contextos.
11. El sistema notacional debe permitir producir, cambiar, preservar, distribuir y archivar unidades de estudio.

Estructura de EML

En la figura 1.2 podemos ver la subdivisión de las partes en que se compone el entorno de aprendizaje de EML, siendo su parte principal la unidad de aprendizaje que es la que describe todo el entorno.

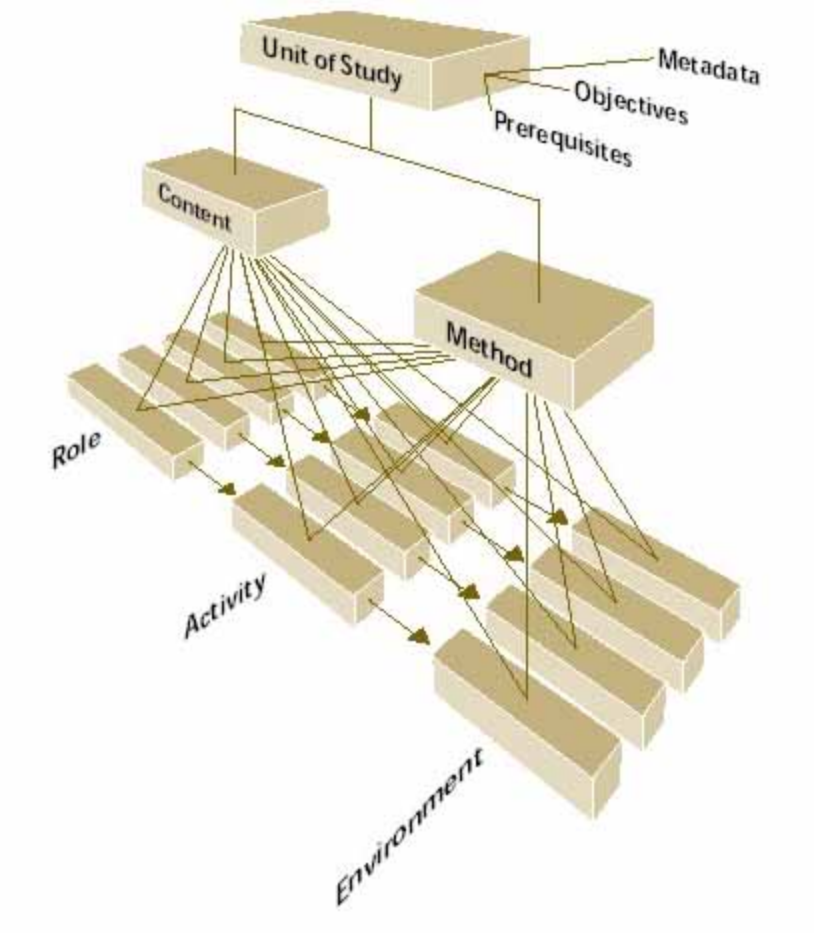


Figura 1.2

Unidad de estudio

En la figura 2.2 podemos ver la estructura de una unidad de estudio según la estructura que determina Edubox 1.0

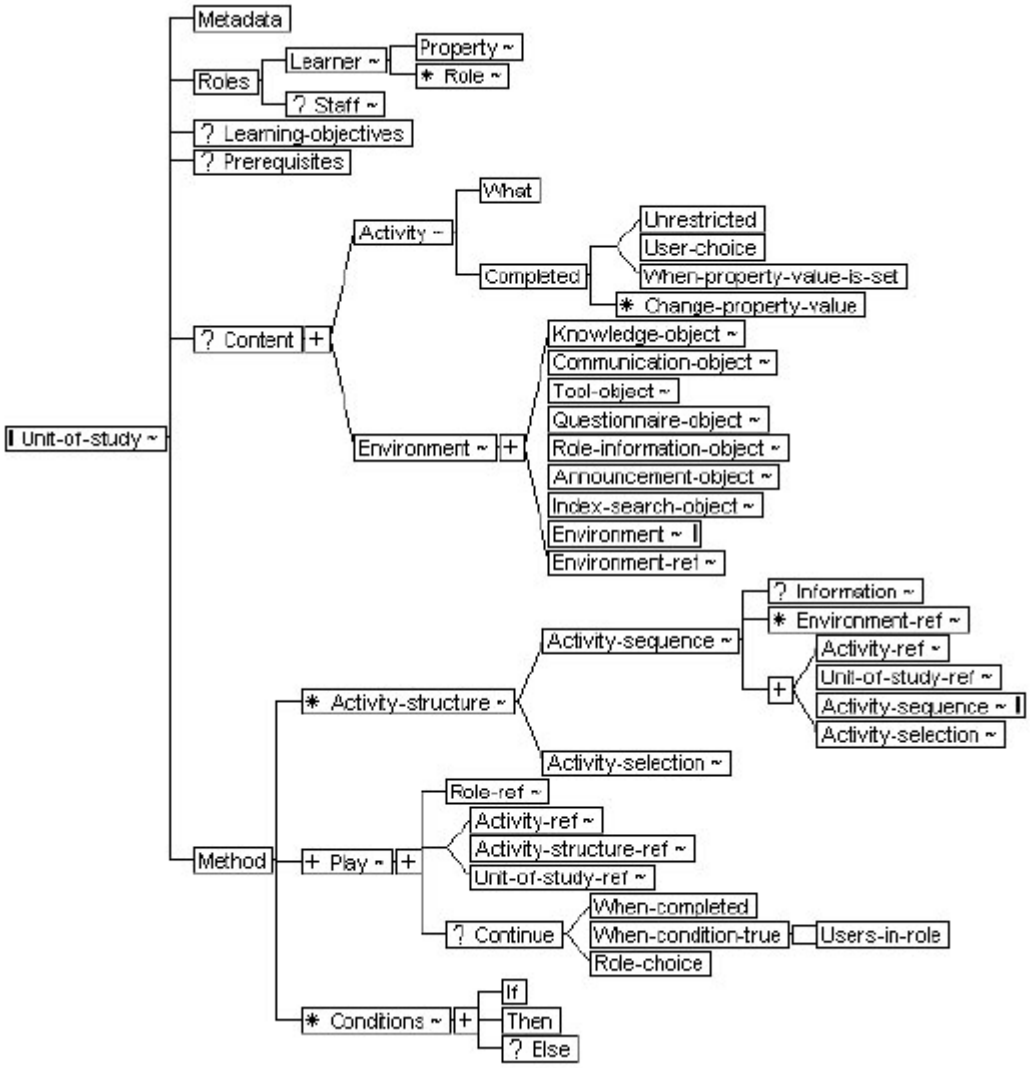


Figura 2.2

Partes de la unidad de estudio

En la figura 2.2 podemos ver las partes principales de las que se que compone una unidad de estudio.

- *Metadatos*. Aquí aparecen los datos que describen el documento.
- *Roles*. Aquí se definirán los distintos roles de las personas que van a interactuar con la unidad de estudio
- *Objetivos de aprendizaje*. En esta sección describimos los objetivos que se tienen que cumplir al completar el aprendizaje de la unidad de estudio.
- *Prerrequisitos*. Aquí se realizará una descripción de los prerrequisitos (conocimientos previos o necesidad de haber cursado algún estudio) para poder realizar la unidad de estudio.
- *Contenidos*. Mediante un complejo de actividades y entornos podemos describir los contenidos del curso. Aquí estarán los Knowledge-object (en concreto dentro de los Enviroment que son los entornos) que serán los documentos finales en los que estén los contenidos. También aquí se definirán los exámenes o cuestionarios (Questionnaire-object)
- *Método*. En esta etiqueta podemos definir la estructura del curso (Activity-structure), las actividades que va a realizar cada rol definido

(Play) y una sección de posibles condiciones que se deben cumplir (Conditions).

Knowledge-object (contenidos)

Como hemos visto, dentro de Content definimos los contenidos del curso. Si ahondamos en esta estructura podremos encontrar que en los entornos (Enviroment) podemos definir los Knowlege-object que son los documentos finales en los que ponemos los contenidos. En la figura 2.3 podemos ver como es un Knowledge-object:

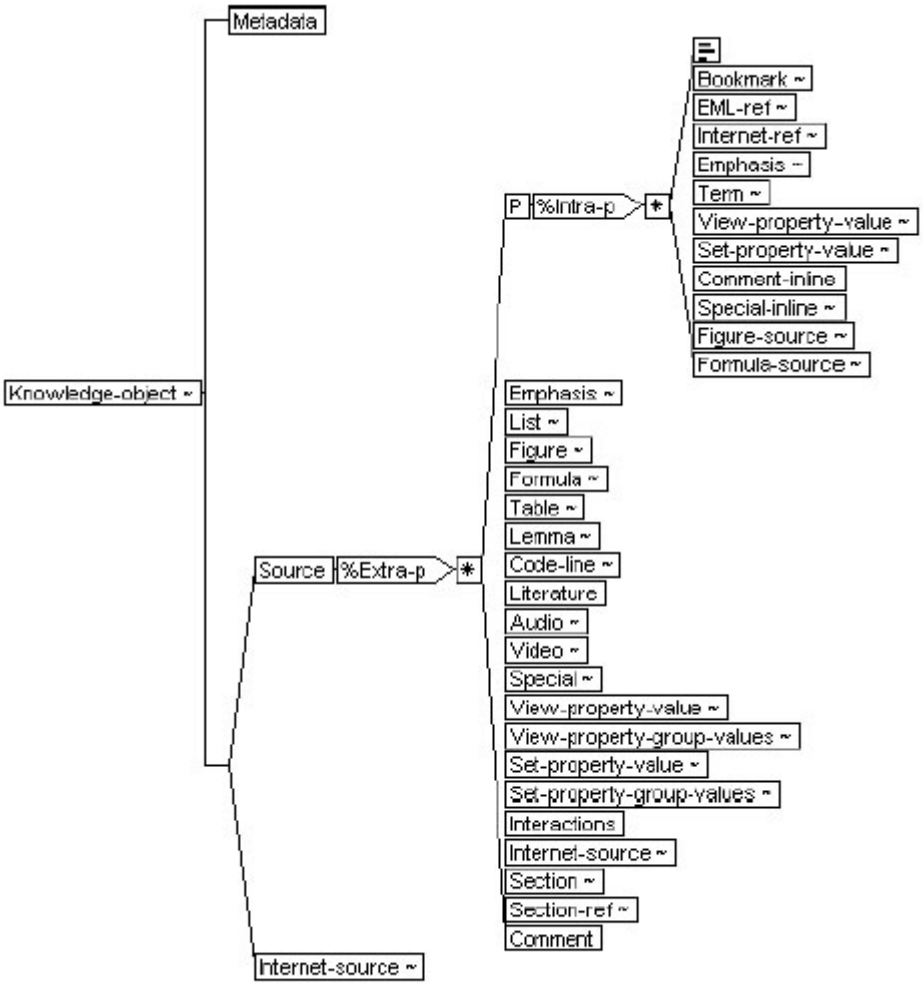


Figura 2.3

Partes del Knowledge-object

En la figura 2.3 se describe la estructura general de un knowledge-object. Podemos ver que hay una gran variedad de posibilidades para confeccionar nuestros documentos. No tenemos por qué usar todo lo que se ha descrito en la figura 2.3, lo importante es que todas las componentes están a nuestra disposición para ser utilizadas. Veamos las partes de las que se compone el Knowledge-object

- **Metadata**. Como ya hemos dicho en este curso, casi cualquier parte de la estructura EML es susceptible de tener metadatos.
- **La fuente de datos**. Puede ser una Source (definimos los datos en el mismo document) o una Internet-source (se hace referencia a una dirección en la que tendrán que estar los datos descritos en EML). Lo normal es usar un Source. Podemos observar que dentro del Source hay infinidad de maneras de realizar la organización aunque esto quizás no se pueda observar bien en el gráfico.

Meta-Modelo Pedagógico

Esto es un intento que va más allá de lo que es la pura informática para adaptar el modelo EML a la enseñanza desde un punto de vista práctico. Veremos los subsistemas que componen este modelo y el gran esfuerzo que la Open University de Holanda ha invertido para que EML sea aplicable a la realidad y modele un gran número de comportamientos que otros estándares no tienen en cuenta.

Introducción

¿Que es el meta-modelo pedagógico?

El nombre de meta-modelo pedagógico puede resultar un poco extraño, es nuestra traducción particular de lo que los creadores de EML llaman pedagogical meta-model. El profesor Rob Koper (universidad abierta de los Países Bajos (OUNL)) expresa en http://eml.ou.nl/introduction/docs/ped-metamodel.pdf que desde su punto de vista el meta-modelo pedagógico es un modelo que modela los distintos modelos pedagógicos. Es decir, que los distintos modelos pedagógicos que queramos crear pueden ser derivados de este meta-modelo. Como todos los modelos, éste es una abstracción de la realidad. No debe confundirse con la realidad propiamente dicha y no es el único modelo posible de descripción del aprendizaje de la instrucción. Los autores de EML también remarcan que el diseño de los cursos puede ser diferente en el momento en el que se crean de cuando realmente son instanciados o usados en la realidad. De este modo el diseño de los cursos no intenta abstraer todos los detalles del curso pero sí la mayoría de los puntos.

Subsistemas del meta-modelo

En la figura 3.1 podemos ver los diferentes paquetes en que se divide el modelo. La figura es un diagrama UML

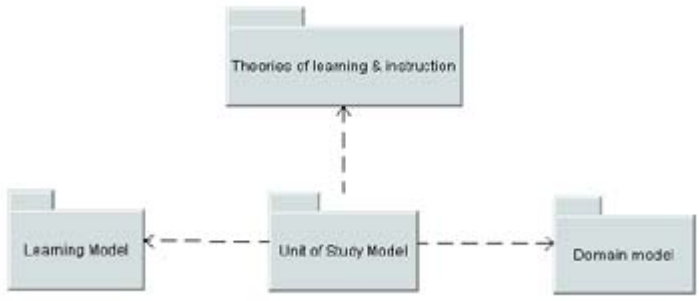


Figura 3.1

Como podemos ver en la figura 3.1, el meta-modelo está compuesto por 4 paquetes:

1. **El modelo de aprendizaje**. Describe cómo los estudiantes aprenden y está basado en las cosas en común de las teorías de aprendizaje.
2. **El modelo de unidad de estudio**. Describe cómo las unidades de estudio, que son aplicables en la práctica real, son modeladas, dando el modelo de aprendizaje y dando el modelo de instrucción.

3. **El modelo de dominio**. Describe el tipo de contenido y la organización de los contenidos. Por ejemplo el dominio de economía, leyes, biología, etc.
4. **Teorías de aprendizaje e instrucción**. Describe las teorías, principios y modelos de instrucción.

Descripción de los subsistemas

El modelo de aprendizaje

En la figura 3.2 podemos ver un diagrama UML que describe el modelo de aprendizaje.

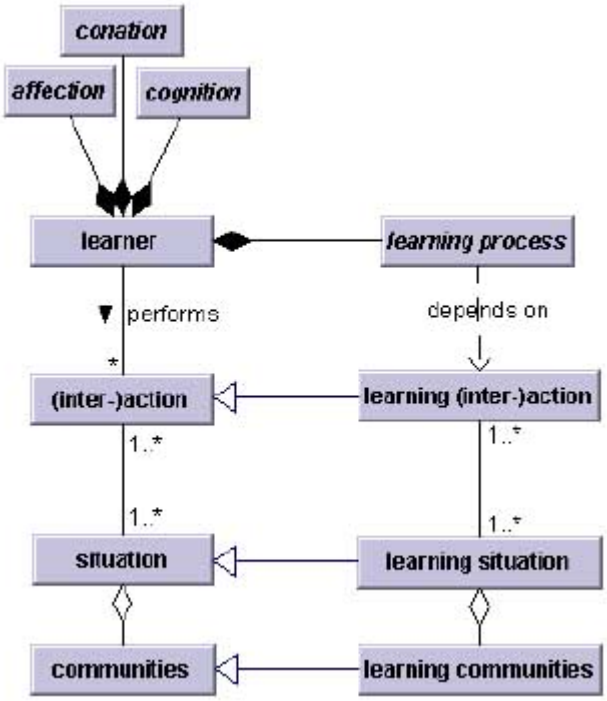


Figura 3.2

El modelo de aprendizaje se basa en los siguientes axiomas:

1. Una persona aprende interactuando en/con el mundo externo.
2. Se puede considerar que el mundo real está compuesto por situaciones personales y sociales que proveen al contexto de acciones.
3. Una situación está compuesta por una colección de objetos y hechos vividos en una interrelación específica.
4. Una parte de las situaciones son las comunidades de prácticas y las comunidades de aprendizaje.
5. Hay diferentes tipos de aprendizaje, el único que nos interesa es el aprendizaje de las medidas de instrucción.
6. El aprendizaje puede ser considerado un cambio en el estado cognitivo de la persona.
7. Las cosas aquí descritas no sólo valen para individuos sino también para grupos de personas o empresas.

El modelo de unidad de estudio

En la figura 3.3 podemos ver el modelo de unidad de estudio.

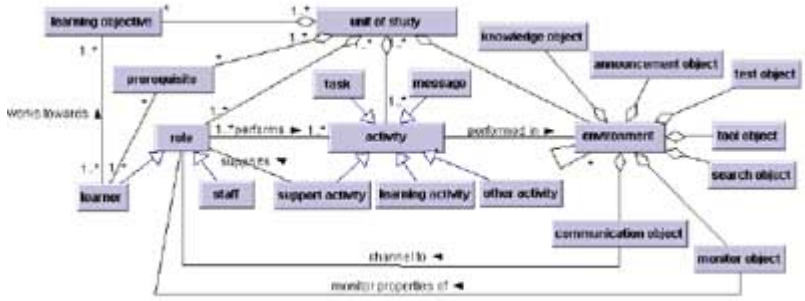


Figura 3.3

Un modelo para una unidad de estudio es el resultado de un proceso de diseño del aprendizaje en el que un producto real (la unidad de estudio) es el resultado. En este modelo se tienen en cuenta temas como los siguientes:

- Los roles de staff y de learner en el proceso de aprendizaje
- Los objetivos del aprendizaje
- Los prerrequisitos de los estudiantes
- Otras características de los estudiantes (estilos de aprendizaje, preferencias,...)
- El dominio del aprendizaje (matemáticas, biología, etc.)
- El contexto del aprendizaje (aprendizaje, biblioteca,...)
- La valoración del aprendizaje.

El modelo de dominio

Cada modelo pedagógico debe tener en cuenta las características del contenido del dominio (matemáticas, derecho, biología,...). Cada dominio tiene su propia organización del conocimiento y competencias.

Teorías de aprendizaje e instrucción

Son los diferentes principios y teorías sobre aprendizaje tal y como vienen en la literatura. Podemos mostrar el diagrama UML de la figura 3.4 para mayor aclaración.

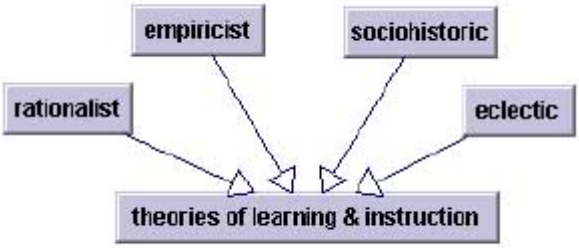


Figura 3.4

Contempla diferentes teorías de enseñanza/aprendizaje:

- Empírico (conductista)

- Racionalista (cognitivista y constructivista)
- Pragmático e histórico-cultural (situacional) o constructivismo social.
- Modelo ecléctico.

Diagrama integrado

Por último, en la figura 3.5 mostramos un diagrama que integra todo lo explicado sobre los subsistemas o paquetes del meta-modelo. Quizás aquí podemos ver de un modo más claro las distintas dependencias entre los paquetes que forman parte del meta-modelo.

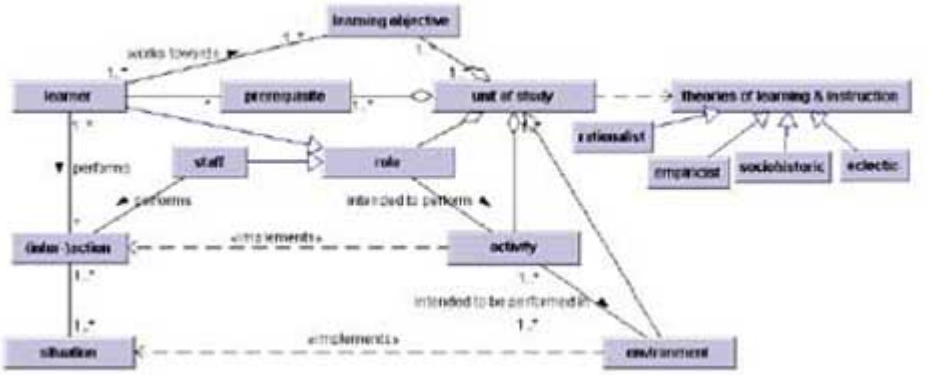


Figura 3.5

IMS Global Learning Consortium (1997)

Introducción

¿Qué es IMS?

El proyecto Instructional Management System (IMS), es un intento por conseguir una especificación para el desarrollo del potencial de Internet como entorno de formación.

IMS reúne un conjunto de organizaciones académicas, comerciales y gubernamentales que trabajan en construir la arquitectura de Internet para el aprendizaje. El proyecto fue fundado en 1997 y existe bajo los auspicios de EDUCAUSE's National Learning Infraestructure Initiative [EDUCAUSE 2002]. El mismo grupo define así su misión: *"El objetivo del proyecto IMS es la amplia adopción de especificaciones que permitirán que contenidos y entornos de aprendizaje distribuidos de múltiples autores puedan trabajar juntos. A tal fin, El proyecto producirá una especificación técnica y un prototipo como prueba de concepto*s" [IMS 1997,2] (Edutec. Revista Electrónica de Tecnología Educativa Núm. 13. / noviembre 00)

No cabe duda de la importancia del trabajo que este grupo viene desarrollando de cara a la interoperabilidad que supondrá la adopción de sus especificaciones como un estándar de facto en la industria y en esa medida, el impulso que traduce para los Entornos Virtuales de Enseñanza y Aprendizaje en Internet.

Inicialmente la labor de IMS se desarrolló tomando como base la educación superior en EEUU, aunque hoy día sus especificaciones engloban gran variedad de contextos educativos, incluyendo formación corporativa y gubernamental.

Los primeros trabajos del IMS se centraron en la definición de un modelo y una arquitectura para los sistemas de aprendizaje distribuido. Sin embargo, sus esfuerzos se reorientaron rápidamente al percatarse de la necesidad de definir previamente un modelo de datos adecuado para describir los recursos, estructuras y demás elementos manejados por los componentes de la arquitectura.

A día de hoy, IMS define y desarrolla especificaciones interoperables usando XML para hacer posible el intercambio de contenidos educativos e información sobre los alumnos entre diferentes sistemas de enseñanza. Estas especificaciones se implementan con el objetivo de hacer más sencillo y más barato el desarrollo de material educativo. Podemos decir que las especificaciones IMS son ya estándares *de facto* para la definición de requisitos educativos y para el desarrollo de productos y servicios relacionados con la educación.

Especificaciones IMS

Las especificaciones realizadas por IMS se centran en diferentes campos que se enuncian a continuación:

Mecanismos de transferencia de los recursos educativos. La especificación "Content Packaging" es la de mayor nivel de implantación en estos momentos. Su objetivo es la creación de paquetes con formato estándar compuestos por Objetos Educativos, y la especificación de los ficheros que hacen referencia a los objetos y las instrucciones para que el Learning Management System pueda organizar los objetos dentro del paquete. Esta especificación ha sido adoptada por ADL como parte del

proyecto SCORM y también usada por Microsoft en su sistema de enseñanza LRN [LRN 2003].

(Se presenta la especificación: **IMS Content Packaging Specification**. Versión 1.1.2. Final Release.)

Metadatos para recursos educativos. La especificación relativa a metadatos del grupo IMS "Learning Resource Metadata" es la principal fuente para el proceso de estandarización del Learning Object Metadata en el IEEE [LOM 2001].

(Se presenta la especificación: **IMS Meta Data Specificatio**n. Versión 1.2.1. Final Release.)

Información sobre perfiles de alumnos. La especificación "Learner Information Package" define un formato para estructurar la información relativa al alumno.

(Se presenta la especificación: IMS Learner Information Package Specification. Versión 1.0.0. Final Release.)

Mecanismos de evaluación. La especificación "Question and Test Interoperability" tiene un gran nivel de aceptación y está siendo utilizada por multitud de herramientas comerciales. Esta especificación proporciona un formato estándar para codificar cuestionarios on-line, exámenes y grupos de exámenes.

(Se presenta la especificación: IMS Question & Test Interoperability Specification. Versión 1.2. Final Release)

Diseño pedagógico. El grupo "Learning Design" se ocupa de describir y codificar las metodologías educativas implícitas en un proceso educativo.

(Se presenta la especificación: IMS Learning Design. Versión 1.0. Final Specification.)

Organización de los contenidos educativos. La especificación "Simple Sequencing", se ocupa de la definición de los mecanismos que permiten la secuenciación de los recursos educativos dentro de un sistema e-learning.

(Se presenta la especificación: IMS Simple Sequencing Public Draft Specification. Versión 1. Public Draft.)

Descripción de sistemas basados en competencias. La especificación "Reusable Competencies Definition" tiene como objetivo definir una nomenclatura estándar para etiquetar los distintos componentes de un sistema de competencias.

(Se presenta la especificación: IMS Reusable Definition of Competency or Educational Objective. Versión 1.0. Final Specification.)

Interoperabilidad entre repositorios digitales. La especificación "Digital Repositories" tiene como objetivo la elaboración de recomendaciones que puedan permitir la interoperabilidad entre repositorios digitales. Este grupo se encuentra en proceso de elaboración de especificaciones y recomendaciones para permitir la interoperabilidad entre repositorios digitales.

(Se presenta la especificación: IMS Digital Repositories Specification. Versión 1. Final Specification.)

Gestión de registro de alumnos. La especificación "Enterprise Specification" [IMS_ES 2002] define una estructura para realizar el intercambio de información de registro de los alumnos y los horarios de los cursos. En la primera fase de desarrollo de la especificación, el objetivo era permitir la interacción entre los LMS, las aplicaciones de Administración de alumnos, y los sistemas de Recursos Humanos. Actualmente, se encuentra en fase de revisión para permitir su extensión a otro tipo de sistemas de aprendizaje, y para definir la especificación de la arquitectura de flujos de mensajes intercambiados.

Estudio de la accesibilidad de sistemas. El término tecnología accesible hace referencia a aquella tecnología a la que se puede tener acceso a través de más de un canal de salida, por lo general en referencia a salidas audibles, visuales, o motoras [IMS_A 2002] (IMS AccessForAll Meta-data Specification). Está orientado a personas con problemas de visión, audición, etc., para que puedan acceder sin problemas al sistema.

Análisis de cada una de las especificaciones

Para cada especificación responderemos dos preguntas claves:

1. **¿Qué es?** Introducción de la especificación, casos de uso, y modelo conceptual que sigue.
2. **¿Qué relación tiene con el resto de las especificaciones?** Se muestran las relaciones y las dependencias que presenta la especificación con el resto.

IMS Content Packaging Specification

¿Qué es?

La necesidad de intercambiar recursos educativos entre los sistemas de aprendizaje electrónico y las herramientas de autor ha motivado la aparición de formatos de empaquetado de contenidos en unidades físicas aisladas. Una unidad en la que se encapsulasen varios recursos educativos junto con toda la información relacionada con ellos, como los metadatos asociados o el modo en que debieran organizarse, permitiría un traslado entre sistemas heterogéneos más cómodo, seguro y eficaz [Anido-Rifón 2001].

Los formatos de empaquetado de contenidos debieran ser neutrales en cuanto a la amplitud de los recursos que permite encapsular, es decir, debieran permitir el encapsulamiento tanto de recursos aislados, como cursos completos e incluso varios cursos juntos.

Esta especificación tiene como objetivo permitir la creación de contenidos reutilizables e intercambiables. Ofrece una forma de empaquetar los contenidos educativos tal cómo un curso, un conjunto de cursos, o cualquier recurso que pueda necesitar un curso.

De esta manera, se necesita que tres de los actores que intervienen en el proceso de enseñanza se impliquen en la utilización de esta especificación [IMSCP_INFO 2001]:

- **Los autores de los contenidos**, que deben construirlos ajustándose a las reglas que esta especificación propone.
- **Los sistemas de administración**, que al ajustarse a este formato son capaces de crear estos paquetes, agregarlos e intercambiarlos. Estos se controlan por **administradores**, que se encargan de elegir las diferentes formas de ver los contenidos y de distribuirlos.
- Los **alumnos**, que serán los que aprendan los contenidos que se les ofrecen mediante un sistema de administración.

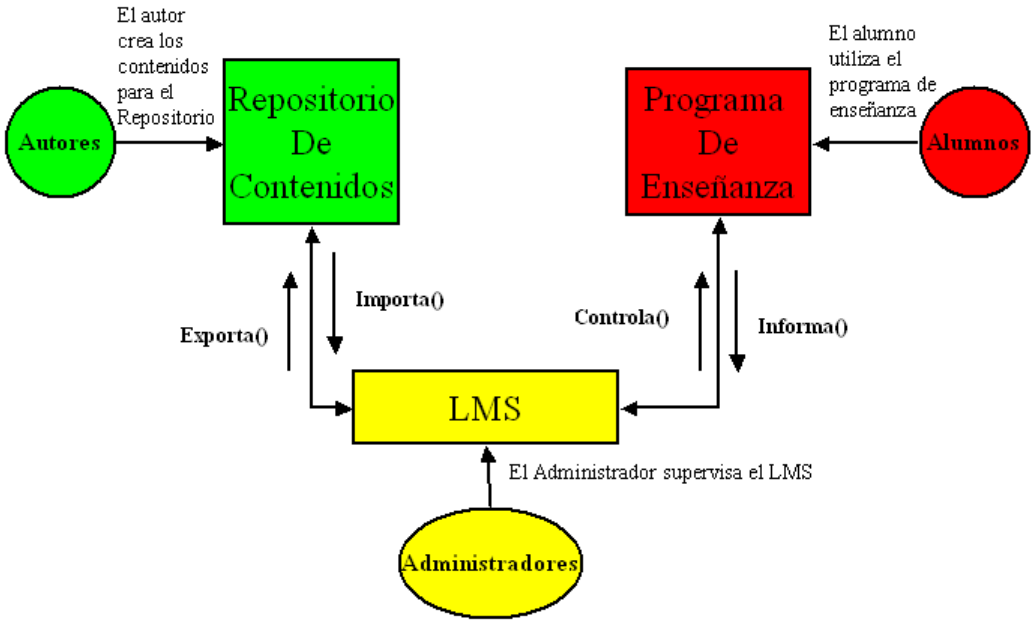


Figura 2. Relación que existe entre los usuarios de la especificación Content Packaging.

En la figura 2 se puede observar la relación que existe entre el almacén de contenidos, el LMS, y el programa que está en contacto con los alumnos.

Estas son las tres partes del sistema de enseñanza que tienen que utilizar esta especificación para que se pueda dar la interoperabilidad de los contenidos.

Utilizando esta propuesta de IMS, es posible encapsular en un solo fichero todos los recursos que conforman un curso y sus metadatos asociados, junto con una o varias organizaciones alternativas de tales recursos.

Estas organizaciones pueden estar definidas utilizando el modelo que se desee, por ejemplo se podría utilizar el CSF descrito por el ADL (se describirá más adelante), si bien se recomienda que al menos una de las organizaciones incluidas en el paquete se especifique mediante el formato Organization (elemento aconsejado por la especificación). De este modo se asegura que los sistemas compatibles con la especificación IMS sean capaces de interpretar cómo están estructurados los recursos contenidos en el paquete.

Además, para asegurar esta interoperabilidad, la especificación Content Packaging almacena los contenidos en paquetes. Un paquete no tiene por qué corresponder a un curso completo. Puede ser parte de un curso, un curso completo, o incluso una colección de cursos. Pero todos los implicados en el proceso de enseñanza que aparecen en la figura 2 deben ser capaces de abrir estos paquetes y de entender lo que dentro de ellos se almacena.

¿Qué relación tiene con el resto de las especificaciones?

Por su naturaleza, esta especificación tiene relación con prácticamente todas las especificaciones de IMS. Por esto enumero sus relaciones en orden inverso, es decir, se detallarán en este mismo apartado para el resto de las especificaciones cuando estas estén relacionadas.

¿Qué es?

La gran complejidad tanto de las estructuras cómo de los contenidos de los recursos de aprendizaje hace necesaria la aparición de información adicional sobre ambos [IMSMD_INFO 2001]. Estos son los llamados metadatos, o información sobre los datos. Son etiquetas descriptivas que aportan información orientada a hacer más eficiente la búsqueda y utilización de los recursos. Estas etiquetas se encuentran agrupadas y ordenadas en un conjunto de estructuras, por lo que también pueden intercambiarse entre sistemas que se ajusten a esta especificación.

El proyecto IMS detectó que una de las primeras tareas en el proceso de estandarización era la de llegar a un acuerdo en los metadatos para recursos educativos. Desde 1998, cuando hicieron la propuesta conjunta con ARIADNE para la creación de LOM (Learning Object Metadata), IMS contribuyó regularmente a su evolución. Actualmente están usando los metadatos de LOM en sus especificaciones.

Por ejemplo, la última especificación de metadatos de IMS está basada en la versión 6.1 de LOM [LTSC 2002].

Sin embargo, IMS considera que el número de elementos definidos en LOM es demasiado grande. Muchas organizaciones dentro de la comunidad IMS recomendaron que se identificase un conjunto más pequeño de elementos LOM para simplificar los esfuerzos iniciales de implementación. La propuesta de metadatos de IMS intenta hacer el esquema de LOM más flexible proporcionando dos especificaciones diferentes: IMS Core (19 metadatos de LOM), que contiene metadatos fundamentales para la descripción de recursos, e IMS-SEL *(Standard Extensión Librar*y), que agrupa el resto de los elementos LOM. IMS se ha apoyado para reducir LOM en la especificación Dublín Core [Core 2002], por lo que IMS LRM (Learning Resource Metadata) no sólo es una aportación, sino que también es un trabajo de confluencia entre las dos iniciativas referentes a metadatos con mayor aceptación en la actualidad [Sancho 2002].

A continuación veremos un ejemplo de la jerarquía propuesta por IEEE para los metadatos y que ha adoptado la especificación que nos ocupa.

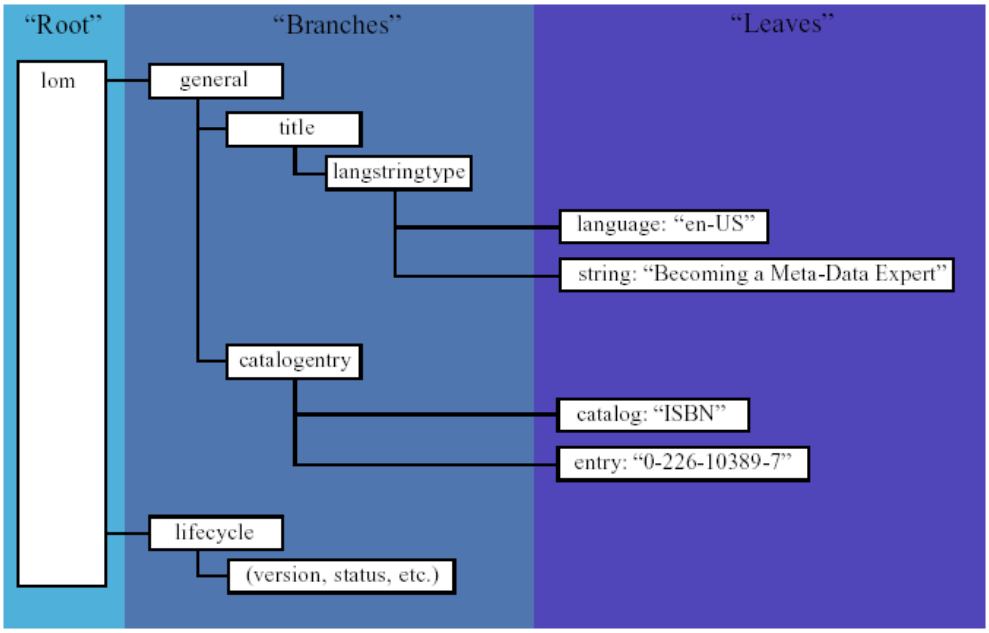


Figura 4. Ejemplo de la jerarquía propuesta por IEEE para los Metadatos. Desde la raíz hasta las hojas.
(IMSMD_BEST 2001)

En la figura 4 se muestra uno de los muchos caminos posibles que hay para llegar a una de las hojas desde la raíz. La raíz es un elemento <lom> que posee un total de nueve hijos. En la figura se hace un seguimiento del hijo <general> y a su vez de su hijo <title>. De esta manera vemos cómo quedaría reflejado el título y el lenguaje utilizado del recurso al que fuese referido este metadato. Este ejemplo muestra claramente cómo se irían definiendo los diferentes campos que propone la especificación.

Se da la paradoja de que aunque nos encontramos ante la especificación más amplia de todas, también resulta que todos sus campos son opcionales. No hay ningún elemento imprescindible, lo cual facilita enormemente su uso por parte de un diseñador, ya que puede tomar los metadatos como un consejo pero nunca como una ligadura.

¿Qué relación tiene con el resto de las especificaciones?

La especificación IMS Learning Resource Metadata se encuentra fuertemente relacionada con todas las demás, ya que cualquier objeto educativo puede ser el objetivo de los metadatos. Aún así, veremos las que describen los objetos que normalmente son ampliados con los metadatos descritos en esta especificación.

IMS Content Packaging

La especificación IMS Learning Resource Meta Data aporta información relevante al manifiesto tal como el ciclo de vida que ha llevado el curso, o detalles sobre las características educativas del mismo. Creemos que todo curso debería ir acompañado de su archivo de metadatos correspondiente, ya que esto reduciría el tiempo de comprensión del mismo por parte de los receptores.

IMS Learning Design

Existe una relación complementaria entre estas dos especificaciones. Como hemos apuntado anteriormente, la especificación Learning Design (LD) incluye en sus elementos información pedagógica muy útil para los cursos. En cambio, la especificación IMS Learning Resource Metadata aporta información a cualquier objeto educativo, no sólo los cursos, por lo que desde el punto de vista de estos resulta más vaga. Proponemos entonces que sea la especificación LD la que aporte información extra a los cursos y dejar que sean los metadatos los que se encarguen del resto de objetos implicados en el proceso de enseñanza.

IMS Learner Information Package Specification

¿Qué es?

Sin duda la parte más importante de un proceso de enseñanza son las personas implicadas en él. Uno de los grandes logros de la enseñanza electrónica ha sido el conseguir una transferencia de información más personalizada que en la enseñanza tradicional. Este es el ámbito del IMS Learner Information Package Specification (LIP). IMS LIP incluye los resultados obtenidos en PAPI (Public and Private Information) [PAPI 2000], que es la propuesta de estandarización de información del alumno realizada por el IEEE. Las relaciones entre PAPI e IMS LIP se detallan en la figura 5

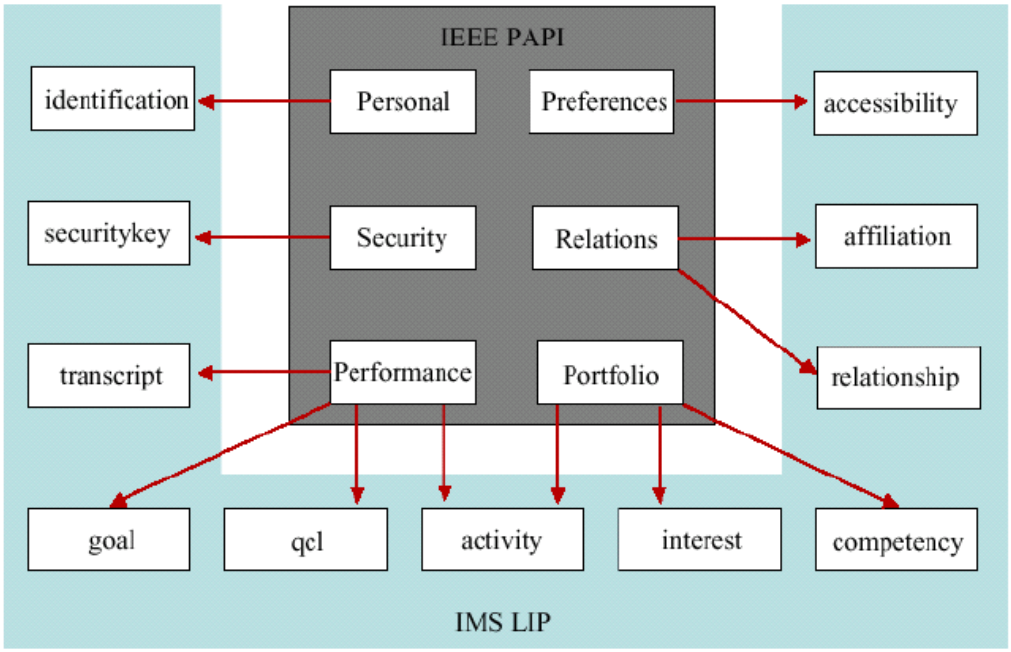


Figura 5. Relación entre IMS LIP y PAPI (IMSLIP_BEST 2001)

Esta nueva especificación nos indica la forma de almacenar la información referente a un alumno (o grupo de alumnos) o incluso a un productor de contenido educativo [IMSLIP_INFO 2001]. El objetivo de la misma es definir una estructura que permita el intercambio de paquetes con información relativa a cualquiera de los implicados en el sistema de enseñanza. Junto a esto se conseguiría:

- Llevar un registro del historial, objetivos y logros.
- Conseguir un registro de alumnos y profesores en el sistema.
- Ajustar las oportunidades de aprendizaje a las necesidades del alumno.

Esta especificación introduce un nuevo concepto: Sistema de Información del Alumno (SIA). Esto supone toda una arquitectura montada alrededor de la información obtenida de los implicados en el proceso de enseñanza. En la figura 6 se muestran los componentes de un SIA.

Los componentes clave mostrados en la figura 6 son:

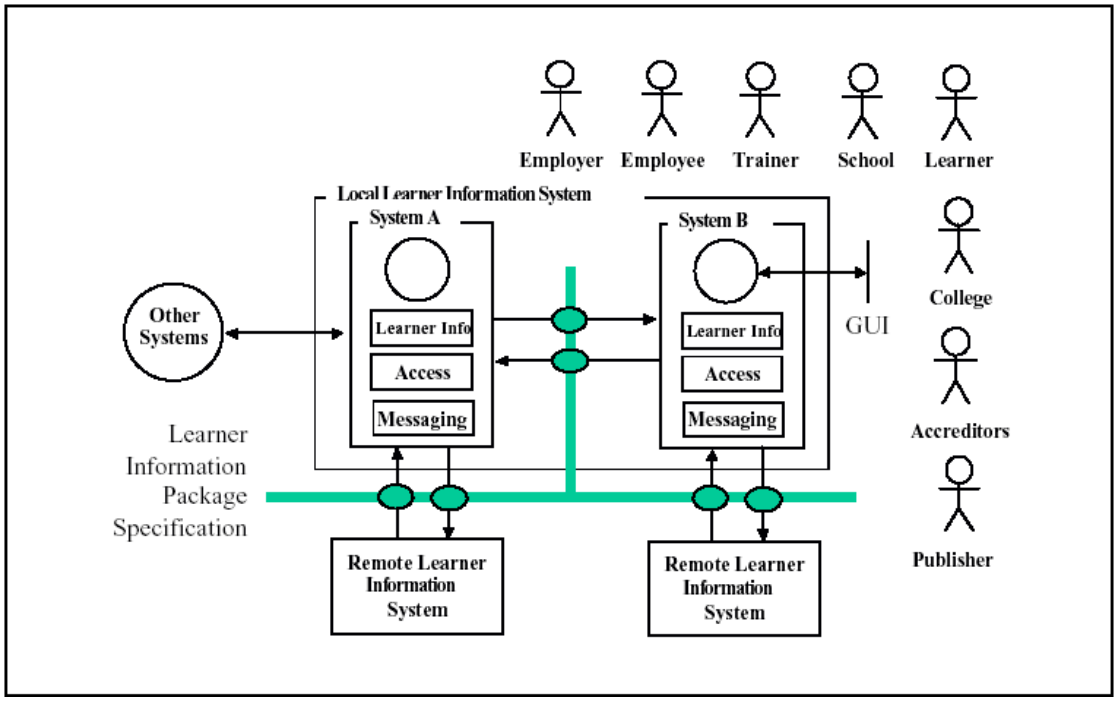


Figura 6. Componentes de un sistema de información de alumnos. (IMSLIP_INFO 2001)

Local Learner Information System (Sistema local de información de alumnos), son los servidores locales accesibles por los usuarios.

Remote Learner Information System (Sistema remoto de información de alumnos), son servidores que almacenan copias totales o parciales de la información que existe en los servidores locales.

Estructuras de datos*:*

- Learner Info, datos actuales del alumno.
- Access, permisos para la visualización de los datos.
- Messaging, protocolo de mensajería que se usará para el intercambio de datos.

El hecho de incluir en un sistema de enseñanza información sobre personas no es algo aislado. Para poder hacerlo, el sistema tiene que cumplir una serie de requisitos. De hecho, esta especificación no es válida si el sistema que va a implementarla no cumple las siguientes condiciones:

Escalabilidad. El sistema debe estar preparado para gestionar un número grande de archivos que incluyen información de usuario. Una buena forma de hacerlo es en conjunción con la especificación IMS Content Packaging vista anteriormente.

Privacidad y control de los datos. Ahora no estamos almacenando, cómo en otras especificaciones, datos de cursos. Los datos que ahora se manejan son datos personales, y cómo tal, deben ser absolutamente confidenciales. La especificación está diseñada para que le sea sencillo implementar un sistema de protección a cualquier sistema. Un sistema que no sea capaz de garantizar la seguridad no debería tratar de utilizar esta especificación.

Flexibilidad. Cómo se verá cuando nos sumerjamos en esta propuesta, en ella se almacenan datos del tipo de navegación del alumno en los cursos. Estos datos están ligados con la estructura de los cursos. Ya que los cursos pueden estar renovándose continuamente, el sistema debe controlar los posibles cambios en los datos de navegación con cada cambio de la organización de un curso.

El modelo de datos que se propone es estructurado. Además, dentro de los documentos de IMS (al igual que en el resto de las especificaciones) se incluye un documento que enlaza este modelo de datos con XML. Pero no hay que olvidar que esto se trata de una propuesta, por lo que se podría utilizar cualquier otro formato para su almacenamiento (por ejemplo. una base de datos).

La información almacenada es de dos tipos. Por un lado se guardan datos del tipo nombre del alumno, cursos completados, o preferencias en la visualización de los cursos. Por otro lado se almacenan una serie de metadatos referidos a cada uno de los campos dónde se han almacenado los datos. Estos metadatos son del tipo:
Información temporal sobre los datos, información referente a seguridad e información sobre identificación e indexación.

¿Qué relación tiene con el resto de las especificaciones?

IMS Content Packaging

Tal y cómo hemos visto en el apartado anterior, nuestra especificación se encuentra estrechamente ligada con el CP por la relación entre las organizaciones de un curso y la navegación por parte de un alumno. Una vez definida la organización de un curso en CP, su elemento <*activity*> será el encargado de indicar qué partes de esta organización han sido recorridas. Esto resultará decisivo a la hora de establecer si un alumno cumple los prerrequisitos necesarios para acceder a otras partes del curso.

Además, IMS recomienda el empaquetamiento de los archivos LIP dentro del formato descrito por el CP cuando se quieren intercambiar perfiles de alumnos entre sistemas.

IMS Enterprise Specification

La especificación LIP que nos ocupa, no sólo puede definir personas, sino también grupos de personas, lo que entra en el ámbito de la especificación Enterprise. Aún así, se recomienda el uso de esta última para la definición e intercambio de datos para el caso de grupos por ser éste su objetivo último.

IMS Question&Test Interoperability

Aunque dentro del elemento <*activity*> se proponga una forma de almacenamiento de resultados del alumno, no se considera una manera óptima de hacerlo. Es la especificación IMS Question&Test Interoperability la encargada de almacenar, de manera que sea posible su intercambio, los resultados de los alumnos. Por tanto, aunque dentro de LIP se nos ofrezca, no debemos caer en la trampa de utilizar esta especificación para el intercambio de resultados ya que no es su finalidad.

IMS Question&Test Interoperability (QTI)

¿Qué es?

En este capítulo se va a hacer un análisis detallado de la especificación de IMS Question&Test Interoperability.

Una parte indispensable de la enseñanza es su método de evaluación al alumno. En la enseñanza electrónica también se considera fundamental ofrecer, cómo recta final del proceso de aprendizaje, una herramienta que permita examinar el trabajo realizado por la persona que trata de aprender.

Para ello nace esta especificación de IMS, que ha sido creada con la intención de ofrecer una estructura básica que describa la forma de representar evaluaciones (assessments) y sus calificaciones correspondientes. El objetivo es conseguir una forma de representar exámenes y resultados de manera que estos sean intercambiables entre los diferentes LMS (Learning Management System). Así, podríamos disponer de almacenes de preguntas y bases de datos con los resultados obtenidos por los alumnos a los que cualquier sistema de enseñanza electrónica podría acceder. Esta especificación nos sorprende con algo que no habíamos visto en las anteriores: engloba dos especificaciones en una. Por un lado nos encontramos con la descripción de ASI (assessment, section, item) que se encarga de la estructura que tienen que seguir los objetos que se encargan de la evaluación, y por otro lado la estructura de los objetos que almacenan la información sobre los resultados obtenidos por el participante. Estas dos especificaciones son independientes entre sí, y no es necesario el uso de una de ellas para utilizar la otra. En este trabajo sólo nos centraremos en la especificación ASI.

QTI y el proceso de enseñanza

En el siguiente gráfico se ilustra dónde encaja esta nueva especificación dentro del complicado proceso de enseñanza, y las partes de las que se compone.

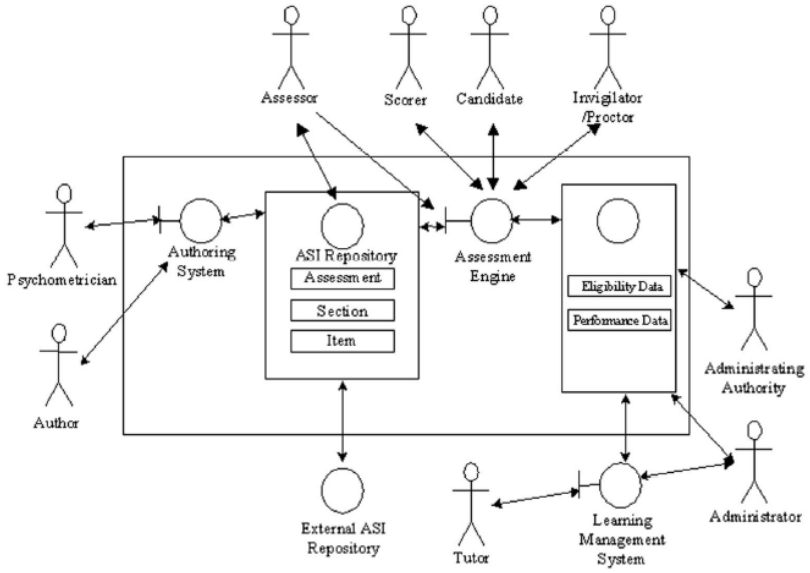


Figura 8. Componentes del sistema de exámenes. (IMSQTI_INFO 2002)

Aquí se pueden diferenciar las siguientes partes [IMSQTI_INFO 2002]:

Sistema de autoría (authoring system). Es el encargado de facilitar la creación y edición de los exámenes. Estos están compuestos por *Item*s, *Section*s, y *Assessments* (representados por las siglas ASI).
Repositorio ASI (ASI Repository). Es una base de datos local dónde se almacenan los ASIs. Esta base de datos debe proporcionar funcionalidades para la búsqueda y recuperación de los objetos que almacena para así facilitar el trabajo al sistema de autoría, al motor de evaluación, o para cualquier consulta independiente. El *External* **ASI Repository** se refiere a la capacidad de importar objetos de un repositorio externo gracias a la interoperabilidad de esta especificación.
Motor de evaluación (assessment engine). Se encarga de generar los resultados de los test en función de los objetos del repositorio ASI que se hayan usado, ya que cada uno de ellos lleva asociada una puntuación que se le otorga al candidato en caso de que se responda correctamente.
Repositorio de datos de los candidatos (Elegibility Data, y Performance Data). Es dónde se almacenan los datos de los resultados obtenidos por los alumnos en cada uno de los exámenes. Es importante remarcar que este repositorio es accesible directamente por el LMS (Learning Managment System), y que no es parte de la especificación Learner Information Profile, aunque ya se ha propuesto la inclusión de estos datos en versiones posteriores.
Learning Managment System o LMS, es el proceso o sistema responsable de toda la arquitectura de enseñanza.

IMS Learning Design

¿Qué es?

Esta especificación es una integración entre el Educational Modelling Languaje (EML) realizado por la Open University de Holanda (OU), y el resto de las especificaciones existentes en IMS [IMSLD_INFO 2003]. El objetivo fundamental de esta nueva especificación es el desarrollo de un entorno que permita la diversidad pedagógica y la innovación a la vez que hace posible el intercambio e interoperabilidad de contenidos educativos.

Antes de que el grupo de trabajo de IMS se planteara este objetivo, la OU de Holanda había realizado un estudio en profundidad de una amplia gama de necesidades pedagógicas que, posteriormente, reflejó en un meta-lenguaje llamado EML. Obviando la parte pedagógica, cualquier diseño de aprendizaje pasa por un *Método* que desarrolla una serie de *Actividades* para unos individuos que desempeñan unos *Roles* dentro del proceso de enseñanza. Estas, son las tres palabras claves que se introducen dentro de EML. Desde nuestro punto de vista, son tres las características principales de este lenguaje [Sancho 2002]:

- Implementación en XML de un modelo jerárquico que describe los contenidos educativos desde un enfoque pedagógico.
- Clasificación semántica de los objetos educativos y definición de las diferentes dependencias que existen entre ellos.
- Creación de una notación capaz de adaptarse tanto a las piezas que componen el entorno educativo, cómo al entorno educativo en su totalidad.

El primer paso fue el utilizar EML tal y cómo lo habían desarrollado sus creadores. A continuación se fue llevando a cabo una labor de integración que ha llevado a realizar muchos cambios sobre el meta-lenguaje original.

El primer problema con que se encontró IMS fue el modo de reducir EML sin que este perdiera su significado. EML resulta demasiado extenso, por lo que se ha tenido que conseguir un equilibrio entre la versatilidad y la complejidad que conlleva un lenguaje tan amplio. Poco a poco se han ido eliminando partes de EML que, no se consideraban lo suficientemente relevantes, o bien resultaban redundantes con el resto de las especificaciones. Un ejemplo de esto último lo tenemos en la capacidad para evaluar los resultados de un test que contemplaba EML. Esta parte ha sido descartada por pertenecer al ámbito de la especificación QTI (aunque esta no lo contempla por el momento, e IMS ha apostado por el entorno propuesto por ADL [ADL 2002] para solucionarlo).

En la última versión de esta especificación, se ha conseguido un meta-lenguaje que permite englobar diferentes estrategias pedagógicas dentro del mismo diseño, abarcando así las necesidades de diferentes usuarios. Además, dentro del amplio abanico de estrategias pedagógicas, contempla lo que podríamos llamar "estrategia mixta", que permite entrelazar la enseñanza electrónica con otros métodos de enseñanza (presencial, libros, periódicos, etc.) dentro de la misma unidad de aprendizaje.

Podemos afirmar que se ha conseguido con esta especificación un método para dotar de un enfoque pedagógico a los objetos educativos.

IMS Learning Design se encuentra especificada en tres niveles de implementación. Se presentan diferentes esquemas [XML Schemas] para cada uno de los niveles. Veamos que incluye cada uno de ellos:

Nivel A. Contiene el núcleo de la especificación, por lo que es aquí dónde se implanta todo el vocabulario nuevo que LD aporta.
Nivel B. Añade Propiedades y Condiciones al nivel anterior. Esto permite añadir secuencias e interacciones más complejas basadas en las características del alumno. El que se ofrezcan estas características en un esquema aparte permite usarlas de forma independiente, normalmente como una mejora en la especificación IMS Simple Sequencing.
Nivel C. Añade Notificación al nivel B. Aunque esto pueda parecer un avance pequeño respecto del anterior nivel, supone un gran paso que facilita la implementación de la comunicación entre objetos educativos.

Esta división implica que en un documento escrito bajo esta especificación deberá aparecer el nivel de implementación que se utiliza.

Los objetivos que persigue esta especificación son [IMSLD_INFO 2003]:

Globalidad. Debe ser capaz de describir el proceso de aprendizaje en su totalidad dentro de una unidad de aprendizaje, incluyendo referencias a objetos de aprendizaje digitales y no digitales, y a todos los servicios que sean necesarios para completar el proceso.
Flexibilidad Pedagógica. La especificación tiene que ser capaz de aportar significado y funcionalidad pedagógica a todos los elementos que están dentro de una unidad de aprendizaje. Debe además permitir todos los enfoques pedagógicos sin decantarse por ninguno en concreto.
Personalización. Dentro de un diseño de aprendizaje se han de poder describir aspectos referentes a la personalización. Por tanto, el contenido y las actividades descritas en una unidad de aprendizaje deben poder adaptarse según las preferencias, necesidades, y circunstancias de los usuarios. Además, se le deben dar al estudiante facilidades para modelar el entorno a su gusto.
Formalización. Se tiene que conseguir una descripción formal del diseño de aprendizaje para que sea posible su procesamiento automático.
Abstracción. La descripción debe tener un nivel de abstracción que permita repetir la misma ejecución con parámetros diferentes y diferentes usuarios.
Interoperabilidad. Los diseños de aprendizaje deben ser intercambiables.
Compatibilidad. La especificación es compatible con el resto de las especificaciones de IMS.
Reusabilidad. Permite integrar todo tipo de productos educativos, y reutilizarlos en diferentes contextos.

¿Qué relación tiene con el resto de las especificaciones?

La naturaleza de esta especificación hace que tenga una fuerte relación con el resto.

Además, el hecho de que haya sido realizada por otra organización aumenta el grado de acoplamiento, ya que muchos aspectos que contempla ya se encontraban descritos en las especificaciones anteriores.

Veamos una a una todas las especificaciones con las que guarda relación:

IMS Content Packaging

Tal y cómo hemos mostrado en el apartado anterior, es aconsejable que el IMS Learning Design vaya integrado dentro del manifiesto para crear una Unidad de Aprendizaje.

IMS Simple Sequencing

Al igual que pueden introducirse los elementos de esta especificación dentro del Content Packaging para indicar la forma de realizar la secuencia de aprendizaje, también se puede utilizar para definir la forma de secuenciar tanto los recursos dentro de un objeto educativo cómo los objetos educativos dentro de un entorno. El elemento raíz de esta especificación (<learning_design>) posee un atributo obligatorio llamado *sequence-used* que informa si se van a usar los elementos de IMS Simple Sequencing.

IMS Meta-Data

A lo largo de toda la especificación Learning Design aparecen contenedores destinados a albergar meta datos.

IMS Question and Test Interoperability

La principal pregunta que se nos plantea es la forma de integrar esta especificación dentro del nuevo entorno de aprendizaje que propone Learning Design. Se puede hacer de dos formas. La primera consiste en añadir una referencia a un esquema, dentro del elemento <environment>/<learning-object>, que permita integrar los elementos de QTI. De esta manera, los exámenes pueden considerarse actividades y tener así todas las ventajas del LD. La segunda manera es menos elegante, y se trata de añadir los exámenes cómo recursos separados dentro del manifiesto.

IMS Learner Information Package

La información de los implicados en el proceso educativo es el primer paso para poder crear un diseño de aprendizaje (por ejemplo, para asignar roles). Por ello, toda la estructura de elementos definida en la especificación LD puede asignarse a IMS LIP.

MS Reusable Definition of Competency or Educational Objective (RDCEO)

Tal y cómo se vio en el apartado anterior, tanto los Objetivos educativos cómo los Prerrequisitos pueden referirse a recursos definidos bajo la especificación RDCEO.

IMS Enterprise

Esta especificación se puede usar para asignar los roles a los alumnos y profesores en el momento que se aplicase un diseño de aprendizaje.

ADL SCORM

Aunque no acostumbramos en este apartado a relacionar las especificaciones con iniciativas de otros consorcios, queremos resaltar la utilización que se le está dando a esta especificación en SCORM. A día de hoy, ya se recomienda en este consorcio la inclusión de IMS LD para aportar valor pedagógico a los contenidos educativos, y ya hay cursos de ejemplo desarrollados por ADL en los que se incluye IMS LD.

IMS Simple Sequencing

¿Que es?

Esta especificación describe las reglas necesarias para controlar el flujo de las actividades educativas basándose en los resultados obtenidos por los alumnos en sus interacciones con los contenidos educativos [IMSSS_INFO 2002]. En línea con la filosofía de IMS, esta información de secuenciamiento debe ser intercambiable entre diferentes sistemas educativos con ayuda de herramientas de exportación e importación. Los componentes de los sistemas de administración encargados de procesar la información descrita en esta especificación reciben el nombre de "motor de secuenciamiento".

La palabra "simple" en el nombre de la especificación se debe a que sólo abarca una pequeña parte de las infinitas formas de secuenciar los contenidos. Aún así, los tipos de secuenciamiento que presenta son el fruto de la experiencia obtenida en varios sistemas de enseñanza que están en marcha, por lo que representan un segmento muy productivo.

Simple Secuencing está orientada exclusivamente al alumno. No define secuencias para ninguno de los otros implicados en el proceso de enseñanza cómo podrían ser los profesores, tutores, etc. Está orientada a aplicaciones Web, y aunque también podría adaptarse a otros tipos de aplicaciones, perdería parte de su potencia. Las interfaces encargadas de presentar los contenidos y la forma de interactuar con el LMS están fuera del alcance de esta especificación.

En la figura 11, se muestra el espacio completo de tipos de secuenciamiento, y, rodeado en azul, la parte que pretende abarcar esta especificación. En esta figura distinguimos:

Secuencias dirigidas. Son aquellas en las que no se permite ninguna interacción con el usuario. Los contenidos se presentan de una forma fija, es decir, la acción del alumno no modifica la forma de mostrar los contenidos. Aunque los contenidos sean fijos, es posible presentarlos aleatoriamente o siguiendo caminos diferentes según prefiera el tutor.

Secuencias guiadas por el alumno. Se permite decidir al alumno los contenidos que desea visualizar. Esta elección puede ser total *(Full choic*e), en dónde el usuario podría elegir que contenido quiere ver en cada momento, o parcial (partial choice), en donde se le imponen restricciones tales como prerrequisitos. Un ejemplo de elección total sería mostrar el árbol de navegación completo al alumno, darle la posibilidad de navegar por todos los contenidos.

Secuencias adaptativas. Este es el grado más alto a la hora de mostrar contenidos ordenados. En este tipo de secuencias, el sistema es capaz de decidir la manera de secuenciar los contenidos basándose en las características y referencias del alumno. Esta especificación sólo las contempla de una manera limitada debido a su innegable dificultad. Cómo ejemplo de secuencias adaptativas imitadas valdría un sistema en el que, basándose en la información almacenada en el archivo de personalización del alumno, se mostrasen los contenidos de un determinado modo.

Todas las actividades que administra esta especificación se organizan con la ayuda de un árbol de actividades *(activity tre*e). Éste es un árbol cuyas ramas no tienen por qué tener la misma longitud, y cuyos elementos tienen que cumplir que toda la información del padre debe estar contenida en los hijos. Además, la especificación define la forma de recorrer el árbol en profundidad. Así, en la figura 12 el primer elemento sería A, seguido de AA, AAA, AAB, AAC, AB, etc.

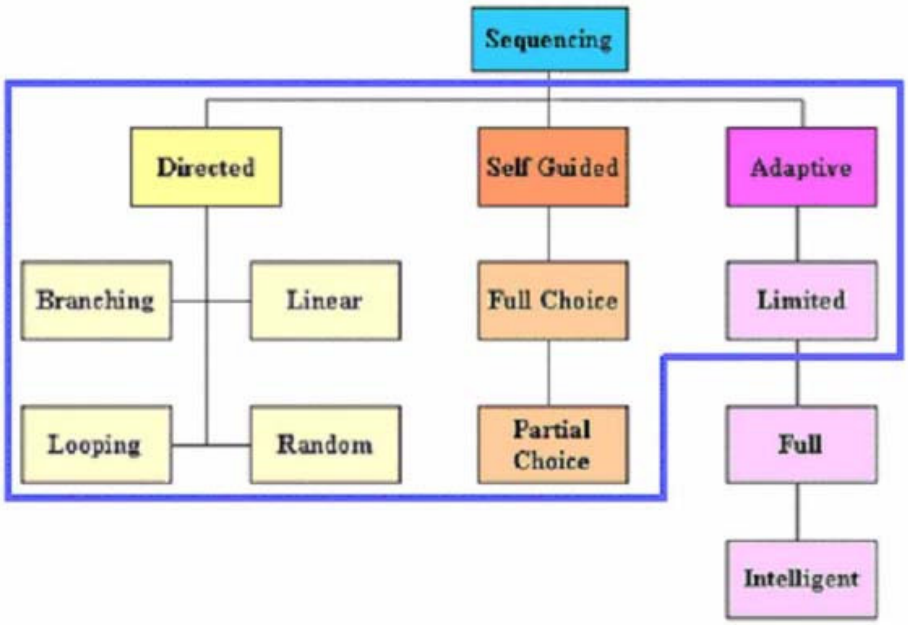


Figura 11. Espacio de secuencias abordado por Simple Sequencing. (IMSSS_INFO 2002)

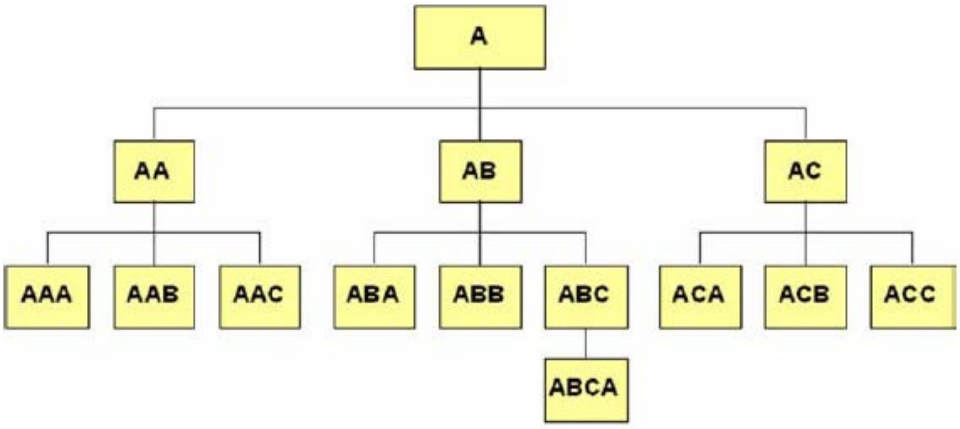


Figura 12. Ejemplo de un árbol de actividad. (IMSSS_BEST 2002)

Suponiendo que un árbol de actividad ha sido creado e inicializado, vamos a hacer un repaso por todos los pasos que se recorren durante el proceso de secuenciado. [IMSSS_INFO 2002]

1. El alumno accede al LMS y establece un contexto dentro de una unidad de aprendizaje.
2. El sistema inicia el proceso de secuenciamiento emitiendo una petición de navegación (normalmente del tipo “start” o “resume all”).
3. Con la información de seguimiento del alumno y la petición de secuenciamiento, se recorre el árbol en busca de la actividad adecuada para presentársela al alumno (que corresponderá con un nodo del árbol).
4. Una vez seleccionada una actividad, se determina si ésta puede ser mostrada (permisos, y prerrequisitos). En caso afirmativo, se preparan todos los recursos asociados a la actividad. En caso contrario, todo el proceso de secuenciamiento se detiene a la espera de otra petición de navegación.
5. Una vez recibido el recurso de aprendizaje, el alumno interactúa con él y todos los procesos encargados del secuenciamiento permanecen a la espera de nuevas peticiones.

6. El alumno, el sistema o una actividad determinada invocan un evento de navegación (Continuar, Anterior, Salir, etc.). En ese momento, el sistema informa al proceso de secuenciamiento de que se ha producido dicho evento.
7. La petición es traducida por una petición de salida, que hará que se abandone la actividad actual, y una de secuenciamiento, con el objetivo de pedir la siguiente actividad. En caso de que la petición de salida indicara que el alumno quiere abandonar la sesión, la petición de secuenciamiento se eliminaría.
8. Se actualiza la información de seguimiento del alumno y finaliza la actividad.
9. El proceso se repite desde el paso 3.

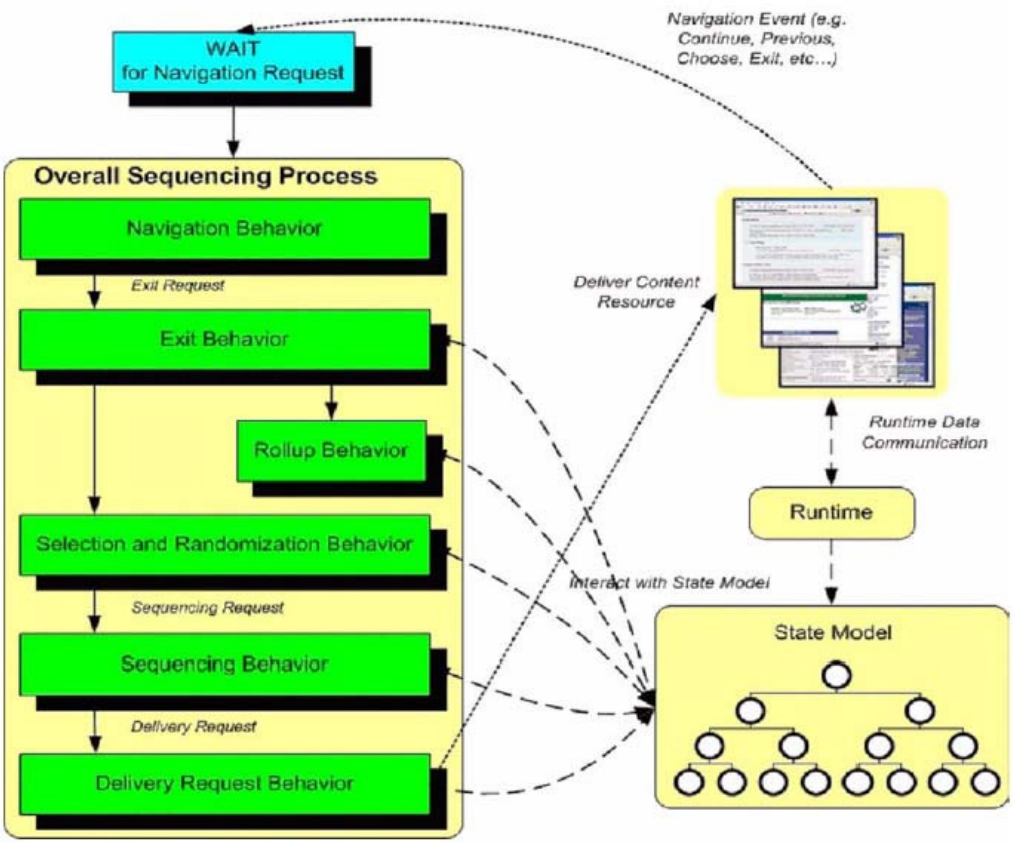


Figura 15. Diferentes pasos en el proceso de secuenciamiento. (IMSSS_INFO 2002)

En la figura 15 se muestra un proceso de secuenciamiento con todas sus partes. En la parte izquierda de la figura se muestra la secuencia de control del proceso. En una operación normal, el proceso pasa primero por el proceso de navegación, después el de salida, el de secuenciamiento, y por último el que muestra la actividad. Después de esto, se produce una espera (wait) hasta que el alumno interacciona con el contenido.

La parte derecha de la figura revela la vista que tiene el alumno del proceso. Los contenidos se muestran al alumno, que interacciona con ellos. El estudiante dispara eventos que ponen en marcha el proceso.

Durante todo el proceso de secuenciamiento, los modelos de estado (árbol de actividad) y el modelo de seguimiento permanece en memoria, por lo que se accede a ellos mediante interfaces en tiempo de ejecución.

¿Qué relación tiene con el resto de las especificaciones?

Esta especificación se encuentra fuertemente acoplada con el resto, pues cualquiera de las especificaciones que tratan con contenidos puede añadir las utilidades que presente IMS Simple Sequencing.

Veamos una a una todas las especificaciones con las que guarda relación, aunque son las dos primeras con las que más grado de conexión presenta:

IMS Content Packaging.

Esta es la especificación con la que más relación tiene. No sólo se aconseja la integración de los elementos definidos en esta especificación dentro del manifiesto, sino que IMS Simple Sequencing, aunque puede aportar información de secuenciamiento a cualquier tipo de elemento, está especialmente orientado a los elementos definidos en IMS CP. Los elementos de esta especificación pueden considerarse como una extensión de Content Packaging que añaden información de secuenciamiento.

IMS Question and Test Interoperability.

Hemos visto que Simple Sequencing define la visualización de actividades de una manera similar a como lo hacía IMS QTI. Ambas especificaciones permiten la elección de ciertas actividades de un conjunto así como la visualización automática de las mismas. Una evaluación definida en QTI se puede considerar una actividad. Así, se podrían utilizar las mejoras incluidas en esta especificación para incrementar las posibilidades de visualización de los exámenes. Aún así, creemos que hace falta pulir la forma de incluir esta especificación en el secuenciamiento de exámenes para que encaje tal y cómo lo hace con IMS CP.

IMS Learning Design.

La definición de las diferentes secuencias que pueden seguir los recursos educativos dentro de un diseño educativo corre a cargo de IMS Simple Sequencing. Para ello, IMS LD tiene un elemento único que indica si se va a utilizar IMS SS para controlar el flujo de la experiencia educativa.

IMS Learner Information Package.

A pesar de no tener mucha relación en la actualidad, queremos incluir esta especificación en la lista porque apostamos a que en un futuro se guarde la información de secuenciamiento en los archivos de personalización de los alumnos.

IMS Reusable Definition of Competency or Educational Objective (RDCEO)

¿Qué es?

Esta especificación define un modelo de información que hace posible describir, referenciar, e intercambiar "competencias" dentro del contexto de aprendizaje distribuido. La palabra "competencias" dentro de esta especificación hace referencia a las habilidades, el conocimiento, las tareas y los resultados educativos de aquellos implicados en el proceso de enseñanza. Por tanto, se ofrece una manera formal de representar cada una de ellas. Este hecho permite un mayor entendimiento entre los diferentes sistemas de enseñanza (ya sean automáticos o humanos), ya que el modelo de información que ofrece se puede utilizar para intercambiar las definiciones descritas.

Nos encontramos por tanto ante la formalización de todas las capacidades que un componente de un sistema de enseñanza puede poseer. La manera de almacenar estas capacidades es la que se muestra en la figura 16.

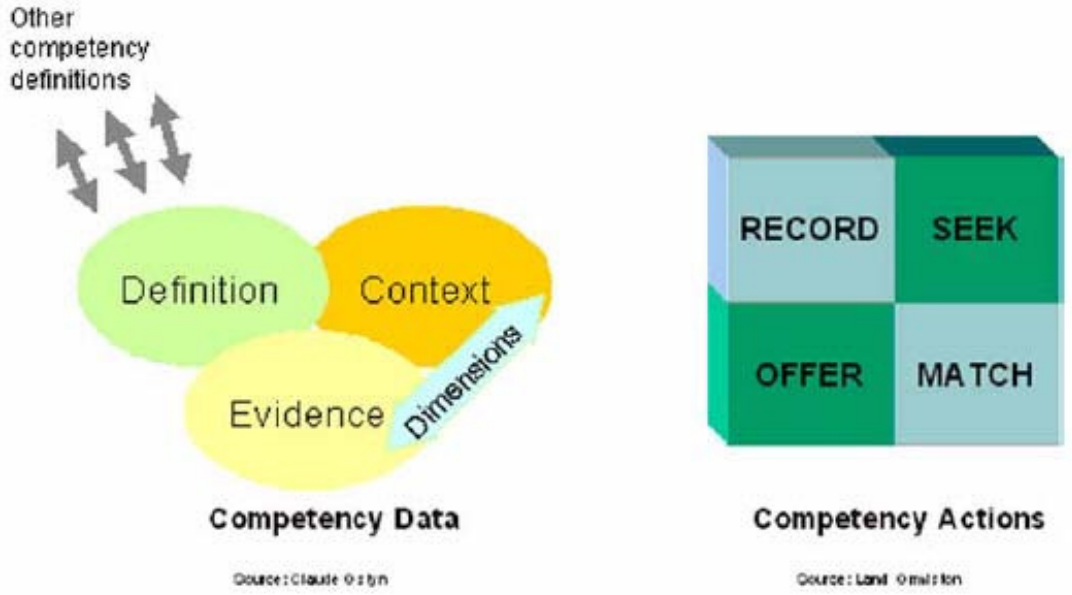


Figura 16. Datos y acciones referentes a una competencia. (IMSRDCEO_BEST 2002)

Esta figura nos muestra los datos que se almacenan referentes a una competencia:
Definición de la competencia.
Contexto o contextos en los que es posible su encuadre.
Diferentes *datos específicos* dentro un contexto determinado (Evidence).
Las *dimensiones* hacen referencia al número de contextos diferentes en los que se puede utilizar una determinada competencia, y las diferentes propiedades que tiene en cada uno de ellos. Por tanto, cada uno de los contextos diferentes posee sus datos específicos.

En la figura 16, también aparecen cuatro ejemplos de acciones posibles sobre las competencias almacenadas: Buscar competencias, almacenarlas, mostrarlas y buscar mediante sus campos. Es importante tener en cuenta que son sólo ejemplos, ya que una vez que se tiene una base de datos de referencias, las acciones posibles son todas aquellas que se pueden llevar a cabo sobre una base de datos cualquiera.

La mayoría de la eficacia de RDCEO descansa en que se puedan referenciar las competencias almacenadas. Para ello, es importante tener una capa de persistencia estable, e identificadores únicos. De hecho, aconsejan que una vez que se ha publicado una competencia con su correspondiente identificador, no se modifique nunca. La

ventaja de esto es poder conseguir almacenes de competencias en dónde sea sencillo realizar búsquedas. A día de hoy, existen almacenes de competencias pero tienen el problema, que se trata de resolver con esta especificación, de no ser intercambiables por utilizar una notación diferente para la representación de las competencias.

¿Qué relación tiene con el resto de las especificaciones?

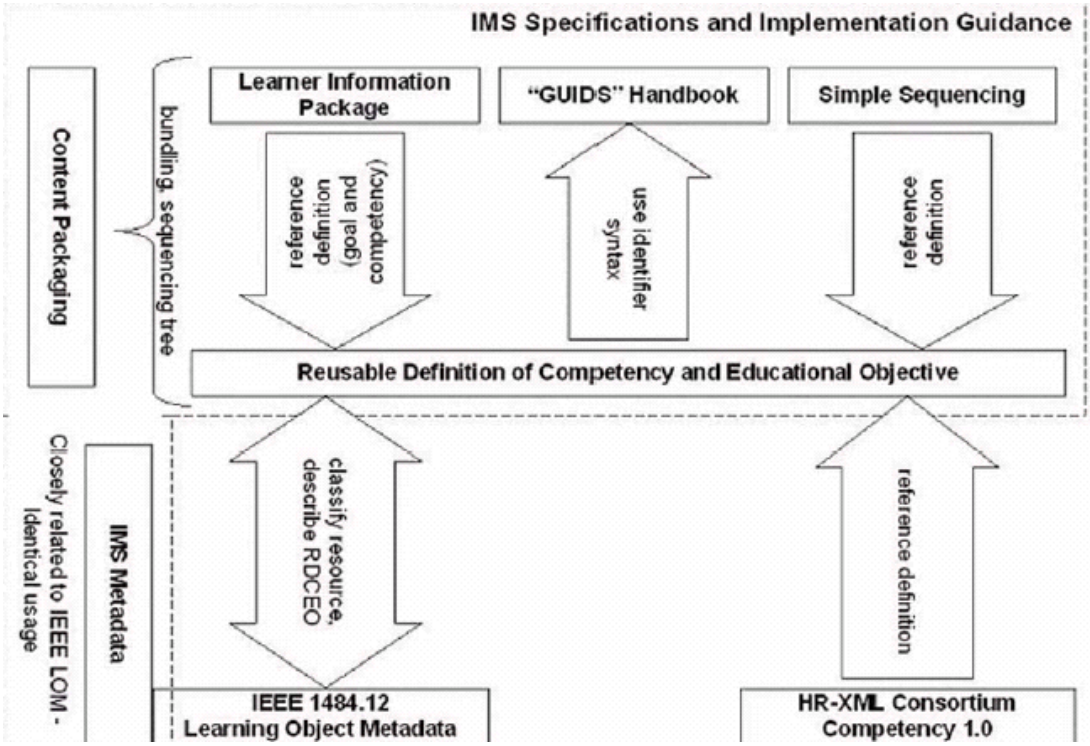


Figura 20. Rol de RDCEO en el entorno IMS y los modelos de datos relacionados. (IMSRDCEO_BEST 2002)

Antes de entrar en detalle con cada una de las especificaciones, hemos incluido un gráfico que muestra el lugar de RDCEO dentro de IMS. La figura 20 se encuentra dividida en dos partes, la primera hace referencia a las etiquetas que se ubicarán dentro del IMS Content Packaging (y por tanto dentro del manifiesto), y las descripciones adicionales de las competencias utilizando Metadatos.

IMS Content Packaging

La principal relación que existe entre estas dos especificaciones es la idea de IMS que consiste en empaquetar definiciones de competencias. Para ello, se ha creado el manual "Using IMS CP to Package Instances of LIP and Other IMS Specifications" dónde se indica la forma de empaquetar competencias u otras especificaciones que hacen referencia a competencias, por ejemplo LIP.

IMS Learner Information Packaging

En RDCEO se ofrecen diferentes formas para describir competencias que pueden ser referenciadas desde LIP. Usando estas dos especificaciones conjuntamente conseguiremos una descripción más completa de las capacidades de los alumnos.

IMS Metadata

Entre estas dos especificaciones existe una relación recíproca. Por un lado, desde un fichero de metadatos se pueden referenciar competencias para añadir riqueza a su significado, y por otro, dentro de la definición de una competencia, hemos visto en el apartado anterior que se puede incluir una sección de metadatos para añadir datos sobre dicha competencia.

IMS Simple Sequencing

Esta especificación ofrece una forma de referenciar competencias mediante un identificador. De esta manera, se rompe la idea de que las secuencias solamente están enfocadas al contenido. Se pueden aplicar las formas de secuenciamiento ofrecidas en IMS SS a los RDCEO's para así conseguir secuencias de competencias.

IMS Digital Repositories Specification (DRS)

IMS DRS no sigue el esquema que viene siendo habitual en el resto de especificaciones. Por eso no la vamos a tratar de la misma manera. En este caso, las dos secciones que venimos exponiendo de forma continua desaparecen para dejar paso a un sólo punto (*¿Que es*?), con sus correspondientes secciones, en el que se explicarán en detalle qué se pretende con esta nueva especificación.

¿Qué es?

El propósito de esta especificación es ofrecer una serie de recomendaciones para conseguir que las funciones propias de los almacenes o repositorios sean comunes a todos los LMS que las cumplan. Estas recomendaciones deberían ser utilizadas en los servicios de los almacenes para que, de esta forma, estos servicios puedan presentar una interfaz común para sus usuarios.

Nos encontramos por tanto, ante la primera especificación que no propone un esquema, sino que pretende ajustarse a los esquemas que ya están descritos en otras especificaciones (IMS Content Packaging o IMS Metadata).

Lo primero que debemos abordar es: ¿Que es un repositorio de recursos educativos?

Según esta especificación, un repositorio es una colección de recursos que son accesibles mediante la red sin que sea necesario un conocimiento previo de su estructura. Además de los componentes recopilados, se contempla un almacenamiento de metadatos que aporten información sobre dichos componentes.

Arquitectura propuesta

En este apartado vamos a mostrar la arquitectura que propone IMS DRS para la creación de un almacén de recursos educativos, que llamaremos *assets*. En la figura 21 se ilustra el espacio donde el e-learning, los repositorios digitales y los servicios de información interactúan, que es exactamente el problema que aborda esta especificación.

En este esquema (figura 21) se pueden observar las tres entidades definidas por esta especificación para abordar el problema (marcadas en verde, azul, y amarillo):

- *Roles.*
- *Componentes para la administración de los recursos.*
- *Servicios.*

Las líneas rojas indican las interacciones entre los principales componentes de la arquitectura, que incluyen:

- *SEARCH, GATHER, ALERT /EXPOSE*
- *REQUEST / DELIVER*
- *SUBMIT / STORE*
- *DELIVER / STORE*

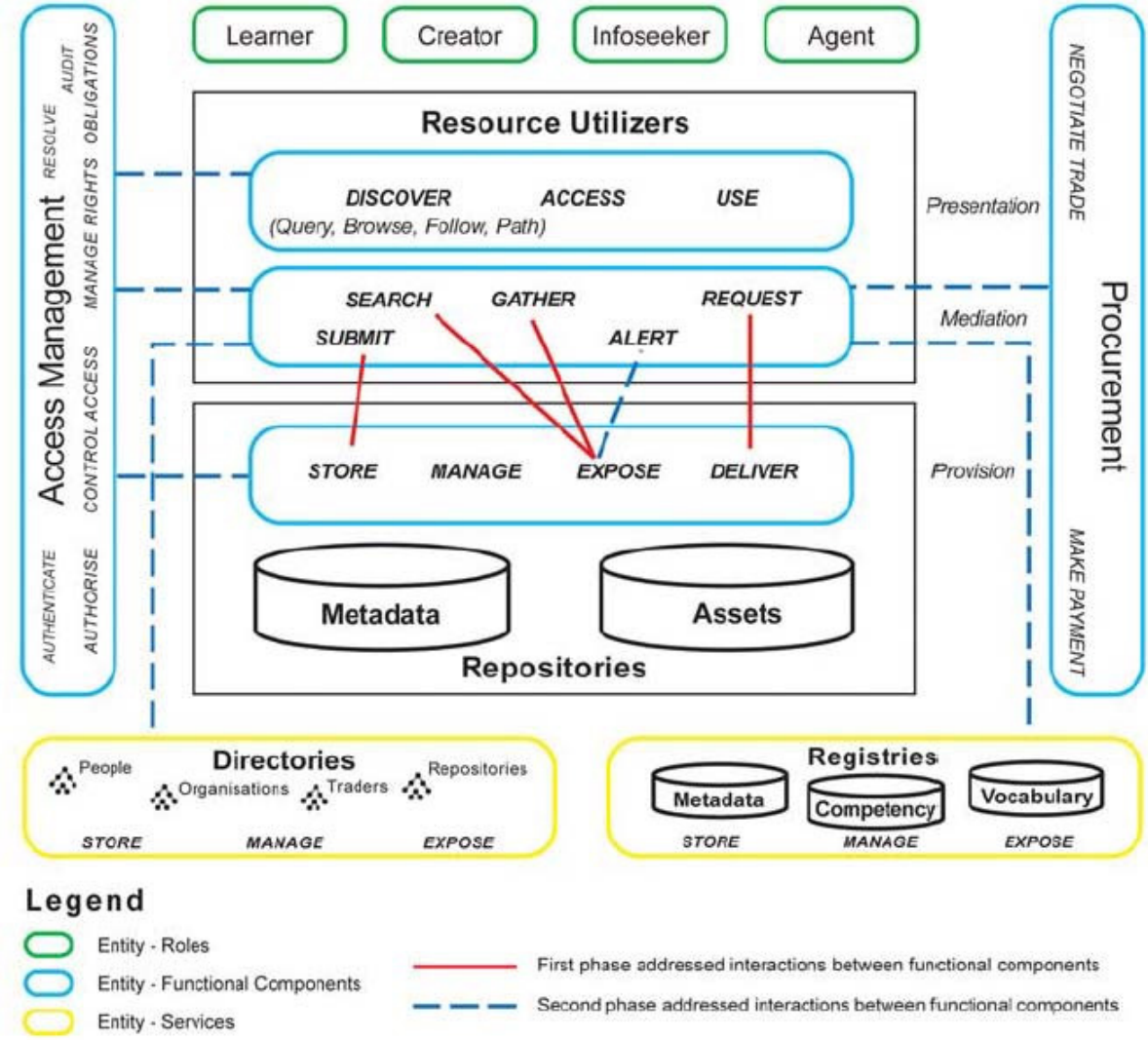


Figura 21. Arquitectura propuesta por IMS RDI para la interacción entre repositorios digitales, LMS's, y servicios de información. (IMSRDI_INFO 2003)

Visto este primer esquema de la propuesta de la especificación, vamos a pasar a explicar más en detalle cada uno de los elementos que se han mostrado. Nos adentramos, por tanto, en el modelo de referencia propuesto.

Modelo de referencia

Tomando como punto de partida la figura 21, ofrecemos un esquema simplificado que incluye las funciones más importantes. En la figura 22 se representan en la parte superior los diferentes roles que pueden acceder al repositorio. A estos tres hay que añadir el rol de Agente, que se encuentra englobado en los servicios de acceso por considerarse que se trata de sistemas de enseñanza que, automáticamente, acceden al almacén.

Veamos los roles que considera esta especificación, que pretenden cubrir todos los tipos de usuarios que pueden verse envueltos en una experiencia educativa.

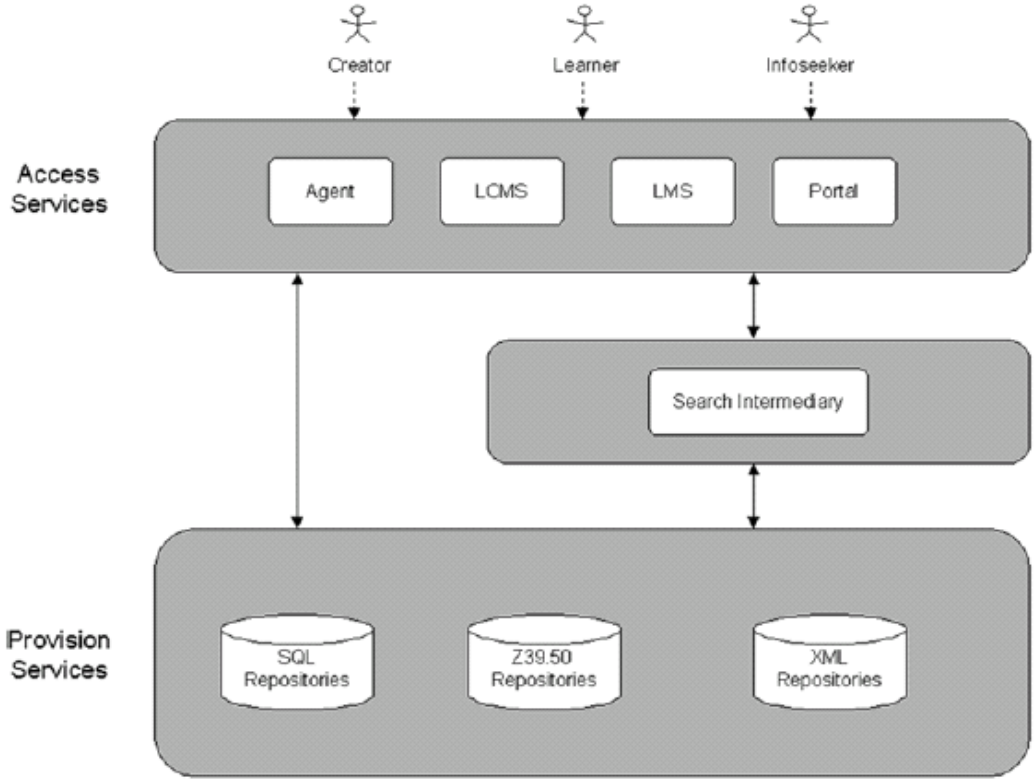


Figura 22. Esquema de acceso a diferentes tipos de repositorios (IMSSS_BEST 2002).

Learner (alumno). Se define como una persona que sigue un curso o que está dentro de un proceso de aprendizaje. Generalmente un alumno usará los diferentes servicios Web que un sistema de enseñanza pueda ofrecer y necesitará los diferentes recursos educativos para poder llevar a cabo su formación. Una vez que un alumno abandona la aplicación de enseñanza, su Rol cambiará al de *Infoseeker* (buscador de información).
Creator (creador). Es una persona encargada de la creación de los objetos educativos, de los cursos, del secuenciamiento de aprendizaje, o de los programas de las asignaturas.
Infoseeker (buscador de información). Este rol lo componen aquellas personas que buscan información en los repositorios a través de los diferentes servicios de búsqueda. Tanto un alumno como un creador pasan a ser buscadores de información en el momento que buscan dentro de los diferentes recursos de aprendizaje que hay dentro de un almacén. Es importante tener en cuenta que aquellos que se engloban dentro de este rol no tienen por qué estar dentro del proceso de aprendizaje.
Agent (agente). Es una aplicación inteligente capaz de emular el comportamiento de cualquiera de los otros roles. Una vez realizada su tarea, un agente procesará los resultados obtenidos bien de manera automática, bien mediante la intervención de alumnos, creadores o buscadores de información.

Gracias a esta especificación, se consigue acceder a los contenidos almacenados en repositorios desde sistemas de enseñanza (LMS), sistemas de gestión de contenidos educativos (LCMS), portales de búsqueda de contenidos, y desde cualquier agente de software que sea compatible con IMS RDI.

Aunque esto pueda parecer una solución sencilla, el problema comienza cuando se tienen que realizar búsquedas sobre los contenidos almacenados. Pensemos en la gran heterogeneidad de contenidos que se pueden encontrar acumulados. Aunque la utilización de meta datos trata de mitigarlo, la especificación introduce un componente intermedio que podría implementar las siguientes funciones para facilitar las búsquedas:
Un **traductor**, que sea capaz de traducir una petición de búsqueda de un formato a otro para hacerla así comprensible a todos los repositorios.

Un **convertidor**, encargado de hacer que todos los meta datos incluidos en cada *learning object* sean útiles para facilitar las búsquedas.
Un **encargado** de pasar una petición de búsqueda a los diferentes almacenes y de gestionar las respuestas.

Veamos las diferentes funcionalidades que se proponen para la gestión completa de un repositorio.

Search/Expose

Esta funcionalidad hace referencia a la búsqueda de meta datos asociados con los contenidos almacenados en los repositorios. En la figura 23.

Nos encontramos ante un dominio muy extenso y heterogéneo en cuanto a las diferentes tecnologías de almacenamiento que existen. Para solventar esto, se propone una capa intermedia encargada de mediar entre las peticiones y las diferentes formas en que se encuentran almacenadas las respuestas.

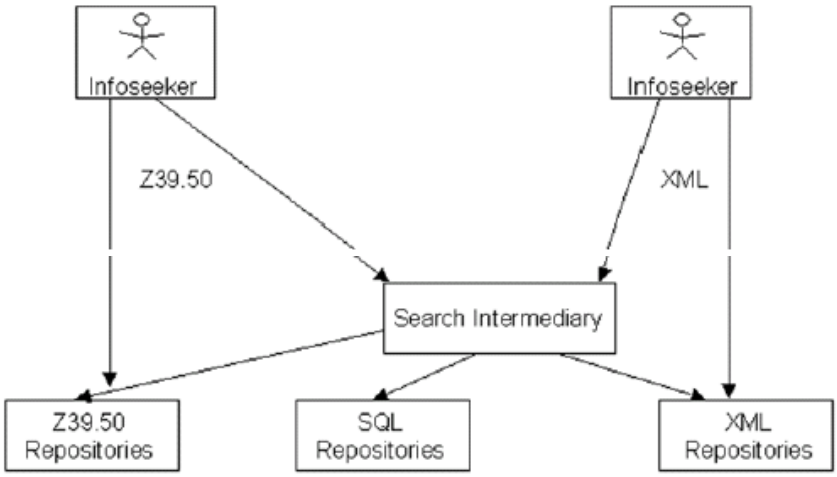


Figura 23. Dominio de búsqueda en diferentes almacenes.

Gather/Expose

Esta funcionalidad proporciona la forma de escribir los meta datos que van a servir para las búsquedas, la forma de agruparlos para facilitar los sondeos futuros y la manera en que se tienen que agregar para formar nuevos repositorios (estos almacenes estarán disponibles para las funciones de búsqueda y alerta).

Esta funcionalidad interactúa con el repositorio de dos maneras diferentes. La primera consiste en solicitar meta datos del repositorio *(pul*l), mientras que en la segunda ofrece al almacén meta datos para que sean almacenados *(pus*h). La figura 24 muestra el modelo de referencia de la funcionalidad *Gather* para una solicitud de meta datos.

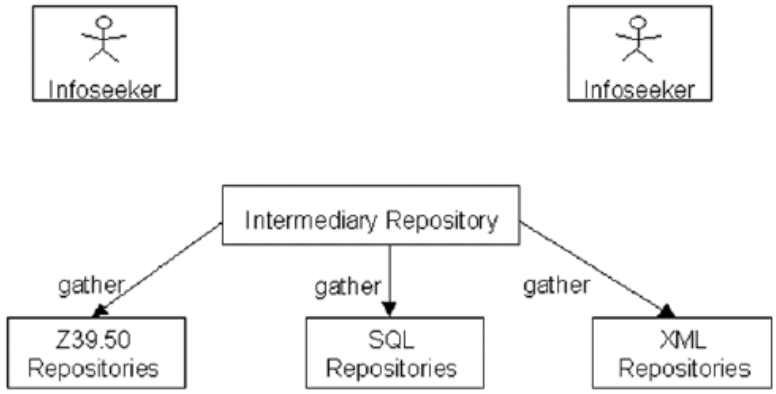


Figura 24. Modelo de referencia para la solicitud de meta datos

Alert/Expose

La especificación RDI contempla esta funcionalidad como un posible componente de un repositorio digital o un servicio intermedio encargado de mandar correos electrónicos. Esta funcionalidad se plantea como un trabajo futuro y no se encuentra especificada al día de hoy, por lo que no ahondaremos más en la misma.

Submit/Store

Esta funcionalidad hace referencia a la forma de almacenar un objeto en un almacén y la forma que tomará una vez almacenado para hacer posible su recuperación. El lugar desde el cual se toma el objeto para su almacenamiento puede ser otro repositorio, un sistema enseñanza, el disco duro del desarrollador, o cualquier punto de la red.

Generalmente, la recuperación y almacenamiento de objetos serán llevados a cabo mediante FTP, por lo que esta especificación tiene las siguientes propuestas para evitar las debilidades de esta forma de comunicación:

Añadir al protocolo mecanismos de encriptación para asegurar la integridad de los contenidos.

No ofrecer acceso directo al servidor mediante FTP por los problemas de seguridad que esto conlleva.

Tener en cuenta que FTP no ofrece ningún mecanismo que asegure que un objeto ha sido copiado en su totalidad de un punto de la red al almacén.

Dentro de esta funcionalidad hay que tener muy en cuenta las recomendaciones propuestas en IMS CP. Esta especificación dice: “La interoperabilidad de los sistemas implica que puedan importar, exportar, agregar y desagregar paquetes de contenidos” [IMSCP_INFO 2001]. Para ello se propone utilizar un paquete en formato comprimido que contiene el objeto educativo, su registro de meta datos, y un manifiesto que describe los contenidos del paquete. Por tanto, si se quiere mantener la coherencia en las diferentes especificaciones, se tiene que utilizar este formato para intercambiar contenidos entre los sistemas educativos y los almacenes.

La función de recuperación de objetos educativos debe cumplir que los paquetes se envíen utilizando mensajes SOAP con ficheros adjuntos que tendrán el formato que Content Packaging propone.

La función de almacenamiento de objetos educativos permitirá que dichos objetos se guarden según el formato propuesto por IMS CP.

A continuación se presenta la figura 25 que muestra la diferencia en la comunicación entre almacenes que cumplen esta especificación y aquellos que no la cumplen.

*NOT*A: La notación DRI se refiere a Digital Repositories Interoperability, que es análoga a la notación DRS que venimos tratando durante toda la especificación.

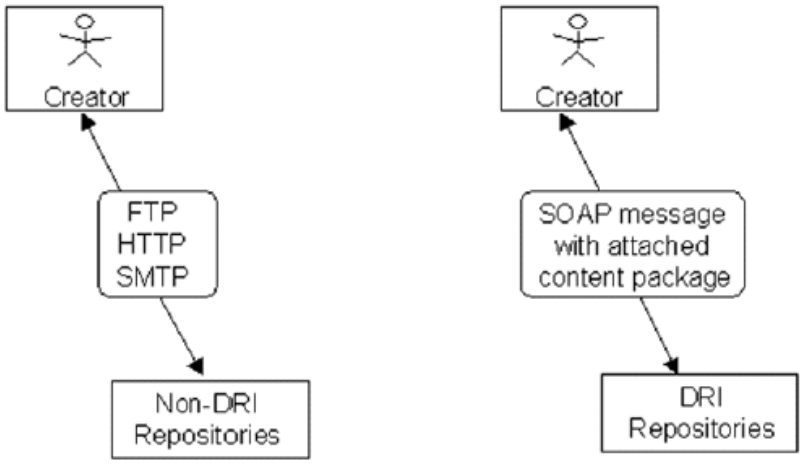


Figura 25. Diagrama de las funcionalidades de almacenamiento y recuperación de almacenes educativos.

Request/Deliver

La función Request es la petición de acceso a un recurso que realiza un usuario del sistema una vez que lo ha localizado gracias a los meta datos que lleva asociados. Deliver se refiere a la respuesta que le da el repositorio, que le otorga o le niega el acceso al recurso.

Conclusiones

Esta especificación se compone de una serie de recomendaciones enfocadas a facilitar la creación de repositorios o almacenes de objetos educativos. Igual que en el resto de las especificaciones propuestas, el éxito de la misma depende del grado de aceptación y utilización que consiga.

Se ha realizado un estudio detallado de los problemas con que se enfrenta el creador de repositorios y se han propuesto tecnologías capaces de resolverlos. Estas no son las únicas tecnologías capaces de afrontar estos problemas, pero sí las más extendidas.

Ya que cuantos más repositorios las utilicen más valor adquirirá la especificación, es importante que las tecnologías resulten familiares para quienes creen el almacén.

Conclusión de IMS

A diferencia de lo visto en el análisis de Dublín Core y LOM, ya no estamos frente a un conjunto de elementos que permiten metaanotar recursos, sino frente a un conjunto de especificaciones que comprenden, prácticamente en su totalidad, los requisitos de la tecnología e-learning. Esto se logra a través de especificaciones que se centran en diferentes campos, permitiendo por un lado una gran cobertura de los aspectos centrales de e-learning, así como también la posibilidad de adoptar determinadas especificaciones de acuerdo a las necesidades concretas de una comunidad, sin tener que adoptar el conjunto total de ellas.

IMS está permitiendo que los diferentes LMS empiecen a hablar entre sí, posibilitando que la portabilidad de contenidos entre diferentes plataformas sea una realidad.

El continuo desarrollo y la corta edad de IMS y sus especificaciones se traduce en que la incorporación que han hecho las herramientas para poder soportar las reglas IMS se transforme en algo de constante cambio, y que aún requiere disponer de personas con un perfil calificado para lograr hacer funcional la ejecución de IMS en las plataformas. Muchas veces es necesario trabajar directamente sobre los archivos XML y modificar algunos de sus valores antes de poder mover contenidos entre las plataformas e incluso dentro de una misma plataforma. Por lo tanto, si bien IMS es algo que ya está funcionando, aún carece de la solidez de un estándar y una facilidad para ser manipulada por usuarios finales.

IMS ocupará un lugar importante dentro del futuro de las plataformas, por lo tanto, trabajar y conocer estas especificaciones al momento de tomar decisiones sobre las herramientas y recursos e-learning pueden ser vitales para asegurar la continuidad y permanencia de un proyecto e-learning en el tiempo.

Algo importante para observar es cómo los distintos estándares utilizan y se apoyan en los esfuerzos logrados por otras organizaciones. En este caso IMS, utiliza y participa del proceso de estandarización del LOM.

Por otro lado, veremos más adelante que ADL-SCORM utilizó los esfuerzos realizados por IMS.

SCORM (1999)

Introducción

En Noviembre de 1997 el Departamento de Defensa de EE.UU. y la oficina de Ciencia y Tecnología de la Casa Blanca lanzaron la iniciativa Advanced Distributed Learning [ADL 2002]. ADL surge como respuesta a las necesidades de uno de los mayores consumidores de software del mundo y forma parte del esfuerzo que el gobierno norteamericano viene realizando con el objetivo de conseguir una enseñanza de calidad, en el que también están implicados los departamentos de Educación y Trabajo.

Alcance

El ADL se ha centrado desde el principio en el aprendizaje sobre la Web y ha recogido "lo mejor" de las iniciativas más relevantes (el sistema de descripción de cursos en XML de IMS, y el mecanismo de intercambio de información mediante una API de AICC, que se verá en este mismo capítulo), mejorándolas para crear su propia iniciativa: SCORM, Shareable Courseware Object Reference Model (Modelo de Referencia para Objetos de Cursos Compartidos) [SCORM 2001], que propone un entorno de ejecución, un modelo de metadatos y un modelo de la estructura de los cursos, y cuya primera versión data de principios de 1999.

A partir de Enero del 2000, cambia el significado de las siglas SCORM, que pasan a significar Shareable **Content** Object Reference Model (Modelo de Referencia para Objetos de **Contenidos** Compartidos). Esto se debió a la intención de mostrar que SCORM se aplicaría a los contenidos en lugar de a los cursos. De esta manera, SCORM pasaba a referirse a las partes que componían los cursos, por lo que aumentaba su generalidad.

Con la versión 1.2 de SCORM, se incluyó una aplicación de empaquetamiento de contenidos derivada directamente del IMS Content Packaging, que permitía cambiar del antiguo formato de contenidos a las especificaciones de IMS. Esto supuso el comienzo del trabajo conjunto que realizan estas organizaciones (IMS y ADL), y que aún continúa al día de hoy. En esta versión se engloba el trabajo de organizaciones como IBM, AICC, IMS, e IEEE.

Scorm es un conjunto de especificaciones para desarrollar, empaquetar y distribuir educación de alta calidad y material de enseñanza siempre que y donde sean requeridos. [Jones 2002]

SCORM proporciona un marco de trabajo y una referencia de implementación detallada que permite a los contenidos y a los sistemas que utilicen SCORM "hablar" con otros sistemas, logrando así interoperabilidad, reusabilidad y adaptabilidad [Foix C. 2002].

En la figura 26 mostramos un gráfico de la evolución de SCORM desde su fecha de creación.

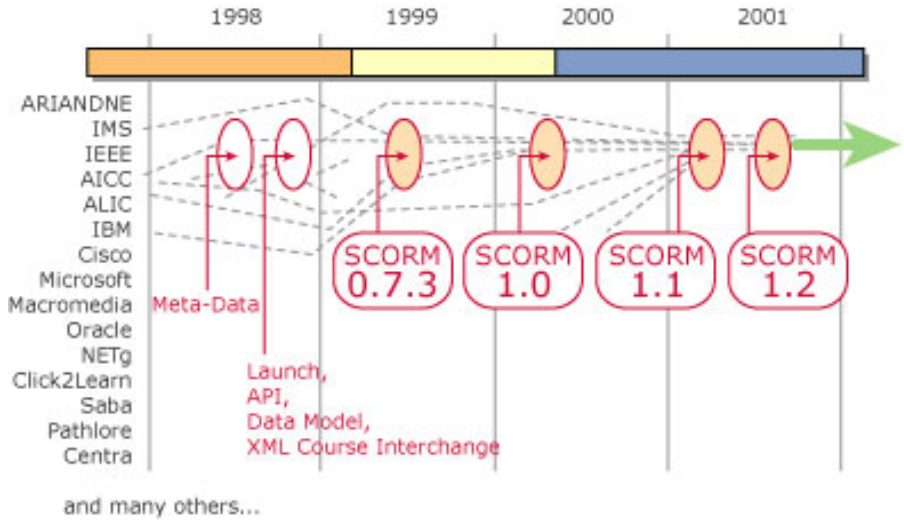


Figura 26. Gráfico de convergencia de ADL/SCORM (ADL 2001a)

Recomendaciones y propuestas en ADL

Las principales aportaciones de SCORM han sido la definición de un Modelo de Agregación de Contenidos y de un Entorno de ejecución para los objetos educativos.

Además, SCORM se puede ver como un modelo de referencia que incluye una serie de especificaciones y de guías que contienen los requisitos imprescindibles que deben cumplir los contenidos educativos. Estos requisitos están orientados a facilitar la reusabilidad, accesibilidad, interoperabilidad y durabilidad de aquellos contenidos que los cumplan [ADL 2001a].

SCORM reúne los esfuerzos de diferentes organizaciones para llegar a un modelo general que puede ser tomado como referencia para la creación de sistemas de aprendizaje. En el siguiente gráfico se muestra esta labor de integración y recolección, pues podemos ver de qué organización ha tomado cada una de las aportaciones que propone además de mostrarnos las diferentes partes en las que se componen las especificaciones de ADL.

En la figura 27 vemos las organizaciones que ADL ha integrado.
Veamos cuales son los principales campos educativos en los que SCORM está implicado:

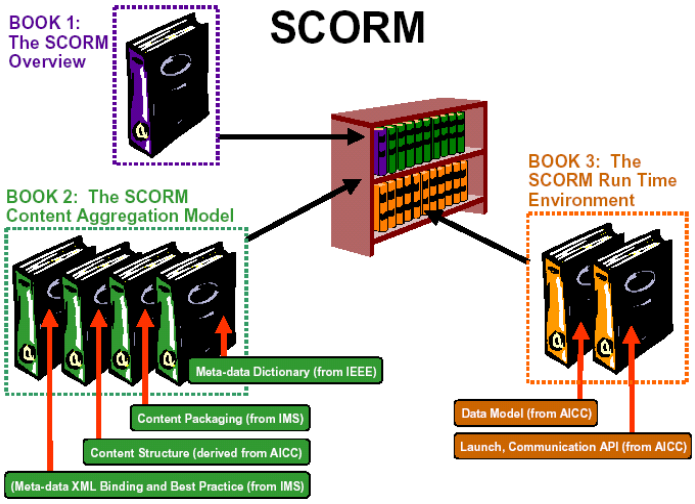


Figura 27. Composición de las especificaciones de SCORM y sus fuentes (ADL 2001a)

Entorno de ejecución (Run Time Environment). Definición de un entorno de ejecución que incluye: un protocolo específico para la ejecución de contenidos Web, un API (Aplication Programming Interface) entre el contenido y el LMS y un modelo de datos que define el flujo de datos intercambiado entre el entorno LMS y el contenido que se ejecuta en el entorno de ejecución [Sancho 2002]. Todo basado en el trabajo realizado por AICC [AICC 2002].
Modelo para el tratamiento de contenidos (Content Aggregation Model). Incluye:

- Definición de una representación mediante XML de la estructura de los cursos que puede utilizarse para definir todos los elementos de un curso (Content Structure derivado de AICC).
- Definición de las relaciones jerárquicas entre estos cursos y las referencias externas requeridas para intercambiar un curso de entorno LMS a otro. Basado en IMS Content Packaging.

Respecto a los metadatos, SCORM se apoya en la especificación 1.0 de IMS cuando se implementan los elementos LOM en XML, por lo que ambas especificaciones son compatibles. Cuando las especificaciones de IMS se refieren a metadatos definen un esquema junto con ejemplos de implementación en XML, pero no indican cómo se deben aplicar estas definiciones a los diferentes tipos de recursos educativos. SCORM hace una serie de recomendaciones en el uso de los metadatos aplicados a estos recursos.

SCORM ha adoptado un subconjunto de los metadatos descritos en el modelo de información de metadatos para recursos educativos de IMS que a su vez está basado en LOM. También hace referencia a la especificación obligatoria en XML de IMS para validar sus implementaciones. SCORM aplica las definiciones de metadatos para describir los tres componentes básicos de su modelo, que explicamos a continuación: elementos media *(asset*s), contenidos (SCOs) y cursos. De este modo establecen la relación entre una especificación general y un modelo específico propio.

A continuación, antes de entrar en detalle en el modelo de tratamiento de contenidos y el entorno de ejecución propuesto por SCORM, veremos algunos términos acuñados por esta iniciativa que nos servirán para su posterior análisis:

SCO *(Sharable Content Object)*. Es la mínima colección de contenidos capaz de comunicarse con el LMS. Dentro del contexto de SCORM, un SCO se considera un objeto con plena capacidad, por lo que además de ser reutilizable (lo que lo hace independiente del contexto de aprendizaje), la navegación sólo es posible dentro del mismo y no es posible acceder a ningún contenido que no se encuentre definido en su interior. Un SCO necesita de un LCMS (Sistema de gestión de contenidos educativos) para su acceso, y posee información referente a su ejecución. Está compuesto por Assets (figura 28).

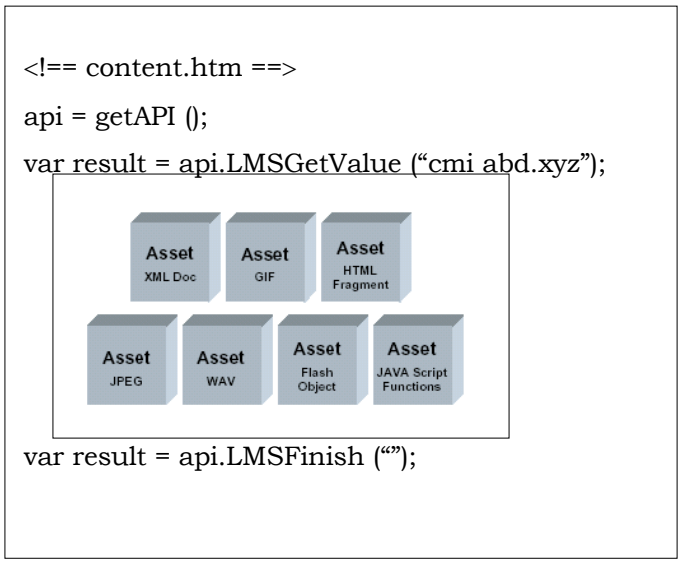


Figura 28. Composición de un SCO.

Asset. Es la parte más pequeña de un contenido educativo (una página HTML, por ejemplo). Los assets no pueden llamarse desde fuera del SCO al que pertenecen. No necesitan comunicarse con el Entorno de ejecución ya que de eso se encarga su SCO (figura 29).

SCA (Sharable Content Assets). Este es un nuevo término que se define en la versión 1.3 de SCORM para identificar a ciertos SCO´s que no poseen información de ejecución.

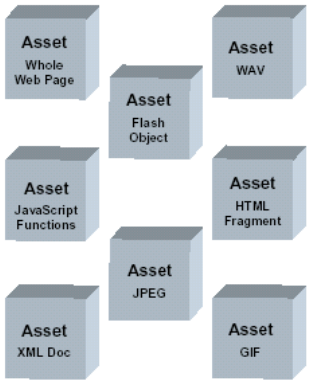


Figura 29. Diferentes ejemplos de Assets (ADL 2001c)

Modelo para el tratamiento de contenidos

Dentro de este modelo, SCORM trata tanto los Metadatos como la forma de estructurar los contenidos educativos. Ya que los metadatos fueron tratados en detalle en el capítulo anterior, y que aquí se utiliza la misma especificación adaptada a los contenidos especiales de SCORM, no los analizaremos en este trabajo, y sólo nos centraremos en la propuesta de estructuración de contenidos.

El objetivo del modelo de agregación de contenidos de SCORM es proveer un medio común de componer contenidos educativos desde diversas fuentes compartibles y reusables. Define cómo un contenido educativo puede ser identificado, descrito y agregado dentro de un curso o una parte de un curso, y cómo puede ser compartido por diversos LMS o por diversos repositorios [Foix C. 2002].

El proceso de diseño y creación de un curso comprende la construcción de un conjunto de objetos de contenidos educativos, relacionados entre sí mediante cierta estructura. Para facilitar esto, se ha definido un formato llamado "Content Structure Format" (CSF), cuyo objetivo es proporcionar un medio de agregación de bloques de contenidos, aplicando una estructura y asociándola a una taxonomía para que tengan una representación y un comportamiento común en cualquier LMS.

Esta propuesta de formato está basada en el modelo de AICC, pero se ha especificado en XML para facilitar su integración en entornos Web, y se ha ampliado para incluir características adicionales tales como la posibilidad de referenciar registros de metadatos del tipo IMS. Además, en la versión 1.2 de SCORM se eliminaron algunos de los registros por considerarlos de poca utilidad. A continuación mostramos la representación gráfica de la DTD del CSF (Figura 29.a) en la que se distinguen dos bloques principales de información [Sancho 2002]:

- **GlobalProperties**. Engloba los datos de carácter global relativos al curso.
- **Block**. Define la estructura del curso. Para realizar la secuenciación de eventos que se llevarán a cabo durante el curso, se utilizarán los elementos *prerequisites* y *completionReq*, que constituyen el mecanismo mediante el cual los contenidos informan al LMS la interacción deseada con el alumno.

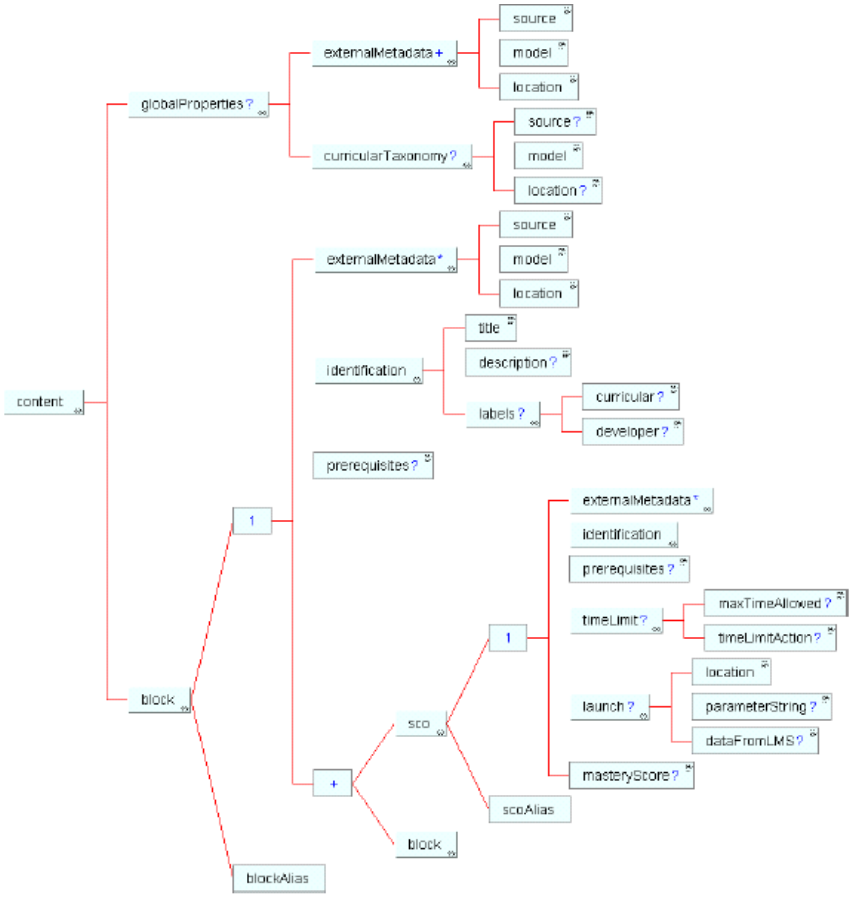


Figura 29.a. Esquema XML del CSF. (ADL 2001c)

El CSF es el formato necesario para mover un contenido educativo de un lugar a otro, pero no es suficiente por sí mismo. Es necesario agregar y guardar los contenidos en un paquete. En versiones anteriores, se utilizaba un método propio definido en el propio CSF para realizar esta función, pero en la última versión se ha decidido encargárselo al IMS Content Packaging. De esta manera, se introduce en SCORM el concepto de *ite*m, y tanto los SCO´s como los Assets se introducen como recursos educativos dentro de los *items* del Content Packaging. Así, el elemento *content* definido en CSF se asigna directamente a un elemento *organization* en el manifiesto asegurando así su compatibilidad. Lo mostramos en la figura 30. Se ha incluido el esquema del CSF en el Apéndice 3 de este trabajo para su consulta.

La información de secuenciamiento almacenada dentro del elemento block queda dentro del elemento organization del imsmanifest.xml, y es aquí dónde el LMS debe buscarla para ofrecer un secuenciamiento correcto de los contenidos.

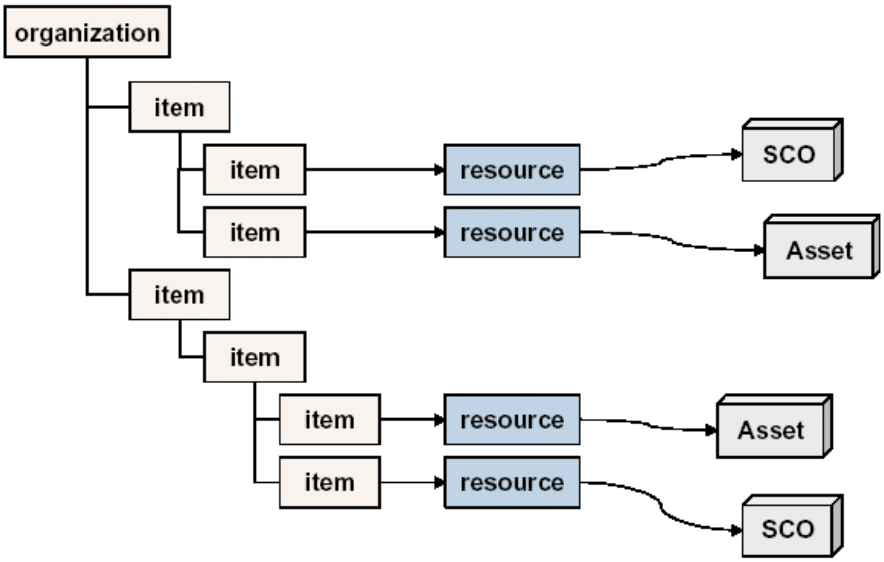


Figura 30. Integración de los elementos de SCORM dentro del manifiesto de IMS (ADL 2001c)

Propuesta de entorno de ejecución

Una de las claves para la reutilización de contenidos educativos entre sistemas diferentes es la separación entre los contenidos en sí y el sistema software que se encarga de gestionarlos (entorno de ejecución). Las responsabilidades más destacadas de este último son la entrega al alumno de un contenido educativo, el seguimiento de la interacción entre el alumno y el contenido y la toma de decisión del siguiente contenido a entregar basándose en la definición estática y dinámica del curso y la actuación previa del estudiante [Anido-Rifón 2001].

La propuesta de SCORM llama LMS (Learning Management System) al entorno de ejecución capaz de mantener información sobre el usuario, de lanzar Objetos Educativos y comunicarse con ellos, y de interpretar instrucciones sobre secuenciación entre objetos. Por otro lado, como ya hemos visto, SCORM se refiere con el término SCO a los contenidos.

La comunicación entre el LMS y los contenidos educativos se realiza mediante una interfaz que estandariza los protocolos de comunicación proporcionando métodos para que el LMS pueda conseguir el estado actual (inicializado, finalizado, etc.) de los contenidos y para el envío de datos entre ambos.

La definición de los entornos de ejecución ha evolucionado en diferentes fases a lo largo de los últimos años. Inicialmente, cuando los sistemas de enseñanza eran independientes, el AICC definió una interfaz de comunicación basada en ficheros sobre el sistema operativo MS-DOS. Más tarde, en colaboración con la iniciativa ADL, la interfaz basada en ficheros fue sustituida por una basada en el protocolo HTTP.

Esta nueva interfaz estaba claramente orientada a redes TCP/IP, por lo que permitía la interrelación entre las computadoras implicadas en el proceso de enseñanza.

La utilización de una interfaz (API, Application Program Interface) proporciona una forma estandarizada para que los contenidos se comuniquen con el LMS, aunque la implementación de esta comunicación es transparente para el desarrollador de los contenidos. La forma en la que esté implementada la API no es importante para los desarrolladores, pero todos deben usar la interfaz externa que ofrece la funcionalidad de la API. Ésta esconde los detalles de implementación a los contenidos permitiendo con ello la reutilización e interoperabilidad de los mismos.

Comunicación SCO-LMS

El adaptador de la API (API Adapter) es el software que expone las diferentes funciones de la API. Este adaptador hace posible la comunicación entre los SCO´s y el LMS, pues es el medio que se les ofrece a los SCO´s para conversar con el LMS. En la figura 31 ofrecemos un esquema de lo que sería esta comunicación.

Con este adaptador se pretende conseguir una independencia entre los contenidos y el entorno de ejecución. De esta manera, los encargados de la implementación de entornos de enseñanza, no tienen más que tener en cuenta las funciones que ofrece la API, y los creadores de contenidos no se tienen que preocupar por el tipo de sistema en el que serán mostrados o el tipo de plataforma. La aparición de este adaptador supone la separación total entre contenidos y sistemas.

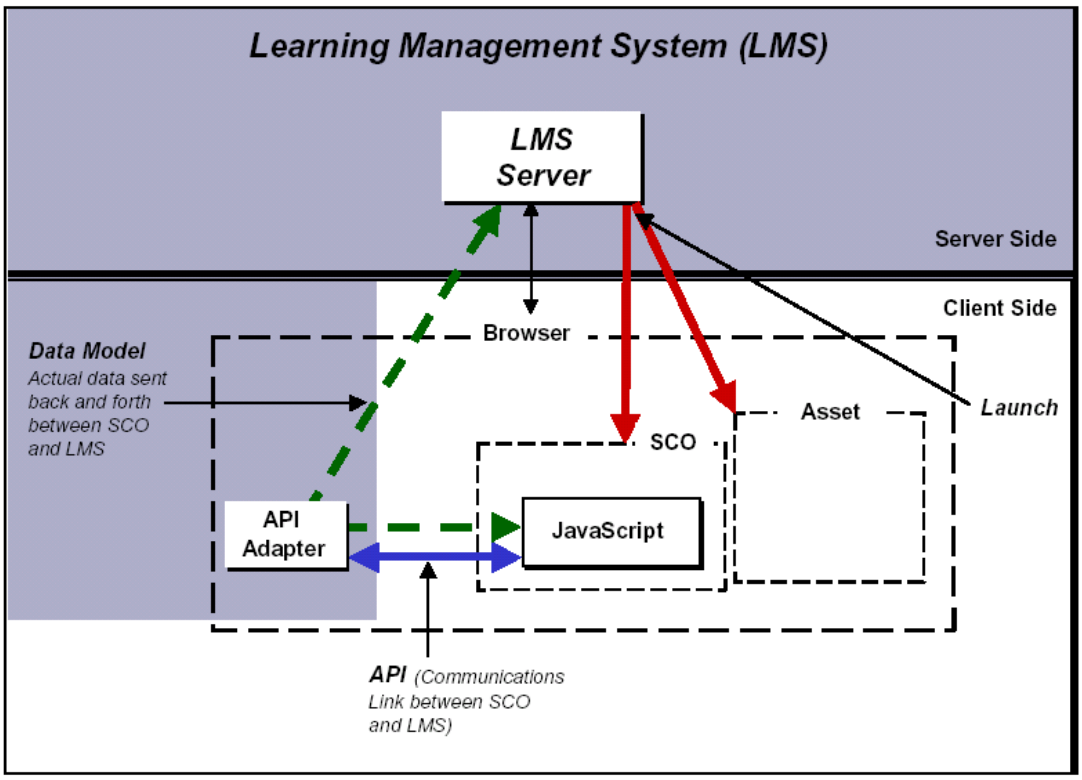


Figura 31. Esquema de comunicación entre los SCO´s y el LMS mediante API (ADL 2001b)

Además, este adaptador posee distintas funcionalidades, que han sido fruto de la experiencia de las instituciones que ha integrado ADL en el desarrollo de sistemas educativos. Veamos a modo de ejemplo algunas de ellas:

- Estado de la ejecución. Dos funciones de la API, *LMSInitialize*("") y *LMSFinalize*("") gestionan la inicialización y finalización de un SCO.
- Gestión del estado. La API tiene tres métodos para el manejo de errores: *LMSGetLastError*(""), *LMSGetErrorNumber*(*errornumber*), *LMSGetDiagnostic*(*parameter*).
- Transferencia de datos. El resto de las funciones de la API se utilizan para la transferencia de datos desde y hacia el LMS: *LMSGetValue(data model element), LMSSetValue(data model element, value)* y *LMSCommit*("").

Proceso de empaquetamiento de los SCO´s

Los SCOs son unidades auto contenidas de aprendizaje, lo que significa que pueden ser utilizadas para constituir paquetes de objetos superiores pero no pueden disgregarse en unidades más pequeñas [Sancho 2002]. Para la creación de un paquete de SCO´s, son necesarios los siguientes pasos:

- Localizar los objetos y organizarlos en una estructura.
- Adjuntar instrucciones para informar al LMS cuál es la secuencia entre los objetos utilizando las instrucciones del API Adapter visto en el apartado anterior. Hay que recalcar que las instrucciones indican cómo moverse de objeto a objeto, pero no se indica cómo hacerlo dentro de un mismo SCO.
- Los objetos e instrucciones deben constituirse en un paquete portable. A este proceso se le conoce como "empaquetamiento de contenidos".

Una vez creados los paquetes, estos se cargarán en el LMS, que será el encargado de mostrarlos de acuerdo con la información que adjuntan. Cuando el alumno termina de visualizar uno de los SCO´s, este se comunica con el LMS, que decide cuál es el siguiente SCO que debe mostrar al alumno basándose en las Instrucciones de distribución que vienen en el paquete.

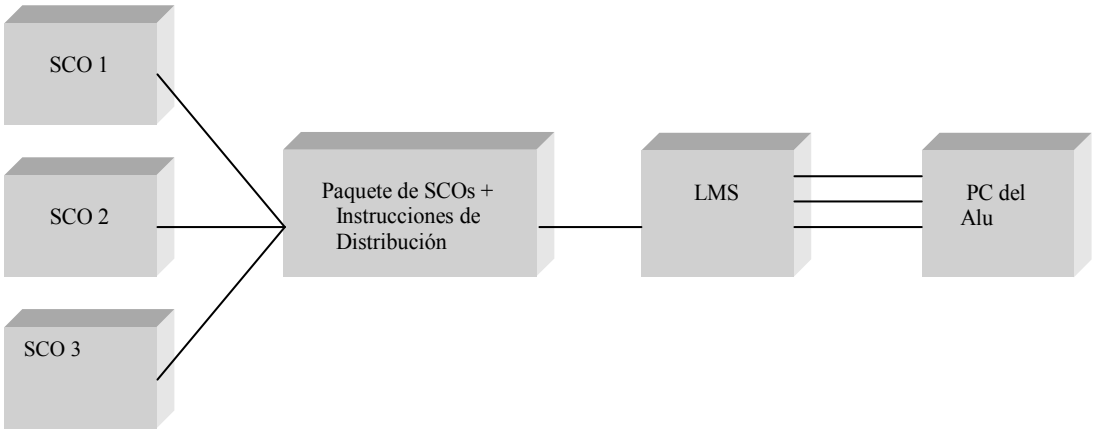


Figura 32. Modelo SCORM (Eduworks 2002)

Versión 2004

La versión 2004, lanzada en enero de 2004 introdujo un cuarto libro: Sequencing and Navigation. Esta nueva versión usó los otros tres libros para seguir la evolución de los estándares y los armonizó con el libro Sequencing and Navigation Con esta versión el documento de SCORM se considera estable.

Con el lanzamiento de SCORM 2004, el ADL ha decidido cambiar las versiones de SCORM para poder mantener cada libro independientemente.

El número de especificaciones y el tamaño de los documentos han hecho necesario este cambio para manejar las revisiones y correcciones del conjunto de documentos.

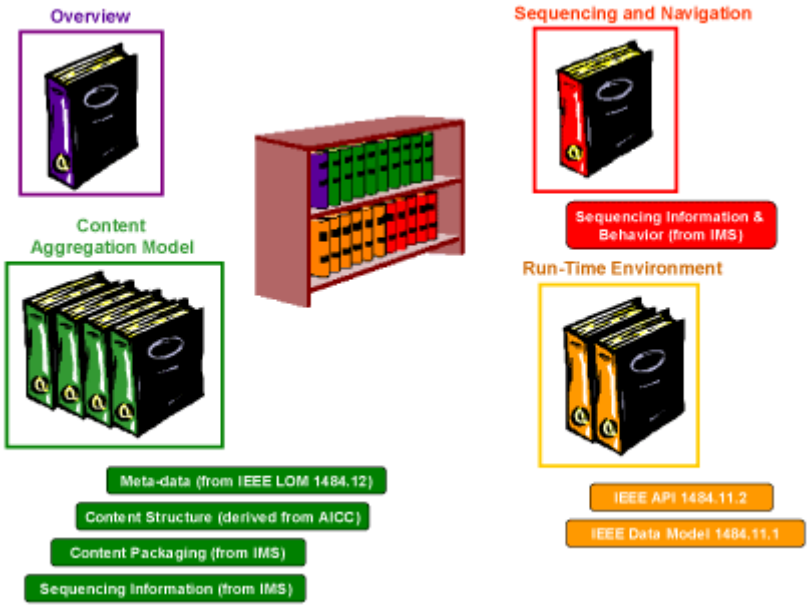
Cada uno de los libros de SCORM ahora lleva su propia versión comenzando con este lanzamiento en la "versión 1.3." Los cambios en el futuro se aplicarán solamente al libro afectado.

SCORM 2004 abarca cuatro libros separados:

- SCORM Overview: Cubre la historia y los objetivos de la iniciativa de ADL y de SCORM, incluyendo las especificaciones y los estándares utilizados por SCORM.
- SCORM Content Aggregation Model (CAM): Describe los componentes usados en una experiencia de aprendizaje, cómo empaquetar esos componentes para el intercambio de sistema a sistema, cómo describir esos componentes para permitir la búsqueda y el descubrimiento.
- SCORM Run-Time Environment (RTE): Describe los requisitos del sistema de gerencia de aprendizaje (LMS) para manejar el ambiente run-time. El libro también cubre el protocolo de comunicación entre un LMS y SCOs.
- SCORM Sequencing and Navigation (SN): Describe cómo el contenido de SCORM se puede ordenar a través de un sistema de acontecimientos

Lista de especificaciones y estándares incorporados en SCORM 2004:

- 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



Conclusión

Concluimos que una de las grandes ventajas del modelo SCORM es que el contenido puede comunicar información sobre el alumno a cualquier sistema LMS utilizando un método estándar basado en JavaScript (la API Adapter está basada en JavaScript). La especificación SCORM determina exactamente cuales son las piezas que se pueden recuperar y actualizar. Esta información incluye: identificación del alumno, nombre, puntuación en tests, tiempo empleado en el Objeto Educativo, y sus preferencias de visualización [Sancho 2002].

En el modelo SCORM, es el contenido el que inicia y termina la comunicación informando de ello al LMS.

Una de las mayores ventajas de SCORM es el modo en que integra los distintos esfuerzos realizados por organismos como AICC, IEEE e IMS.

Sin embargo, SCORM no cubre todos los aspectos relacionados con la tecnología e-learning, por ejemplo no específica qué modelos de aprendizaje deben ser usados.

En Scorm, la fase de desarrollo requiere experiencia técnica adicional para cumplir las pautas de conformidad del modelo de referencia. Los programadores y desarrolladores deben tener los conocimientos básicos de la tecnología requerida para aplicar SCORM, basada en las guias Aggregation Model, Run-time Environment, y Sequencing y Navigation para los Objetos de Aprendizaje. Ellos deben ser capaces de estructurar los datos apropiadamente y agregar metadatos a las organizaciones, agregaciones, actividades, SCOs y assets.[Pisel 2004].

XML Lenguaje de Marcas Extensible (1996)

Introducción

Como es sabido HTML (Hypertex Markup Language) se ha convertido en un lenguaje de inmensa popularidad durante los últimos años. También debemos notar que nos hemos encontrado con sus propias limitaciones, algunas de las cuales se han querido subsanar con scripts, javascripts, Active X, HTML dinámico, etc; pero en la realidad todas estas herramientas no aportan una solución global a las limitaciones del HTML. La implantación de XML viene a eliminar este tipo de situaciones permitiendo la creación de herramientas más estructuradas.

¿Qué es XML? **(**http://www.xml.com)

XML es un lenguaje de marcas que ofrece un formato para la descripción de datos estructurados, el cual conserva todas las propiedades importantes del SGML. Es decir, XML es un metalenguaje, dado que con él podemos definir nuestro propio lenguaje de presentación y, a diferencia del HTML, que se centra en la representación de la información, XML se centra en la información en sí misma. La particularidad más importante de XML es que no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que las crea a su antojo, dependiendo del contenido del documento. De esta forma, los documentos XML con información sobre libros deberían tener etiquetas como <AUTOR>, <EDITORIAL>, <Nº_DE_PÁGINAS>, <PRECIO>, etc., mientras que los documentos XML relacionados con educación incluyen etiquetas del tipo de <ASIGNATURA>, <ALUMNO>, <CURSO>, <NOTA>, etc.
Por ejemplo en la siguiente tabla se muestra la información incluida por un código típico HTML y su versión equivalente en XML. Se puede apreciar en este ejemplo, que es mucho más fácil de entender la representación en XML.

HTML	**XML**
<TABLE>	**<LIBROS>**
<TR>	**<LIBRO>**
<TD>Título**</TD>**	**<TITULO>**AutoSketch**</TITULO>**
<TD>Autor**</TD>**	**<AUTOR>**Ramón Montero**</AUTOR>**
<TD>Precio**</TD>**	**<PRECIO>**33**</PRECIO>**
</TR>	**</LIBRO>**
<TR>	**<LIBRO>**
<TD>AutoSketch**</TD>**	**<TITULO>**Windows 98**</TITULO>**
<TD>Ramón Montero**</TD>**	**<AUTOR>**Jaime Perez**</AUTOR>**

<TD>33**</TD>**	**<PRECIO>**3.250**</PRECIO>**
</TR>	**</LIBRO>**
<TR>	**<LIBRO>**
<TD>Windows 98**</TD>**	**<TITULO>**Web Graphics**</TITULO>**
<TD>Jaime Perez**</TD>**	**<AUTOR>**Ron Wodaski**</AUTOR>**
<TD>3.250**</TD>**	**<PRECIO>**8.975**</PRECIO>**
</TR>	**</LIBRO>**
<TR>	**</LIBROS>**
<TD>Web Graphics**</TD>**	
<TD>Ron Wodaski**</TD>**	
<TD>8.975**</TD>**	
</TR>	
</TABLE>	

Objetivos y Orígenes

XML fue desarrollado por un grupo de trabajo bajo los auspicios del consorcio World Wide Web (W3C) a partir de 1996. Éste fue constituido en 1994 con el objetivo de desarrollar protocolos comunes para la evolución de Internet. Se trata de un consorcio de la industria internacional con sedes conjuntas en el Instituto Tecnológico de Massachussets, de Estados Unidos, el Instituto Nacional de Investigación en Informática y Automática europeo y la Keio University Shonan Fujisawa Campus de Japón. El W3C tiene como misión la publicación para uso público de protocolos o estándares globales de uso libre. Al comenzar el proyecto, los objetivos planteados por el grupo de desarrollo de XML fueron diez puntos [Young M. 2000]:

1. XML debe ser directamente utilizable sobre Internet.
2. XML debe soportar una amplia variedad de aplicaciones.
3. XML debe ser compatible con SGML.
4. Debe ser fácil la escritura de programas que procesen documentos XML.
5. El número de características opcionales en XML debe ser absolutamente mínimo, idealmente cero.
6. Los documentos XML deben ser legibles por los usuarios de este lenguaje y razonablemente claros.
7. El diseño de XML debe ser formal, conciso y preparado rápidamente.
8. XML debería ser simple pero perfectamente formalizado.
9. Los documentos XML deben ser fáciles de crear.
10. La brevedad en las marcas XML es de mínima importancia.

Características del XML

Introducción

XML es un formato basado en texto, específicamente diseñado para almacenar y transmitir datos. Un documento XML se compone de elementos XML, cada uno de los cuales consta de una etiqueta de inicio, de una etiqueta de fin y de los datos comprendidos entre ambas etiquetas. Al igual que los documentos HTML, un documento XML contiene texto anotado por etiquetas. Sin embargo, a diferencia de HTML, XML admite un conjunto ilimitado de etiquetas, no para indicar el aspecto que debe tener algo, sino lo que significa. Por ejemplo: un elemento XML puede estar etiquetado como precio, número de pedido o nombre. El autor del documento es quien decide qué tipo de datos va a utilizar y qué etiquetas son las más adecuadas.
Los documentos XML son fáciles de crear. En este ejemplo se utiliza XML para describir un parte meteorológico. Este documento se puede guardar con una extensión de XML, por ejemplo Tiempo.xml.

<reporte-clima>
<fecha>March 25, 1998**</fecha>**
<hora>08:00**</hora>**
<area>
<departamento>MVD**</ departamento >**
<ciudad>Montevideo**</ciudad>**
<pais>Uruguay**</pais>**
</area>
<medidas>
<cielo>parcialmente nublado **</cielo>**
<temperatura>16**</temperatura>**
<viento>
<direccion>SO**</direccion>**
<velocidad>16**</velocidad>**
</viento>
<h-indice>51**</h-indice>**
<humedad>87**</humedad>**
<visibilidad>10**</visibilidad>**
<uv-indice>1**</uv-indice>**
</medidas>
</reporte-clima>

En lugar de describir el orden y la disposición de la presentación de los datos, las etiquetas indican qué significa cada elemento de datos (si es un elemento <fecha>, un elemento <area>, etc.). Cualquier receptor de estos datos puede descodificar el documento y utilizarlo para sus propios fines.

Estándares abiertos

XML se basa en una tecnología desarrollada a partir de estándares probados y optimizada para la Web. La iniciativa XML consta de un conjunto de estándares relacionados entre sí:

XML *(Extensible Markup Language)*. Es una recomendación, que significa que el estándar es estable y que los desarrolladores Web y de herramientas pueden adoptarlo plenamente.

Namespaces. En XML es una recomendación que describe la sintaxis y la compatibilidad de los espacios de nombres para los intérpretes de XML.

DOM *(Document Object Model)*. Es una recomendación que ofrece un estándar para el acceso mediante programación a los datos estructurados (a través de scripts), de modo que los desarrolladores puedan interactuar de forma coherente con los datos basados en XML y computarlos.

XSL *(Extensible Stylesheet Language)*. XSL es la cara de presentación del XML. Este lenguaje debe representar la información existente en los documentos XML, de forma independiente a la plataforma utilizada.

XML Linking Language. Es un lenguaje que ofrece vínculos en XML parecidos a los de HTML, pero más potentes. Los vínculos pueden tener varias direcciones y pueden existir en el nivel de los objetos, no sólo en el nivel de las páginas.

Características Principales

Extensible

Dentro de XML se pueden definir un conjunto ilimitado de etiquetas. Mientras que las etiquetas de HTML pueden utilizarse para desplegar una palabra en negrita o itálica, XML proporciona un marco de trabajo para etiquetado de datos estructurados. Un elemento de XML puede declarar que sus datos asociados sean el precio de venta al público, un impuesto de venta, el título de un libro o cualquier otro elemento de datos deseado. Al irse adoptando las etiquetas XML a lo largo de una intranet de alguna organización y a lo ancho de la Internet, habrá una correspondiente habilidad para buscar y manipular datos sin importar las aplicaciones dentro de las cuales se encuentre.

Representación estructural de los datos

XML proporciona una representación estructural de los datos que ha probado ser ampliamente implementable y fácil de distribuir. Las implementaciones industriales en la comunidad del SGML y en otros lugares han demostrado la calidad intrínseca y la fortaleza industrial del formato de datos con estructura de árbol de XML. XML es un subconjunto de SGML que está optimizado para su transmisión por Web; al estar definido por el Consorcio de la World Wide Web, asegura que los datos estructurados serán uniformes e independientes de aplicaciones o compañías. Esta interoperabilidad resultante está dando el impuso de inicio a una nueva generación de aplicaciones de Web para comercio electrónico.

El lenguaje XML proporciona un estándar de datos que puede codificar el contenido, la semántica y el esquema de una amplia variedad de casos que van desde simples a complejos, por ejemplo XML puede ser utilizado para marcar lo siguiente:

- Un documento ordinario.
- Un registro estructurado, tal como un registro de citas u órdenes de compra.
- Un registro de datos, tal como el resultado de una consulta.

- Metacontenido acerca de un sitio Web, tal como un Formato de Definición de Canal (*Channel Definition Format*, CDF).

Presentaciones gráficas, tales como la interfase de usuario de una aplicación

Una vez que los datos estén en el escritorio del cliente, pueden ser manipulados, editados, y presentados de una gran variedad de maneras, sin viajes de regreso al servidor. Los servidores se pueden convertir ahora en más escalables, debido a las menores cargas de cálculo y ancho de banda. Además, dado que los datos son intercambiados en el formato XML, pueden ser fácilmente mezclados desde diferentes fuentes.

Los datos son separados de la presentación y el proceso

El poder de XML es que mantiene la separación entre la interfase de usuario y los datos estructurados. HTML especifica como visualizar datos en un navegador, en cambio XML define el contenido. XML sólo utiliza etiquetas para describir los datos, tales como el nombre de la ciudad, temperatura y presión barométrica. Para presentar los datos en un navegador XML, éste utiliza hojas de estilo tales como el Lenguaje de Estilo Extensible (XSL) y las Hojas de Estilo en Cascada (CSS). XML separa los datos de la presentación y el proceso, permitiendo desplegar y procesar los datos tal como usted desee, al aplicar diferentes hojas de estilo y aplicaciones.

Esta separación de datos de la presentación permite una integración de datos perfecta de fuentes diversas. La información de clientes, órdenes de compra, resultados de investigaciones, pagos de facturas, registros médicos, datos de catálogo y cualquier otra información se puede convertir a XML, permitiendo a los datos ser intercambiados en línea tan fácilmente como las páginas de HTML despliegan datos hoy. Los datos codificados en XML pueden ser transmitidos sobre la Web hasta el escritorio. No es necesario retroajustar información en formatos propietarios almacenados en bases de datos o documentos de mainframes y, debido a que se usa el HTTP para transmitir documentos XML sobre la red, no se necesitan cambios para esta función. Los documentos XML son fáciles de crear; si está familiarizado con el HTML, puede aprender rápidamente a crear uno.

Conversión de los datos XML en autodescriptivos

Los datos codificados en XML son autodescriptivos, pues las etiquetas descriptivas están entremezcladas con los datos. El formato abierto y flexible utilizado por XML permite su uso en cualquier lugar donde sea necesario intercambiar y transferir información. Dado que XML es independiente del HTML, se puede insertar código XML en documentos HTML. El W3C ha definido un formato mediante el cual se pueden encapsular en páginas HTML los datos basados en XML. Al incrustar datos XML en una página HTML, se pueden generar varias vistas a partir de los datos entregados, utilizando los datos semánticos que contiene el XML.

DTD

El DTD (definición del tipo de documento, Document Type Definition) proporciona la gramática para una clase de documentos XML. Esta gramática contiene la definición del conjunto de etiquetas que puede contener esa clase de documentos XML, los nombres que pueden utilizarse en los elementos, dónde pueden aparecer y

cómo se interrelacionan entre ellos. Se puede decir que un DTD es una definición exacta de la gramática de un documento.

Los documentos XML enviados con un DTD se reconocen como "XML válido". En este caso, un intérprete de XML podría comparar los datos entrantes con las normas definidas en el DTD para comprobar que los datos se han estructurado correctamente. Los datos enviados sin un DTD se reconocen como "bien formados". En XML no existen DTDs predefinidos, por lo que es labor del diseñador especificar su propia DTD para cada tipo de documento XML. En la especificación de XML se describe la forma de definir DTDs particularizadas para documentos XML, que pueden ser internas (cuando van incluidas junto al código XML) o externas (si se encuentran en un documento propio).

Un ejemplo de un DTD que defina la estructura de un documento XML relacionado con libros, podría ser un documento conteniendo el siguiente código:

```
<!ELEMENT LIBROS (LIBRO)+>
<!ELEMENT LIBRO (TITULO,AUTOR,PRECIO)>
<!ELEMENT TITULO (#PCDATA)>
<!ELEMENT AUTOR (#PCDATA)>
<!ELEMENT PRECIO (#PCDATA)>
```

En el DTD del ejemplo se definen los elementos (!ELEMENT) que integran el documento XML. En la primera línea se indica que el elemento principal es LIBROS, del que dependen uno o más (+) elementos LIBRO. La segunda línea sirve para especificar que el elemento LIBRO contiene tres elementos (TITULO,AUTOR,PRECIO) que se deben marcar en dicho orden. Las restantes líneas aclaran que los elementos TITULO, AUTOR y PRECIO contienen valores de cadenas de texto.

Esquemas

Últimamente se está imponiendo otra forma más eficaz de definir la estructura de un documento XML, conocida como esquemas. Un esquema es una especificación formal de las normas de un documento XML, que indica qué elementos se permiten en un documento y en qué combinaciones están permitidas. Los nuevos lenguajes de esquemas, definidos en las propuestas XML-Data (Datos de XML) y DCD (Descripción del contenido del documento) enviadas por el XML-Data Working Group al W3C, proporcionan las mismas funciones que un documento DTD. No obstante, puesto que estos lenguajes de esquema son extensibles, los desarrolladores pueden ampliarlos con información adicional, como los tipos de datos, la herencia y las normas de presentación. Esto hace que los nuevos lenguajes de esquemas sean mucho más potentes que los DTD.

La expresión de esquemas dentro de XML aumenta la potencia del formato XML, pues permite que el software examine determinados datos para comprender su estructura, sin necesitar ninguna descripción previa incorporada de la estructura de los datos. Con un esquema, un autor puede definir exactamente qué nombres de elementos se permiten en un documento y, dentro de cada elemento, qué subelementos, atributos y relaciones se admiten. El autor puede importar fragmentos de otros esquemas, así como ampliar tipos a través de la herencia. Todo ello permite establecer relaciones complejas entre los elementos sin perder la simplicidad de la estructura de árbol léxico.

Fortalezas y Debilidades del XML

La meta fundamental de XML es facilitar la cooperación y la interoperabilidad entre módulos que pertenecen a diferentes sistemas, e incluso a diferentes organizaciones. Entonces, ¿por qué XML se ha convertido en tecnología de alto nivel tan rápidamente? La respuesta es simple: XML es texto simple, pero es también elegante. Un lenguaje de marcas hace fácil identificar (y extraer en un tiempo posterior) segmentos específicos de información con significados especiales. Usar texto simple y mantener el contenido universalmente interpretable son dos condiciones para implementar una tecnología exitosamente entre sistemas heterogéneos. Una cadena de texto se puede transferir y se puede manejar fácilmente en cualquier plataforma destino. Sin embargo, si usted sólo planea definir un protocolo basado en texto para hacer que sus módulos se comuniquen, usted no necesita necesariamente XML. El ASCII, de hecho, ha existido durante mucho tiempo. La diferencia recae en las etiquetas XML que se utilizan para delimitar segmentos de información.

La principal fortaleza de XML es también, actualmente, su debilidad principal. XML es demasiado genérico para ser utilizado sin definir externamente la sintaxis exacta de un documento y cómo puede localizarse y extraerse cada fragmento de datos intercambiados. Tal estandarización es más simple de alcanzar dentro de una sola aplicación corporativa, incluso si atraviesa varias plataformas heterogéneas de hardware y de software. Conseguir el mismo resultado en un contexto más amplio requiere un enorme esfuerzo para llegar a algunas definiciones estándares. Este proceso puede tomar años en ser completado [MSDN Latinoamérica].

Desgraciadamente, cuando se trata de integrar sistemas de diferentes organizaciones, especialmente de una manera abierta, XML muestra algunas limitaciones. XML trata de la descripción de los datos, pero solamente es útil cuando la gente está de acuerdo en la manera que los datos deben ser descritos. Una factura, por ejemplo, es lógicamente el mismo tipo de documento para cualquier proceso que lo manipule. Pero diferentes procesos pueden convertirla en formatos binarios diferentes. A pesar del formato de almacenamiento físico, el documento necesita ser transmitido y ser recibido en un formato comúnmente reconocido disponible en todas las plataformas. Por supuesto, XML se puede utilizar como este formato.

Imagine el enorme costo que supondría para los consumidores y las empresas si cada empresa definiera su propia forma de publicar la información. Incluso con la Web, los costos asociados a la configuración y el mantenimiento de un sitio Web superan la capacidad de algunas empresas. Al no haber ningún límite en el número de empresas que podrían publicar esta información, la falta de estándares que definan el modo de publicarla de una forma segura y controlada tendría como consecuencia miles y miles de implementaciones, enfoques de exploración de la Web y profundidad del contenido distintos.

Por este motivo, en la actualidad se están definiendo esquemas por grupos sectoriales con similares intereses, de forma que existirán esquemas estándares avalados por asociaciones de empresas y organismos que garanticen que cualquier usuario que las adopte como suyas, trabaje con las mismas etiquetas. Como ejemplos de estos esquemas estándares tenemos: CDF - Channel Definition Format (define canales para enviar información periódica a los clientes), CML - Chemical Markup Language (define información del sector químico), MathML - Mathematical Markup Language (define datos matemáticos), SMIL - Synchronized Multimedia Integration Language (define presentaciones de recursos multimedia).

DOM

Supongamos que usted es un programador Visual Basic y ha recibido algunos datos en un documento XML. Ahora desea extraer la información del documento XML

e integrar esos datos en sus soluciones de Visual Basic. Por supuesto, podría escribir código para analizar el contenido del documento XML, pues no deja de ser un documento de texto. Sin embargo, esta solución no sería muy productiva y desaprovecharía una de las ventajas de XML**:** el ser una forma estructurada de representar datos. Una forma mejor de recuperar información de documentos XML es utilizar un analizador o intérprete de XML. Un intérprete de XML es, a grandes rasgos, un programa que lee un documento XML y permite disponer de sus datos.

En su calidad de programador en Visual Basic, querrá utilizar un intérprete que sea compatible con el DOM (modelo de objeto de documento, Document Object Model) de XML. Éste define un conjunto estándar de comandos que los intérpretes exponen para facilitarle el acceso al contenido de los documentos XML desde sus programas. Un intérprete de XML que sea compatible con DOM toma los datos de un documento XML y los expone mediante un conjunto de objetos que se pueden programar. Además, un intérprete de XML puede utilizar un DTD o un esquema para determinar si un documento es válido.

DOM para XML es un modelo de objetos que muestra el contenido de un documento XML. La Especificación de nivel 1 del DOM del W3C define actualmente lo que debería mostrar un DOM como propiedades, métodos y eventos. Para utilizar XML DOM, hay que crear una instancia de un intérprete XML. Por ejemplo para ello, Microsoft® muestra XML DOM mediante un conjunto de interfaces COM estándar en Msxml.dll. El archivo Msxml.dll contiene la biblioteca de tipos y el código de implementación para trabajar con documentos XML.

A continuación se mencionan como ejemplos algunos de los métodos y propiedades especificados por DOM:

- Método "Load", que permite cargar documentos XML procedentes de un disco local, de la red o de una dirección URL.
- Propiedad "Async", determina si el intérprete de XML carga los documentos de manera asincrónica o no.
- Propiedad " ReadyState", indica en qué estado se encuentra la carga del documento
- Propiedad "ValidateOnParse", indicar al intérprete que no valide el documento.
- Propiedad "ChildNodes", muestra la lista de nodos XML DOM.
- Propiedad "Level", que devuelve el número de nodos secundarios existentes.
- Propiedad "HasChildNodes", facilita el recorrido de la jerarquía de nodos para examinar elementos, atributos y valores.

Documentos XML en el Web

Desplegar documentos

Un navegador, después de chequear la sintaxis del código del documento, debe presentar la información del documento con un formato determinado. Los documentos HTML utilizan las descripciones de formatos internas del propio navegador, o si existen descripciones CSS (que son opcionales), utilizan la información de la hoja de estilo para ajustar la presentación en la pantalla. Los documentos XML siempre necesitan normas que describan su presentación. Para describir cómo se deben presentar los documentos

XML podemos optar por dos soluciones: las mismas descripciones CSS que se utilizan con HTML y/o las descripciones que se basan en XSL.

Si ya existía una forma de definir las presentaciones de los documentos Web, la interrogante que puede surgir es cuál fue el motivo que llevó a desarrollar otra forma específica para XML? La respuesta es que CSS es eficaz para describir formatos y presentaciones, pero no sirve para decidir qué tipos de datos deben ser mostrados y cuáles no. Esto es, CSS se utiliza con documentos XML en los casos en los que todo su contenido debe mostrarse sin mayor problema. XSL no sólo permite especificar cómo queremos presentar los datos de un documento XML, sino que también sirve para filtrar los datos de acuerdo a ciertas condiciones. Se parece un poco más a un lenguaje de programación.

CSS

Las especificaciones CSS (hojas de estilo en cascada, Cascading Style Sheets) deben comprenderse como una descripción del formato en el que se desea que aparezcan las entidades definidas en un documento. Por ejemplo, si se define una hoja de estilo ligada con un documento HTML con el siguiente código:

```
P {font-family: Verdana; font-size: 10 pt}
TABLE {border: 2; font-family: Tahoma; font-size: 9 pt}
H3 {font-family: Comic Sans MS; font-size: 12 pt; color: blue}
```

Se indica al navegador que presente los textos incluidos entre <P> y </P> con un tipo de letra Verdana de 9 puntos, las tablas con una fuente de letra Tahoma de 9 puntos y un ancho de 2 en los bordes, y los titulares incluidos entre <H3> y </H3> con una letra Comic Sans MS de 12 puntos y color azul. Utilizar CSS con XML es similar, con la excepción de que las etiquetas son diferentes a las de HTML [Young M. 2000]. Un código como el siguiente indica para un documento XML cómo debe mostrar las etiquetas <AUTOR>, <PRECIO> y <TITULO>:

```
AUTOR {display:block; font-family:Arial; font-size:small; width:30em}
PRECIO {display:block; padding:1.2em; font-size:x-small}
TITULO {display:block; font-size:x-large; text-align:center; color:#888833}
```

XSL

SGML tiene su propio estándar para la representación de sus documentos, el DSSSL (Semántica de Estilo de Documento y Lengua de Especificación, ISO10179) (http://www.bibliodgsca.unam.mx/tesis/tes7cllg/sec_26.htm), que en realidad es un lenguaje de programación completo y muy potente. Por tanto, ya que XML es una versión reducida de SGML parecía lógico hacer también una versión reducida del DSSSL, llamada en este caso XSL (lenguaje de hojas de estilo extensible, Extensible Stylesheet Language) [Gómez O. Tutorial sobre XML].

Básicamente, XSL es un lenguaje de hojas de estilos diseñado para su utilización en el Web. XSL debe representar la información recogida en los documentos XML de forma independiente a la plataforma utilizada o al medio de representación.

En cuanto a la inclusión de imágenes en las páginas XML, estas son enlaces, que pueden representarse por alguno de los tipos soportado por las especificaciones XLink y

XPointer. XSL, además de permitir la descripción de la presentación física, también posibilita la ejecución de bucles, sentencias del tipo IF...THEN, selecciones por comparación, operaciones lógicas, ordenaciones de datos, utilización de plantillas, etc. Para que el lector se haga una pequeña idea de cómo es un código XSL, a continuación se muestra un sencillo ejemplo que permitiría mostrar todos los contenidos de las etiquetas <TITULO>, <AUTOR> y <PRECIO> de un documento XML, mediante un bucle sin condiciones:

```
<xsl:template match="/">
<HTML>
<BODY>
<xsl:for-each select="/LIBROS/LIBRO">
Título:
<xsl:value-of select="TITULO"/><BR/>
Autor:
<xsl:value-of select="AUTOR"/><BR/>
Precio:
<xsl:value-of select="PRECIO"/> pesetas<BR/>
</xsl:for-each>
</BODY>
</HTML>
</xsl:template>
```

Vínculos entre Documentos XML (XLink/XPointer)

Los enlaces e hipervínculos son tan importantes para los documentos XML que el W3C ha sacado las especificaciones que las controlan fuera de las descripciones del DTD, creando dos normas: XLink y XPointer. Estas definen el modo de enlace entre diferentes documentos. Este lenguaje va más allá de los enlaces simples que sólo soporta el HTML. Esta especificación soporta por ejemplo las siguientes características:

- Denominación independiente de la ubicación.
- Enlaces que pueden ser también bidireccionales.
- Enlaces que pueden especificarse y gestionarse desde fuera del documento a los que se apliquen (esto permitirá crear en un entorno intranet/extranet un banco de datos de enlaces en los que se puede gestionar y actualizar automáticamente).
- Hiperenlaces múltiples (anillos, múltiples ventanas, etc.).
- Enlaces agrupados (múltiples orígenes).

XLink define la forma en la que los documentos XML deben conectarse entre sí. XPointer describe cómo se puede apuntar a un lugar específico de un determinado documento XML. Resumiendo, XLink determina el documento al que se desea acceder y XPointer marca el lugar exacto en dicho documento. Al contrario de lo que ocurre con HTML, en XML existen dos tipos básicos de hipervínculos: simples y extendidos. Un ejemplo de un hipervínculo simple sería:

```
<AUTOR xlink:href="autores.xml#juan" xlink:show="new">
<NOMBRE>Juan Primero Segundo</NOMBRE>
</AUTOR>
```

Otro ejemplo de un hipervínculo extendido podría ser:

```
<EDITOR_AUTOR xlink:extended>
<xlink:locator href="#ana" id="editor"/>
<xlink:locator href="autores.xml#juan" id="autor"/>
<xlink:arc from="editor" to="autor" show="replace"/>
</EDITOR_AUTOR xlink:extended>
```

En el primero se puede observar la definición de un hipervínculo simple que se abre en una nueva ventana (show="new"), mientras que en el segundo se define un hipervínculo con tres posibilidades diferentes: a una sección determinada del documento (#ana), o a un determinado lugar de otro documento (autores.xml#juan), o a una zona delimitada por dos marcadores (editor y autor). El funcionamiento de un hipervínculo simple no tiene secretos, pues es similar al que se utiliza en HTML.

Resource Description Framework (RDF) (1997)

Introducción

Resource Description Framework (RDF) [Infraestructura para la Descripción de Recursos] es una base para procesar metadatos; proporciona interoperabilidad entre aplicaciones que intercambian información legible por máquina en la Web. RDF se destaca por la facilidad para habilitar el procesamiento automatizado de los recursos Web. RDF puede utilizarse en distintas áreas de aplicación; por ejemplo: en *recuperación de recursos* para proporcionar mejores prestaciones a los motores de búsqueda, en *catalogación* para describir el contenido y las relaciones de contenido disponibles en un sitio Web, una página Web, o una biblioteca digital particular, en *agentes de software inteligentes* para facilitar el intercambio y para compartir conocimiento; en la *calificación de contenido*, en la descripción de *colecciones de páginas* que representan un "documento" lógico individual, en la descripción de los *derechos de propiedad intelectual* de las páginas web, y para expresar las *preferencias de privacidad* de un usuario, así como las *políticas de privacidad* de un sitio Web. RDF junto con las *firmas digitales* será la clave para construir el "Web de confianza" para el comercio electrónico, la cooperación y otras aplicaciones.

Introduciremos un modelo para representar metadatos en RDF así como una sintaxis para codificar y transmitir estos metadatos de tal forma que maximicen la interoperabilidad de servidores y clientes web desarrollados independientemente. La sintaxis presentada aquí utiliza el Extensible Markup Language [Lenguaje de Marcado Extensible] (XML): uno de los objetivos de RDF es hacer posible especificar la semántica para las bases de datos en XML de una forma normalizada e interoperable. RDF y XML son complementarios: RDF es un modelo de metadatos y sólo dirige por referencia muchos de los aspectos de codificación que requiere el almacenamiento y transferencia de archivos (tales como internacionalización, conjuntos de caracteres, etc.). Para estos aspectos, RDF cuenta con el soporte de XML. Es importante también entender que esta sintaxis XML es sólo una sintaxis posible para RDF y que pueden surgir formas alternativas para representar el mismo modelo de datos RDF.

Objetivo

El objetivo general de RDF es definir un mecanismo para describir recursos que no cree ninguna dependencia sobre un dominio de aplicación particular, ni defina (a priori) la semántica de algún dominio de aplicación. La definición del mecanismo debe ser neutral con respecto al dominio, sin embargo el mecanismo debe ser adecuado para describir información sobre cualquier dominio.

Como resultado de la reunión de distintas comunidades que están de acuerdo en los principios básicos de la representación y transposición de metadatos, RDF está influido de varias fuentes diferentes. Las principales influencias provienen de la propia *Comunidad de Normalización de la Web* en forma de metadatos HTML y PICS, la *comunidad bibliotecaria*, la *comunidad de los documentos estructurados* en forma de SGML y sobre todo XML, y también de la *comunidad de representación del conocimiento (KR)*. También han contribuido al diseño de RDF otras áreas de la tecnología; incluidos los lenguajes de modelado y programación orientada a objetos, así como las bases de datos. Mientras RDF surge de la comunidad KR [de la representación del conocimiento], la bibliografía relacionada con este campo, advierte que RDF no especifica un mecanismo para el razonamiento. RDF puede definirse como

un sistema simple. Un mecanismo de razonamiento debe construirse sobre este sistema de referencia.

RDF Básico

Modelo RDF básico

El fundamento o base de RDF es un modelo para representar propiedades designadas y valores de propiedades. El modelo RDF se basa en principios perfectamente establecidos de varias comunidades de representación de datos. Las propiedades RDF pueden recordar a atributos de recursos y en este sentido corresponden con los tradicionales pares de atributo-valor. Las propiedades RDF representan también la relación entre recursos y por lo tanto, un modelo RDF puede parecer un diagrama entidad-relación. De forma más precisa, los esquemas RDF —que son objetos específicos de la categoría del modelo de datos RDF — son diagramas ER [Entidad Relación]. En la terminología del diseño orientado a objetos, los recursos corresponden con objetos y las propiedades corresponden con objetos específicos y variables de una categoría.

El modelo de datos de RDF es una forma de sintaxis-neutral para representar expresiones RDF. La representación del modelo de datos se usa para evaluar la equivalencia en significado. Dos expresiones RDF son equivalentes si y sólo si sus representaciones del modelo de datos son las mismas. Esta definición de equivalencia permite algunas variaciones sintácticas en expresiones sin alterar el significado.

El modelo de datos básico consiste en tres tipos de objetos:

Recursos: Todas las cosas descritas por expresiones RDF se denominan *recursos*. Un recurso puede ser una página Web completa; tal como el documento HTML "http://www.w3.org/Overview.html" por ejemplo. Un recurso puede ser una parte de una página Web; por ej. un elemento HTML o XML específico dentro del documento fuente. Un recurso puede ser también una colección completa de páginas; por ej. un sitio Web completo. Un recurso puede ser también un objeto que no sea directamente accesible vía Web, por ej. un libro impreso. Los recursos se designan siempre por URIs más identificadores de anclas opcionales. Cualquier objeto puede tener una URI; la extensibilidad de URIs permite la introducción de identificadores para cualquier entidad imaginable.

Propiedades: Una *propiedad* es un aspecto específico, característica, atributo, o relación utilizada para describir un recurso. Cada propiedad tiene un significado específico, define sus valores permitidos, los tipos de recursos que puede describir, y sus relaciones con otras propiedades.

Sentencias: Un recurso específico junto con una propiedad denominada, más el valor de dicha propiedad para ese recurso es una *sentencia RDF* [RDF statement]. Estas tres partes individuales de una sentencia se denominan, respectivamente, sujeto, predicado y objeto. El objeto de una sentencia (es decir, el valor de la propiedad) puede ser otro recurso o pude ser un literal; es decir, un recurso (especificado por una URI) o una cadena simple de caracteres [string] u otros tipos de datos primitivos definidos por XML. En términos RDF, un *literal* puede comprender en su contenido marcado XML pero ya no puede valorarse más por un procesador RDF. Existen varias restricciones sintácticas en cómo se puede expresar el marcado en literales.

Ejemplos

Considerar como ejemplo simple la sentencia:

Ora Lassila es el creador [autor] del recurso http://www.w3.org/Home/Lassila.

Esta sentencia comprende las siguientes partes:

Sujeto (Recurso)	http://www.w3.org/Home/Lassila
Predicado (Propiedad)	Creator
Objeto (literal)	"Ora Lassila"

En este documento podríamos representar gráficamente una sentencia RDF usando gráficos etiquetados (también denominados "diagramas de nodos y arcos"). En estos gráficos, los nodos (dibujados como óvalos) representan recursos y los arcos representan propiedades denominadas. Los nodos que representan cadenas de literales pueden dibujarse como rectángulos. La sentencia citada anteriormente se representaría gráficamente como:

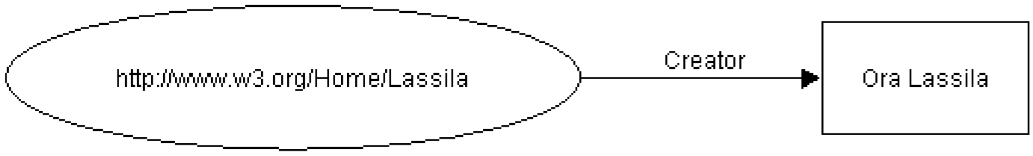


Figura 1. Diagrama de nodo y arco simples

Nota: La dirección de la flecha es importante. El arco siempre empieza en el sujeto y apunta hacia el objeto de la sentencia. Este diagrama simple puede también interpretarse o leerse "*http://www.w3.org/Home/Lassila tiene como creador a Ora Lassila*", o en general "*<sujeto> TIENE <predicado> <objeto>*".

Ahora, imaginemos que queremos decir algo más sobre las características del creador de este recurso. Tal declaración en lenguaje natural podría ser:

El individuo cuyo nombre es Ora Lassila, correo electrónico <lassila@w3.org>, es el creador de http://www.w3.org/Home/Lassila.

La intención de esta frase es precisar el valor de la propiedad **Creator** [Creador] una entidad estructurada. En RDF tal entidad se representa como otro recurso. La sentencia anterior no proporciona un nombre a ese recurso; es anónimo, por ello el gráfico a continuación lo representa como un óvalo vacío:

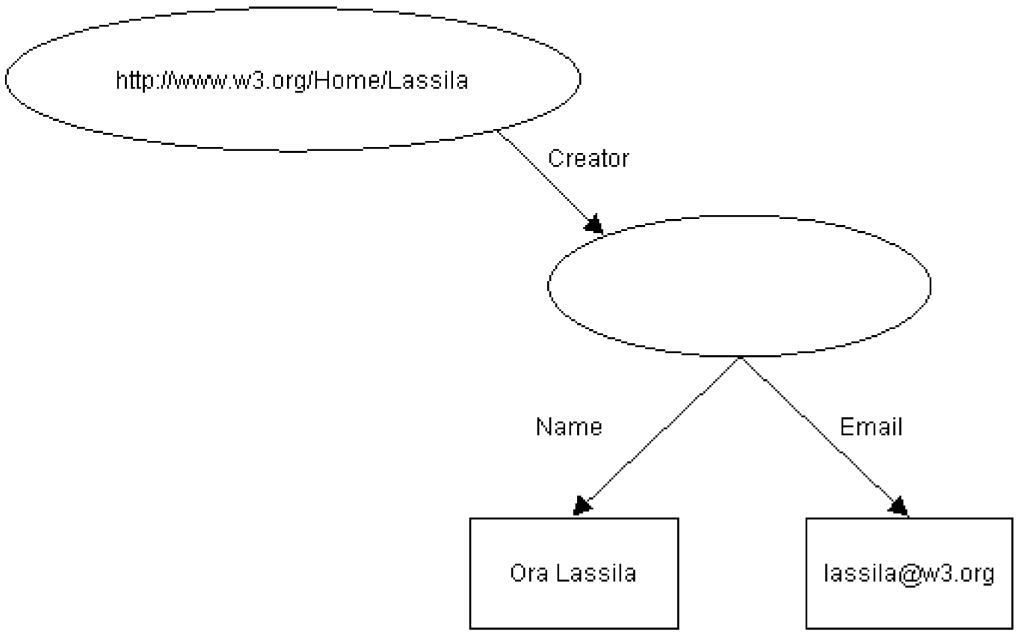


Figura 2. Propiedad con valor estructurado

Nota: correspondiendo con la lectura de la nota anterior, este diagrama puede leerse: *"http://www.w3.org/Home/Lassila tiene el creador cualquiera y este cualquiera tiene el nombre Ora Lassila y el correo electrónico lassila@w3.org*".

A la entidad estructurada del ejemplo anterior se le puede asignar un único identificador. El diseñador de la aplicación de la base de datos hace la elección del identificador. Continuando con el ejemplo, imagine que un identificador empleado se utiliza como único identificador para un recurso "persona". Las URIs que sirven como única clave para cada empleado (definido por la organización) podría ser entonces algo como: http://www.w3.org/staffId/85740. Ahora podemos escribir dos frases o sentencias:

El individuo al que se refiere el identificador de empleado id 85740 se llama Ora Lassila y tiene la dirección de correo lassila@w3.org. Ese individuo creó el recurso http://www.w3.org/Home/Lassila

El modelo RDF para estas frases o sentencias es:

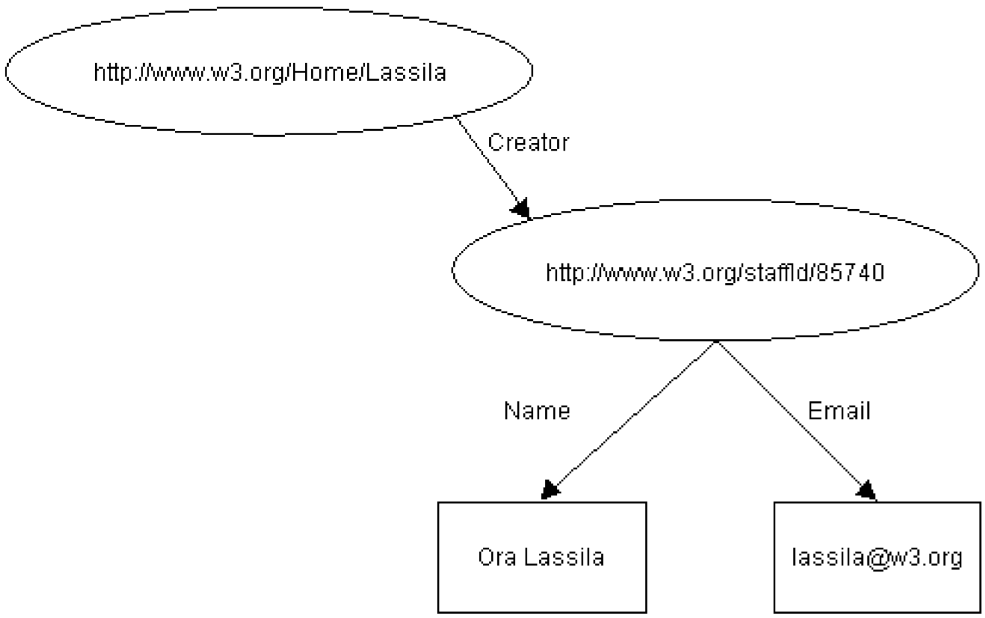


Figura 3. Valor estructurado con identificador

Obsérvese que este gráfico es idéntico al anterior con la adición de la URI para el recurso que antes era anónimo. Desde el punto de vista de una segunda aplicación que interrogue este modelo, no hay distinción entre la sentencia hecha de forma individual y la sentencia realizada por partes. Sin embargo, algunas aplicaciones necesitarán hacer esta distinción y RDF lo soporta.

Sintaxis RDF básica

El modelo de datos RDF proporciona un marco abstracto y conceptual para definir y utilizar metadatos. Necesita también una sintaxis concreta para crear e intercambiar metadatos. RDF utiliza XML codificado como su sintaxis de intercambio. RDF necesita también la facilidad de los namespaces XML para asociar con precisión cada propiedad con el esquema que define dicha propiedad.

Esquemas y Namespaces

Cuando escribimos una frase en lenguaje natural utilizamos palabras que encierran la intención de transmitir un significado inequívoco. Ese significado es fundamental para entender el enunciado y, en el caso de aplicaciones de RDF es crucial para establecer que el procesamiento correcto se dé como se esperaba. Es muy importante que tanto el escritor como el lector de una sentencia [declaración] entiendan el mismo significado para los términos utilizados, tales como **Creator, approvedBy, Copyright**, etc. o resultará una gran confusión. En un medio a escala global como es World Wide Web no es suficiente confiar en el entendimiento cultural de conceptos tales como la "autoría de una página" ["creatorship"]; conviene ser lo más preciso posible.

En RDF el significado se expresa a través de un esquema. Podemos pensar en un esquema como en una especie de diccionario. Un esquema define los términos que se utilizarán en una sentencia [declaración] RDF y le otorgará significados específicos.

Un esquema es un sitio donde se explican las definiciones y restricciones de uso de las propiedades. Para evitar confusiones entre definiciones independientes --y posiblemente conflictivas-- del mismo término, RDF utiliza la facilidad de los *namespaces* de XML. Los namespaces ["espacios de nombre"] son simplemente una forma de asociar el uso específico de una palabra en el contexto del diccionario (esquema) en que se puede encontrar una definición determinada. En RDF, cada predicado utilizado en una sentencia [declaración] debe ser identificado con un sólo namespace, o esquema. Sin embargo un elemento **Description** puede contener sentencias con predicados de varios esquemas.

Valores de propiedad cualificados

Muchas veces el valor de una propiedad es algo que tiene información contextual adicional que se considera "parte de" dicho valor. En otras palabras, es necesario cualificar [o distinguir] los valores de las propiedades. Ejemplos de tales distinciones incluyen la denominación de una unidad de medida, un vocabulario restringido particular, u otros comentarios. Para algunos usos resulta apropiado utilizar el valor de la propiedad sin identificadores o calificadores. Por ejemplo, en la frase " el precio de aquel lápiz es 75 centavos de dólar" en la mayoría de los casos es suficiente decir simplemente "el precio de aquél lápiz es 75".

En el modelo RDF un valor de propiedad cualificada es simplemente otra instancia de un valor estructurado. El objeto de la sentencia original es este valor estructurado y los calificadores [identificadores] representan más propiedades de dicho recurso. El principal valor que se cualificará se dará como el valor de la propiedad *value* del recurso en cuestión.

Contenedores

Normalmente es necesario referirse a una colección de recursos, por ejemplo, para expresar que un trabajo se creó por más de una persona, o para enumerar los estudiantes de un curso, o los módulos de un software en un paquete. Los contenedores RDF se usan para mantener tales listas de recursos o literales.

Modelo Contenedor

RDF define tres tipos de objetos contenedores:

Bag: Una lista desordenada de recursos o literales. Los ***Bags*** se utilizan para indicar que una propiedad tiene múltiples valores y que no es significativo el orden en que se den tales valores. *Bag* podría usarse para dar una lista de números de parte, donde el orden de procesamiento de las partes no tiene importancia. Se permite duplicar valores.

Sequence: Una lista ordenada de recursos o literales. ***Sequence*** se usa para manifestar que una propiedad tiene múltiples valores y que el orden de los valores es significativo. *Sequence* podría usarse, por ejemplo para conservar un orden alfabético de valores. Se permite duplicar valores.

Alternative: Una lista de recursos o literales que representan alternativas para un valor (individual) de una propiedad. *Alternative* podría usarse para proporcionar una lengua alternativa de traducción para el título de un trabajo, o para proporcionar una lista de mirrors de Internet en los que se podría encontrar un recurso. Una aplicación que utiliza

una propiedad cuyo valor es una colección alternativa, sabe que se puede elegir como correcto uno cualquiera de los ítems en la lista.

Nota: Las definiciones de **Bag** *y* **Sequence** *permiten explícitamente duplicar valores. RDF no define un concepto puntual de* **Set***, que podría ser un* Bag *sin duplicados, porque la esencia de RDF no impone un mecanismo de ejecución en el caso de que se violen estas restricciones*

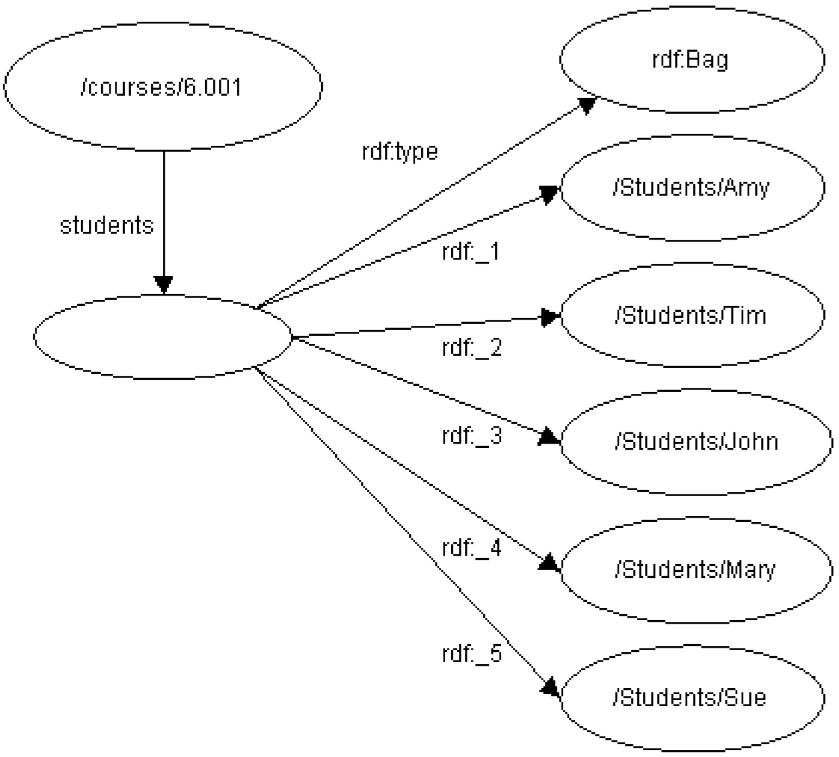


Figura 4. Ejemplo de Contenedor *Bag* simple

Declaraciones sobre declaraciones [sentencias sobre sentencias]

Además de hacer sentencias sobre los recursos Web, RDF puede usarse para crear sentencias sobre otras declaraciones RDF; nos referiremos a éstas como sentencias de alto nivel [*higher-order statements*]. Para realizar declaraciones sobre otras declaraciones, tenemos que construir un modelo de la sentencia original; este modelo es un nuevo recurso al que podemos anexar propiedades adicionales.

Modelado de sentencias

Las sentencias se hacen sobre los recursos. Un modelo de una sentencia es el recurso que necesitamos para poder hacer una nueva declaración (una sentencia de alto nivel) sobre la declaración modelada.

Por ejemplo, consideremos la frase:

Ora Lassila es el creador [autor] del recurso http://www.w3.org/Home/Lassila.

RDF estimaría esta sentencia como un hecho. Si, en lugar de eso, escribimos la frase:

Ralph Swick dice que Ora Lassila el creador [autor] del recurso http://www.w3.org/Home/Lassila.

No hemos dicho nada sobre el recurso http://www.w3.org/Home/Lassila; en cambio, hemos expresado el hecho sobre la afirmación que hizo Ralph. Para expresar este hecho en RDF, tenemos que modelar la declaración original como un recurso con cuatro propiedades. Este proceso se denomina formalmente "*reification" [transformación de algo abstracto en concreto]* en el contexto de la Representación del Conocimiento. Un modelo de sentencia se denomina "*reified statement" [sentencia transformada de lo abstracto a lo concreto]*. (http://www.w3.org/TR/2003/PR-rdf-primer-20031215/#reification)

RDF define las siguientes propiedades para modelar sentencias:

Subject: La propiedad *subject* identifica el recurso que describe la sentencia modelada; es decir, el valor de la propiedad *subject* es el recurso sobre el cual se hizo la sentencia original (en nuestro ejemplo, http://www.w3.org/Home/Lassila).

Predicate: La propiedad *predicate* identifica la propiedad original en la declaración modelada. El valor de la propiedad *predicate* es un recurso que representa la propiedad específica en la sentencia original (en nuestro ejemplo, el creador **creator**).

Object: La propiedad *object* identifica el valor de la propiedad en la sentencia modelada. El valor de la propiedad *object* es el objeto en la sentencia original (en nuestro ejemplo "Ora Lassila").

Type: El valor de la propiedad *type* describe el tipo del nuevo recurso. Todas las sentencias transformadas [de lo abstracto a lo concreto] son instancias de RDF: sentencias; es decir tienen una propiedad *type* cuyo objeto es RDF: sentencia. La propiedad *type* se usa más comúnmente también para declarar el tipo del recurso.

Un nuevo recurso con las cuatro propiedades anteriores representa la sentencia original y puede usarse tanto como objeto de otras declaraciones, como tener una sentencia adicional sobre él. El recurso con estas cuatro propiedades no es un substituto de la sentencia original, es un modelo de la declaración. Una sentencia [declaración] y sus correspondientes sentencias transformadas [de lo abstracto a lo concreto, *reified*] existe independientemente sin las otras. La representación gráfica RDF se presenta para contener el hecho de la sentencia si y sólo si dicha sentencia se presenta en forma gráfica, sin considerar si la sentencia correspondiente transformada está presente.

Para modelar el ejemplo anterior, podemos adjuntar otra propiedad a la sentencia transformada (es decir, **"attributedTo"**) con un valor apropiado (en este caso "Ralph Swick").

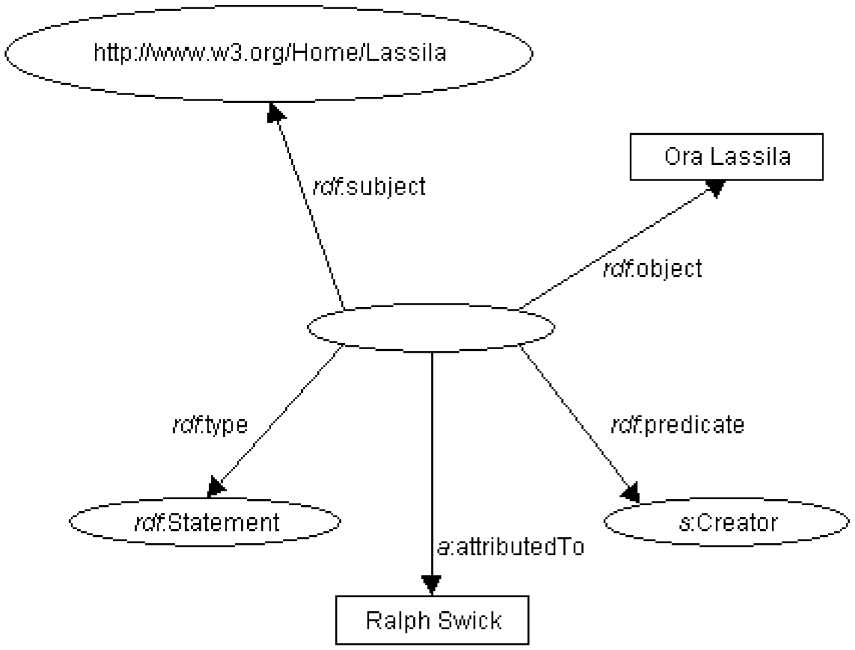


Figura 8. Representación de una sentencia transformada de lo abstracto a lo concreto

Modelo formal para RDF

Esta especificación presenta tres representaciones del modelo de datos; como 3-tuplas (triples), como gráfico, y en XML. Estas representaciones tienen un significado equivalente. El mapeo entre las representaciones que se utilizan en esta especificación no pretenden forzar de ninguna manera la representación interna usada en la implementación.

El modelo de datos de RDF se define formalmente como sigue:

1. Hay un conjunto denominado *Recursos [Resources]*.
2. Hay un conjunto denominado *Literales [Literals]*.
3. Hay un subconjunto *Recursos [Resources]* denominado *Propiedades [Properties]*.
4. Hay un conjunto denominado *Sentencias [Statements]*, cada elemento de los cuales es un triple de la forma: {pred, sub, obj}

Donde **pred** es una propiedad (miembro de *Properties*), **sub** es un recurso (miembro de *Resources*), y **obj** puede ser tanto un recurso como un literal (miembro de *Literals*).

Podemos ver un conjunto de sentencias (miembros de *Statements*) como un gráfico directamente etiquetado: cada recurso y literal es un vértice, un triple **{p, s, o}** es un arco de **s** hasta **o** etiquetado por **p**. Esto se ilustra en la figura 11.

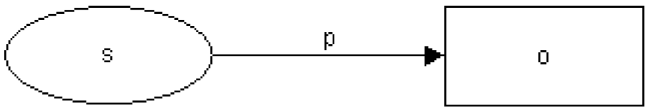


Figura 11. Plantilla gráfica de una sentencia simple

Esto puede leerse bien como:

o *es el valor de* ***p*** *para* ***s***

o como (de derecha a izquierda):

s *tiene una propiedad* ***p*** *con un valor* ***o***

o incluso:

la ***p*** *de* ***s*** *es* ***o***

Por ejemplo, la frase:

Ora Lassila es el creador [autor] del recurso http://www.w3.org/Home/Lassila

se representaría gráficamente como se presenta a continuación:

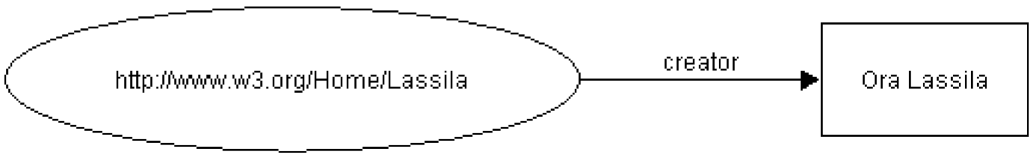


Figura 12. Gráfico de una sentencia simple

y el triple (miembro de *Statements*) correspondiente, sería

{creator, [http://www.w3.org/Home/Lassila], "Ora Lassila"}

La notación [*I*] denota el recurso identificado por la URI *I* y las comillas denotan un literal.

Utilizando los triples, podemos explicar qué sentencias están transformadas. Dada una sentencia:

{creator, [http://www.w3.org/Home/Lassila], "Ora Lassila"}

podemos expresar la transformación [reification] de esto como un nuevo recurso X, así:

{type, [X], [RDF:Statement]}
{predicate, [X], [creator]}
{subject, [X], [http://www.w3.org/Home/Lassila]}
{object, [X], "Ora Lassila"}

Desde el punto de vista de un procesador RDF, los hechos (es decir, las sentencias o *statements*) son triples que son miembros de *Statements*. Por lo tanto, la sentencia original permanece como un hecho a pesar de que se transforme [de lo abstracto a lo concreto], dado que el triple que representa la sentencia original permanece en *Statements*. Sencillamente hemos añadido cuatro triples más.

La propiedad denominada "**type**" se define para proporcionar la tipografía primigenia. La definición formal de **type** es:

1. Hay un elemento de *Propiedades [Properties]* conocido como **RDF:type.**
2. Miembros de *sentencias [Statements]* de la forma **{RDF:type, sub, obj}** pueden satisfacer lo siguiente: **sub** y **obj** son miembros de *Recursos [Resources]*. que colocan restricciones adicionales en el uso de **type**.

Además, la especificación formal que expresa la transformación de lo abstracto a lo concreto, es:

1. Hay un elemento de *Recursos [Resources]*, no contenido en *Propiedades [Properties]*, conocido como **RDF:Statement.**
2. Hay tres elementos en *Propiedades [Properties]* conocidos como **RDF:predicate**, **RDF:subject** y **RDF:object**.
3. La transformación [Reification] del triple **{pred, sub, obj}** de las sentencias es un elemento de *Recursos [Resources]* que representa el triple transformado y los elementos s_1, s_2, s_3, y s_4 de *Statements* así:

   s_1: {RDF:predicate, r, pred}
   s_2: {RDF:subject, r, subj}
   s_3: {RDF:object, r, obj}
   s_4: {RDF:type, r, [RDF:Statement]}

El recurso **r** en la definición anterior se llama sentencia transformada *[reified statement]*. Cuando un recurso representa una sentencia transformada; es decir, que tiene una propiedad **RDF:type** con un valor de **RDF:Statement**, entonces dicho recurso puede tener exactamente una propiedad **RDF:subject**, una propiedad **RDF:object**, y una propiedad **RDF:predicate**.

Como se describe en la sección 3, normalmente es necesario representar una colección de recursos o literales, por ejemplo para declarar que una propiedad tiene una secuencia ordenada de valores. RDF define tres tipos de colecciones: listas ordenadas, denominadas *Sequences*, listas desordenadas, denominadas *Bags*, y listas que representan alternativas para el valor (único) de una propiedad, denominadas *Alternatives*.

Formalmente, estos tres tipos de colecciones se definen por:

1. Hay tres elementos de *Recursos [Resources]*, no contenidos en *Propiedades [Properties]*, conocidos como **RDF: Seq, RDF:Bag**, y **RDF:Alt.**
2. Hay un subconjunto de *Propiedades [Properties]* que corresponden con los ordinales (1, 2, 3, ...) denominados ***Ord***. Nos referimos a los elementos de ***Ord*** como **RDF:_1, RDF:_2, RDF:_3, ...**

Para representar una colección ***c***, crear un triple **{RDF:type, *c*, *t*}** donde ***t*** es una de los tres tipos de colecciones RDF:Seq, RDF:Bag, o RDF:Alt. Los triples restantes {RDF:_1, *c*, r_1}, ..., {RDF:_n, *c*, r_n}, ... apuntan hacia cada uno de los miembros r_n de la colección. Para una colección individual el recurso puede ser como mucho un triple cuyo predicado es cualquier elemento dado de ***Ord*** y debe usarse en una secuencia que comience con RDF:_1. Para recursos que son objetos específicos de una categoría *[instances]* del tipo de colección **RDF: Alt,** debe ser exactamente un triple cuyo predicado es: **RDF:_1** y que es el valor por defecto para recursos alternativos (esto es, debe tener siempre al menos una alternativa).

Conclusión

Hemos visto a RDF como un modelo semántico para la representación del contenido web, y hemos dejado entrever las posibilidades que conllevan su modelo y sintaxis, así como la flexibilidad de su esquema.

Aunque ya es una recomendación del W3C, RDF parece más bien un estándar futuro para las bibliotecas digitales.

Por otro lado, vemos la necesidad actual de una restricción estructural para proporcionar métodos inequívocos de expresión semántica. RDF no es más que la infraestructura que permite esa restricción gracias a la codificación, intercambio y la reutilización de metadatos estructurados.

El modelo RDF se está pues, desarrollando para mediar ante esa gran cantidad de modelos de metadatos surgidos en los últimos años y cubrir las necesidades de los vendedores y proveedores de información y, en mi opinión, convertirse en el componente esencial de cualquier biblioteca digital efectiva.

Sin embargo, y pese a los últimos trabajos del W3C, muchas de las páginas web existentes en HTML no son documentos XML bien formados. Por ello, el modelo semántico del RDF sólo es aplicable a sistemas de información con propósitos específicos.

Diferencias entre RDF y XML

RDF	XML
• El **modelo** RDF y el modelo XML son fundamentalmente diferentes.	
• RDF tiene un modelo muy simple consistente en *arcos etiquetados*. • Cualquier conjunto específico de declaraciones RDF forma un *grafo* que puede serializarse en XML.	El modelo de datos XML es un *árbol etiquetado* orientado a marcas de texto. Son *menos flexibles* para expresar metadatos.
• Los **recursos** RDF utilizados en RDF y en los esquemas XML son fundamentalmente diferentes.	
• En RDF, los *nodos* no son nodos dentro del documento mismo, sino cualquier *recurso que tiene una URI*, y la mayoría de las veces reside fuera del propio documento RDF. Así, RDF se diseña para ser un lenguaje de *metadatos*.	Los nodos a los que se refiere un esquema XML son nodos *dentro de un documento XML*, en un lugar específico en la estructura de un documento.
• La **semántica** de los esquemas XML y RDF es fundamentalmente diferente.	
• RDF tiene principalmente una interpretación *semántica*. • RDF se utiliza para construir [modelar] conocimiento, donde las representaciones basadas en las estructuras de árbol no son suficientes.	Los esquemas XML tienen principalmente una interpretación *sintáctica*, restringiendo el conjunto de documentos XML que pueden elaborarse. Los esquemas XML se utilizan para modelar documentos XML.

La diferencia puede formularse de esta manera:

XML y el esquema XML es un lenguaje para modelar datos, y RDF es un lenguaje para modelar metadatos. Cuando los metadatos necesitan ser codificados como datos, una sintaxis XML es muy útil. Sin embargo, modelar metadatos en puro XML restringe severamente su flexibilidad.

Parte II

Introducción

Un ámbito en el cual se presentan problemas de portabilidad de contenidos es el ámbito universitario, para el cual queremos analizar la posibilidad de aplicación de los estándares analizados anteriormente.

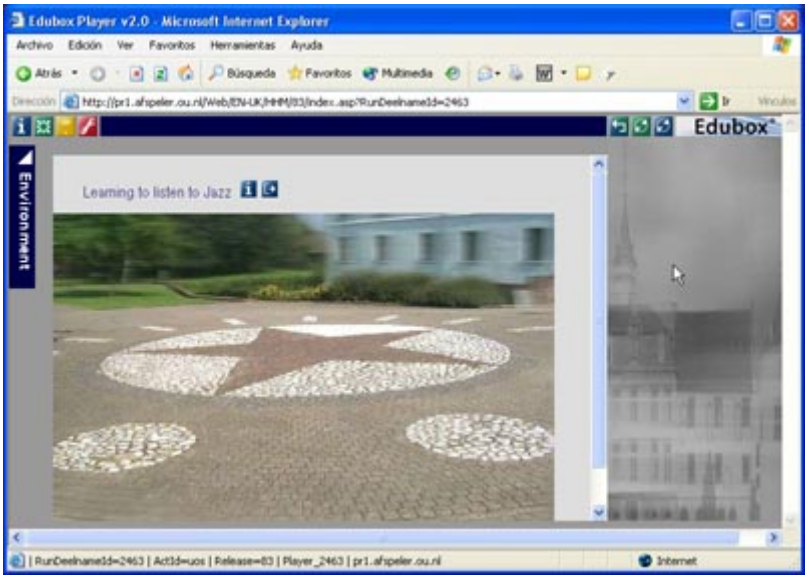
La posibilidad de cooperación entre universidades que se debía a la, casi fortuita, posesión de una plataforma común, hoy gracias a IMS y también a SCORM, el número de universidades que se ven beneficiadas va en crecimiento, ya que ellas podrán intercambiar información de sus alumnos, contenidos, evaluaciones y la estructura del curso manteniendo sus plataformas.

Esto nos lleva a reflexionar que muchas universidades se verán beneficiadas, ya que la migración entre plataformas y la actualización de las mismas no será tan costosa. Además, resulta importante motivar a las universidades que han optado por construir sus propias soluciones a participar de las especificaciones IMS y SCORM, ya que integrar sus sistemas a los estándares harán que sus LMS pasen a pertenecer a un mundo en el cual es posible compartir recursos educativos, tal como fue concebida Internet: como una fuente de intercambio de información.

Veamos algunos proyectos basados en estos estándares:

Proyecto Edubox Player v2.0

Como ejemplo de implementación vamos a ver el Edubox Player v 2.0.



Edubox Player v.2.0

Esta implementación está disponible para todos los usuarios registrados en la página de EML de la Open University of the Netherlands. En ella podemos aprender a escuchar Jazz. Para ello hay una serie de actividades, pruebas, exámenes y ejemplos sobre los distintos ritmos de Jazz.

Para poder seguir este curso se nos dan dos rutas posibles. Una es siguiendo el recorrido histórico del Jazz y la otra tratando los diferentes estilos. Esto nos da una idea de que se puede aprender un conocimiento de distintas. Desde el primer momento nos damos cuenta de la influencia de EML en el método de aprendizaje. En todo momento tenemos accesible un panel con las actividades que podemos realizar para adquirir el conocimiento requerido. Esto lo podemos ver en la figura 1.

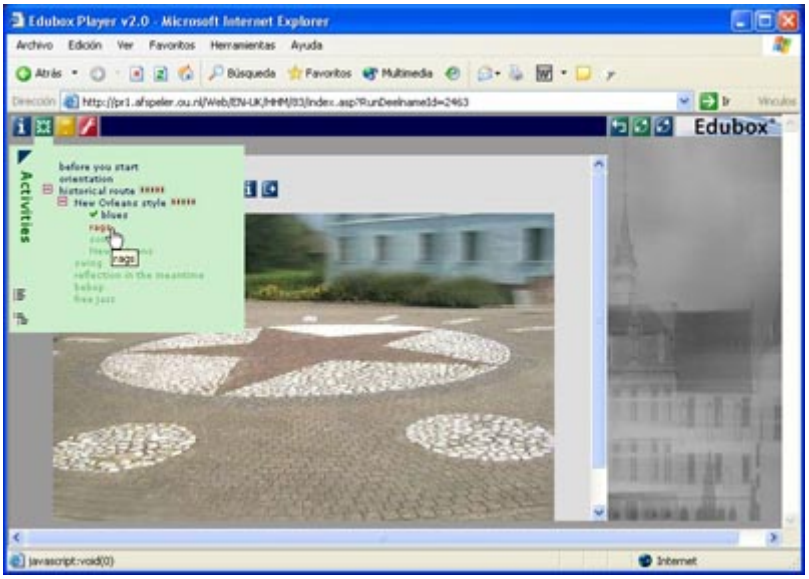


Figura 1. Panel de actividades de Edubox

En la figura 2 podemos observar como el Edubox Player nos permite escuchar ejemplos de los ritmos de Jazz a través de su página. Para ello no tenemos más que tener configurado el programa Real Player en nuestra PC.

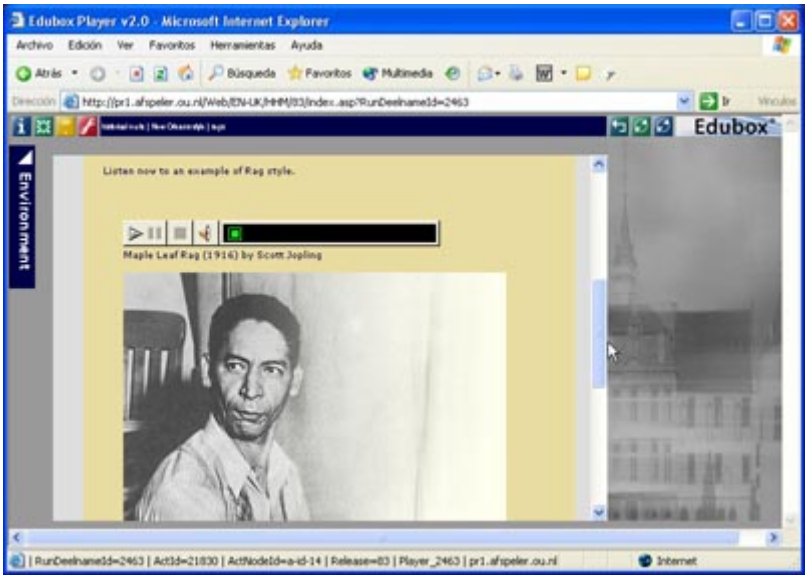


Figura 2. Ejemplos en Real Player

Del mismo modo que en la imagen anterior se nos presentaban ejemplos reales de sonido de los ritmos de Jazz, ahora vamos a ver los sofisticados exámenes del Edubox Player. En el examen que ponemos como ejemplo en la figura 3, escuchamos un ritmo de Jazz a través del Real Player y a nuestra derecha elegimos el estilo al cual pertenece. Además de esto, se nos presentan ciertas ayudas para poder realizar nuestros exámenes así como la historia de lo que hemos ido haciendo en el curso.

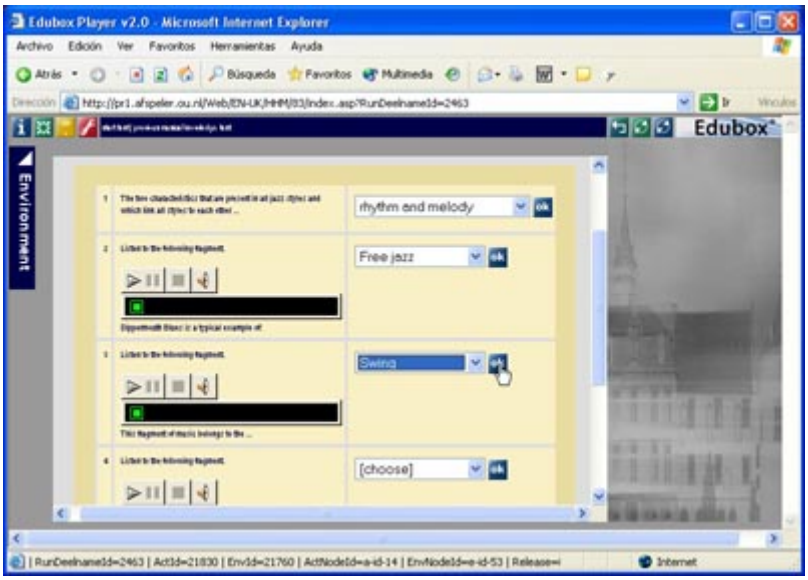


Figura 3. Exámenes en el Edubox Player 2.0

Además el Edubox Player trae más facilidades como por ejemplo una historia con las referencias a internet donde podemos encontrar información sobre los temas expuestos en los contenidos. Otra facilidad es un glosario de términos donde se nos describen las palabras clave (algo muy importante en la especificación de EML) en términos comprensibles por el estudiante. Esto lo podemos ver en las figuras 4 y 5.

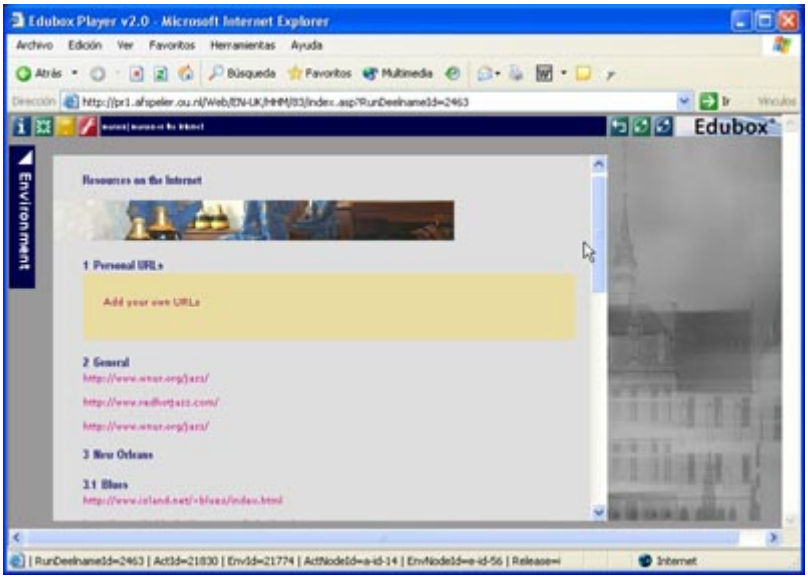


Figura 4. Historia de direcciones de internet

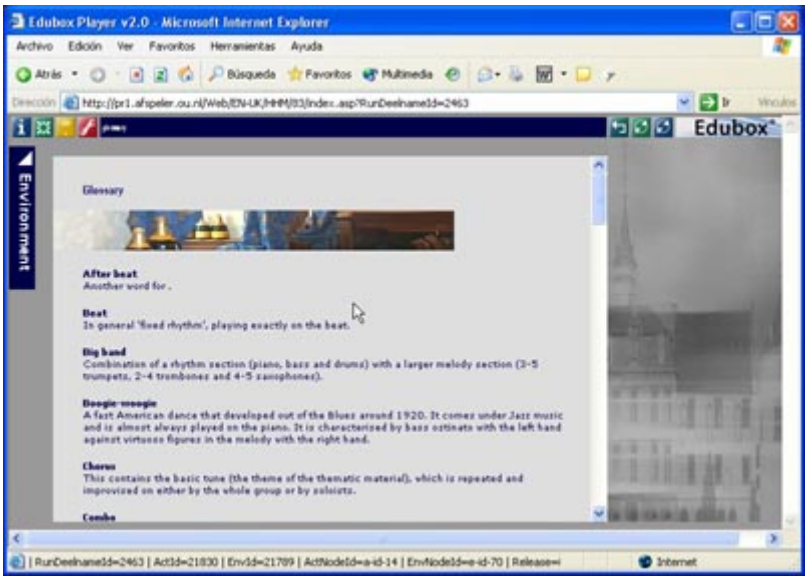


Figura 5. Glosario de términos

En la figura 6 podemos ver el panel de Enviroment en el que se nos presentan distintos entornos para realizar la actividad elegida.

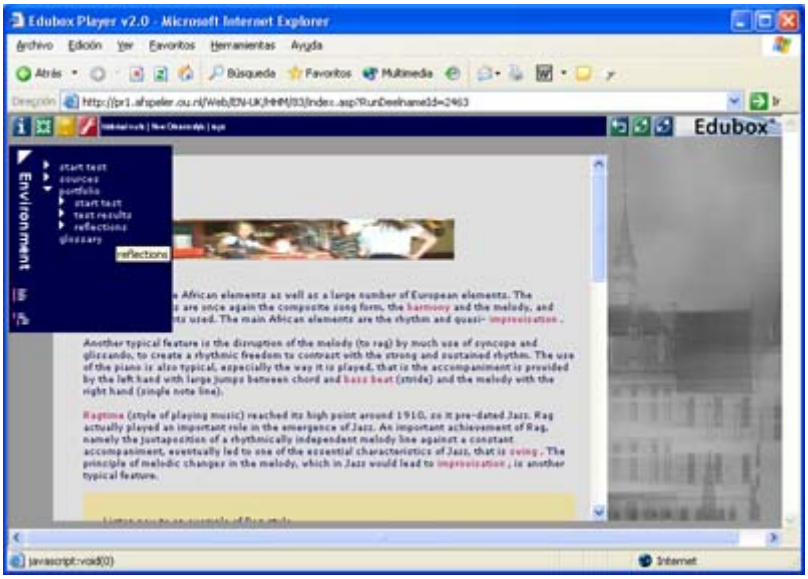


Figura 6. Panel de Enviroment

Conclusión

Podemos observar que en Edubox Player 2.0 el aprendizaje presenta características de personalización, como lo propone EML. Vemos que el alumno no está obligado a hacer un determinado recorrido del curso, que las herramientas no son de uso obligatorio y que el panel de Enviroment permite configurar el entorno de aprendizaje.

También podemos observar la simplicidad de la interface permitiendo al alumno concentrarse en la actividad que está llevando a cabo sin distracciones provocadas por el contenido presentado en la interfaz.

Proyecto <e-aula>

<e-aula> es un proyecto que tiene como uno de sus objetivos prioritarios el desarrollo de un entorno de investigación que permita evaluar diferentes propuestas de estandarización. Más concretamente, se pretende analizar el potencial de los estándares educativos en la construcción de entornos de aprendizaje personalizado, mediante un conjunto de sistemas que implementen la infraestructura básica (ej. creación, acceso y reutilización de los contenidos).

El sistema <e-aula> sirvió en su comienzo como evaluador de las especificaciones IMS y EML mediante la implementación de funciones básicas de cada uno de ellos. Actualmente, IMS ha englobado a EML, por lo que <e-aula> se ha centrado únicamente en IMS, y ha aumentado el número de funciones implementadas, poniendo así a prueba la propuesta IMS en un sistema real de enseñanza con aplicación en la universidad.

El objetivo de IMS de definir especificaciones que hagan posible la interoperabilidad de aplicaciones y servicios de enseñanza distribuida, se ha concretado, al día de hoy, en diez especificaciones. Cada una de ellas está enfocada a una necesidad distinta del proceso de enseñanza. En el proyecto <e-aula>, hay cuatro de ellas que se están evaluando e implementando:

- Meta Data Specification (Versión 1.2.1. Final Release).
- IMS Question & Test Interoperability Specification (Versión 1.2. Final Release).
- IMS Learner Information Package Specification (Versión 1.0.0. Final Release).
- IMS Content Packaging (Versión 1.1.2. Final Release).

Evaluación de las diferentes especificaciones de IMS con el proyecto <e-aula>

Trascripción del documento presentado por los desarrolladores

En este apartado se muestran los problemas y mejoras de las especificaciones IMS con que se han encontrado los desarrolladores del proyecto <e-aula>. Se encuentran agrupados según la especificación a la que afectan. (Este análisis fue realizado por el Departamento de Sistemas Informáticos y Programación de la Facultad de Informática de la Universidad Complutense de Madrid)

1 - IMS Meta Data Specification (MD)

En el proyecto <e-aula> se implementa esta especificación en su totalidad. Los desarrolladores han indicado como aspecto negativo de la especificación a la hora de implementarla, la falta de campos orientados a la formación pedagógica. Este problema ha sido resuelto con la incorporación de EML.

2 - IMS Question & Test Interoperability Specification (QTI)

En <e-aula> los cursos no sólo incorporan evaluaciones sino que además se ha creado una herramienta de autoría de exámenes.

QTI ofrece una descripción y representación muy detallada de los distintos tipos de preguntas que se pueden incorporar en una prueba de evaluación. Define el elemento pregunta, que contiene toda la información referente a una cuestión (pregunta en sí, posibles respuestas y respuesta correcta), pero se encuentra con un problema a la hora de catalogar estos elementos.

Esta clasificación no puede hacerse según la información que almacenan ya que cada cuestión y sus posibles respuestas tienen una estructura propia, y además, cada cuestión planteada puede requerir de una o más respuestas distintas. Tampoco se los puede clasificar según la pregunta que plantean porque resultaría demasiado amplio, y nos encontraríamos con un problema mayor: la forma de redactar la misma pregunta de cada autor.

Para solucionar este problema, en <e-aula> se ha optado por clasificarlas por el tipo de respuesta que ofrecen (no en contenido, sino teniendo en cuenta la estructura de la respuesta). Por ejemplo, varias respuestas con una sola de ellas válida, o respuestas de ordenación de elementos.

La especificación ofrece una gran cantidad de tipos de respuestas, lo que la hace muy versátil, pero el problema es que la mayoría de ellas no serán utilizadas en un entorno real por resultar muy enrevesadas, y, en cambio, han faltado recomendaciones para la corrección de las respuestas de texto libre (aquellas donde el alumno ofrece una respuesta sin restricciones, en la que puede escribir libremente), que están entre las más comunes. En el documento "ASI Best Practice and Implementation Guide" (http://www.imsglobal.org/question/qtiv1p2/imsqti_asi_bestv1p2.html) sería conveniente incluir un apartado dedicado en su totalidad a guiar a los desarrolladores en la forma de corrección de este tipo de respuestas, que por su naturaleza resultan diferentes al resto ya que necesitan la ayuda de un tutor para ser corregidas.

En <e-aula> se está trabajando en la opción de paralizar la corrección del examen hasta que el tutor califique las preguntas de texto libre, y retomarla entonces.

3 - IMS Learner Information Package Specification (LIP)

En <e-aula> se han incluido ficheros de perfil de los usuarios del sistema.

Como sabemos, dentro de los ficheros de personalización de los alumnos se guarda el historial de la navegación que el alumno ha realizado en cada curso. Esto implica una referencia a cada uno de los archivos visitados. El problema aparece en el momento que se quiere modificar un curso. Si se añaden o eliminan archivos, o si se alteran las referencias entre ellos, la organización del curso cambia. Esto se verá reflejado en el manifiesto del curso según la especificación "Content Packaging", pero no ocurrirá lo mismo en el fichero de personalización del alumno, que no registra la modificación de la estructura del curso. De esta manera, todos los archivos de los alumnos que hayan visitado un curso que haya modificado su estructura pasan a ser inservibles.

Como se puede intuir, este problema tiene una difícil solución. El hecho de que la modificación de un curso conlleve que haya que desechar todos los ficheros de personalización de alumnos que lo hayan visitado, o como mínimo, eliminar todos los árboles de navegación que lo contengan, supone un precio demasiado alto que ningún sistema debe estar dispuesto a pagar. Todas las soluciones barajadas pasan por restricciones a la hora de modificar los cursos. Una posible solución obliga a que cada curso modificado se diera de alta como un nuevo curso, para no contaminar así el árbol de navegación. El problema es que, de esta manera, se obliga a un alumno a volver a pasar por el mismo sitio por dónde ya había pasado en el curso anterior con cualquier cambio realizado en el curso, por pequeño que fuera. Por tanto, el concepto de prerrequisitos perdía totalmente su consistencia.

En <e-aula> se ha decidido implementar el sistema de la siguiente manera:

1. Los contenidos existentes de los cursos son inamovibles. Es decir, los contenidos que ya estaban no se pueden ni eliminar ni modificar. Lo que sí es posible, es añadir partes a un curso, siempre que esto no suponga una alteración de su organización.

2. Los prerrequisitos sólo pueden añadirse en una dirección, desde el material nuevo al antiguo. Por tanto, si se añade material, este sí puede tener dependencias con el material que ya existía, pero nunca al contrario.

Con esto, <e-aula> apuesta por mantener de forma íntegra el seguimiento de los cursos, a cambio de unas ciertas restricciones en sus posibles ampliaciones.

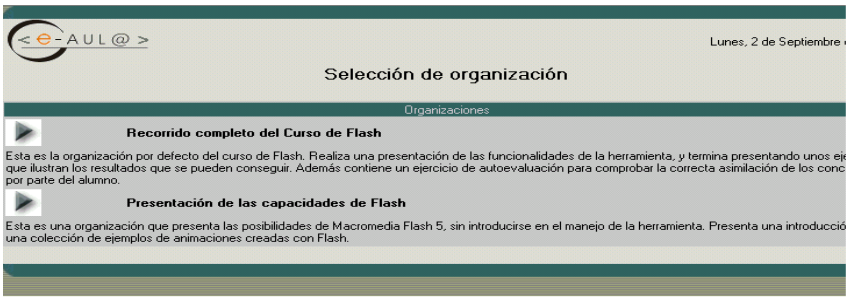


Figura 1. Elección de la organización en <e-aula> IMS.

Existe otra posible solución al problema (planteada por los desarrolladores de <e-aula>) que, aunque de mayor complejidad, lo resolvería en su totalidad. La idea se basa en un patrón de diseño para aplicaciones hipermedia propuesto para gestionar un historial de navegación, el patrón Navegational Observer [Rossi 1996]. Tomando como base este patrón, llegaríamos a la necesidad de un cambio en las especificaciones LIP y CP de IMS. La organización y el histórico no irían embebidos en cada especificación, sino que ambas llevarían referencias a un fichero independiente que se encargaría de ambas cosas. De esta manera, el cambio en las organizaciones de los cursos no afectaría al LIP, ya que, con cambiar este fichero independiente sería suficiente. En resumen, la idea es que ambas especificaciones compartan una pizarra dónde escribir sus cambios.

Esta idea no ha sido implementada aún en <e-aula>, sino sólo debatida.

4 - IMS Content Packaging (CP)

Esta especificación plantea tres posibilidades de adaptación que se han implementado en <e-aula>:

1. Definición de organizaciones alternativas para presentar el contenido de los cursos. El término "organización" que se utiliza en la especificación, hace referencia a la estructuración de los contenidos a modo de índice o tabla de navegación que, o bien se presentan directamente al alumno para que él mismo elija la presentación más adecuada a sus objetivos (figura 2), o bien es el sistema quien selecciona automáticamente la más adecuada en función de la información disponible sobre el alumno (normalmente en base a su nivel de conocimiento).
2. Definición de prerrequisitos de acceso a determinados recursos u objetos. En este sentido es habitual requerir el acceso previo a algún otro objeto, o demostrar un nivel de conocimiento determinado en función de las calificaciones obtenidas por medio de algún sistema de evaluación. Esto permite generar dinámicamente la estructura presentada al alumno en función de su interacción con el entorno educativo.
3. El sistema permite mostrar la información con distinto nivel de detalle en función del conocimiento especificado por el usuario (tres niveles de detalle -alto, medio y bajo-). Por el momento, la elección del nivel de conocimiento se realiza manualmente por el alumno, aunque en estos momentos el sistema está siendo modificado para adaptar automáticamente el contenido mostrado en función de los resultados obtenidos en una evaluación previa (todo el sistema de evaluación cumple con la especificación QTI del

grupo IMS). Las figuras 2 y 3 muestran el contenido del mismo objeto presentado a alumnos con niveles de conocimiento alto y bajo respectivamente.

Figura 2. Contenidos mostrados al alumno con nivel de conocimiento alto.

Figura 3. Contenidos mostrados al alumno con nivel de conocimiento bajo.

A continuación se comentan dos mejoras del sistema <e-aula> que no se contemplan en IMS.

Ambas están basadas en el uso dinámico que se le da a XML en nuestro proyecto, y que suponen un gran avance tanto en interoperabilidad cómo en reusabilidad.

- La primera es la utilización del manifiesto propuesto en IMS CP para visualizar el árbol de navegación del curso. En <e-aula> se utiliza una transformación XSLT que transforma el imsmanifest.xml en algo más reducido y preparado para que el sistema lo utilice cómo árbol de navegación (figura 4).

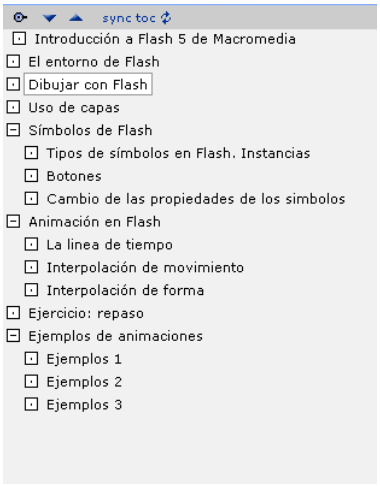


Figura 4. Árbol de navegación de un curso en <e aula>

- La segunda supone una alternativa respecto de la recomendación del uso de HTML para marcar los contenidos de los cursos que hace IMS [IMSLD_INFO]. En <e-aula>, se prefirió utilizar XML en lugar de HTML para marcar los contenidos de los cursos por las siguientes razones: se gana en interoperabilidad de contenidos, la transformación de XML en cualquier otro formato (no necesariamente HTML) es sencilla mediante XSL o FO (formatting objects), la visualización de los contenidos no dependerá del navegador. Por estas razones, en <e-aula> se ha apostado por contenidos XML, que se mandan, junto con hojas de estilo XSL que los transforman, para facilitar su visualización.

Conclusiones

La proliferación de aplicaciones e-learning desarrolladas según el modelo de objetos educativos está siendo acompañada de un proceso de estandarización (en el que están implicados muchas organizaciones, consorcios y proyectos) de los distintos aspectos de la tecnología que permitiría, entre otras cosas, conseguir contenidos reutilizables entre sistemas y plataformas.

El objetivo principal de la primera fase del proyecto <e-aula> fue establecer una valoración del potencial aportado por dos de estas iniciativas de estandarización (LOM/IMS y EML) desde los dos puntos de vista que son clave para el éxito de las aplicaciones e-learning: la capacidad para definir mecanismos que permitan reutilización/autoría de los contenidos educativos y la posibilidad de adaptar el proceso de aprendizaje a las necesidades específicas de los alumnos.

En una segunda fase, se ha centrado en la iniciativa IMS, y en la última versión de la especificación Learning Design.

Como objetivos para la siguiente fase del proyecto <e-aula>, caben destacar:

- La incorporación de mecanismos de modelado de usuario [Fernández-Manjón 1998] relacionados con un sistema de competencias asociado a los objetos educativos.
- Inclusión dentro de los metadatos de información relativa al dominio de conocimiento, por medio de taxonomías que permitan clasificar los objetos de acuerdo a su contexto de aplicación.
- Inclusión de otras especificaciones de IMS. IMS Learning Design está siendo probada en el sistema para añadirle sentido pedagógico. También

se baraja el incluir IMS Simple Sequencing para evaluar sus propuestas de secuenciamiento de contenidos.
- Por último, lograr que la herramienta de autoría de exámenes pueda ser ejecutada de una manera local, para evitar que el autor tenga que estar conectado al sistema durante toda la creación del examen.

Prototipo Final

- <e-Aula>. Prototipo Final basado en estándares IMS con curso de UML y Photoshop (en inglés).

Prototipos Anteriores

- <e-Aula>. Prototipo y Curso de creación de interfaces gráficas de usuario en Java. Sistema basado en EML
- <e-Aula>. Creación de un entorno de clase virtual en Internet. Prototipo y curso de IMS.

Parte III

***Introducción**:*

Como hemos visto en la Parte I de esta tesis, internet tiene el potencial de revolucionar la educación y la educación tiene el potencial de revolucionar Internet. Cuando se usa Internet con fines educativos encontramos que es difícil obtener información relevante al tema que nos interesa. Como intento de solución a esta problemática, hemos visto distintos metadatos y especificaciones de varias organizaciones que están trabajando en esta área.

Esta misma problemática se puede acotar a una determinada comunidad educativa. Como puede ser una universidad, una facultad, un laboratorio, etc. En estos ambientes la disponibilidad del material educativo no es siempre la deseada, así como la accesibilidad y el mantenimiento de los mismos.

En estos ámbitos también podemos aplicar metadatos y especificaciones sobre e-learning que garanticen los objetivos de accesibilidad, interoperabilidad, durabilidad y reutilización de los materiales didácticos basados en Web.

A continuación haremos un análisis más a fondo de SCORM a fin de lograr la construcción de contenido educativo que pueda ser ejecutado en diferentes LMS que implementen SCORM 1.2.

Guías para la definición e implementación:

- Scorm 1.2

Para construir nuestro contenido, debemos comenzar por recordar algunos conceptos:

Objeto de Aprendizaj*e* (Learning Object): Es una entidad digital o no digital, el cual puede ser usado, rehusado o referenciado durante el aprendizaje soportado por la tecnología. Existen tres características básicas de un objeto de aprendizaje: Accesibilidad, Reusabilidad/Adaptabilidad e Interoperabilidad.

Como vimos en parte I de este trabajo Scorm, utiliza tres componentes básicos: assets, SCOs y cursos.

LMS*:* Entorno de ejecución capaz de mantener información sobre el usuario, de lanzar objetos educativos y comunicarse con ellos, y de interpretar instrucciones sobre secuenciación entre objetos.

Propuesta

Crear objetos de aprendizaje que puedan ser usados, reusados, ejecutados y visualizados en distintos LMS.

Assets o Sharable Resources.(Contenido compartido)

Son recursos básicos que son compartidos entre distintos SCOs. Un Asset debe tener los meta-data apropiados que lo hagan identificable en el repositorio.

Sharable Content Object SCO (Clase, asociación de contenidos)

Asociación de recursos o Assets. Se incluye una especificación indicando cual será el asset principal, que será llamado por el LMS. El SCO es el componente del nivel más inferior que podría ser usado en un curso. Por lo tanto el SCO debe proveer contenido de aprendizaje por sí mismo, y debe poder ser ejecutado por el LMS.

Un SCO nunca debe linkear directamente a otro SCO o elemento externo al propio SCO. Ya que para ser reusable el SCO debe ser independiente del contexto de estudio.

Un SCO debe ser descrito con un SCO Meta-Data

SCORM Content Aggregation (Curso, asociación de clases)

Es una colección de de SCOs asociados a través del **SCORM** Manifiest File. Es un mapa que ha de ser utilizado para registrar los recursos educativos dentro de unidades de instrucciones (curso, capítulo, módulo, etc.), que aplican y asocian las taxonomías educativas. Éste puede ser referido en el Content Aggregation Meta-data.

El mecanismo para enlazar los contenidos puestos en el Content Aggregation Meta-data es el Content Package.

Para ejecutar el contenido educativo, un LMS debe proveer las siguientes funciones:

- Importación del contenido.
- Navegación del contenido

Cómo debe ser la estructura de nuestro contenido

Content Packaging

Es la información descriptiva del curso que se encuentra en el Manifiest File, el cual es llamado imsmanifiest.xml. Este archivo básicamente trata tres características:

1. Descripción del curso
2. Secuencia del curso
3. Recursos del curso

IMSMANIFIEST.XML

Este archivo debe ser localizado en el nivel superior de los directorios del curso. El archivo presenta 4 secciones.

1. Sección preamble, contiene la información requerida por el LMS y el XML para chequear y validar este archivo.
2. Sección <metadata>, guarda la descripción del curso. Esta sección puede ir vacía, pero esto impedirá su búsqueda en los repositorios, por lo tanto, si uno desea que sea accesible debe ser incluida apropiadamente.
3. Sección <organizations>, describe la secuencia de SCOs que conforman el curso. Cada uno de estos elementos se vinculan con un recurso por medio del atributo identifierref que se encuentra dentro de la sección resources.
4. Sección <resources>, contiene la lista de archivos utilizados por cada SCO.

Cada Sección inicia con un único tag xml y debe ser formado de la siguiente manera.

- PREAMBLE

  <manifest identifier = "MANIFEST" .... >

- META-DATA

  <metadata>

  ... (title, keywords, etc.)

  </metadata>

- ORGANIZATION

  <organizations default = "Linear">

  ... descripción de la secuencia del curso (ítems)

  </organizations>

- RESOURCES

  <resources>

  ... metadata de los recursos y localización de los archivos usados en el curso

  </resources>

Veamos un ejemplo para la sección <organizations> - Veremos la Forma Jerárquica, que es la más utilizada.

Esta sección describe el paso de SCO a SCO y relaciona esta secuencia con los archivos del curso. EL LMS usa esta información para armar la tabla de contenidos del curso.

```
<organizations default = "MyCourse">
  <organization identifier = "MyCourse">
    <item identifier = "TOC1" identifierref = "FirstSCO">
      <title>Introduction to Flying</title>
    </item>
    <item identifier="TOC2">
      <title>Flight Navigation</title>
```

```
        <item identifier = "TOC2a" identifierref = "SecondSCO">
          <title>Flight Rules </title>
        </item>
        <item identifier = "TOC2b" identifierref = "ThirdSCO">
          <title>Electronic Navigation </title>
        </item>
      </item>
    </organization>
</organizations>
```

Sección <resources> -

La sección <resources> vincula la secuencia descripta en <organizations> con los materiales del curso. Aquí también se describe la lista de todos los materiales del curso, incluyendo páginas webs, imágenes, archivos multimedia etc.

```
<organizations default = "MyCourse">
  <organization identifier = "MyCourse">
    <item identifier = "TOC1" identifierref = "FirstSCO">
      <title>SCORM Packaging </title>
    </item>
    <item identifier = "TOC2" identifierref = "SecondSCO">
      <title>SCORM Metadata</title>
    </item>
  </organization>
</organizations>

<resources>
  <resource identifier = "FirstSCO" type = "webcontent" adlcp:scormtype = "sco" href="index.html">
    <file href = "index.html"/>
    <file href = "end.html"/>
  </resource>
  <resource identifier = "SecondSCO" type = "webcontent" adlcp:scormtype = "sco" href="page1.html">
    <file href = "page1.html"/>
    <file href = "page2.html"/>
    <file href = "page3.html"/>
    <file href = "fig1.gif"/>
    <file href = "myJavascript.js"/>
  </resource>
</resources>
```

Como se ve en el ejemplo anterior en el elemento <item>, el atributo identifierref establece la relación con el recurso, elemento <resource>, atributo de relación identifier.

En el caso que se utilicen los mismos materiales por varios SCOs, se debe definir una relación de dependencia. Por Ejemplo

```
<resources>
  <resource identifier = "FirstSCO" type = "webcontent" adlcp:scormtype = "sco" href="index.html">
    <file href = "index.html"/>
    <file href = "end.html"/>
    <dependency identifierref = "sharedFiles"/>
  </resource>
  <resource identifier = "SecondSCO" type = "webcontent" adlcp:scormtype = "sco" href="page1.html">
    <file href = "page1.html"/>
    <file href = "page2.html"/>
    <file href = "page3.html"/>
    <dependency identifierref = "sharedFiles"/>
```

```
  </resource>
  <resource identifier = "sharedFiles" type = "webcontent" adlcp:scormtype = "asset" href="fig1.gif">
    <file href = "fig1.gif"/>
    <file href = "myJavascript.js"/>
  </resource>
</resources>
```

Aquí el FirstSCO y SecondSCO, comparten ciertos archivos, por lo que se generó un recurso independiente al cual linkean.

¿Cómo se implementa la navegación sobre el material?

Según hemos visto en la parte I del trabajo, SCORM utiliza un API para la comunicación ente el SCO y el LMS.

Hemos visto que SCORM utiliza un entorno de ejecución que incluye:

- protocolo específico para la ejecución de contenidos Web.
- un API entre el contenido y el LMS
- un modelo de datos que define el flujo de datos intercambiado entre el entorno LMS y el contenido que se ejecuta en el entorno de ejecución.

Comunicación del SCO y el LMS mediante el API

La comunicación de un SCO con el LMS mediante el API adapter pasa por varios estados para una instancia determinada del SCO durante el tiempo de ejecución.

Los distintos estados del API adapter especifican una respuesta determinada del API adapter para un input dado. Durante cada uno de estos estados hay diferentes actividades por donde un SCO puede pasar.

Los estados en los que puede encontrarse el API son: no inicializado, inicializado y finalizado.

-<u>No inicializado</u>:

Describe el estado del SCO que está entre el lanzamiento real y la ejecución de LMSInitialize (“”). Durante este estado el SCO tiene la responsabilidad de encontrar el API adapter suministrado por el LMS. Una vez que el API Adapter ha sido encontrado por el SCO, se permite que el SCO invoque las siguientes funciones del API

- LMSInitialize(“”)
- LMSGetLastError()
- LMSGetErrorString()
- LMSGetDiagnostic()

-<u>Inicializado</u>:

Describe el estado en que el SCO está situado entre la ejecución de LMSInitialize (“”) y LMSFinish (“”).

-Finalizado:

Describe el estado en que el SCO ha llamado a la función LMSFinish (“”). Si al ser invocado LMSFinish (“”), el API adapter devuelve “false” entonces al SCO se le permite llamar a:

- LMSGetLastError()
- LMSGetErrorString()
- LMSGetDiagnostic()

Si el API adapter devuelve “false” no hay garantía de que éste responda correctamente a cualquier función.

Uso del código de error del API

El SCO debe tener una forma de valorar si cualquier función de llamada al API se ha ejecutado con éxito y si no ha sido así, determinar cuál fue el error.

Con la función LMSGetLastError() averiguamos el número de error de la función anterior.

Los distintos valores que esta función puede devolver son:

Cód.	Descripción	Utilidad
“0”	No hay error	No se ha encontrado ningún error. La función se ha ejecutado perfectamente
“101”	Excepción general	Usado para indicar funciones generales.
“201”	Argumento inválido	Se usa cuando hay una llamada a una función del modelo de datos del SCORM Run Time Environment que no existe. Se usa cuando un argumento inválido pasa por el API
“202”	Los elementos no pueden tener hijos	Se usa cuando se llama a LMSGetValue() en cualquier modelo de datos que no soporte hijos.
“203”	El elemento no es un array y por tanto no se puede ejecutar “count”	Se usa cuando se llama a LMSGetValue() en cualquier categoría o elemento de un modelo de datos que no soporte count.
“301”	No inicializado	Se usa cuando se llama a cualquier función del API antes de ejecutar LMSInitialize(“”)

“401”	Error no implementado	Se usa cuando se hace una llamada al modelo de datos que no es soportada por el LMS o si se está usando otro modelo de datos fuera del SCORM Run-Time Environment Data Model.
“402”	El elemento es una palabra clave y por tanto no se le puede asignar dicho valor	Se usa cuando una llamada a LMSSetValue() implica utilizar una palabra clave.
“403”	El elemento es de sólo lectura	Se usa cuando una llamada a LMSSetValue() implica utilizar un elemento de sólo lectura.
“404”	El elemento es de sólo escritura	Se usa cuando una llamada a LMSSetValue() implica utilizar un elemento de solo escritura.
“405”	Tipo de dato incorrecto	Se usa cuando se hace un intento de asignar un valor a una variable que no corresponde con su tipo.

Reglas generales del API

La siguiente lista resume las reglas generales del API:

- En los nombres de las funciones hay diferencias entre mayúsculas y minúsculas (case-sensitive) y deben expresarse siempre tal y como son mostrados aquí.
- Idem con los parámetros y argumentos. Los parámetros siempre se expresan en minúsculas.
- Cada llamada a una función del API borra el código de error (a no ser que la función llamada sea de administración de errores).

Responsabilidades del LMS

Las responsabilidades a cargo del LMS al momento de trabajar con el adaptador API son las siguientes:

- El LMS debe lanzar el SCO en una ventana contenida en un marco principal donde esté el API adapter.
- El API adapter debe ser suministrado por el LMS.
- El acceso al API desde el SCO debe ser mediante Java-Script
- El adaptador API debe ser accesible vía DOM como un objeto llamado “API”

Encontrar el API

El SCO tiene la responsabilidad de enviar la llamada a LMSInitialize(“”) y LMSFinish(“”) al API. Para hacer esto el contenido debe ser capaz de localizar el API adapter que se presenta con el LMS.

Es responsabilidad del LMS suministrar un API adapter en la ventana que contenga la jerarquía DOM de forma que el SCO pueda buscar el marco principal de forma recursiva y/o abrir la ventana con la jerarquía DOM para encontrar el API.

Es responsabilidad del contenido encontrar y establecer comunicaciones con el adaptador API del LMS. La forma de hacerlo no está estandarizada por SCORM.

Veamos cómo sería la implementación para que el SCO encuentre el API:

```
var MAX_PARENTS_TO_SEARCH = 500;

/*
ScanParentsForApi
-Busca todos los parents de la ventana hasta encontrar un objeto
llamado "API". Si se encuentra un objeto con este nombre, se devuelve
una referencia al mismo. De lo contrario la function retorna null.
*/
function ScanParentsForApi(win)
{
      var nParentsSearched = 0;

      while ((win.API== null) &&
                  (win.parent != null) && (win.parent != win) &&
                  (nParentsSearched <= MAX_PARENTS_TO_SEARCH))
      {     nParentsSearched++;
            win = win.parent;
      }

      return win.API;
}

function GetAPI()
{
      var API = null;

      if ((window.parent != null) && (window.parent != window))
      {
            API = ScanParentsForApi(window.parent);
      }
      if ((API == null) && (window.top.opener != null))
      {
            API = ScanParentsForApi(window.top.opener);
      }

      return API;
}

      Veamos un ejemplo de uso desde un SCO

var objScormApi;

objScormApi = GetApi();

if (objScormApi == null){
      alert("Error al buscar el API");
}

objScormApi.LMSInitialize("");
```

Modelo de datos

Introducción al modelo de datos

El propósito de establecer un modelo de datos común es asegurarse que la información sobre el SCO pueda ser seguida por diferentes LMS. Si, por ejemplo, se determina que seguir la puntuación de un alumno es un requerimiento del sistema, entonces es necesario establecer una vía común en el contenido para informar al LMS de las puntuaciones. Si los SCO usan su propio sistema de puntuaciones, los sistemas de aprendizaje no sabrían como recibir, almacenar o procesar la información.

Hay un número de modelos de datos bajo desarrollo en varias comunidades y organizaciones estándar. Este borrador de las especificaciones del modelo de datos intenta agrupar funcionalmente la información que va a ser intercambiada entre el SCO y el LMS. Los ejemplos incluyen: interfaz con información del estudiante, preguntas y autoevaluación, información del estado del sistema, valoración, etc. En el lanzamiento de la actual versión de SCORM este borrador de las especificaciones del modelo está todavía bajo desarrollo.

Modelo de datos del entorno de ejecución en SCORM

El modelo de datos del entorno de ejecución de SCORM deriva directamente del modelo de datos de AICC CMI.

Para identificar el modelo de datos en uso, todos los nombres de los elementos descritos en esta sección empiezan por "cmi". Así señalamos que los elementos provienen del modelo de datos de CMI. Otros modelos de datos empezarán de otra forma cuando sean desarrollados.

Reglas Generales del entorno de ejecución del modelo de datos de SCORM

La siguiente lista resume las reglas generales:

- El primer símbolo en el nombre del elemento de datos identifica el modelo de datos. "cmi" indica modelo de datos CMI. Esto aumenta la funcionalidad del API permitiéndole manejar otros modelos de datos.
- Hay tres palabras reservadas. Son todas minúsculas y van precedidas de un carácter de subrayado.
- _versión: palabra clave usada para determinar la versión del modelo de datos que admite el LMS.
- _children: palabra clave usada para determinar qué elementos del modelo de datos son admitidos por el LMS.
- _count: palabra clave usada para determinar el número de elementos que hay en un momento dado en una lista.
- Todos los arrays empiezan a contar elementos por cero.
- Los nombres del modelo de datos diferencian entre mayúsculas y minúsculas (case-sensitive)
- El modelo de datos está implementado en un SCO. Un SCO no puede acceder los elementos de datos de otro SCO.

Elementos del modelo de datos

Los elementos del modelo de datos están divididos en 2 categorías: obligatorios y opcionales. Cuales elementos son obligatorios y cuales son opcionales viene especificado en la guía AICC CMI001 para compatibilidad entre 2 ordenadores.

Los elementos de datos obligatorios deben ser admitidos por nuestro LMS. Además el LMS también puede proporcionar implementación para admitir algunos o todos los elementos del modelo de datos opcionales.

El uso de todos los elementos de datos por parte del SCO es opcional. Sólo es obligatorio que el SCO use las funciones del API LMSInitialize(“”) y LMSFinish(“”).

Gestión de listas

Para obtener o introducir un valor en una lista debemos utilizar el número índice. La única vez que un número índice se puede omitir es cuando hay solamente un elemento en la lista.

Podemos utilizar la palabra reservada *_count* para saber el número de elementos de una lista.

Si el SCO no conoce el número de registros sobre objetivos en un array, puede empezar la cuenta del estudiante actual con cero. Esto sobrescribiría cualquier información almacenada sobre objetivos que estuviera almacenada en primera posición. Que se sobrescriba o que se anexe es una decisión que hace el creador del SCO.

Se pueden hacer llamadas a los elementos de una lista mediante la notación “.n”. Por ejemplo “cmi.objective.3.status”

Modelo de datos del entorno de ejecución

<table>
<tr><td colspan="2">cmi.core

Hijos del cmi.core:
student_id, student_name, lesson_location, credit, lesson_status, entry, score, total_time, lesson_mode, exit, session_time</td></tr>
<tr><td colspan="2">cmi.core._children</td></tr>
<tr><td>Llamadas soportadas por el API:
LMSGetValue()

Obligatorio en LMS: Sí

Tipo de dato:
CMIString255

Accesibilidad del SCO:
Sólo lectura</td><td>Definición: La palabra clave _children se usa para determinar todos los elementos en la categoría principal que soporta el LMS. Si un elemento no tiene hijos (children) pero los soporta, se devuelve una cadena vacía. Si el elemento no es soportado por el LMS también se devuelve una cadena vacía pero podemos darnos cuenta haciendo un pedido sobre el último error en el cual se dirá que el elemento no es soportado por el LMS.

Uso: Para determinar qué elementos de datos de cmi.core están admitidos por el LMS.

Formato: El valor devuelto es una lista separada con comas del nombre de los elementos en la categoría principal que están soportados por el LMS

Comportamiento del LMS:
• Inicialización: El conjunto de hijos soportados para este. Por tanto una llamada a LMSGetValue() devolvería la lista apropiada de hijos soportados.
• LMSGetValue(): LMS devuelve una lista separada por comas de los elementos admitidos.
o Llamada de ejemplo al API:
LMSGetValue("cmi.core._children")
o Ejemplo de valores devueltos:
“student_id,student_name,lesson_location,credit,lesson_status, entry, score,total_time, exit,session_time”
o Código de error:
▪ 401 – Error de no implementación. Si el elemento cmi.core._children no es soportado por el LMS se devolvería una cadena vacía. NOTA: el cmi.core._children debe ser admitido por el LMS ya que el elemento es obligatorio.
• LMSSetValue(): El LMS debería establecer un código de error de acuerdo con lo siguiente::
o Código de error:</td></tr>
</table>

	▪ **402 – Valor establecido inválido: el elemento es una palabra clave.** Si el elemento es admitido por el LMS (el elemento debe ser admitido ya que el elemento es obligatorio y una llamada a LMSSetValue sobre este elemento debe colocar el código de error en 402.
	▪ **401 – Error de no implementación.** Si el elemento no está soportado el código de error se establece en 401 por el LMS para indicar que el elemento no es soportado. NOTA: El elemento debe ser soportado por el LMS ya que el elemento es obligatorio
	Ejemplo de uso del SCO:
	Los SCOs pueden usar llamadas al cmi.core._children para determinar si un cierto elemento está implementado por el LMS.
	
	var coreChildren = LMSGetValue("cmi.core._children");
	if (coreChildren.indexOf("student_name") != -1)
	{
	studentName = LMSGetValue("cmi.core.student_name");
	}

cmi.core.student_id	
Llamadas admitidas por el API: LMSGetValue() **Obligatorio para el LMS:** Sí **Tipo de datos:** CMIIdentifier **Accesibilidad al SCO:** Solo lectura	**Definición:** Un único identificador alfa-numérico que representa un único usuario del sistema LMS. **Uso:** Usado para identificar un estudiante. **Formato:** Hasta 255 caracteres alfa-numéricos sin espacios. Guiones y comas están permitidos. Los puntos no están permitidos. No hay diferencia entre mayúsculas y minúsculas.. **Comportamiento del LMS:** • **Inicialización:** El LMS es responsable, basado en el registro del estudiante • **LMSGetValue():** Devuelve el valor actual almacenado por el LMS al estudiante. o **Ejemplo de valor devuelto:** "Joe_Student1" "JS-2000" "joe2000-3" o **Código de error:** ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio. • **LMSSetValue():** El LMS debería actualizar el código de error de acuerdo con lo siguiente: o **Código de error:** ▪ **403 Los elementos son de sólo lectura .** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que el elemento es obligatorio) y se hace una llamada a LMSSetValue() en este elemento, entonces el LMS debería actualizar el código de error a 403. o **Llamadas admitidas por el API:** LMSGetValue() **Obligatorio para el LMS:** Sí **Tipo de datos:** CMIIdentifier **Accesibilidad al SCO:** Solo lectura ▪ **Ejemplo del uso del SCO:** El SCO podría querer mostrar el identificador del estudiante.

	var coreStudentID = LMSGetValue("cmi.core.student_id");
cmi.core.student_name	
Llamadas admitidas por el API: LMSGetValue() **Obligatorio para el LMS:** Sí **Tipo de datos:** CMIString255 **Accesibilidad al SCO:** Solo lectura	**Definición:** Normalmente el nombre oficial del estudiante en la lista del curso. Es un nombre completo con apellidos. **Uso:** Se usa para representar el nombre oficial del estudiante en el sistema. **Formato:** Apellidos, nombre, inicial (nombres ingleses) . El apellido y el nombre deben ir separados por comas **Comportamiento del LMS:** • **Inicialización:** El LMS es responsable, basado en el registro del alumno. • **LMSGetValue():** Devuelve el valor actual almacenado en el LMS o **Ejemplo de valores devueltos:** "Estudiante, José Pérez." "Estudiante, Miguel Rioja." o **Código de error:** ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio. • **LMSSetValue():**El LMS debería actualizar el código de error de acuerdo con lo siguiente: o **Código de error:** ▪ **403 Los elementos son de sólo lectura .** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que el elemento es obligatorio) y se hace una llamada a LMSSetValue() en este elemento, entonces el LMS debería actualizar el código de error a 403. ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio. **Ejemplo del uso del SCO:** El SCO podría querer mostrar el nombre del estudiante. var coreStudentName = LMSGetValue("cmi.core.student_name");
cmi.core.lesson_location	
Llamadas admitidas por el API: LMSGetValue() LMSSetValue() **Obligatorio LMS:** sí **Tipo de datos:** CMIString255 **Accesibilidad SCO:** Lectura/Escritura	**Definición y uso:** Esta variable se corresponde con el punto de salida que tuvo el alumno la última vez que estuvo trabajando con el SCO. Así se facilita la vuelta del alumno al lugar donde abandonó el SCO y por tanto a continuar el aprendizaje por donde lo había dejado. **Formato:** La implementación depende. El LMS simplemente almacena la información y la devuelve al SCO cuando el estudiante está volviendo al aprendizaje si el SCO pregunta por ello. El formato siempre es correcto ya que es el SCO quien manda la información a almacenar y quien la recibe y por tanto eso no pasa por cuenta del LMS. Si el alumno accede por primera vez al SCO, lesson_location puede ser igual a una cadena vacía. **Comportamiento del LMS:** • **Inicialización:** El LMS debería introducir una cadena vacía en esta variable. Sin embargo el SCO puede modificar este valor y recuperarlo posteriormente. • **LMSGetValue():** Devuelve el valor actual almacenado de dicha variable o **Código de error:** ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio.

	• **LMSSetValue():** Actualiza el valor de la variable al valor indicado. El valor debe coincidir con el tipo de dato para este elemento
	o **Código de error:**
	▪ **405 – Tipo de dato incorrecto.** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que es obligatorio) y una llamada invoca a LMSSetValue() con un valor que no es el tipo de dato correcto.
	▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio.
	
	Ejemplo de uso del SCO:
	El SCO puede usar lesson_location como un marcador:
	-Al lanzar el SCO, el SCO puede posicionar al estudiante donde el estudiante lo dejó en una sesión anterior
	- -Al salir del SCO, el SCO puede almacenar la posición del estudiante para que continúe en la misma posición la próxima vez que vuelva a entrar.
	
	// Ejemplo de SCO escrito en Java Script y HTML
	// Esta función podría existir en una función llamada onLoad() que se ejecutaría cuando la página HTML se carga
	
	var coreSCOLocation = LMSGetValue("cmi.core.lesson_location");
	if (LMSGetLastError() == "0")
	{
	// coreSCOLocation contiene un enlace definido en la página HTML
	
	// Empezar la lección donde el estudiante la dejó
	window.location.hash = coreSCOLocation;
	}
	else
	{
	// Procesamiento de errores
	}

cmi.core.credit	
Llamadas al API admitidas: LMSGetValue() **Obligatorio LMS:** Sí **Tipo de dato:** CMIVocabulary (Credit) **"credit"** **"no-credit"** **accesibilidad del SCO:** Sólo lectura	**Definición:** Indica si el estudiante está siendo evaluado por el LMS basado en la puntuación o en forma de aprobado/suspenso. **Uso:** Usado por el LMS para indicar si el alumno está haciendo el curso para ser puntuado. cmi.core.crdit se utiliza junto con lesson_mode y también junto con lesson_status **Formato:** Hay dos valores posibles: • **"credit"**. Significa que el estudiante va a ser evaluado. El LMS le dice al SCO que los datos que el SCO manda al LMS son para ser evaluados. • **"no-credit"**. Significa que el estudiante no va a ser evaluado. El LMS le dice al SCO que los datos que el SCO manda al LMS no los utiliza para evaluar al estudiante. **Comportamiento del LMS:** • **Inicialización:** El LMS es responsable de determinar si el estudiante está haciendo el curso para ser evaluado o para no serlo. • **LMSGetValue():** Devuelve el valor actual de la variable almacenada por el LMS. o **Ejemplo de valores devueltos:** "no-credit" "credit" o **Códigos de error:** • **401 – Error no implementado.** Si este elemento no es admitido

	se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio.
	• **LMSSetValue():**El LMS debería actualizar el código de error de acuerdo con lo siguiente:
	o **Código de error:**
	▪ **403 Los elementos son de sólo lectura.** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que el elemento es obligatorio) y se hace una llamada a LMSSetValue() en este elemento, entonces el LMS debería actualizar el código de error a 403.
	▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio.
	
	Ejemplo de uso del SCO:
	El SCO podría usar el valor devuelto desde el LMS para determinar lo que se muestra en el explorador. Puede haber diferentes contenidos en el SCO que es mostrado en el explorador según sea para evaluarlo o no evaluarlo.
	
	var creditFlag = LMSGetValue("cmi.core.credit")
	if (creditFlag == "credit")
	{
	// El estudiante está haciendo el curso para ser evaluado.
	}
	else
	{
	// El estudiante está haciendo el curso para no ser evaluado.
	}

cmi.core.lesson_status	
Llamadas admitidas por el API: LMSGetValue() LMSSetValue() **Obligatorio LMS:** Sí **Tipo de datos:** CMIVocabulary (estado) **passed** **completed** **failed** **incomplete** **browsed** **not attempted** **Accesibilidad del SCO:** Leer/Escribir	**Definición:** Este es el estado actual del estudiante tal y como es determinado por el LMS. Hay 6 valores de estado posibles. **Uso:** Normalmente el SCO determina su propio estado y lo pasa al LMS. Si en cmi.core.credit se almacena el valor credit y el manifiesto del SCO tiene la capacidad de evaluar el LMS puede cambiar el estado a "aprobado" o "suspenso" dependiendo de la puntuación del estudiante y comparada con las instrucciones del SCO. 1) Si el SCO no tiene la capacidad de evaluar el LMS no tiene la capacidad de sustituirle. 2) Si el estudiante realiza el SCO para no ser evaluado no hay cambio en la variable lesson_status con una excepción: si lesson_mode está en "browsed", la variable "lesson_status" puede cambiar a "browsed" incluso si cmi.core.credit está en non-credit. **3)** Al volver a entrar en el SCO, el LMS puede cambiar de estado a "aprobado", "suspenso" o "visto". "Aprobado" y "suspenso" están definidos según los criterios del SCO. Se cambiará de estado a "visto" cuando el SCO fue lanzado inicialmente en estado "pendiente de ver". **Formato:** Hay 6 posibles valores (los valores del vocabulario deben ser en inglés para que sea totalmente compatible con todo tipo de cursos): • **passed**: El número necesario de objetivos en el SCO ha sido aprobado o se logró la puntuación para aprobar. • **completed**: El SCO puede o no puede ser aprobado pero todos sus contenidos pueden haber sido vistos por el estudiante. Esto es lo que indica la variable completed. • **failed**: El estudiante ha suspendido los contenidos del SCO. En cambio esto no obliga a que todos los contenidos hayan sido vistos. • **incomplete**: El SCO ha sido empezado pero no terminado • **browsed**: El estudiante ya ha lanzado el LMS antes. • **not attempted**: significa que el estudiante hizo un intento de cargar el curso pero por alguna razón el curso ni siquiera ha sido empezado. Quizás

el alumno tan sólo ha leído el árbol de contenidos y decidió que no estaba preparado para afrontar el curso. Cualquier algoritmo del SCO puede ser usado para cambiar el valor "not attempted" a "incomplete"

Comportamiento del LMS:

- **Inicialización:** Si es la primera vez que el nuevo estudiante accede al SCO, la variable lesson_status está situada en el valor "not attempted". El LMS es responsable de situar el valor inicial en "not attempted"

    - Otros comportamientos obligatorios para el LMS: si un SCO almacena un valor en cmi.core.lesson_status entonces no hay ningún problema. Sin embargo, SCORM no obliga al SCO a establecer la variable cmi.core.lesson_status. Por tanto hay algunos requerimientos que debe satisfacer el LMS para estos casos:
        - En el lanzamiento inicial el LMS debería establecer el valor "not attempted" en cmi.core.lesson_status.
        - Cuando se ejecuta LMSFinish() o el usuario sale de la aplicación el LMS debe establecer cmi.core.lesson_status en "completed".
        - Una vez establecido el valor de cmi.core.lesson_status en "completed", el LMS debe mirar si el SCO tiene activo el módulo de evaluación comprobando la variable cmi.student_data.mastery_score, si es admitida, o el manifiesto del SCO. Si se suministra un módulo de evaluación y el SCO estableció valores en cmi.core.score.raw el LMS comparará el módulo de evaluación con cmi.core.score.raw y almacenará en cmi.core..lesson_status "passed" o "failed" según haya aprobado o suspendido respectivamente. Si no se suministra módulo de evaluación el Lms dejará cmi.core..lesson_status como "completed", es decir, terminado.
- **LMSGetValue():** Devuelve el valor almacenado en el modelo de datos. Debe ser una del conjunto de variables definidas.
    - **Código de error:**
        - **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio.
- **LMSSetValue():** Actualiza el valor de la variable al valor indicado. El valor debe coincidir con el tipo de dato para este elemento
    - **Código de error:**
        - **405 – Tipo de dato incorrecto.** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que es obligatorio) y una llamada invoca a LMSSetValue() con un valor que no es el tipo de dato correcto.
        - **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio.

- **Ejemplo de asignación y recuperación de variables::**
  "completed"
  "failed"
  "browsed"

Uso del SCO:

- passed – usado cuando el SCO se usa para evaluar.
- failed – usado cuando el SCO se usa para evaluar.
- completed – usado cuando el SCO no se usa para evaluar.
- incomplete – usado cuando se abandona la aplicación antes de terminarla, ya sea evaluando o no evaluando.
- browsed – usado cuando el lesson_mode sea "browse"
- not attempted – El SCO nunca debería introducir este valor en lesson_status (es el valor inicial que asigna el LMS cuando el estudiante

	accede por primera vez al SCO.
	
	Ejemplo del uso del SCO:
	var lessonStatus = LMSGetValue("cmi.core.lesson_status");
	if (lessonStatus == "failed")
	{
	// El estudiante suspendió el SCO, actuar en consecuencia.
	}
	else
	{
	// El estudiante aprobó el SCO, actuar en consecuencia.
	}

cmi.core.entry	
Llamadas admitidas al API LMSGetValue() **Obligatorio LMS:** Yes **Tipo de datos:** CMIVocabulary (Entry) **"ab-initio"** **"resume"** **""** – ***cadena vacía*** **Accesibilidad SCO:** Sólo lectura	**Definición:** Indicación de si el estudiante ha accedido al SCO con anterioridad. **Uso:** Cuando un estudiante carga el SCO por primera vez, cmi.core.entry debería establecerse en "AB-inicio" por el LMS. Si un estudiante vuelve a entrar en un contenido suspendido. Se establecería el valor "resume" en dicha variable. **Formato:** Tres valores posibles: • **"ab-initio":** Esto indica que es la primera vez que un estudiante entra al SCO. Puesto que un estudiante podría haber pasado todos los objetivos de un SCO haciendo un pre-test, el valor "not attempted en lesson_status no es un indicador fiable. Esto significaría que un SCO podría ser aprobado sin que el estudiante ni siquiera lo haya visto antes. • **"resume":** Indica que el estudiante ha estado en el SCO con anterioridad. El estudiante está empezando de nuevo un SCO suspendido. • **"":** La cadena vacía se usa para representar una entrada del estudiante en el SCO que no es ninguna de las anteriores. **Comportamiento del LMS:** ▪ **Inicialización:** En el lanzamiento inicial del LMS se debe inicializar el modelo de datos con "ab-initio". o Comportamiento posterior: Al recibir LMSFinish() o cuando el estudiante se sale de la aplicación, el LMS debe establecer cmi.core.entry en "" – cadena vacía- o "resume". Esto se determina por el LMS analizando la variable cmi.core.exit. Si vale "suspend" se pondrá en cmi.core.entry en "resume" hasta la próxima vez que se inicie la aplicación. Si hay otro valor o no hay valor en cmi.core.exit , cmi.core.entry debe establecerse en "" –cadena vacía-. ▪ **LMSGetValue():** Devuelve el valor almacenado, el cual debe ser una palabra del conjunto de palabras admitidas. o **Ejemplo de valores devueltos:** "AB-inicio" "resume" o **Código de error:** ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio. • **LMSSetValue():**El LMS debería actualizar el código de error de acuerdo con lo siguiente: o **Código de error:** ▪ **403 Los elementos son de sólo lectura.** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que el elemento es obligatorio) y se hace una llamada a LMSSetValue() en este elemento, entonces el LMS debería actualizar el código de error a 403. ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos

	por el LMS dado que el elemento es obligatorio.
	
	Ejemplo del uso del SCO:
	var entryStatus = LMSGetValue("cmi.core.entry")
	if (LMSGetLastError() == "0")
	{
	if (entryStatus == "resume")
	{
	// El estudiante está volviendo a empezar el SCO
	}
	else
	{
	// Esta es la primera vez que el estudiante se introduce en el SCO
	}
	}
	else
	{
	// Hay un error, actuar consecuentemente
	}

cmi.core.score Indica el rendimiento del estudiante Children de cmi.core.score: raw, min, max	
cmi.core.score._children	
Llamadas soportadas por el API LMSGetValue() **Obligatorio LMS: Sí** **Tipo de datos:** CMIString255 **Accesibilidad del SCO:** Sólo lectura	**Definición:** La palabra clave _children se usa para determinar todos los elementos de la categoría de puntuación que son soportados por el LMS. Si un elemento no tiene children, pero los admite, se devuelve una cadena vacía. Si un elemento no es admitido no se devuelve ningún valor. Una solicitud del último error puede verificar que el elemento no es admitido. **Uso:** Se usa para determinar los elementos (children) de cmi.core.score que son admitidos por el LMS. Raw es el único elemento obligatorio que debe ser admitido. **Formato:** El valor devuelto es una lista separada por comas de todos los nombres de la categoría de puntuación que son admitidos por el LMS. **Comportamiento del LMS:** ▪ **Inicialización:** El conjunto de children admitidos para este grupo. Por tanto con una llamada a LMSGetValue(), la lista apropiada de elementos admitidos es devuelta. ▪ **LMSGetValue():** Devuelve una lista separadas por comas de los elementos admitidos. o **Ejemplo de llamada al API:** LMSGetValue("cmi.core.score._children") o **Ejemplo de valores devueltos:** "raw" – *el LMS debe admitir al menos este elemento*" o **Código de error:** ▪ **401 – Error de no implementación**. Si el elemento cmi.core._children no es soportado por el LMS se devolvería una cadena vacía. NOTA: el cmi.core._children debe ser admitido por el LMS ya que el elemento es obligatorio. • **LMSSetValue():** El LMS debería establecer un código de error de acuerdo con lo siguiente:: o **Código de error:** ▪ **402 – Valor establecido inválido: el elemento es una palabra clave.** Si el elemento es admitido por el LMS (el elemento debe ser admitido ya que el elemento es obligatorio y una llamada a LMSSetValue sobre este elemento debe colocar el código de error en 402. ▪ **401 – Error de no implementación.** Si el elemento no está soportado el código de error se establece en 401 por el LMS para indicar que el elemento no es soportado. NOTA: El elemento debe ser soportado por el LMS ya que el elemento es obligatorio

	Ejemplo de uso del SCO:
	var scoreChildren = LMSGetValue("cmi.core.score._children");
	if (coreChildren.indexOf("min") != -1)
	{
	LMSSetValue("cmi.core.score.min","10");
	}

cmi.core.score.raw	
Llamadas admitidas por el API: LMSGetValue() LMSSetValue() **Obligatorio LMS:** Sí **Tipo de datos:** CMIDecimal o CMIBlank **Accesibilidad SCO:** Lectura/Escritura	**Definición:** Indicación de la actuación del estudiante durante su último intento en el SCO. Esta puntuación puede ser determinada y calculada de cualquier manera que tenga sentido para el diseñador del SCO. Por ejemplo, podría reflejar el porcentaje de objetivos completado, podría ser el resultado bruto en un test con múltiples posibilidades. El cmi.core.score.raw debe ser un valor normalizado entre 0 y 100. **Uso:** Cuando el estudiante carga el SCO por primera vez, el cmi.core.score.raw debe estar en cadena vacía. Las siguientes veces cmi.core.score.raw refleja lo que fue grabado en la sesión previa del estudiante. Si el SCO no tiene valor en cmi.core.score.raw se debe devolver una cadena vacía. **Formato:** número decimal o en blanco. **Comportamiento del LMS:** ▪ **Inicialización:** El LMS debe inicializar esto con una cadena vacía. El SCO es el responsable de establecer este valor. Si LMSGetValue() se ejecuta antes de que el SCO haya establecido este valor, entonces el LMS debería devolver una cadena vacía. ▪ **LMSGetValue():** Devuelve el valor almacenado en el modelo de datos. o **Llamada de ejemplo del API:** LMSGetValue("cmi.core.score.raw") ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio. • **LMSSetValue():** Actualiza el valor de la variable al valor indicado. El valor debe coincidir con el tipo de dato para este elemento o **Código de error:** ▪ **405 – Tipo de dato incorrecto.** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que es obligatorio) y una llamada invoca a LMSSetValue() con un valor que no es el tipo de dato correcto. ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio. . ▪ **Ejemplo sobre valores devueltos:** "90" "85.7" "" **Ejemplo sobre el uso del SCO:** El SCO podría usarse para seguir el resultado neto del estudiante en el SCO. LMSSetValue("cmi.core.score.raw","85");
cmi.core.score.max	
Llamadas admitidas por el API: LMSGetValue() LMSSetValue() **Obligatorio LMS:** No	"" **Definición:** La máxima puntuación o el número total de aciertos que el estudiante podría haber logrado. El cmi.core.max debe ser un valor normalizado entre 0 y 100. **Uso:** indica la máxima puntuación que el estudiante podría haber alcanzado.

Tipo de datos:
CMIDecimal or
CMIBlank
 	
Accesibilidad SCO:
Leer/Escribir
	Formato: número decimal o en blanco.
	
	Comportamiento del LMS:
	▪ **Inicialización:** El LMS debería inicializarlo con una cadena vacía (“”) en el lanzamiento inicial del SCO. El SCO es responsable de tratar este valor. Si se ejecuta un LMSGetValue() antes de que el SCO haya almacenado algo en esta variable entonces el LMS debería devolver una cadena vacía.
	
	▪ **LMSGetValue():** Devuelve el valor almacenado en el modelo de datos. El valor devuelto debe ser de tipo CMIDecimal o CMIBlank.
	o **Ejemplo de una llamada del API:** LMSGetValue("cmi.core.score.max")
	o **Código de error:**
	401 – Error de no implementación. Si el elemento no está soportado el código de error se establece en 401 por el LMS para indicar que el elemento no es soportado. NOTA: El elemento debe ser soportado por el LMS ya que el elemento es obligatorio
	▪ **LMSSetValue():** Sitúa el elemento del modelo de datos en el valor dado. Éste debe corresponder al tipo de datos para este elemento.
	o **Ejemplo de llamada al API:** LMSSetValue("cmi.core.score.max","100")
	o **Código de error:**
	▪ **405 – Tipo de dato incorrecto.** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que es obligatorio) y una llamada invoca a LMSSetValue() con un valor que no es el tipo de dato correcto.
	▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido.
	▪ **Ejemplo de valores devueltos o almacenados:**
	“100"
	"5"
	
	Ejemplo de uso del SCO:
	var scoreChildren = LMSGetValue(“cmi.core.score._children”);
	if (coreChildren.indexOf(“max”) != -1)
	{
	LMSSetValue(“cmi.core.score.max”,"100");
	}

cmi.core.score.min	
Llamadas admitidas por el API: LMSGetValue() LMSSetValue() **Obligatorio LMS:** No **Tipo de datos:** CMIDecimal or CMIBlank **Accesibilidad SCO:** Leer/Escribir	**Definición:** La mínima puntuación o el número total de aciertos que el estudiante podría haber logrado. El cmi.core.max debe ser un valor normalizado entre 0 y 100. **Uso:** indica la mínima puntuación que el estudiante podría haber alcanzado. **Formato:** número decimal o en blanco. **Comportamiento del LMS:** ▪ **Inicialización:** El LMS debería inicializarlo con una cadena vacía (“”) en el lanzamiento inicial del SCO. El SCO es responsable de tratar este valor. Si se ejecuta un LMSGetValue() antes de que el SCO haya almacenado algo en esta variable entonces el LMS debería devolver una cadena vacía. ▪ **LMSGetValue():** Devuelve el valor almacenado en el modelo de datos. El valor devuelto debe ser de tipo CMIDecimal o CMIBlank. o **Ejemplo de una llamada del API:** LMSGetValue("cmi.core.score.max") o **Código de error:** **401 – Error de no implementación.** Si el elemento no está soportado el código de error se establece en 401 por el LMS para indicar que el elemento no es soportado. ▪ **LMSSetValue():** Sitúa el elemento del modelo de datos en el valor dado.

	Éste debe corresponder al tipo de datos para este elemento.
	○ **Ejemplo de llamada al API:** LMSSetValue("cmi.core.score.max","100")
	○ **Código de error:**
	▪ **405 – Tipo de dato incorrecto.** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que es obligatorio) y una llamada invoca a LMSSetValue() con un valor que no es el tipo de dato correcto.
	▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido.
	▪ **Ejemplo de valores devueltos o almacenados:** "100" "5"
	
	Ejemplo del uso del SCO:
	var scoreChildren = LMSGetValue("cmi.core.score._children");
	if (coreChildren.indexOf("min") != -1)
	{
	LMSSetValue("cmi.core.score.min","10");
	}

cmi.core.total_time	
Llamadas admitidas por el API: LMSGetValue() **Obligatorio LMS:** Sí **Tipo de datos:** CMITimespan **Accesibilidad SCO:** Sólo lectura	**Definición:** Tiempo acumulado de todas las sesiones de un estudiante en el SCO. **Uso:** Usado para realizar el seguimiento del tiempo total que ha pasado un alumno con un SCO. El LMS debe inicializarlo en un valor por defecto cuando el SCO es lanzado por primera vez y entonces usar los valores descritos por el SCO (session_time) para llevar una cuenta acumulada. **Formato:** Horas, minutos y segundos separados por dos puntos. HHHH:MM:SS.SS Horas tiene un mínimo de 2 dígitos y un máximo de 4 dígitos. Los minutos consistirán en 2 dígitos exactos. Los segundos tendrán 2 dígitos con un punto decimal opcional para indicar décimas o centésimas. **Comportamiento del LMS:** ▪ **Inicialización:** El LMS debe inicializarlo en "0000:00:00.00" en el primer lanzamiento del SCO. ○ **Comportamiento adicional:** Un SCO es capaz, en una sola ejecución, de llevar múltiples almacenamientos del cmi.core.total_time. Cuando el SCO ejecuta LMSFinish() o el usuario sale del sistema, el LMS debería coger el último cmi.core.session_time que el SCO almacenó y acumularlo en cmi.core.total_time. En el siguiente lanzamiento del SCO, una llamada a LMSGetValue() para la variable cmi.core.total_time, el LMS debería devolver el tiempo total acumulado. Los Lms no deben acumular los tiempos múltiples enviados por los SCO con llamadas a LMSSetValue() para la variable cmi.core.session_time. Si se hacen llamadas múltiples a LMSSetValue() por cmi.core.session_time, el LMS debe sobrescribir cualquier valor existente de esa variable. ▪ **LMSGetValue():** Devuelve el valor almacenado en el modelo de datos. El valor devuelto debe de ser de tipo CMITimespan. ○ **Ejemplo de llamada al API:** LMSGetValue("cmi.core.total_time") ○ **Ejemplo de valores devueltos:** "00:29:00" "01:27:45.5" ○ **Código de error:** ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio. • **LMSSetValue():**El LMS debería actualizar el código de error de acuerdo

	con lo siguiente:
	o **Código de error:**
	▪ **403 Los elementos son de sólo lectura .** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que el elemento es obligatorio) y se hace una llamada a LMSSetValue() en este elemento, entonces el LMS debería actualizar el código de error a 403.
	▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: los elementos deben ser admitidos por el LMS dado que el elemento es obligatorio.
	
	Ejemplo de uso del SCO:
	var totalTime = LMSGetValue("cmi.core.total_time");
	if (LMSGetLastError() == "0")
	{
	// Uso de cmi.core.total_time
	}

cmi.core.lesson_mode	
Llamadas admitidas por el API: LMSGetValue() **Obligatorio LMS:** No **Tipo de datos:** CMIVocabulary (Mode) **"browse"** **"normal"** **"review"** **Accesibilidad SCO:** Sólo lectura	**Definición:** Identifica el comportamiento deseado del SCO después del lanzamiento de este. Muchos SCO tienen un único comportamiento. Sin embargo, algunos SCOs pueden presentar distintas cantidades de información o presentarla en distinto orden reflejando las distintas formas de aprendizaje basadas en los deseos del diseñador del SCO. Los diseñadores pueden hacer que un SCO tenga comportamientos distintos ilimitados. Este estándar admite la comunicación de tres parámetros que pueden resultar en comportamientos distintos del SCO. **Uso:** Se usa para representar los diferentes modos en que un SCO puede ser lanzado. Se usa junto con lesson_status. **Formato:** Tres valores posibles: • **"browse":** El estudiante quiere hacer una vista previa pero no necesariamente ser evaluado. • **"normal":** indica que el SCO se debe comportar de forma que el alumno sea evaluado. • **"review":** El estudiante ya ha visto el contenido al menos una vez y ha sido evaluado. Si lesson_mode no es reconocido o hay errores entonces el SCO asume el estado "normal" por defecto. **Comportamiento del LMS:** ▪ **Inicialización:** El LMS debería determinar el modo en que el SCO está siendo lanzado. Nota: ahora mismo en el modelo SCORM 1.2 no existe forma de saber si un contenido de aprendizaje puede ser lanzado de diferentes maneras. ▪ **LMSGetValue():** Devuelve el valor almacenado en el modelo de datos. o **Ejemplo de llamada al API:** LMSGetValue("cmi.core.lesson_mode") o **Ejemplo de valores devueltos:** "browse" "normal" "review" o **Código de error:** ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. • **LMSSetValue():**El LMS debería actualizar el código de error de acuerdo con lo siguiente: o **Código de error:** ▪ **403 Los elementos son de sólo lectura.** Si el

	elemento es admitido (el elemento debe ser admitido por el LMS dado que el elemento es obligatorio) y se hace una llamada a LMSSetValue() en este elemento, entonces el LMS debería actualizar el código de error a 403.
	
	▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido.
	
	Ejemplo de uso del SCO:
	var mode = LMSGetValue("cmi.core.lesson_mode");
	if (LMSGetLastError() == "0")
	{
	if (mode == "browse")
	{
	// El estudiante está en modo "browse"
	}
	else if (mode == "review")
	{
	//El estudiante ya ha visto el contenido y ha sido evaluado. Debe estar revisando.
	}
	else
	{
	// El estudiante está lanzando el SCO en modo normal
	}
	}

cmi.core.exit	
Llamadas admitidas por el API: LMSSetValue() **Obligatorio LMS:** sí **Tipo de datos:** CMIVocabulary (Exit) **"time-out"** **"suspend"** **"logout"** **""** ***- empty string*** **Accesibilidad SCO:** Sólo escritura	**Definición:** una indicación de como o porqué el estudiante abandonó el SCO. **Uso:** Se usa para indicar la razón por la que el estudiante abandonó la aplicación. **Formato:** Tres valores posibles: • **"time-out":** Indica que el SCO finalizó la aplicación porque ha pasado un tiempo excesivo o el tiempo máximo dado ha sido excedido. Este tiempo se puede encontrar en el manifiesto(adlcp:maxtimeallowed). • **"suspend":** Indica que el estudiante sale del SCO con intención de continuar más tarde en el mismo punto donde lo dejó. • **"logout":** Indica que el estudiante salió de la aplicación desde dentro del SCO en lugar de volver al LMS para salir. Esto implica que el SCO pasó el control al LMS y el LMS automáticamente retiró al estudiante de la aplicación – después de actualizar los valores correspondientes en el modelo de datos- • **"" :** Se usa para representar un estado normal de salida. **Comportamiento del LMS:** ▪ **Inicialización:** El elemento no necesita ser inicializado. Nunca se hace un LMSGetValue sobre este elemento. El elemento se controla con el SCO. o **Comportamiento adicional:** Un SCO tiene la opción de establecer cmi.core.exit en uno de los cuatro valores. Basado en dicho valor, el comportamiento del LMS debería ser como sigue: ▪ Si el SCO establece cmi.core.exit en "time-out", el LMS debería establecer cmi.core.entry en "" (cadena vacía) en el siguiente lanzamiento del SCO. ▪ Si el SCO establece cmi.core.exit en "suspend", el LMS debería establecer cmi.core.entry en "resume" en el siguiente lanzamiento del SCO. ▪ Si el SCO establece cmi.core.exit en "logout", el LMS debería establecer cmi.core.entry en ""(cadena vacía) la próxima vez que se lance el SCO. Además, el LMS debería sacar al estudiante del curso cuando el SCO que estableció cmi.core.exit en "logout" haya ejecutado LMSFinish() o el usuario abandone la aplicación. ▪ Si el SCO establece cmi.core.exit en ""(cadena vacía), el LMS debe establecer cmi.core.entry en "" la próxima

	vez que el SCO sea lanzado. ▪ Si el SCO no estableció cmi.core.exit en ningún valor entonces el LMS debería establecer cmi.core.entry en “”(cadena vacía) la próxima vez que se lance el SCO. ▪ **LMSGetValue():** El LMS debería establecer un código de error de acuerdo con lo siguiente y devolver una cadena vacía. o **Código de error:** ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía y un código de error se establece para indicar que el elemento no es admitido. NOTA: el elemento debe ser admitido por el LMS ya que es obligatorio. ▪ **404 – El elemento es de solo escritura.** Si un SCO intenta llamar a LMSGetValue() en este elemento, el LMS debe establecer el código de error en 404 y devolver una cadena vacía. ▪ **LMSSetValue():** Establece nuevos valores en un elemento del modelo de datos. El valor debe coincidir con el tipo de dato de ese elemento: o **Ejemplo de llamada al API:** LMSSetValue("cmi.core.exit","logout") o **Ejemplo de valores establecidos:** "time-out” "suspend" "logout" o **Código de error:** ▪ **405 – Tipo de dato incorrecto.** Si el elemento es admitido (el elemento debe ser admitido por el LMS dado que es obligatorio) y una llamada invoca a LMSSetValue() con un valor que no es el tipo de dato correcto. ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía. Y el código de error se actualiza indicando que el elemento no es admitido. NOTA: el elemento debe ser admitido por el LMS ya que es obligatorio. **Ejemplo del uso del SCO:** LMSSetValue("cmi.core.exit","time-out")
cmi.core.session_time	
Llamadas admitidas por el API: LMSSetValue() **Obligatorio LMS:** sí **Tipo de dato:** CMITimespan **Accesibilidad SCO:** Sólo escritura	**Definición:** Es la cantidad de horas, minutos y segundos que el estudiante ha pasado con el SCO en el momento en que lo abandona, es decir, el tiempo de la sesión en un sólo uso del SCO. **Uso:** Se usa para seguir el tiempo de la sesión de un alumno. El LMS usa este tiempo para determinar el cmi.core.total_time. **Formato:** HHHH:MM::SS.SS Horas, minutos y segundos separados por dos puntos. HHHH:MM:SS.SS Horas tiene un mínimo de 2 dígitos y un máximo de 4 dígitos. Los minutos consistirán en 2 dígitos exactos. Los segundos tendrán 2 dígitos con un punto decimal opcional para indicar décimas o centésimas. **Comportamiento del LMS:** ▪ **Inicialización:** El elemento no necesita ser inicializado por el LMS. Nunca hay una llamada con LMSGetValue() a este elemento. o **Comportamiento adicional:** Un SCO es capaz de llevar de una vez múltiples establecimientos de cmi.core.session_time. Cuando se ejecuta LMSFinish() o el usuario abandona la aplicación, el LMS debe acumular el cmi.core.session_time al cmi.core.total_time. El LMS no debería acumular múltiples tiempos enviados al LMS por llamadas a LMSSetValue() sobre cmi.core.session_time. Si se hiciera, el LMS debería sobrescribir cualquier valor existente. ▪ **LMSGetValue():** El LMS debería establecer un código de error de acuerdo

	con lo siguiente y devolver una cadena vacía. o **Código de error:** ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía y un código de error se establece para indicar que el elemento no es admitido. NOTA: el elemento debe ser admitido por el LMS ya que es obligatorio. ▪ **404 – El elemento es de solo escritura.** Si un SCO intenta llamar a LMSGetValue() en este elemento, el LMS debe establecer el código de error en 404 y devolver una cadena vacía. ▪ **LMSSetValue():** o **Ejemplo de llamada al API:** LMSSetValue(“cmi.core.session_time”,"0010:34:34.56") o **Ejemplo de valores devueltos:** "0010:34:34.56" "05:15:00" o **Código de error:** ▪ **205 – Tipo de dato incorrecto:** Si un LMSSetValue() falla al no coincidir el tipo de dato de la variable que lleva con el que debería tener. ▪ **401 – Error no implementado.** Si este elemento no es admitido se devuelve una cadena vacía y un código de error se establece para indicar que el elemento no es admitido. NOTA: el elemento debe ser admitido por el LMS ya que es obligatorio. **Ejemplo del uso del SCO:** LMSSetValue("cmi.core.session_time","0000:12:30")

Hemos analizado:

- La estructura del archivo imsmanifest.xml que nos permitirá informar al LMS sobre la estructura del contenido y de los archivos utilizados en él.
- El modo en que el contenido debe acceder al API para poder comunicarse con el LMS (java-script)
- El uso del código de error del API a fin de que el contenido pueda determinar si cualquier función de llamada al API se ha ejecutado con éxito y si no ha sido así, determinar cuál fue el error.
- El modelo de datos que nos permitirá intercambiar información entre el contenido y el LMS

Implementación de un SCO

Comenzaremos por implementar el java-script que le permitirá a nuestro contenido encontrar al API para, a través del mismo, comunicarse con el LMS.

```
/*****************************************************************************
**
** Nombre del archivo: APIWrapper.js
**
**Este archivo es parte del ejemplo que provee ADL
*****************************************************************************/

var _Debug = false;

// codigos de error
var _NoError = 0;
var _GeneralException = 101;
var _ServerBusy = 102;
var _InvalidArgumentError = 201;
var _ElementCannotHaveChildren = 202;
var _ElementIsNotAnArray = 203;
var _NotInitialized = 301;
var _NotImplementedError = 401;
var _InvalidSetValue = 402;
var _ElementIsReadOnly = 403;
var _ElementIsWriteOnly = 404;
var _IncorrectDataType = 405;


// definición de variables locales
var apiHandle = null;
var API = null;
var findAPITries = 0;


/*****************************************************************************
**
** Function: doLMSInitialize()
** Entradas:  NO
** Retorna:   true si la inicialización fue satisfactoria, o
**            false si la inicialización falló.
**
** Descripción:
** Inicializa la comunicación con el LMS llamando a la función LMSInitialize
** que debe ser implementada por el LMS.
**
*****************************************************************************/
function doLMSInitialize()
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Error al buscar el API.");
    return "false";
  }

  var result = api.LMSInitialize("");

  if (result.toString() != "true")
  {
    var err = ErrorHandler();
  }

  return result.toString();
}

/*****************************************************************************
**
```

```
** Function doLMSFinish()
** Entradas:  NO
** Retorna:   true si termina satisfactoriamente
**            false si falla.
**
** Descripción:
** Cierra la comunicación con el LMS llamando a la función LMSFinish
** la cual debe ser implementada por el LMS
**
*****************************************************************************/
function doLMSFinish()
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Error al buscar el API.");
    return "false";
  }
  else
  {


    var result = api.LMSFinish("");
    if (result.toString() != "true")
    {
      var err = ErrorHandler();
    }

  }

  return result.toString();
}

/*****************************************************************************
**
** Function doLMSGetValue(name)
** Entradas:  name - string que representa la categoría o elemento del modelo de datos cmi
**          (ej. cmi.core.student_id)
** Retorna:  El valor asignado por el LMS al elemento.
**
** Descripción:
** Se invoca el método LMSGetValue
**
*****************************************************************************/
function doLMSGetValue(name)
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Error al buscar el API.");
    return "";
  }
  else
  {
    var value = api.LMSGetValue(name);
    var errCode = api.LMSGetLastError().toString();
    if (errCode != _NoError)
    {
      // si ocurrió un error se muestra la descripción del mismo
      var errDescription = api.LMSGetErrorString(errCode);
      alert("LMSGetValue("+name+") failed. \n"+ errDescription);
      return "";
    }
    else
    {

      return value.toString();
    }
  }
}
```

```
/*******************************************************************************
**
** Function doLMSSetValue(name, value)
** Entradas:  name -string que representa la categoría o elemento del modelo de datos cmi
**          value -valor que debe ser asignado al elemento
** Retorna:  true si la asignación es exitosa
**          false si falla.
**
** Descripción:
** Se invoca la función LMSSetValue
**
*******************************************************************************/
function doLMSSetValue(name, value)
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Error al buscar el API.");
    return;
  }
  else
  {
    var result = api.LMSSetValue(name, value);
    if (result.toString() != "true")
    {
      var err = ErrorHandler();
    }
  }

  return;
}

/*******************************************************************************
**
** Function doLMSCommit()
** Entradas:  NO
** Retorna:  NO
**
** Descripción:
** Se invoca la función LMSCommit
**
*******************************************************************************/
function doLMSCommit()
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Error al buscar el API.");
    return "false";
  }
  else
  {
    var result = api.LMSCommit("");
    if (result != "true")
    {
      var err = ErrorHandler();
    }
  }

  return result.toString();
}

/*******************************************************************************
**
** Function doLMSGetLastError()
** Entradas:  NO
** Retorna:  El código de error que fue seteado por la última función invocada del LMS
**
** Descripción:
```

```
** Invoca la función LMSGetLastError
**
*****************************************************************************/
function doLMSGetLastError()
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert(Error al buscar el API.");
    return _GeneralError;
  }

  return api.LMSGetLastError().toString();
}

/*****************************************************************************
**
** Function doLMSGetErrorString(errorCode)
** Entradas:  codigo de error
** Retorna:  La descripción textual que corresponde al código de error
**
** Descripción:
** Llama a la función LMSGetErrorString
**
*****************************************************************************/
function doLMSGetErrorString(errorCode)
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Error al buscar el API.");
  }

  return api.LMSGetErrorString(errorCode).toString();
}

/*****************************************************************************
**
** Function doLMSGetDiagnostic(errorCode)
** Entradas:  código de error
**
** Descripción:
** Llama a la función LMSGetDiagnostic
**
*****************************************************************************/
function doLMSGetDiagnostic(errorCode)
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Error al buscar el API.");
  }

  return api.LMSGetDiagnostic(errorCode).toString();
}

/*****************************************************************************
**
** Function LMSIsInitialized()
** Entradas:  NO
** Retorna:  true si el API del LMS es inicializada correctamente. False en caso contrario.
**
** Descripción:
** Determina si el API del LMS fue inicializada correctamente o no.
**
*****************************************************************************/
function LMSIsInitialized()
{

  var api = getAPIHandle();
```

```
  if (api == null)
  {
    alert("Error al buscar el API.");
    return false;
  }
  else
  {
    var value = api.LMSGetValue("cmi.core.student_name");
    var errCode = api.LMSGetLastError().toString();
    if (errCode == _NotInitialized)
    {
      return false;
    }
    else
    {
      return true;
    }
  }
}

/*****************************************************************************
**
** Function ErrorHandler()
** Entradas:  NO
** Retorna:  Valor actual del código de error del LMS
**
** Descripción:
** Determina si hubo error antes de la llamda al API, y desplega un mensaje al usuario. Si el código de error tiene **
** asociado un texto lo desplega.
**
*****************************************************************************/
function ErrorHandler()
{
  var api = getAPIHandle();
  if (api == null)
  {
    alert("Error al buscar el API.");
    return;
  }

  var errCode = api.LMSGetLastError().toString();
  if (errCode != _NoError)
  {
    var errDescription = api.LMSGetErrorString(errCode);

    if (_Debug == true)
    {
      errDescription += "\n";
      errDescription += api.LMSGetDiagnostic(null);
       }

    alert(errDescription);
  }

  return errCode;
}

/*****************************************************************************
**
** Function getAPIHandle()
** Entradas:  NO
** Retorna:  valor de APIHandle
**
** Descripción:
** Retorna el manejo del objeto API si fue seteado,
** sino retorna null
**
*****************************************************************************/
function getAPIHandle()
{
```

```
  if (apiHandle == null)
  {
    apiHandle = getAPI();
  }

  return apiHandle;
}


/*****************************************************************************
**
** Function findAPI(win)
** Entradas:  win - un obleto window
** Retorna:  Si se encuentra el objeto API, se retorna el mismo, sino se retorna null
**
** Descripción:
** La función busca un objeto llamado API
**
*****************************************************************************/
function findAPI(win)
{
  while ((win.API == null) && (win.parent != null) && (win.parent != win))
  {
    findAPITries++;
    // Nota: 7 es un número arbitrario, pero debería ser suficiente
    if (findAPITries > 7)
    {
      alert("Error al buscar el API.");
      return null;
    }

    win = win.parent;

  }
  return win.API;
}



/*****************************************************************************
**
** Function getAPI()
** Entradas:  NO
** Retorna:  Si encuentra un objeto API, el mismo es retornado, de lo contrario retorna null
**
** Descripción:
** Esta función busca un objeto llamado API, primero en la ventana actual
** de la jerarquía de frames y entonces, de ser necesario, en la ventana actual de la jerarquia de opener window
**
**
*****************************************************************************/
function getAPI()
{
  var theAPI = findAPI(window);
  if ((theAPI == null) && (window.opener != null) && (typeof(window.opener) != "undefined"))
  {
    theAPI = findAPI(window.opener);
  }
  if (theAPI == null)
  {
    alert("Error al buscar el API ");
  }
  return theAPI
}
```

También debemos recordar que el contenido debe comenzar y concluir la comunicación con el LMS. Para esto utilizamos las siguientes funciones:

```
/*******************************************************************************
**
** Archivo: SCOFunctions.js
**
** Descripción: Este archivo contiene funciones javaScript que son utilizados en el SCO
**
/*******************************************************************************
*******************************************************************************/

function loadPage()
{
  var result = doLMSInitialize();
}

function unloadPage()
{

  result = doLMSFinish();
}
```

Ahora construiremos nuestro contenido, que mostrará en cada página el nombre del usuario que lo está ejecutando (para esto deberá comunicarse con el LMS a través del API, utilizando las funciones del archivo APIWrapper.js)

```
<html>
<head>
<meta http-equiv="expires" content="Tue, 05 DEC 2000 01:00:00 GMT">
<meta http-equiv="Pragma" content="no-cache">
<script    language=javascript            src="../APIWrapper.js"></script>
<script    language=javascript            src="../SCOFunctions.js"></script>
<title>prueba scorm</title>
</head>
<body onunload="return unloadPage()">
        <p align="right">
        <font color=lightslategray>
        <b><i><label>SCO 01</label></i></b>
        </font>
        </p>
        <h1>esto es una prueba</h1>
        <hr>
        <br>

        <b>El usuario que esta ejecutando la prueba es :<b>

        <script    language="javascript">
        loadPage();
        var   studentName = "!";
        var   lmsStudentName = doLMSGetValue(  "cmi.core.student_name" );

        if ( lmsStudentName  != "" )
        {
          studentName = " " + lmsStudentName +   "!";
        }

        document.write(studentName);
        </script>


        <hr>

</body>

</html>
```

Ahora debemos construir el archivo imsmanifest.xml

```
<?xml version="1.0"?>
<manifest identifier="SingleCourseManifest" version="1.1"
        xmlns="http://www.imsproject.org/xsd/imscp_rootv1p1p2"
        xmlns:adlcp="http://www.adlnet.org/xsd/adlcp_rootv1p2"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.imsproject.org/xsd/imscp_rootv1p1p2 imscp_rootv1p1p2.xsd
                        http://www.imsglobal.org/xsd/imsmd_rootv1p2p1 imsmd_rootv1p2p1.xsd
                        http://www.adlnet.org/xsd/adlcp_rootv1p2 adlcp_rootv1p2.xsd">
  <organizations default="B0">

    <organization identifier="B0">

      <title>Curso SCORM</title>

      <item identifier="B100" isvisible="true">
        <title>prueba</title>

        <item identifier="S100001" identifierref="R_S100001" isvisible="true">
          <title>sco1</title>
        </item>
      </item>
      <metadata>
        <schema>ADL SCORM</schema>
        <schemaversion>1.2</schemaversion>
        <adlcp:location>prueba.xml</adlcp:location>
      </metadata>
    </organization>
  </organizations>
  <resources>

    <resource identifier="R_S100001" type="webcontent"
          adlcp:scormtype="sco" href="prueba/Lesson01/sco01.htm">
      <file href="prueba/Lesson01/sco01.htm" />
      <dependency identifierref="R_D1"/>
    </resource>

    <resource identifier="R_D1" adlcp:scormtype="asset"
          type="webcontent" xml:base="prueba">
      <file href="APIWrapper.js"/>
    </resource>

  </resources>
</manifest>
```

Podemos observar que los metadatos se encuentran en archivos xml. En nuestro caso tenemos metaanotado el recurso prueba en el archivo prueba.xml

```
<?xml version="1.0"?>
<lom xmlns="http://www.imsglobal.org/xsd/imsmd_rootv1p2p1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.imsglobal.org/xsd/imsmd_rootv1p2p1 imsmd_rootv1p2p1.xsd">

  <general>
    <title>
      <langstring>Prueba de Contenido SCORM 1.2</langstring>
    </title>
    <language>es</language>
    <description>
      <langstring>SCO básico para la demostración de uso de SCORM</langstring>
    </description>
    <keyword>
      <langstring>SCO</langstring>
    </keyword>
    <keyword>
      <langstring>SCORM</langstring>
```

```
    </keyword>
    <keyword>
      <langstring>ADL</langstring>
    </keyword>
    <aggregationlevel>
      <source>
        <langstring xml:lang="x-none">LOMv1.0</langstring>
      </source>
      <value>
        <langstring xml:lang="x-none">3</langstring>
      </value>
    </aggregationlevel>
  </general>

  <lifecycle>
    <version>
      <langstring>1.1</langstring>
    </version>
    <status>
      <source>
        <langstring xml:lang="x-none">LOMv1.0</langstring>
      </source>
      <value>
        <langstring xml:lang="x-none">Final</langstring>
      </value>
    </status>
    <contribute>
      <role>
        <source>
          <langstring xml:lang="x-none">LOMv1.0</langstring>
        </source>
        <value>
          <langstring xml:lang="x-none">Author</langstring>
        </value>
      </role>

      <date>
        <datetime>2005-03-08</datetime>
      </date>
    </contribute>
  </lifecycle>

  <metametadata>
    <catalogentry>
      <catalog>test</catalog>
      <entry>
        <langstring>test 1000</langstring>
      </entry>
    </catalogentry>
    <metadatascheme>ADL SCORM 1.2</metadatascheme>
    <language>es</language>
  </metametadata>
  <technical>
    <format>text/html</format>
    <location type="URI">prueba/</location>
    <requirement>
      <type>
        <source>
          <langstring xml:lang="x-none">LOMv1.0</langstring>
        </source>
        <value>
          <langstring xml:lang="x-none">Browser</langstring>
        </value>
      </type>
      <name>
        <source>
          <langstring xml:lang="x-none">LOMv1.0</langstring>
        </source>
        <value>
          <langstring xml:lang="x-none">Microsoft Internet Explorer</langstring>
```

```
        </value>
      </name>
      <minimumversion>5.0</minimumversion>
    </requirement>
  </technical>

  <educational>
    <learningresourcetype>
      <source>
        <langstring xml:lang="x-none">LOMv1.0</langstring>
      </source>
      <value>
        <langstring xml:lang="x-none">Narrative Text</langstring>
      </value>
    </learningresourcetype>
  </educational>

  <rights>
    <cost>
      <source>
        <langstring xml:lang="x-none">LOMv1.0</langstring>
      </source>
      <value>
        <langstring xml:lang="x-none">no</langstring>
      </value>
    </cost>
    <copyrightandotherrestrictions>
      <source>
        <langstring xml:lang="x-none">LOMv1.0</langstring>
      </source>
      <value>
        <langstring xml:lang="x-none">no</langstring>
      </value>
    </copyrightandotherrestrictions>
    <description>
      <langstring>Solo debe ser utilizado como ejemplo.</langstring>
    </description>
  </rights>

  <classification>
    <description>
      <langstring>Este curso ha sido implementado para la tesis Usando XM para metaanotar recursos
educativos.</langstring>
    </description>
    <keyword>
      <langstring>SCO</langstring>
    </keyword>
    <keyword>
      <langstring>SCORM</langstring>
    </keyword>
  </classification>
</lom>
```

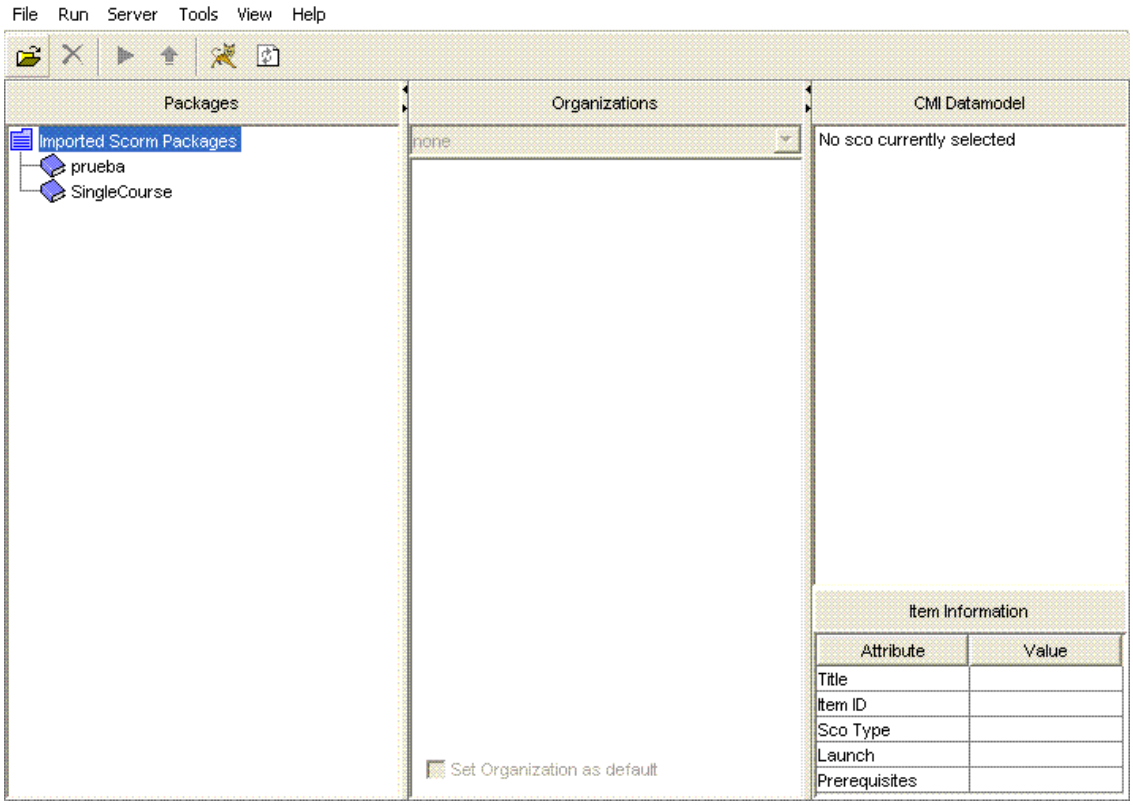
Ahora, armamos la siguiente estructura de directorios:

Curso
- Prueba
  - Lesson01
    - Sco01.html
  - SCOFunctions.js
  - APIWrapper.js
- Imsmanifest.xml
- Prueba.xml
- ims_xml.xsd
- adlcp_rootv1p2.xsd
- imscp_rootv1p1p2.xsd
- imsmd_rootv1p2p1.xsd

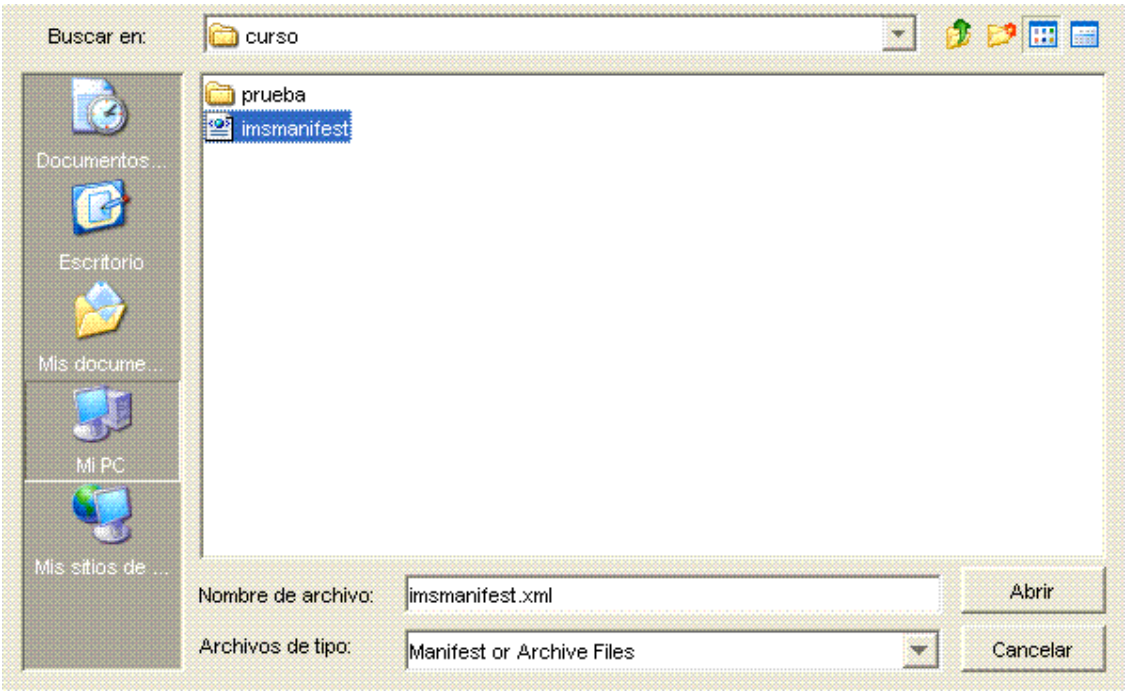
Estamos en condiciones de importar nuestro contenido en un LMS que soporte SCORM 1.2. Lo importaremos en Reload Scorm Player, para luego ejecutarlo. ReloadPlayer es una herramienta que permite testear los paquetes SCORM.

Las acciones para este paso son:

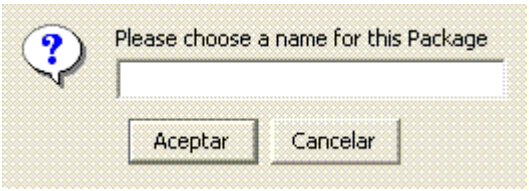
- Seleccionar el icono “Import Scorm Package” (primer ícono de la barra de íconos)



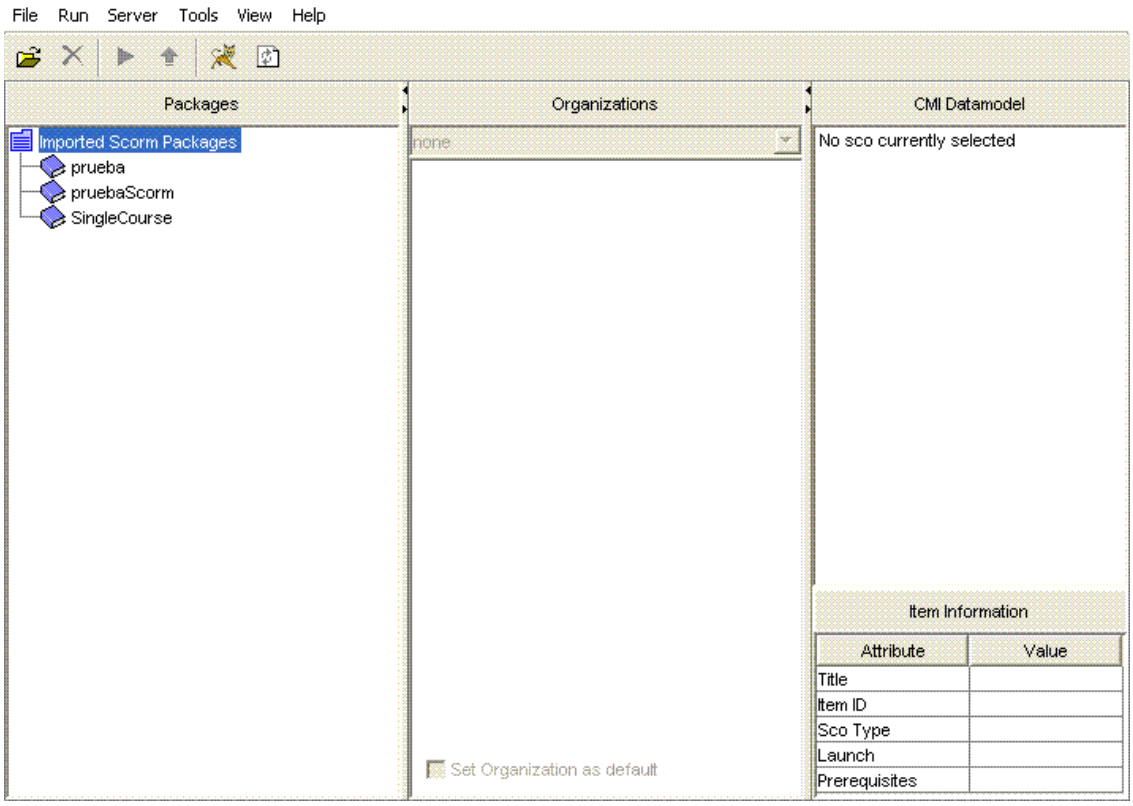
- Seleccionar el archivo imsmanifest.xml



- Escribir el título con el que se verá el contenido en el árbol "Imported Scorm Packages"

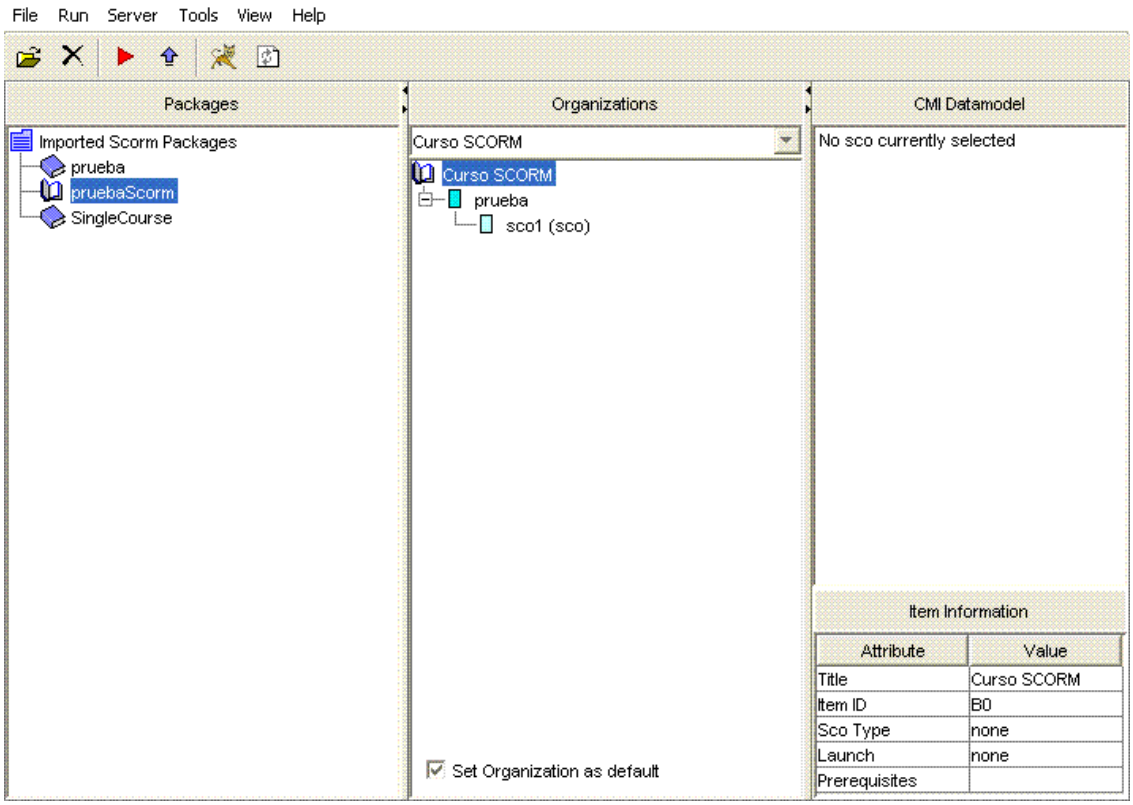


Si ingresamos el nombre "pruebaScorm", ahora nuestro curso aparecerá en el árbol con dicho nombre:

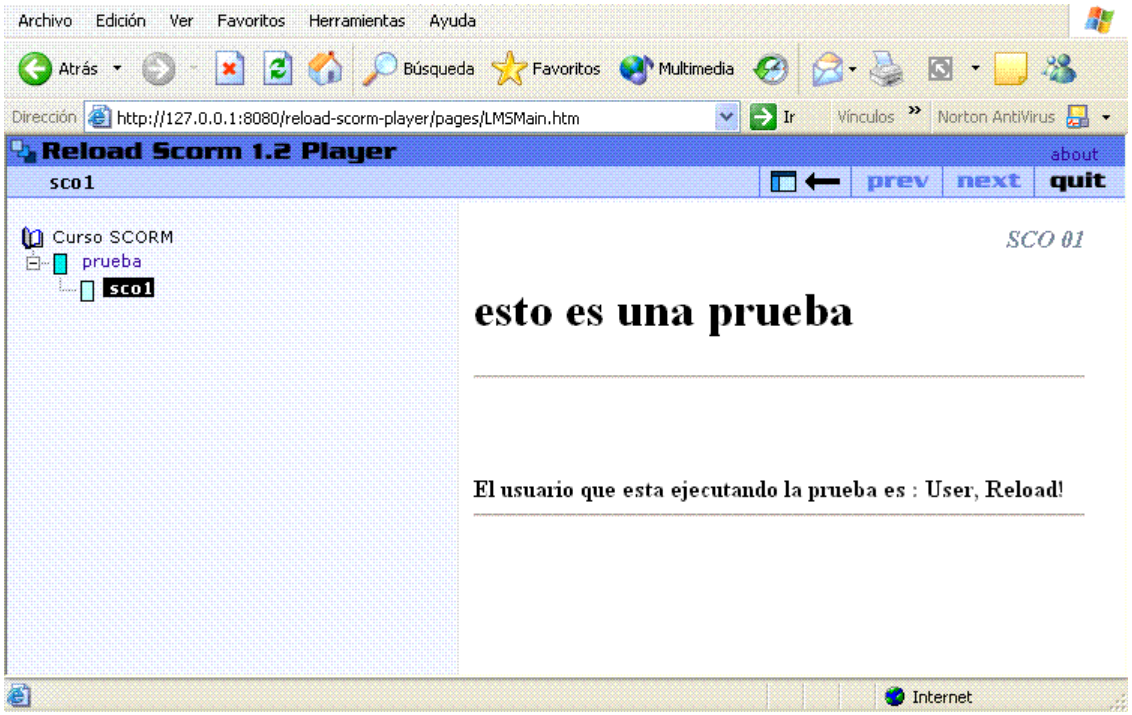


Las acciones para ejecutar el contenido importado son:

- Seleccionar el paquete pruebaScorm en el panel “Packages”. En el panel “Organization” se ve el índice del contenido.
- Hacer click en el ícono “Run Scorm Package”



Se podrá observar que se abre un navegador y se ejecuta tal y como se ha definido en el ReloadEditor.



Este mismo contenido podemos importarlo en cualquier LMS que soporte SCORM 1.2

Implementación de un Curso SCORM

Como aplicación de esta tesis se implementó un curso de Internet para docentes con SCORM 1.2

Este curso está dirigido a docentes e instruye sobre el uso de herramientas que ofrecen las tecnologías de la información y de la comunicación y cómo los docentes pueden utilizarlas en las aulas.

El objetivo de este curso de capacitación es brindar al docente la posibilidad de ampliar, profundizar y transferir a contextos específicos de enseñanza aprendizaje sus habilidades en el uso de las Tecnologías de la Información y de la Comunicación (TIC).

La actualización docente supone reciclar, reconvertir o incorporar nuevos elementos que se articulan con saberes previos adquiridos en otras instancias de formación.

Es de suma importancia que los docentes aprendan y se actualicen en el uso de las herramientas informáticas adecuadas para incorporarlas en la enseñanza cotidiana en las aulas.

Este curso se utilizará para capacitar en el uso de herramientas para el acceso a los distintos servicios sobre Internet como el correo electrónico y charlas en línea.

El curso se divide en tres secciones: contenido, actividades y autoevaluaciones. Dentro de cada sección contiene un índice con los distintos módulos:

- Nociones Generales
- El navegador
- Buscadores y herramientas
- Correo Electrónico
- Foros de Discusión
- Otros Servicios

En la sección contenido cada módulo contiene el desarrollo del tema, en la sección actividades cada módulo contiene una página con ejercicios a realizar por parte de los alumnos y en la sección autoevaluaciones cada módulo contiene una autoevaluación con preguntas múltiple choice sobre el tema.

Este curso SCORM podrá ser cargado en cualquier LMS que soporte Scorm, y gracias al modelo de datos CMI se podrá conocer el estado de navegación de las unidades por parte de los alumnos, así como también el resultado de las autoevaluaciones.

Conocer el estado de navegación del curso permite al LMS mostrar al alumno la siguiente página cada vez que comienza una nueva navegación.
También permitirá al LMS mostrar los tiempos de conexión de cada alumno así como también el tiempo en que se resolvió una autoevaluación.

Las funciones javaScript que se utilizaron para mantener en el LMS el estado de navegación del curso fueron:

(Como ejemplo mostramos las funciones para el módulo “Otros Servicios” del curso)

```
<script type="text/javascript" language="JavaScript">
// Manage page navigation
var znNavigablePages=29;  -- cantidad de páginas  a navegar
var znThisPage=1;
// Use an array to keep track of which pages have been visited
// We will treat this SCO as complete if every navigable page has been visited
var zaVisitedPages = new Array(znNavigablePages) – páginas navegadas por el alumno
function NextPage() {
 if (znThisPage < znNavigablePages){
   znThisPage++;
   myStage.location.href = "uni3_mod14_pag" + znThisPage + ".html"
 }
}
function PreviousPage() { -- función que permite navegar la página anterior
 if (znThisPage > 1){
   znThisPage--;
   myStage.location.href = "uni3_mod14_pag" + znThisPage + ".html"
 }
}
function GoToPage(n) { -- función que permite navegar una página en particular
 if ((!isNaN(n)) && (n >= 1) && (n <= znNavigablePages)){
   myStage.location.href = "uni3_mod14_pag" + n + ".html"
 }
}
function SetResumen() { -- función que permite navegar el resumen del módulo

   myStage.location.href = "resumen14.html"

}
function SetPage(n) {   -- función que permite navegar una página en particular
 if ((!isNaN(n)) && (n >= 1) && (n <= znNavigablePages)){
   myStage.location.href = "uni3_mod14_pag" + n + ".html"
 }
}
function SetThisPage(n) {  -- -- función que registra la página navegada
 var i = 0; var nCnt = 0
 znThisPage = n;
 zaVisitedPages[n-1] = true;
 for (i = 0 ; i < znNavigablePages; i++){
   if (zaVisitedPages[i]) { nCnt++ }
 }
 if (nCnt == znNavigablePages) {
   SCOSetStatusCompleted()
 }
 // -- se envia la información al LMS
 SCOSetValue("cmi.core.exit","suspend");
 SCOSetValue("cmi.core.lesson_location", znThisPage);
 SCOSetValue("cmi.suspend_data",zaVisitedPages.join(","));
 SCOCommit()
}
function AllDone() {  -- fin de la navegacion
 znThisPage = znNavigablePages + 1;
 myStage.location.href = "endpage.htm";
            SCOCommit()
}
function SCOInitData() {  -- inicio de la navegacion
            var loc = 1;
 if (SCOGetValue("cmi.core.entry") == "resume"){

   var SuspendData = SCOGetValue("cmi.suspend_data")
   if (SuspendData.length > 0) {
     zaVisitedPages = SuspendData.split(",")
   }
   var loc =(parseInt(SCOGetValue("cmi.core.lesson_location")));

                        if (isNaN(loc)) loc = 1;
```

```
  }

        GoToPage(loc)
}
function SCOSaveData() {
  if (znThisPage > znNavigablePages) {
    SCOSetValue("cmi.core.exit","") // this is the endpage, no need to resume
  }
}
</script>
```

Estas funciones se encuentran en la primer página de cada unidad llamada sco.html

Las funciones utilizadas para enviar al LMS el resultado de las autoevaluaciones fueron:

```
<script type="text/javascript" language="JavaScript">
var cant=0;
function EvalMultipleChoiceItem(obj,i) {  -- registra el valor de la pregunta i
        var v = obj.value;
        var objectiveID = "pregunta00" + i;
        if (!isNaN(v)) {
                if (v==100)
                        {SCOSetObjectiveData(objectiveID, "status", "passed");
                         cant=cant + 10;
                         }
                else
                        {SCOSetObjectiveData(objectiveID, "status", "failed");
                        cant=cant - 10;
                        }


                //SCOSetValue("cmi.core.score.raw",v); // this will also set pass/fail status
                //alert(SCOGetValue("cmi.core.score.raw"));
                SCOCommit() // to make sure that, no matter what happens, this has been recorded
        }
}
function ShowSCORMStatus(){-- función que envia al LMS el resultado de la autoevaluacion
        SCOSetValue("cmi.core.score.raw",cant/5);
        SCOCommit();
        alert("Nota del Examen: " + cant/5);

}
</script>
```

Estas funciones se encuentran en la página evaluacion.html de cada unidad. Estas funciones utilizan el archivo SCORMObjectiveLogic.js

que contiene las funciones propuestas por ADL para el manejo de objetivos, que en este caso son llamadas por las funciones anteriores para registrar el resultado de cada pregunta multiple choice.

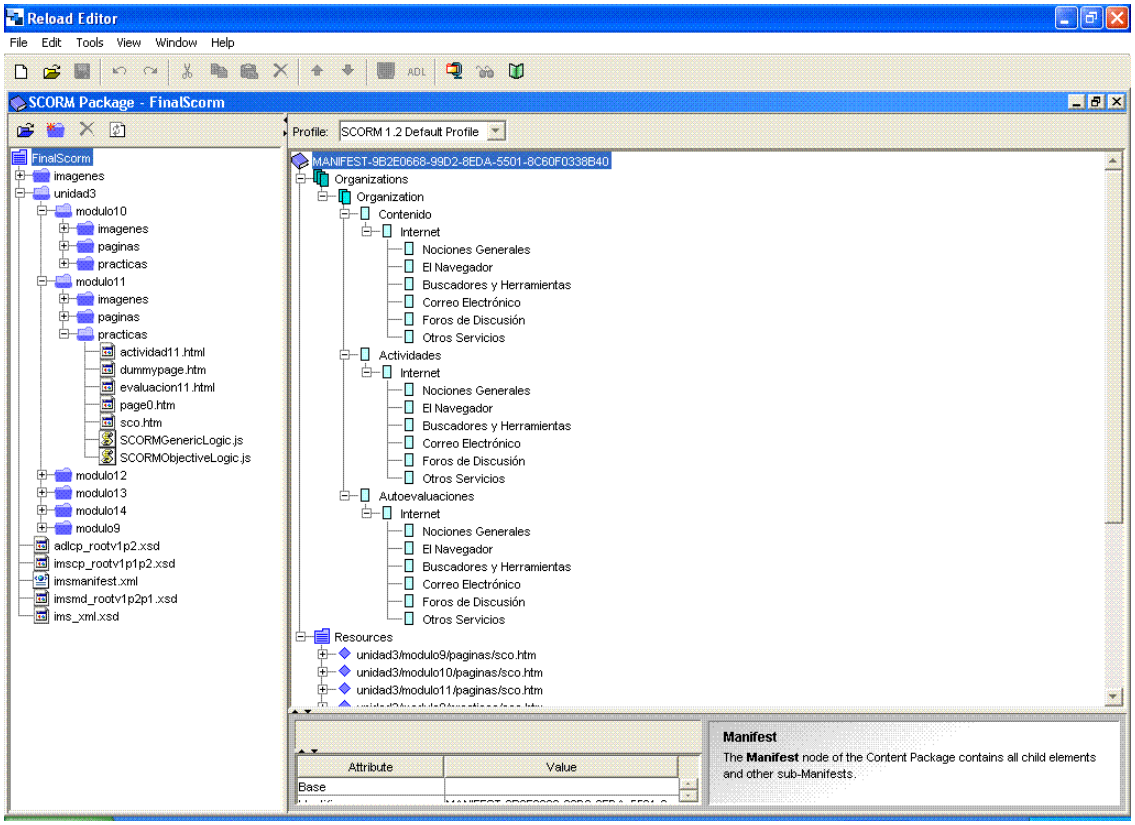
```
function SCOSetObjectiveData(id, elem, v) {
            var result = "false";
  var i = SCOGetObjectiveIndex(id);
  if (isNaN(i)) {
    i = parseInt(SCOGetValue("cmi.objectives._count"));
    if (isNaN(i)) i = 0;
    if (SCOSetValue("cmi.objectives." + i + ".id", id) == "true"){
            result = SCOSetValue("cmi.objectives." + i + "." + elem, v)
                        }
  } else {
                        result = SCOSetValue("cmi.objectives." + i + "." + elem, v);
                        if (result != "true") {
      // Maybe this LMS accepts only journaling entries
                        i = parseInt(SCOGetValue("cmi.objectives._count"));
                                    if (!isNaN(i)) {
                                                if (SCOSetValue("cmi.objectives." + i + ".id", id) == "true"){
                                    result = SCOSetValue("cmi.objectives." + i + "." + elem, v)
                                                }
                                    }
                        }
  }
            return result
}

function SCOGetObjectiveData(id, elem) {
  var i = SCOGetObjectiveIndex(id);
  if (!isNaN(i)) {
    return SCOGetValue("cmi.objectives." + i + "."+elem)
  }
  return ""
}

function SCOGetObjectiveIndex(id){
  var i = -1;
  var nCount = parseInt(SCOGetValue("cmi.objectives._count"));
  if (!isNaN(nCount)) {
                        for (i = nCount-1; i >= 0; i--){ //walk backward in case LMS does journaling
      if (SCOGetValue("cmi.objectives." + i + ".id") == id) {
        return i
      }
    }
  }
  return NaN
}
```

Luego para la comunicación con el LMS se utilizaron las funciones propuestas por ADL, en el archivos SCORMGenericLogic.js

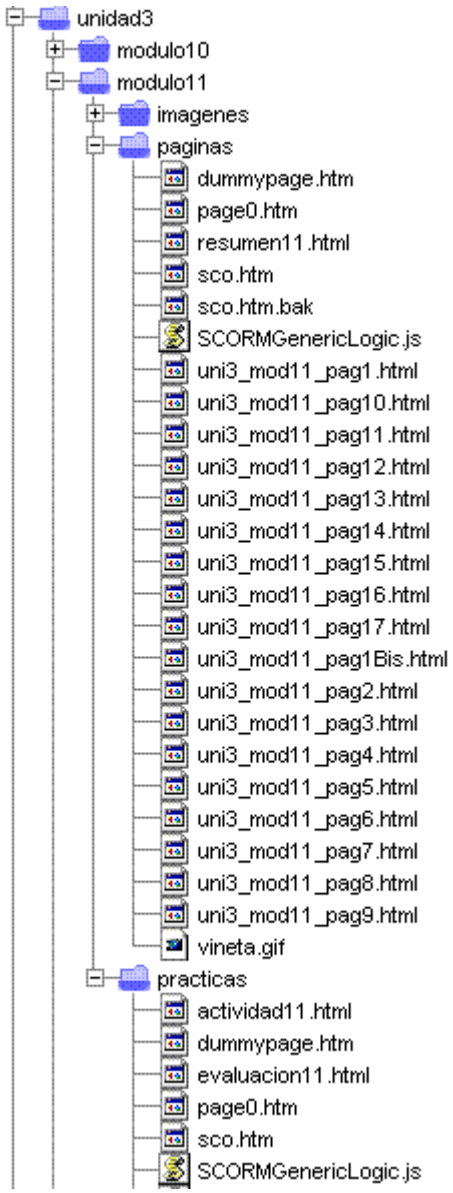
Cada página del curso fue modificada a fin de ejecutar desde cada página html las funciones correspondientes.

Creación del paquete Scorm

Una vez creadas las páginas html con sus respectivas funciones javaScripts, se utilizó la herramienta Reload Editor 1.2 para construir el paquete SCORM.

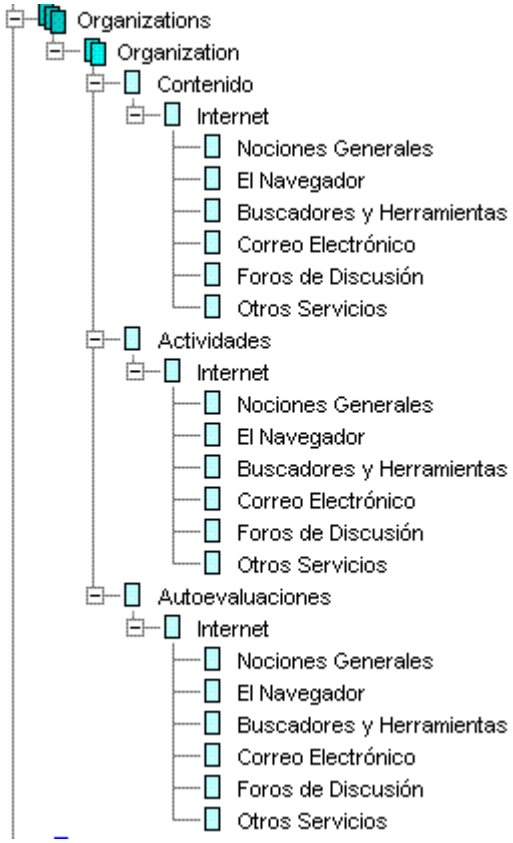


Veamos la distribución de los html y javaScripts creados (frame izquierdo):

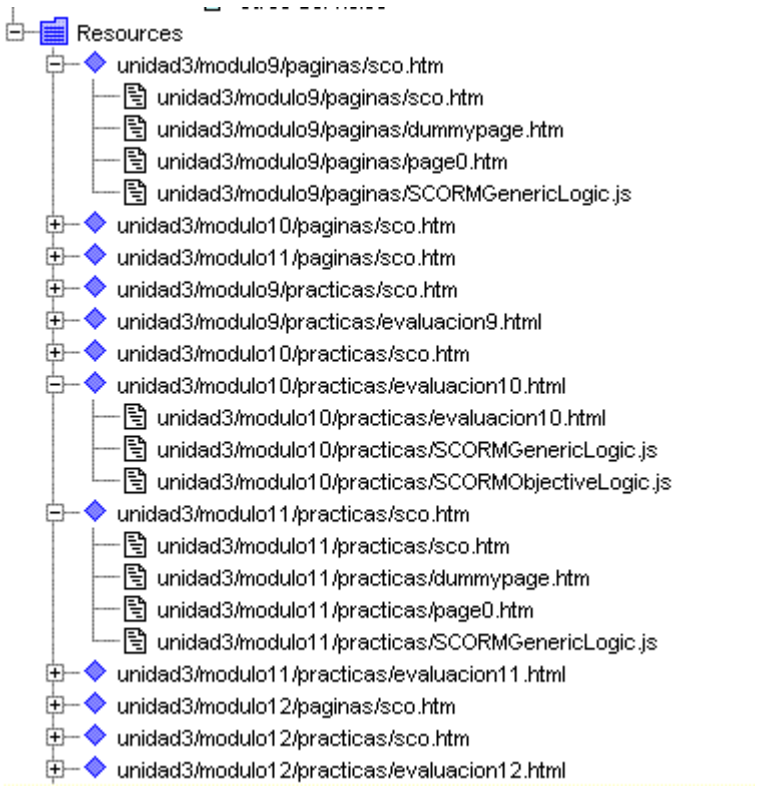


Veamos en detalle la organización del curso (frame derecho parte superior):

El curso contiene una sola organización para el contenido. Esta organización está dividida en tres secciones: contenido, actividades y autoevaluaciones como mencionamos anteriormente. Luego dentro de cada sección tenemos el acceso a cada módulo.

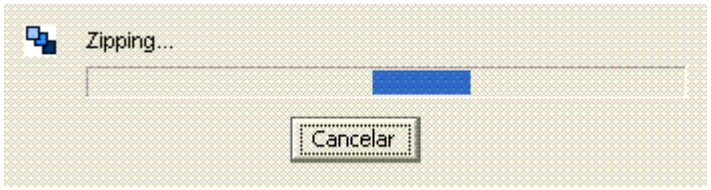


Veamos los recursos utilizados (frame derecho parte inferior):



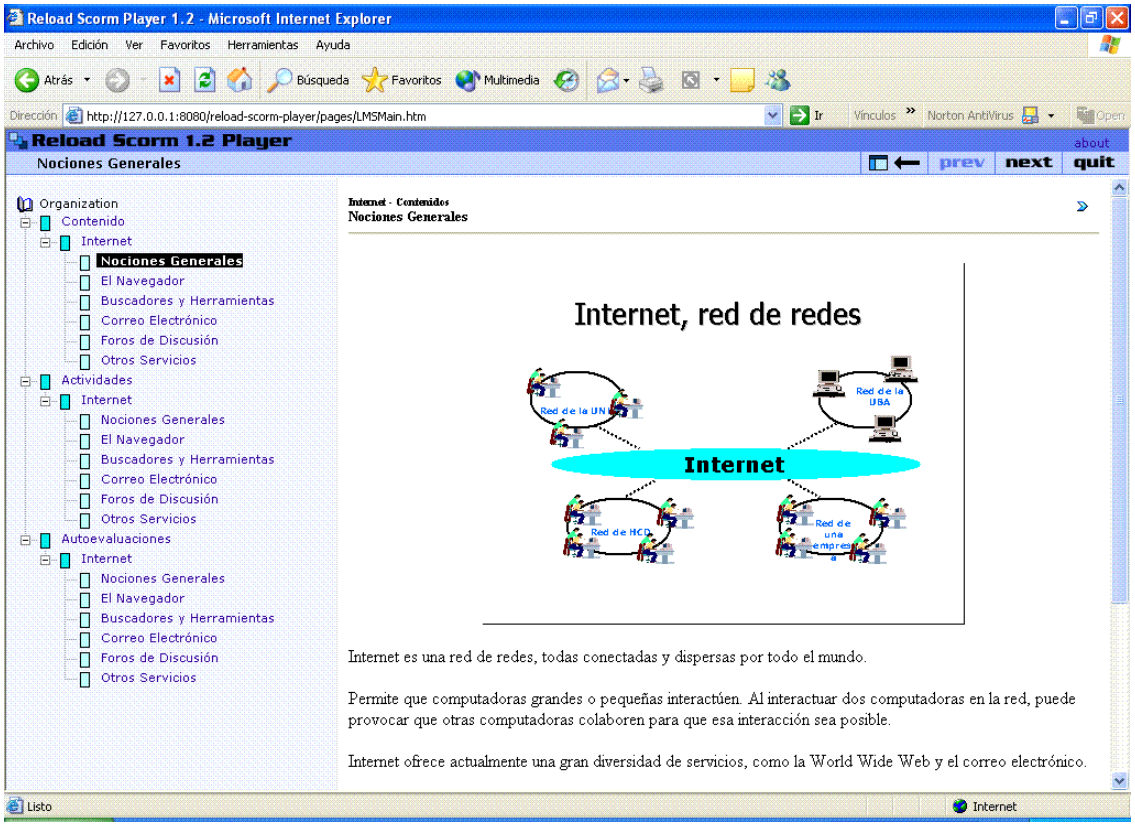
Para cada módulo tenemos el sco correspondiente al contenido, el sco correspondiente a la actividad y el sco correspondiente a la autoevaluación. Cada uno de ellos con sus assets.

Utilizando la función Zip Content Package creamos el paquete SCORM del curso.



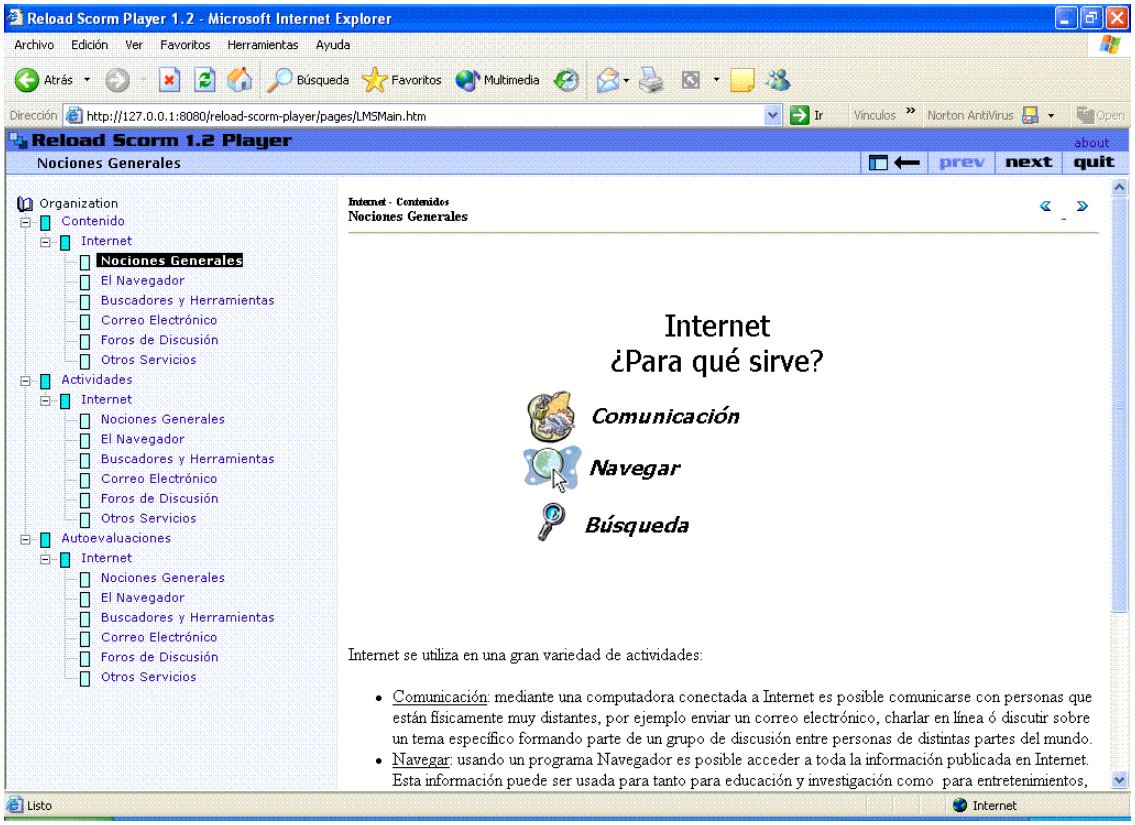
Navegación del curso en Reload Scorm Player 1.2

Vemos la ejecución del curso con Reload Scorm Player 1.2:



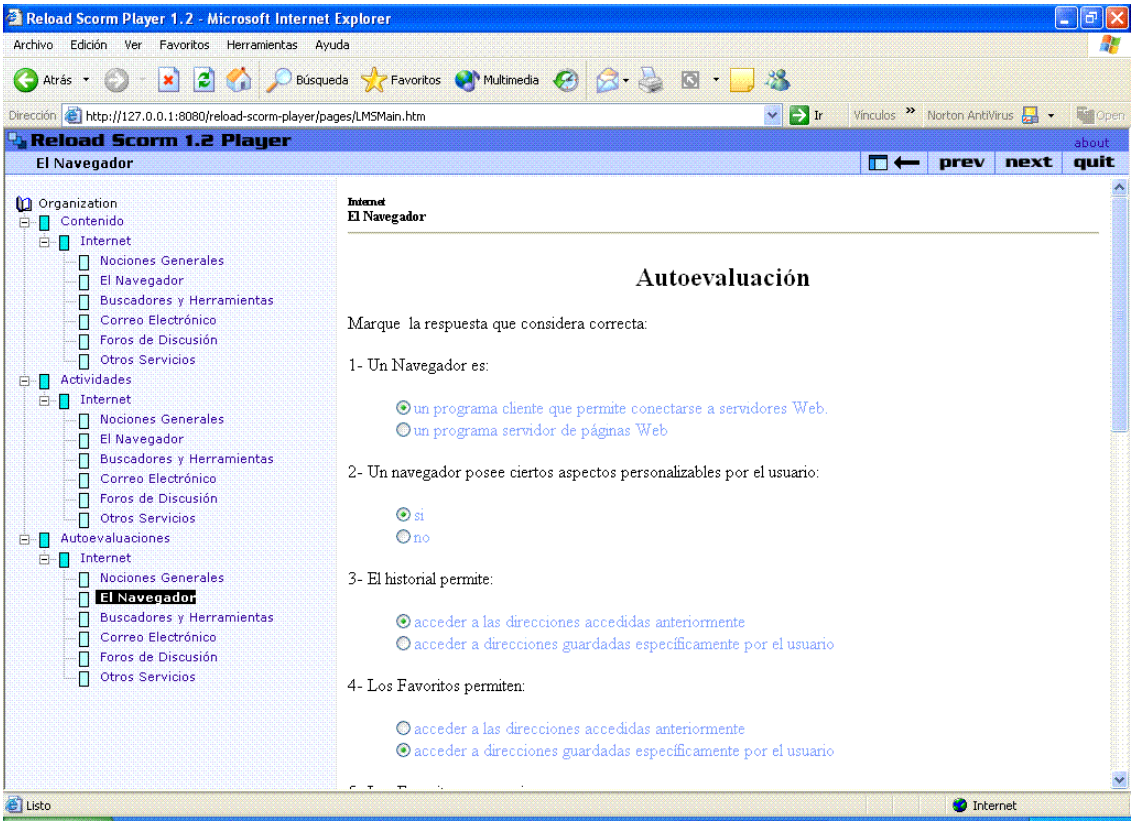
Podemos observar que el índice del curso corresponde con la organización establecida al crear el paquete SCORM.

Veamos, por ejemplo, que ocurre si navegamos esta página, cerramos el curso y volvemos a navegarlo:



Como dijimos antes el LMS abre el curso en la página siguiente. Esto ocurre gracias al envio de información de navegación por parte del cantenido al LMS a través de las funciones javaScript presentadas anteriormente.

Veamos la navegación de un examen:



Al enviar el examen obtenemos la nota que el contenido envia al LMS, utilizando las funciones javaScript vistas anteriormente.

Luego esta información es enviada al LMS para almecenarla en su base de datos.

Navegación del curso en MOODLE

Qué es Moodle?

Moodle es un paquete de software para la creación de cursos y sitios Web basados en Internet.

El entorno de aprendizaje de Moodle está basado en los principios pedagógicos constructivistas, con un diseño modular que hace fácil agregar contenidos que motivan al estudiante. [Alejandro Escalante 2005]

Moodle se distribuye gratuitamente como Software libre (Open Source) (bajo la Licencia pública GNU).

Caracteristicas de Moodle

- Moodle es un producto activo y en evolución.
- Promueve una pegadogía constructivista social (colaboración, actividades, reflexión crítica, etc.).
- Apropiada para el 100% de las clases en línea, así como también para complementar el aprendizaje presencial.
- Tiene una interfaz de navegador de tecnología sencilla, ligera, eficiente, y compatible.
- Soporta Scorm en su totalidad.

Versión Utilizada

Moodle 1.5

Utilización de Moodle

Para aprender a trabajar con Moodle se utilizaron los cursos de prueba para estudiantes y profesores de la academia interactiva accesible en http://academia-interactiva.com/moodle/course/category.php?id=9

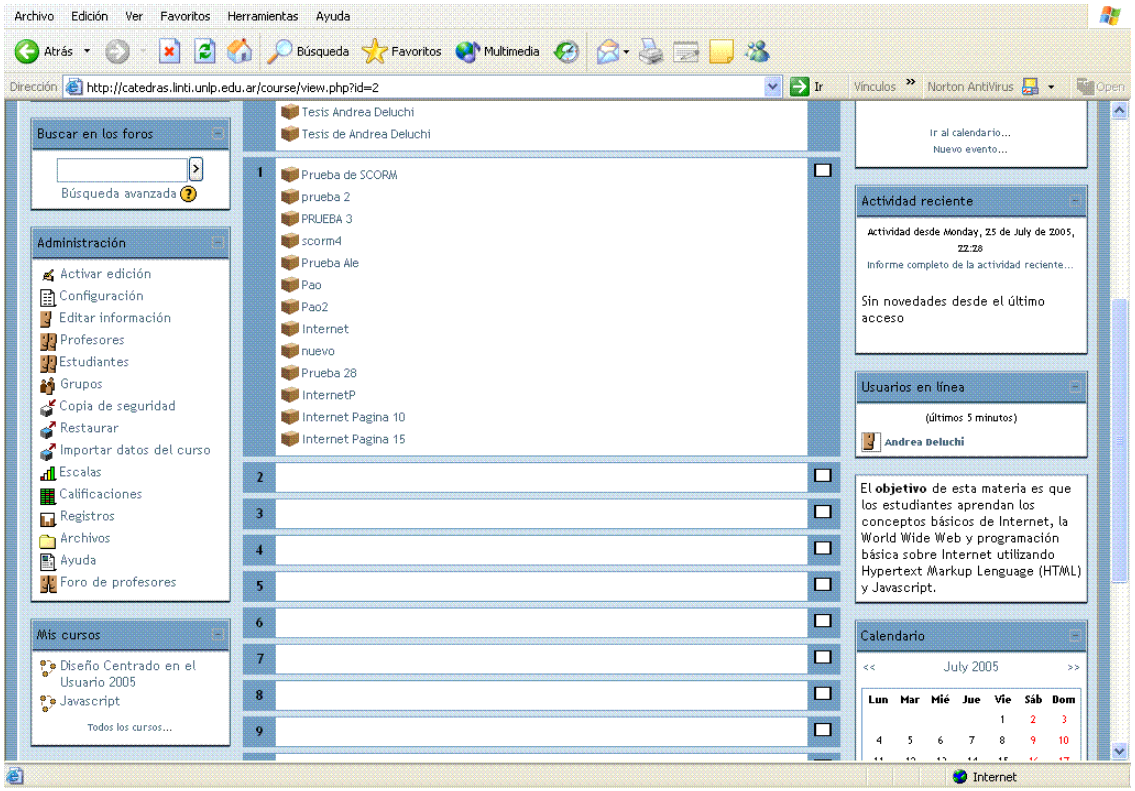
Subir el paquete Scorm

Subamos el paquete SCORM creado al Moodle para mostrar cómo el mismo curso puede navegarse en distintos LMS's que soporten SCORM.

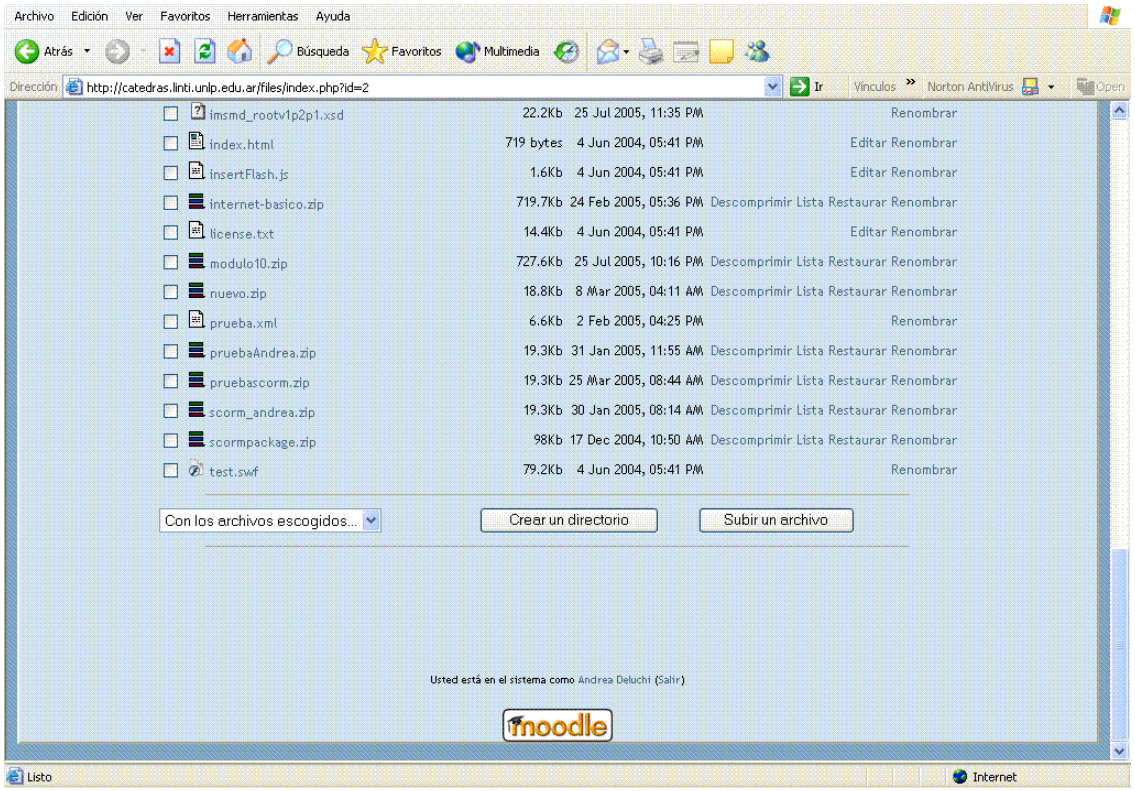
En un curso de Moodle podemos añadir distintos tipos de actividades. Y Scorm es una más de las que admite este entorno virtual.

Lo primero que necesitamos es tener nuestro paquete Scorm en el sistema. Por esto hace falta subir el fichero comprimido a la carpeta de ficheros del curso donde queremos tener la actividad Scorm.

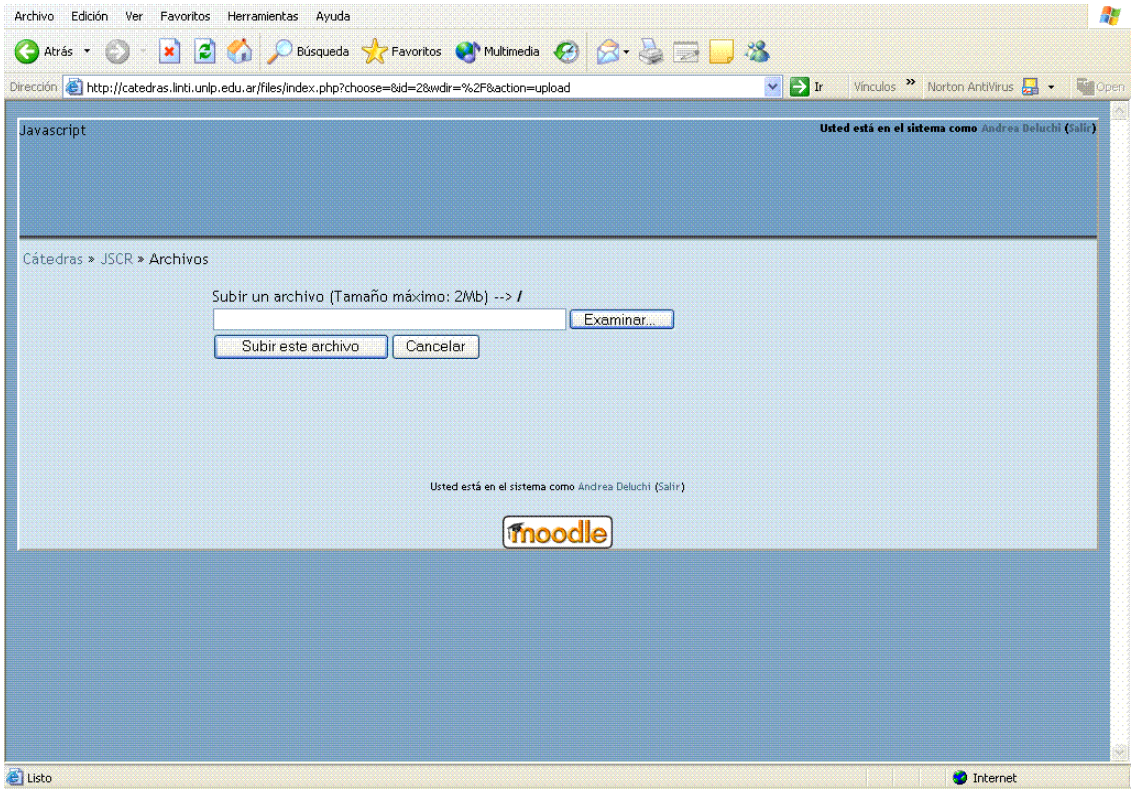
Con los privilegios de profesor vamos a nuestro curso – Administración – Archivos.



Hacemos click en el botón “Subir un Archivo”



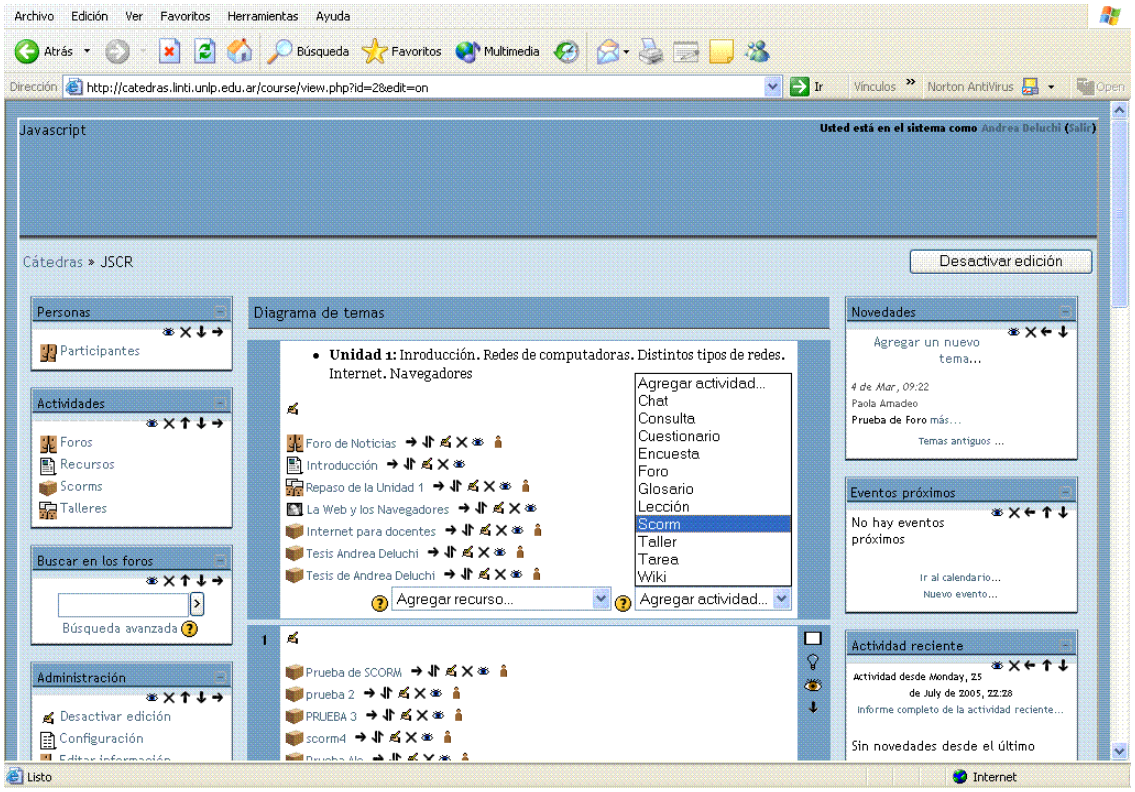
Cuando se nos abre el cuadro de diálogo seleccionamos nuestro paquete. Luego el paquete Scorm aparecerá en la lista de archivos del curso.



Crear la actividad Scorm

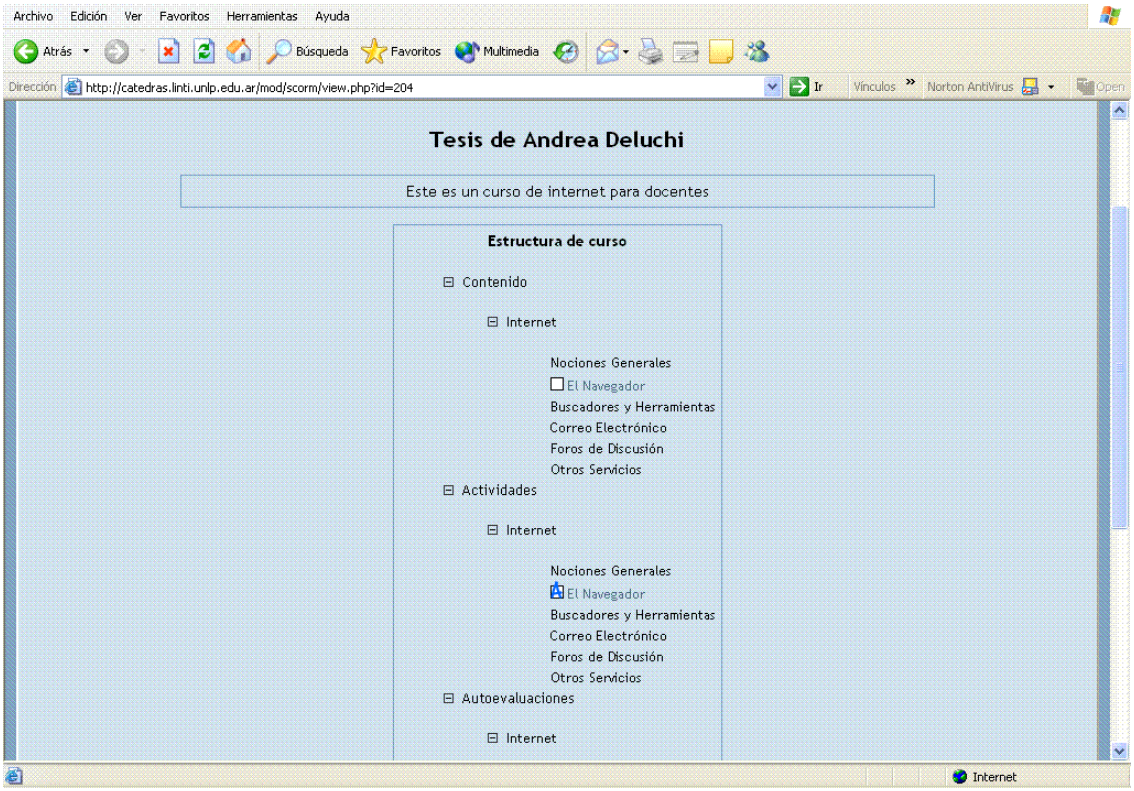
Debemos seleccionar el botón “Activar Edición”, lo cual nos permitirá añadir cualquier tipo de actividad.

Luego Seleccionamos la opción “Agregar Actividad Scorm”. Con esto se nos presentará una pantalla en la cual debemos seleccionar el fichero comprimido que cargamos anteriormente y completar información sobre la actividad que estamos creando.

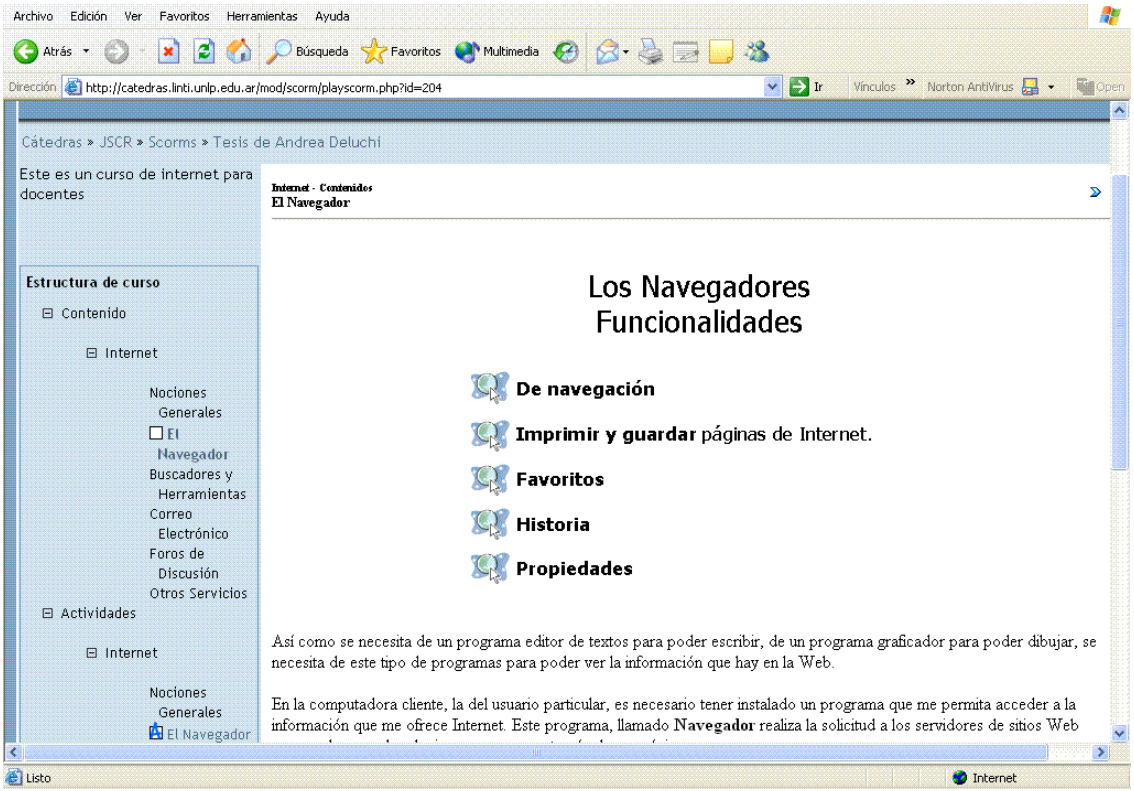


Mostremos cómo se ve nuestro curso en Moodle

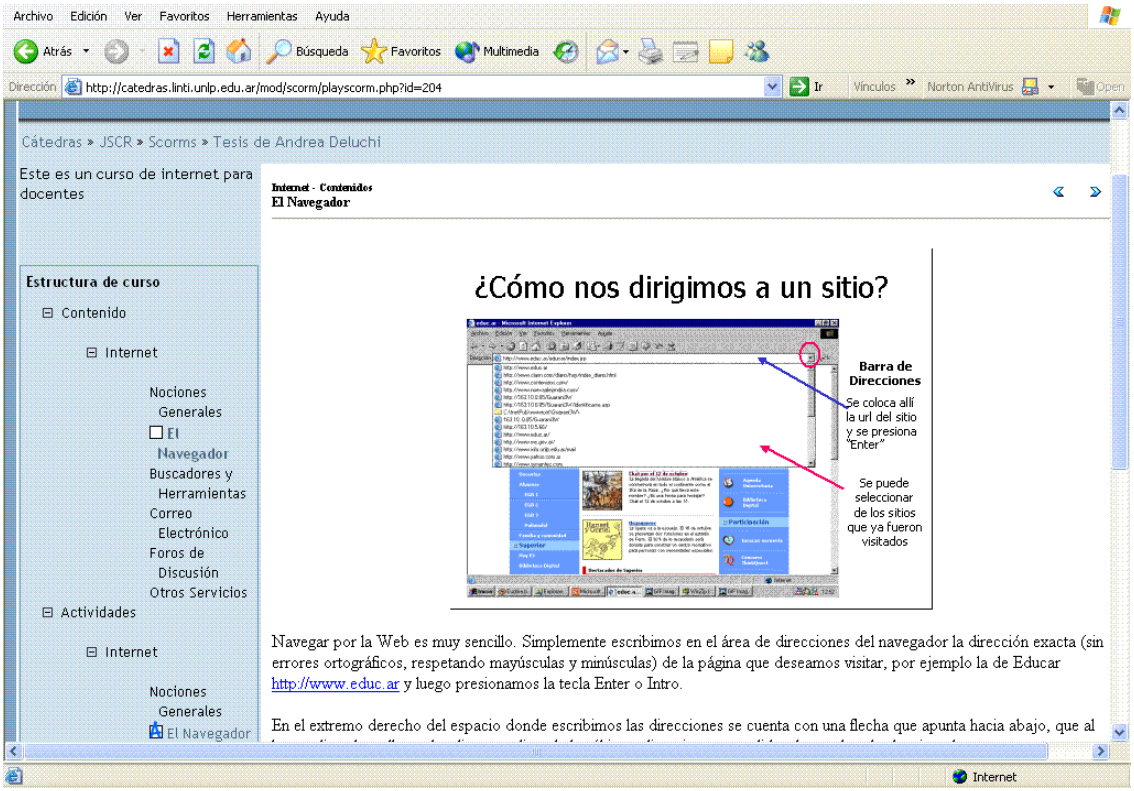
Seleccionando la actividad Scorm que hemos creado se presentará la siguiente pantalla:



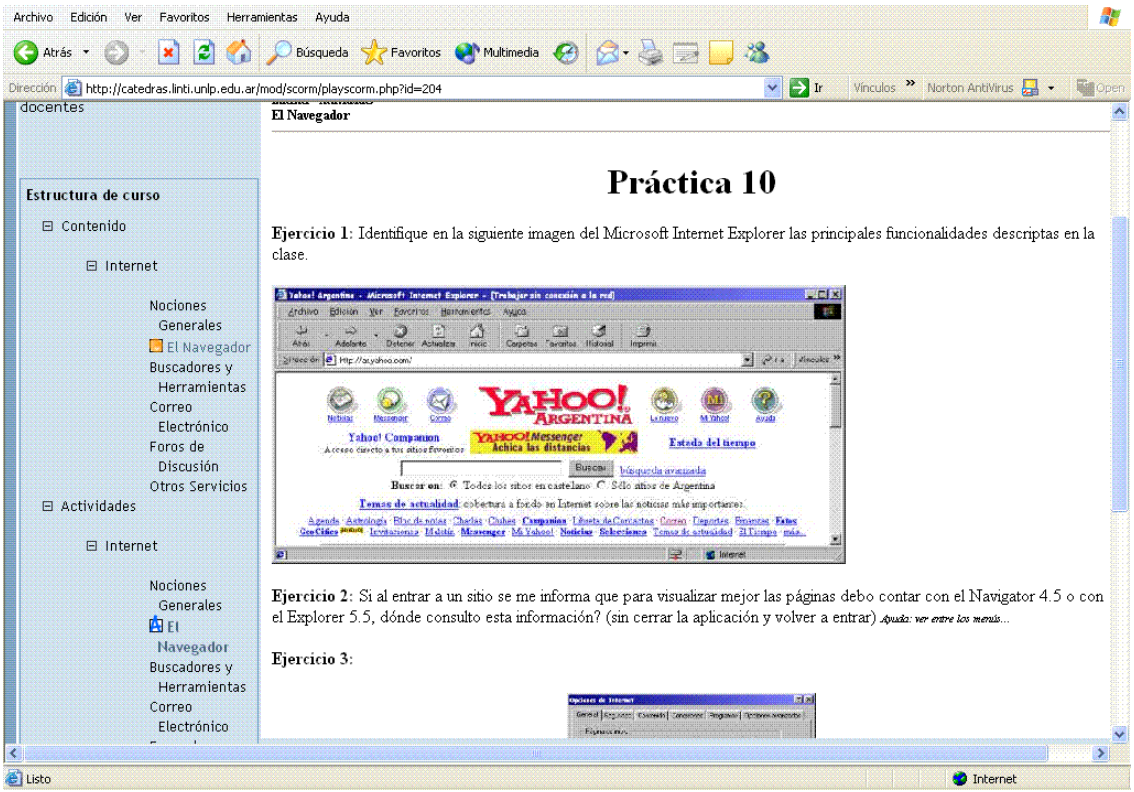
En esta pantalla se presenta el índice del curso. Comencemos la navegación del mismo:



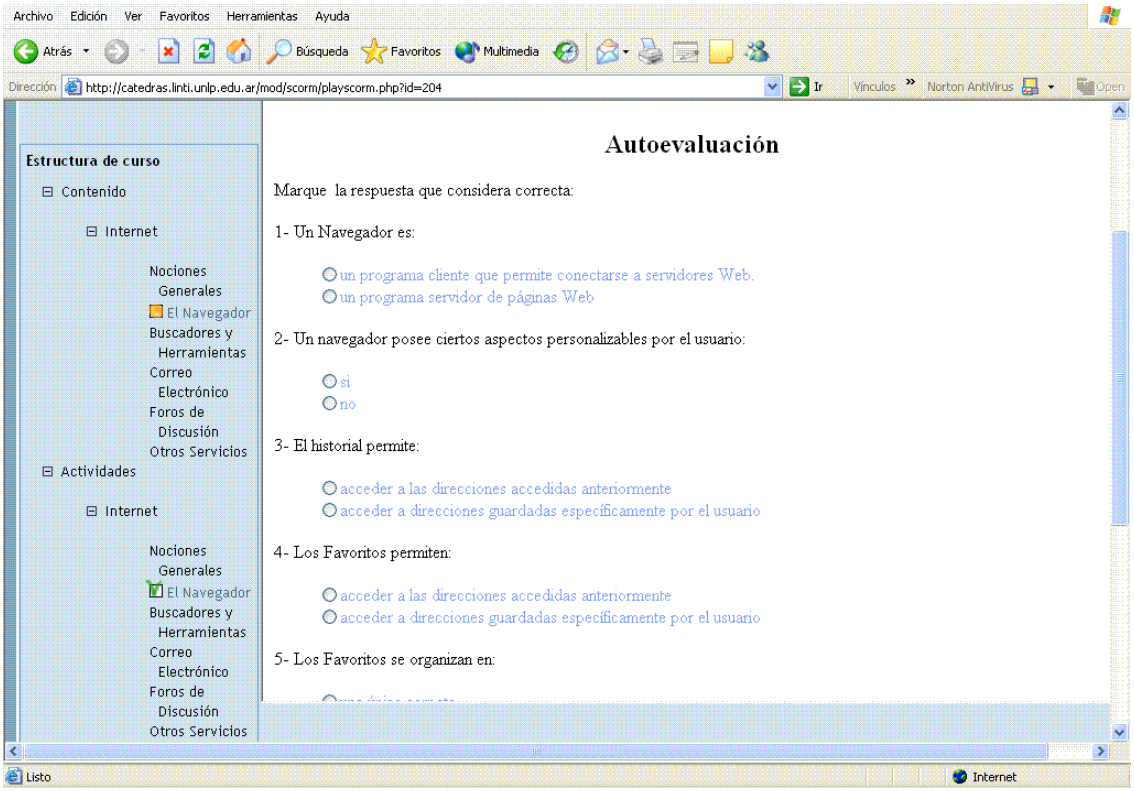
Navegamos la siguiente página



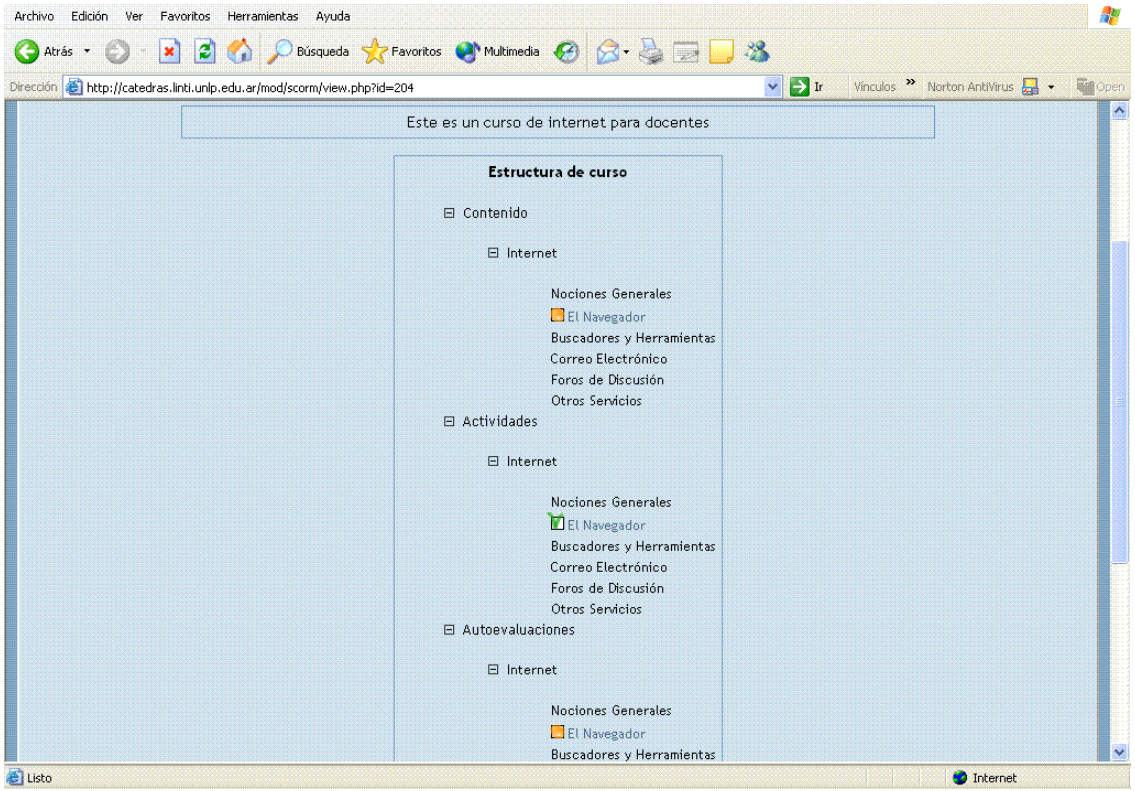
Navegamos la actividad del módulo



Ahora la autoevaluación



Si salimos del curso e intentamos navegarlo nuevamente se presenta el índice con algunos cambios

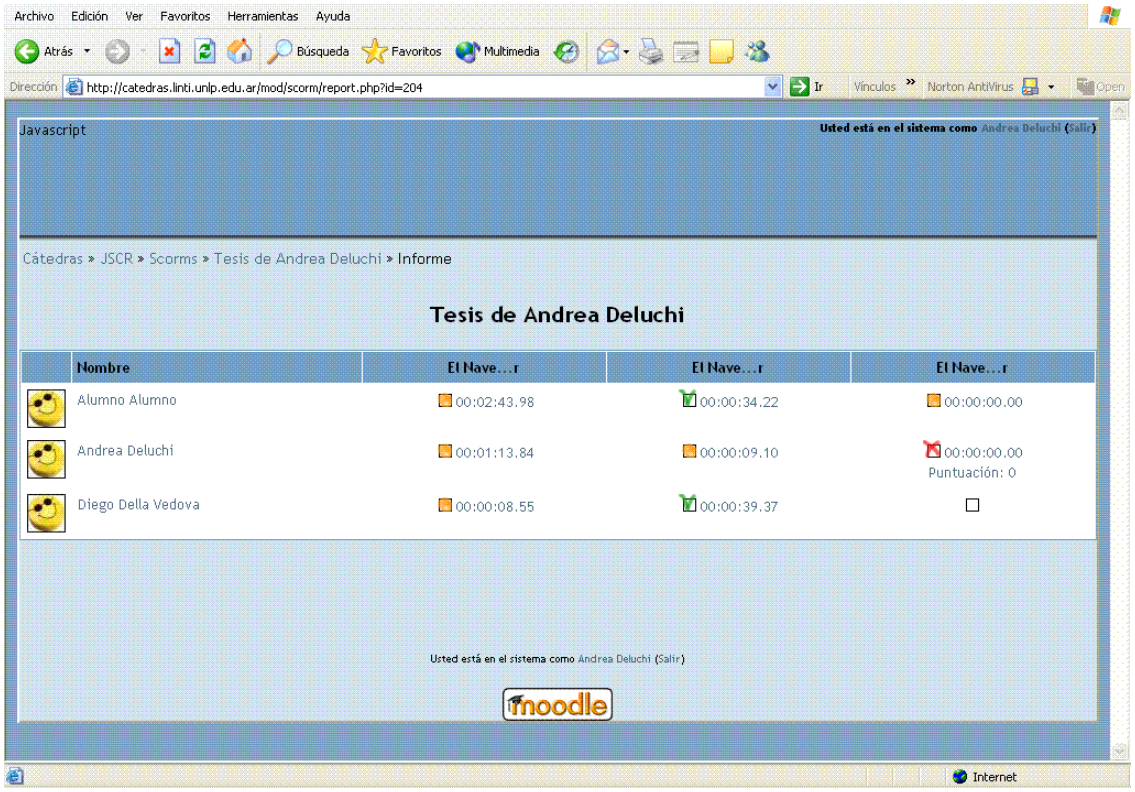


Vemos que se indica con diferentes colores el estado de navegación del mismo.

Por ejemplo en la actividad del segundo módulo (módulo que estamos navegando) se muestra un tilde en verde. Esto significa que se navegó toda la actividad propuesta. En el contenido se muestra un círculo amarillo con lo cual se indica que se comenzó la navegación pero no se culminó.

Lo mismo ocurre con la autoevaluación.

Si ingresamos a la sección de informes de la actividad Scorm que creamos, se muestra lo siguiente:

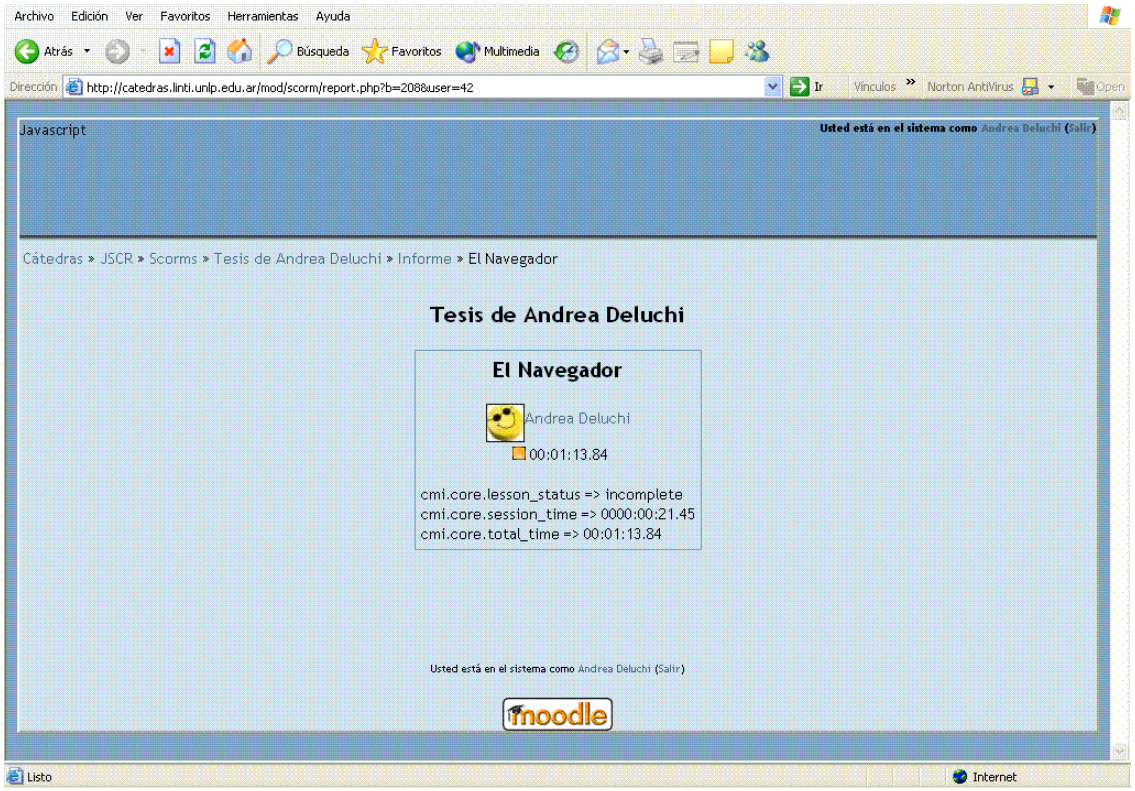


Vemos que el alumno Alumno Alumno ha navegado el curso. Vemos el tiempo de navegación de cada sección, el estado (indicado con colores) y un link a cada sección.

Lo mismo para el resto de los alumnos. Podemos observar que la alumna Andrea Deluchi navegó el contenido, la actividad y resolvió el examen, siendo insatisfactorio el resultado del mismo.

Para el alumno Diego Della podemos ver que no navegó el examen.

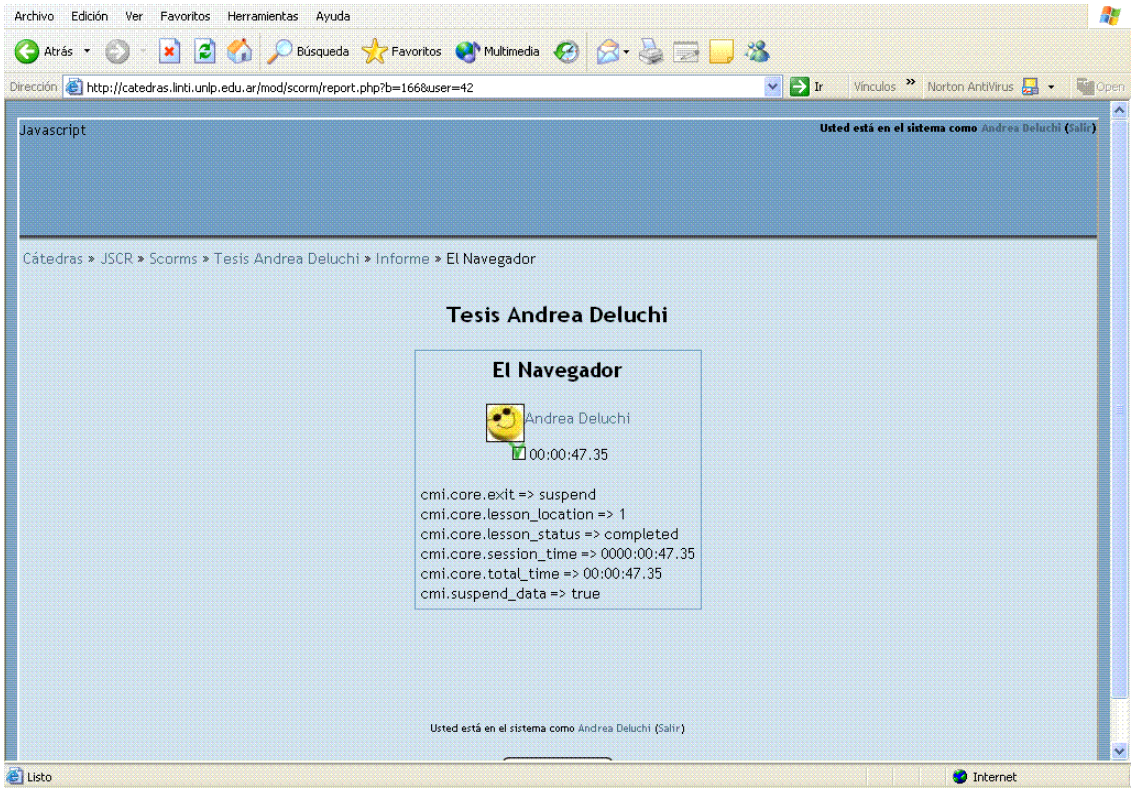
Si seleccionamos el primer link de la alumna Andrea Deluchi, se muestra lo siguiente:



La información presentada al profesor consiste en:

- Se indica que la navegación del contenido ha sido suspendida
- El tiempo total de navegación

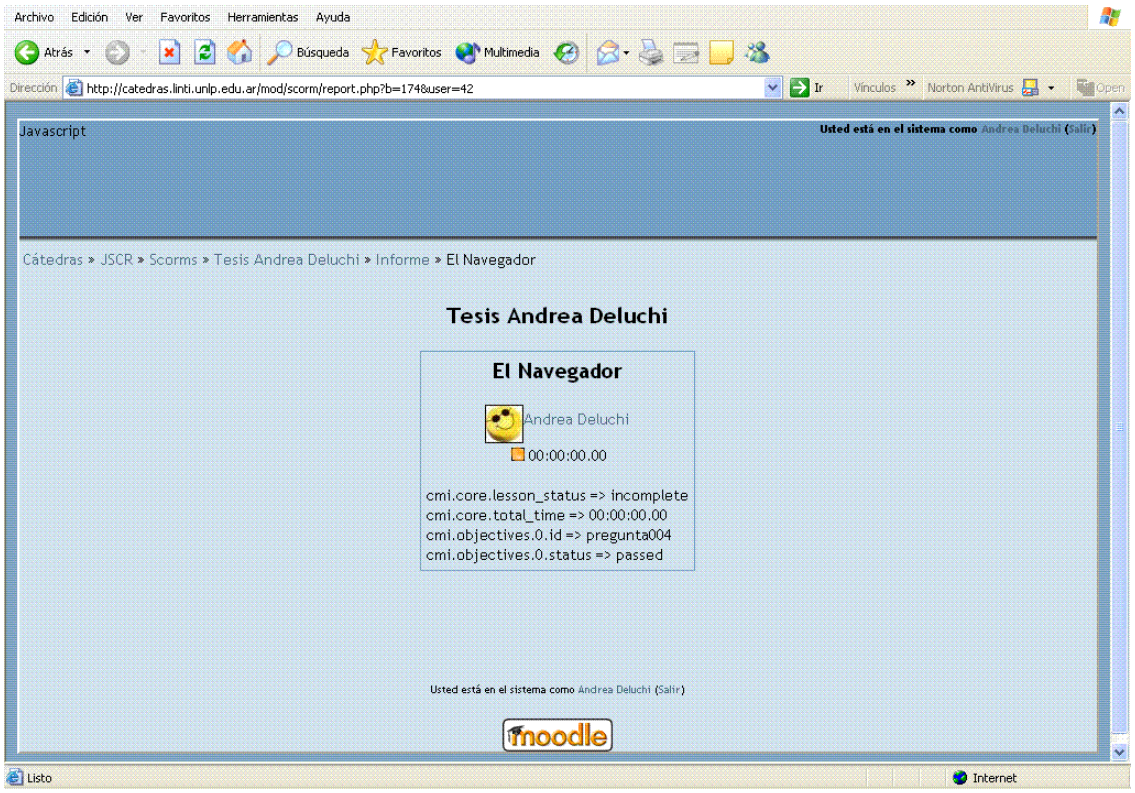
Seleccionemos el segundo link de la tabla de informes:



La información presentada al docente consiste en:

- Indicar que la navegación ha sido concluida
- El tiempo de navegación

Si seleccionamos el link tres de la tabla de informes:

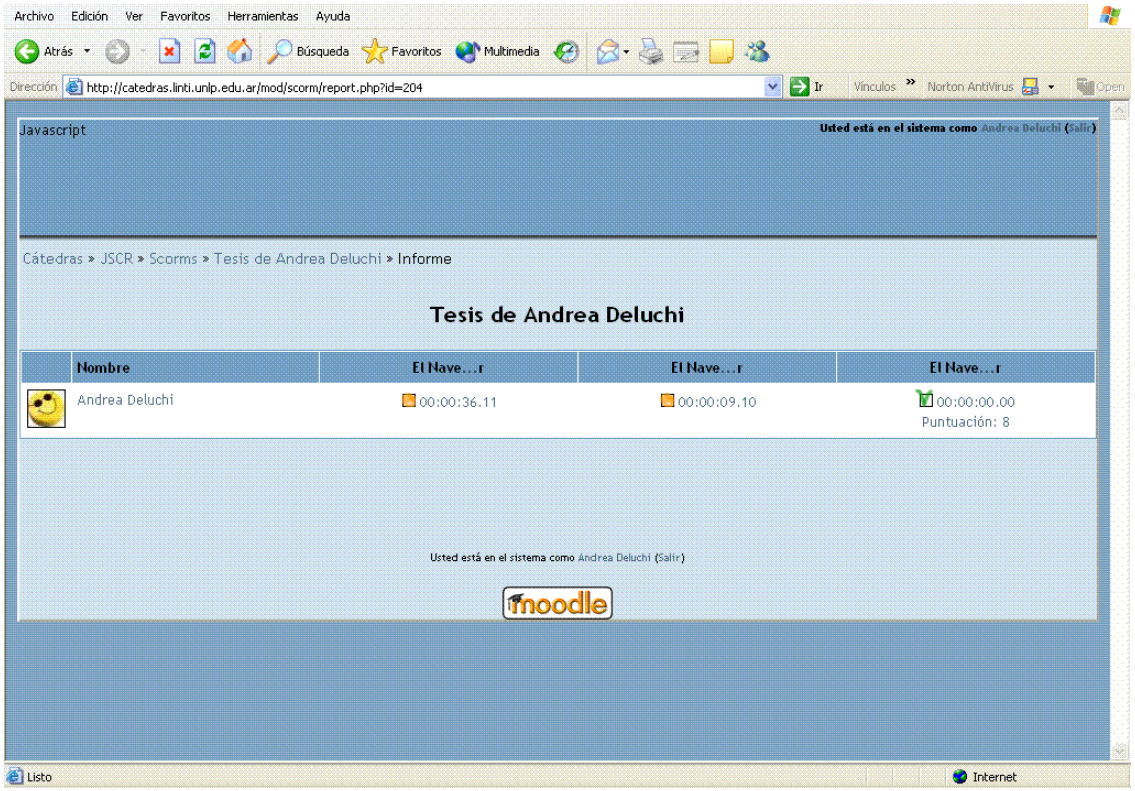


Se indica al profesor que:

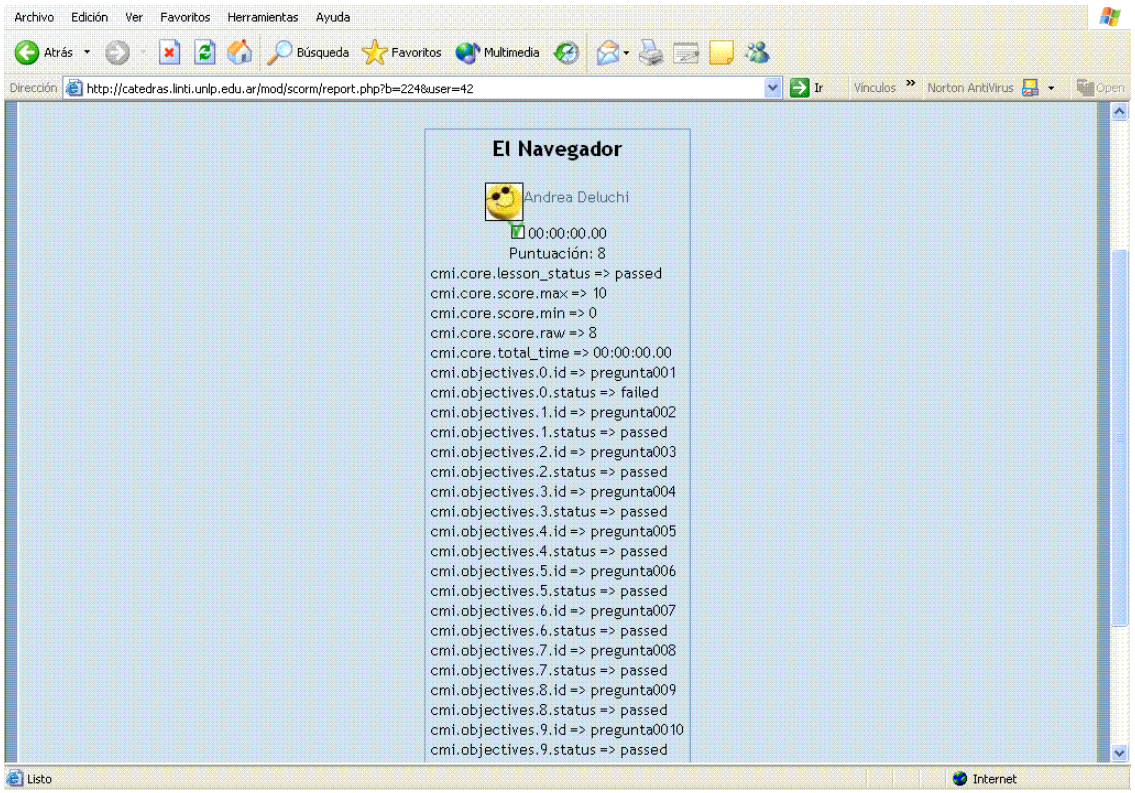
- El examen no se ha completado
- Que se resolvió la pregunta 4 correctamente, pero el examen no se envió.

Si navegamos nuevamente el curso y resolvemos el examen:

Veamos el informe de navegación:



Seleccionamos el link tres:



Se informa al profesor, que el alumno aprobó el examen con nota 8. Y se indica el resultado de cada pregunta. Vemos que el alumno respondió 9 preguntas con éxito y una fue errónea. Como las correctas suman y las incorrectas restan el resultado de la autoevaluación es 8.

De este modo podemos observar como a través del modelo de datos CMI el LMS recibe información del contenido y luego la informa a través de los informes de navegación.

Con esto hemos podido comprobar que un mismo paquete Scorm puede navegarse en distintos LMS's, donde cada uno, de acuerdo al grado de implementación del modelo de datos CMI, permite una comunicación con el contenido más o menos completa.

Conclusión:

Hemos analizado la potencia de diferentes estándares de e-learning, que brindan accesibilidad, interoperabilidad, durabilidad y reutilización de los materiales didácticos basados en Web.

De este análisis surge que el estándar con mayores ventajas en cuanto a su implementación es SCORM dado que permite que el contenido pueda comunicar información sobre el alumno a cualquier sistema LMS utilizando un modelo estándar basado en JavaScript.

Dichas ventajas están dadas en el modo en que Scorm integra los distintos esfuerzos realizados por organismos como AICC, IEEE e IMS. Esto hace que en un solo estándar encontremos las ventajas del resto.

Por otro lado SCORM facilita el modo de transportar el contenido de un sistema a otro utilizando únicamente un fichero instalable. Este fichero es de fácil construcción a través de cualquier herramienta, como por ejemplo Reload Editor, que fue utilizada para esta tesis. Cualquier persona que haya construido su curso, con importar todas las páginas en la herramienta y especificar cual va a ser la estructura del curso y sus metadatos, podrá fácilmente construir su paquete SCORM.

Estas ventajas han determinado la elección de SCORM para la construcción del curso sobre conceptos básico de internet para docentes.

La mayor complejidad en la construcción del curso SCORM, está dada en la utilización de funciones javaScript para permitir que el contenido envíe información al LMS. La complejidad de estas funciones crece a medida que crece el número y el detalle de la información que se quiere enviar.

Una persona que quiere que su curso no sea únicamente navegable en cualquier LMS que soporte SCORM, sino que también requiere comunicación con el mismo, se encuentra con esta dificultad. Por eso es importante que se continúe con el análisis de este estándar focalizando la comunicación entre el contenido y el LMS, a fin de construir herramientas que permitan construir páginas html en las cuales se pueda, de modo fácil, incluir funciones que permitan enviar información al LMS en donde se ejecute el curso.

Hoy en día, es muy sencillo construir un paquete SCORM pero no es así con la creación del contenido cuando éste debe establecer comunicación con el LMS.

El análisis realizado en esta tesis, así como la presentación del curso que permite comunicación con el LMS, y la presentación de las funciones javaScript que permiten llevar a cabo esta tarea, muestran claramente la necesidad de contar con herramientas que asistan a personas no expertas en la creación de contenido SCORM.

Es mi deseo que esta tesis sea el comienzo de un camino de estudio y de implementación del estandar SCORM, a fin de poder automatizar la creación de cualquier tipo de contenido SCORM y poder así, generar contenido portable para las cátedras de la facultad como se ha planteado en la introducción de este tesis.

Referencias

Referencias

Anido-Rifón, L. (2001). "Contribución a la Definición de Arquitecturas Distribuidas para Sistemas de Aprendizaje Basados en Ordenador Utilizando CORBA". Departamento de Ingeniería Telemática ETSI de Telecomunicación. Vigo, Universidad de Vigo.

Sancho, P. (2002). "Lenguajes de marcado y su aplicación en el dominio de las tecnologías de aprendizaje Web". Revisión de las principales iniciativas de estandarización. Madrid, Universidad Complutense de Madrid.

Jones, Edgard R. (2002). "Implications of SCORM; and Emerging E-learning Standards On Engineering Education". Proceedings of the 2002 ASEE Gulf-Southwest Annual Conference, The University of Louisiana at Lafayette, March 20 – 22, 2002, accesible en http://falcon.tamucc.edu/~ejones/papers/ASEE02.pdf.

Pisel, K; Lindsey, A. (2004). "Developing SCORM-Compliant Media-Rich Graduate-Level Distance Education: A Case Study of Best Practices". Proceedings of the 20th Annual Conference on Distance Teaching and Learning, August 4-6, 2004, Madison, Wisconsin, Monona Terrace Convention Center, accessible en http://www.uwex.edu/disted/conference/Resource_library/proceedings/04_1333.pdf.

Foix C., Zavando. S. (2002). "Estándares e-learning: Estado del Arte". Versión 1.0. Centro de Tecnologías de Información. INTEC, Corporación de Investigación Tecnológica de Chile.

Young M. (2000). "XML Step by Step". Second Edition. Publicado por Microsoft Press.

MSDN Latinoamérica (2001). "Intercambio de Información a través de Internet utilizando XML". Citado el 1 de mayo de 2001. S.L. Accesible en http://www.asia.microsoft.com/latam/msdn/articulos/2000/04/art02/#top

Gómez O. (2001). "Tutorial sobre XML". Citado el 1 de mayo del 2001. Málaga. Accesible en http://face.el.uma.es/imasd/xml/xml.html

Rossi G., Garrido A., Carvalho S. (1996). "Design Patterns for Object-Oriented Hypermedia Applications". In Pattern Languages of Program Design II, Addison Wesley, 1996.

Fernández-Manjón, B., Fernández-Valmayor, A. (1997). "Improving World Wide Web Educational Uses Promoting Hypertext and Standard General Markup Language Content-Based Features". Education and Information Technologies. Publisher: Springer Science+Business Media B.V., Formerly Kluwer Academic Publishers B.V. ISSN: 1360-2357 (Paper) 1573-7608

(Online). DOI: 10.1023/A:1018613330173. Issue: Volume 2, Number 3. Date: September 1997. Pages: 193 - 206

Fernandez-Manjón B., López Moratalla J., Martinez Ortiz I., Moreno Ger P. (2005). "Objetos Educativos y Estandarización en e-learning: experiencias en el sistema <e-aula>". Publicado en RED, Revista de Educación a Distancia. Publicación en línea. Murcia (España). Año IV. Número monográfico III.- 1 de Abril de 2005. Accesible en http://www.um.es/ead/red/M3/fernandez_manjon10.pdf

Pilar Sancho, Baltasar Fernandez-Manjón (2002). "<e-aula>: Entorno de Aprendizaje Personalizado Basado en Estándares Educativos". Publicado el 28 de octubre de 2002 en el taller en sistemas hipermedia colaborativos y adaptativos desarrollado dentro de las VII Jornadas de Ingeniería del Software y Bases de Datos. Granada. Accesible en http://giig.ugr.es/~taller/2003/pilar%20sancho.pdf.

David Griffiths, Josep Blat, Rocío Garcia y Sergio Sayazo (2004). **"**La aportación de IMS Learning Design a la creación de recursos pedagógicos reutilizables". Paper presentado en la conferencia SPDECE 2004 de Objetos Educativos, en la Universidad de Alcalá. Accesible en http://www.unfold-project.net/siteresources/archived/projectupdates/general_resources_folder/introsld/Griffiths_Final.pdf/download

Alejandro Escalante (2005). "Educación a Distancia usando Moodle, una alternativa bajo Software Libre". Publicado en el Segundo Congreso Nacional de Software Libre - USUARIA 2005 - XX Congreso Nacional de Informática, Teleinformática y Telecomunicaciones - USUARIA 2005. Accesible en http://www.softlibre.org.ar/softlibre/alejandro-escalante.pdf.

Sitios Consultados

Dublín Core (http://www.dublincore.org)

Expressing Simple Dublin Core in RDF/XML accesible en http://dublincore.org/documents/2001/11/28/dcmes-xml/

Dublin Core Metadata Element Set, Version 1.1. accesible en http://dublincore.org/documents/1999/07/02/dces/

Dublin Core Qualifiers accesible en http://dublincore.org/documents/2000/07/11/dcmes-qualifiers/

DCMI Abstract Model accesible en http://www.ukoln.ac.uk/metadata/dcmi/abstract-model/

Metainformación – Dublín Core. Elementos del conjunto de metadatas de Dublín Core: Descripción de Referencia, accesible en http://www.rediris.es/metadata/dublin_core_elements.es.html

LOM Learning Object Metadata (http://ltsc.ieee.org)

IEEE LTSC LOM (2001). Draft Standard for Learning Object Metadata, Final version 1.2 accesible en http://ltsc.ieee.org/wg12/index.htmlhttp://ltsc.ieee.org/wg12/index.htm

ISO/IEC 10646-1:2000 Information technology - Universal Multiple-Octet Coded Character Set (UCS), accesible en http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=29819

ISO/IEC 11404:1996 Information technology - Programming languages, their environments and system software, accesible en http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=19346

IMS Instructional Management System (http://www.imsglobal.org)

Proyecto PupitreNet. La Especificación del Proyecto IMS: Instructional Management System. Publicado en Edutec (Revista Electrónica de Tecnología Educativa). Número 13 / Noviembre 00, accesible en http://edutec.rediris.es/Revelec2/Revelec13/ims.PDF

IMS Learning Resource Metadata Information Model accesible en http://www.imsproject.org/metadata/mdinfo01.html

IMS Learning Resource Metadata XML Binding Specification accesible en http://www.imsproject.org/metadata/mdbind01.html

IMS Learning Resource Metadata Best Practices and Implementation Guide accesible en http://www.imsproject.org/metadata/mdbest01.html

IMS_ES (2002). Enterprise Specification, Version 1.1 Final Release accesible en http://www.imsglobal.org/enterprise/index.html

IMS_A (2002)**.** IMS AccessForAll Meta-Data accesible en http://www.imsglobal.org/accessibility/index.html

IMSCP_INFO (2001). IMS Content Packaging Information Model. Version 1.1.2 Final Specification accesible en http://www.imsglobal.org/content/packaging/index.html

IMSMD_INFO (2001). IMS Learning Resource Metadata Information Model. Version 1.0 Final Specification accesible en http://www.imsglobal.org/metadata/mdinfo01.html

LTSC (2002). Learning Object Metadata (LOM): Base Scheme. Version 6.1 accesible en http://ltsc.ieee.org/news/20021210-LOM.html

PAPI, I. (2000). Public and Private Information, Draft 6, Institute of Electrical an Electronics Engineers Inc accesible en http://edutool.com/papi/drafts/06/papi_learner_06.html

IMSLIP_INFO (2001). IMS Learner Information Package Information Model. Version 1.0 Final Specification, accesible en http://www.imsglobal.org/profiles/index.html

IMSQTI_INFO (2002). IMS Question & Test Interoperability: ASI Information Model Specification. Version 1.2 Final Specification, accesible en http://www.imsglobal.org/question/qtiv1p2/imsqti_asi_infov1p2.html

IMSLD_INFO (2003). IMS Learning Information Model. Version 1.0 Final Specification, accesible en http://www.imsglobal.org/learningdesign/ldv1p0/imsld_infov1p0.html

IMSSS_INFO (2002). IMS Simple Secuencing Information and Behavior Model. Version 1.0 Final Specification, accesible en http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_infov1p0.html

IMSRDCEO_BEST (2002). IMS Reusable Definition of Competency or Educational. Objective Best Practices. Version 1.0 Final Specification, accesible en http://www.imsglobal.org/competencies/rdceov1p0/imsrdceo_bestv1p0.html

IMSRDI_INFO (2003). MS Digital Repositories Interoperability Information Model. Version 1.0 Final Specification, accesible en http://www.imsglobal.org/digitalrepositories/index.html

EML Educational Modelling Language (http://eml.ou.nl)

EML (Learning Object Metadata), accesible en http://www.iua.upf.es/~jblat/material/doctorat/students/jccbis/Estandares_EML.htm

Scorm (http://www.adlnet.org/)

SCORM 2004 2nd Edition, accesible en http://www.adlnet.org/downloads/70.cfm

SCORM XML Controlling Document - SCORM CAM Version 1.3 Content Packaging Extensions XML XSD Version 1.0, accesible en http://www.adlnet.org/downloads/58.cfm

SCORM XML Controlling Document - SCORM CAM Version 1.3 Sequencing Extensions XML XSD Version 1.0, accesible en http://www.adlnet.org/scorm/history/2004/documents.cfm

SCORM 2004 Data Model Content Example, accesible en http://www.adlnet.org/scorm/history/2004/DMCE.cfm

SCORM 1.2, accessible en http://www.adlnet.org/scorm/history/12/index.cfm

AICC CMI001, accesible en http://www.aicc.org/docs/tech/cmi001v3-5.pdf http://www.aicc.org/docs/tech/cmi001v3-5.pdf

XML (http://www.w3.org/XML/)

Extensible Markup Language (XML) 1.0 Specification, accesible en http://www.w3.org/TR/REC-xml/

Processing XML 1.1 documents with XML Schema 1.0 processors, accesible en http://www.w3.org/TR/2005/NOTE-xml11schema10-20050511/

XML Schema Part 0: Primer Second Edition, accesible en http://www.w3.org/TR/xmlschema-0/

XML Schema Part 1: Structures Second Edition, accesible en http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Datatypes Second Edition, accesible en http://www.w3.org/TR/xmlschema-2/

Extensible Markup Language (XML), accesible en http://xml.coverpages.org/xml.html

RDF (http://www.w3.org/RDF/)

RDF/XML Syntax Specification (Revised), accesible en http://www.w3.org/TR/rdf-syntax-grammar/

RDF Vocabulary Description Language 1.0: RDF Schema, accesible en http://www.w3.org/TR/rdf-schema/

Resource Description Framework (RDF): Concepts and Abstract Syntax, accesible en http://www.w3.org/TR/rdf-concepts/

RDF Primer, accessible en http://www.w3.org/TR/2003/PR-rdf-primer-20031215/#reification

RDF y Metadata, accesible en http://www.xml.com/pub/a/2001/01/24/rdf.html.

Proyecto e-aula (http://eaula.sip.ucm.es/)

Moodle (http://moodle.org/)

Academia Interactiva Virtual. Aprendiendo a trabajar con Moodle, accessible en http://academia-interactiva.com/moodle/course/category.php?id=9

Reload Scorm Player (http://www.reload.ac.uk/scormplayer.html)

Reload Editor (http://www.reload.ac.uk/editor.html)