FACULTAD DE INFORMATICA
UNLP

TESINA DE LICENCIATURA

TITULO: Facilitando la creación y uso de objetos de aprendizaje en entornos de Software Libre

AUTORES: Nidia Banchero - Martín Dellarupe

DIRECTOR: Lic. Javier Diaz

CODIRECTOR: Lic. María Alejandra Schiavoni

CARRERA: Licenciatura en Informática

Resumen

Actualmente la creación de material educativo que respete los estándares es fundamental para lograr contenido reusable e interoperable. El proceso de creación de contenido compatible con SCORM comprende una serie de etapas que requieren conocer fundamentos de programación y la implementación del estándar en detalle. Estas etapas comprenden desde la identificación de los objetos de aprendizaje hasta la incorporación de metadatos y funciones de comunicación con el LMS para registrar la actividad del alumno. El objetivo de este trabajo es facilitar la incorporación de comunicación entre cada uno de los objetos de aprendizaje que integran un curso y el LMS que los soporta. Para lograr este objetivo extendimos un editor HTML basado en Software Libre, FCKEditor, agregándole la funcionalidad necesaria que incorpora código JavaScript en los OA para implementar algunas de las componentes de comunicación de la API de SCORM. En la elección de las funciones de comunicación se tomó como base el modelo de cursos desarrollados por el Instituto Nacional de Tecnología de la Información (ITI) Brasileño en su proyecto Centro de Difusión de Tecnología y Conocimiento (CDTC). Esta herramienta resulta fácil de usar y disminuye el nivel de complejidad en la creación de un curso SCORM por parte de personas no expertas en código HTML o JavaScript.

Líneas de Investigación

- E-Learning
- Entornos de aprendizaje (Metadatos, Estándares, SCORM)
- Software Libre

Trabajos Realizados

Para poder realizar este trabajo, debimos estudiar y manejar conceptos tales como educación a distancia, objetos de aprendizaje, recursos educativos reusables, entornos de aprendizaje y estándares. Investigar estos temas en profundidad, familiarizarnos con los conceptos y con las herramientas para trabajar sobre ellos. Se estudió en detalle el estándar SCORM y también el entorno Moodle, como ejemplo de sistema LMS Open Source. Se analizaron distintos editores HTML de código abierto con el fin de ver cual era la mejor opción para extenderlo e incorporarle contenido SCORM al material educativo que se genere. Se generó una aplicación "FCKScorm for E-Learning" que favorece al enriquecimiento de los cursos para hacerlos reusables y poder recuperarlos en motores de búsqueda SCORM. Se planteó un caso de uso.

Conclusiones

Nuestro trabajo consistió en tomar un editor HTML de código abierto y extenderlo para incorporarle contenido SCORM al material educativo que se genere, obteniendo una herramienta que favorece al enriquecimiento de los cursos para hacerlos reusables y poder recuperarlos en motores de búsqueda SCORM. La herramienta obtenida representa una contribución concreta y significativa para el desarrollo de contenido educativo en entornos de software libre, ya que permite generar recursos que respetan el estándar SCORM, lo cual no está soportado por las herramientas actuales bajo esta filosofía. Dado que existe una brecha entre la definición del modelo SCORM y el proceso global de desarrollo de cursos por parte de docentes que diseñan cursos de aprendizaje, el uso de esta herramienta puede resultar una ayuda considerable para que puedan utilizar el estándar SCORM más fácilmente.

Trabajos Futuros

Extensión de la herramienta desarrollada incorporando nuevas funciones de comunicación de manera que permita la creación de otros modelos de cursos. Creación de cursos en forma independiente del modelo que propone la aplicación.

Fecha de la presentación: Diciembre 2008

Facilitando la creación y uso de objetos de aprendizaje en entornos de Software Libre

Facultad de Informática
Universidad Nacional de La Plata

Director: Lic. Javier Diaz
Coodirector: Lic. María Alejandra Schiavoni
Alumnos: Martin Dellarupe Legajo: 1385/9
Alumnos: Nidia Banchero Legajo: 1633/7

Indice de Contenidos

Agradecimientos **5**

Introducción **6**

Unidad 1: Conceptos teóricos **7**

Introducción 7
1.1 E-Learning 7
1.1.1 Generalidades, procesos, herramientas y estándares 8
1.1.2 Plataformas de E-Learning 9
1.1.3 ¿Qué son los Estándares en sistemas E-Learning? 10
1.1.4 Estándares y Especificaciones 11
1.1.5 La importancia de los perfiles de aplicación 11
1.1.6 Servicios y prestaciones generales de los LMS 11
1.2 Los Objetos de Aprendizaje y Otros Recursos de Educación en la Web 15
1.2.1 Plataformas para el aprendizaje en Web 17
1.2.2 Objetos de aprendizaje 17
1.2.3 Metadatos. La clave para la reutilización 19
1.2.4 Bibliotecas digitales y repositorios de objetos de aprendizaje 20
1.3 SCORM (Sharable Content Object Reference Model) 21
1.3.1 Conceptos Básicos 22
1.3.1.1 SCO (Sharable Content Object). 22
1.3.1.2 Assets o Sharable Resources (Contenido compartido) 23
1.3.1.3 Metadatos 23
1.3.1.4 API de SCORM 24
1.3.2 LMS (Learning Management System) 24
1.3.3 Modelo para el tratamiento de contenidos 25
1.3.4 Entorno de ejecución SCORM 26
1.3.5 Proceso de empaquetamiento de los SCO´s 27
1.3.6 SCORM Content Aggregation (Curso, asociación de clases) 27
1.3.7 Comunicación del SCO y el LMS mediante el API 30
Conclusiones 37

Unidad 2: Herramientas para el manejo de cursos estandarizados..... 38

Introducción 38
2.1 MOODLE 38
2.1.1 Características generales 38
2.1.2 Utilizando Moodle para cargar un paquete SCORM 40
2.1.3 Añadir un paquete SCORM al curso 40
2.2 Reload Editor & Reload Player 42
2.2.1 Entorno de trabajo de Reload Editor 43
2.2.2 Empaquetamiento de contenidos con Reload Editor 44
2.2.3 Empaquetamiento de contenidos ADL SCORM con Reload Editor 46
2.3 Editores HTML Open Source 48
2.3.1 HTMLArea 48
2.3.1.1 Características 49
2.3.1.2 Entorno del editor en la aplicación Moodle 50
2.3.1.3 Posibles alternativas de reemplazo en Moodle 53
2.3.1.4 Cómo se inserta el editor HtmlArea en cualquier aplicación 54
2.3.2 FCKeditor 56
2.3.2.1 La barra de herramientas 57
2.3.2.2 Instalación de FCKeditor en cualquier aplicación 60
2.3.2.3 Definiendo los Skins 60
2.3.2.4 Determinando los Botones del Editor 61
2.3.3 NVU 62
2.3.3.1 Características 62
2.3.3.2 Modos de edición 63
2.3.3.3 Propiedades 63
2.3.3.4 Entorno de trabajo 64
Conclusiones 67

Unidad 3: Construcción de la aplicación "FCKScorm for E-Learning" 68

Introducción 68
3.1 Etapas iniciales del desarrollo de la aplicación 68
3.1.1 Propuesta inicial 68
3.1.2 Modelo de curso seleccionado 69
3.1.3 Cambios realizados a la propuesta inicial 70
3.2 Especificación de la aplicación 70
3.2.1 Funcionalidades provistas por la aplicación 71
3.3 Desarrollo de la aplicación "FCKScorm for E-Learning" 76

3.3.1 Instalación del Editor HTML FCKeditor 76
3.3.2 Creación de carpetas y archivos de la aplicación 76
3.3.3 Funciones agregadas al editor FCKeditor 78
3.3.4 Componentes internas del editor FCKeditor 81
3.3.5 Cambios realizados al código fuente original 83
3.3.6 Skins. Definición de imágenes para la barra SCORM 92
3.4 Ejemplo de un curso SCORM con FCKScorm 93
3.4.1 Pasos para el armado del curso con FCKScorm 94
3.4.2 Armado del paquete SCORM con Reload Editor 102
3.4.3 Visualización del curso SCORM con Moodle 106

Conclusión 116

Referencias: 118

Agradecimientos

Muchas personas han contribuido, de una u otra forma, algunas incluso sin saberlo, a la elaboración de este trabajo. A todas ellas queremos expresar nuestro más sincero agradecimiento.

En especial:

A nuestros hijos, Franco y Camila, por saber esperarnos y entendernos cuando estábamos ocupados haciendo "un trabajo para la facultad".

A nuestra amiga Dalila y su familia, por sus consejos y horas de guardería con nuestros hijos.

Al director de nuestra tesis, Lic. Javier Díaz y a la codirectora, Lic. María Alejandra Schiavoni, por la gran ayuda prestada, no sólo técnica, dispuestos en todo momento a atender las dudas y propuestas que iban surgiendo.

Martín:

A mi esposa Nidia, el motor de este trabajo, ya que sin su fuerza de voluntad inquebrantable, este trabajo nunca hubiera podido ser realizado.

A toda mi familia. En especial a mis padres, hermanos y abuelos por apoyarme y brindarme siempre las herramientas necesarias para poder lograr mis objetivos.

Nidia:

A mi esposo Martín, por nunca decirme NO y acompañarme en todo. Gracias a ese apoyo hoy pudimos concluir este trabajo.

A mi mamá Eve y a mis hermanos por haberme ayudado a llegar a este momento, ya que sin el esfuerzo de todos ellos no podría haber cumplido muchos de mis objetivos.

Y muy especialmente a mi mamá, por enseñarme a no bajar los brazos nunca, recordándome siempre su lema: "Siempre que uno quiere puede, solo hay que buscar la forma de lograrlo".

Introducción

Internet se ha posicionado como la tecnología más exitosa de la historia de las comunicaciones y es por ello que se ha convertido en un medio fundamental para comunicar y transmitir diferentes tipos de contenidos con objetivos diversos.

La educación en general y la educación a distancia en particular forman parte de los temas que más se vieron favorecidos por Internet. Gracias a la Web, que a través de su interfaz gráfica y de la tecnología de hipertextos ha permitido desarrollar nuevas aplicaciones y descubrir nuevos horizontes, se generó un medio amigable y abierto que posibilita desde muchos puntos la interacción del alumno, surgiendo así diversos modelos de educación basados en Web.

La educación a distancia ha tenido un proceso de evolución a la par de las nuevas tecnologías que van surgiendo y la modalidad más innovadora es el *E-Learning*, que hace uso de los servicios y facilidades de Internet. Existen herramientas y recursos de apoyo que permiten a docentes y alumnos llevar a cabo el aprendizaje. Entre las herramientas más utilizadas para los cursos *E-Learning* están las plataformas LMS (*Learning Management Systems*), que facilitan el aprendizaje distribuido y colaborativo a partir de actividades e información, de forma síncrona o asíncrona, utilizando los servicios de comunicación de Internet como el correo, los foros, las videoconferencias y el chat.

La aparición de los numerosos sistemas y recursos educativos, ha llevado a establecer recomendaciones y estándares que permitan su uso eficiente. La estandarización de las tecnologías aplicadas a la formación pretende facilitar la reutilización de recursos y la interoperabilidad entre sistemas y software heterogéneo. El estándar que ha tenido más auge es SCORM (*Sharable Content Object Reference Model*), el cual describe cómo las unidades de contenido se relacionan unas con otras, a diferentes niveles de granularidad, usando objetos de aprendizaje que cumplan con su especificación, a los cuales llama SCO (*Sharable Content Object*).

Cualquier persona que haya construido un curso SCORM, habrá podido observar que utilizando una herramienta como Reload Editor, fácilmente puede importar en ella los objetos que forman parte del curso (documentos Html, imágenes, archivos de audio, etc.) y especificar cuál va a ser la estructura del curso y sus metadatos, obteniendo un paquete SCORM. Si bien la construcción de un paquete SCORM es una tarea sencilla, la mayor complejidad está dada en la utilización de las funciones JavaScript, que permiten que el contenido envíe o tome información del LMS. La complejidad de estas funciones crece a medida que crece el número y el detalle de la información que se quiere enviar.

De esta problemática surge nuestro interés en continuar en el camino de estudio del estándar SCORM, a fin de poder facilitar la creación de cualquier tipo de contenido que respete este estándar y poder así, generar recursos portable, reusable e interoperable. Nuestro proyecto continúa el Trabajo Final presentado por la alumna Andrea Deluchi "Usando XML para metaanotar recursos educativos", para lograr la posibilidad de optimizar el proceso de creación de material educativo para las distintas cátedras de la facultad.

Usando esta tesis como base, el objetivo global que pretendemos conseguir es disminuir el nivel de complejidad del proceso de agregar metadatos y funciones de comunicación a las páginas HTML que forman parte del curso SCORM. Este objetivo se llevo a cabo construyendo una herramienta que interactúa directamente con la estructura interna del estándar para facilitar la incorporación del mismo en distintas clases de cursos por parte de personas no expertas en lenguajes y técnicas de programación.

Unidad 1: Conceptos teóricos

Introducción

"*Internet tiene el potencial de revolucionar la educación y la educación tiene el potencial de revolucionar Internet*".

Esta unidad dará el marco teórico de nuestro trabajo de grado. Mencionaremos los temas principales que se contemplaron en el estudio, la información documental reunida y aspectos técnicos tales como definiciones de términos claves dentro del estudio.

Se hará una revisión de los conceptos relacionados al tema de objetos de aprendizaje, así como a otros que se relacionan a éste y al *E-Learning*. También se describirá el estándar SCORM por ser el objetivo principal al que apunta nuestro trabajo de grado.

Esto permitirá contar con un conocimiento general de la teoría que le da marco al desarrollo de este trabajo.

1.1 E-Learning

> ***"Consiste en la educación y capacitación a través de Internet. Este tipo de enseñanza online permite la interacción del usuario con el material mediante la utilización de diversas herramientas informáticas. "***

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*.

1.1.1 Generalidades, procesos, herramientas y estándares

Los procesos de aprendizaje a través de la historia se han visto influenciados por las diferentes tendencias culturales, políticas, económicas, sociales, etc., y hoy en día hay que agregar un nuevo tipo de influencia: "la influencia tecnológica".

Con el auge y desarrollo de la informática y las telecomunicaciones, los procesos de educación se han visto en algunos casos subordinados a los soportes tecnológicos que facilitan el acceso al conocimiento.

Antes de los años 70 ya se preveía la utilización de herramientas audiovisuales como medios de aplicación de la tecnología a la educación y desde la década del 70 se conoce el término *Computer-Based Training* (CBT) (Educación basada en el uso del computadora), como uno de los primeros conceptos que fundamentan lo que hoy se conoce como *E-Learning*.

Los CBT evolucionaron en los años 80 a través de sistemas tutores inteligentes, los cuales usaban técnicas de Inteligencia Artificial (IA) para representar el conocimiento, estos sistemas fueron llamados Entornos de Aprendizaje Inteligentes (*Intelligent Learning Environments* - ILEs).

Desde los comienzos de la década del 90 hubo una autentica revolución en los CBTs, gracias a el desarrollo y masificación de Internet (WBEC, 2000) y es por ello que actualmente no se deben separar los conceptos de *E-Learning*, de los de *Web-Based Learning*.

En los últimos años se han desarrollado un gran número de sistemas *E-Learning*, denominados LMS (*Learning Management System*), muchos de ellos respondiendo a pedidos locales de instituciones o por empresas de desarrollo buscando imponer un sistema comercial. Cada uno de estos LMS fue desarrollado con paradigmas, lenguajes de programación y contextos distintos; pero con el mismo objetivo: eliminar las barreras de tiempos y espacios favoreciendo a los procesos de enseñanza-aprendizaje. Con el objetivo de que estos LMS puedan compartir sus recursos, intercambiando sus contenidos (cursos, alumnos, materiales, etc.) es que se desarrollaron especificaciones que definen la estructura de lo que se llama "*Sharable Content Object*".

Luego a partir de estas especificaciones muchos sistemas cambiaron y otros se desarrollaron siguiendo estas estructuras. El conjunto de especificaciones *Sharable Content Object Reference Model* (SCORM) desarrollado por el *Advanced Distributed Learning* (ADL), son las principales y las que más se están adoptando en la industria.

Algunas definiciones de *E-Learning*:

- *E-Learning* es la convergencia de aprendizaje e Internet. *(Bank of America Securities)*
- *E-Learning* es el uso de tecnología de redes para el diseño, entrega, selección, administración y extensión del aprendizaje. *(Elliott Masie, The Masie Center)*
- *E-Learning* es el aprendizaje facilitado en Internet. Los componentes de *E-Learning* pueden incluir contenidos entregados en múltiples formatos, gestión de experiencias de aprendizaje, redes de trabajo de alumnos, desarrolladores y expertos en contenidos.

- *E-Learning* provee rapidez en el aprendizaje a costos reducidos, incrementando el acceso a la educación con responsabilidades claras para todos los participantes en este proceso. *(Cisco Systems)*
- *E-Learning* es educación en línea entregada en forma síncrona (tiempo real, dirigida por un instructor) o asíncrona.
- *E-Learning* traslada las experiencias de aprendizaje fuera de la tradicional aula de clases, esto es aprendizaje en cualquier momento y en cualquier lugar, sin barreras geográficas o de agenda, confiando en Internet para el acceso a los materiales de aprendizaje e interactuando con expertos y alumnos semejantes

Se define *E-Learning* como la experiencia de obtener conocimiento y habilidades a través de la entrega electrónica de educación, entrenamiento o desarrollo profesional. Esto abarca educación a distancia y aprendizaje asíncrono y puede ser entregada en un ambiente a pedido o en un formato personalizado para un alumno en particular.

E-Learning, o aprendizaje en línea es el término aplicado al aprendizaje disponible en Internet. Esto se refiere al uso de la tecnología de redes de computadoras para crear, entregar, gestionar y soportar procesos de aprendizaje en cualquier momento en cualquier lugar. Estas tecnologías incluyen texto digital, gráficos, modelos en 3-D, audio y vídeo; disponibles para la creación de recursos multimedia que están dirigidas a un amplio espectro de accesibilidad y necesidades de aprendizaje. Adicionalmente herramientas de comunicación como chat en tiempo real, conferencia de audio y vídeo, foros de discusión asíncrona, permiten a los alumnos interactuar en línea con unos y otros, así como con los tutores.

En forma general el término *E-Learning* se refiere ampliamente al aprendizaje basado en tecnología, actualmente parece enfocarse en métodos basados en Web, pero frecuentemente es usado en su más amplio contexto. De igual forma se puede concluir de estas definiciones que los tres grandes componentes de los procesos de *E-Learning* son:

1) Las tecnologías de soporte a los procesos de aprendizaje (redes, hardware, software y herramientas en forma general),
2) Los contenidos o elementos contenedores de información y
3) Las personas que interactúan en el proceso de aprendizaje y de soporte al aprendizaje.

1.1.2 Plataformas de E-Learning

Existen muchas herramientas que brindan soporte a los procesos de *E-Learning*, las cuales se pueden clasificar de acuerdo a su funcionalidad en:

1) ***Herramientas de autor****:* esencialmente son herramientas de creación de recursos multimedia. Típicamente usadas sobre una estación de trabajo individual por un profesional de multimedia para crear recursos multimedia que pueden ser adicionados como módulo dentro de un sistema de gestión. Como ejemplo de este tipo de herramientas: MS-Power Point, Macromedia - Director, Authorware, MSFront Page, etc.

2) ***Aulas de clase virtuales en tiempo real****:* son herramientas que facilitan la entrega de contenidos de forma síncrona y en tiempo real. Como por ejemplo la videoconferencia.

3) ***Sistemas de gestión de aprendizaje* (*Learning Management Systems-LMS*)**. Son herramientas empresariales usadas para gestionar actividades de aprendizaje a través de la habilidad para catalogar, registrar y hacer seguimiento tanto de quienes aprenden como de quienes enseñan y de los contenidos enseñados.

1.1.3 ¿Qué son los Estándares en sistemas E-Learning?

Al hablar sobre un estándar *E-Learning*, nos estamos refiriendo a un conjunto de reglas en común para las compañías dedicadas a la tecnología *Learning*. Estas reglas especifican cómo los fabricantes pueden construir cursos online y las plataformas sobre las cuales son impartidos estos cursos de tal manera que puedan interactuar unas con otras. Estas reglas proveen modelos comunes de información para cursos *E-Learning* y plataformas LMS (*Learning Management System*), que básicamente permiten a los sistemas y a los cursos compartir datos. Esto también nos da la posibilidad de incorporar contenidos de distintos proveedores en un solo programa de estudios.

Estas reglas además, definen un modelo de empaquetamiento estándar para los contenidos. Los contenidos pueden ser empaquetados como "objetos de aprendizaje" (*Learning Object* o OA), de tal forma de permitir a los desarrolladores crear contenidos que puedan ser fácilmente reutilizados e integrados en distintos cursos.

Finalmente, los estándares permiten crear tecnologías de aprendizaje más poderosas, y "personalizar" el aprendizaje basándose en las necesidades individuales de los alumnos.

Básicamente, lo que se persigue con la aplicación de un estándar para *E-Learning* es lo siguiente:

- **Durabilidad:** que la tecnología desarrollada con el estándar evite lo obsoleto de los cursos.
- **Interoperabilidad:** que se pueda intercambiar información a través de una amplia variedad de LMS.
- **Accesibilidad:** Que se permita un seguimiento del comportamiento de los alumnos.
- **Reusabilidad:** Que los distintos cursos y objetos de aprendizaje puedan ser reutilizados con diferentes herramientas y en distintas plataformas.

Esta compatibilidad ofrece muchas ventajas a los consumidores de *E-Learning*.

Garantizan la viabilidad futura de su inversión, impidiendo que sea dependiente de una única tecnología, de modo que en caso de cambiar de LMS la inversión realizada en cursos no se pierde.

Aumenta la oferta de cursos disponibles en el mercado, reduciendo de este modo los costos de adquisición y evitando costosos desarrollos a medida en muchos casos.

Posibilita el intercambio, permitiendo que las organizaciones obtengan rendimientos extraordinarios sobre sus inversiones.

Facilita la aparición de herramientas estándar para la creación de contenido, de modo que las propias organizaciones puedan desarrollar sus contenidos sin recurrir a especialistas en *E-Learning*.

1.1.4 Estándares y Especificaciones

Frecuentemente los términos estándar y especificación se utilizan indistintamente, no obstante, es importante puntualizar su diferencia. Si una tecnología, formato o método ha sido ratificado por algún organismo oficial de estandarización, se trata de un estándar. Pero si una tecnología, formato o método propuesto no ha sido aprobado por algún organismo oficial de estandarización, se trata de una especificación.

El objetivo de establecer un estándar para la definición de metadatos educativos es alcanzar un acuerdo en las características que un elemento de aprendizaje (recurso didáctico, método, técnica) independientemente del sistema informático y el hardware que se utilice debe tener para permitir que los ambientes de *E-Learning* cuenten con las siguientes "habilidades" mencionados anteriormente, accesibilidad, interoperabilidad, durabilidad y reusabilidad.

1.1.5 La importancia de los perfiles de aplicación

Tanto en la práctica como en escenarios empíricos, la implementación de estándares o especificaciones en contextos educativos específicos no es evidente. Cada comunidad, región o país tiene sus propias particularidades, por lo que para el marcado de elementos educativos es necesario considerar diferentes cuestiones.

Por ello, una parte importante del proceso en el perfeccionamiento y aplicación de un estándar es el desarrollo de perfiles de aplicación (*application profiles*), que crean nuevos esquemas dirigidos a situaciones específicas combinando y utilizando uno o más estándares o especificaciones, sin agregar nuevos elementos y manteniendo la interoperabilidad con los estándares (o especificaciones) originales.

1.1.6 Servicios y prestaciones generales de los LMS

En todo proceso de *E-Learning* se identifican tres roles claramente definidos:

1) El que aprende (alumno, estudiante, aprendiz).
2) El que enseña (profesor, maestro, tutor, etc.)
3) El que soporta el proceso (técnicos en redes, expertos en uso de herramientas de *E-Learning*).

A continuación se enumeran de forma general los servicios y prestaciones que típicamente ofrecen las herramientas de LMS mas conocidas.

- ***Servicios para el docente***

Gestión del curso:

- Planificación del curso: estimación y definición de tiempo, recursos, objetivos, etc.
- Vista preliminar del curso.
- Diseño educacional: prerrequisitos, presentación de la información, Personalización del plan de estudios para los alumnos.

Gestión de alumnos:

- Monitorización de alumnos: calificaciones, entrega de trabajos, cumplimiento de talleres.
- Grupos de trabajo: los instructores pueden asignar el material de curso para grupos de alumnos o alumnos individuales.

Gestión de evaluaciones:

- Pruebas en línea: evaluaciones en línea, en las cuales se determina el avance del alumno. Pueden ser auto evaluaciones o calificadas por el profesor.
- Calificaciones en línea: las evaluaciones, trabajos, talleres, pueden ser calificados conectándose al servidor y mirando todo el material que los alumnos han elaborado a partir de los contenidos del curso. Esta puntuación se va guardando en una lista, que quedará en el mismo servidor.
- Gestión de registros: el software puede mostrar cada una de las entradas hechas a la plataforma, el material visitado y consultado con fecha y hora precisa.
- Calificación automatizada: el mismo programa se encarga de dar una puntuación a la prueba dependiendo del trabajo realizado por el alumno y de la escala elegida por el profesor.
- Gestión de diferentes tipos de preguntas: selección múltiple (marca automática), preguntas de complete (marca automática), apareamiento, falso y verdadero.
- Re-direccionamiento de una clase en particular dependiendo de las respuestas a las preguntas: el profesor puede cambiar el curso de la enseñanza de un alumno en particular dependiendo de las respuestas a sus pruebas. Proporcionándole por ejemplo más material, más ejercicios, etc., para mejorar en su labor académica.
- Pruebas sincronizadas (Calificadas con una escala permanente): Son pruebas que serán resueltas por los alumnos en un tiempo determinado, según lo crea conveniente el profesor.
- Gestión de la escala y calificación de las pruebas sincronizadas en línea: se calificará de acuerdo a escalas ya predefinidas en la plataforma, se le asignan diferentes pesos a las preguntas para su calificación si así se requiere. O el profesor mismo puede definir su propia escala para calificar.
- Generación de un conjunto de preguntas al azar: la plataforma de acuerdo a una base de datos de preguntas puede elegir un número de

preguntas determinado por el profesor para que sean respondidas por el alumno.

- Vista preliminar del examen: el profesor puede mirar el examen como lo verá el alumno antes que sea respondido por el alumno. Esto para cerciorarse que las preguntas estén bien definidas y siempre haya una respuesta para cada una de ellas.
- Herramientas complementarias en la evaluación como editores gráficos, editores de ecuaciones, etc.

- ***Servicios para el alumno***

Herramientas de control de acceso:

- Autentificación: tendrá su nombre de usuario y su clave de acceso al curso.
- Cambio de clave: el alumno podrá cambiar su clave cada vez que el crea conveniente.

Herramientas de comunicación:

- *E-mail* privado: es conveniente un correo electrónico que sea solo de uso individual del alumno, en él irán temas que solo atañen a un alumno en particular.
- Sala de *chat* para el curso: hay salas de *chat* para los integrantes del curso, indispensables para que opinen sobre diferentes temas relacionados con el curso. Puede ser bajo la dirección del tutor o la interacción de los mismos alumnos.
- Pizarra: una herramienta importante que hará las veces de tablero en la que el profesor explica temas puntuales del curso, ejemplos, ejercicios, algún gráfico, ecuación, etc.
- Charlas registradas: todas las salas de *chat* tienen la capacidad de registrar todas las charlas que tengan alumnos, profesores, alumnos – profesores, etc.
- Foros de discusión: temas del curso en los que opina el alumno. De estas formas el alumno saldrá de dudas que no se atreve a preguntar en una clase presencial.
- Tablón de anuncios: el alumno puede publicar determinadas actividades pertinentes a él o al curso.

Herramientas de evaluación:

- Auto–evaluación: con la cuál el mismo alumno y profesor pueden darse cuenta de la evolución académica del alumno.
- Pruebas asíncronas: son pruebas que no están sujetas a una fecha y hora determinada. Puede responderlas cuando el alumno estime conveniente.
- Retroalimentación: se pretende que el alumno mejore en sus actividades, de acuerdo a las ayudas que se le presenta según las faltas aciertos en los procesos educativos.
- Acceso del alumno a sus propias notas: es indispensable que el alumno tenga conocimiento de sus calificaciones en las diferentes actividades o en forma acumulativa.

- Acceso a las notas del curso: puede tenerse acceso a las notas de los demás alumnos por parte del alumno si el instructor así lo desea.

Herramientas complementarias:

- Ayuda sobre el uso de la herramienta LMS: se presenta ayudas en línea/fuera de línea para el manejo de la plataforma.
- Glosario automatizado.
- Índice automatizado.
- Gestión de Marcadores: son enlaces a diferentes temas del curso. Para llegar a un tema en especial directamente.
- Ayuda multimedia: hay ayudas elaboradas en diferentes programas por ejemplo flash.
- Envío de archivos: puede hacer sus diferentes actividades en diferentes formatos, por ejemplo MS-Word, MS-Excel, PDF (*Portable Document Format*), etc., y enviar estos archivos, en lugar de trabajar directamente sobre la plataforma.
- Herramienta de búsqueda para los contenidos del curso: facilita encontrar un tema especial dentro del curso, ahorrando tiempo.
- Área de presentación del alumno: cada alumno tiene su propia página de inicio, en la cual él puede dar la información que quiera mostrar a cerca de él mismo.

- ***Servicios para el personal de soporte***

Gestión de instructores:

- Herramientas de ayuda al instructor: el instructor también tiene sus ayudas tanto en línea, como fuera de línea.
- Creación de cuentas: es el encargado de crear las cuentas a los instructores con sus privilegios respectivos.

Gestión de alumnos:

- Herramientas de ayuda al alumno: es un privilegio de ayuda, para facilitar el uso de la plataforma al alumno.
- Registro en línea: puede dar la opción de que el alumno pueda registrarse al curso o no, directamente.
- Acceso a las cuentas del alumno: tiene acceso a las cuentas de los alumnos de todo el curso.
- Creación de cuentas: es el encargado de crear las cuentas a los instructores con sus privilegios respectivos.

Gestión Web:

- Automatización de herramientas.
- Herramientas de acceso remoto: son herramientas que son necesarias en otros equipos.
- Herramientas de recuperación de la información tras un fallo: puede recuperar la información, con herramientas que solo el administrador utiliza, esto es para darle seguridad al curso.

- Interfaz del cliente/web: es el encargado de modificar la presentación de la página web de la plataforma.

Gestión de seguridad:

- Acceso de seguridad: es el encargado de dar un nombre de usuario y una clave de acceso a la plataforma, la cual será cambiada por el usuario posteriormente.
- Nivel variable de la seguridad: puede dar un nivel de seguridad. Dependiendo de los cursos impartidos.
- Registro del estado de la máquina: el único que tiene acceso al registro de la máquina, para controlar tales accesos, ver intentos de violación contra el sistema, fecha y hora exacta de los accesos, etc.

Herramientas de apoyo:

- Ayuda al administrador: tiene su ayuda en línea /fuera de línea.
- Monitorización de recursos: observa el comportamiento de los diferentes recursos del sistema, disponibilidad de almacenamiento del disco duro, antivirus, etc.
- Capacidad para exportar información: puede hacer copias de seguridad de todos los cursos.

1.2 Los Objetos de Aprendizaje y Otros Recursos de Educación en la Web

Internet se ha posicionado como la tecnología más exitosa de la historia de las comunicaciones, se perfila como el medio de comunicación más usado en el planeta. Por ello, en todos los sectores, Internet se ha convertido en un medio fundamental para comunicar y transmitir diferentes tipos de contenidos con objetivos diversos.

Dentro de los servicios que facilita Internet, la Web es la que marcó el gran impulso para su rápida expansión. La Web a través de su interfaz gráfica y de la tecnología de hipertextos ha despertado la imaginación de aquellos que no temen incursionar en nuevas aplicaciones, tanto como productor como consumidor de estas nuevas aplicaciones. El universo de posibilidades para su explotación cada vez es mayor.

En el sector educativo, la Web ha descubierto nuevos horizontes en la educación a distancia, facilitando un medio amigable y abierto que posibilita desde muchos puntos la interacción entre el alumno y los diversos modelos de educación basado en Web.

En torno a estos nuevos modelos que hacen uso extenso de la tecnología para llevar a cabo el proceso de aprendizaje, aparecen también nuevos o renovados conceptos que dan soporte y contexto a las aplicaciones educativas en Internet. En esta unidad se hará una revisión de los conceptos relacionados al tema de objetos de aprendizaje. Inicialmente se ubicará al lector en la influencia que ha tenido la Web en los nuevos modelos educativos y posteriormente se introducirá al tema de objetos de aprendizaje, así como a otros que se relacionan a éste y al E-Learning.

La Web en la educación

La educación a distancia ha tenido un proceso de evolución a la par de las nuevas tecnologías que van surgiendo. Medios como la radio, la televisión, las grabaciones sonoras, el video, el teléfono, y los diferentes servicios de Internet han hecho de la educación a distancia, en diferentes épocas, una alternativa para quienes por limitantes geográficas, ocupacionales o físicas no pueden asistir a los cursos en escuelas presenciales tradicionales.

Las TIC (Tecnologías de la Información y la Comunicación) han aportado a la educación a distancia clásica la incorporación de conceptos de formación síncrona o asíncrona. En la formación asíncrona, que es la educación a distancia clásica, el formador y el alumno no tienen coincidencia ni en tiempo ni en espacio; en la formación síncrona los formadores y los alumnos interactúan en el mismo momento aunque físicamente no estén próximos. En la figura 1, se ejemplifica gráficamente la relación de los diferentes modelos de educación, dependiendo de las variables tiempo y lugar.

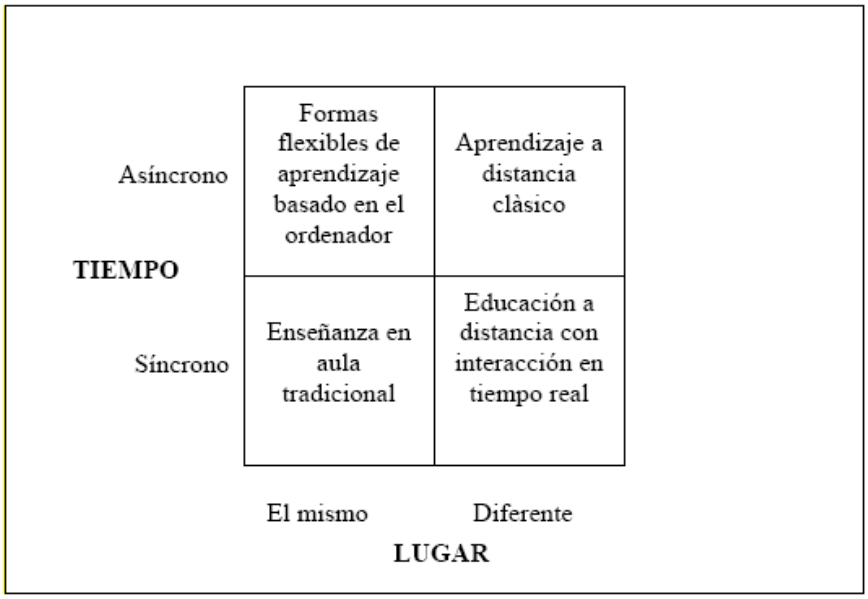


Figura 1. Modelos de educación por la relación tiempo y lugar

La modalidad más innovadora de la educación a distancia es el *E-Learning*, que hace uso de los servicios y facilidades de Internet para hacer posible el proceso de aprendizaje. Estos sistemas han adoptado a la Web como una excelente alternativa para llegar a más estudiantes, con sistemas innovadores más flexibles. La Web permite una interacción síncrona, en la que un lugar virtual y el tiempo deben conjugarse para el intercambio de actividades entre alumno y el docente. También permite la interacción asíncrona en la que la comunicación se lleva a cabo en cualquier momento desde cualquier sitio, siendo esta última la más demandada.

A partir de este juego entre tiempo y espacio y del papel que puede tener la Web dentro del proceso educativo, se han llegado a diferentes modelos que utilizan la Web como medio o recurso para la formación.

1.2.1 Plataformas para el aprendizaje en Web

Entre las herramientas más utilizadas para los cursos *E-Learning* están las plataformas LMS (*Learning Management Systems*), que facilitan el aprendizaje distribuido y colaborativo a partir de actividades e información, de forma síncrona o asíncrona, utilizando los servicios de comunicación de Internet como el correo, los foros, las videoconferencias y el chat.

Un LMS es un software basado en un servidor web que provee módulos para los procesos administrativos y de seguimiento que se requieren para un sistema de enseñanza. Los módulos administrativos permiten, por ejemplo, configurar cursos, matricular alumnos, registrar profesores, asignar cursos a un alumno, llevar reportes de progreso y calificaciones. El alumno interactúa con la plataforma a través de una interfaz web que le permite seguir las lecciones del curso, realizar las actividades programadas, comunicarse con el profesor y con otros alumnos, así como dar seguimiento a su propio progreso con datos estadísticos y calificaciones.

Las funcionalidades de las plataformas varían de un sistema a otro, pero en general todas cuentan con funcionalidades básicas como las que se han mencionado.

Entre las plataformas comerciales más comunes se encuentran Blackboard (Ref. 3) y WebCT (Ref. 2), de software libre la más reconocida es Moodle (Ref. 5).

Algunos LMS también hacen el secuenciamiento de acceso a los recursos que tendrá el alumno en el curso. Como recursos se consideran aquellos de formato digital, como son los diferentes tipos de documentos electrónicos, multimedios, animaciones, videos, simulaciones, etc., todos ellos están dentro de un nuevo concepto llamándolos OA (objetos de aprendizaje).

1.2.2 Objetos de aprendizaje

Los OA o Learning Objects son elementos para la instrucción basada en computadora, esta tecnología se fundamenta en la corriente de las ciencias de la computación conocida como orientada a objetos. La orientación a objetos se basa en la creación de componentes con la intención de que puedan ser reutilizados en múltiples aplicaciones. Esta misma idea se sigue para los OA.

No hay una única definición del concepto de OA y las definiciones son muy amplias:

El IEEE (Ref. 6) dice que los objetos de aprendizaje son "una entidad, digital o no digital, que puede ser utilizada, reutilizada y referenciada durante el aprendizaje apoyado con tecnología", Wiley (Ref. 7) "cualquier recurso digital que puede ser reutilizado para apoyar el aprendizaje", Mason, Weller & Pegler (Ref. 8) los definen como "una pieza digital de material de aprendizaje que direcciona a un tema claramente identificable o salida de aprendizaje y que tiene el potencial de ser reutilizado en diferentes contextos".

El IEEE (Ref. 6) menciona como ejemplos de objetos de aprendizaje los contenidos multimedia, el contenido instruccional, los objetivos de aprendizaje, software instruccional, personas, organizaciones o eventos referenciados durante el aprendizaje basado en tecnología.

Otros autores mencionan ilustraciones, videos, fotografías, animaciones, entre otros. Es entonces muy difícil llegar a una lista de lo que pueden ser OA, ya

que se interpreta como que es la pieza más pequeña de contenido que puede ser considerado como una unidad significativa de aprendizaje.

Aunque se menciona que un OA es "una pieza pequeña", el tamaño de un OA es variable y se conoce como granularidad. No es posible definir la cantidad de información o elementos que un OA debe contener, esto dependerá de las necesidades y capacidades del autor.

Se considera una buena práctica que los OA cubran un único objetivo de aprendizaje y para lograrlo deben mantener independencia del contexto y no requerir de otros recursos, es decir, que sean autosuficientes. Por ejemplo, un curso se divide en módulos, un módulo en lecciones y las lecciones en temas; si la unidad mínima en que puedo fraccionar ese curso en "tema" entonces el desarrollo de OA para dicho curso estará orientado a la fracción "tema" como se ejemplifica en la figura 2.

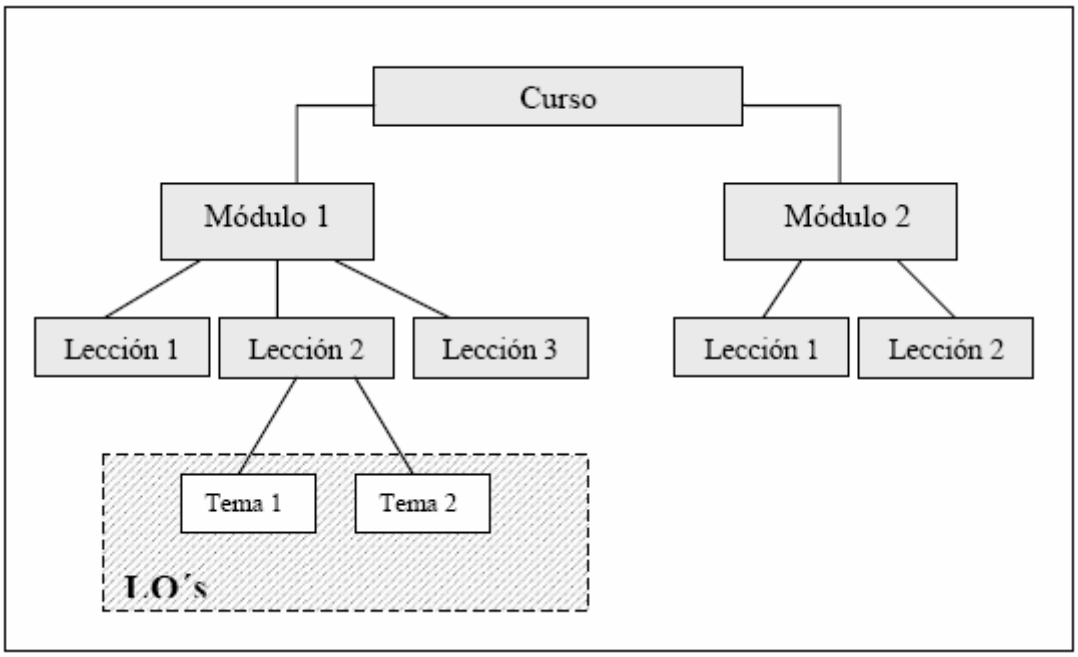


Figura 2. Taxonomía de un curso con LO

De manera general, los contenidos guardan una estructura jerárquica que va de lo general a lo particular y dependerá de los objetivos educativos la amplitud y profundidad que esta jerarquía tenga.

Actualmente, organizaciones como ADL (*Advanced Distributed Learning*) (Ref. 9) y AICC (*Aviation Industry CBT Committee*) (Ref. 10), están trabajando en el desarrollo de estándares flexibles que permitan diversidad en la construcción de cursos de diferentes taxonomías a partir de OA. El modelo que ha tenido más auge es SCORM (*Sharable Content Object Referente Model*), propuesto por ADL, que describe cómo las unidades de contenido se relacionan unas con otras, a diferentes niveles de granularidad, usando OA que cumplan con su especificación y a los cuales llama SCO (*Sharable Content Object*).

Una de las características más resaltadas en las diferentes definiciones de los OA´s es la posibilidad de su reutilización. Dada la modularidad de los OA y su independencia de otros recursos, el uso de éstos en diferentes aplicaciones es una de sus bondades, evitando duplicidad de esfuerzos para el desarrollo de contenidos.

Por ejemplo, si el OA "Tema 1" de la figura 2 fuera "comportamiento de animales mamíferos" este puede utilizarse tanto en un curso de biología como en un curso de psicología animal. El contenido del OA no cambia, únicamente se incluye en otro programa académico que le da un contexto diferente.

Para la reutilización es necesario que el OA cuente con los metadatos que le permitan ser identificado, organizado y recuperado, entre otros aspectos como la categorización y calificación pedagógica del objeto, pero lo más importante es que esos metadatos estén basados en un estándar a fin de asegurar su compatibilidad e interoperabilidad con los sistema que puedan reutilizarlos, ya sean estos plataformas de aprendizaje o repositorios que intercambien contenidos.

1.2.3 Metadatos. La clave para la reutilización

Los metadatos son un conjunto de atributos o elementos necesarios para describir un recurso. A través de los metadatos se tiene un primer acercamiento con el recurso, conociendo rápidamente sus principales características.

Algunos autores hacen la analogía de los metadatos como la envoltura de un producto, en el que se dice qué es y para qué sirve. Un ejemplo educativo puede ser una ficha bibliográfica que tiene ubicación, título, autor, editorial, año de edición, tema y número de páginas de un libro, estos descriptores son lo que se conoce como metadatos. Leyendo la ficha tenemos toda la información del recurso sin haber tenido contacto directo con el libro, esto hace más fácil y ágil la consulta en una biblioteca.

Como puede percibirse el concepto de metadato no es nuevo, pero últimamente ha tomado mayor auge dada la gran cantidad de información digital que es necesario organizar.

Para la descripción de información o recursos digitales existen diversas propuestas de conjuntos de metadatos, por ejemplo, en el ámbito de las bibliotecas digitales se encuentra la Iniciativa de Metadatos Dublin Core (DCMI) (Ref. 11), desarrollada para la descripción de un amplio universo de recursos en red ya que su aplicación es de carácter muy general.

En el ámbito de *E-Learning*, para la descripción de OA, la IEEE a través de la LTSC (Ref. 6), ha desarrollado el estándar LOM (*Learning Object Metadata*) (Ref. 12) el cual especifica la sintaxis y la semántica de los atributos necesarios para describir los objetos de aprendizaje. Este estándar está compuesto de nueve categorías de metadatos, que agrupan elementos con los que se ha pretendido una descripción completa de los recursos educativos. En la tabla 1 se describen las nueve categorías, en picoparéntesis <> se expresa el nombre en inglés tal como se reconoce en el estándar y también se listan los elementos que componen cada categoría.

Cabe mencionar que los metadatos no son parte del recurso mismo, así que en general se suele separar el archivo de metadatos del archivo del OA.

Con el uso de estándares de metadatos estructurados se busca la interoperabilidad entre todos los recursos de información involucrados en un proceso de enseñanza. Así, en un futuro el intercambio de información y de contenidos entre las plataformas y entre los repositorios será transparente para el usuario, facilitando la creación de curriculas y el desarrollo de sistemas con recursos globales para aplicaciones particulares.

Categorías	Descripción
1. General <general>	Información general que describe el objeto de aprendizaje como un todo. Elementos: Identificador, Título, Entrada de catálogo, Lengua, Descripción, Descriptor, Cobertura, Estructura, Nivel de agregación
2. Ciclo de vida <lifecycle>	Características relacionadas con la historia y el estado presente del objeto de aprendizaje y de aquellos que han afectado a éste objeto durante su evolución. Contiene 6 subelementos. Elementos: Versión, Estatus, Otros colaboradores
3.Meta-metadatos <metametadata>	Agrupa información sobre los mismos metadatos, no sobre el objeto de aprendizaje que se está describiendo. Elementos: Identificador, Entrada de catálogo, Otros colaboradores, Esquema de metadatos, Idioma
4.Técnico <technical>	Agrupa los requerimientos y características técnicas del objeto de aprendizaje. Elementos: Formato, Tamaño, Ubicación, Requisitos, Comentarios sobre la instalación, Otros requisitos para plataformas, Duración
5. Educativo <educational>	Condiciones del uso educativo del recurso. Elementos: Tipo de interactividad, Tipo de recurso de aprendizaje, Nivel de interactividad, Densidad semántica, Usuario principal, Contexto [Nivel educativo], Edad, Dificultad, Tiempo previsto de aprendizaje, Descripción, Idioma
6. Derechos <rights>	Condiciones de uso para la explotación del recurso. Elementos: Coste, Copyright y otras restricciones, Descripción
7. Relación <relation>	Define la relación del recurso descrito con otros objetos de aprendizaje. Elementos: Tipo, Recurso
8. Observaciones <annotation>	Comentarios sobre el uso educativo del objeto de aprendizaje. Elementos: Persona, Fecha, Descripción
9. Clasificación <classification>	Descripción temática del recurso en algún sistema de clasificación. Elementos: Finalidad, Nivel táxon (taxonómico), Descripción, Descriptor

Tabla 1. Grupos y elementos del estándar LOM.

1.2.4 Bibliotecas digitales y repositorios de objetos de aprendizaje

Al agrupar o concentrar recursos digitales de información se obtienen colecciones de recursos u objetos digitales, que se organizan con un sistema descriptivo a través de metadatos (catalogación) y se les asocian algunas facilidades para la búsqueda y uso de la información (servicios), conociéndose como bibliotecas digitales.

Las bibliotecas digitales basan el contenido de sus repositorios en objetos de información, refiriéndose a todo tipo de objeto que provea información, como imágenes, videos, animaciones y multimedios. Se podría pensar que las bibliotecas digitales son los repositorios naturales para los OA, sin embargo, la comunidad de estándares del *E-Learning* ha creado sistemas descriptivos mucho más especializados y específicos que los que hasta ahora han utilizado las bibliotecas digitales. Estos repositorios apuntan a la utilización del sistema de metadatos específicamente orientados al campo educativo, como los propuestos por SCORM o IMS que se basan en el estándar LOM y que además de los metadatos ofrecen otras funciones para la interoperabilidad e intercambio de OA entre sistemas.

Entonces, se puede hacer la diferenciación diciendo que las bibliotecas digitales utilizan sistemas de clasificación genéricos y los Repositorios de OA (LOR) son un tipo de bibliotecas digitales en los que se utilizan los estándares de metadatos que han desarrollado los organismos encargados de la estandarización del E-Learning.

Se identifican dos tipos de LOR:

- los que contienen objetos de aprendizaje y sus metadatos, en éstos los objetos y sus descriptores se encuentran dentro de un mismo sistema e incluso dentro de un mismo servidor
- los que contienen sólo los metadatos, en este caso el repositorio contiene sólo los descriptores y se accede al objeto a través de una referencia a su ubicación física que se encuentra en otro sistema o repositorio de objetos.

Y como es común también existen aquellos en los que hay una combinación de los dos tipos anteriores.

Usualmente los LOR operan de forma independiente, aunque es común que los LMS tengan asociado un repositorio que en la mayoría de los casos es sólo para su uso dentro de la misma plataforma.

Como una extensión de los repositorios han aparecido un nuevo género de sistemas de administración dentro del E-Learning y estos son los *Learning Content Managemente Systems* (LCMS) que además de almacenar y organizar los OA permiten el ensamblamiento de éstos para estructurar cursos. Los LCMS son sistemas mucho más complejos que los repositorios ya que incluyen herramientas de autor y funciones de etiquetamiento y ensamblaje de contenidos.

1.3 SCORM (Sharable Content Object Reference Model)

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.

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 (Ref. 14), y el mecanismo de intercambio de información mediante una API de AICC (Ref. 15), que se verá en este mismo capítulo), mejorándolas para crear su propia iniciativa: SCORM, *Sharable Courseware Object Reference Model* (Modelo de Referencia para Objetos de Cursos Compartidos) (SCORM 2001. Ref. 13), 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 *Sharable 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 (Ref. 6). En la figura 3 mostramos un gráfico de la evolución de SCORM desde su fecha de creación.

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).

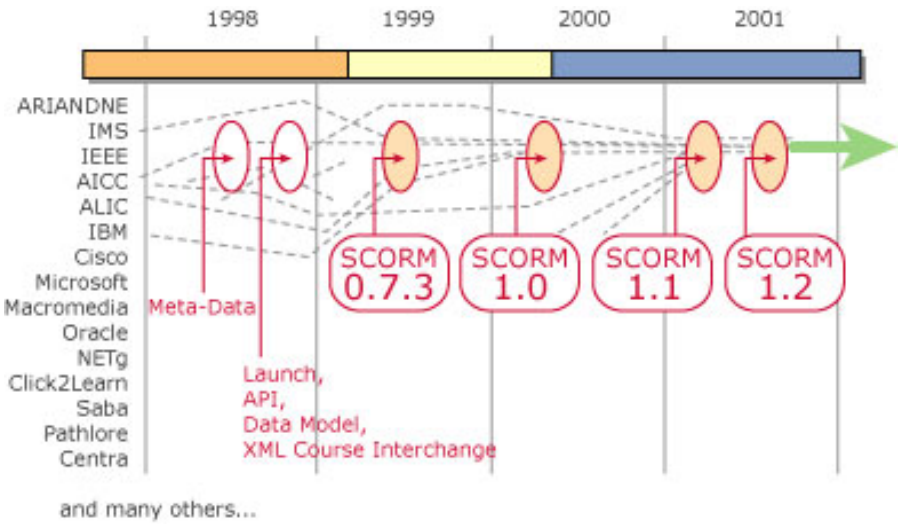


Figura 3. Gráfico de convergencia de ADL/SCORM. (ADL 2001a)

1.3.1 Conceptos Básicos

1.3.1.1 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 como se visualiza en la figura 4.

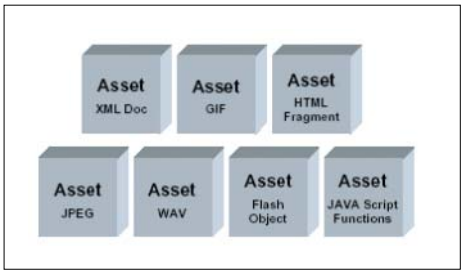


Figura 4. Composición de un SCO

1.3.1.2 Assets o Sharable Resources (Contenido compartido)

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. Un ejemplo de Asset se visualiza en la figura 5.

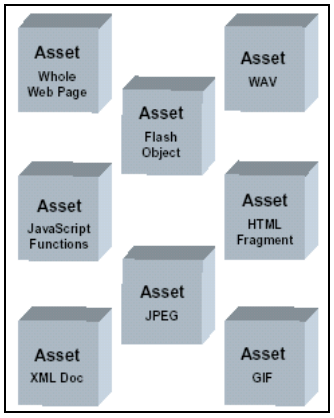


Figura 5. Diferentes ejemplos de Assets (ADL 2001c)

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.

1.3.1.3 Metadatos

Otra de las partes importantes dentro del desarrollo de elementos que cumplan con las especificaciones de SCORM son los metadatos, a continuación describiremos algunas definiciones y aportes realizados respecto de este tema por las organizaciones vinculadas al desarrollo de especificaciones que ayuden a la estandarización de los sistemas *E-Learning* y todo lo referente a ellos.

- **Metadata:** es información sobre un objeto, ya sea físico o digital. Como el número de objetos crece exponencialmente y nuestras

necesidades de expandir el aprendizaje es igualmente dramática, la falta de información o metadata sobre objetos da lugar a un esfuerzo crítico y fundamental en nuestra habilidad para descubrir, manejar y usar objetos.

- ***Learning Objects Metadata* (LOM):** la especificación más reconocida del trabajo del IEEE LTSC (Ref. 6) es la especificación de los Metadatos de los Objetos de Aprendizaje o *Learning Object Metadata* que define elementos para describir los recursos de aprendizaje. IMS y ADL utilizan los elementos y la estructura del LOM en sus respectivas especificaciones.
  La especificación del LOM entrega una guía sobre cómo los contenidos deben ser identificados o "etiquetados", y sobre cómo se debe organizar la información de los alumnos de manera que puedan intercambiarse entre los distintos servicios involucrados en un sistema de gestión de aprendizaje (LMS). La especificación para metadata del IMS consta de tres documentos: IMS Learning Resource Meta-data Information Model, IMS Learning Resource XML Binding Specification, IMS Learning Resource Best Practices and Implementation Guide (Ref. 14).

1.3.1.4 API de SCORM

La API JavaScript de SCORM es una pieza de código que un ambiente virtual de educación (*Virtual Learning Environment*, VLE) provee al browser de un estudiante cuando éste requiere contenido *Learning* de un objeto SCORM. El código captura acciones específicas del estudiante vía instrucciones en el contenido SCORM y las pasa por el VLE.

Las acciones incluyen cosas tales como: cuán lejos ha llegado un estudiante dentro de una unidad, cuánto tiempo le llevó trabajar con la materia, etc.

1.3.2 LMS (Learning Management System)

Se trata de un software para servidores de internet/intranet que se ocupa de:

- Gestionar los usuarios: inscripción, control de sus aprendizajes e historial, generación de informes, etc.
- Gestionar y lanzar los cursos, realizando un registro de la actividad del usuario: tanto los resultados de los tests y evaluaciones que realice, como los tiempos y accesos al material formativo.
- Gestionar los servicios de comunicación que son el apoyo al material online, foros de discusión, charlas, video conferencias; programarlos y ofrecerlos conforme sean necesarios.

En el mercado existen tanto LMS como *Courseware* de muchos fabricantes distintos. Por ello se hace necesaria una normativa que compatibilice los distintos sistemas y cursos a fin de lograr dos objetivos:

- Que un curso de cualquier fabricante pueda ser cargado en cualquier LMS de otro fabricante.
- Que los resultados de las actividades de los usuarios en el curso puedan ser registrados por el LMS.

1.3.3 Modelo para el tratamiento de contenidos

Dentro de este modelo, SCORM trata tanto los metadatos como la forma de estructurar los contenidos educativos.

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.

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 dentro de 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 item, 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 descripto es mostrado en la figura 7.

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

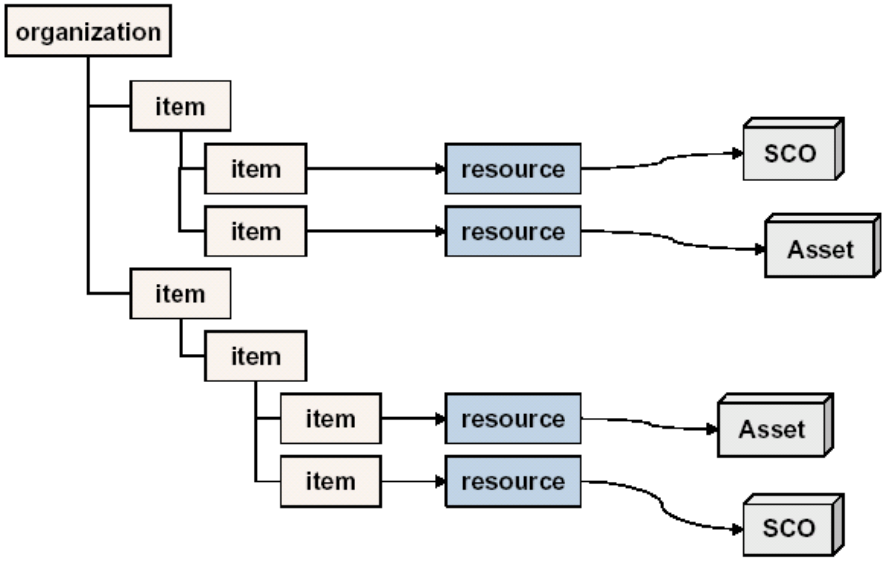


Figura 7. Integración de los elementos de SCORM dentro del manifiesto de IMS. (ADL 2001c)

1.3.4 Entorno de ejecución SCORM

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 intercambio 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.

1.3.5 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. El esquema de la figura 9 muestra el modelo SCORM.

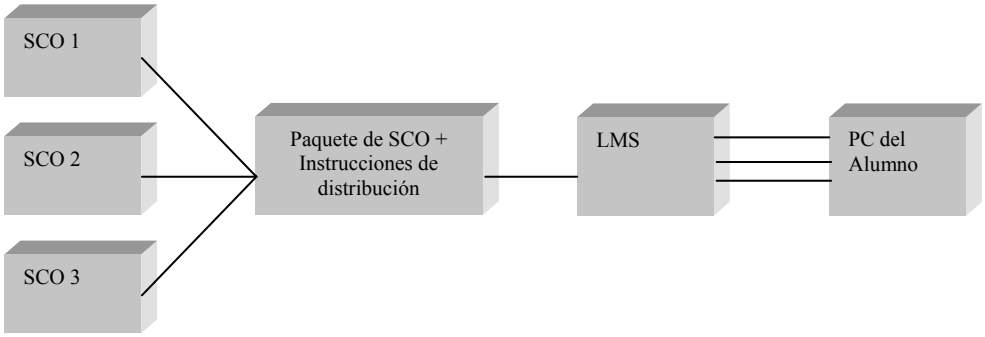


Figura 9 Modelo SCORM. (Eduworks 2002)

1.3.6 SCORM Content Aggregation (Curso, asociación de clases)

La organización de objetos *Learning* de SCORM en un paquete es descripta en una estructura de árbol jerárquica, como una estructura de un curso o jerarquía de contenidos. SCORM no especifica una profundidad particular para este árbol; tampoco especifica una nomenclatura particular para el nombre de los niveles del árbol, como "curso, lección, tópico" o "unidad, módulo, lección".

El mecanismo para enlazar los contenidos puestos en el Content Aggregation Meta-data es el Content Package. Las especificaciones de empaquetamiento de IMS describen un archivo XML dividido en tres partes principales:

- Meta-data: información sobre el curso, utilizando elementos de la IEEE LOM (*Learning Object Metadata*).
- Tabla de contenidos: capítulos, lecciones o cualquier otro punto dentro de los recursos utilizados por el curso.
- Recursos: una lista completa de empaquetamiento de todos los archivos o URLs necesitados por el curso.

El archivo XML resultante es llamado "**manifest**". A través de la extensibilidad de XML y los mecanismos descriptos por el IMS se pueden crear documentos de control que proveen información adicional, detalles y definiciones a las especificaciones iniciales de empaquetamiento.

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>

Sección <organizations>

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.

1.3.7 Comunicación del SCO y el LMS mediante el API

SCORM utiliza un API para la comunicación entre el SCO y el LMS.

SCORM utiliza un entorno de ejecución que incluye:

- protocolo específico para la ejecución de contenidos Web.
- un API entre el contenido y el LMS para la comunicación.
- 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.

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 8 ofrecemos un esquema de lo que sería esta comunicación.

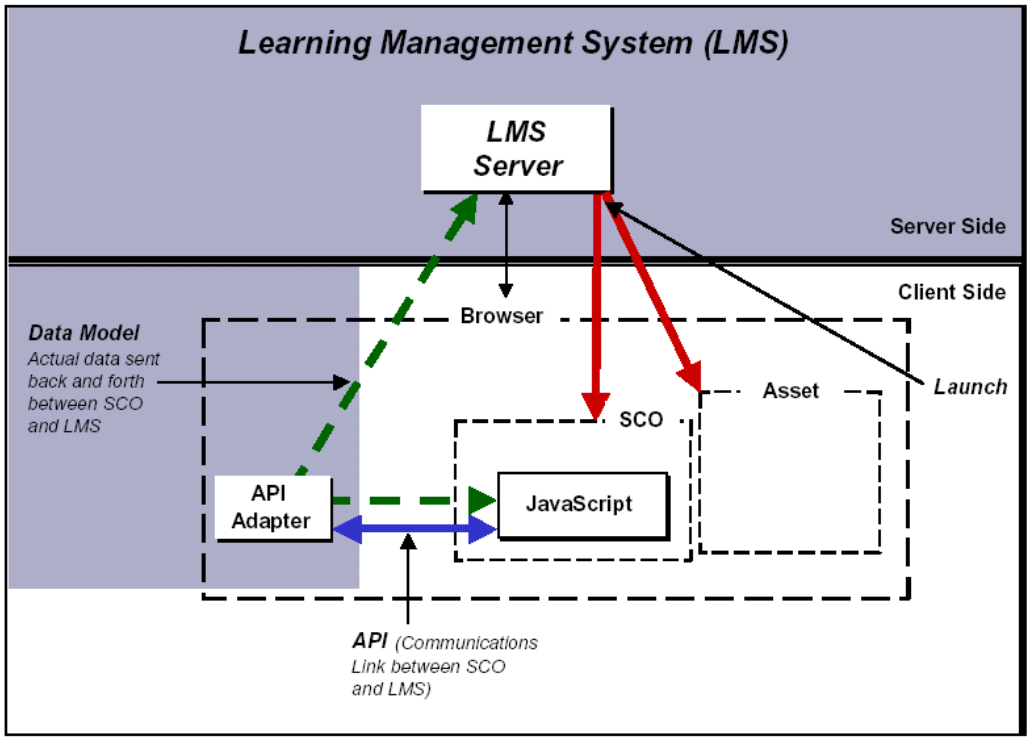


Figura 8. Esquema de comunicación entre los SCO´s y el LMS mediante API. (ADL 2001b)

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.

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)
- El resto de las funciones de la API se utilizan para la transferencia de datos desde y hacia el LMS. Básicamente las tres mas usadas son:

LMSGetValue(data model element), LMSSetValue(data model element, value) y LMSCommit("").

Estado de la ejecución

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 una entrada dada. 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.

1. **No inicializado:** 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()

2. **Inicializado:** Describe el estado en que el SCO está situado entre la ejecución de LMSInitialize ("") y LMSFinish ("").
3. **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

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. Se ha incluido dicho modelo en el Anexo A correspondiente a este trabajo de grado para su consulta.

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:
    1. _versión: palabra clave usada para determinar la versión del modelo de datos que admite el LMS.
    2. _children: palabra clave usada para determinar qué elementos del modelo de datos son admitidos por el LMS.
    3. _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 (Ref. 15) para compatibilidad entre 2 computadoras.

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 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"

Para poder tener una visión más clara de todo lo explicado hasta el momento sobre el entorno de ejecución en SCORM hemos incluido un ejemplo de la implementación de un SCO y la construcción del archivo "imsmanifest.xml" en el Anexo A correspondiente a este trabajo de grado para su consulta.

Conclusiones

El campo de Learning posibilita conseguir, entre otras cosas, aprendizajes personalizados, automatización de tareas, y el aprovechamiento de recursos educativos procedentes de diversas fuentes y en diversos formatos. Ha impuesto nuevos conceptos que buscan la reutilización, permanencia, durabilidad y compatibilidad de recursos para el desarrollo de cursos y programas de formación en línea, facilitando a través de la Web el acceso a la educación

Los objetos de aprendizaje permiten que gran parte de los sistemas de educación basada en Web y de repositorios estén intercomunicados, con lo cual se tendrá acceso a más contenido y mejor sistema educativo con alta tecnología.

Es importante no perder de vista que para la interoperabilidad entre sistemas, los desarrolladores y proveedores deben generar productos que cumplan con los estándares que se imponen en el sector.

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

En cuanto al modelo SCORM una de sus principales características es que el contenido puede comunicar información sobre el alumno a cualquier sistema LMS utilizando un método estándar basado en JavaScript a través de API Adapter. 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 OA y sus preferencias de visualización.

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

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

Unidad 2: Herramientas para el manejo de cursos estandarizados.

Introducción

En esta unidad se van a analizar las características y funcionalidades de un LMS y de un editor de contenidos SCORM. Además se realizará el estudio y la comparación de diferentes editores HTML *Open Source* que son necesarios para la realización de este trabajo de grado.

Se ha tomado como ejemplo de LMS a Moodle y como ejemplo de editor de paquetes SCORM a ReloadEditor. Esto surge a partir del Trabajo de Grado "Usando XML para metaanotar recursos educativos" presentado por la alumna Andrea Deluchi, en el cual se comprobó que ambas herramientas son de gran popularidad, representantes de plataformas gratuitas y fáciles de usar. Nosotros tomaremos estas dos herramientas como complemento del editor HTML que adaptaremos a SCORM.

Además se realiza un análisis de los editores Open Source, y se justificara la elección del editor propuesto para el desarrollo del trabajo.

2.1 MOODLE

Moodle es una herramienta para la creación de cursos y sitios Web basados en Internet. Moodle se distribuye como software libre (bajo la Licencia pública GNU) y como principal característica a tener en cuenta es que se encuentra en constante evolución. Es una herramienta de código abierto, esto significa que Moodle tiene derechos de autor (copyright), pero puede ser usado y modificado siempre que se mantenga el código fuente abierto para todos, no se modifique o elimine la licencia original, y se aplique esta misma licencia a cualquier trabajo derivado de él (Ref. 5).

Moodle es un LMS, una aplicación diseñada para ayudar a los educadores a crear y administrar contenidos educativos reutilizables. Nos permite importar contenidos empaquetados, como los creados con Reload Editor.

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. (Ref. 20)

Moodle es una aplicación Web a la que se accede por medio de un navegador Web (Mozilla Firefox, Microsoft Internet Explorer, etc). Esto significa que para utilizar Moodle se necesita una computadora con un navegador Web instalado y con conexión a Internet. Por supuesto también se necesita conocer la dirección Web (URL) del servidor donde Moodle se encuentre alojado. Para poder acceder a la aplicación se debe estar registrado como usuario del mismo.

2.1.1 Características generales

- 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 navegación de tecnología sencilla, ligera, eficiente, y compatible.
- Soporta SCORM en su totalidad.
- La mayoría de las áreas de introducción de texto (materiales, mensajes de los foros, entradas de los diarios, etc.) pueden ser editadas en forma sencilla, como cualquier editor de texto de Windows.
- En la administración de cursos el profesor tiene control total sobre todas las opciones de un curso.
- En la página principal del curso se pueden presentar los cambios ocurridos desde la última vez que el usuario entró en el curso, lo que ayuda a crear una sensación de comunidad.
- Registro y seguimiento completo de los accesos del usuario. Se dispone de informes de actividad de cada alumno, con gráficos y detalles sobre su paso por cada módulo (último acceso, número de veces que lo ha leído) así como también de una detallada "historia" de la participación de cada alumno, incluyendo mensajes enviados, entradas en el diario, etc. en una sola página.
- Puede funcionar en cualquier computadora en el que pueda correr PHP, y soporta varios tipos de bases de datos (en especial MySQL).
- Estos paquetes pueden editarse usando un editor integrado. Las páginas se almacenan en la base de datos, no como archivo.
- Soporta un rango de mecanismos de autenticación a través de módulos de autenticación, que permiten una integración sencilla con los sistemas existentes.
- Soporta método estándar de alta por correo electrónico, los alumnos pueden crear sus propias cuentas de acceso. La dirección de correo electrónico se verifica mediante confirmación.
- Método LDAP (*Lightweight Directory Access Protocol*): las cuentas de acceso pueden verificarse en un servidor LDAP. El administrador puede especificar qué campos usar.
- IMAP (*Internet Message Access Protocol*), POP3 (*Post Office Protocol 3*), NNTP (*Network News Transfer Protocol*): las cuentas de acceso se verifican contra un servidor de correo o de noticias (news). Soporta los certificados SSL (*Secure Socker Layer*) y TLS (*Transport Layer Security lookups*).
- Base de datos externa: cualquier base de datos que contenga al menos dos campos puede usarse como fuente externa de autenticación.
- Cada persona necesita sólo una cuenta para todo el servidor. Por otra parte, cada cuenta puede tener diferentes tipos de acceso.
- Una cuenta de administrador controla la creación de cursos y determina los profesores, asignando usuarios a los cursos.
- Seguridad: los profesores pueden añadir una "clave de acceso" para sus cursos, con el fin de impedir el acceso de quienes no sean sus alumnos.

Pueden transmitir esta clave personalmente o a través del correo electrónico personal, etc.

- Los profesores pueden dar de baja a los alumnos manualmente si lo desean, aunque también existe una forma automática de dar de baja a los alumnos que permanezcan inactivos durante un determinado período de tiempo (establecido por el administrador).
- Se permite a los alumnos a crear un perfil en línea de sí mismos, incluyendo fotos, descripción, etc. De ser necesario, pueden esconderse las direcciones de correo electrónico.
- Cada usuario puede especificar su propia zona horaria, y todas las fechas marcadas en Moodle se traducirán a esa zona horaria (las fechas de escritura de mensajes, de entrega de tareas, etc.).
- Cada usuario puede elegir el idioma que se usará en la interfaz de Moodle (inglés, francés, alemán, español, portugués, etc.).

2.1.2 Utilizando Moodle para cargar un paquete SCORM

Moodle permite incorporar materiales empaquetados según el Standard SCORM/AICC. Un paquete SCORM es un bloque de material Web que puede incluir páginas Web, gráficos, programas Javascript, y cualquier contenido que pueda visualizarse mediante un navegador Web, que es empaquetado siguiendo el estándar SCORM para objetos de aprendizaje.

El módulo SCORM de Moodle permite cargar fácilmente cualquier paquete SCORM estándar y convertirlo en parte de un curso.

2.1.3 Añadir un paquete SCORM al curso

Para que los contenidos SCORM formen parte de un curso se deberán seguir los siguientes pasos:

1. Activar el *"Modo edición".*
2. Seleccionar *"SCORM"* en el desplegable *"Agregar actividad".*
3. Completar los campos del formulario rellenando los siguientes campos:
    - Nombre: nombre que identifica el contenido del paquete SCORM.
    - Informe: resumen de los contenidos del paquete.
    - Paquete de curso: paquete SCORM previamente subido al sistema. Moodle permite subir el paquete en este momento mediante el botón "Elegir o actualizar el paquete SCORM…".
    - Método de calificación. Podemos elegir entre cuatro opciones diferentes:
        a) Situación de Sco's: muestra el número de páginas completadas (visitadas) por el alumno.
        b) Más alto: muestra la puntuación más alta obtenida por el usuario en los Sco's completadas. Este método y los siguientes implican la calificación de cada una de los Sco's.

c) Promedio: devuelve un promedio de las calificaciones obtenidas en los Sco's del paquete.

d) Suma: muestra la suma de todas las puntuaciones.

- Calificación máxima: en caso de que los Sco's de un paquete sean calificables (tres últimos casos del punto anterior), fija la calificación máxima para cada una de ellas.
- Continuación automática: al activar esta opción cuando un Sco llama al método "cerrar comunicación", el siguiente Sco disponible se abrirá automáticamente. Si la opción se dejara en "No", los usuarios deberán pulsar el botón "Continuar" para seguir.
- Habilitar visión previa: ofrece al alumno la posibilidad de previsualizar los contenidos de los Sco's. El estudiante puede elegir si quiere previsualizar con antelación la actividad (modo explorar) o la quiere abrir en modo de revisión (modo evaluable). Cuando un Sco es completado en el modo de vista previa (explorar), éste se crea con un icono de exploración.
- Ancho y altura: dimensionan la ventana o marco en el que se mostrarán las Sco's.

4. Guardar los cambios pulsando el botón "Guardar cambios".

El proceso de añadir un paquete SCORM al curso se puede visualizar en la figura 10, donde se muestran los campos a completar en la creación.

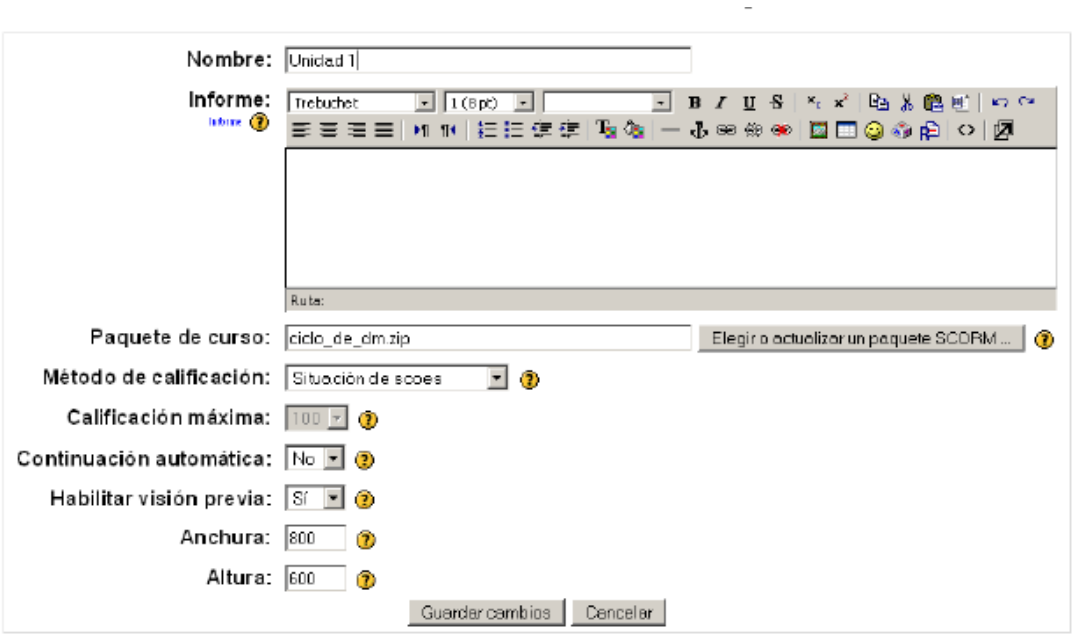


Figura 10. Configuración de un SCORM

2.2 Reload Editor & Reload Player

El proyecto *Reusable eLearning Object Authoring & Delivery* (RELOAD) tiene el propósito de desarrollar herramientas que faciliten el uso de las especificaciones de interoperabilidad de las tecnologías educativas emergentes, como las propuestas por *Advanced Distributed Learning* (ADL) y por *Instructional Management Systems / Global Learning Consortium* (IMS/GLC) (Ref. 4).

RELOAD ha desarrollado varias herramientas, entre ellas Reload Editor y Reload Scorm Player, ambas gratuitas y de código abierto.

Reload Editor es una herramienta para crear y editar paquetes e insertar metadatos conforme a las especificaciones de ADL e IMS. Con el Reload Editor podemos ejecutar y ver nuestros paquetes en un navegador Web. Ahora bien, como sabemos, el contenido SCORM puede ser más complejo, y permite la comunicación con un LMS mediante un entorno de ejecución basado en una serie de APIs. Reload Editor no permite editar y añadir a los paquetes de contenido las características que le permitan comunicarse con la plataforma (LMS), para ello el usuario necesitara herramientas de autor y de edición Web (MSFrnt-Page, Authorware, etct). Si disponemos de tal contenido, para probarlo tendremos que cargarlo en un LMS que soporte tales características. Reload Player permite solventar esa situación.

Reload Player no es un *Learning Management System* (LMS), sino un "player", es decir un ambiente de ejecución que nos permite probar paquetes de contenido.

Como hemos mencionado anteriormente RELOAD Editor facilita la creación del empaquetamiento de contenido basado en SCORM.

Reload Editor al crear los paquetes de contenidos genera automáticamente:

- 4 archivos en el caso de que sea un paquete de contenidos: imsmanisfest.xml, imscp_v1p1.xsd, imsmd_v1p1.xsd y ims_xml.xsd
- 5 archivos en el caso de que sea un paquete de contenidos SCORM: imsmanifest.xml, adlcp_rootv1p2.xsd, imscp_rootv2p1p2.xsd, imsmd_rootv1p2p1.xsd y ims_xml.xsd

El primer archivo describe los contenidos del paquete y puede incluir una descripción opcional de la estructura del contenido. El nombre "imsmanifest.xml" es obligatorio y el archivo debe aparecer en la raíz de cualquier paquete de contenidos válido. Los tres o cuatro archivos restantes son copias locales de los documentos XML de esquemas.

Reload Editor permite asignar los recursos al paquete y sus respectivos metadatos, de forma muy amigable. No es necesario que la persona que realice esta labor (diseñador instruccional, experto en el tema, programador, etc.) conozca en profundidad el lenguaje XML y sus tecnologías asociadas, salvo que se requiera hacer modificaciones que impliquen ingresar al código. Sin embargo se debe tener conocimiento del estándar que se esta aplicando.

Otro de los elementos importantes que debe tener el archivo imsmanifest.xml son los metadatos de los recursos que conforman el contenido. La herramienta Reload Editor tiene un módulo que permite ingresar esta información basado en el estándar LOM de la IEEE (Ref. 6), utilizado por el modelo SCORM en su especificación (Ref. 13).

Una vez creado el paquete de contenidos, Reload Editor permite exportar dicho paquete generando un archivo .zip listo para ser importado por algún LMS que soporte contenido SCORM, asegurando interoperabilidad entre plataformas.

El paquete SCORM básicamente queda conformado por el archivo imsmanifest.xml y todos los recursos asignados al curso en cuestión.

2.2.1 Entorno de trabajo de Reload Editor

El espacio de trabajo consiste básicamente en tres paneles:

- El panel de recursos (a la izquierda),
- El panel del manifest (derecha) y
- El panel de atributos (abajo).

El panel del manifest es el más importante y representará la estructura del paquete de contenidos, el cual contiene un Manifiesto, que a su vez contiene Organizaciones y Recursos.

El panel de atributos contiene información relativa al elemento que tengamos seleccionado. Incluye una tabla de Atributo-Valor y una sección de información sensible al contacto, acerca del elemento que se encuentre seleccionado.

La figura 11 muestra una vista del espacio de trabajo de Reload Editor.

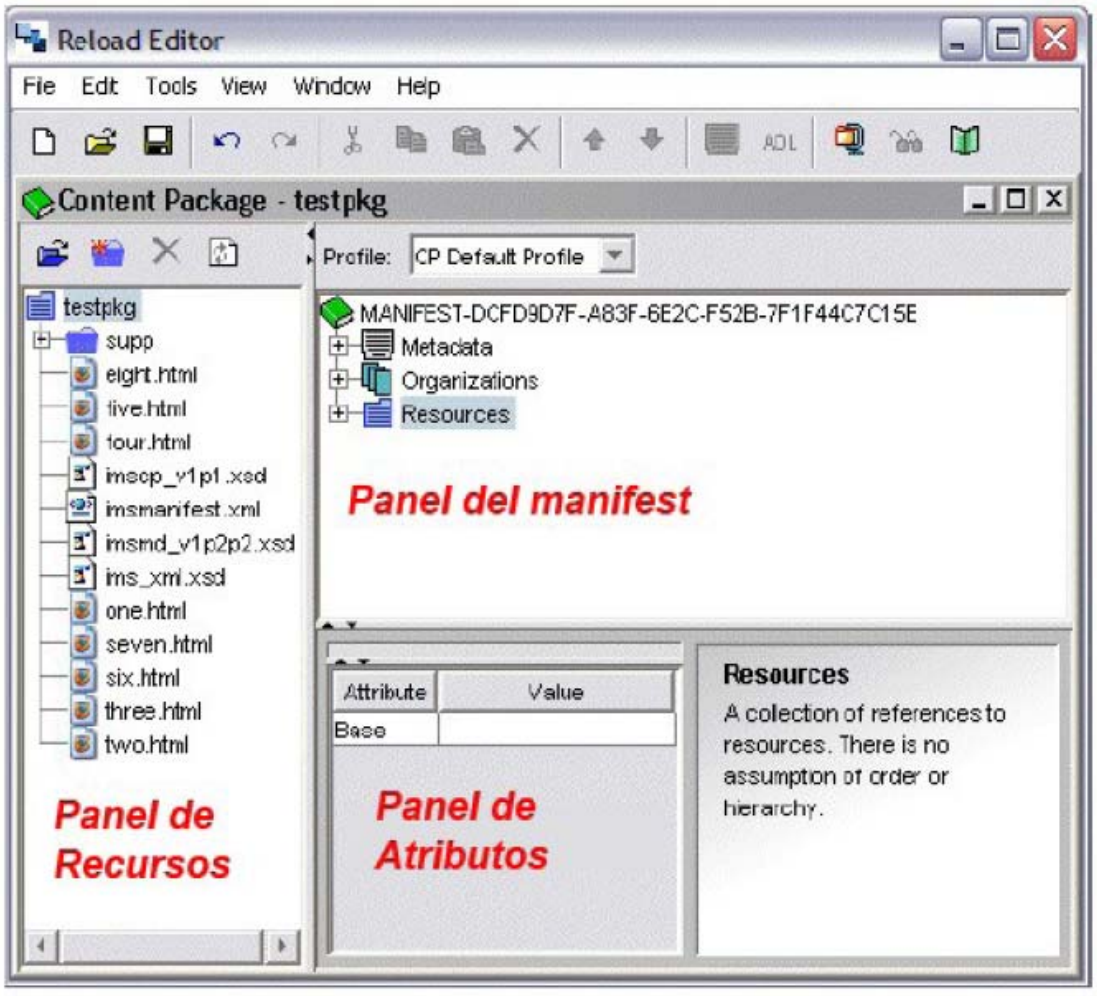


Figura 11. Espacio de trabajo de Reload Editor

Cada una de las ventanas representa un paquete. Podemos abrir tantas ventanas como queramos.

2.2.2 Empaquetamiento de contenidos con Reload Editor

Reload Editor permite el empaquetamiento de contenidos para dos tipos de paquetes: paquete de contenidos y paquete SCORM.

Crear Paquete de Contenidos

Por default, el editor inicia sin ningún archivo abierto. Para crear un nuevo paquete de contenido se debe seleccionar en el menú la opción: Archivo -> Nuevo -> IMS Content Package. Se abrirá un cuadro de diálogo solicitando seleccionar una carpeta para crear el paquete, luego de seleccionarla se abrirá una nueva ventana con el nombre de la carpeta seleccionada como título, tal como se muestra en la figura 12.

Reload Editor al crear el paquete, crea automáticamente 4 archivos: el archivo manifiesto y los esquemas.

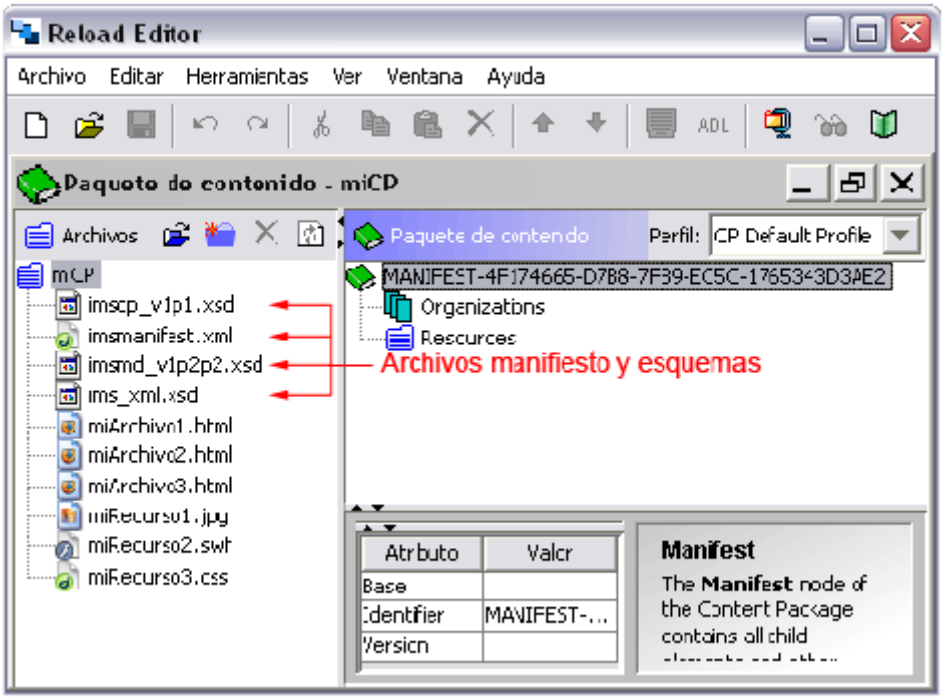


Figura 12. Paquete de contenido

Añadir los recursos

Hasta el momento se han dados los pasos para crear un paquete, pero éste aún se encuentra vacío. No tiene organizaciones, ítems, ni recursos referenciados. Por default, Reload Editor siempre inicia los paquetes vacíos.

Para añadir los recursos, sólo es necesario seleccionarlos del panel de recursos y arrastrarlos hacia el panel manifiesto, sobre la carpeta recursos.

Añadir una organización

Un paquete consiste de una o más organizaciones de contenido. Por defalult, el paquete creado por Reload no tiene organizaciones.

Para agregar una organización se debe seleccionar el icono Organizations en el panel manifiesto y presionar el botón derecho del Mouse sobre él. En el menú contextual seleccionar "Añadir Organization".

Seleccionar la nueva organización y, en el panel de atributos, asignarle el nombre "Principal" en el cuadro blanco.

Reload Editor permite asignar una o más organizaciones a un mismo paquete. Cuando hay varias organizaciones en el mismo paquete, una de éstas debe estar especificada como la organización por default. Si ninguna organización está especificada como default, entonces la primera organización es asumida como la organización por default.

Para especificar una organización como default seleccionar el ícono Organizations, y en el panel de atributos, en la opción "Default Organization" seleccionar la organización que será por default.

Añadir un Item

Finalmente, podemos añadir algún contenido a nuestro paquete. Básicamente, creamos la estructura de nuestro paquete al añadir items a una organización.

Para añadir un item:

- Selecciona la organización Principal (creada en el paso anterior) en el panel manifiesto y presionar el boton derecho del mouse. En el menú contextual seleccionar "Añadir Item".
- Seleccionar el nuevo Item y, en el panel de atributos, asígnale un nombre, por ejemplo "Item 1" en el cuadro blanco.

Reload Editor permite crear uno o más ítems por cada organización del paquete.

Al crear el ítem, en el panel de atributos, se puede indicar que dicho ítem haga referencia a algún elemento (recurso). Por ejemplo se podría decir que el "Item 1" referencia al "miArchivo3.html"

Pasos para exportar como paquete de contenidos

En primer lugar se debe guardar el paquete generado. Luego debemos exportarlo como un archivo comprimido zip. Para esto seleccionar de la barra de herramientas "Crear Paquete" o desde el menú Archivo seleccionar la opción "Crear Paquete de contenido".

Se abrirá un cuadro de diálogo, crear una nueva carpeta y dentro de ésta guarda tu archivo con el nombre a elección (incluyendo la extensión .zip, pero sin las comillas).

El paquete ya esta exportado y listo para ser importado por algún LMS.

2.2.3 Empaquetamiento de contenidos ADL SCORM con Reload Editor

Crear Paquete SCORM

Para crear un nuevo paquete SCORM:

- En el menú Archivo - Nuevo - Paquete SCORM 1.2.
- Se abrirá un cuadro de diálogo solicitando seleccionar una carpeta para crear el paquete SCORM. Como se muestra en la figura 13.

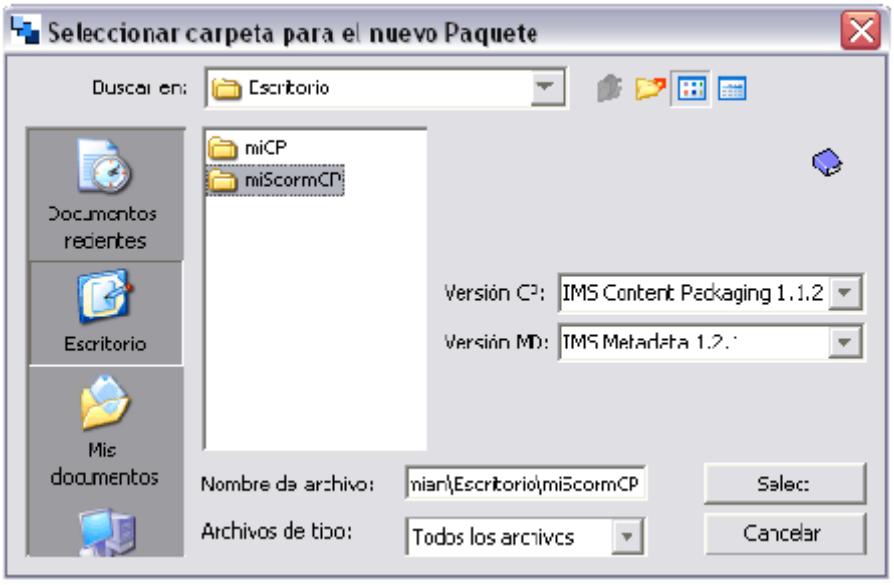


Figura 13. Nuevo paquete SCORM

- Luego de seleccionar se abrirá una nueva ventana con el nombre de la carpeta seleccionada como título. Lo descripto se puede observar en al figura 14.

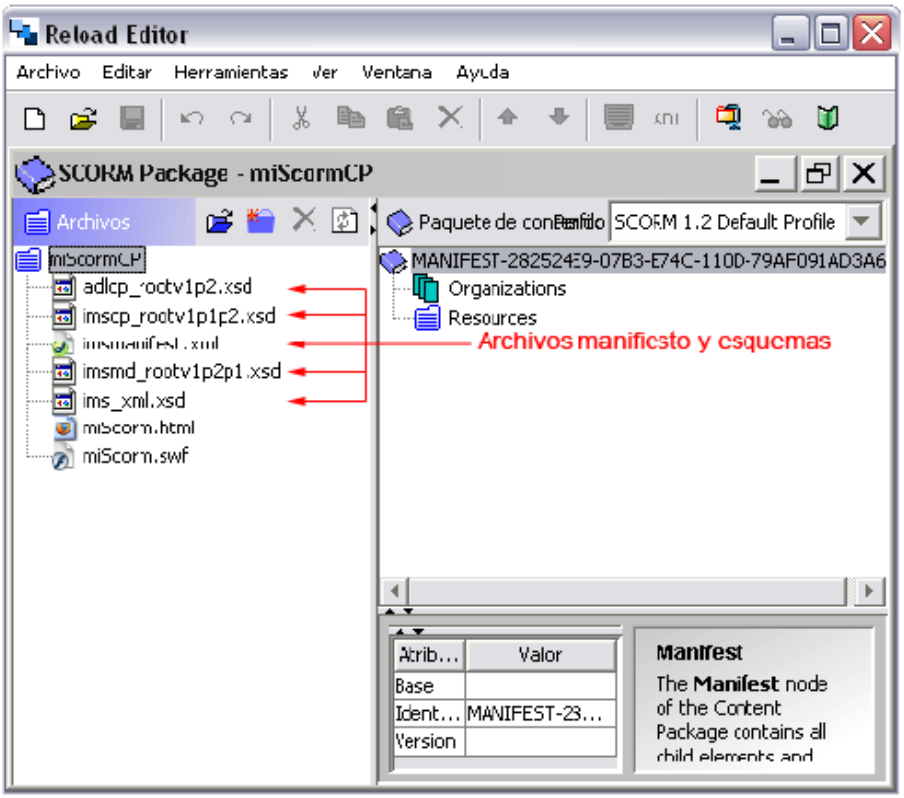


Figura 14. Paquete SCORM

Como se puede observar en la figura 14, Reload Editor al crear el paquete, crea automáticamente 5 archivos: el archivo manifiesto y los esquemas.

Añadir los recursos

Para añadir los recursos, sólo es necesario seleccionarlos del panel de recursos y arrastrarlos hacia el panel manifiesto, sobre la carpeta recursos.

Tipos de recursos

Un SCORM CP puede manejar dos tipos de recursos: Sharable Content Object (Contenidos Educativos Reutilizables. En adelante, SCO) y Asset (recurso).

- SCO: Son los archivos que se comunican con el LMS por medio de SCORM. Por ejemplo, es una página Web Html programada con javascript.
- Asset: Son los archivos complementarios que están vinculados al SCO, pueden ser archivos jpg, swf, css, etc. Los asset no pueden establecer comunicación directa con el LMS, deben hacerlo por medio del SCO. Por ejemplo: Un archivo swf (flash) debe comunicarse con la página html que le contiene, para que ésta se comunique con el LMS.

Para definir un SCO o Asset ir al panel de manifiesto, en la carpeta recursos y seleccionar el archivo correspondiente, por ejemplo: "miScorm.html".

Luego ir al panel de atributos y, en la opción "SCORM Type", elegir sco o asset según el caso. Como se visualiza en la figura 15.

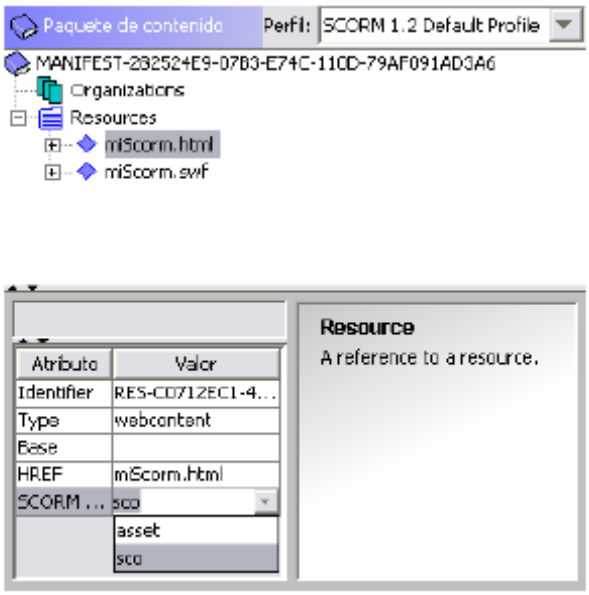


Figura 15. Tipos de SCORM

Para poder definir organizaciones, ítems y referencias a elementos se deben seguir los mismos pasos definidos anteriormente para paquetes de contenidos.

Pasos para exportar como paquete SCORM

En primer lugar se debe guardar el paquete generado. Luego debemos exportarlo como un archivo comprimido zip. Para esto seleccionar de la barra de herramientas "Crear Paquete" o desde el menú Archivo seleccionar la opción "Crear Paquete de contenido".

Se abrirá un cuadro de diálogo, crear una nueva carpeta y dentro de ésta guarda tu archivo con el nombre a elección (incluyendo la extensión .zip, pero sin las comillas).

El paquete ya esta exportado y listo para ser importado por algún LMS.

2.3 Editores HTML Open Source

2.3.1 HTMLArea

HTMLArea es un editor HTML WYSIWYG Web, gratis y personalizable (Ref. 17).

Funciona dentro de un navegador. Utiliza características no estándar, implementadas en Internet Explorer 5.5 o superior para Windows y en Mozilla 1.3 o superior (para cualquier plataforma), por lo que solo funcionará en uno de estos browsers.

Los derechos de autor del HTMLArea pertenecen a InteractiveTools.com (Ref. 18) y a Dynarch.com (Ref. 19) y se liberó bajo una licencia de tipo BSD.

HTMLArea fue creado y desarrollado hasta la versión 2.03 por InteractiveTools.com. La versión 3.0 fue desarrollada por Mihai Bazon para InteractiveTools. Contiene código patrocinado por terceros.

Es una herramienta para crear elementos de formulario, parecidos a los <textarea>, pero con la particularidad de que permiten introducir texto con estilos, como negritas, subrayados, distintos tipos de fuentes e incluso, tablas o imágenes. En definitiva, provee de las funciones típicas de los editores HTML WYSIWYG, pero dentro de un campo de formulario de una página Web.

HtmlArea se incluye en la página con unas pocas líneas Javascript fáciles de escribir. Con ello obtenemos un editor que permite funcionalidades como:

- Formatear texto con negritas, cursivas y subrayados
- Cambiar la tipografía y el color
- Alinear los distintos párrafos
- Incluir listas, líneas horizontales, links, imágenes, etc.

Como el programa está realizado en Javascript y trabaja únicamente del lado del cliente, lo podremos utilizar en cualquier tipo de servidor (no requiere programación ASP o PHP). La desventaja es que funciona únicamente en versiones de Internet Explorer 5.5 o superiores. Por lo menos, en otros navegadores, no dará errores, sino que simplemente veremos un campo <textarea> normal.

2.3.1.1 Características

HTMLArea ha llegado a la versión 3.0. Esta versión soporta:

- Barra de herramientas personalizable
- Fácil internacionalización
- Infraestructura basada en Plugin
- Genera código HTML W3 compatible (con algunas excepciones)
- Tiene un subconjunto de métodos abreviados similares a los de Microsoft Word
- Editor a pantalla completa
- Corrector ortográfico (a través del plugin externo "SpellChecker")

Es muy sencillo, tanto en su instalación como en la utilización. Es muy interesante también porque se puede utilizar simplemente desde Javascript. No hay necesidad concreta de disponer de un servidor con posibilidades de programación en ASP o PHP, aunque lo lógico es que los utilicemos para combinarlos con el editor WYSIWYG y permitir que los usuarios actualicen información de la base de datos, incluyendo sus estilos, imágenes, etc.

Pero su principal desventaja es que no esta actualizado y no brinda soporte técnico a los programadores.

2.3.1.2 Entorno del editor en la aplicación Moodle

Moodle utiliza un interfaz fácil e intuitiva con el que resulta sencillo familiarizarse rápidamente. Por lo general la información más relevante es mostrada en el centro de la pantalla mientras que a la izquierda y a la derecha se muestran los llamados "bloques" de Moodle. Los bloques son utilizados para albergar toda clase de herramientas y funcionalidades.

Una vez creado un curso, comenzará el proceso de añadir contenidos al mismo como por ejemplo: editar una página de texto o web, acceso a una archivo o URL, etc. La manera de añadir contenidos en un curso pasa por activar el "Modo Edición", situarnos en cualquiera de las secciones del nuevo curso creado y elegir una de las opciones del desplegable "Agregar Recurso", como por ejemplo "Editar una pagina de texto". En la figura 16 se puede observar las opciones del desplegable.

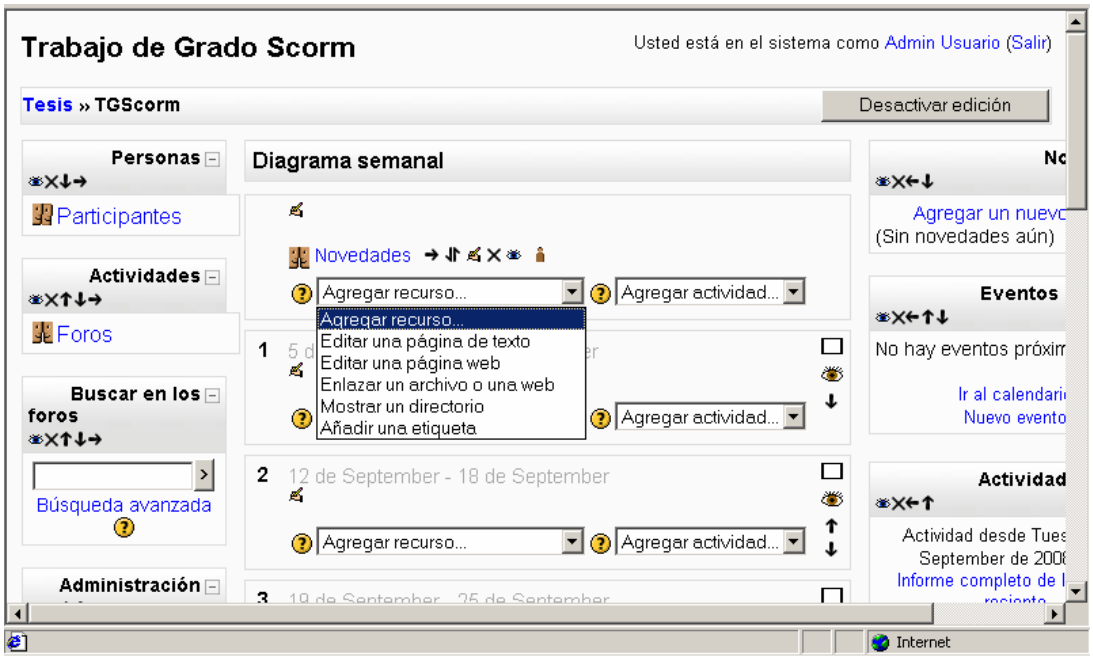


Figura 16. Pantalla de añadir contenidos en un curso

En esta unidad nos centraremos en el editor que brinda Moodle a sus usuarios para la creación y edición de paginas HTML, ya que es fundamental en el objetivo de nuestro desarrollo del trabajo de grado.

Editar una página de texto

Una página de texto es un fragmento de texto plano sin ningún tipo de formato. Pueden escribirse párrafos y espacios en blanco, pero nada más.

Las páginas de texto son muy sencillas de crear:

1. Activar el modo edición.
2. Seleccionar la opción "Añadir una página de texto" del menú "Agregar Recurso".

3. A continuación Moodle mostrará una página para construir el texto con varios campos a cumplimentar: (Vease la figura 17)
    - Nombre: el nombre que tendrá la página de texto. Este nombre será el mostrado en la sección correspondiente de la página principal del curso. Los alumnos accederán a la página pulsando sobre este nombre, por lo que es importante que sea un nombre descriptivo de la información con la que se encontrarán.
    - Informe: en este campo se añadirá un resumen de los contenidos de la página de texto. Los alumnos podrán acceder a dicho resumen mediante el bloque Actividades. Aunque estamos editando una página de texto plano el resumen puede ser añadido mediante el editor html integrado en Moodle.
    - Texto completo: este es el lugar dónde añadir el texto de la página. Si se poseen conocimientos de html podemos añadir etiquetas para mejorar el formato del texto.
    - Formato: Moodle da la posibilidad de que el mismo LMS formatee el texto si elegimos la opción "Automático", de dejar el texto plano o de dar formato sólo al texto escrito si se optara por la opción "Markdown".
    - Ventana: seleccionando la opción "Mostrar ajustes" se puede decidir que el nuevo recurso aparezca en la misma ventana de navegación o en una nueva (popup). En este caso se deben decidir las características de la ventana nueva, tales como su tamaño, la barra de desplazamiento o los menús que muestra.

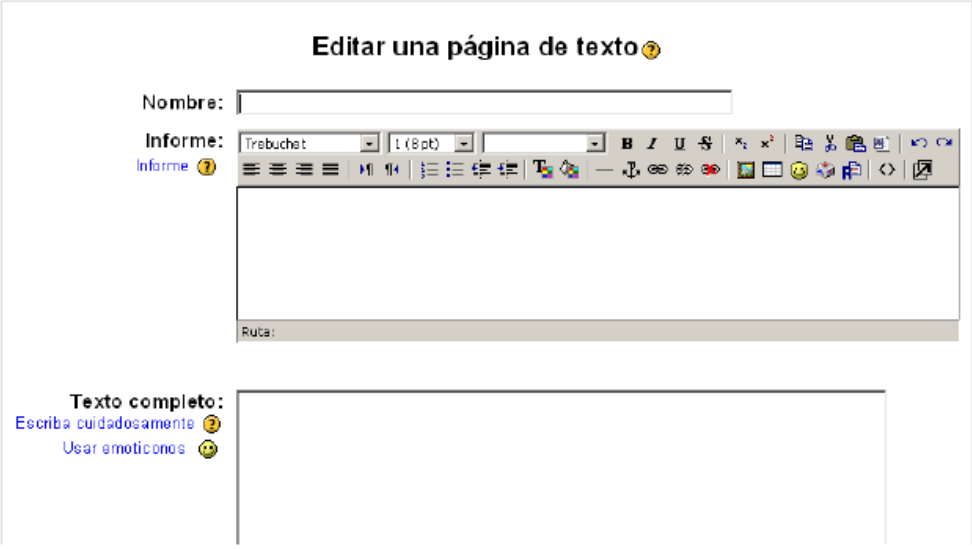


Figura 17. Pantalla de edición de una página de Texto

Una vez ingresada toda la información solo bastara con pulsar el botón "Guardar los Cambios"

Editar una pagina Web

Moodle tiene un editor html integrado con el que se pueden generar documentos que pueden ser interpretados por un navegador Web. El editor de html de Moodle tiene un aspecto similar al de cualquier procesador de textos y permite dar formato a nuestros documentos dándonos la posibilidad, entre otras cosas, de incorporar tablas, imágenes, hipervínculos, etc.

Este editor no funciona en todos los navegadores Web. Su funcionalidad está demostrada en Netscape 7, Internet Explorer 5.5 o superior, Mozilla 1.7 y Firefox, pero no funciona con Safari, Camino u Opera.

Para añadir una página Web a un curso habrá que seguir los siguientes pasos:

1. Activar el "*Modo Edición*".
2. Seleccionar "*Editar una página Web*" del menú "*Agregar recurso*" en la sección en la que queramos añadir la página.
3. Se abre una ventana similar a la de edición de una página de texto. (Como la que se muestra en la figura 9) Rellenar los campos "*Nombre*" e "*Informe*" del formulario.
4. Crear el documento rellenando el campo "*Texto completo*" haciendo uso del editor HTML. Utilizando la barra de herramientas del editor como se muestra en la figura 18.
5. Decidir si la página Web se abrirá en la misma ventana del navegador o en una nueva. En este último caso habrá que elegir las características de ésta.
6. Pulsar el botón "Guardar Cambios". A continuación se nos mostrará la página creada con el formato escogido tal y como la vería cualquier alumno del curso.

Figura 18. Barra de herramientas Editor HTML de Moodle.

En la tabla 2 se describen cada una de las funcionalidades de la barra de herramientas disponibles en el editor HTML de Moodle:

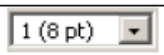
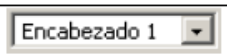
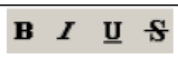
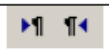
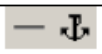
	Elección de fuente.
	Elección del tamaño de la fuente.
	Elección del estilo.
	Negrita, cursiva, subrayado y tachado.
	Subíndices y superíndices
	Copiar, Cortar, Pegar y limpiar html de Word.
	Deshacer y rehacer
	Justificados a la izquierda, al centro, a la derecha y completo.
	Color del texto y del fondo.
	Sangrías.
	Numeración, listas por puntos, y tabulaciones.
	Líneas separadoras y puntos de anclaje.
	Hipervínculos, romper hipervínculos, añadir imágenes y tablas.
	Añadir emoticonos y caracteres especiales.
	Añadir etiquetas html.
	Maximizar el editor (muy útil si se utiliza Microsoft Explorer como navegador)

Tabla 2. Funciones de la barra de herramientas del editor Html de Moodle.

2.3.1.3 Posibles alternativas de reemplazo en Moodle

Con la instalación de Moodle se provee este editor, el cual es utilizado para crear y editar código HTML (Ref. 20).

El problema es que HTMLArea ha sido discontinuado, a tal punto que ni siquiera está disponible para bajarlo de su página oficial. Es más, dicha página se convirtió en un repositorio de diferentes editores HTML, gratis y comerciales.

Ante esta situación, cuando se presenta un error con el editor, no existe soporte a donde recurrir.

Por lo tanto, la comunidad Moodle esta analizando reemplazarlo por otro editor HTML más actualizado y que se encuentre activo. Para ello existen algunas

propuestas, pero antes de hacerlo, se están estudiando los desarrollos ya realizados que integran HTMLArea de manera que lo existente se pueda adaptar fácilmente al nuevo editor que se elija en su reemplazo.

En la página http://docs.moodle.org/en/Development:Moodle-specific_customisations_to_the_HTML_editor y en el foro http://moodle.org/mod/forum/discuss.php?d=76912 (cuyas entradas datan desde Jul/07 a Ene/08) se muestran las diferentes posiciones y opiniones que usuarios y desarrolladores de Moodle tienen respecto al tema del editor HTML.

Los cuatro editores propuestos para reemplazar a HTMLArea en Moodle son:

- **TinyMCE:** Bajo licencia LGPL. Se encuentra en constante desarrollo. Un punto en contra es su gran tamaño, lo que conlleva un mayor tiempo de carga de las páginas que lo incluyen.
  http://tinymce.moxiecode.com/

- **Xinha:** Bajo licencia BSD. Este proyecto es el sucesor del editor HTMLarea. Por tratarse de un proyecto Open Source, cuando los desarrolladores originales discontinuaron HTMLarea, otros desarrolladores continuaron el proyecto bajo este nuevo nombre.
  http://xinha.webfactional.com/

- **FCKeditor:** cuenta con tres tipos de licencias, GPL, LGPL o MPL. Se encuentra en constante desarrollo.
  http://www.fckeditor.net/

- **YUI Rich Text Editor:** Bajo licencia BSD. Es el más pequeño de todos pero aún esta en versión Beta y no tiene tantas características como Tiny o FCK.
  http://developer.yahoo.com/yui/editor/

Cualquiera de las cuatro opciones, son ampliamente usadas por otros proyectos Open Source y están en constante desarrollo, con versiones recientes de Mayo de 2007 o posteriores.

En cuanto a la compatibilidad con los distintos browser, todos funcionan en Internet Explorer y Firefox. Pero solamente TinyMCE y YUI-RTE actualmente cuentan con soporte para Opera y/o Safari, mientras que FCK si bien lo tiene como objetivo, da una buena justificación de porque aún no da soporte en su página: http://dev.fckeditor.net/wiki/Compatibility

2.3.1.4 Cómo se inserta el editor HtmlArea en cualquier aplicación

La instalación es fácil (o debería serlo). Es necesario descomprimir el archivo ZIP en una carpeta accesible a través del servidor Web.

Primer paso, descargar el software desde la página de inicio de HTMLArea. El archivo es muy pequeño y viene en .zip, por lo que debemos descomprimirlo en nuestra computadora.

Una vez descomprimido, encontramos ejemplos e instrucciones para su funcionamiento.

Ejecutando el archivo llamado example.html se puede hacer una pequeña prueba, para ver si el navegador soporta HTMLArea.

A continuación presentamos un ejemplo sencillo:

1. Crear un archivo HTML, para hacer nuestro primer ejemplo
2. En el mismo directorio donde hemos descomprimido el software. Lo editamos y colocamos en la cabecera el siguiente código:

```
<script language="Javascript1.2">
<!--
// Carga de htmlarea
_editor_url = "" // URL del archivo htmlarea var win_ie_ver =
parseFloat(navigator.appVersion.split("MSIE")[1]);
if (navigator.userAgent.indexOf('Mac') >= 0) { win_ie_ver = 0; }
if (navigator.userAgent.indexOf('Windows CE') >= 0) { win_ie_ver = 0; }
if (navigator.userAgent.indexOf('Opera') >= 0) { win_ie_ver = 0; }
if (win_ie_ver >= 5.5) {
document.write('<scr' + 'ipt src="' +_editor_url+ 'editor.js"');
document.write(' language="Javascript1.2"></scr' + 'ipt>');
} else { document.write('<scr'+'ipt>function editor_generate() { return
false; }</scr'+'ipt>'); }
// -->
</script>
```

3. Lo único que hay que editar en este código es la variable "_editor_url", a la que tenemos que asignar la ruta donde se encuentran los archivos de htmlArea. Como en este caso el archivo de ejemplo está en el mismo directorio que htmlArea, asignamos un string vacío a la variable:

   ```
   _editor_url = "" // URL del archivo htmlarea
   ```

   Crear un formulario con un campo textarea dentro.

   ```
   <form>
   <textarea name="campo1" style="width:500px; height:200px;">
   </textarea>
   </form>
   ```

   Nos fijamos que el campo textarea tiene un nombre, que luego utilizaremos. También hemos definido con atributos de estilos el ancho y alto del campo. Esto último es opcional pues, como cualquier campo textarea, también podríamos haber definido sus dimensiones con los atributos cols y rows.

4. Por último, debemos indicar que ese campo debe mostrarse como un contenedor HTML y no como un <textarea> normal. Para ello debemos incluir, a continuación del <textarea>, este código Javascript.

   ```
   <script language="JavaScript1.2" defer>
   editor_generate('campo1');
   </script>
   ```

Es una llamada a la función que se encarga de generar el editor HTML a partir del campo <textarea>. Nos fijamos que la función recibe un string con el nombre del campo <textarea> que se desea convertir a htmlArea.

Para conseguir que el campo htmlArea tenga un texto por defecto, simplemente debemos insertar ese texto entre <textarea> y </textarea>. Podemos insertar código HTML y se mostrará dentro de la propia página.

2.3.2 FCKeditor

FCKeditor es un editor de texto que reside en el servidor y es accesible a través de un navegador Web convencional (Ref. 21). Al integrar FCKeditor en la pagina Web, se crean los medios adecuados para que los usuarios editen sus propias paginas Web.

FCKeditor cuenta con las siguientes características:

- Es un editor WYSIWYG avanzado compatible con Internet Explorer 5.5+, Firefox o Netscape.
- Permite formatear un texto, como podría hacerlo un editor como MS Word, pero en Internet. Permite por ejemplo:
  - ❖ Formatear el texto (fuente, tamaño, color, estilo, negrita, itálica, alineamiento. indentación, listados,...)
  - ❖ Formatear tablas (colores, bordes,...)
  - ❖ Copiar/pegar
  - ❖ Crear enlaces
  - ❖ Cargar XHTML 1.0, CSS,...
  - ❖ Multilingüismo
  - ❖ Gestión de imágenes
  - ❖ Gestión de «anclajes» (enlaces internos de la página)
  - ❖ Compatibilidad con Internet Explorer 5.5+ así como los navegadores basados en Gecko (Mozilla/Firefox/Netscape) ...
- Se utiliza online, vía web, con lo cual puede ser incluido en cualquier página ya existente.
- Es una aplicación liviana que no requiere de ningún tipo de instalación en la máquina cliente.
- Soporta ASP, ASP.NET, ColdFusion, PHP, Java y JavaScript.
- A esta altura el editor ha sido profundamente probado, y se ofrece en una versión estable que puede ser integrada fácilmente en cualquier aplicación existente.
- Esta traducido a la mayoría de los idiomas, en particular al español.

Los resultados de cualquier acción o comando son inmediatamente visibles en la pantalla. En caso de error puede utilizarse el botón Deshacer  , que elimina el efecto de la última operación realizada. También pueden utilizarse el botón

Rehacer que permite repetir la última acción realizada. Si ha trabajado con procesadores de texto tradicionales se encontrará enseguida familiarizado con este nuevo editor.

2.3.2.1 La barra de herramientas

FCKeditor facilita la elaboración de textos para la web y nos evita tener que escribir el código HTML. Si tiene conocimientos de HTML puede ir directamente al código HTML y aplicar correcciones o añadir alguna función especial.

El lenguaje HTML se compone de elementos, cada uno de los cuales realiza una determinada función en el documento (títulos de distintos niveles, párrafos, figuras, etc.). Cada elemento contiene una marca o tag que define el tipo de elemento que es. Estos códigos de texto, conocidos como marcas HTML o tags, los inserta FCKeditor automáticamente cuando interactuamos con él, evitándonos de este modo el tener que conocer el lenguaje HTML para crear documentos para la web.

Pulsando sobre el botón Fuente HTML 

puede seleccionar si desea presentar el texto en lenguaje HTML y realizar modificaciones sobre el mismo o bien emplear el editor en modo WYSIWYG (*What You See Is What You Get* – Lo Que Ves Es Lo Que Obtienes) y dejar que sea el propio FCKeditor quien genere el código HTML.

Posee las funciones estándar de edición como copiar, cortar y pega. Si seleccionamos un texto de la ventana del editor se activan los comandos Cortar  o Copiar  . Luego, al activar el comando Pegar  el texto seleccionado se introduce en la posición donde está el cursor.

Formato de la fuente

FCKeditor, como cualquier otro editor de textos, le permite las funciones básicas para dar ciertas características a la fuente de su texto como puede ser el color, colocar texto en cursiva, negrita, subrayado, etc.

Existen otros dos botones en la barra de herramientas que pueden servirle de gran utilidad. El primero de ellos, "Seleccionar Todo"  , le permite seleccionar todo el texto de su documento y el segundo, "Eliminar Formato"  , eliminar el formato de la fuente del texto seleccionado.

El formateado de párrafos puede realizarse, como el resto de las operaciones, a través de los botones de la barra de herramientas.

Para aprender a utilizar los formatos, es conveniente perder el miedo a utilizar las distintas opciones y a hacer pruebas con ellas. En caso de error, se puede deshacer la última operación realizada con el botón Deshacer que permite deshacer varias operaciones anteriores y no sólo la última.

Asimismo, puede seleccionar en la parte inferior de la barra de herramientas, las siguientes opciones de formato de párrafo:

- Estilo y Formato: los estilos son una de las capacidades más importantes de un procesador de textos. Gracias a ellos es relativamente fácil dar formato a un documento, y más fácil aún modificar ese formato. Los estilos son conjuntos de características de formato que se aplican a determinados párrafos. Cada estilo tiene un nombre bien determinado.
- Tamaño: seleccione el tamaño de la fuente del texto.

Vínculos y Referencias

Una referencia cruzada remite al lector a información situada en otra parte del documento, como por ejemplo cuando se cita en un párrafo una figura, una tabla o un apartado con su número correspondiente.

Se tiene la opción de incluir una gran variedad de información en las referencias cruzadas. Si el contenido o posición de la información se modifica, se actualiza automáticamente la referencia cruzada para que refleje el cambio. Para agregar una referencia cruzada, sólo tiene que pulsar el icono Referencia  e introducir un nombre para referenciar el texto adjunto.

Posteriormente, si quiere vincular otra parte del texto a esta referencia, deberá Agregar/Editar Vínculo  , seleccionar el tipo de vínculo como una Referencia a esta página y escoger de la lista el nombre de la referencia que escogió en el paso anterior.

Como es posible comprobar en el cuadro de diálogo que aparece, puede crear vínculos tanto a referencias dentro del texto, como a una URL o correo electrónico, sólo debe seleccionar el tipo de vínculo y completar el resto de campos necesarios.

Para eliminar un vínculo creado, deberá seleccionar el texto que posee un vínculo asociado y pulsar el botón de Eliminar Vínculo  .

Tablas

Las tablas son muy fáciles de utilizar. En la barra de herramientas hay un botón que permite insertar tablas en el texto. Para insertar una tabla en el texto basta con colocar el cursor en el lugar en el que deba ser insertada, y hacer click en el botón Insertar Tabla  . A continuación se abrirá un cuadro de diálogo donde podrá determinar el número de filas y columnas que se desea tenga la tabla, además de otros parámetros como la altura y ancho, el tamaño de los bordes, el título, etc. Además, haciendo click con el botón secundario del mouse sobre una tabla, puede en todo momento añadir y/o suprimir filas y columnas, o también combinarlas.

Dentro de cada celda de una tabla se aplican los mismos criterios de formato que afectan a los caracteres y a los párrafos.

Búsqueda y sustitución de texto

Éstas son también capacidades de FCKeditor que pueden resultar de utilidad. El botón Buscar  permite encontrar un determinado texto en el documento. Puede seleccionar si atender o no a que las letras sean mayúsculas o minúsculas.

El botón Reemplazar  está también en la barran de herramientas, con la diferencia de que permite opcionalmente sustituir el texto encontrado por un texto alternativo. Este comando permite reemplazar las cadenas de texto una a una o realizar todas las sustituciones directamente, seleccionando Reemplazar o Reemplazar Todo, respectivamente. Existen opciones para considerar diferentes o no las letras mayúsculas y minúsculas, así como para considerar sólo palabras completas o también partes de una palabra.

Insertar imagen

Con FCKeditor puede también insertar una imagen en el texto, pero no trabaja realmente con los archivos de imagen, sino con su URL. Esto significa que si desea insertar una imagen que está alojada en una página web, por ejemplo, podrá hacerlo insertando la ruta de la imagen. También puede insertar imágenes alojadas en su propia computadora. Para ello, el archivo correspondiente debe subirse previamente a una carpeta de imágenes dentro de su cuenta de usuario.

Para insertar la imagen deberá pulsar en el icono Insertar Imagen  y luego, en el cuadro de diálogo que le aparecerá, pulsar en Ver Servidor. A continuación podrá seleccionar un archivo de imagen de su cuenta de usuario. Para insertar un archivo de su computadora, deberá elegir una carpeta filtrada de imágenes, pulsar el botón Examinar, seleccionar el archivo deseado, pulsar el botón Upload para subir el archivo a su cuenta de usuario y por último, hacer click en el nombre del archivo creado en su cuenta. En el mismo cuadro de diálogo, también podrá ajustar ciertos parámetros como la altura y el ancho final de la imagen.

Otras funcionalidades de FCKeditor

Finalmente, para completar esta descripción de las funciones de FCKeditor tenemos que hacer referencia a 4 botones de la barra de herramientas cuyo cometido aún no hemos comentado:

- Insertar Línea Horizontal  : pulsando este botón puede introducir una línea horizontal separadora en el texto.
- Insertar Caracter Especial  : seleccione esta opción si desea introducir un carácter especial en el texto. Para ello, sólo debe hacer click sobre el símbolo que desea insertar y éste aparecerá automáticamente en la posición del cursor.
- Insertar Emoticons  : de modo parecido a la opción anterior, puede insertar un emoticono en la posición actual del cursor haciendo click sobre aquél en el cuadro de diálogo que aparecerá al pulsar el botón correspondiente de la barra de herramientas.
- Teclado universal  : Si desea insertar un texto con caracteres de otro alfabeto como árabe, búlgaro o croata puede hacerlo fácilmente utilizando el teclado que aparecerá en pantalla cuando pulse el botón correspondiente en la barra de herramientas. Pulse Ok cuando termine y el texto aparecerá en la posición del cursor.

2.3.2.2 Instalación de FCKeditor en cualquier aplicación

A modo de ejemplo en esta unidad solamente trataremos los ejemplos del editor FCKeditor en el lenguaje JavaScript.

Instalación con Javascript

Luego de descargar el editor (Ref. 21), procedemos a realizar lo siguiente:

1. Creamos una carpeta bajo el nombre de FCKeditor (o el de tu gusto) donde colocaremos el código que viene en la carpeta editor (cuando lo descarguemos), con esto hecho procedemos a llamar al JavaScript, dentro de las etiquetas head colocamos:

```
<script type="text/javascript" src="/FCKeditor/fckeditor.js"></script>
```

2. Crear el formulario. Existen dos formas de hacerlos:

   - **Método 1**:
     Un primer metodo es ubicarnos dentro de la etiqueta body, en el lugar donde queremos que vaya el formulario y colocar lo siguiente:

```
<script type="text/javascript">
        var oFCKeditor = new FCKeditor( 'FCKeditor1' ) ;
        oFCKeditor.Create() ;
</script>
```

   - **Método 2**:
     El otro método consiste en reemplazar el Textarea de tu formulario ya existente, para esto debe tener un name o un id que lo individualice. El script quedaría de la siguiente forma:

```
<html>
<head>
<script type="text/javascript" src="/FCKeditor/fckeditor.js"></script>
<script type="text/javascript">
        window.onload = function(){
                var oFCKeditor = new FCKeditor( 'MyTextarea' ) ;
                oFCKeditor.ReplaceTextarea() ;
        }
</script>
</head>
<body>
<textarea id="MyTextarea" name="MyTextarea">Este es <strong>el valor
</strong> inicial</textarea>
</body>
</html>
```

3. Probar el formulario en el navegador

2.3.2.3 Definiendo los Skins

FCKeditor nos brinda 3 skins con los cuales podemos empezar a trabajar:

1. Default
2. Office2003
3. Silver

Skins que se encuentran dentro de las carpetas de iguales nombres. Por lo que si uno quiere rediseñar los colores de fondo, bordes, dimensiones, etc. Se debe acudir a estas carpetas.

Sin embargo, antes hacer la personalización, debemos elegir el skin de la siguiente manera:

- Abrir el archivo fckconfig.js, archivo que define el diseño, orden y distribución de los botones del editor, entre otras funciones, desde luego.
- En la línea 31 del archivo, se encuentra lo siguiente:

  FCKConfig.SkinPath = FCKConfig.BasePath + 'skins/default/' ;
  Lo cual modificamos por lo siguiente si queremos que se muestre el Skin de office2003:
  FCKConfig.SkinPath = FCKConfig.BasePath + 'skins/office2003/' ;
  De igual manera, para el Skink Silver
  FCKConfig.SkinPath = FCKConfig.BasePath + 'skins/silver/' ;

- Recargamos la página donde esta el editor, y vemos como cambia.

2.3.2.4 Determinando los Botones del Editor

Ubicar los botones del editor en el orden más adecuado, el que nos proporcione mayor comodidad, es algo que realizamos hasta con el propio MS Word. Por lo que es importante definirlo a gusto del cliente.

En el archivo fckconfig.js situarse en la línea 69, en esta línea empezamos a definir los botones que irán en el editor (modelo avanzado "default"), pero si nosotros queremos mostrar sólo algunos botones del editor, siguiendo el ejemplo de la instalación del editor en JavaScript, sería:

Abrir el archivo fckeditor.js y en la línea 29 modificar:

this.ToolbarSet = toolbarSet || 'Default' ;

por

this.ToolbarSet = toolbarSet || '**Office2003**' ;

Ahora el editor quedaría como se visualiza en la figura 19 (Skin: Office2003 personalizado):

Figura 19. Barra de herramientas de FCKeditor

Para modificar el color del editor, es muy simple, bastan con modificar archivo CSS FCKeditor, dentro de la carpeta del skin que hayamos elegido.

2.3.3 NVU

Herramienta gratuita para diseñar sitios Web (Ref. 22).

El creador de este proyecto es Daniel Glazman.

NVU es un editor de páginas web WYSIWYG es un programa que se puede utilizar en diferentes Sistemas Operativos o sea que es multiplataforma. Se encuentra basado en Mozilla Composer, pero de ejecución independiente.

Inicialmente pensado para rivalizar con aplicaciones como Dreamweaver o FrontPage, NVU es una alternativa para los que no tienen un gran dominio del HTML.

2.3.3.1 Características

- Ofrece una amplia variedad de herramientas para crear una página web, entre otros un servidor FTP integrado, un entorno de edición WYSIWYG intuitivo, con la posibilidad de pasar fácilmente al modo de código fuente.
- Para los colores, dispone de un editor muy fácil de usar, junto a un editor CSS eficaz para principiantes con escasas nociones de hojas de estilo.
- Está basado en el motor Gecko, el mismo que usa la familia de navegadores web Mozilla, lo que nos asegura una compatibilidad total con los estándares de edición web (XUL, CSS, XML y JavaScript). Otra característica es la posibilidad de personalizar la interfaz y barras de herramientas.
- NVU está disponible para Linux, Mac OS X y Microsoft Windows, aunque puede compilarse para cualquier plataforma con el Netscape Portable Runtime. Mozilla Composer está todavía en desarrollo como parte de la suite Mozilla para otras plataformas.
- NVU es muy fácil de usar e intuitivo.

Última versión: 1.0 fue liberado 29 de junio de 2005

2.3.3.2 Modos de edición

Como la mayoría de las interfaces de aplicaciones estándar, NVU está diseñado con:

- Barra de navegación en la parte superior, que incluye comandos genéricos de gestión de archivos, así como herramientas específicas para editar y modificar páginas.
- Barra de estado en la parte inferior que proporciona información de utilidad.

Presenta distintas formas de visualizar las páginas Html:

- Modo de edición normal: en este modo se muestra la página como un WYSIWYG (lo que ves es lo que obtienes), permitiendo una edición normal, insertando texto, imágenes, tablas, etc. Se ve la apariencia final de la página, pero las zonas y los objetos están recuadrados para facilitar su manipulación. Los objetos dinámicos no se ven.
- Etiquetas HTML: en esta vista, se muestran las "etiquetas" del lenguaje HTML en fondo amarillo, viendo la estructura de la página de una manera visual.
- Código fuente HTML: en esta vista se puede editar directamente el código HTML de la página en texto plano.
- Vista previa: vista previa de la página prácticamente igual a como se verá en el navegador. Desde esta vista también puede editarse el texto y el formato de los objetos.

En cualquiera de estos modos se puede editar el contenido de la página. Mediante las solapas ubicadas en la zona inferior, se puede cambiar fácilmente de un modo a otro.

2.3.3.3 Propiedades

- Editar páginas WYSIWYG (lo que usted ve, es lo que usted obtiene), convirtiendo la creación Web en algo tan sencillo como transcribir una letra con su procesador de textos.
- Administrar archivos integrados vía FTP.
- Código HTML fiable, capaz de funcionar con los navegadores más populares de la actualidad.
- Intercambiar entre el modo WYSIWYG y HTML utilizando pestañas.
- Soporte para formularios, tablas y plantillas.

2.3.3.4 Entorno de trabajo

Al ejecutar desde el escritorio el icono de acceso directo a NVU, automáticamente se despliega la siguiente pantalla principal o área de trabajo como se visualiza en la figura 20.

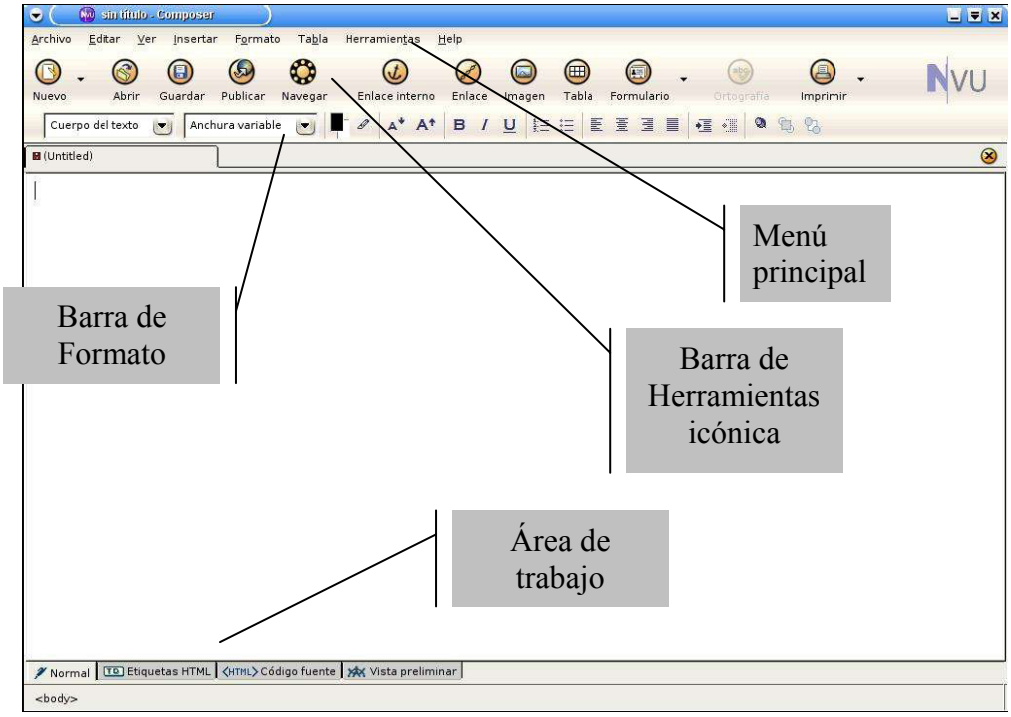


Figura 20. Editor NVU

El área de trabajo está concebida como el espacio que permite desplazarse entre las diferentes pantallas, ofreciéndonos la posibilidad de visualizar código (HTML), observar los cambios realizados y la forma definitiva en la que se publicará la página. En la figura 20 se puede observar al pie de la imagen lo mencionado anteriormente que es graficado en la figura 21.

Figura 21. Distintas área de trabajo de NVU

- Normal: Escenario que permite comenzar a crear elementos propios del HTML, con estructuras prediseñadas como tablas, imágenes, texto, entre otras.
- Etiquetas: Permite la edición de componentes al igual que en la vista normal y vista previa, sólo que se especifica sobre cuál etiqueta está el foco de trabajo. Está representada por HTML o TAGS.
- Fuente o código: Permite visualizar y editar el código fuente que se ha generado.

- Vista preliminar: Permite la previsualización de la estructura del documento.

Menú principal

La Barra de Menú Principal, está ubicada en la parte superior de la pantalla, contiene una serie de opciones imprescindibles para realizar diversas funciones así como los íconos de acceso directo más usados. La figura 22 se visualiza la barra.

Figura 22. Menú principal

Funciones del Menú principal:

- Archivo: se contemplan opciones como crear, abrir y guardar e imprimir archivos.
- Editar: Las opciones de edición son estándares en casi todas las herramientas; aquí se pueden localizar las opciones para copiar, pegar, deshacer, rehacer, seleccionar, buscar y reemplazar. NVU nos brinda una opción adicional denominada "Preferencias". En esta opción se configuran ciertas características del documento, las cuales se aplicarán cada vez que se abra la página en construcción. Entre los atributos están: colores del texto, enlaces (visitados, activos) y campos (como la cantidad de páginas recientemente visitadas).
- Ver: En esta opción se configuran las barras de herramientas, definiendo qué elementos se desean visualizar en el área de trabajo. También constituye otra vía para desplazarse entre los diversos modos de edición (normal, etiquetas, fuente y vista preliminar), otra opción es la de cambiar el Zoom del escenario, el cual viene graduado por defecto en 100%.
- Insertar: Permite incorporar elementos a la escena, tales como tablas, formularios, imágenes, anclas y enlaces externos. Uno de los puntos primordiales y de gran interés es la inserción de caracteres especiales, los cuales son necesarios dentro de cualquier contenido que requiera un código especial para poder ser visualizado.
- Los objetos inteligentes de los cuales dispone NVU y que pueden ser insertados, no se visualizan en el área de trabajo normal, etiqueta o vista preliminar, sólo hasta que se vea en un explorador se puede tener la seguridad de qué objeto funciona correctamente.
- Formato: Permite cambiar la apariencia de los elementos del documento como: tipografía, tamaño, color, estilo, formato del párrafo, inserción de listas, sangría, color del párrafo, letra y fondo.
- Tabla: Las tablas permiten tener mayor orden en la estructura de contenido bien sea gráfico o textual. Aquí podemos insertar, seleccionar y borrar una tabla, además de unir celdas seleccionadas, y definir el color de fondo de la misma.
- Herramientas: Se utiliza para validar el código generado en HTML, asignar y administrar las contraseñas por trabajo, asignar rutinas JavaScript desde una consola especial, así como para verificar la sintaxis

generada. La opción de mayor uso es el Editor de hojas de estilos o CSS (Cascade Style Sheet), esto se trata de una especificación sobre los estilos físicos aplicables a un documento HTML, los CSS tratan de dar la separación definitiva de la lógica (estructura) y el físico (presentación) del documento.

- Help (Ayuda): En esta opción está el enlace a la ayuda en línea desde el sitio oficial de NVU. Cabe mencionar que la mayoría de los documentos que se encuentran en la Web están en otros idiomas y la información en español es bastante escasa.

Menú contextual

En el área de trabajo, específicamente en la parte superior, están los íconos que dan acceso a las opciones de creación de elementos:

- Nuevo: Permite crear un nuevo documento en blanco, generando una pestaña de acceso identificada con el nombre que se asigno al archivo en el momento de guardarlo.
- Abrir: Permite ubicar un archivo existente a partir de una dirección o nombre del archivo.
- Guardar: Almacena el documento, proyectando una ventana donde se asignara el titulo que esta tendrá en el explorador.
- Publicar: Representa la publicación del documento desarrollado para un Web site. En esta pantalla se deben especificar datos puntuales del servidor, (IP. contraseña, usuario) y la estructura raíz que se debe respetar al momento de publicar.
- Navegar: Permite visualizar el documento en el explorador seleccionado.
- Enlace: Permite asignarle a un determinado objeto, una dirección externa o interna que genere un llamado a un enlace o link.
- Ancla: Se utiliza para identificar ciertos bloques en la estructura de navegación, realizando un desplazamiento de manera interna en los textos del documento.
- Imagen: Permite insertar un archivo de imagen, tomando en cuenta las rutas donde se encuentran estas ubicadas.
- Tabla: Conjunto de celdas que permiten organizar la estructura de la información a publicar. Igualmente adecua el formato de textos y gráficos presentes en el contenido.
- Formulario: Permite tener una interactividad en la Web, facilidad en planteamientos de selección múltiples, accesibilidad para la creación de cuadros de diálogo, texto, entre otros.
- Imprimir: Permite la impresión del contenido que se visualiza en el área de trabajo normal.

En la figura 23 se visualiza la barra.

Figura 23. Menú contextual

Barra de formato

Esta barra tiene las funcionalidades básicas de cualquier editor. Permite modificar la apariencia de documentos y contenidos. La apariencia se determina a través tamaño, alineación, tipografía, color, sangría, entre otras. En la figura 24 se visualiza la barra

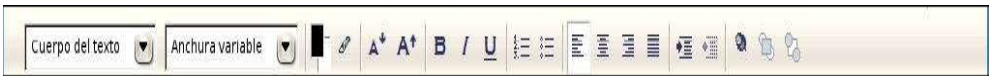


Figura 24. Barra de Formatos

Conclusiones

En esta unidad hemos mostrado las características de las distintas herramientas que entran en juego en la construcción y visualización de contenido compatible con SCORM.

Como herramienta para armar paquete SCORM hemos visto Reload Editor: permite crear paquetes SCORM de forma muy sencilla. Su interfaz es amigable y de fácil aprendizaje. El usuario puede armar sus cursos en forma rápida, pero debe armar sus páginas desde un editor HTML externo y luego importarlas en el Reload Editor, pues no posee la funcionalidad para poder crear páginas.

Como LMS vimos a Moodle: es la plataforma líder para toda institución mediana y grande. En constante evolución y de código abierto. Es un LMS compatible con SCORM que permite importar contenidos empaquetados, como los creados con Reload Editor y construir cursos integrados por páginas HTML (sin funciones de comunicación) a través del editor incorporado. Es una herramienta completa, con buena documentación para el usuario aunque la documentación para el desarrollador es escasa.

De la comparación de los diferentes editores HTML Open Source, con el fin de determinar cual era la mejor opción para poder aplicar nuestra propuesta de extenderlo, hemos concluido que:

- NVU queda descartado porque si bien es un editor muy completo y relativamente sencillo para el usuario, la última versión oficial data de febrero de 2005 y aunque existe un desarrollo alternativo no oficial llamado KompoZer que trata de seguir con el proyecto, no está claro cual va a ser su futuro.
- HTMLArea (incluido como editor en Moodle), también queda descartado porque ha sido discontinuado. Ante esta situación, tampoco es una buena opción seguir trabajando sobre este proyecto ya que no cuenta con soporte de ninguna clase y la comunidad Moodle esta analizando cual es la mejor opción para sustituirlo.
- FCKeditor, es el editor elegido, debido a que cuenta con las características básicas, está en constante desarrollo, es sencillo y fácil de utilizar.

Unidad 3: Construcción de la aplicación "FCKScorm for E-Learning"

Introducción

La mayor complejidad en el proceso de construcción de contenido educativo que soporta SCORM, está dada en aprovechar la funcionalidad de comunicación entre el LMS y el contenido a través de la utilización de las funciones JavaScript.

El objetivo de esta tesis se basa en poder facilitar la creación de contenido SCORM disminuyendo el nivel de complejidad del proceso de agregar metadatos y funciones de comunicación a las páginas HTML que forman parte del curso. De esta manera, es posible generar contenido portable entre cualquier sistema LMS compatible con SCORM.

Para lograr el objetivo mostraremos cómo se puede extender un editor de código HTML (Open Source), para que permita agregar las funciones JavaScript utilizadas en la comunicación entre un paquete SCORM y un LMS en el momento de la creación o edición de una página.

Para ello, seleccionamos a modo de ejemplo algunas de las funciones de comunicación provistas por la API SCORM. El incorporar todo el abanico de funciones incluidas en el estándar SCORM sería una tarea demasiado larga para este trabajo y solo significaría extender la herramienta implementada siguiendo la misma filosofía empleada.

En la unidad anterior, hemos comparado diferentes editores HTML Open Source con el fin de determinar cuál era la mejor opción para poder aplicar nuestra propuesta y hemos seleccionado a FCKeditor como el más apropiado para nuestra tesis.

A partir de este editor construiremos una aplicación Web a la que llamaremos "FCKScorm for E-Learning", donde se integrará el editor FCKeditor "personalizado" para generar código HTML con contenido SCORM en forma totalmente transparente al usuario, permitiendo agregar el seguimiento de la actividad del alumno e información relativa a las evaluaciones.

De esta forma se podrían crear las páginas con nuestra Aplicación "FCKScorm", generar el paquete con Reload Editor y visualizarlo con Moodle como se explicó en la unidad anterior.

3.1 Etapas iniciales del desarrollo de la aplicación

3.1.1 Propuesta inicial

A partir de la idea propuesta en el ejemplo de la tesis "Usando XML para metaanotar recursos educativos" de la alumna Andrea Deluchi (Ref. 1) tratamos de adaptar el editor FCKeditor dotándolo de las opciones necesarias para poder generar dicho ejemplo.

Inicialmente pensamos en una aplicación que estaría formada por una sola página donde se visualizaría el editor FCKeditor adaptado por nosotros, conteniendo todas las funcionalidades necesarias y que desde allí se pudieran realizar todas las

acciones para plasmar la propuesta de este trabajo de grado. Esto es, que desde la barra de herramientas icónica del FCKeditor se pudieran llevar a cabo todas las tareas relacionadas con la creación de un curso SCORM, ya sea agregando contenido SCORM para la comunicación con un LMS o también tareas necesarias como edición de un página existente, guardado de una página, etc.

Para ello pensamos en agregar seis nuevos botones a la barra icónica del editor con funcionalidades correspondientes a SCORM.

Si bien se llegó a armar un prototipo a partir de ésta idea y se desarrollaron algunas de las funcionalidades descriptas, nos encontramos con inconvenientes como la necesidad de tener que acotar las funcionalidades de la API SCORM debido a la gran cantidad que posee. Por otro lado, esta propuesta respondía solo a un modelo de plantilla de curso SCORM, es decir, al ejemplo propuesto en la tesis de la alumna Andrea Deluchi.

A partir de esto, se buscaron otras alternativas para que las funcionalidades de la aplicación "FCKScorm" puedan ser ampliamente aprovechadas, logrando que la aplicación final responda a un modelo de curso más genérico.

Aquí nos encontramos con otro punto a resolver que era encontrar un modelo de curso genérico sobre el que pudiéramos trabajar y desarrollar nuestro trabajo de grado. Ante ésta necesidad de búsqueda de un modelo de cursos usado en forma masiva la codirectora nos plantea investigar el Proyecto CDTC brasileño, del cual seleccionamos finalmente el modelo de curso a implementar.

3.1.2 Modelo de curso seleccionado

El modelo de curso seleccionado es el utilizado por el Instituto Nacional de Tecnología de la Información (ITI) Brasileño en su proyecto CDTC (Centro de Difusión de Tecnología y Conocimiento) (Ref. 24). Los cursos organizados por este Instituto se vienen impartiendo desde hace dos años y han sido tomados por más de 100.000 alumnos.

El ITI (Instituto Nacional de Tecnología de la Información), a través del CDTC (Centro de Difusión de Tecnología y Conocimiento) propone unir los esfuerzos del sector público, privado y las universidades, con el objetivo de ampliar el conocimiento de la sociedad en el uso del software libre.

El ITI de Brasil y la ONTI Argentina (Oficina Nacional de Tecnologías de Información) firmaron un acuerdo donde la Argentina se compromete a contribuir en el enriquecimiento de software libre. Dentro del grupo de colaboradores en este tema se encuentra la UNLP y a partir de nuestra propuesta de tesis vimos que quizás nuestro trabajo de grado podría llegar a ser un aporte más a este objetivo.

CDTC viene implementando desde el año 2006 una serie de cursos a distancia sobre distintos temas y destinado a una gran diversidad de usuarios. Actualmente, la plataforma que ofrece estos cursos posee 23.000 usuarios registrados, 3.600 vacantes ocupadas en cursos y más de 100.000 vacantes ya fueron concluidas en cursos anteriores.

La organización del contenido de estos cursos se explicará más adelante en esta unidad en el ejemplo.

3.1.3 Cambios realizados a la propuesta inicial

A continuación se describen las funcionalidades pensadas originalmente que fueron desarrolladas, pero finalmente quitadas o modificadas de la idea original:

- Botones SaveScorm y SaveScormInic: Ambos botones permitían realizar prácticamente lo mismo, abrían una nueva ventana del browser donde se mostraba el resultado final de la página HTML generada por el usuario utilizando el editor adaptado. La diferencia entre los botones era que SaveScormInic le agregaba al resultado obtenido, es decir al código de la página, la inicialización y la finalización de la comunicación LMS-SCORM. Como se explicó en la unidad 1 estas funciones son necesarias para que un SCO comience y termine la comunicación con el LMS. El editor internamente modificaba la etiqueta body del HTML agregando las funciones de onload="SCOInitialize()", onunload="SCOFinish()" y onbeforeunload="SCOFinish()". En cambio, SaveScorm solamente permitía guardar el resultado generado.

  Por otro lado, ambas funcionalidades agregaban internamente al código de la página las llamadas a los archivos JavaScript necesarios para la comunicación, los cuales son provistos por la API.

- Botones PaginaSiguiente y PaginaAnterior: Estos botones fueron modificados de la idea inicial pero existen en el trabajo final, con lo cual lo detallaremos más adelante.

Los botones SaveScorm y SaveScormInic fueron eliminados porque se ajustaban únicamente al ejemplo propuesto en la tesis de la alumna Andrea Deluchi.

Finalmente se optó por modelar la aplicación para que permita construir una clase de curso que tiene una estructura particular y utiliza un conjunto de funciones de comunicación determinada, siguiendo un modelo prefijado. A continuación se detalla la aplicación desarrollada.

3.2 Especificación de la aplicación

Para poder plasmar nuestra propuesta en una aplicación real, construimos una aplicación Web que permite al usuario generar la estructura del curso, crear páginas con o sin el agregado de funciones específicas para la comunicación con un sistema LMS y la posibilidad de importar páginas generadas fuera de la aplicación.

La aplicación se diseñó para ser ejecutada en un servidor Web donde se almacenará toda la estructura del curso y las páginas generadas, con el objetivo de que pudiera ser accedida desde cualquier máquina conectada al servidor.

Con este diseño fue necesario brindarle al usuario la funcionalidad de exportar todo el curso generado con la aplicación fuera del servidor y de esta manera poder generar un paquete SCORM con una herramienta como ReloadEditor.

En la figura 25 se muestra un esquema con los pasos a seguir en la construcción de un paquete SCORM. Desde la creación de las páginas HTML con nuestro editor hasta la visualización con alguna herramienta LMS.

En el mismo es posible ver que el autor puede crear páginas HTML con funciones JavaScript incorporadas, sin conocer el lenguaje de programación.

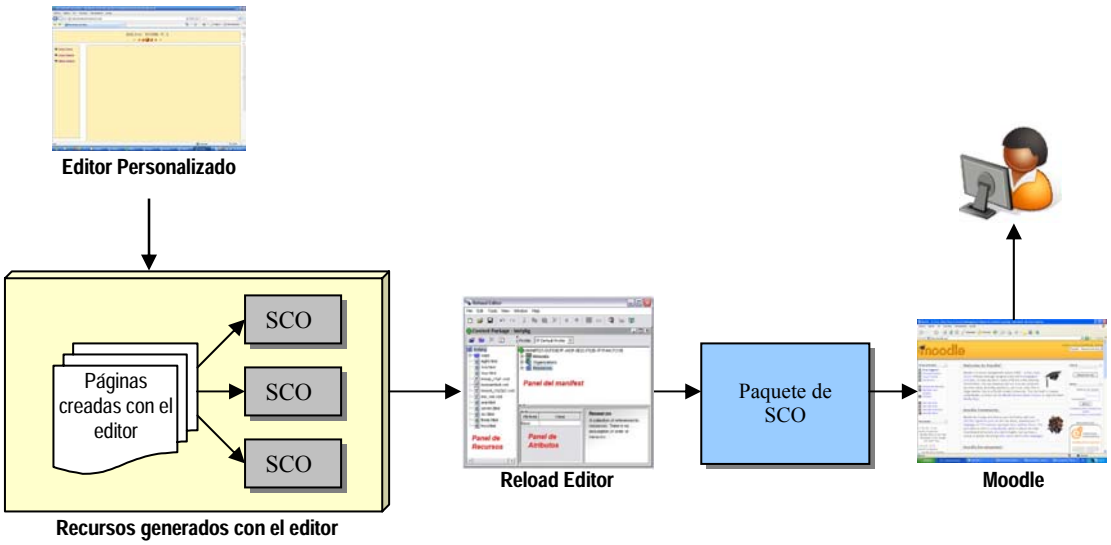


Figura 25. Esquema de creación de un paquete SCORM.

3.2.1 Funcionalidades provistas por la aplicación

A continuación se detallarán las funcionalidades provistas por la aplicación. La forma de uso se explicará más adelante en esta unidad a través de un ejemplo.

La aplicación cuenta con un menú principal en el panel izquierdo desde el que se puede acceder a las funciones necesarias para el desarrollo de un curso SCORM. La figura 26 muestra la interfaz de la aplicación.

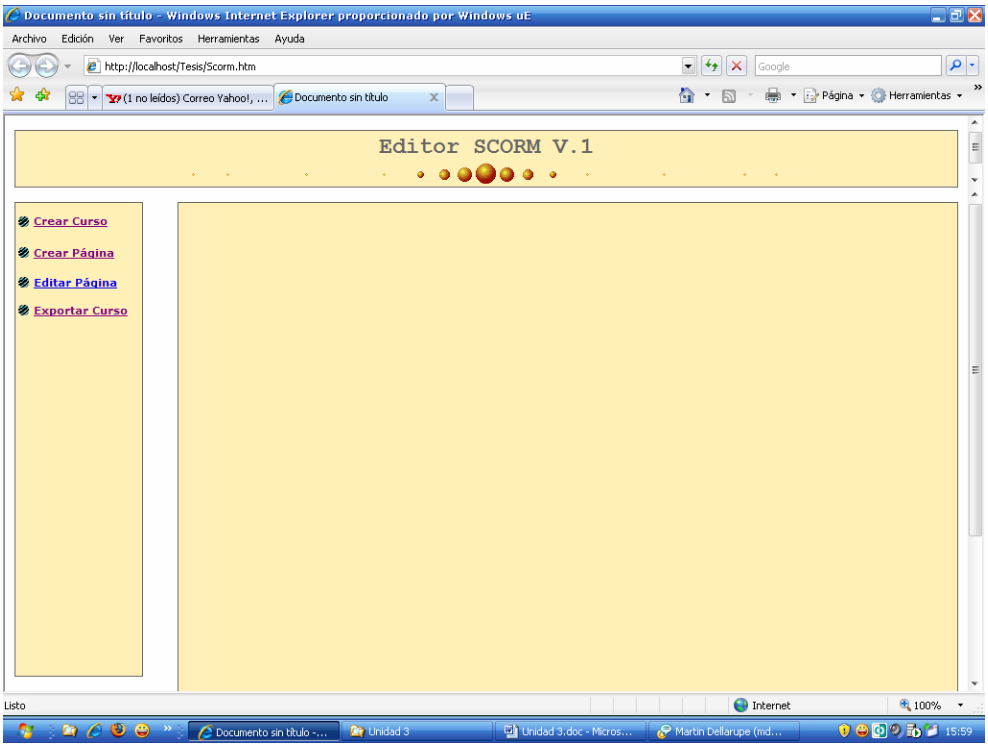
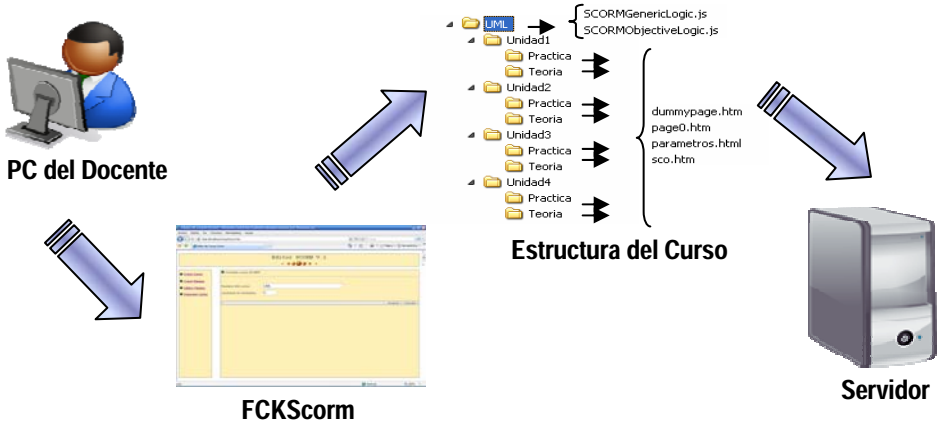


Figura 26. Aplicación "FCKScorm for E-Learning"

- **Crear Curso**

El proceso para crear un curso se puede visualizar en el esquema 1.



Esquema 1. Proceso de creación de un curso con "Editor SCORM V.1"

A partir de ciertos parámetros solicitados al usuario, la aplicación genera la estructura de carpetas que formarán un nuevo curso, siguiendo el modelo seleccionado, utilizado por el ITI. Los cursos se dividen en unidades y a su vez, cada unidad está formada por dos secciones, Teoría y Práctica.

Los parámetros requeridos son "Nombre del curso" y "Cantidad de unidades". Una vez ingresados ambos parámetros, la aplicación genera una carpeta con el nombre elegido para el curso y dentro, tantas carpetas como unidades se hayan especificado, numerándolas a partir de 1: Unidad1, Unidad2, …, Unidadn.

Dentro de cada unidad, la aplicación genera las carpetas Teoría y Práctica. Una vez realizado esto, copia los archivos dummypage.htm, page0.htm y sco.htm desde una ubicación fija en el servidor. Estos archivos fueron definidos por nosotros en forma genérica y son necesarios para formar un SCO. Luego, el conjunto de archivos de cada carpeta Teoría o Práctica, compuesto por los archivos agregados inicialmente por la aplicación más los creados por el usuario, formarán un SCO.

Dentro de las carpetas "Teoría" y "Practica" de cada unidad, la aplicación copia los archivos SCORMGenericLogic.js y SCORMObjectiveLogic.js que contienen las funciones propuestas por ADL para la comunicación con el LMS.

La figura 27 muestra la estructura del curso generada por la aplicación.

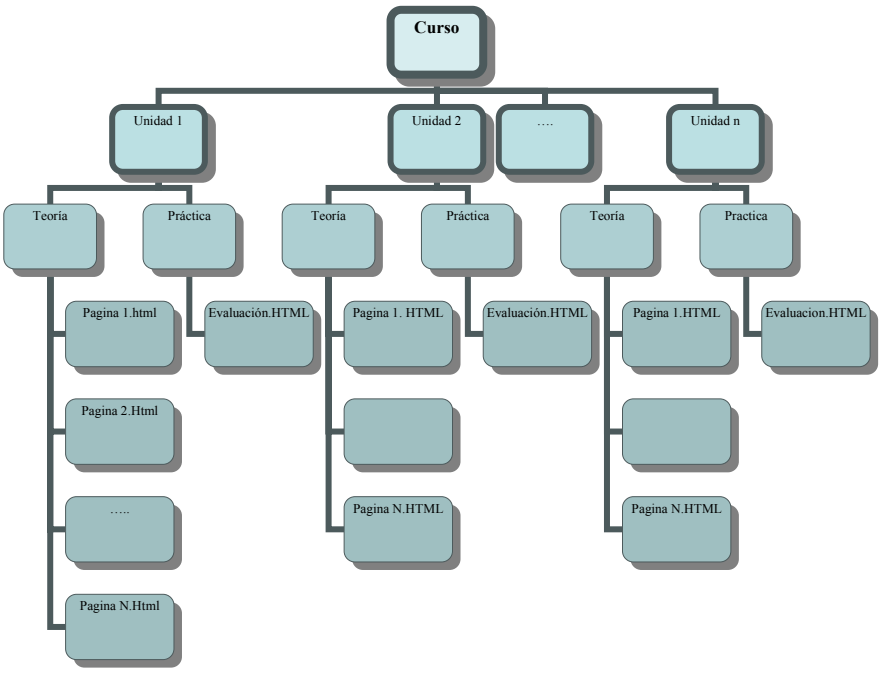


Figura 27. Estructura de un curso con editor FCKEditor

- **Crear página**

Permite crear una nueva página, la cual se agregará, según los datos que se ingresen, en una sección de Teoría o de Práctica de una unidad de algún curso existente en el servidor. Para ello se solicitan los parámetros Número de Unidad, Nombre del Curso y Número de Página.

Aquí es donde se integra el editor FCKeditor extendido, el cual ofrece además de las funciones básicas del editor, la posibilidad de agregar contenido SCORM a través de funciones JavaScript en el momento de la creación de la página. Estas funciones van a permitir entre otras cosas, registrar los resultados de las evaluaciones, ver la navegación del alumno por el curso y fundamentalmente permitir luego que esa página forme parte de un SCO.

La incorporación de las funciones JavaScript a la página se realiza a través de los nuevos botones agregados a la barra de herramientas del editor FCKEditor.

Para las páginas que formen parte de la Teoría, el usuario debe utilizar los botones "Página Anterior" y "Página Siguiente" para agregarle las funciones que permiten registrar la navegación del alumno. En tanto que para la página de Evaluación que forme parte de la Práctica, el usuario debe utilizar el botón "Radio Button" para armar el cuestionario múltiple choice y el botón "Calcular Nota" para agregarle las funciones que permiten registrar los resultados de las evaluaciones.

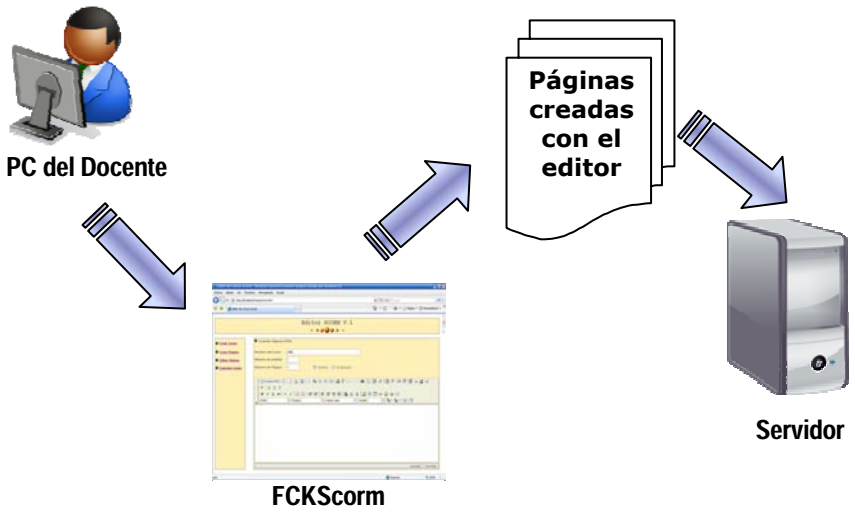
Una vez finalizado el diseño de la página, el usuario deberá presionar el botón "Guardar" para que la página se guarde en el servidor, según los parámetros ingresados.

Para las páginas que formen parte de la Teoría, la aplicación asignará automáticamente el nombre "unidadX_pagY.html" donde X es el número de unidad ingresado e Y es el número de página de la unidad. Por ejemplo en el caso de

páginas correspondientes a una teoría quedaría "unidad1_pag1.html", "unidad1_pag2.html", etc.

Para la página de evaluación que forma parte de la Práctica, la aplicación asignará automáticamente el nombre "evaluacion.html". En este tipo de páginas no se tiene en cuenta el parámetro Número de Página ya que en el modelo representado existe una única evaluación.

El proceso de creación de páginas se puede ver en el esquema 2.



Esquema 2. Proceso de creación de páginas con "FCKScorm"

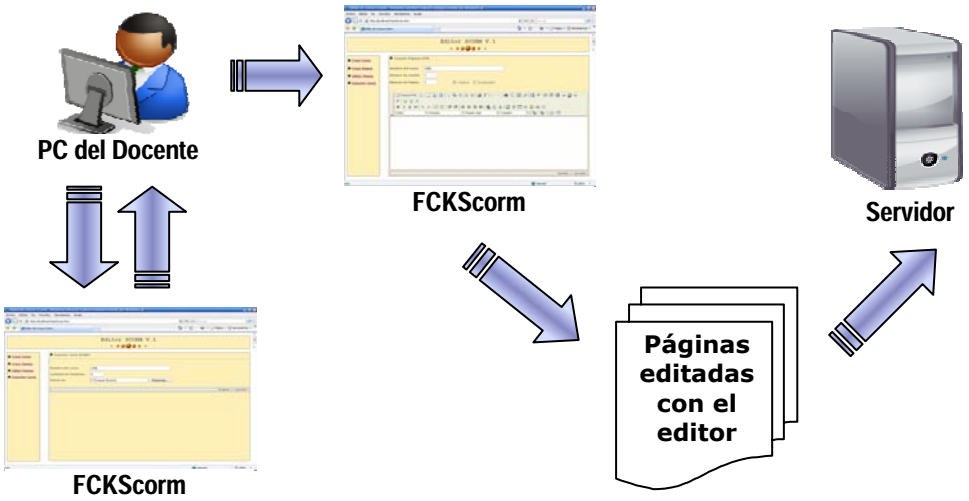
- **Editar página**

Permite seleccionar una página existente en la máquina local para su edición.

En el primer paso se presenta una pantalla donde el usuario debe escribir o seleccionar la ubicación de la página a editar. Una vez ingresada la ruta deberá presionar el botón "Abrir Página". Allí se pasa a otra pantalla con la misma funcionalidad que en la opción "Crear Página", pero en este caso el contenido de la página seleccionada se encuentra cargado en el editor FCKeditor para su edición.

La funcionalidad de editar una página existente no forma parte del FCKeditor original por lo que fue necesario buscar una forma de realizarlo y se optó por hacerlo en estos dos pasos, los cuales serán explicados en detalle en la sección donde se desarrolla el ejemplo práctico.

El proceso de edición de páginas se puede ver en el esquema 3.



Esquema 3. Proceso de edición de páginas con "FCKScorm"

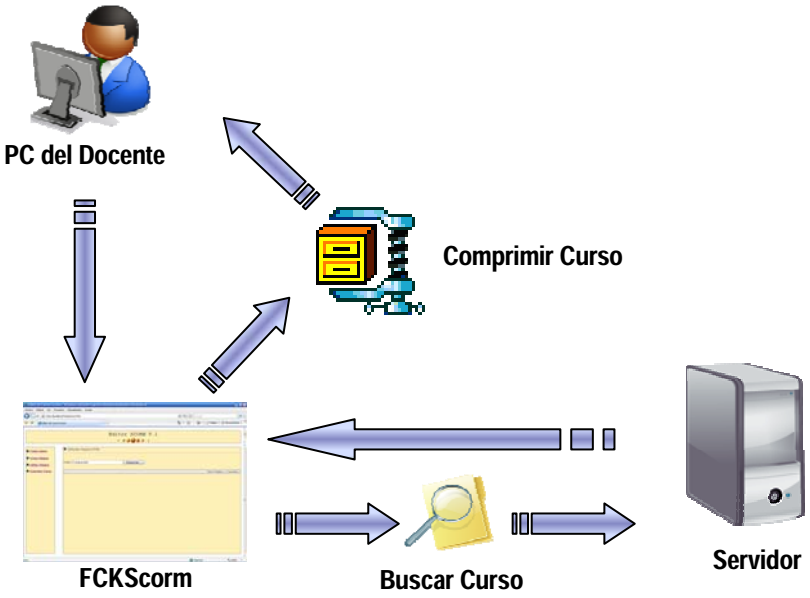
- **Exportar curso**

Permite obtener un archivo comprimido con la estructura completa de un curso generado por la aplicación en el servidor.

Esta opción se desarrolló ante la necesidad de que el usuario pueda llevar a su máquina local el curso creado con la aplicación, para poder armar el paquete SCORM con Reload Editor o alguna otra herramienta.

Aquí se presenta una pantalla donde se solicita al usuario que indique el nombre del curso a exportar y seleccione la ruta de la máquina local donde se almacenará el archivo comprimido y con que nombre.

El proceso de exportar un curso se puede ver en el esquema 4.



Esquema 4. Proceso de exportar un curso con "FCKScorm"

3.3 Desarrollo de la aplicación "FCKScorm for E-Learning"

Para poder llevar a cabo este trabajo de tesis, se utilizaron distintas herramientas de software:

- Un servidor Web.
- Software para construcción y edición de sitios y aplicaciones Web
- Editor HTML: FCKeditor

A continuación se describirán los pasos realizados durante el desarrollo y las funciones agregadas al editor FCKeditor.

3.3.1 Instalación del Editor HTML FCKeditor

El código fuente y los archivos necesarios para su instalación se obtienen de http://www.fckeditor.net/.

Aunque las instrucciones señalan que descomprimamos el paquete en el directorio FCKeditor bajo la raíz de archivos de nuestro servidor Web, en principio podemos situar el paquete en cualquier lugar. Para cambiar la ubicación, tendremos que configurar la propiedad BasePath del objeto editor. Esto se puede realizar de dos formas:

- Cambiando el *BasePath* dentro de la función constructora para la clase editor de forma tal que apunte a la ubicación donde hemos dejado nuestro editor. Al estar la ruta incrustada en el constructor, deberíamos comentar o eliminar el código extra dentro de la página donde invocamos al editor. La clase constructora es fckeditor.js
- Setear el *BasePath* al momento de instanciar el editor en la página que se cree para mostrar dicho editor

3.3.2 Creación de carpetas y archivos de la aplicación

Para generar la aplicación Web se crearon las siguientes carpetas y archivos:

1. En la raíz del servidor creamos la carpeta "Tesis". Esta carpeta contiene los archivos necesarios del FCKEditor y toda la aplicación desarrollada.
2. Dentro de la carpeta "Tesis" creamos la carpeta "FCKeditor", donde colocamos el código fuente obtenido de la página del editor.
3. Dentro de la carpeta "Tesis" creamos una página HTML a la cual llamamos "Scorm.htm" que será la página inicial de nuestra aplicación. Se diseño con tres frames: uno para el título, "Titulo.htm", otro para el menú, "Menu.htm" y el último que contendrá las páginas a abrir según el menú, por defecto se muestra la página "Cuerpo.htm".
4. Dentro de la carpeta "Tesis" creamos una carpeta "Curso" que tiene los siguientes archivos:
    - sco.htm
    - dummypage.htm

- page0.htm
- SCORMGenericLogic.js
- SCORMObjectiveLogic.js

Estos archivos son necesarios para completar el curso creado por el usuario, ya que las páginas creadas con el editor luego formarán los correspondientes SCO's que permitirán armar el paquete SCORM y poder visualizarlo en algún LMS.

Las tres primeras páginas fueron diseñadas en forma genérica para que puedan ser incorporadas en cualquier sección del curso, mientras que los dos últimos archivos son los propuestos por la API SCORM para las funciones que permiten mantener en el LMS el estado de navegación de las unidades por parte de los alumnos, así como también el resultado de las autoevaluaciones

La página sco.htm es la página que implementa en forma genérica las llamadas a las funciones JavaScript que se utilizaron para mantener en el LMS el estado de navegación del curso y de autoevaluación. Internamente estas funciones están diseñadas siguiendo el modelo de datos CMI.

5. Cada opción del menú tiene su página correspondiente que, como se dijo, serán cargadas en el frame central de la página Scorm.htm.

   La página correspondiente a la opción "Crear página" se la llamo "editorHtml.asp" y es la que contiene al editor FCKeditor adaptado.

   Para integrar el editor en esta página, se agrego dentro de las etiquetas head la llamada a la clase JavaScript del editor FCKeditor con la siguiente instrucción:

   ```
   <script type="text/javascript" src="/FCKeditor/fckeditor.js"></script>
   ```

   Luego, se agregó la referencia al editor siguiendo los paso del método 1 propuesto en la unidad 2.

6. Accediendo desde un navegador a nuestra aplicación se puede observar la pantalla inicial y al seleccionar la opción "Crear Página" se visualizara el editor como muestra la figura 29.

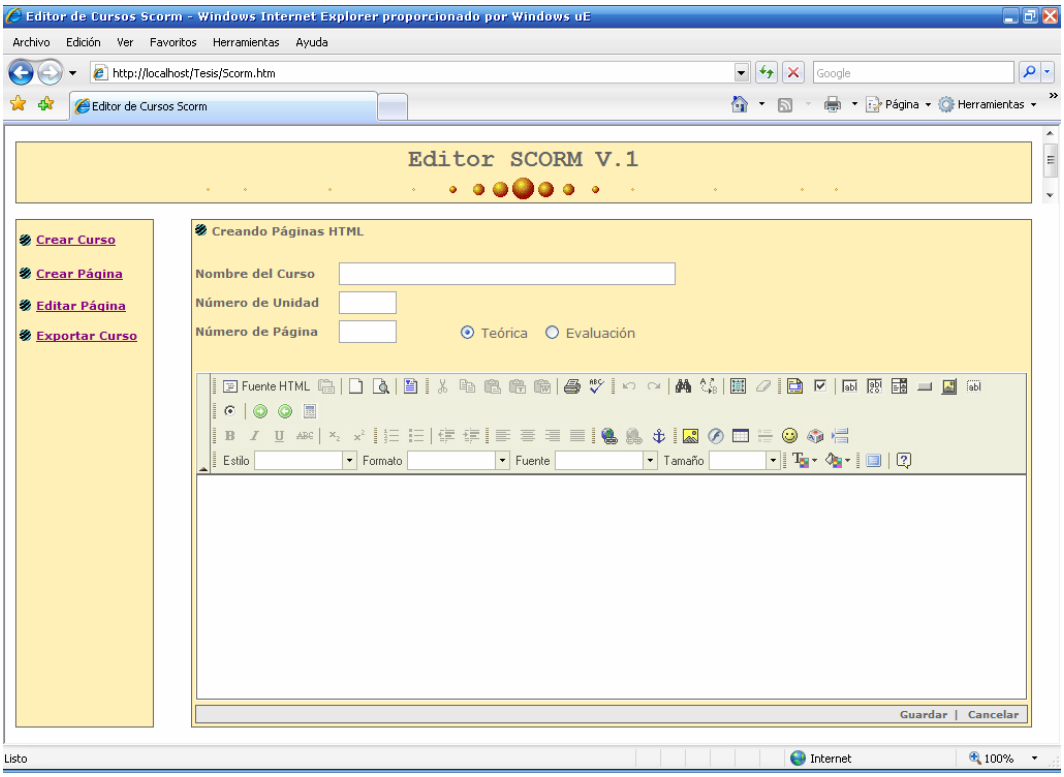


Figura 29. Aplicación FCKScorm – Opción "Crear Página"

3.3.3 Funciones agregadas al editor FCKeditor

El editor FCKeditor se modificó de forma tal que permite construir fácilmente cursos de enseñanza que se adapten al modelo "Teoría – Evaluación". Cabe aclarar que la evaluación se hará con el estilo múltiple choice.

Como se dijo anteriormente para las páginas que formen parte de la Teoría, el usuario puede utilizar los botones "Página Anterior" y "Página Siguiente" para agregarle las funciones que permiten registrar la navegación del alumno, en tanto que para la página de evaluación que forme parte de la Practica, el usuario puede utilizar los botones "Radio Button" y "Calcular Nota" para agregarle las funciones que permiten registrar los resultados de las evaluaciones.

Para adaptar el editor FCKeditor a nuestras necesidades decidimos crear una nueva barra icónica para el menú manteniendo las funciones básicas de cualquier editor HTML más los nuevos botones necesarios para permitir la comunicación de un curso SCORM con el LMS. Para tal fin agregamos los siguientes botones:

- **Botón "Radio Button":** este botón brinda dos funcionalidades similares que se diferencian entre sí porque una de ellas incluye una función JavaScript provista en la API SCORM.

  El usuario puede utilizar esta opción para agregar tanto un botón de radio "normal" como uno con contenido SCORM seleccionando en la ventana de dialogo que se abre que tipo de botón será.

  En el caso de las páginas de Evaluación, el usuario debería seleccionar la opción SCORM al crear un cuestionario con preguntas y posibles valores de respuesta para cada caso, para que luego cuando sea generado el SCO, este pueda comunicar los resultados obtenidos por el alumno al sistema LMS.

Al seleccionar esta opción, el editor abrirá una ventana emergente, solicitando al usuario el ingreso de ciertos parámetros:

1. Nombre: nombre con el cual la aplicación identificara al tag "radio button", es decir, valor que se asignara a la etiqueta "name".
2. Valor: es el valor que se le asignara a la etiqueta "value". En nuestro caso al ser utilizado para una página de evaluación este campo correspondería a una posible respuesta valida o no.
3. Un campo de chequeo indicando si el "radio button" aparecerá seleccionado o no.

Hasta acá son los campos que FCKeditor solicita para un tag "radio button" común. Para la personalización del editor hemos agregado los siguientes campos:

4. Un campo de chequeo que indica si el "radio button" soporta SCORM o no, es decir, en el caso que sea chequeado por el usuario el editor generara automáticamente en el tag la llamada a la función JavaScript correspondiente a la registración de la respuesta a la pregunta para la comunicación LMS - SCORM. En caso contrario de no ser chequeado este campo, el tag se generara sin la llamada a la función.
5. Nro. Pregunta: este campo será requerido si el campo del punto 4 fue chequeado e indica a que número de pregunta del cuestionario que esta diseñando el usuario se corresponde el tag "radio button". Este valor es usado por la aplicación para que al momento de generar la llamada a la función JavaScript para la registración de la respuesta del alumno, ésta reconozca a que pregunta corresponde la respuesta.

- **Botón "Página Siguiente":** permite ir de la página actual a la página siguiente de la unidad donde se guardará. El usuario podrá utilizar esta opción cuando desee que la página que está diseñando pueda ser navegada y que eso quede registrado en el LMS. Esta opción debería ser utilizada solamente para las páginas correspondientes a la Teoría.

  Al seleccionar esta opción, el editor abrirá una pantalla solicitando que se ingrese el nombre con el que desea que aparezca el link, como por ejemplo "Siguiente" o "Página Siguiente" y el número de la unidad donde se creará la página.

  Internamente el editor generará un link con el nombre ingresado y agregara al tag "link" la llamada a la función JavaScript que permite registrar la navegación del alumno.

- **Botón "Página Anterior":** permite ir de la página actual a la página anterior de la unidad donde se guardará. El usuario podrá utilizar esta opción cuando desee que la página que está diseñando pueda ser navegada y que eso quede registrado en el LMS. Esta opción debería ser utilizada solamente para las páginas correspondientes a la Teoría.

  Al seleccionar esta opción, el editor abrirá una pantalla solicitando que se ingrese el nombre con el que desea que aparezca el link, como por ejemplo "Anterior" o "Página Anterior" y el número de la unidad donde se creará la página.

  Internamente el editor generará un link con el nombre ingresado y agregara al tag "link" la llamada a la función JavaScript que permite registrar la navegación del alumno.

- **Botón "Calcular Nota":** permite registrar la nota correspondiente a una evaluación. El usuario podrá utilizar esta opción cuando desee que se registre en el LMS la nota obtenida por el alumno en la evaluación. Esta opción debería ser utilizada solamente para las páginas correspondientes a la Práctica.

  Al seleccionar esta opción, el editor abrirá una ventana emergente, solicitando al usuario el ingreso de ciertos parámetros:

  1. Nombre: nombre con el cual la aplicación identificara al tag "button", es decir, valor que se asignara a la etiqueta "name".
  2. Valor: es el valor que se le asignara a la etiqueta "value".

  Internamente el editor generará el tag "button" con la llamada a la función JavaScript que permite registrar la nota obtenida por el alumno.

La figura 30 muestra como quedó definida la nueva barra icónica del editor de páginas HTML.

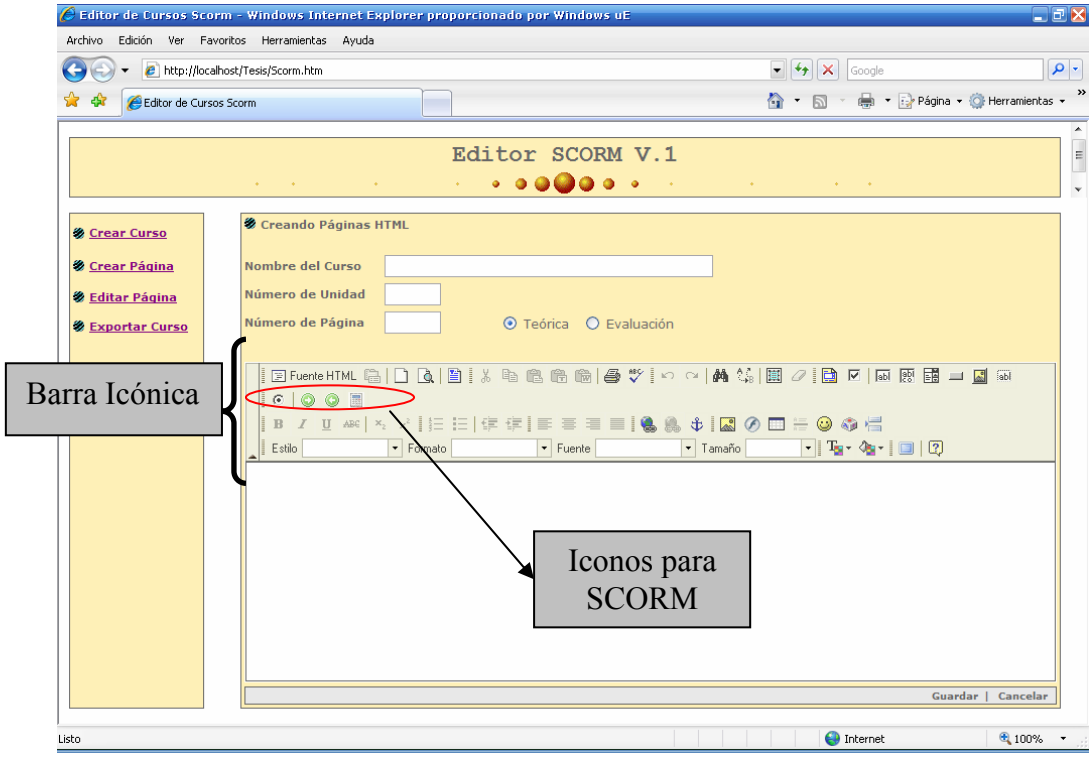


Figura 30. Visualización de la nueva barra icónica del editor FCKEditor.

El editor FCKeditor no proporciona el desarrollo de las funciones para abrir una página existente o guardar la página que se está creando.

Para contemplar la opción de "Guardar" como se explicó anteriormente se habían creado los botones "SaveScorm" y "SaveScormInic" que tenían como principal funcionalidad guardar la página creada con el editor.

Para desarrollar la opción de "Abrir" una página existente, se investigó cual era la mejor forma de hacerlo en nuestra idea inicial. La página que mostraba al editor en la aplicación estaba definida como una página HTML y se había optado por

definir en la página un tag "textArea" el cual contenía al editor. Esto se realizó siguiendo los pasos para la incorporación del editor en una página con el método 2 descripto en la unidad 2. Encontrándonos con un problema al enviar los datos al contenedor del editor FCKeditor, ya que al momento de setearle a la etiqueta "value" el contenido de la página que el usuario seleccionó para abrir, no se actualizaba el contenedor del editor. Esto se debe a que el editor FCKeditor trabaja con un iframe, el cual debe ser actualizado antes de ejecutar y enviar el post del formulario. Darnos cuenta de que el editor trabaja con un iframe y debíamos actualizarlo antes, nos llevo bastante tiempo. Observamos que al setear la página no se actualizaba el editor pero si presionábamos "F5" se actualizaba la página y el editor pasaba a contener la página seleccionada. No encontrábamos información referente a este problema hasta que en un foro había una persona que explicaba que el tema del iframe y algunos problemas que ocasionaba y de ahí surge la solución.

Si bien el problema de editar una página fue resuelta hay que tener en cuenta que el editor FCKeditor filtra las páginas que contengan el tag <meta>, es decir:

<meta http-equiv="" content="text/html; charset=iso-8859-1">.

Si uno prueba abrir una página con dicho tag, gráficamente la visualizara pero a la hora de guardarla el editor ignora dicha página, es decir, ignora todos los restantes tag que forman el HTML.

Lo que puede hacer el usuario es editar la página que contenga dicho tag, eliminarlo y luego editarla con la aplicación.

De todos modos el haber investigado la edición de una página y haberlo resuelto no nos sirvió para desarrollar la opción actual ya que la página que muestra el editor (editorHtml.asp), fue definida como ASP para poder contemplar la estructura de los cursos a desarrollar con el editor y como se explica a continuación se opto por redefinir el contenedor del editor siguiendo los pasos del método 1 descripto en la unidad 2.

3.3.4 Componentes internas del editor FCKeditor

La mayor desventaja de FCKeditor es la falta de documentación. Lo poco que existe hace repetidas referencias a los ejemplos proporcionados. Aunque los ejemplos nos ayudaron a ver lo que hace el código fuente, tuvimos que investigar, decodificar y seguir arduamente el programa para poder identificar y comprender cada una de las funciones de dicho editor.

Los ejemplos que nos proporciona FCKeditor los encontraremos en el directorio _samples, para cada lenguaje de desarrollo proporcionado.

Cada uno de los cuatro ejemplos de ASP proporcionados en el paquete muestra los diferentes aspectos del editor. Por ejemplo, uno de estos ejemplos nos permite elegir la barra de herramientas, otro el idioma, etc. Cada uno de los componentes puede añadirse a cualquier página, de manera que podemos construir un editor personalizado que nos permita seleccionar cualquiera de las opciones.

Para poder analizar algunos de los archivos que a continuación se nombran tuvimos que descompactar el código interno, ya que los archivos que componen FCKeditor son distribuidos con el código compactado para que sea más rápida su carga.

Al hablar de código compactado nos referimos a que el código fuente de cada uno de los archivos se presenta sin ningún tipo de identación, una instrucción

a continuación de la otra, lo cual hizo más ardua la tarea de entender y poder seguir el código fuente.

El siguiente es un ejemplo que muestra un fragmento de código fuente. En él puede observarse la declaración de variables, funciones y estructuras de control separadas por ";" todo en una misma línea, lo cual dificulta la comprensión y seguimiento del código fuente.

```
var ALT=4000;String.prototype.Contains=function(A){return (this.indexOf(A)>-
1);};String.prototype.Equals=function(){var A=arguments;if
(A.length==1&&A[0].pop) A=A[0];for (var i=0;i<A.length;i++){if (this==A[i])
return true;};return false;};String.prototype.IEquals=function(){var
A=this.toUpperCase();var B=arguments;if (B.length==1&&B[0].pop) B=B[0];for
(var i=0;i<B.length;i++){if (A==B[i].toUpperCase()) return true;};return
false;};String.prototype.ReplaceAll=function(A,B){var C=this;for (var
i=0;i<A.length;i++){C=C.replace(A[i],B[i]);};return C;};
```

A continuación describiremos los archivos y carpetas que hemos modificado para poder adaptar las funcionalidades de las nuevas características del editor.

- fckconfig.js: define la configuración inicial del editor, su apariencia y variables de uso interno. En nuestro caso es fundamental este archivo porque es aquí donde vamos a tener que definir nuestra barra personalizada de iconos SCORM.

  Ubicación archivo: \FCKeditor\

- fckeditorcode_ie.js: define la funcionalidad de cada uno de los iconos de la barra del editor. Este archivo se encuentra compactado para un mejor rendimiento de carga.

  Ubicación archivo: \FCKeditor\editor\js\

- es.js: archivo de lenguaje, este es el correspondiente al español. Se define cada uno de los textos a mostrar por el editor.

  Ubicación archivo: \FCKeditor\editor\lang\

- Ventanas de diálogos: nos referimos a la ventana que se abre al presionar un icono de la barra del editor, solicitando el ingreso de algunos parámetros. Es el caso, por ejemplo del icono "Agregar un Botón". La figura 28 muestra la pantalla emergente al presionar dicho icono. Estas ventanas son páginas HTML.

  Ubicación archivo: \FCKeditor\editor\dialog\

Por ejemplo en el caso del botón es \FCKeditor\editor\dialog\fck_button.html

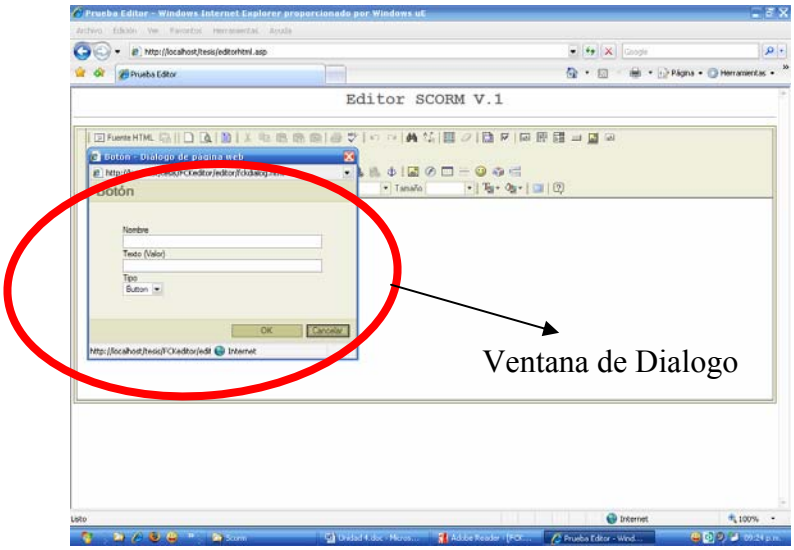


Figura 28. Ventana de Dialogo correspondiente al agregado de un boton del Editor FCKeditor

- Imágenes de la barra icónica: todas las imágenes que se muestran en la barra del editor se encuentran en la carpeta skins según el tipo de barra a mostrar:

  \FCKeditor\editor\skins\default\

  \FCKeditor\editor\skins\office2003\

  \FCKeditor\editor\skins\silver\

En cada una de estas carpetas existen dos subcarpetas, images y toolbar. En la carpeta toolbar se encuentran cada una de las imágenes de la barra icónica.

A continuación se explicará en detalle las adaptaciones realizadas al editor para permitir la creación de páginas con o sin el agregado de las funciones JavaScript que nos brinda el modelo de datos CMI de la API SCORM para la comunicación de las páginas con un sistema LMS.

3.3.5 Cambios realizados al código fuente original

Detallaremos las modificaciones realizadas a cada una de las componentes internas del editor FCKeditor:

➢ **Archivo fckconfig.js:**

En este archivo de configuración del editor hemos definido la nueva barra icónica "SCORM" y configuramos el lenguaje a español. Para realizar ambas personalizaciones:

- Barra icónica: definimos una nueva barra icónica bajo el nombre "SCORM", agregando el siguiente código:

```
FCKConfig.ToolbarSets["SCORM"] = [
        ['Source','DocProps','-','NewPage','Preview','-','Templates'],
        ['Cut','Copy','Paste','PasteText','PasteWord','-','Print','SpellCheck'],
```

```
        ['Undo','Redo','-','Find','Replace','-','SelectAll','RemoveFormat'],
        ['Form','Checkbox','-
,'TextField','Textarea','Select','Button','ImageButton','HiddenField'],
        '/',
```

//Botones nuevos correspondientes a la funcionalidad de SCORM

```
        ['RadioScorm','-','PaginaSiguiente','PaginaAnterior','CalcularNota'],
```

//Fin botones nuevos correspondientes a la funcionalidad de SCORM

```
        '/',
        ['Bold','Italic','Underline','StrikeThrough','-','Subscript','Superscript'],
        ['OrderedList','UnorderedList','-','Outdent','Indent'],
        ['JustifyLeft','JustifyCenter','JustifyRight','JustifyFull'],
        ['Link','Unlink','Anchor'],
        ['Image','Flash','Table','Rule','Smiley','SpecialChar','PageBreak'],
        '/',
        ['Style','FontFormat','FontName','FontSize'],
        ['TextColor','BGColor'],
        ['FitWindow','-','About']
]
```

La barra esta formada por los botones básicos del editor más los nuevos botones definidos por nosotros para soportar la funcionalidad LMS - SCORM que ya hemos descripto. Dicha barra será la que se visualiza al cargar la aplicación Web del editor personalizado.

- Configuración del lenguaje: hemos deshabilitado la auto-detección del idioma y fijamos el lenguaje por defecto en español. Para realizar esto nos situamos en la línea 61 de este archivo y cambiamos

```
FCKConfig.AutoDetectLanguage    = true ;
FCKConfig.DefaultLanguage       = 'en' ;
```

Por

```
FCKConfig.AutoDetectLanguage    = false ;
FCKConfig.DefaultLanguage       = 'es' ;
```

➢ **Archivo fckeditorcode_ie.js:**

Este es el archivo más importante del editor FCKeditor porque en el se implementa las distintas funcionalidades de cada uno de los botones de la barra que se muestra al instanciarlo.

Queremos destacar que la interpretación de cada una de las funcionalidades descriptas en este archivo requirió un minucioso trabajo de descompactación del código JavaScript porque como se dijo anteriormente este archivo se encuentra compactado para una carga más rápida lo cual nos dificulto mucho poder buscar e interpretar las funcionalidades.

A continuación detallaremos las modificaciones que se realizaron:

- En primer lugar indicamos cuales serán las imágenes que se mostrarán para cada uno de los botones que hemos definido en el archivo "fckconfig.js" para nuestra barra "SCORM". Para ello, creamos un skin que contiene todas las imágenes de la nueva botonera como se describió en la Unidad 2 y modificamos la función "FCKToolbarItems.GetItem", que asocia a cada botón de la barra su

correspondiente imagen y ToolTip.
Para modificar la función, en la sentencia switch (A) agregamos las nuevas opciones de los botones que soportan LMS-SCORM, como ya se definió.

Por cada uno de los botones debimos crear una nueva instancia de la clase "FCKToolbarButton" indicando:

1. Primer parámetro: el nombre del botón que debe coincidir con el nombre que le pusimos al crear la barra SCORM en el archivo "fckconfig.js".
2. Segundo parámetro: con que nombre debe buscar la clase FCKLang en el archivo es.js la descripción del toolTip para ese botón.
3. Tercer parámetro: la imagen del botón. Existen dos formas de indicar cual es la imagen que se desea cargar para un botón dado:

   ❖ **Método 1:** se indica como último parámetro de la función "FCKToolbarButton" el número de imagen. Este número lo saca de la imagen que se encuentra en el gif "fck_strip.gif" cuyo path es "\FCKeditor\editor\skins\scorm\". Por ejemplo:

```
B=new
FCKToolbarButton('RadioScorm',FCKLang.RadioButton,null,null,null,null,50)
```

   ❖ **Método 2:** se indica como último parámetro de la función "FCKToolbarButton" el valor null, entonces dicha función buscara en "\FCKeditor\editor\skins\scorm\toolbar\" la imagen con el nombre indicado en el primer parámetro, en este caso "paginaAnterior.gif". Por ejemplo:

```
B=new
FCKToolbarButton('PaginaAnterior',FCKLang.PaginaAnterior,null,null,null,null,null);
```

A continuación se puede observar el fragmento de código que se ha modificado en la función GetItem().

```
FCKToolbarItems.GetItem=function(A){
        var B=FCKToolbarItems.LoadedItems[A];
        if (B) return B;
        switch (A){
        ….
        ….
        case 'RadioScorm':
                B=new
FCKToolbarButton('RadioScorm',FCKLang.RadioButton,null,null,null,null,50);
                break;
        case 'CalcularNota':
                        B=new
        FCKToolbarButton('CalcularNota',FCKLang.CalcularNota,null,null,null,null,null);
                        break;
```

```
            case 'PaginaSiguiente':
                B=new
        FCKToolbarButton('PaginaSiguiente',FCKLang.PaginaSiguiente,null,null,null,null,null);
                break;
            case 'PaginaAnterior':
                B=new
        FCKToolbarButton('PaginaAnterior',FCKLang.PaginaAnterior,null,null,null,null,null);
                break;

        default:
            alert(FCKLang.UnknownToolbarItem.replace(/%1/g,A));
        return null;
};
```

- El paso siguiente fue localizar la función "FCKCommands.GetCommand" que es ejecutada cuando se presiona un botón de la barra icónica. Se modificó para agregar la funcionalidad de los botones nuevos correspondientes a SCORM. Una vez en la función modificamos el switch (A) agregando las opciones de: "RadioScorm", "PaginaSiguiente", "PaginaAnterior" y "CalcularNota", cada una de estas opciones deberá llamar a la función correspondiente según su funcionalidad.

  Todos los botones deberán abrir una ventana emergente, solicitando al usuario el ingreso de parámetros. Para ello debemos instanciar la función "FCKDialogCommand", definida por FCKeditor indicando como parámetros:

  1. Cual es el tipo de ventana a abrir
  2. Con que nombre debe buscar la clase FCKLang en el archivo es.js la descripción que se mostrara como título en la ventana emergente.
  3. Path que indica cual es el HTML que debe abrir como ventana emergente. En nuestro caso el HTML es fck_radioButtonSCORM.html, fck_buttonScormNota.html, fck_buttonSiguiente.html y fck_buttonAnterior.html respectivamente. Estas páginas fueron creadas por nosotros para soportar la comunicación LMS-SCORM.
  4. y 5. Dimensiones de la ventana que se muestra.

Ejemplo de la llamada a la función para el caso del "Radio button". Para los otros botones ver Anexo B.

La siguiente sentencia es la intancia de FCKDialogCommand que se agrego al case de la instrucción switch (A) de la función FCKCommands.GetCommand .

```
new
FCKDialogCommand('Radio',FCKLang.RadioButton,'dialog/fck_radioButtonSCORM.html',380,3
50);
```

Esta instruccion abre la ventana emergente correspondiente al "Radio Button" como se explicó anteriormente. Cuando la ventana se ciera por la accion del usuario en el editor HTML se puede observar el icono del "radio buton" al cual el usuario escribira al lado el texto.

Para el caso de un "Radio Button" que no soporta SCORM internamente el editor genera el siguiente tag en la página que se esta creando:

```
<p><input  type="radio" name="Repuesta 1" value="0" /> un navegador</p>
```

Para el caso de que soporte SCORM generara el siguiente tag en la página que se esta creando:

```
<p><input onclick="JavaScript:EvalMultipleChoiceItem(this,1)" type="radio"
name="Repuesta 1" value="0" /> un navegador</p>
```

En este ultimo caso genera en el evento onClick del "Radio Button" la llamada a las funciones para la registracion de la evaluacion del alumno.

Una vez finalizada la personalización del Editor debimos nuevamente compactar este archivo para seguir manteniendo el estándar del editor.

- **Archivo fckeditorcode_ie.js: otras modificaciones no vigentes**

La funcionalidad de los botones "PaginaSiguiente" y "PaginaAnterior", como se mencionó en la sección "Cambios realizados a la propuesta inicial", fue modificada respecto de la idea original.

Inicialmente para implementar la funcionalidad de los dos botones fue necesario crear tres funciones como se describen a continuación, donde XXX corresponde al nombre que nosotros queremos ponerle:

1. XXXCommand: setea el campo Name del tag a generar con el nombre indicado por nosotros.

   Ejemplo de los nombres de la funciones:
   FCKLinkPaginaSigScormCommand() y

   FCKLinkPaginaAntScormCommand()

2. XXXCommand.prototype.Execute: para ambos casos, esta función agregaba un tag "link" en la página que el usuario estaba creando. El tag redireccionaba dinámicamente a la página siguiente o anterior respectivamente.

3. XXXCommand.prototype.GetState: retornaba el valor cero para indicar el estado.

Por ejemplo para el botón "PaginaSiguiente" las funciones que se habían definido fueron:

```
var FCKLinkPaginaSigScormCommand=function(){
        this.Name='LinkPagina';
};


FCKLinkPaginaSigScormCommand.prototype.Execute=function(){
        FCK.InsertHtml  ('<a href="JavaScript:window.parent.NextPage()">Pagina
Siguiente');
};

FCKLinkPaginaSigScormCommand.prototype.GetState=function(){return 0;};
```

A continuación se muestra como la función GETCOMMAND llama internamente a las definidas por nosotros

```
FCKCommands.GetCommand=function(A){
        var B=FCKCommands.LoadedCommands[A];
        if (B) return B;
        switch (A){
        …..
```

//Botones correspondientes a la comunicación Sco-LMS

```
        case 'PaginaSiguiente':
                B=new FCKLinkPaginaSigScormCommand();
                break;
        case 'PaginaAnterior':
                B=new FCKLinkPaginaAntScormCommand();
                break;
```

//Fin botones nuevos correspondientes a la comunicación Sco-LMS

```
…..
…..

};
```

Esta funcionalidad no prosperó porque al insertar en el editor el tag *'<a href="JavaScript:window.parent.NextPage()">Pagina Siguiente'*, o el de página anterior, debíamos pasarle a la función NextPage o PreviewPage el "Número de Unidad" al que pertenece la página que se esta diseñando. Este dato es uno de los parámetros solicitados al momento de crear la página.

El problema era que ese dato estaba fuera del área del editor y no encontramos la forma de obtenerlo desde el archivo "fckeditorcode_ie.js". Desde el archivo solo podíamos obtener los tags definidos en el diseño de la página dentro del editor , es decir, que no teníamos forma de capturar el valor del campo "Número de Unidad". Ante esta situación decidimos redefinir las funciones como ya se describió.

También se habían definido dos botones más llamados "SaveScorm", "SaveScormInic". Para esos botones, creamos llamadas a funciones nuevas: FCKPreviewScormCommand() y FCKPreviewScormInicCommand() respectivamente. El código que generamos para estas funciones puede verse en el Anexo B.

Si bien estas funciones, finalmente quedaron fuera de la aplicación desarrollada, nos pareció importante destacarlas porque por un lado formaron parte del proceso de construcción y por otro, puede ser de utilidad para alguna funcionalidad futura, es decir, el conocimiento sobre como se pueden definir nuevas funcionalidades de otra forma a las planteadas actualmente.

Son formas diferentes de adaptar las funciones del editor. Darnos cuenta de cómo podíamos en su momento desarrollarlo fue bastante costoso y no fue directo, sino que hubo bastante análisis, depuración de código y desarrollo.

- **Archivo es.js:**

Nos situamos al final del archivo y definimos el vocabulario para los botones SCORM:

En nuestro caso deberíamos agregar:

```
RadioScorm:        "Radio Button",
CalcularNota:      "Calcular Nota",
```

PaginaSiguiente: "Página Siguiente",

PaginaAnterior: "Página Anterior",

➢ **Ventanas de Dialogo:**

En la carpeta "dialog" de FCKeditor se encuentran definidas todas las ventanas de diálogos. En nuestro caso debemos modificar la ventana correspondiente a:

1. **Radio Button:** esta ventana la hemos modificado para que el usuario indique si el radio button a agregar tendrá funcionalidad SCORM o no. En el caso de tener funcionalidad SCORM pediremos que el usuario ingrese el número de pregunta como describimos más arriba.

Para poder hacer esto debimos crear una nueva página HTML la cual llamamos "fck_radioButtonSCORM.html" y la guardamos en "\FCKeditor\editor\dialog\". La llamada a la página desde el icono del editor se visualiza en la figura 31.

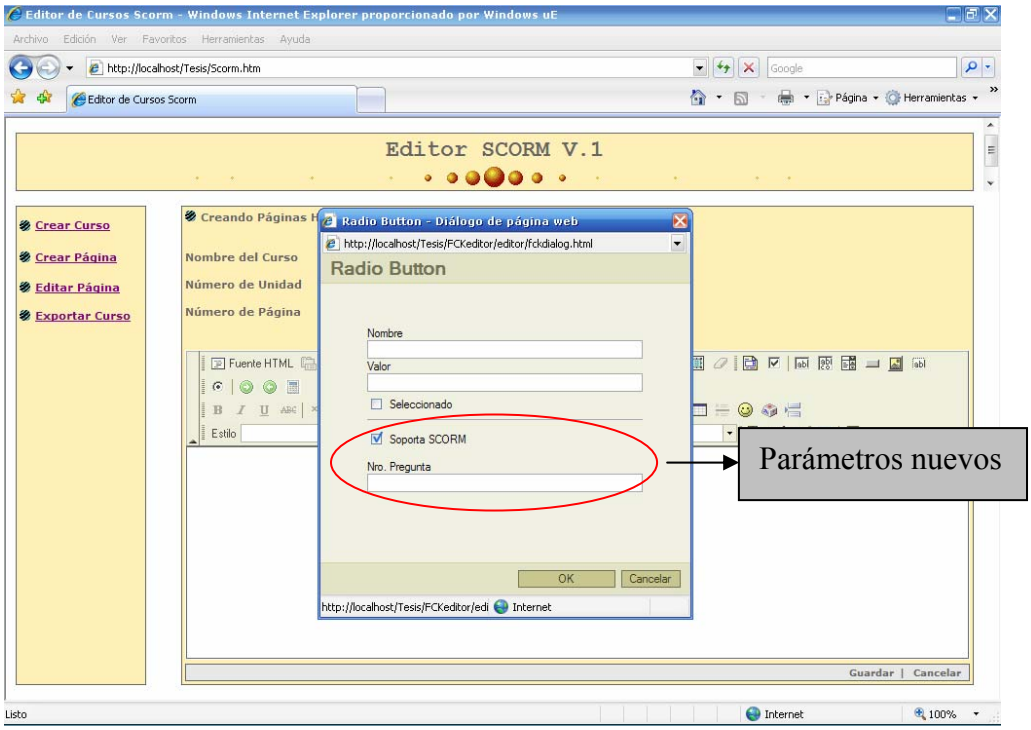


Figura 31. Ventana de "Radio button"

Esta página es una copia de la página "fck_radiobutton.html" que viene con los fuentes del editor y a la cual agregamos los campos para manejo de SCORM.

Una vez creada la página la editamos con algún editor de texto, nos situamos en la línea 143 y agregamos las siguientes líneas siguiendo la estructura de la página original:

....
....

```
<tr>
        <td>
```

```
                <input name="Scorm" type="checkbox" id="Scorm"
onChange="activarNroPregunta()"                checked>
                <label>Soporta SCORM</label>
                </td>
        </tr>
        <tr>
                <td> </td>
        </tr>
        <tr>
                <td><span fcklang="DlgCheckboxPregunta">Nro pregunta</span></td>
        </tr>
        <tr>
                <td>
                <input name="txtNroPreg" type="text" id="txtNroPreg" style="WIDTH:
100%" size="20"                value="" >
                </td>
        </tr>
….
….
```

Luego nos situamos en la función JavaScripts "function Ok()", y la reemplazamos por esta otra:

```
function Ok(){

//Nuevo para funcionalidad SCORM
        if ((GetE('Scorm').checked) && (!validarNro())){
                return false;
        }

        if ( !oActiveEl ) {
                oActiveEl = oEditor.FCK.EditorDocument.createElement( 'INPUT' ) ;
                oActiveEl.type = 'radio' ;
                oActiveEl = oEditor.FCK.InsertElementAndGetIt( oActiveEl ) ;
        }

        if ( GetE('txtName').value.length > 0 )
                oActiveEl.name = GetE('txtName').value ;

        if ( oEditor.FCKBrowserInfo.IsIE )
                oActiveEl.value = GetE('txtValue').value ;
        else
                SetAttribute( oActiveEl, 'value', GetE('txtValue').value ) ;

        var bIsChecked = GetE('txtSelected').checked ;
        SetAttribute( oActiveEl, 'checked', bIsChecked ? 'checked' : null ) ;    // For Firefox
        oActiveEl.checked = bIsChecked ;

//Nuevo para funcionalidad SCORM

        if (GetE('Scorm').checked) {

        SetAttribute( oActiveEl, 'onClick', "JavaScript:EvalMultipleChoiceItem(this,"
+ GetE('txtNroPreg').value+ ")" ) ;          // For Firefox

        return true ;
}
```

También se debe agregar las siguientes funciones JavaScript para validaciones:

```
function activarNroPregunta (){
	if (GetE('Scorm').checked){
		SetAttribute( GetE('txtNroPreg'), 'readonly', false) ;
	}else{
		SetAttribute( GetE('txtNroPreg'), 'value', ' ' ) ;
		SetAttribute( GetE('txtNroPreg'), 'readonly', true ) ;
	}
}

function validarNro (){
	if (GetE('txtNroPreg').value == ""){
		alert("El Nro. de Pregunta es un dato obligatorio");
		return;
	}

	if (isNaN(GetE('txtNroPreg').value)){
		alert("El Nro. de Pregunta debe ser un nro");
		return;
	}

	return true;
}
```

2. **Calcular Nota:** debimos crear una nueva página HTML. La guardamos con el nombre "fck_buttonScormNota.html" en el mismo path que la página del punto 1.

Para poder crear esta página se tomo como base la página original que se abre al seleccionar el "Botón" de la barra icónica del editor FCKeditor.

Esta página no podía ser utilizada para definir el botón "Calcular Nota" ya que permite setear la propiedad "type" del botón seleccionando entre el tipo "button" y "submit", y "Calcular Nota" debe ser únicamente de tipo "button".

También debimos modificar la función JavaScript "Ok", ya que debíamos agregar las llamadas a las funciones JavaScript de la API LMS para la comunicación LMS-SCORM. A continuación mostramos como quedo el código fuente de la función "OK"

```
function Ok(){
	if ( !oActiveEl ) {
		//Crear el boton
		oActiveEl = oEditor.FCK.EditorDocument.createElement( 'INPUT' ) ;
		oActiveEl.type = "button";
		oActiveEl = oEditor.FCK.InsertElementAndGetIt( oActiveEl ) ;
	}

	oActiveEl.name = GetE('txtName').value ;
	SetAttribute( oActiveEl, 'value', GetE('txtValue').value ) ;
	SetAttribute( oActiveEl, 'onClick', "ShowSCORMStatus()" ) ;	// For Firefox

	return true ;
}
```

La figura 32 muestra la página cuando es invocada desde el menú "Calcular Nota"

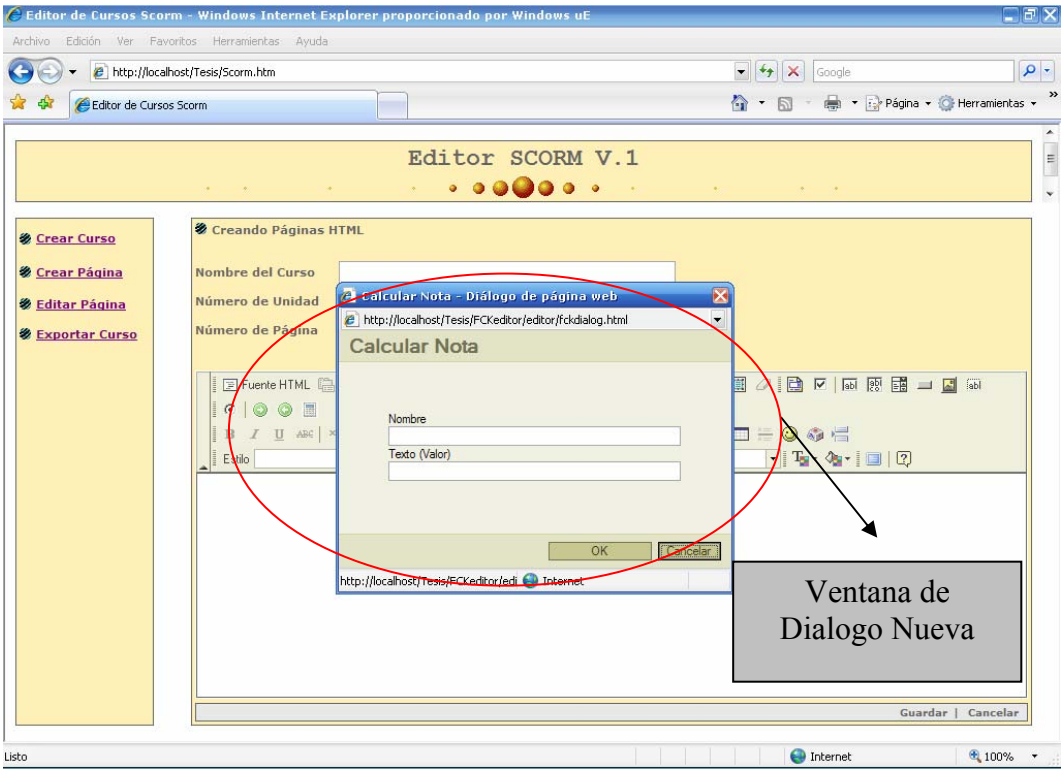


Figura 32. Ventana Calcular Nota

3.3.6 Skins. Definición de imágenes para la barra SCORM

FCKeditor nos brinda 3 skins como mencionamos en la unidad 3, pero para poder realizar una personalización acorde a nuestro tema decidimos definir un nuevo skin para SCORM, de esta forma realizamos una separación entre lo proporcionado por el editor y las funcionalidades nuevas del editor para SCORM.

Para realizar esto creamos una carpeta "scorm" en "\FCKeditor\editor\skins\" siguiendo la estructura de los demás skins.

Dentro de la carpeta "toolbar", colocamos los botones de nuestra barra icónica "SCORM". Los botones deben ser imágenes con extensión ".gif". También se debe tener en cuenta que luego las imágenes serán invocadas por su nombre desde el archivo "fckeditorcode_ie.js"

Una vez realizado los pasos descriptos debería quedarnos lo siguiente y las imágenes estarían listas para ser invocadas por nuestro editor:

\FCKeditor\editor\skins\scorm\
images (copiar las mismas imágenes que la carpeta default)
toolbar (copiar las mismas imágenes que la carpeta default y agregar las nuevas imágenes, como por ejemplo paginaSiguiente.gif)
fck_contextmenu.css
fck_dialog.css
fck_editor.css
fck_strip.gif

3.4 Ejemplo de un curso SCORM con FCKScorm

Como caso práctico de esta tesis se tomó como base el curso UML del proyecto CDTC mencionado anteriormente y se lo construyó utilizando la herramienta implementada.

El contenido del curso original se distribuye en cuatro lecciones: Introducción, Conceptos Iniciales, Diagramas de Estructuras y Diagramas de Comportamiento. En la figura 33 es posible ver la pantalla inicial con el índice del curso a partir del cual se puede acceder a cada lección y a la evaluación final.

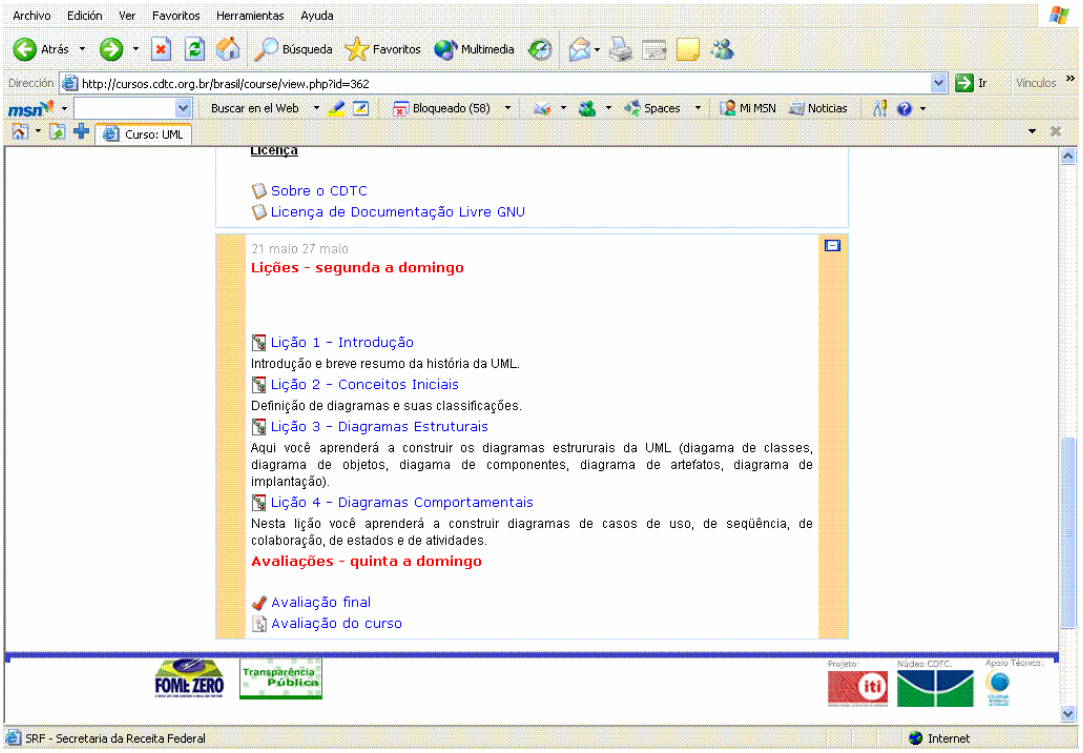


Figura 33. Estructura del curso original UML – ITI Brasil

Partiendo de este curso y utilizando la aplicación desarrollada, mostraremos que es posible reestructurarlo de manera de seguir las normas del estándar SCORM.

La restructuración consistirá en tomar las lecciones del curso existente y crear para cada una, dos Objetos de Aprendizaje (OA) auto-contenidos e independientes del contexto. Uno de los OA corresponderá al contenido propiamente dicho de la lección y el otro a la evaluación de la misma.

Las funciones de comunicación del curso con el LMS son incorporadas en forma automática a través del editor.

Esta forma de construir los cursos permite llegar al mismo producto final que el original, sin agregarle dificultad a la etapa de construcción, pues el usuario "creador" no depende de gente con conocimientos de programación para incorporales un mejor seguimiento de la actividad del alumno ya que es posible evaluar cómo fue su actividad, el tiempo total de navegación del curso, la página en que abandonó la navegación, el estado de navegación para cada lección (completa

o incompleta) y toda la información relativa a las evaluaciones. Al mismo tiempo que se obtiene material reusable y fácil de mantener.

Como se explicó, todos los cursos deben dividirse en unidades. En este caso cada lección representará una unidad. Dentro de cada unidad se debe dividir en dos secciones, la sección Teórica y la sección Práctica, con sus respectivas páginas.

Cada página del curso es creada o editada por el editor FCKeditor SCORM V.1 con el fin de que puedan ejecutar las funciones correspondientes propuestas por ADL, en el archivo SCORMGenericLogic.js para la comunicación con el LMS.

Una vez que el usuario finaliza la creación del curso con el editor deberá proceder a la creación del paquete SCORM y luego a su navegación con un LMS que soporte SCORM.

3.4.1 Pasos para el armado del curso con FCKScorm

En primer lugar se debe acceder a la aplicación, es decir, escribir en el browser http://localhost/Tesis/Scorm.htm.

En nuestro caso optamos por crear las páginas desde cero, copiando solamente el contenido de cada página original. Queremos dejar en claro que no es la única forma de hacerlo podríamos también editar cada una de las páginas y modificarlas según las necesidades.

1. **"Crear Curso":** se visualizará la ventana que se muestra en la figura 33, solicitándonos algunos parámetros para crear en el servidor la estructura del curso. Los parámetros solicitados son:
    - Nombre del curso. En nuestro caso UML.
    - Cantidad de unidades que forman nuestro curso. En nuestro caso 4.

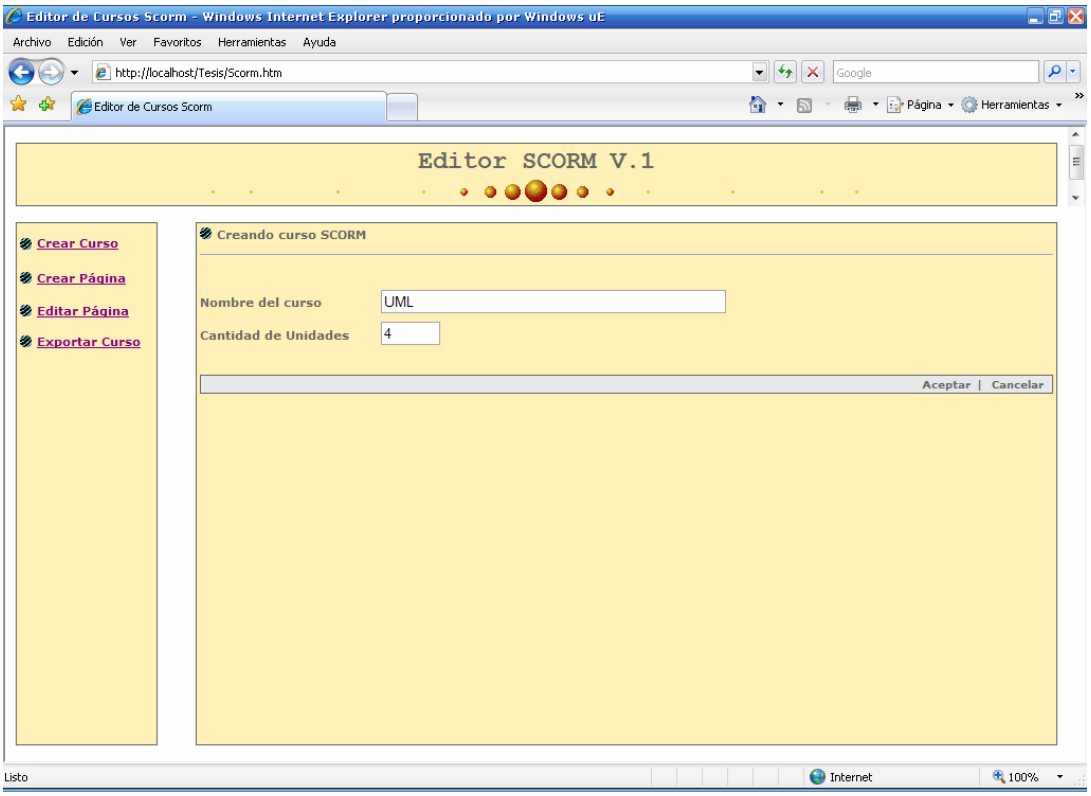


Figura 33. Pantalla de Crear Curso

Al presionar el botón "Aceptar", automáticamente se crea en el servidor la estructura del curso como se muestra es la figura 33.a:

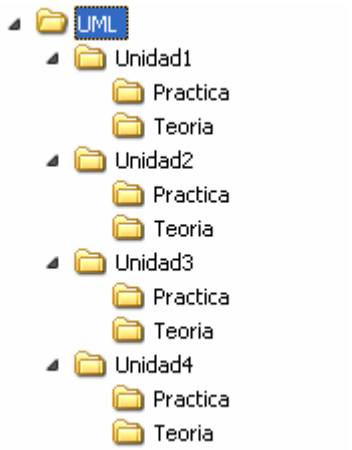


Figura 33.a. Estructura del curso UML

Al mismo tiempo que se crea la estructura del curso, la aplicación copia automáticamente en cada una de las carpetas "Teoría" de cada unidad el archivo SCORMGenericLogic.js y en las carpetas "Practicas" copia los archivos SCORMGenericLogic.js y SCORMObjectiveLogic.js, que contienen las funciones propuestas por ADL para la comunicación con el LMS y para el manejo de objetivos, respectivamente.

Al generar las carpetas Teoría y Práctica, en cada unidad, la aplicación copia los archivos dummypage.htm, page0.htm y sco.htm desde una ubicación fija en el servidor. Como ya se explicó, todos estos archivos son necesarios tanto para

formar un SCO como al momento de generar el paquete SCORM y visualizarlo con algún LMS que lo interprete.

Si el proceso de creación del curso fue exitoso se le presenta al usuario una pantalla donde se le indica *"Se ha creado satisfactoriamente el curso UML"*, en caso contrario se indica *"No se ha podido crear el curso UML"*.

2. **"Crear Página":** una vez creada la estructura del curso debemos crear las páginas de contenidos y evaluaciones para cada unidad. Al seleccionar desde el menú esta opción se presentará al usuario una pantalla solicitando algunos parámetros y luego se abrirá el editor FCKeditor adaptado para comenzar a crear cada una de las páginas.

   Los parámetros solicitados son:

   - Nombre del curso. En nuestro caso UML.
   - Número de unidad al que pertenece la página. En nuestro caso pertenece a la Unidad 1
   - Número de página dentro de la unidad. En nuestro caso corresponde a la página 1 de la Unidad 1.
   - Seleccionar el tipo de página a crear, es decir, si es una página teórica o de evaluación.

   A continuación detallamos por separado la creación de una página teórica y una página de evaluación.

2.1. Crear página teórica

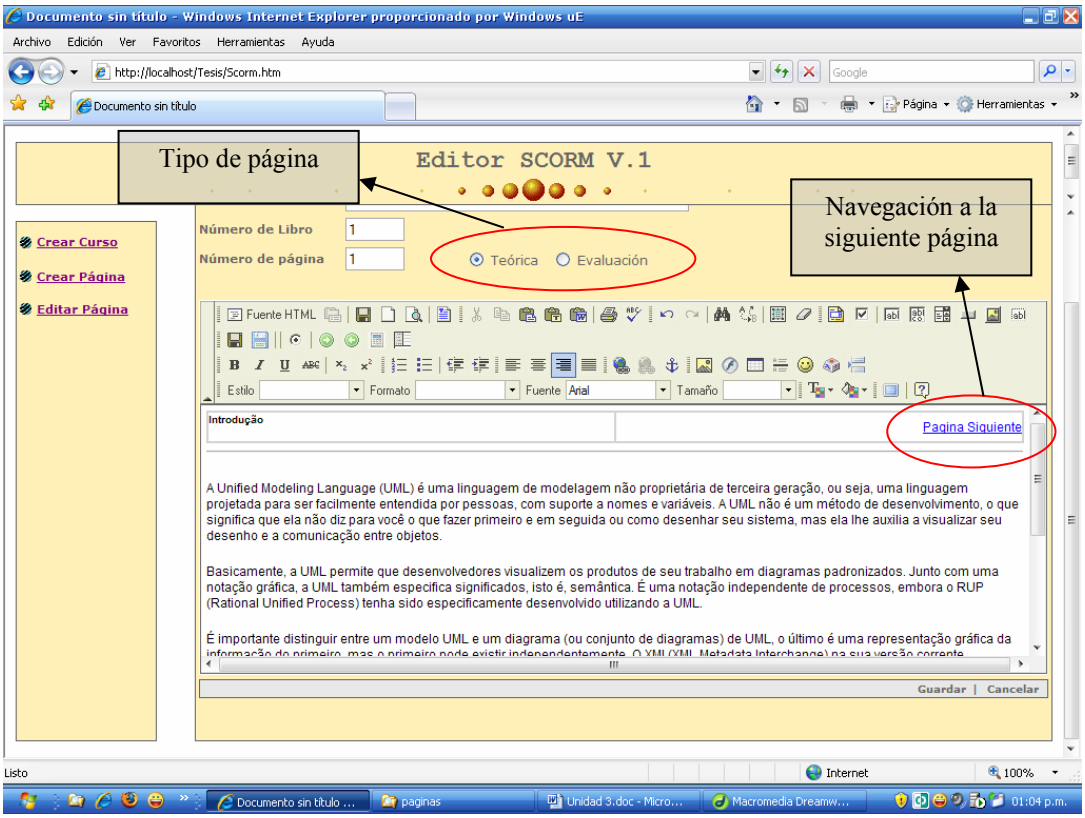


Figura 34. Crear Página

Como siguiente paso debemos crear, es decir, rediseñar las páginas teóricas de cada unidad del curso UML Brasil con nuestro editor para permitir la navegación entre las páginas y así poder registrar el seguimiento del alumno, como se muestra en la figura 34.

Pasos a seguir:

I. Crear una tabla con dos columnas, en una columna escribir el texto "Introdução" y en la otra columna agregamos con el editor la navegación a la página siguiente. Para esto seleccionamos de la barra de herramientas la opción "Página Siguiente". Se abrirá la pantalla para la inserción del link a la página siguiente, como se visualiza en la figura 35, solicitando lo parámetros: descripción del link, en nuestro caso "Página Siguiente" y número de unidad al que pertenece la página, en este caso la número 1.

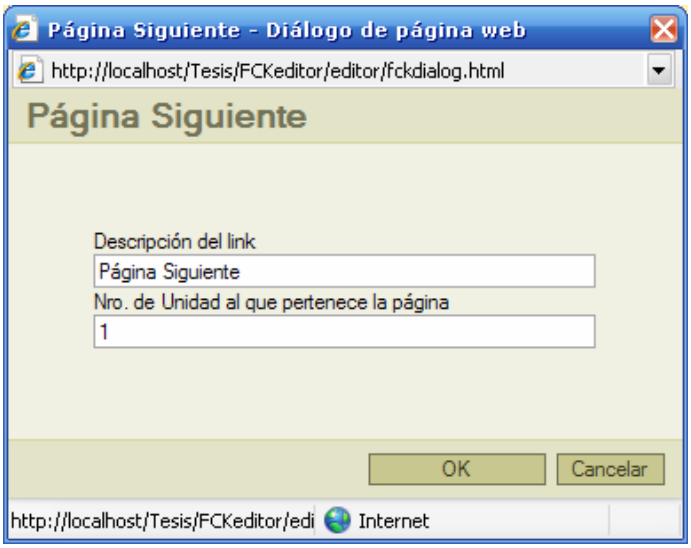


Figura 35. Pantalla correspondiente al botón "PáginaSiguiente" del editor adaptado.

Al presionar el botón de "OK" automáticamente agrega en la posición donde estábamos parados al momento de seleccionar la opción, el link correspondiente con la llamada a la función JavaScript detallada anteriormente.

II. Escribimos el texto correspondiente a la página.

III. Repetimos el paso I. La primer columna la dejamos vacia y en la segunda columna agregamos con el editor la navegación a la página siguiente.

Tener en cuenta que la página descripta en este ejemplo es la primera de la unidad por lo tanto solamente tiene navegación a la página siguiente, en las páginas sucesivas deberíamos agregar también la navegación a la página anterior con la opción del editor "Página Anterior". Excepto en la última página que solamente tendrá navegación a la página anterior.

Debemos aclarar que no es la única forma de rearmar las páginas originales pues el usuario "creador" podría diseñarla con otro criterio. Lo

más importante es que a cada una de las páginas del curso original le agreguemos los botones "PaginaSiguiente" y "PaginaAnterior" para lograr obtener luego la navegación del alumno por el curso.

Una vez creada la página, seleccionamos "Guardar". La página será creada en el servidor con el nombre "unidad1_pag1.html" en el path "/UML/unidad1/teoria/".

Si la creación fue exitosa se visualizará el resultado final de la página creada. En caso contrario se presentará al usuario el texto "La página no se pudo crear."

2.2. Crear página de evaluación

Para rediseñar las páginas de evaluación de cada unidad del curso UML con nuestro editor debemos armar un cuestionario con múltiples opciones y al pie de la página agregar el botón de "Calcular Nota".

Pasos a seguir:

I. Crear una tabla con 2 columnas, en una columna escribir el texto "Introdução".

II. Escribimos el texto correspondiente a la página de evaluación de forma tal que quede cada pregunta con sus posibles respuestas. Cada respuesta deberá ser agregada a la página como un "Radio Button". Para esto seleccionamos de la barra de herramientas la opción "Radio Button". Se abrirá la pantalla para la inserción de "Radio Buton", como se visualiza en la figura 36, solicitando los parámetros: nombre del botón, en nuestro caso "Respuesta1"; valor, en nuestro caso las respuestas correctas tendrán un valor de 100 y las incorrectas de 0; el campo "Seleccionado" no debe quedar tildado; el campo "Soporta SCORM" debe quedar tildado y agregamos el valor del último parámetro "Nro. Pregunta" que en este caso es la 1.

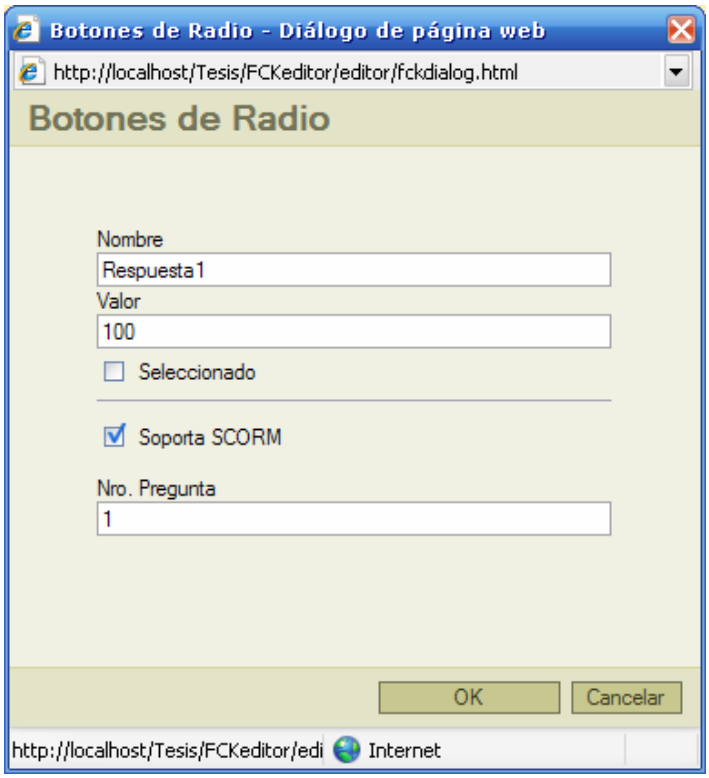


Figura 36. Pantalla correspondiente al botón "Radio button" del editor adaptado.

Al presionar el botón de "OK" automáticamente agrega en la posición donde estábamos parados al momento de seleccionar la opción el "Radio Button" correspondiente con la llamada a la función JavaScript detallada anteriormente. Hasta acá lo que se hizo fue generar el tag correspondiente al "Radio Buton". A continuación nos posicionamos al lado del "Radio Button" que se visualiza en la página que estamos creando y escribimos la respuesta correspondiente a ese radio.

III. Una vez finalizado el cuestionario agregamos el botón "Calcular Nota" al final del cuestionario. Para esto seleccionamos de la barra de herramientas la opción "Calcular Nota". Se abrirá la pantalla para la inserción del botón, como se visualiza en la figura 37, solicitando los parámetros: nombre, es el nombre que se le asigna a la propiedad name del botón, en nuestro caso "Calcular" y texto, es la descripción que se visualiza en el botón, en nuestro caso "Calcular Nota".

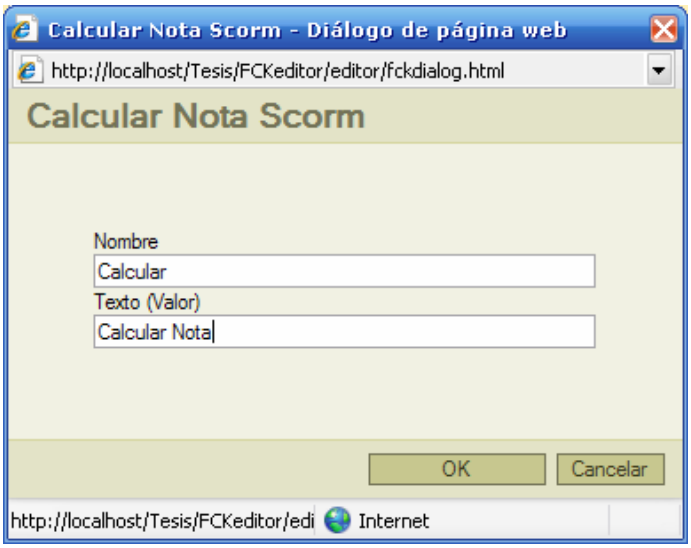


Figura 37. Pantalla correspondiente al botón "Calcular Nota" del editor adaptado.

Al presionar el botón de "OK" automáticamente agrega en la posición donde estábamos parados al momento de seleccionar la opción, el botón "Calcular Nota" correspondiente con la llamada a la función JavaScript detallada anteriormente.

Una vez creada la página, seleccionamos "Guardar". La página será creada en el servidor con el nombre "evaluacion1.html" en el path "/UML/unidad1/practicas/".

Si la creación fue exitosa se visualizará el resultado final de la página creada, como muestra en la figura 38. En caso contrario se presentará al usuario el texto "La página no se pudo crear."

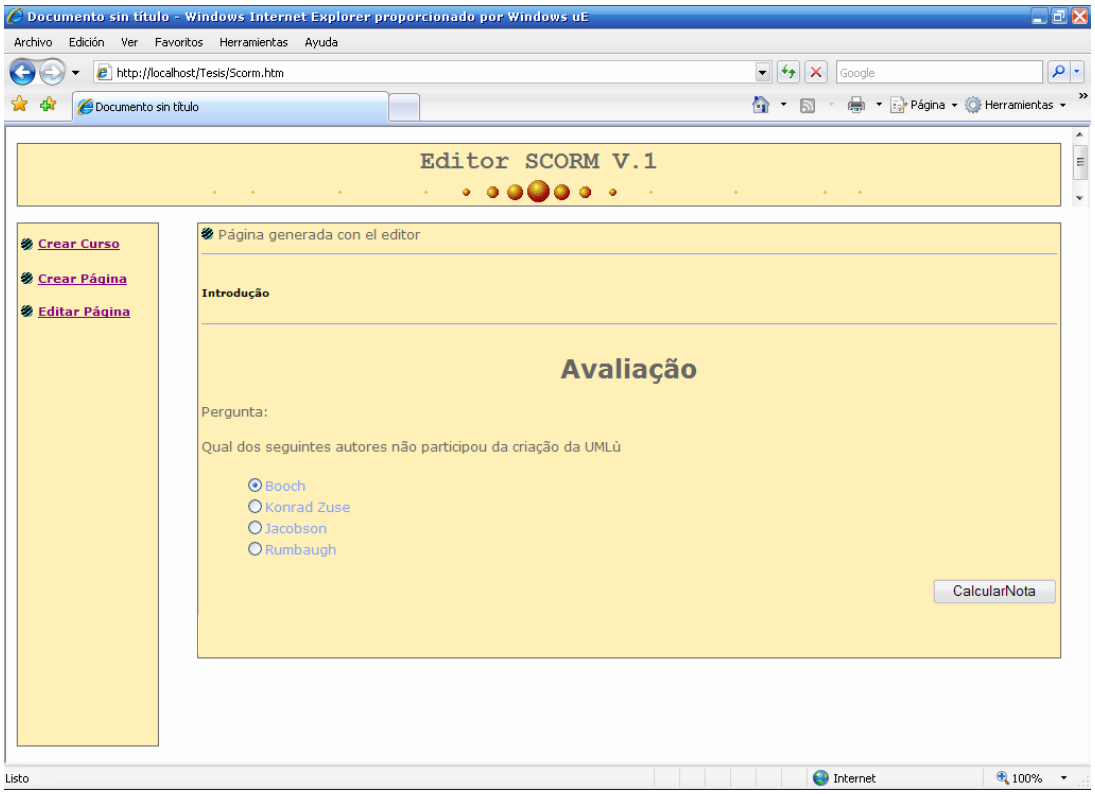


Figura 38. Visualización de una página creada con "FCKScorm"

3. **"Editar una página":** esta opción permite abrir páginas existentes en la máquina local como se explicó anteriormente. Al seleccionar la opción, se abre una pantalla solicitando que el usuario seleccione la página a editar ingresando la ubicación (path), como muestra la figura 39. Una vez seleccionada deberá presionar el botón "Cargar Página" y automáticamente se visualizará la página seleccionada en una pantalla con la misma funcionalidad que en la opción "Crear Página", pero en este caso el contenido de la página seleccionada se podrá visualizar en el editor FCKeditor para su edición. A partir de acá el tratamiento con esta página es el mismo que para la opción "Crear Página", es decir, que a partir de este punto el usuario deberá repetir los pasos anteriormente detallados.

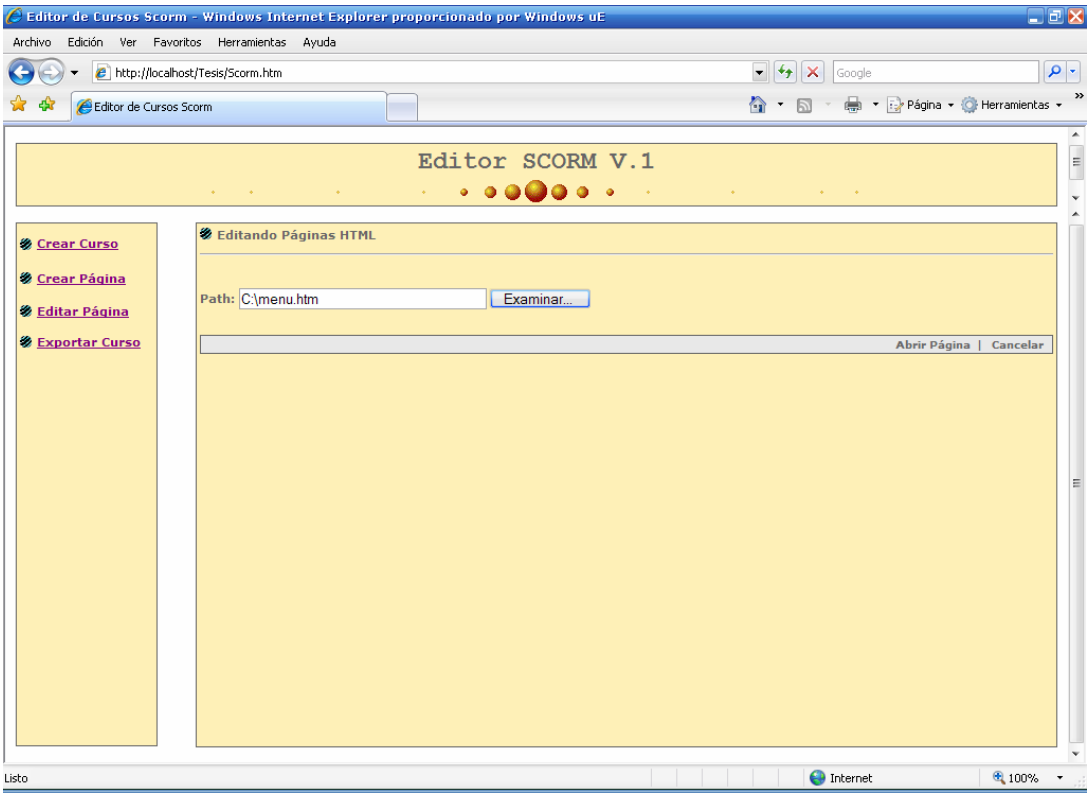


Figura 39. Pantalla de edición de una página.

4. **"Exportar Curso":** esta opción permite al usuario obtener en su maquina algún proyecto creado en el servidor. Al seleccionar la opción se abre una pantalla solicitando: el nombre del curso, en nuestro caso UML; la cantidad de unidades, en nuestro caso 4; y la ubicación (path) donde quiere guardar el proyecto en su maquina. Al aceptar los datos ingresados la aplicación genera un archivo ".zip", es decir, un archivo comprimido con el nombre del curso. Este archivo contiene toda la estructura del curso generado por el usuario y el formato que es usado en una herramienta de creación de paquetes SCORM como el Reload Editor.

Con los pasos descriptos hasta ahora el usuario podrá crear sus cursos SCORM. Luego deberá crear el paquete SCORM utilizando Reload Editor.

3.4.2 Armado del paquete SCORM con Reload Editor

Una vez creadas las páginas html con el editor extendido "FCKScorm", se utilizó la herramienta Reload Editor para construir el paquete SCORM.

Al abrir Reload Editor seleccionamos la opción de "Crear Paquete SCORM" en el cual seleccionamos el curso UML que hemos creado y automáticamente se presentará una pantalla con la estructura del mismo como se visualiza en la figura 40.

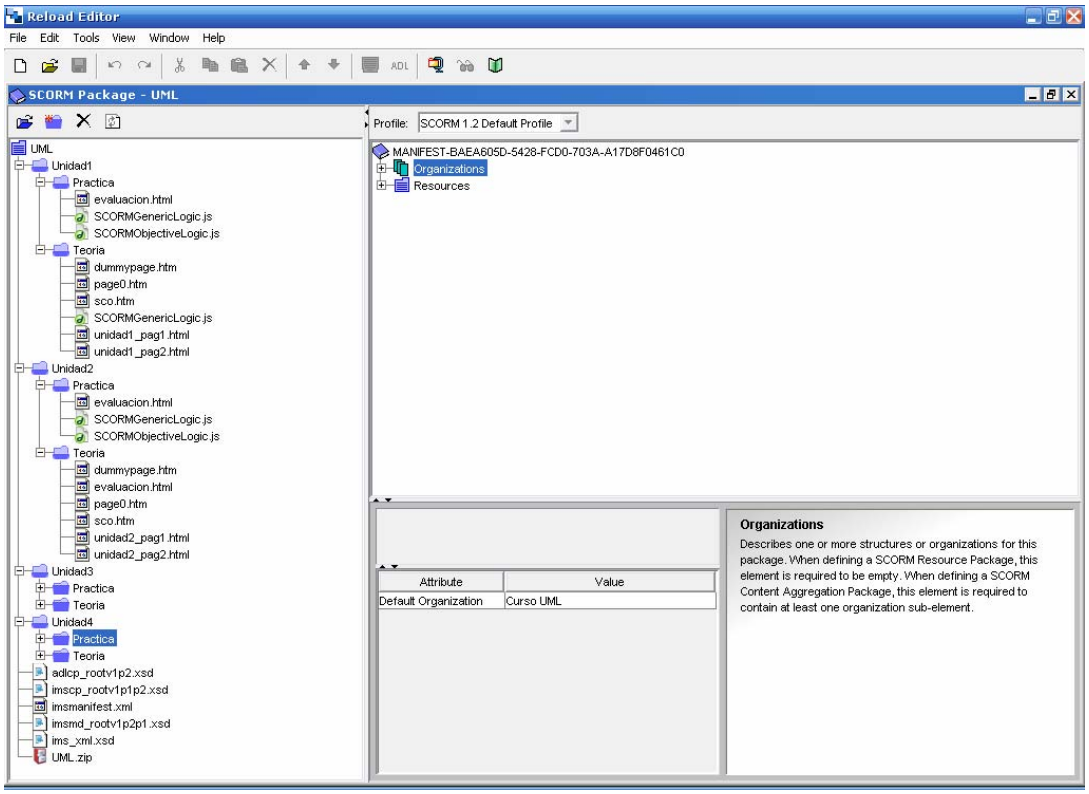


Figura 40. Estructura del curso UML visualizado con Reload Editor

En el margen izquierdo podemos ver la distribución de los html y javaScripts creados como se muestra en la figura 40.a:

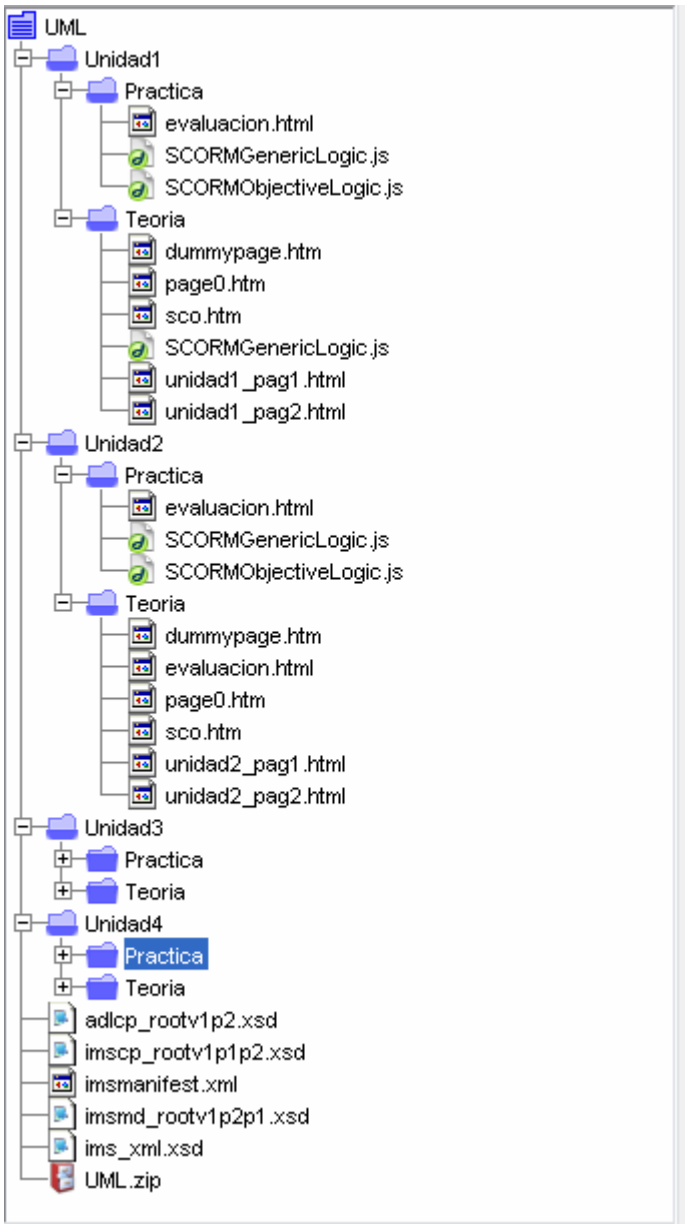


Figura 40.a. Distribucion del curso UML

En el margen derecho debemos crear las organizaciones para nuestro curso. Este curso contiene una sola organización para el contenido. Esta organización está dividida en cuatro unidades: Introdução, Conceitos Iniciais, Diagramas Estruturais y Diagramas Comportamentais. Luego dentro de cada sección tenemos Lição y Avaliação con su respectivo acceso a cada módulo. Como se muestra en la figura 41.b.

Figura 40.b. Organizaciones definidas para el curso UML

En el margen derecho en la parte inferior podemos ver los recursos utilizados. Para cada unidad tenemos el sco correspondiente a Introdução, Conceitos Iniciais, Diagramas Estruturais y Diagramas Comportamentais, como así también los sco's correspondiente a las Avaliaçãos. Cada uno de ellos con sus assets, como se visualiza en la figura 41.c.

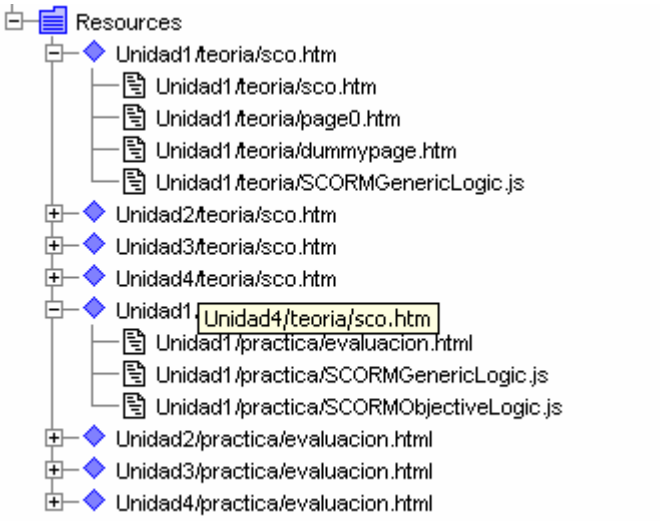
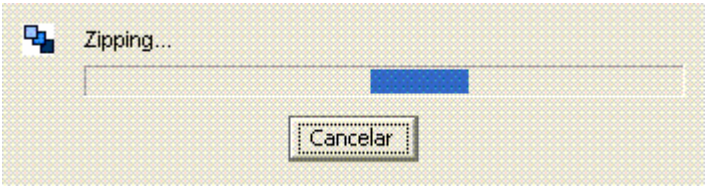


Figura 40.c. Recursos definidas para el curso UML

Una vez finalizado el proceso de definición de las organizaciones y los recursos, utilizamos la función "Zip Content Package" para crear el paquete SCORM del curso.



3.4.3 Visualización del curso SCORM con Moodle

Subir el paquete SCORM

Como último paso subamos el paquete SCORM creado al Moodle para mostrar cómo el curso UML puede navegarse en un LMS que soporte 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, como muestra la figura 41.

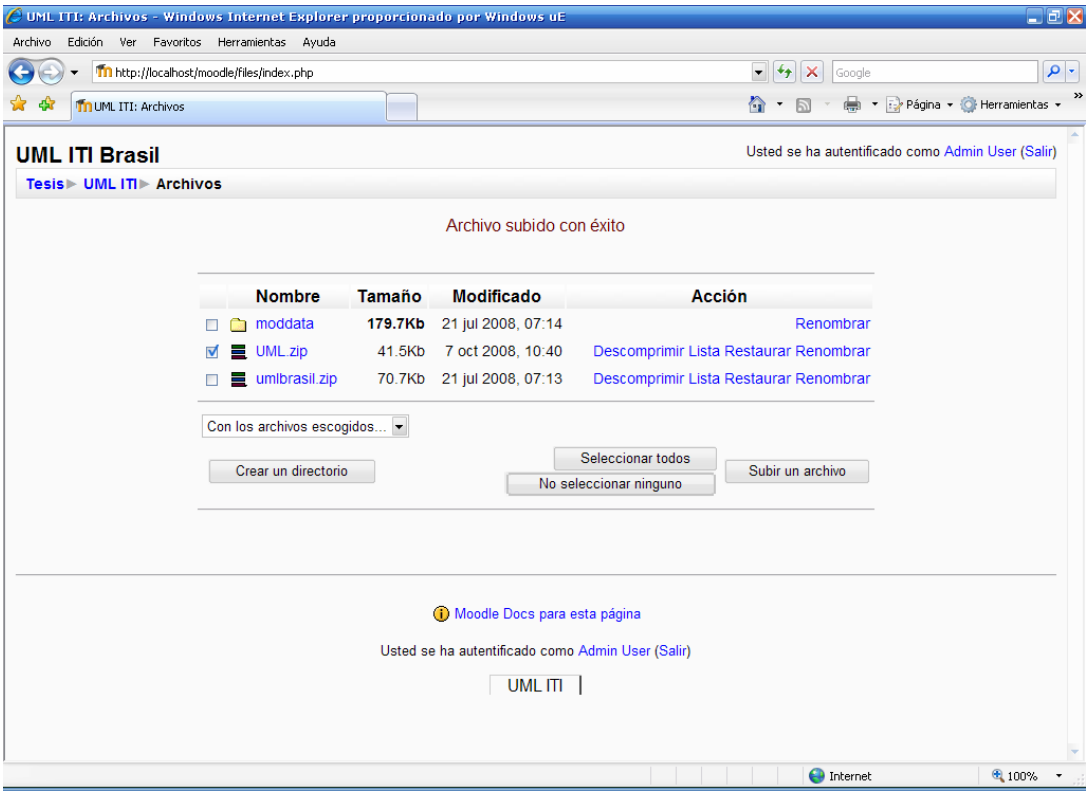


Figura 41. Lista de archivos subidos a Moodle

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, como se muestra en la figura 42.

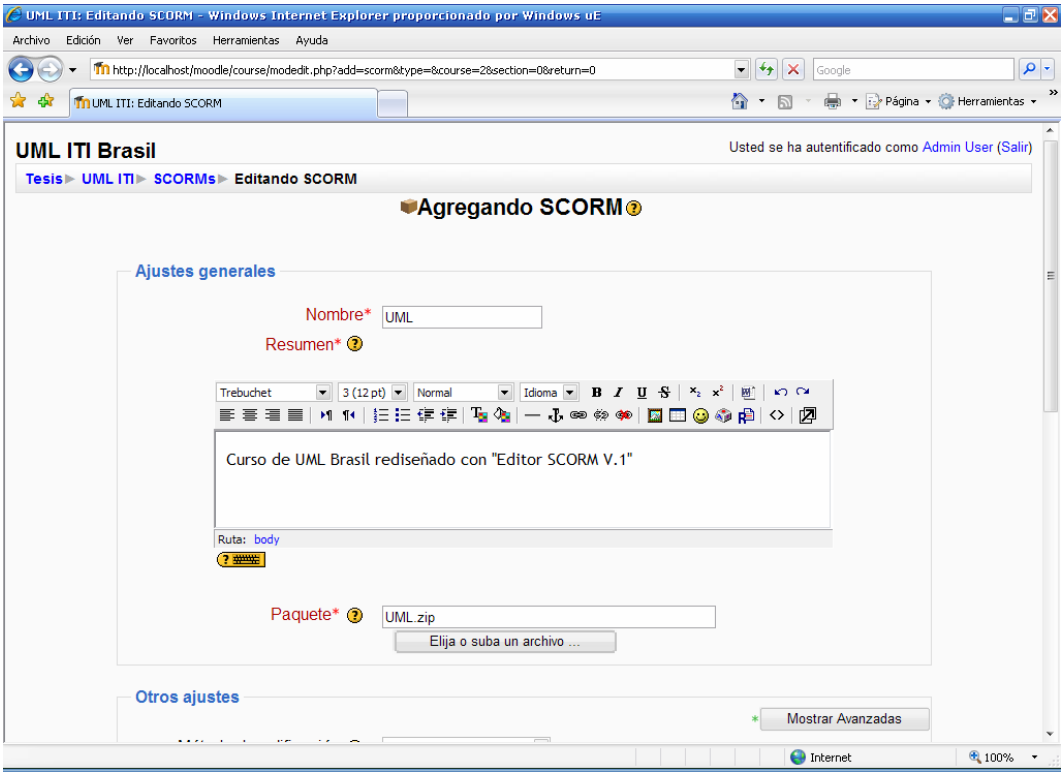


Figura 42. Pantalla para agregar un curso SCORM a Moodle

Una vez finalizada la carga de los datos obligatorios para el curso debemos presionar el botón "Guardar Cambios y Mostrar".

Mostremos cómo se ve nuestro curso en Moodle

Automáticamente al haber presionado el botón "Guardar Cambios y Mostrar" se nos presenta la pantalla con la estructura de nuestro curso UML, como se visualiza en la figura 43. En esta pantalla se presenta el índice del curso y a partir de acá presionando en el botón "Entrar" podremos comenzar a navegarlo.

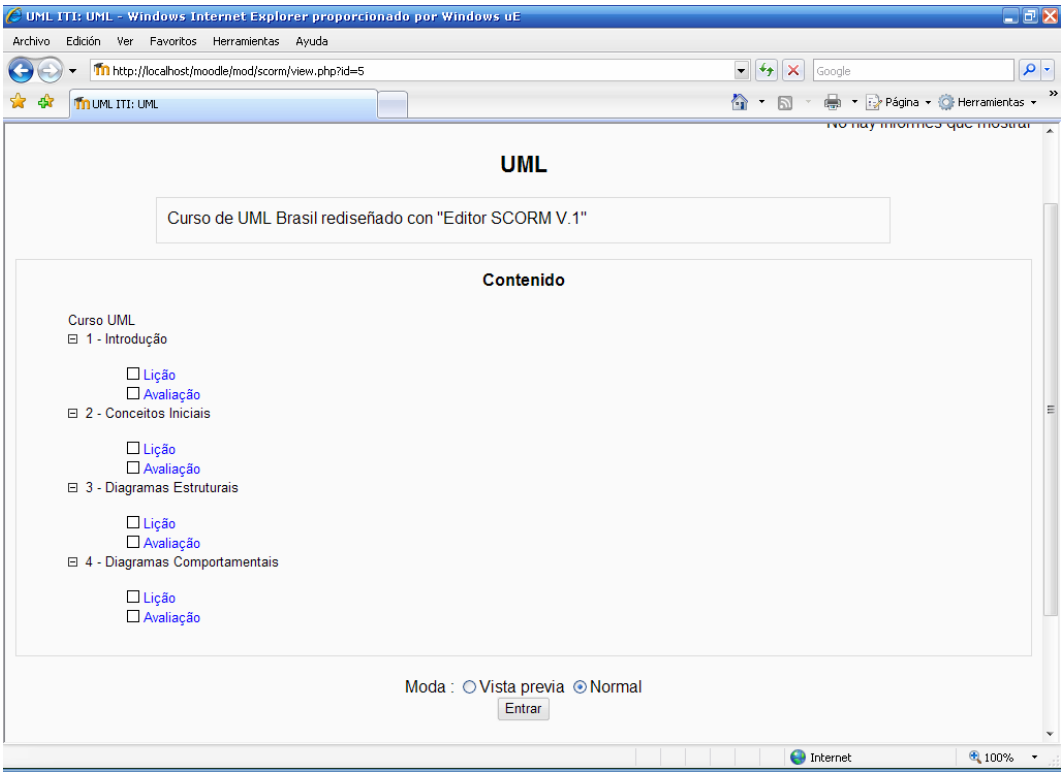


Figura 43. Visualización del curso UML en Moodle

Seleccionamos por ejemplo la unidad "Introdução" y luego la opción "Lição" y se visualizara la pagina mostrada en la figura 43.

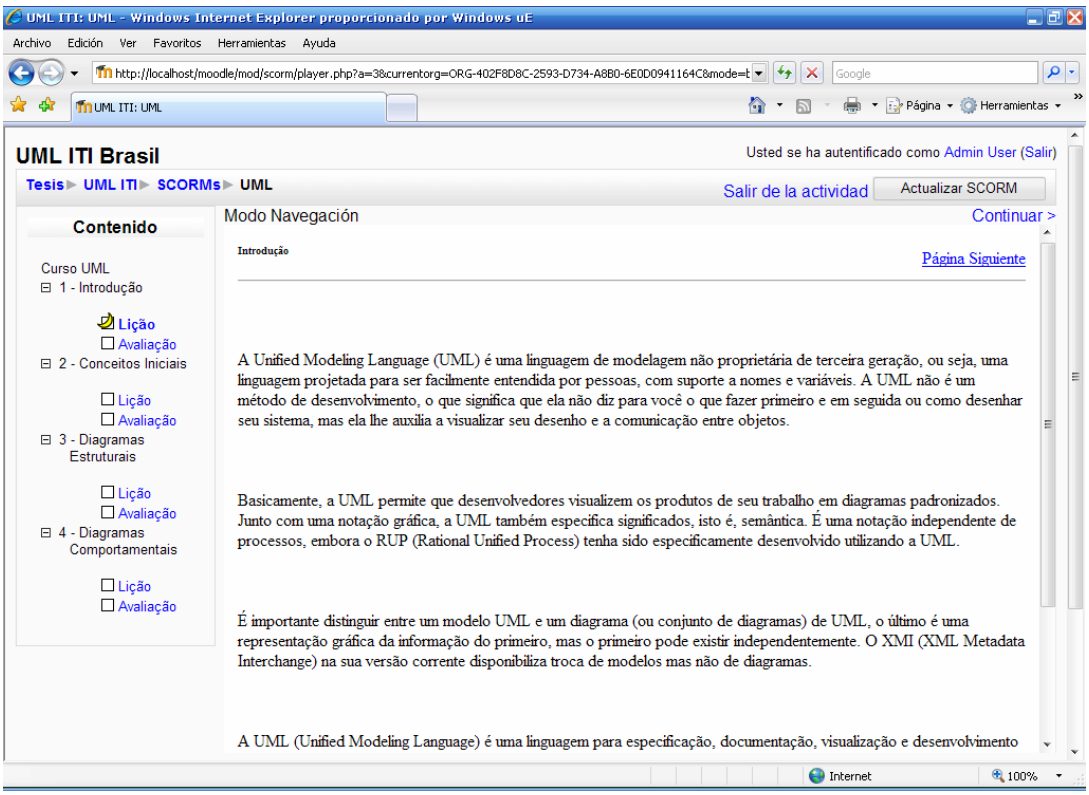


Figura 44. Navegación de la unidad Introdução - Lição en Moodle.

Navegamos la "Avaliacao" y se presentara la evaluación como se muestra en la figura 45.

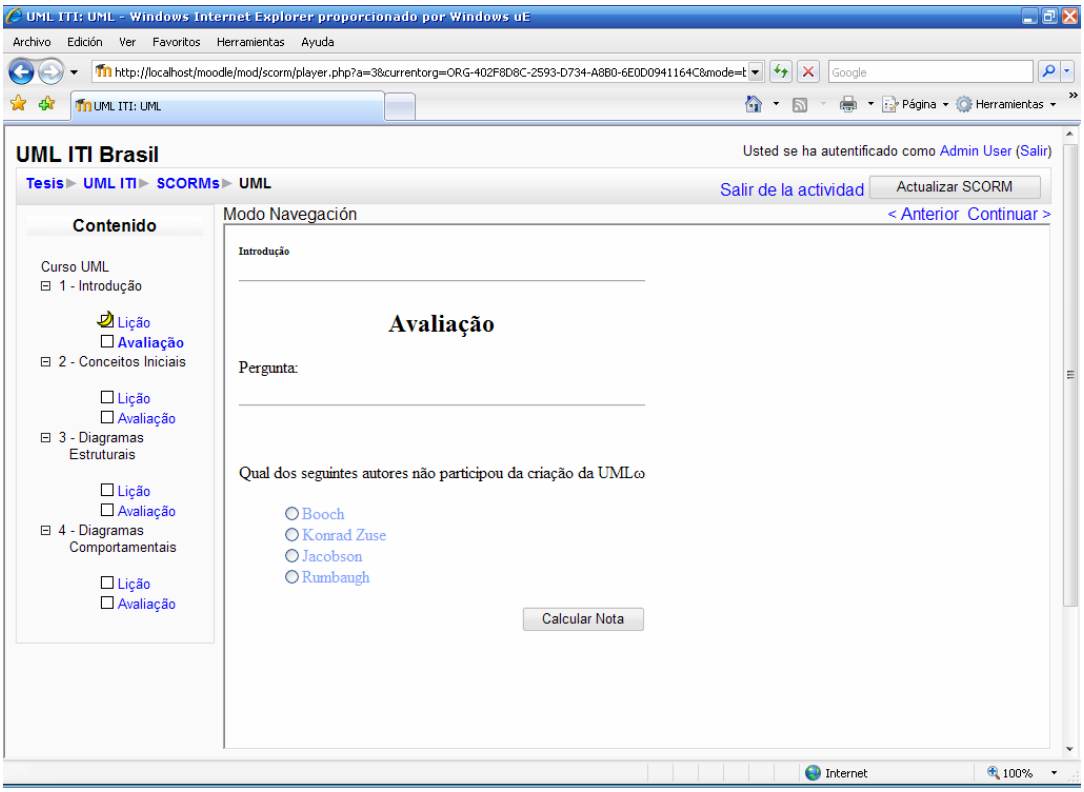


Figura 45. Navegación de la unidad Introdução - Avalaição en Moodle.

Si salimos del curso e intentamos navegarlo nuevamente se presenta el índice con algunos cambios como se observa en la figura 46. Vemos que se indica con diferentes iconos el estado de navegación del mismo. Por ejemplo en la unidad 1 "Introdução - Lição" se muestra una luna amarilla, esto significa que se navegó toda la opción "Introdução - Lição" propuesta. En cambio en la opción "Introdução - Avalaição" se muestra un icono de una flecha con lo cual se indica que se comenzó la navegación pero no se culminó.

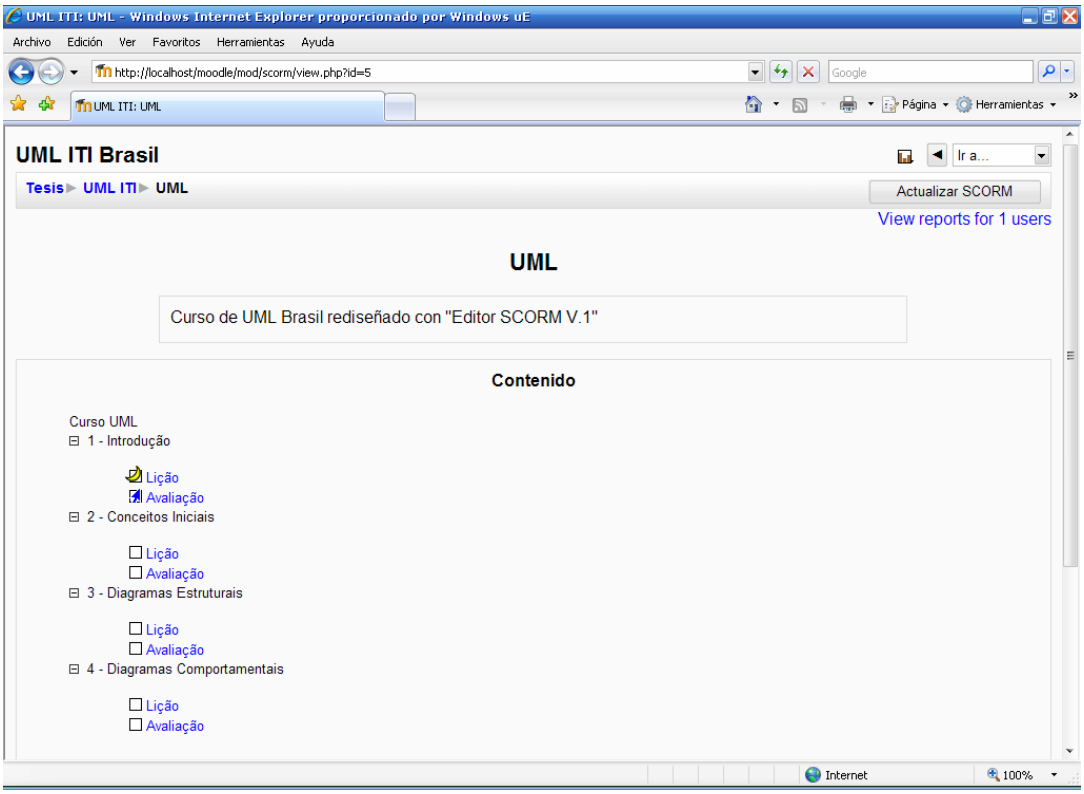


Figura 46. Índice del Curso con el estado de la navegación.

Si ingresamos a la sección de informes de la actividad SCORM que creamos, se muestra la pantalla de la figura 47.

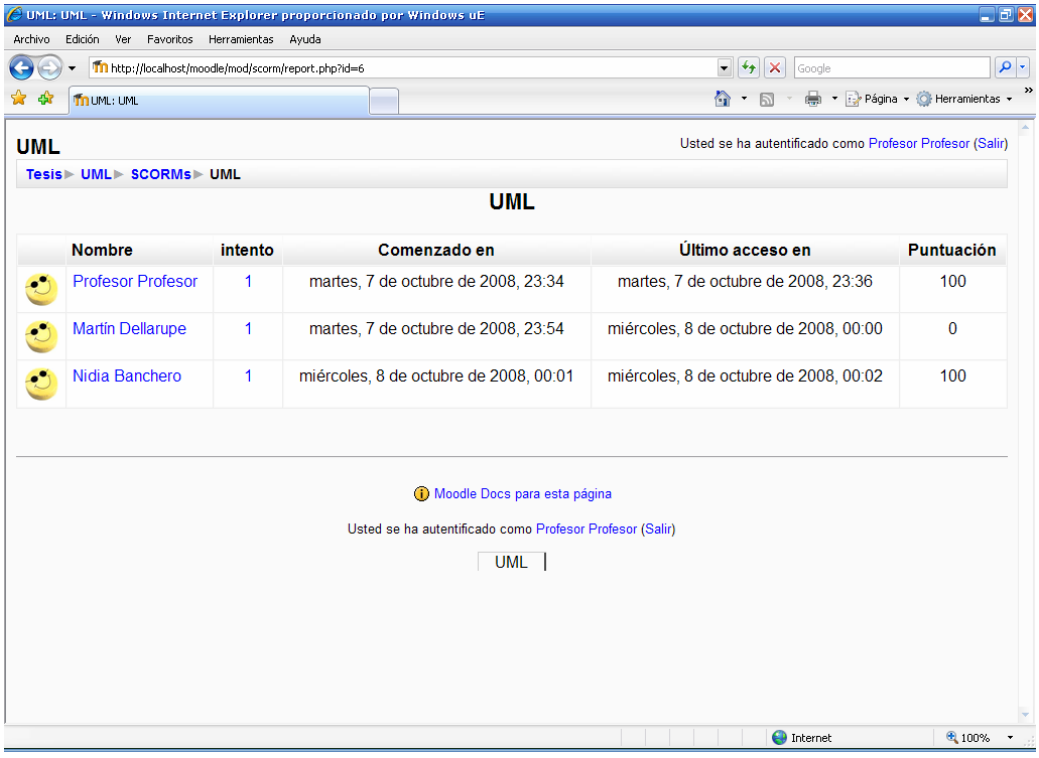


Figura 47. Informe de actividad del curso UML.

En la figura 47 vemos para cada usuario un link a la cantidad de intentos, cuando comenzó la navegación, cuando fue su último acceso y la puntuación alcanzada.

Si seleccionamos el alumno Nidia Banchero y accedemos al link de la cantidad de intentos (1) se presentara la pantalla que se muestra en la figura 48.

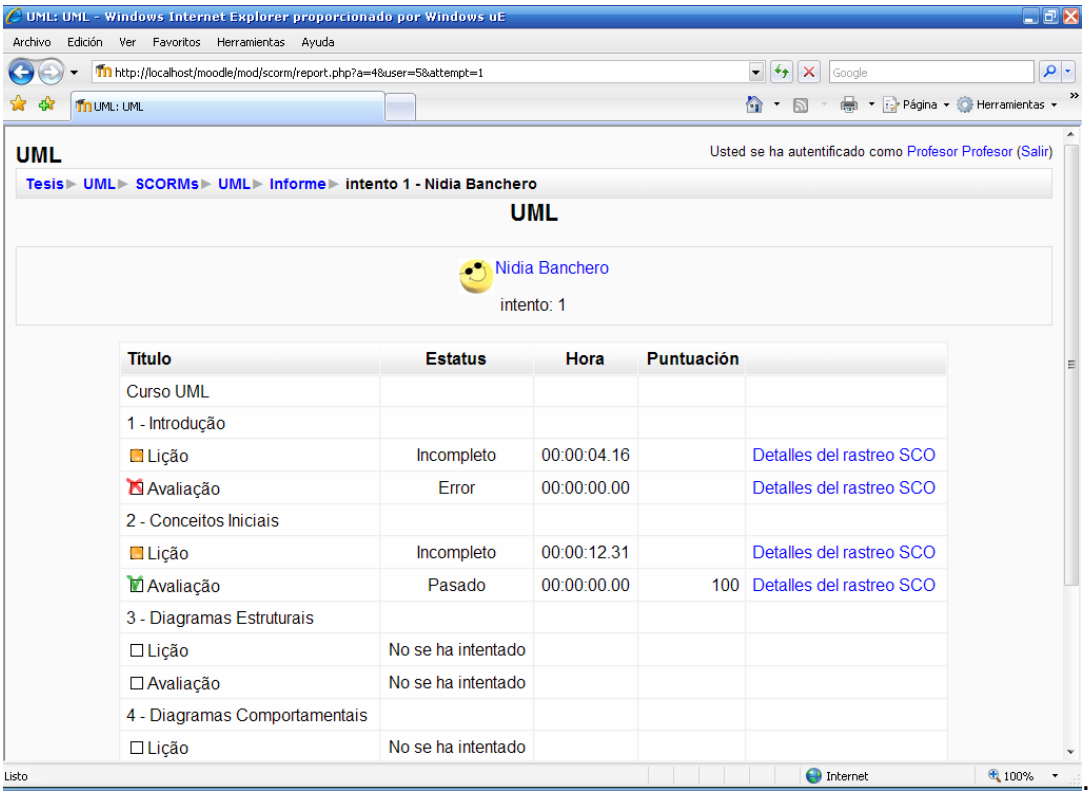


Figura 48. Información de la navegación de un alumno.

Se presenta al profesor una planilla con la información de la navegación donde se indica para cada unidad:

- el estado. Si el tema de la unidad son páginas navegables se indica: Completo, Incompleto y No se ha intentado. En el caso de ser una evaluación se indica: "Error", si no se logró, "Pasado" si la respuesta fue correcta y "No se ha intentado" si no se navego.
- hora: tiempo total de navegación para el tema.
- puntuación en el caso de ser una evaluación se presenta la puntuación obtenida.
- un link al detalle del rastreo SCO.

Si seleccionamos el link "Detalle del rastreo SCO" podemos ver para el tema seleccionado la pantalla que se muestra en la figura 49. Donde se presenta al profesor información mas detallada de la navegación del alumno.

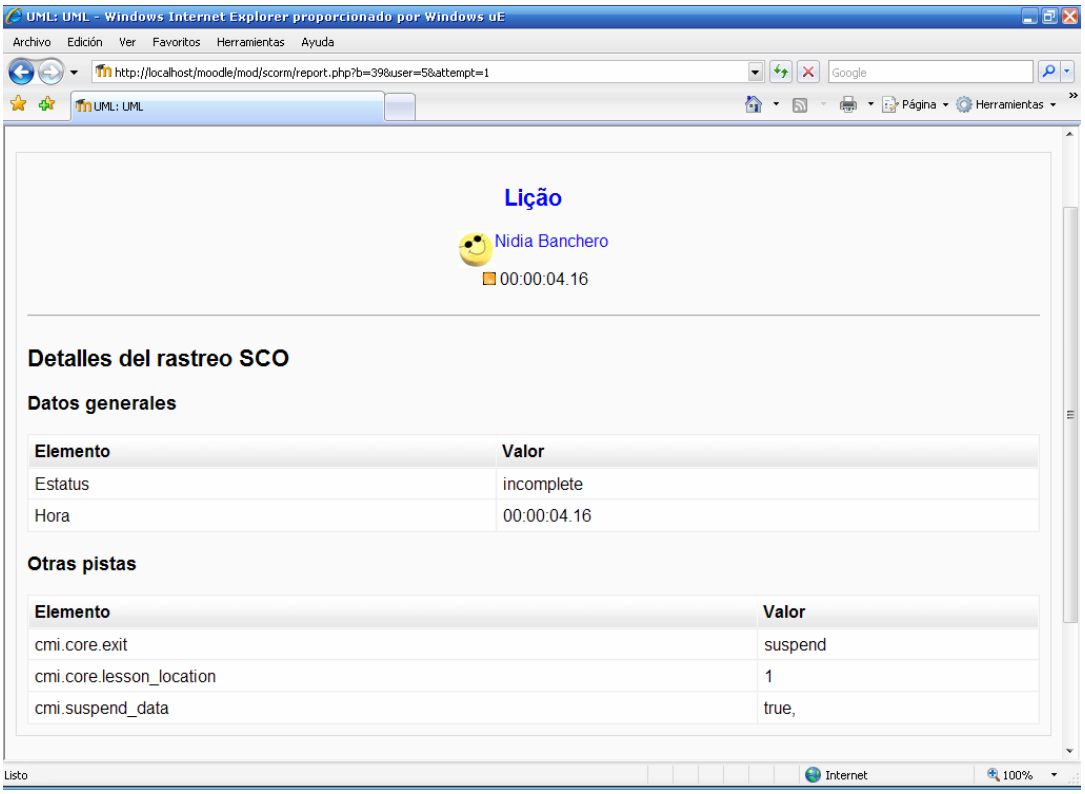


Figura 49. Detalle del rastreo SCO

Si navegamos nuevamente el curso como la alumna Nidia Banchero y resolvemos el examen veremos que la planilla de información de la navegación se actualizó, como se puede visualizar en la figura 50. Ahora la evaluación para el primer tema de unidad "Introducao" se presenta con el estado "Pasado" y en puntuación la nota obtenida.

Figura 50. Informe de la navegación de un alumno

Hay que tener en cuenta que para el ejemplo que hemos seleccionado del curso UML solamente las evaluaciones involucran una única pregunta. En el caso que el cuestionario abarque varias preguntas al seleccionar el link "Detalles del rastreo de SCO" se indica el profesor el resultado de cada pregunta.

Otra forma de ver el informe de navegación del alumno es accediendo por el link del nombre de la planilla que se muestra en la figura 47. Donde al ingresar al link del alumno Nidia Banchero se presenta la pantalla que se visualiza en la figura 51.

Figura 51. Información de un alumno en particular

Al seleccionar la solapa "Informes de actividad" y luego "Informe Completo" se presenta al profesor la pantalla que se muestra en la figura 52, con el detalle de rastreo SCO en forma completa para cada unidad.

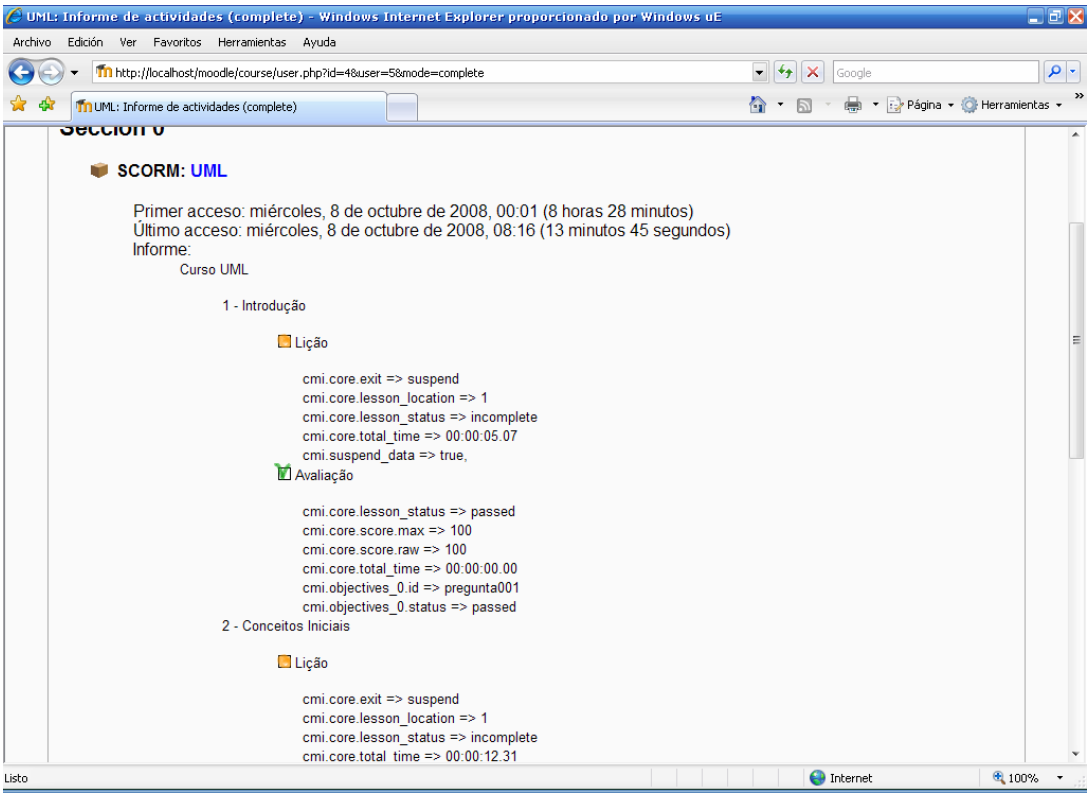


Figura 52. Informe completo de la navegación del alumno

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.

Conclusión

El objetivo de este trabajo es optimizar el proceso de creación de material educativo para las distintas cátedras de la facultad, facilitando una de las etapas de la creación de contenido educativo como es la implementación de la comunicación entre los OA y el LMS.

Para poder llevar a cabo este trabajo, en primer lugar debimos estudiar y manejar conceptos tales como educación a distancia, objetos de aprendizaje, recursos educativos reusables, entornos de aprendizaje y estándares. Debimos investigar en profundidad estos temas, familiarizarnos con los conceptos y con las herramientas existentes para trabajar sobre ellos.

Aquí observamos que si bien existe un gran número de sistemas LMS propietarios, cada día existen más herramientas de software libre que permiten realizar las mismas tareas que los sistemas comerciales marcando una clara tendencia hacia el desarrollo y enriquecimiento de software libre.

Se estudió en detalle el estándar SCORM y también el entorno Moodle, como ejemplo de sistema LMS Open Source.

Actualmente la construcción de material educativo que respete un estándar, es de vital importancia para la generación de recursos reusables e interoperables. En la etapa de construcción del material educativo, normalmente se utilizan editores HTML. Estos editores, ya sean versiones comerciales o de software libre, si bien permiten crear de manera sencilla un curso de aprendizaje, no brindan la posibilidad de generar contenido educativo respetando el estándar SCORM.

Nuestro trabajo consistió en tomar un editor HTML de código abierto y extenderlo para incorporarle contenido SCORM al material educativo que se genere.

Para poder seleccionar el editor adecuado debimos analizar varias herramientas de este tipo. Esto requirió configurar diferentes ambientes, proceder a su instalación y posterior prueba para conocer su funcionamiento y una vez realizados estos pasos, proceder a analizar y comprender el código fuente de cada uno, para lo cual, en ciertos casos fue necesario introducirnos en distintos lenguajes de programación. Esta etapa fue una de las más enriquecedoras del trabajo, ya que nos permitió habituarnos a interpretar código fuente desarrollado por otras personas y poder extenderlo.

FCKeditor, fue el editor elegido, debido a que cuenta con las características básicas de cualquier editor, está en constante desarrollo, es sencillo y fácil de utilizar. El mayor problema para extenderlo fue la falta de documentación existente. Aunque los ejemplos nos ayudaron a ver lo que hace el código fuente, tuvimos que investigar, decodificar y seguir arduamente el programa para poder identificar y comprender cada una de las funciones del editor. Además, para poder analizar el código fuente tuvimos que descompactar el código interno, ya que los archivos que componen FCKeditor son distribuidos con el código compactado para que sea más rápida su carga.

La aplicación final responde a un modelo de curso específico ya que fue necesario acotar las funciones SCORM implementadas debido a que la API SCORM provee una gran cantidad de funciones para el envío de información al LMS. Se seleccionaron las funciones SCORM más comúnmente utilizadas y se buscó un modelo de curso genérico, que además sea utilizado actualmente en forma masiva, sobre el cual pudieran ser aplicadas estas funciones.

Una vez seleccionado el modelo de curso y planteada la aplicación se generó un caso de prueba, rearmando un curso tomado del sitio de referencia (ITI) y construyendo las mismas páginas con nuestro editor. Como paso siguiente debimos

verificar que el curso generado respondiera al estándar SCORM pero comportándose de la misma forma que los cursos originales. En esta etapa, fue necesario generar el ambiente para utilizar el software ReloadEditor que nos permitiera construir el paquete SCORM y también generar el entorno de aprendizaje Moodle para ver como se comportaba en un LMS el curso generado.

Se pudo concluir que la funcionalidad obtenida era la misma, logrando además generar recursos educativos que respetasen el estándar SCORM.

Con la herramienta obtenida se favoreció al enriquecimiento de los cursos para hacerlos reusables y poder recuperarlos en motores de búsqueda SCORM. Además tiene la ventaja de ser open source, lo que representa una contribución significativa porque no existen actualmente herramientas bajo esta filosofía.

Esta herramienta representa una contribución concreta para el desarrollo de contenido educativo en entornos de software libre, ya que permite generar recursos que respetan el estándar SCORM, lo cual no está soportado por las herramientas actuales.

Dado que existe una brecha entre la definición del modelo SCORM y el proceso global de desarrollo de cursos por parte de docentes que diseñan cursos de aprendizaje, el uso de esta herramienta puede resultar una ayuda considerable para que puedan utilizar el estándar SCORM más fácilmente.

Se deja como líneas futuras de trabajo poder plantear diferentes modelos de cursos SCORM o la creación de cursos en forma independiente del modelo que propone la aplicación.

Referencias:

Ref. 1 "Usando XML para metaanotar recursos educativos" - Facultad de Informática - Universidad Nacional de La Plata - Tesis de Andrea Deluchi

Ref. 2 http://www.webct.com/

Ref. 3 http://www.blackboard.com/

Ref. 4 http://www.reload.ac.uk/

Ref. 5 http://moodle.org/
http://moodle.org/login/index.php

Ref. 6 IEEE: ltsc.ieee.org
IEEE Learning Technology Standards Committee (LTSC), Learning Object Metadata Working Group, en línea http://ltsc.ieee.org/wg12/s_p.html/ (2001).
IEEE LTSC LOM (2001). Draft Standard for Learning Object Metadata, Final version 1.2 accesible en http://ltsc.ieee.org/wg12/index.htm

Ref. 7 Wiley, D. A Connecting Learning Objects to Instructional Design Theory: A definition, a metaphor, and a taxonomy, Utah State University (2002).

Ref. 8 Mason R., Weller, M., Pegler, C. "Learning in the Connected Economy" The Open University course team, IET, Open University (2003).

Ref. 9 http://www.adlnet.org/
http://www.adlnet.org/index.cfm?flashplugin=1&fuseaction=home

Ref. 10 http://www.aicc.org/

Ref. 11 http://dublincore.org/usage/terms/dc/currentelements

Ref. 12 LOM Learning Object Metadata
http://ltsc.ieee.org

Ref. 13 SCORM (http://www.adlnet.org/)
http://xml.coverpages.org/scorm.html

SCORM 2004 and 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

Ref. 14 IMS: www.imsglobal.org

Ref. 15 AICC: www.aicc.org
AICC CMI001, accesible en
http://www.aicc.org/docs/tech/cmi001v3-5.pdf

Ref. 16 ADL: www.adlnet.org

Ref. 17 http://www.htmlarea.com

Ref. 18 http://interactivetools.com/

Ref. 19 http://dynarch.com/

Ref. 20 http://docs.moodle.org/en/Development:Moodle specific_customisations_to_the_HTML_editor
http://moodle.org/mod/forum/discuss.php?d=76912

Ref. 21 http://www.fckeditor.net/

Ref. 22 http://www.nvu.com
http://es.wikipedia.org/wiki/Nvu
http://www.utilidades-utiles.com/Descargar- nternet-Nvu.html
http://www.faq-mac.com/mt/archives/013006.php

Ref. 23 http://cursos.cdtc.org.br

Facilitando la creación y uso de objetos de aprendizaje en entornos de Software Libre

Anexo

Facultad de Informática
Universidad Nacional de La Plata

Director: Lic. Javier Diaz
Coodirector: Lic. María Alejandra Schiavoni
Alumnos: Martin Dellarupe Legajo: 1385/9
Alumnos: Nidia Banchero Legajo: 1633/7

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.
○ Llamada de ejemplo al API: LMSGetValue("cmi.core._children")
○ Ejemplo de valores devueltos: “student_id,student_name,lesson_location,credit,lesson_status,entry, score,total_time, exit,session_time”
○ 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::
○ 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</td></tr>
</table>

	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. ○ **Ejemplo de valor devuelto:** "Joe_Student1" "JS-2000" "joe2000-3" ○ **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: ○ **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. ○ **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 ○ **Ejemplo de valores devueltos:** "Estudiante, José Pérez." "Estudiante, Miguel Rioja." ○ **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: ○ **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:**	**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

Sólo lectura	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.
	○ **Ejemplo de valores devueltos:**
	"no-credit"
	"credit"
	○ **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:
	○ **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"**	**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

"" – *cadena vacía* **Accesibilidad** **SCO:** Sólo lectura	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::
	○ **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. ○ **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 ○ **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.
	○ **Ejemplo de una llamada del API:** LMSGetValue("cmi.core.score.max")
	○ **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.
	○ **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 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	**Definición:** La mínima puntuación o el número total de aciertos que el estudiante podría haber logrado.

API:
LMSGetValue()
LMSSetValue()
 	
Obligatorio LMS:
No
 	
Tipo de datos:
CMIDecimal or CMIBlank
 	
Accesibilidad SCO:
Leer/Escribir
	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.
	○ **Ejemplo de una llamada del API:** LMSGetValue("cmi.core.score.max")
	○ **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:
	○ **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.
	
	○ **Ejemplo de llamada al API:** LMSGetValue("cmi.core.lesson_mode")
	○ **Ejemplo de valores devueltos:** "browse" "normal" "review"
	○ **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:
	○ **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. ○ **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	**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

Tipo de dato: CMITimespan **Accesibilidad SCO:** Sólo escritura	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. ○ **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. ○ **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():** ○ **Ejemplo de llamada al API:** LMSSetValue("cmi.core.session_time","0010:34:34.56") ○ **Ejemplo de valores devueltos:** "0010:34:34.56" "05:15:00" ○ **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 llamada al API, y despliega un mensaje al
usuario. Si el código de error tiene ** ** asociado un texto lo despliega.
**
*************************************************************************
*************/
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>
```

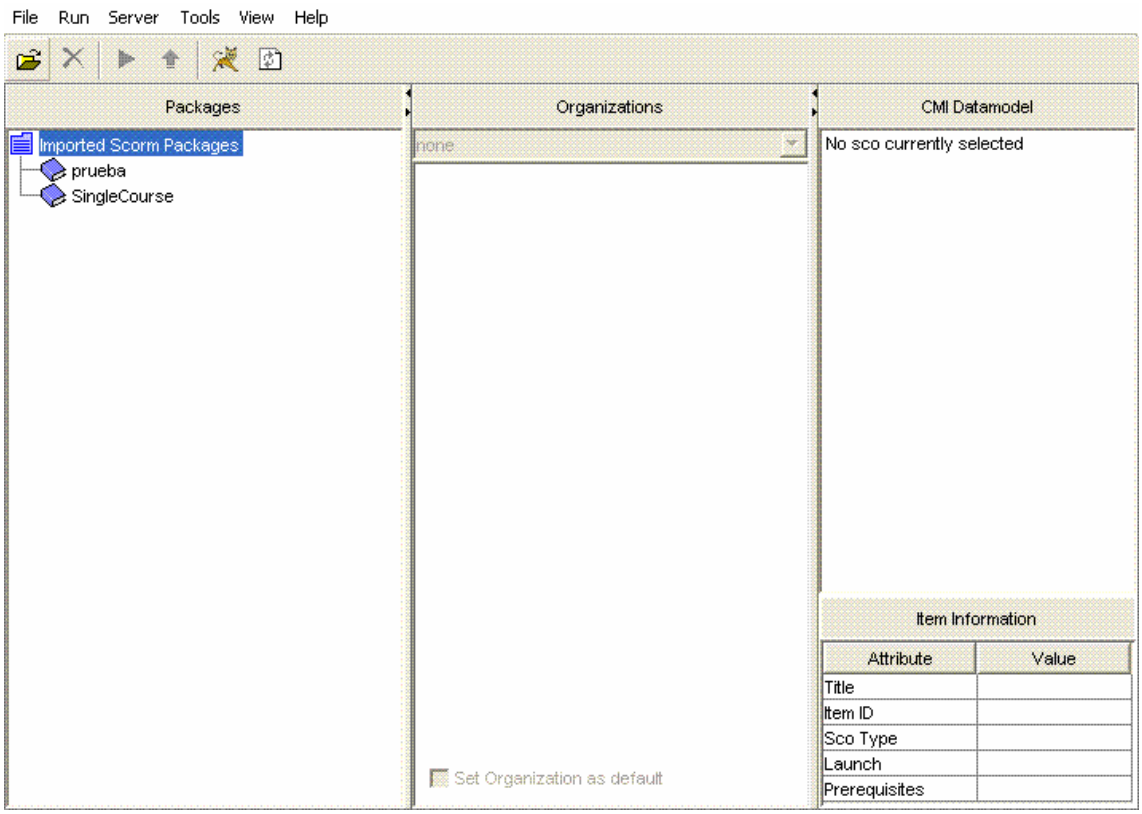
Ahora, armamos la siguiente estructura de directorios:

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

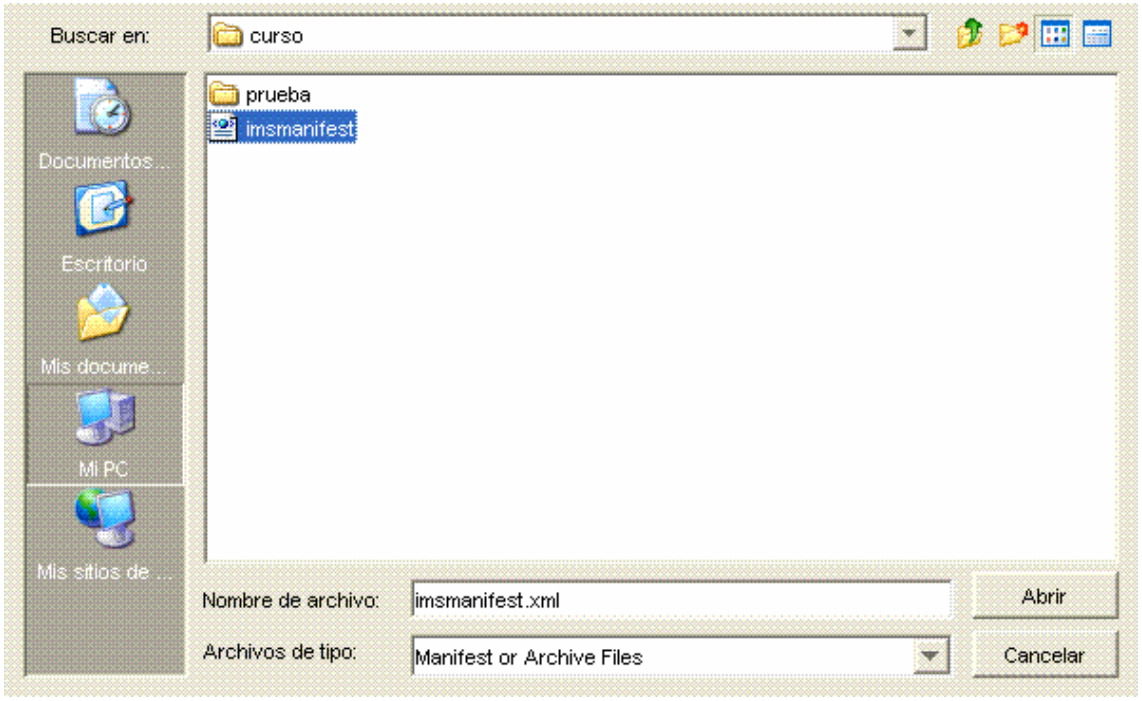
Estamos en condiciones de importar nuestro contenido en un LMS que soporte SCORM 1.2. Lo importaremos en Reload Scorm Player, para luego ejecutarlo. ReloadPlayer es una herramienta que permite testear los paquetes SCORM.

Las acciones para este paso son:

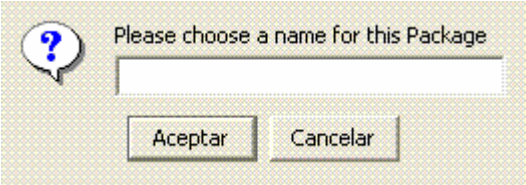
- Seleccionar el icono "Import Scorm Package" (primer ícono de la barra de íconos)



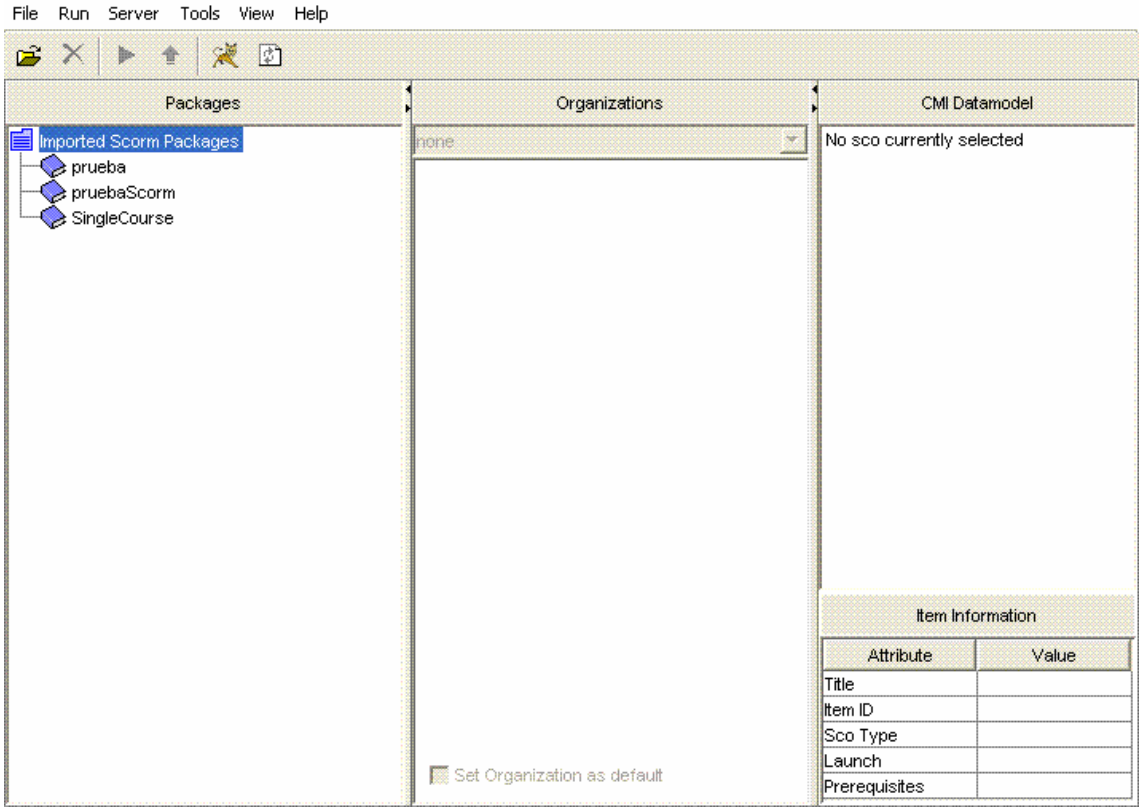
- Seleccionar el archivo imsmanifest.xml



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

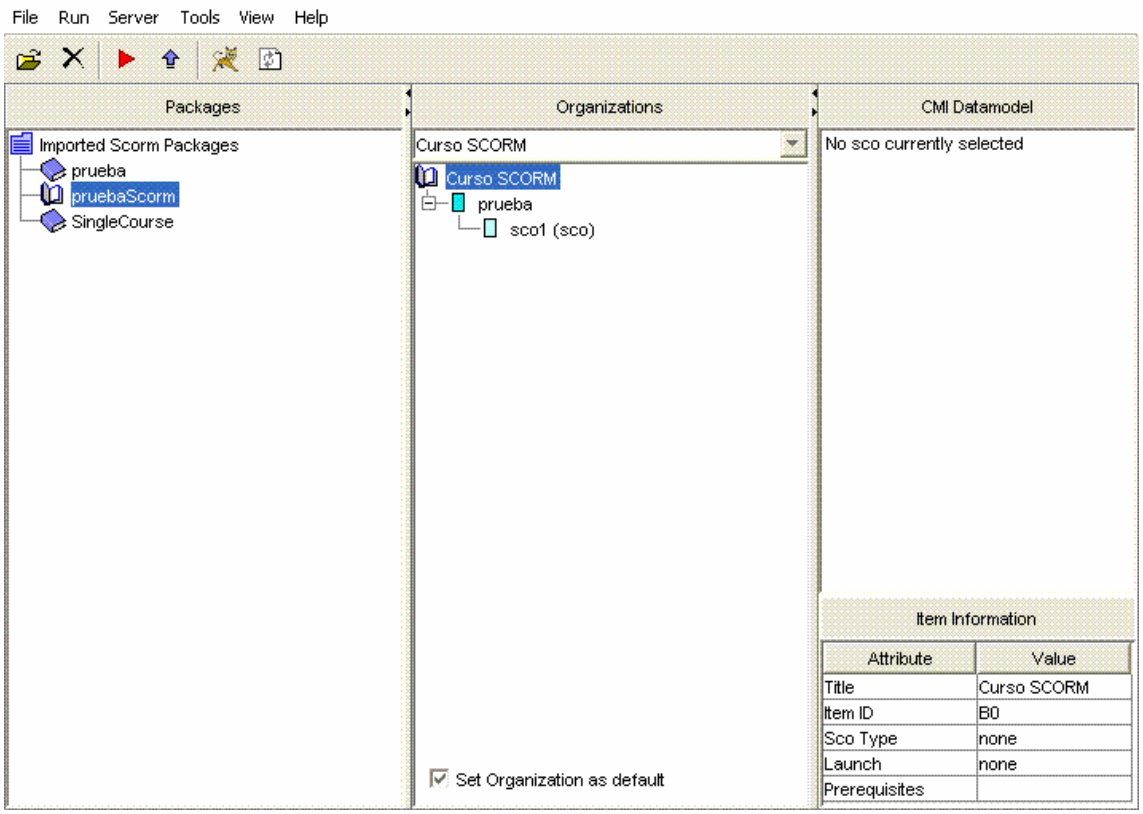


Si ingresamos el nombre "pruebaScorm", ahora nuestro curso aparecerá en el árbol con dicho nombre:

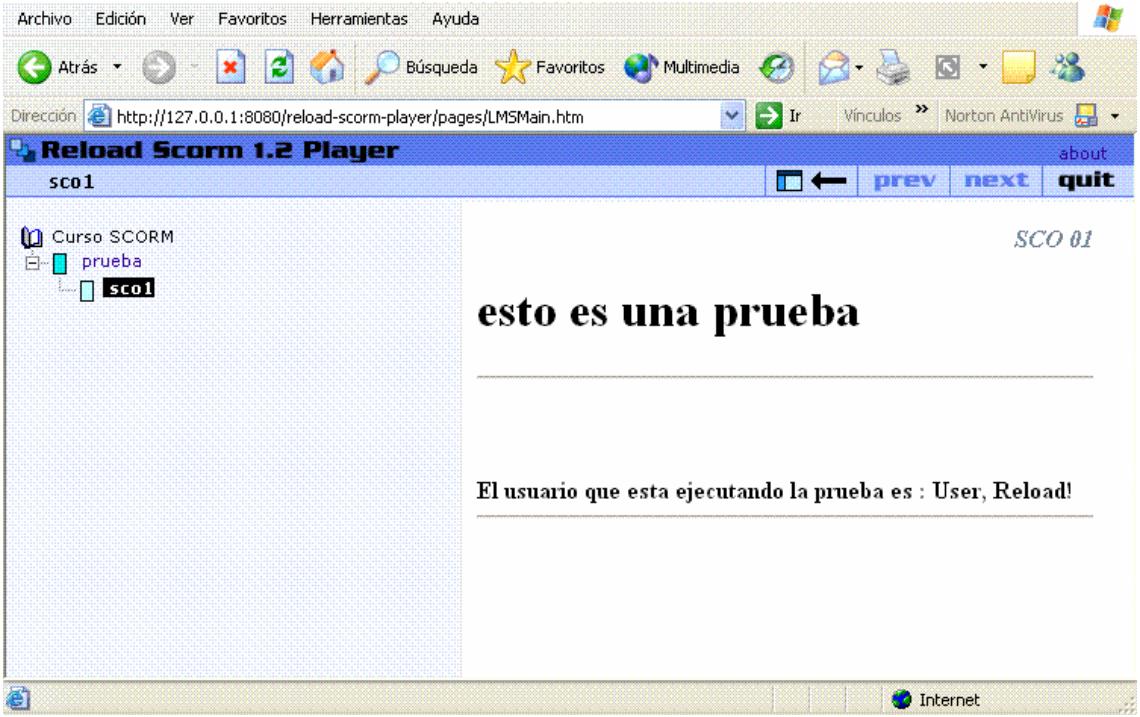


Las acciones para ejecutar el contenido importado son:

- 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

FCKScorm for E-Learning

Archivo fckeditorcode_ie.js

- ***Función: FCKCommands.GetCommand***

```
FCKCommands.GetCommand=function(A){
    var B=FCKCommands.LoadedCommands[A];
    if (B) return B;
    switch (A){
    …
    …
      case 'RadioScorm':
        B=new
      FCKDialogCommand('Radio',FCKLang.RadioButton,'dialog/fck_radioButtonSCORM.html'
      ,380,350);
        break;
      case 'CalcularNota':
        B=new
      FCKDialogCommand('Button',FCKLang.CalcularNota,'dialog/fck_buttonScormNota.html'
      ,380,230);
        calcularNota = 'true';
        break;
      case 'PaginaSiguiente':
        B=new
      FCKDialogCommand('PaginaSiguiente',FCKLang.PaginaSiguiente,'dialog/fck_buttonSigu
      iente.html',380,230);
        break;
      case 'PaginaAnterior':
        B=new
      FCKDialogCommand('PaginaAnterior',FCKLang.PaginaAnterior,'dialog/fck_buttonAnterio
      r.html',380,230);
        break;
…..
….
};
```

Funcionalidades de los botones en la etapa inicial

Para los casos detallados a continuación en la función "FCKCommands.GetCommand" del archivo "*fckeditorcode_ie.j"s* se invoco a las siguientes funciones que detallamos a continuación.
Si bien esta funcionalidad no prospero como se explico en el informe final puede ser de utilidad para las futuras líneas de desarrollo y es por esta razón que decidimos incluirlas en dicho anexo y con esto demostramos que la solución propuesta para el editor no es la única forma de poder definir funcionalidades en el editor FCKeditor.

- ***Pagina Siguiente:***

```
var FCKLinkPaginaSigScormCommand=function(){
        this.Name='LinkPagina';
};

FCKLinkPaginaSigScormCommand.prototype.Execute=function(){
        FCK.InsertHtml  ('<a href="JavaScript:window.parent.NextPage()">Pagina
Siguiente');
```

```
};

FCKLinkPaginaSigScormCommand.prototype.GetState=function(){return 0;};
```

- ***Pagina Anterior***

```
var FCKLinkPaginaAntScormCommand=function(){
        this.Name='LinkPagina';
};

FCKLinkPaginaAntScormCommand.prototype.Execute=function(){
        FCK.InsertHtml ('<a href="JavaScript:window.parent.PreviousPage()">Pagina
Anterior');
};

FCKLinkPaginaAntScormCommand.prototype.GetState=function(){return 0;};
```

- ***Preview de las paginas HTML***

```
/**********************************************************************
Esta función abre un nuevo browser con la pagina HTML generada con el Editor.
Agrega las llamadas a los java scripts correspondientes al API SCORM
Agrega las funciones JavaScripts necesarias para las funcionalidades de los botones
agregados para la funcionalidad SCORM
/**********************************************************************

PreviewScorm:function(){

        var A=FCKConfig.ScreenWidth*0.8;
        var B=FCKConfig.ScreenHeight*0.7;
        var C=(FCKConfig.ScreenWidth-A)/2;
        var
        D=window.open('',null,'toolbar=yes,location=no,status=yes,menubar=yes,scrollbars
=yes,resizable=yes,width='+A+',height='+B+',left='+C);
        var E;
        if (FCKConfig.FullPage){
                if (FCK.TempBaseTag.length>0)
                        E=FCK.TempBaseTag+FCK.GetXHTML();
                else E=FCK.GetXHTML();
        }else{
                E=FCKConfig.DocType + '<html
dir="'+FCKConfig.ContentLangDirection+'"><head>' + '<title>Editor SCORM</title><link
href="estilos.css" rel="stylesheet" type="text/css"><SCRIPT LANGUAGE=Javascript
src="script/scripts.js"></SCRIPT><SCRIPT LANGUAGE=Javascript
src="SCORMGenericLogic.js" type="text/javascript"></SCRIPT><SCRIPT
LANGUAGE=Javascript src="SCORMObjetiveLogic.js" type="text/javascript"></SCRIPT>' +
FCK.TempBaseTag+'<title>'+FCKLang.Preview +'</title>' + '<script language="JavaScript"
type="text/JavaScript">function saveHtml(){
document.getElementById("tbGuardar").style.display ="none";
window.document.execCommand(\'saveas\');}</script>'
+_FCK_GetEditorAreaStyleTags()+'</head><body>'+'<table width="100%" border="0"
align="center" cellspacing="1"><tr><td>' + FCK.GetXHTML()+'</td></tr></table><table
id="tbGuardar" name ="tbGuardar" width="100%" border="0" align="center"
cellspacing="1" class="botones"><tr><td><A onmouseover="cursorMano()"
onmouseout="cursorAuto()" onClick="saveHtml()">Guardar</A></td></tr></table>' +
'</body></html>';

        };
```

```
	D.document.write(E);
	D.document.close();
}

/**************************************************************************
Esta función abre un nuevo browser con la pagina HTML generada con el Editor.
Agrega en el Body del HTML las funciones de inicialización y finalización de la
comunicación LMS – SCO
Agrega las llamadas a los java scripts correspondientes al API SCORM
Agrega las funciones JavaScripts necesarias para las funcionalidades de los botones
agregados para la funcionalidad SCORM
**************************************************************************/

PreviewScormInic:function(){
	var A=FCKConfig.ScreenWidth*0.8;
	var B=FCKConfig.ScreenHeight*0.7;
	var C=(FCKConfig.ScreenWidth-A)/2;
	var
	D=window.open('',null,'toolbar=yes,location=no,status=yes,menubar=yes,scrollbars
=yes,resizable=yes,width='+A+',height='+B+',left='+C);
	var E;
	if (FCKConfig.FullPage){
		if (FCK.TempBaseTag.length>0)
			E=FCK.TempBaseTag+FCK.GetXHTML();
		else E=FCK.GetXHTML();
	}else{

	if (calcularNota == 'true') {
		E=FCKConfig.DocType + '<html dir="' + FCKConfig.ContentLangDirection +
'"><head>' + '<title>Editor SCORM</title><link href="estilos.css" rel="stylesheet"
type="text/css"><SCRIPT LANGUAGE=Javascript
src="script/scripts.js"></SCRIPT><SCRIPT LANGUAGE=Javascript
src="SCORMGenericLogic.js" type="text/javascript"></SCRIPT><SCRIPT
LANGUAGE=Javascript src="SCORMObjetiveLogic.js" type="text/javascript"></SCRIPT>' + '
<SCRIPT LANGUAGE=Javascript type="text/javascript"> var cant=0; function
EvalMultipleChoiceItem(obj,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;}SCOCommit()}} function
ShowSCORMStatus(){SCOSetValue("cmi.core.score.raw",cant/5);SCOCommit();} function
saveHtml(){ document.getElementById("tbGuardar").style.display ="none";
window.document.execCommand(\'saveas\');} </SCRIPT>' + FCK.TempBaseTag +
_FCK_GetEditorAreaStyleTags()+'</head><body onload="SCOInitialize()"
onunload="SCOFinish()" onbeforeunload="SCOFinish()">'+'<table width="100%"
border="0" align="center" cellspacing="1"><tr><td>' +
FCK.GetXHTML()+'</td></tr></table><table id="tbGuardar" name ="tbGuardar"
width="100%" border="0" align="center" cellspacing="1" class="botones"><tr><td><A
onmouseover="cursorMano()" onmouseout="cursorAuto()"
onClick="saveHtml()">Guardar</A></td></tr></table>' + '</body></html>';
	}else{
		E=FCKConfig.DocType + '<html
dir="'+FCKConfig.ContentLangDirection+'"><head>' + '<title>Editor SCORM</title><link
href="estilos.css" rel="stylesheet" type="text/css"><SCRIPT LANGUAGE=Javascript
src="script/scripts.js"></SCRIPT><SCRIPT LANGUAGE=Javascript
src="SCORMGenericLogic.js" type="text/javascript"></SCRIPT><SCRIPT
LANGUAGE=Javascript src="SCORMObjetiveLogic.js" type="text/javascript"></SCRIPT>' +
FCK.TempBaseTag + '<script language="JavaScript" type="text/JavaScript">function
saveHtml(){ document.getElementById("tbGuardar").style.display ="none";
window.document.execCommand(\'saveas\');}</script>'
+_FCK_GetEditorAreaStyleTags()+'</head><body onload="SCOInitialize()"
onunload="SCOFinish()" onbeforeunload="SCOFinish()">'+'<table width="100%"
border="0" align="center" cellspacing="1"><tr><td>' +
```

```
FCK.GetXHTML()+'</td></tr></table><table id="tbGuardar" name ="tbGuardar"
width="100%" border="0" align="center" cellspacing="1" class="botones"><tr><td><A
onmouseover="cursorMano()" onmouseout="cursorAuto()"
onClick="saveHtml()">Guardar</A></td></tr></table>' + '</body></html>';
        }
        };
        D.document.write(E);
        D.document.close();
},
```