LOMEditor 2006:

Editor de objetos de aprendizaje en base a plugins

Autores: Abraham Alonso Calvo | Elena Cañas Caicoya | Daniel Nieto Sanz
Dirigido por: Prof. Antonio Sarasa Cabezuelo
Dpto. Sistemas Informáticos y Programación
Curso académico: 2005 | 2006

Proyecto de Sistemas Informáticos
Facultad de Informática
Universidad Complutense de Madrid

Índice de contenidos

0 RESUMEN DEL PROYECTO

0.1 Resumen

Internet ha revolucionado el mundo de las comunicaciones y los negocios. Esta nueva tecnología abre también un abanico de posibilidades al aprendizaje de las personas, así como a las nuevas formas de diseño de material educativo.

La tecnología conocida como Objetos de Aprendizaje (OA) es candidata a ser la tecnología de elección para futuras generaciones, gracias a su potencial (reusabilidad, adaptabilidad, escalabilidad y capacidad de generación)

Los OA están basados en el paradigma de Orientación a Objetos de la Ciencia Informática. Pueden construirse pequeños OA (en comparación a un curso) que podrán ser reutilizados en diferentes contextos. Estas unidades estarán en la red al alcance (simultáneo) de los consumidores de objetos, esta es una de las grandes ventajas de los OA.

Interesa a la comunidad científica, estandarizar y normalizar las tecnologías educativas, los trabajos de instituciones y agencias (LTSC, ARIADNE, IMS) han proporcionado una serie de estándares.

Son necesarias herramientas que implementen estos estándares, y que permitan el manejo de OAs. LOMEditor 2006 pretende cumplir este objetivo.

0.2 Abstract

Internet has transformed the communications landscape and the way we do business. This technology has opened an array of possibilities for people to learn, as well as new ways to design teaching material.

The technology known as "Learning Objects" (LO's) is destined to become the chosen technology for future generations due to its potential (adaptability, re-usage, economies of scale and capacity of generation).

LO's are based on the paradigm of object orientation of computer science. We can build small LO's (compared to a whole academic year) that can be used over again in different contexts. These units will be available in the web at hand (simultaneously) of objects consumers; this is one of the biggest advantage of the LO's.

It is in the advantage of the scientific community to normalize and standardize education technologies. Institutions and agencies (LT SC, ARIADNE, IMS) have been working to provide several standards.

We need tools that comply with these standards and that unable us to work with LO's. LOMEditor 2006 main purpose is to do achieve this.

0.3 Autorización

Se Autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado:

Fdo. Elena Cañas Caicoya

Fdo. Daniel Nieto Sanz

Fdo. Abraham Alonso Calvo

En Madrid, a 3 de Julio de 2006.

0.4 Lista de palabras clave para búsquedas bibliográficas

Objeto aprendizaje, E-learning, IMS, Content package, Simple sequencing, Metadata, Plugins, Repositorio.

1 Introducción

1.1 Estándares

Interesa a la comunidad científica, estandarizar y normalizar las tecnologías educativas, los trabajos de instituciones y agencias han proporcionado una serie de estándares.

Con este objetivo se funda en 1996 LTSC (Learning Technology Standards Committee) de IEEE (Institute of Electrical and Electronics Engineers). Fundamentalmente se busca la interoperabilidad entre universidades, corporaciones y resto de organizaciones del mundo.

La UE empieza a financiar ARIADNE (Alliance of Remote Instructional Authoring and Distribution Networks for Europe), así como en EEUU comienza IMS (Instructional Management Systems). ADL empieza a desarrollar estándares técnicos para la generación de objetos de aprendizaje.

1.2 ¿Qué es un Objeto de Aprendizaje?

Un objeto de aprendizaje es una unidad de información digital, con objetivo didáctico (David A.Willey). Se trata de un conjunto de recursos que, organizados y normalizados, constituyen un paquete de datos.
Un objeto de aprendizaje esta formado por:

-Recursos: archivos físicos que contienen la información educativa

-Organización de estos recursos

-Meta datos: descripción del objeto de aprendizaje mediante atributos

Según la especificación IMS_CP, la estructura de un OA es:

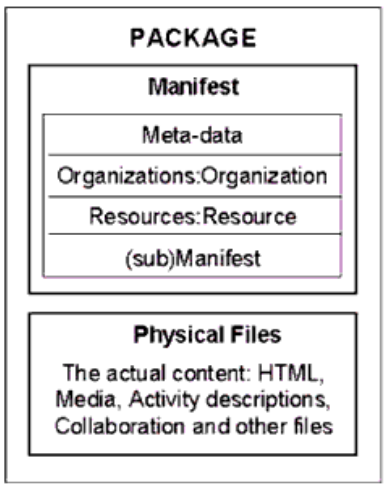


Figura 1. Estructura de un O.A. Fuente: IMS

El OA empaquetado según IMS_CP, esta formado por su manifiesto y sus archivos físicos.
El manifiesto (Manifest) es un archivo XML con las secciones:

-Meta-data: describe el propio paquete y alguno de sus componentes

-Organizations: establece la estructura de los recursos

-Resources: referencia recursos externos o archivos físicos del paquete y relaciones entre los mismos

2 EL PROYECTO LOMEditor 2006

2.1 Planteamiento del proyecto

El proyecto LOMEditor 2006 pretende dar un mejor soporte a la creación y edición de objetos de aprendizaje.

El punto de partida es el proyecto LOMEditor 2005, fruto del cual surgió la herramienta LOMEditor 2005, capaz de abrir y editar objetos de aprendizaje IMS Content Package, así como de crearlos desde cero. Esta herramienta ofrece también la posibilidad de evaluar la calidad y de componer objetos de aprendizaje.

El proyecto LOMEditor 2006 cumple los objetivos:

- Modularizar la herramienta LOMEditor 2005 dotándola de una estructura de piezas software enchufables o plugins (mecanismo de extensión que permite añadir nuevas funcionalidades o modificar las existentes).

- Añadir las funcionalidades de edición y creación de

    - Objetos de aprendizaje IMS Metadata
    - Objetos de aprendizaje con secuenciación (IMS Simple Sequencing).

- Añadir el acceso a un repositorio remoto de OA

Estos objetivos se funden en una nueva versión de la herramienta: LOMEditor 2006 a base de plugins, un la herramienta pionera en dar soporte a objetos que siguen la especificación de secuenciación propuesta por IMS (IMS Simple Sequencing).

2.2 Puntos clave de investigación

LOMEditor 2005 era un entorno que permitía la creación y edición de objetos de aprendizaje IMS Content Package, así mismo tenía funciones novedosas como la composición en paralelo y en serie de OA, y la evaluación CBR de OA.

LOMEditor 2006 mantiene estas funcionalidades, pero las amplía añadiendo otras nuevas como: soporte para plugins, que pueden "enchufarse" a la aplicación, a demás ofrece la posibilidad de crear y editar

objetos IMS Simple Sequencing, opción que herramientas como RELOAD (versión Carnelling) han implementado ya, pero dando este soporte unido a SCORM. Sin embargo, nuestra herramienta da soporte a la secuenciación con independencia de SCORM.

2.2.1 Plugins

Aplicaciónes basadas en plugins

Para crear una aplicación basada en plugins, se parte de una aplicación que no tiene ninguna funcionalidad (podría tener alguna por defecto), pero que admite módulos que implementan un estándar predeterminado y que permiten gestionar las funcionalidades que preste cada módulo.

Para crear este tipo de arquitectura, se aplica el polimorfismo. Una serie de objetos implementarán un interfaz, estos objetos serán los plugins. Esto, en términos de patrones de diseño se conoce como Factoría.

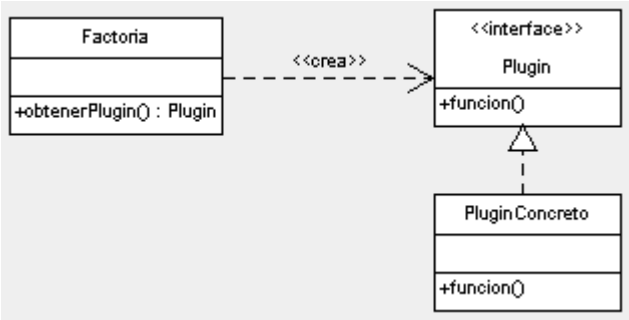


Figura 2: Patron Factoría e Interfaz de plugins. Fuente: El rincón del programador: http://www.elrincondelprogramador.com/default.asp?pag=articulos/leer.asp&id=61

Cada módulo es un plugin, y se desarrolla a la par con la aplicación, para luego aislarlo y permitir incluir las funcionalidades de este módulo a la aplicación, o no, a gusto del usuario.

Pasar de una arquitectura modelo-vista-controlador a una arquitectura de plugins

Para hacer de LOMEditor una herramienta a base de plugins, se parte de una primera versión (LOMEditor 2005) con arquitectura modelo – vista –controlador, en la que la lógica de la aplicación está concentrada en el controlador. Deciden separarse dos de las utilidades de la herramienta, que están modularizadas cada una en su respectivo paquete de clases, pero que a su vez están referenciadas desde el controlador.

Se crean dos controladores, uno para referenciar a cada módulo de utilidad, a su vez, a cada utilidad se le añade una clase

que implementa el interfaz de los plugins, esta clase hará referencia a los métodos del controlador de cada utilidad.

Cada paquete de clases que contiene una utilidad, se comprime en un fichero *Jar*, que se colocará en un directorio determinado (accesible para la aplicación).

Por otro lado, se define una clase manejadora de plugins que, al arrancar, "investiga" en dicho directorio si se dispone de algún fichero "enchufable" y si es así lo analiza e incluye a las opciones de gestión de los objetos de aprendizaje.

La aplicación al desnudo (sin plugins cargados), permite abrir y editar objetos de aprendizaje. Una funcionalidad extraída de la aplicación original es la composición de objetos de aprendizaje, con la que se creo el primer plugin. Se creó un segundo plugin con la funcionalidad de evaluación CBR y base de datos.

Para poder hacer uso de los plugins, éstos deberán implementar un interfaz común, de modo que desde la aplicación principal, se invocará a un método (que implementan todos los plugins) pasándole ciertos parámetros (en concreto una cadena con las opciones) de modo que cada plugin ejecute la acción que se le demanda desde el interfaz.

Al arrancar la aplicación, el plugin manager busca los plugins en el sistema, una vez encontrado uno, se analiza el fichero *Jar*, que ha de contener un manifiesto en el que se especifiquen el nombre del plugin, y las opciones de gestión del objeto que permite. En función a estas opciones, se crea un nuevo menú (dinámicamente) en el interfaz de la aplicación que permita invocar a cada una de estas nuevas funcionalidades.

Desarrollo de nuevos plugins

Para el desarrollo de nuevos plugins se dispone de una versión de la aplicación, llamada Versión de Desarrollo, en la cual se implementan los plugins, se prueban y posteriormente se empaquetan para ser "enchufados" en la versión que se presenta al usuario.

2.2.2 La Secuenciación y sus etiquetas

¿Que es la secuenciación?

Básicamente la secuenciación es una herramienta que nos permite indicar como debe consumirse un objeto de aprendizaje, en definitiva se trata de una serie de requisitos y normas que deben satisfacerse para el correcto aprovechamiento del objeto de aprendizaje. Por ejemplo, tal vez nos interese que el alumno que está trabajando sobre un objeto de aprendizaje en particular, no pueda pasar a la siguiente actividad hasta que

no haya respondido adecuadamente una serie de preguntas, o tal vez, que visione una serie de diapositivas antes de continuar; o por el contrario que tenga un tiempo limite para pasar a la siguiente actividad cumpliendo una serie de objetivos.

Como se puede apreciar la secuenciación nos permite establecer un control sobre el flujo a través de un objeto de aprendizaje, de este modo los educadores que hagan uso de nuestra herramienta podrán no solo proporcionar unos contenidos a sus alumnos, sino que además serán capaces de asegurarse que los consumen en el orden que les asegure un máximo aprovechamiento de los mismos.

¿Como se consigue añadir secuenciación a un objeto de aprendizaje?

Para incluir secuenciación en un objeto de aprendizaje basta con incluir de forma adecuada en base a unos estandares, una serie de etiquetas xml (descritas en el Apéndice B: Etiquetas de secuenciación (elementos de la secuenciación)) en el manifiesto de dicho objeto. Estas etiquetas serán analizadas posteriormente y comparadas con las gramáticas que describen la secuenciación para ser validadas. Una vez hecho esto están listas para ser interpretadas por una herramienta como LOMEditor y de este modo el objeto de aprendizaje que contenga dichas etiquetas puede ser consumido y modificado, en base a las reglas de secuenciación que en el fueron incluidas.

Las gramáticas que permitien conocer como esta estructurada la secuenciación se encuentran en la página de ims (www.imsproject.org) así como la descripción de cada una de sus etiquetas y ejemplos sobre como incluirlas adecuadamente en los objetos de aprendizaje.

3 ESPECIFICACIÓN DE REQUISITOS DEL SISTEMA

3.1 Introducción

3.1.1 Descripción general

Como ya se ha comentado previamente el objetivo de este proyecto era ampliar las posibilidades de una herramienta de autoría de objetos de aprendizaje ya existente.

La arquitectura de la aplicación soporta plugins, fácilmente ampliable.

Se permite así mismo la edición y creación de tres tipos de objetos de aprendizaje respetando en todo momento la línea que marcan el estándar de e-learning IMS.

- IMS Metadata
- IMS Content Package
- IMS Simple Sequencing

La validación de los OA es independiente de la versión de la gramática, lo que hace que la herramienta no se quede obsoleta en caso de evolución de dichos estándares.

La herramienta, al abrir un objeto existente valida su manifiesto asociado para verificar que cumple la estructura definida en una de las tres gramáticas previamente mencionadas.

Cada objeto de aprendizaje es un paquete que contiene un manifiesto, su gramática y el resto de recursos (si los hubiera), por lo que en el caso de que aparezca una nueva versión de la gramática, bastara que un desarrollador software cambie el nombre de la versión y la herramienta estará completamente actualizada.

3.1.2 Ámbito de la aplicación

El sistema ofrece funcionalidades básicas de un sistema de autoría de objetos de aprendizaje, edición de objetos y creación de los mismos

La arquitectura en módulos del sistema ha sido modificada y se han creado plugins a partir de dichos módulos, estos plugins son completamente independientes a la aplicación y solo constituirán una funcionalidad de la misma en el caso de que el usuario así lo desee.

Se crearon partiendo de módulos preexistentes en LOMEditor 2005 los plugins:

- Plugin Composición de OA
- Plugin Evaluación CBR y Base de datos

Así mismo se creó un nuevo plugin (Plugin Repositorio) que ofrece la posibilidad de conectarse a una lista de repositorios vía Web, así como de añadir direcciones de nuevos repositorios que surjan en la red.

Lo más novedoso es la implementación del estándar de IMS Simple Sequencing, por tanto se podrán crear y editar tres tipos de OA:
IMS Metadata, IMS Content Package e IMS Simple Sequencing.

3.1.3 Producto a entregar

El producto a entregar consta de las secciones:

Sección 1: Transformación de los módulos de Composición y Evaluación CBR y Base de datos a Plugins

Sección 2: Creación de un nuevo plugin :Plugin Repositorio

Sección 3: Ampliación de las opciones de edición y creación de OA a IMS Metadata e IMS Simple Sequencing

Funcionalidades añadidas a la herramienta de autoría

_____

1. Apertura de objeto de aprendizaje: se permite elegir entre los tipos: IMS Metadata, IMS Content Package e IMS Simple Sequencing, previa validación.

2. Edición de un objeto de aprendizaje (Se añadió la posibilidad de editar Metadata y Simple Sequencing)
    2.1. Adición de nuevos elementos.
    2.2. Eliminación de elementos.
    2.3. Modificación de atributos de elementos.

3. Creación de un nuevo objeto de aprendizaje: pudiendo elegirse entre: IMS Metadata, IMS Content Package e IMS Simple Sequencing

4. Acceso a repositorios de OA vía Web

5. Ayuda: completada con las nuevas opciones

3.2 Contexto de uso

3.2.1 Perfiles de usuario

Distinguiremos entre tres tipos de usuarios:

- Usuario Tutor: Utiliza objetos de aprendizaje para impartir conocimiento. Abre y visualiza OAs.
  Es lógico pensar que este usuario sea un profesor o tutor, que podrá explotar las posibilidades audiovisuales de los OA, que pueden estar formados por recursos como vídeos, imágenes, paginas html, etc.

- Desarrollador e-learning: es el productor de OAs, crea y edita objetos de aprendizaje, compone objetos de aprendizaje, utiliza repositorios de objetos, evalúa la calidad de sus objetos, etc.
  Es el usuario que más podrá explotar las posibilidades que proporciona LOMEditor 2006 para el manejo de OAs

3.2.2 Usos del software

Principalmente para dar soporte a la enseñanza: cursos anuales, cursos rápidos, manuales, etc. Siempre con el objetivo de organizar y compartir el conocimiento, y contando con que esta herramienta permite una actual presentación de contenidos.

3.3 Servicios – Requisitos funcionales

Sección 1:Plugin Composición, Plugin Base de Datos y Evaluación CBR

Los servicios Composición, Evaluación CBR y base de datos, constituyen plugins, por lo tanto, el usuario que quiera acceder a estos servicios, debe guardar en la carpeta de plugins aquéllos cuyas funcionalidades desee utilizar.
En cuanto a estas funcionalidades, son las mismas que ofrecía la herramienta LOMEditor 2005, por tanto, quedan fuera del alcance de esta memoria.

Sección 2: Creación de un nuevo plugin: Plugin Repositorio

Este plugin ofrece una lista de repositorios de objetos de aprendizaje, a los que se podrá acceder vía web.También permite almacenar nuevas direcciones en la lista de repositorios, asi como eliminarlas o modificar las ya existentes.

Sección 3: Ampliación de las opciones de edición y creación de OA a IMS Metadata e IMS Simple Sequencing.

Seccion3.1. Apertura de objetos de aprendizaje

La opción abrir objeto de aprendizaje, discrimina entre tres tipos de objetos. El usuario podrá elegir el tipo de objeto que desea abrir (bien IMS Metadata, bien IMS Content Package o bien IMS Simple Sequencing). Al utilizar esta opción, la herramienta hará una previa validación del OA. Para ello, validará el manifiesto del objeto (imsmanifest.xml) frente a su gramática o gramáticas asociadas:

Tipo de objeto → Gramática/s asociada/s

IMS Metadata → imsmd_v1p2p2.xsd
IMS Content Package→ imsmd_v1p2p2.xsd, imscp_v1p1.xsd
IMS Simple Sequencing→ imsmd_v1p2p2.xsd, imscp_v1p1.xsd, imsss_v1p0.xsd

La aplicación espera como entrada un objeto zipeado, que contenga en su directorio raíz el manifiesto, su/s gramática/s y resto de recursos (si los hubiera). Una vez recibido, el objeto se descomprime y se valida. Una vez validado, se visualizara su contenido en un árbol de navegación y un panel HTML.

El árbol muestra la jerarquía extraída del manifiesto, metadatos, organizaciones y recursos (en caso de que los hubiera).
Los recursos son los archivos con los contenidos didácticos, y organizaciones es la forma de organizar estos recursos.
El panel HTML, muestra el contenido real del OA. Está dividido en dos subpaneles, el izquierdo muestra las organizaciones y recursos en forma de lista, y el derecho muestra el contenido de dichos recursos, en caso de ser seleccionados de la lista.

Seccion3.2. Edición de un objeto de aprendizaje

Para los tres tipos de objetos de aprendizaje (IMS Metadata, IMS Content Package, IMS Simple Sequencing), la edición permitirá al usuario añadir nuevos hijos al árbol de contenidos, modificar los atributos y eliminar

componentes del objeto de aprendizaje. Para modificar o eliminar los atributos, hay que operar sobre el árbol de contenidos, que muestra la estructura del manifiesto asociado al objeto.

Hay dos tipos de metadatos, metadatos compuestos por otros metadatos (tienen hijos y se consideran nodos padre) y metadatos simples con un valor determindado (no pueden tener hijos y se consideran nodos hijos u hojas).

Para acceder a las posibilidades de edición siempre dependiendo del tipo de nodo, se pulsará al botón derecho del ratón sobre el nodo a editar y se desplegará un menú con las opciones disponibles. Con un nodo padre podemos:

- Insertar nodo
- Eliminar nodo
- Añadir y modificar atributos

En los nodos hijo se puede:

- Insertar información
- Modificar información
- Añadir y modificar atributos.

Una vez realizada una modificación, se reflejará automáticamente en el árbol.

Seccion3.2.1 Inserción de nuevos elementos

La inserción de nuevos elementos se gestiona mediante un menú emergente que aparece cuando el usuario pulsa botón derecho sobre un nodo del árbol de edición.

Las opciones de este menú emergente dependerán del tipo de nodo sobre el que se haya pinchado y la estructura marcada por la gramática del objeto editado.

Solo se puede insertar un elemento si el nodo es padre, y el número de elementos estará limitado por la gramática del objeto.

Seccion3.2.2. Eliminación de un elemento

Cualquier nodo, bien sea padre o hijo, dispondrá de una opción en el menú emergente que permitirá la eliminación del nodo completo.

Esto se reflejará en el árbol como la supresión del nodo y de los posibles hijos que tenga, en el caso de que sea un nodo padre.

Seccion3.2.3. Modificación de atributos

Para modificar un objeto, pincharemos con el botón derecho sobre el nodo a modificar y pulsamos la opción "modificar". Automáticamente aparecerá un panel en la parte inferior de la aplicación, en la que podemos editar o modificar el atributo seleccionado.
La gestión de esta funcionalidad de la herramienta se realizará en el panel inferior de la ventana principal de la herramienta de autoría.

Seccion3.3. Guardar cambios en el objeto

Para ello la herramienta escribe el contenido del árbol de edición de contenidos en el fichero imsmanifest.xml, que almacena la estructura del objeto de aprendizaje.

Seccion3.4. Guardar un objeto de aprendizaje en forma comprimida

Para guardar un objeto, se creará un archivo comprimido con la utilidad Zip, en el que se almacenarán en el directorio raíz los archivos, según el tipo de objeto:

Tipo de objeto → Archivos

IMS Metadata →imsmanifest.xml, imsmd_v1p2p2.xsd
IMS Content Package→ imsmanifest.xml, imsmd_v1p2p2.xsd, imscp_v1p1.xsd
IMS Simple Sequencing→ imsmanifest.xml, imsmd_v1p2p2.xsd, imscp_v1p1.xsd, imsss_v1p0.xsd

y el resto de recursos (imágenes, videos, documentos pdf, etc.), si los hubiera.

Seccion3.5. Creación de un nuevo objeto de aprendizaje

La opción nuevo objeto de aprendizaje, discrimina entre tres tipos de objetos. El usuario podrá elegir el tipo de objeto nuevo que desea (bien IMS Metadata, bien IMS Content Package o bien IMS Simple Sequencing). Al utilizar esta opción, la herramienta utilizara una de las plantillas que tiene según el tipo de objeto.

Tipo de objeto → Se crean por defecto los archivos

IMS Metadata →imsmanifest.xml, imsmd_v1p2p2.xsd
IMS Content Package→ imsmanifest.xml, imsmd_v1p2p2.xsd, imscp_v1p1.xsd
IMS Simple Sequencing→ imsmanifest.xml, imsmd_v1p2p2.xsd, imscp_v1p1.xsd, imsss_v1p0.xsd

El árbol muestra la jerarquía extraída del manifiesto, metadatos, organizaciones y recursos.

manifest
--- metadata
--- organizations
--- resources

El panel HTML, muestra el contenido del OA. Está dividido en dos subpaneles, el izquierdo muestra las organizaciones y recursos en forma de lista, y el derecho muestra una plantilla HTML, ya que los objetos nuevos no tienen recursos.

A partir de estos objetos mínimos, que se mostrará en el árbol de edición de contenidos tras ser generado, el usuario podrá generar el objeto de aprendizaje que desee, utilizando el resto de opciones de la herramienta de autoría.

Seccion3.6. Ayuda

Seccion3.6.1. Manual de usuario

El usuario puede consultar el manual de ayuda sobre el manejo de la herramienta (véase Apéndice A), que contiene todas las posibilidades incorporadas en la nueva versión de la misma.

El manual explica el uso de la barra de herramientas y los menús que provocan los eventos de la aplicación.

Seccion3.6.2. Información corporativa

La herramienta dispondrá de acceso a un panel de información corporativa, detalles de la versión de la herramienta, desarrolladores, etc.

Seccion3.6.3. Información de elemento

El panel inferior de la parte derecha de la pantalla principal de la herramienta, muestra información sobre el nodo del árbol que este pulsado con el ratón, con objetivo de facilitar la creación de contenidos a los usuarios.

3.4 Servicios potenciales (requisitos emocionantes)

3.4.1 Creación de un espacio Web en el que promocionar la herramienta

Se creará un espacio web para dar a conocer la herramienta. Las opciones que ofrecerá este espacio son:

- Descargar la herramienta: código fuente
- Descargar el manual de usuario
- Descargar la documentación del proyecto
- Descargar la presentación del proyecto

3.5 Restricciones y limitaciones – Requisitos no funcionales

3.5.1 Requerimientos hardware

3.5.1.1 Mínimos

- Portátil con pantalla TFT 14.1” o mayor con área de pantalla >= 1024 x 768
- Monitor 15” con área de pantalla >= 1024 x 768
- Espacio libre en disco: 120 MB

- Intel Pentium® 3 (o equivalente) a 800 MHz
- 256 MB de RAM
- Lector de CD-ROM 16x, ratón y teclado
- Modem 56k

3.5.1.2 Óptimos

- Portátil con pantalla TFT 14.1” con área de pantalla = 1024 x 768
- Monitor 17” con área de pantalla = 1024 x 768
- Espacio libre en disco: 175 MB
- Pentium® IV, Pentium® Centrino o AMD® 1250
- 1024 MB de RAM
- Lector de CD-ROM 32x, ratón y teclado
- Router wi-fi

3.5.2 Requerimientos software

3.5.2.1 Mínimos

- Microsoft® Windows® 98 / ME / NT4.0 /2000 / XP
- Máquina virtual de Java con JDK 1.4.2 o posterior
- Navegador Microsoft® Internet Explorer 5 o posterior

3.5.2.2 Óptimos

- Microsoft® Windows® XP
- Máquina virtual de Java con JDK 1.5
- Navegador Microsoft® Internet Explorer 6

3.5.3 Interfaces externas

La aplicación interactúa con elementos externos:

Sistema operativo:

La herramienta LOMEditor 2006 hara llamadas al sistema operativo en aquellos casos en los que sea necesario invocar algun programa externo para mostrar contenidos de los objetos de aprendizaje, esto se debe a las restricciones impuestas por el API de Java, que no soporta todos los formatos de documentos. En estos casos, la herramienta lanzará el navegador Microsoft® Internet Explorer, y el recurso será mostrado allí.

Periféricos:

Se requieren ratón, y teclado para explotar las funcionalidades de la herramienta.

Máquina virtual de JAVA:

LOMEditor 2006 esta implementado en JAVA. La máquina virtual de java (desarrollada por Sun Microsystems) ejecuta el código resultante de la compilación de nuestro programa.

El usuario debe tener instalada la version del JDK 1.4.2 o superior, que contiene librerías y funciones avanzadas que consume LOMEditor 2006.

3.5.4 Documentación de usuario

Manual de usuario

Se creará un manual para el usuario de gran contenido visual, con capturas de pantalla para cada funcionalidad añadida en esta version de la herramienta. Se comenta asi mismo paso a paso las acciones a realizar por parte del usuario hasta conseguir aplicar al objeto de aprendizaje cada una de las funcionalidades de la herramienta.

En el manual se describen las funciones del menu, asi como las funciones de los botones de la barra de herramientas. También se incluyen ejemplos paso a paso de algunas de las funcionalidades básicas como crear un OA, bien sea IMS Metadata, IMS Content Package o IMS Simple Sequencing.

3.5.5 Requerimientos de desarrollo

Equipo de desarrollo

El profesor y director del proyecto, dirigirá al grupo de desarrolladores, formado por tres miembros, alumnos de la Facultad de Informática de la UCM.

Se harán reuniones periódicas para la gestión de requisitos de la herramienta y el control de calidad del software, actuando en estas el profesor y director como un cliente de software real.

Herramientas para el desarrollo del software

- Borland JBuilder® 2006 Enterprise

  Ofrece un entorno de desarrollo óptimo para la creación de interfaces de usuario asi como para la gestión de proyectos Java

- MySQL® Server 4.0

  Se utilizará como servidor de bases de datos. Es imprescindible para que el módulo de evaluación de objetos de aprendizaje funcione correctamente, ya que el CBR que utiliza este módulo implementará su base de casos en una base de datos relacional MySQL.

- MySQL® Control Center

Esta aplicación, que MySQL ofrece libremente a sus usuarios, es utilizada principalmente como interfaz amigable sobre la base de datos. El usuario de LOMEditor no requerirá de este programa, ya que el trabajo sobre la base de datos es transparente para el. Sin embargo, el desarrollador del sistema encuentra en MySQL Control Center una herramienta muy útil para su trabajo. Será, por tanto, utilizada principalmente en labores de desarrollo y pruebas sobre la base de datos.

- Macromedia® DreamweaverMX® 2004

  Ha sido utilizado para editar archivos HTML, XML y XSD.

- Microsoft® Word 2002

  Ha sido utilizado para editar textos, concretamente en las tareas de documentacion y edicion de manuales.

3.6 Métodos de prueba

3.6.1 Tipos de pruebas

El grupo de derarrolladores de LOMEditor 2006 ha estado formado por tres personas

En un principio el grupo se constituyó en un grupo de diseñadores que actuaron conjuntamente dando forma a la arquitectura del sistema.

Despues el grupo se contituyó en grupo de desarrollo para lo cual era necesario repartir tareas, implementar funcionalidades por separado y posteriormente integrarlas. Esto da lugar a un amumento potencial de errores.

Se definió entonces un plan de pruebas, que comienzan por las pruebas que cada desarrollador realiza sobre la funcionalidad que ha implementado.

Tras la integración en la aplicación principal, vuelven a realizarse baterías de pruebas.

Finalmente, ante el director del proyecto se realizan las pruebas de certificación.

1. **Pruebas Personales**

Realizadas por cada desarrollador sobre la funcionalidad de software que ha implementado. De no ser superadas, no se dará por válido el módulo implementado y tendra que ser depurado y probado de nuevo.

Han de documentarse las pruebas y los errores encontrados.

2. **Pruebas tras la integración**

Superadas las pruebas personales, e integradas las funcionalidades implementadas por cada desarrollador, se pasa a realizar la prueba tras la integración.

Con esta nueva batería de pruebas se intenta verificar que las funcionalidades añadidas no producen errores en la versión global del producto.

3. **Pruebas de certificación**

Son las pruebas que supervisa el director del proyecto, una vez superadas, se valida el prototipo.

3.6.2 Método de actuación en caso de pruebas no superadas

Si el error aparece en el periodo de pruebas personales, es el desarrollador de dicha funcionalidad el que tiene la responsabilidad de subsanar dicho error, aunque esto suponga un retraso en la integración y consecuentemente en la entrega del prototipo.

Este tipo de retrasos se tienen en cuenta a la hora de planificar una entrega, consecuentemente, se consideran de bajo riesgo.

En caso de que el error aparezca en el periodo de pruebas tras la integración será el conjunto de los tres desarrolladores los que intentarán subsanar los fallos, de no ser posible, la funcionalidad que da el error habrá de ser implementada de nuevo.

Se actuará de la misma manera en caso de que el error se de en la fase de pruebas de certificación.

3.6.3 Resultados esperados

Con este proceso de pruebas y certificación se busca principalmente poder ofrecer al cliente un producto que cumpla sus expectativas y garantice su funcionamiento.

Más concretamente buscamos construir una herramienta altamente estable, que ofrezca un elevado nivel de fiabilidad al usuario en cualquier situación de uso en la que éste se pueda encontrar.

A nivel de diseño, tras una serie de prototipos, el objetivo es que LOMEditor alcance a satisfacer los deseos del cliente en este aspecto, sin que por ello su funcionalidad o robustez se vean en absoluto afectadas.

A nivel de comportamiento se busca y espera obtener un producto a la altura de sus competidores en este sector, superando incluso a estos últimos en cuanto a funcionalidad y servicios.

4 CASOS DE USO DEL SISTEMA

Se van a detallar los casos de uso de las funcionalidades añadidas al LOMEditor 2005.

Casos de uso:

1. Abrir objeto de aprendizaje
2. Insertar secuenciación en un Objeto
3. Nuevo objeto de aprendizaje
4. Modificar el valor de los atributos de secuenciación
5. Acceder a un repositorio de objetos via web
6. Añadir un nuevo repositorio
7. Modificar un repositorio
8. Eliminar un repositorio

4.1 Caso de uso "Abrir objeto de aprendizaje"

Objetivo en concreto	Apertura manual de un objeto de aprendizaje que se encuentra guardado en formato zip. Elegimos el tipo de objeto(IMS Content package, IMS Simple sequency, IMS metadata), descomprimiremos el fichero y mostraremos el objeto en la herramienta.	
Entradas	Objeto de aprendizaje del tipo seleccionado en formato zip, directorio destino.	
Precondiciones	El fichero zip debe contener un objeto de aprendizaje del tipo seleccionado correcto y la carpeta destino debe existir.	
Salidas	Ficheros descomprimidos en la carpeta destino.	
Poscondición si éxito **Poscondición si fallo**	El objeto de aprendizaje se descomprime en el directorio destino y se muestra su contenido en la herramienta. Se muestra al usuario una ventana de error.	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario accede al Menú Principal, Archivo, Abrir, Abrir Objeto de Aprendizaje ó al botón Abrir de la barra de herramientas.
	2	El sistema muestra un cuadro de diálogo para seleccionar el tipo de objeto de aprendizaje a abrir.
	3	El sistema mostrará un cuadro de diálogo para navegar por el equipo del usuario.
	4	Seleccionamos en el cuadro de diálogo el fichero .zip que contiene el objeto de aprendizaje que deseamos abrir.
	5	Seleccionamos la carpeta destino donde vamos a descomprimir el contenido del objeto de aprendizaje.
	6	El sistema analiza el objeto de aprendizaje, comprobando su corrección. Si lo valida, se muestra.
	7	El sistema mostrará un árbol de directorios donde quedará reflejado el objeto de aprendizaje, facilitando de esta forma su edición.
Secuencia alternativa	**Paso**	**Acción**
	8	Si el objeto de aprendizaje no es válido, el sistema no mostrará el árbol de representación, sino que alertará de lo ocurrido mediante un mensaje de error.

4.2 Caso de uso "Insertar Secuenciacion en un Objeto"

Objetivo en concreto	Aplicar a algún nodo del árbol secuenciación, para poder utilizar esa funcionalidad	
Entradas	Modelo Java del objeto de aprendizaje abierto.	
Precondiciones	Debe existir un objeto de aprendizaje abierto del tipo Simple Secuency.	
Salidas		
Poscondición si éxito	Se le asigna a un nodo la propiedad de secuenciación según la haya configurado el usuario, y se muestra gráficamente en la pantalla.	
Poscondición si fallo	Se muestra al usuario una ventana de error, no habiéndose insertado la secuenciación..	
Actores	Usuario y el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario sitúa el ratón sobre el nodo del árbol donde desea actuar y presiona el botón derecho.
	2	El sistema mostrará un menú emergente donde se despliegan las diferentes opciones para este nodo, extraídas del modelo generado durante la validación. Se seleccionara "Sequency"
	3	En el panel de abajo a la derecha nos aparecerán todas las opciones de secuenciación. Se configurará como el usuario lo desee.
	4	El sistema generará un nuevo elemento y lo insertará como hijo del nodo seleccionado por el usuario, y por tanto, del elemento que este último contiene y representa.
	5	Se mostrará gráficamente el resultado.
Secuencia alternativa	**Paso**	**Acción**
	3	El objeto no admite secuenciación y no se mostrará la opción. Tiene que ser un objeto IMS Simple sequency.

4.3 Caso de uso "Nuevo Objeto de aprendizaje"

Objetivo en concreto	Crear un nuevo objeto de aprendizaje, vacío, sin datos, para luego ser completado por el usuario	
Entradas	Modelo Java de un objeto de aprendizaje sin información.	
Precondiciones	Programa abierto y funcionando.	
Salidas	Objeto de aprendizaje al desnudo.	
Poscondición si éxito **Poscondición si fallo**	El objeto es mostrado en el árbol, en el panel izquierdo de la aplicación. Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa en archivo nuevo, o en el botón de nuevo de la barra de herramientas.
	2	El sistema muestra una venta de selección en la que se seleccionará el tipo de objeto de aprendizaje que se quiere crear.
	3	El sistema lanza un cuadro de diálogo en el que se seleccionara el directorio en el que se quiere guardar el nuevo objeto de aprendizaje.
	4	Se muestra un objeto sin datos a completar.
Secuencia alternativa	**Paso**	**Acción**
	4	Si el directorio destino no existe se muestra un mensaje de error.

4.4 Caso de uso "Modificar el valor de los atributos de secuenciacion"

Objetivo en concreto	Modificar el valor de un atributo de la secuenciación de un nodo	
Entradas	Objeto de aprendizaje con secuenciacion	
Precondiciones	Un objeto abierto de tipo IMS Simple Secuency	
Salidas	Objeto modificado	
Poscondición si éxito **Poscondición si fallo**	El atributo se modifica en el árbol. Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa el botón derecho del ratón sobre el nodo de secuenciacion a modificar.
	2	El sistema muestra el valor de los atributos en el panel inferior
	3	El usuario modifica el atributo y pulsa aceptar
	4	Se guarda el atributo modificado y se muestra en el árbol
Secuencia alternativa	**Paso**	**Acción**
	4	Si el valor del atributo no está en el rango de posibles valores se mostrará un error.

4.5 Caso de uso "Acceder a un repositorio de objetos vía web"

Objetivo en concreto	Acceder a una página web en la que haya objetos de aprendizaje para descargar y usar en la aplicación.	
Entradas		
Precondiciones	Plugin repositorio cargado en la aplicación	
Salidas	Repositorio web	
Poscondición si éxito **Poscondición si fallo**	Se abre un navegador con la página seleccionada Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa en el menú repositorio y luego lanzar repositorio.
	2	El sistema muestra una ventana con los repositorios almacenados
	3	El usuario pulsa sobre uno de los repositorios
	4	Se abre un navegador con el repositorio seleccionado
Secuencia alternativa	**Paso**	**Acción**
	4	Si no hay ningún repositorio guardado no se podrá abrir ninguna web

4.6 Caso de uso "Añadir una direccion nueva a la lista de repositorios"

Objetivo en concreto	Añadir un nuevo repositorio a la base de datos de repositorios web	
Entradas		
Precondiciones	Plugin repositorio cargado en la aplicación	
Salidas	Repositorio web	
Poscondición si éxito **Poscondición si fallo**	Se añade una nueva referencia web a un repositorio. Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa en el menú repositorio y luego lanzar repositorio.
	2	El sistema muestra una ventana con los repositorios almacenados
	3	El usuario pulsa el botón añadir.
	4	El sistema muestra una nueva ventana para introducir el nuevo repositorio.
Secuencia alternativa	**Paso**	**Acción**
	4	Si es una direccion erronea se muestra un mensaje de error

4.7 Caso de uso "Modificar una direccion de la lista de repositorios"

Objetivo en concreto	Modificar la dirección web de un repositorio de la base de datos de repositorios web	
Entradas		
Precondiciones	Plugin repositorio cargado en la aplicación	
Salidas	Repositorio web	
Poscondición si éxito **Poscondición si fallo**	Se modifica una referencia web a un repositorio. Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa en el menú repositorio y luego lanzar repositorio.
	2	El sistema muestra una ventana con los repositorios almacenados
	3	El usuario selecciona el repositorio a modificar y pulsa el botón modificar.
	4	El sistema muestra la dirección web modificada
Secuencia alternativa	**Paso**	**Acción**
	4	Si no hay ninguna dirección web no se puede eliminar ninguna

4.8 Caso de uso "Eliminar una direccion de la lista de repositorios"

Objetivo en concreto	Eliminar un repositorio de la base de datos de repositorios web	
Entradas		
Precondiciones	Plugin repositorio cargado en la aplicación	
Salidas	Repositorio web	
Poscondición si éxito **Poscondición si fallo**	Se elimina una referencia web a un repositorio. Se muestra al usuario una ventana de error.	
Actores	Usuario y sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El usuario pulsa en el menú repositorio y luego lanzar repositorio.
	2	El sistema muestra una ventana con los repositorios almacenados
	3	El usuario selecciona una direccion web y pulsa el botón eliminar.
	4	El sistema elimina la referencia web de la base de datos de direcciones a repositorios.
Secuencia alternativa	**Paso**	**Acción**
	4	Si no hay ninguna no se puede eliminar nada.

5 ARQUITECTURA DEL SISTEMA

En cuanto a la arquitectura que ha de tener un sistema de estas características, es conocido que la calidad del software depende directamente de los estándares y patrones de diseño software seguidos. Esto será determinante para la modularidad, portabilidad y reusabilidad y mantenimiento de la herramienta.

La heramienta LOMEditor 2005, de la que partiamos siguía el patrón de diseño Modelo-Vista-Controlador (Model View Controller)

Modelo Vista Controlador (MVC) es una arquitectura de software que separa el modelo de datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes.

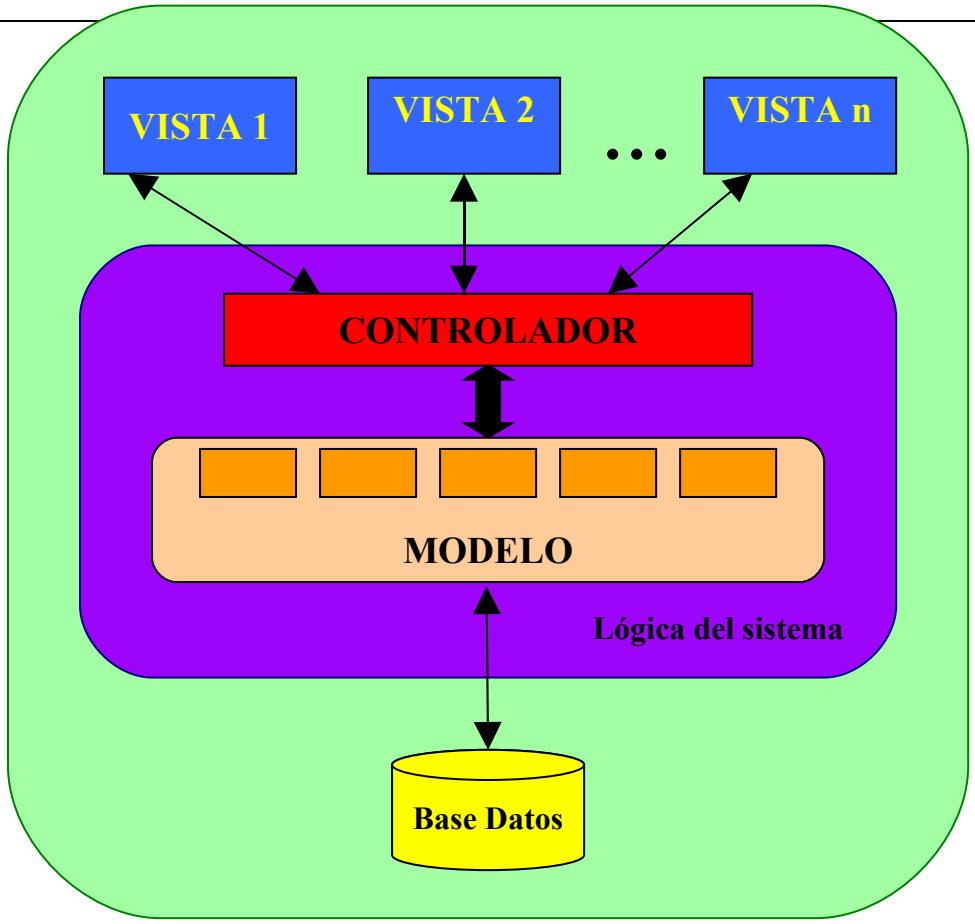
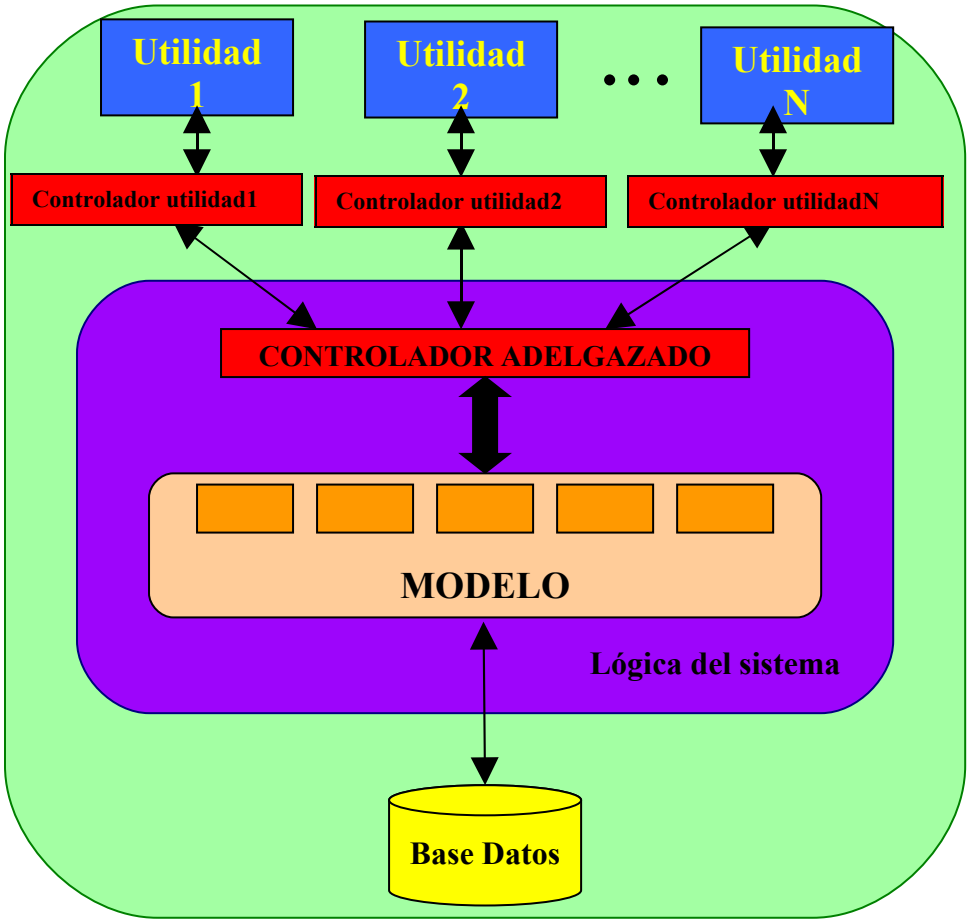


Figura 5.1.- Esquema diseño modelo vista controlador

LOMEditor 2006 ha tratado de ser una herramienta modular adaptada a las ultimas tendencias en diseño de aplicaciones software, una de estas tendencias es el desarrollo de aplicaciones que soporten plugins. Como ya hemos comentado anteriormente, partíamos de una herramienta con arquitectura MVC, en la que la lógica de la aplicación está concentrada en el controlador. Deciden separarse dos de las utilidades de la herramienta, que están modularizadas cada una en su respectivo paquete de clases, pero que a su vez están referenciadas desde el controlador.

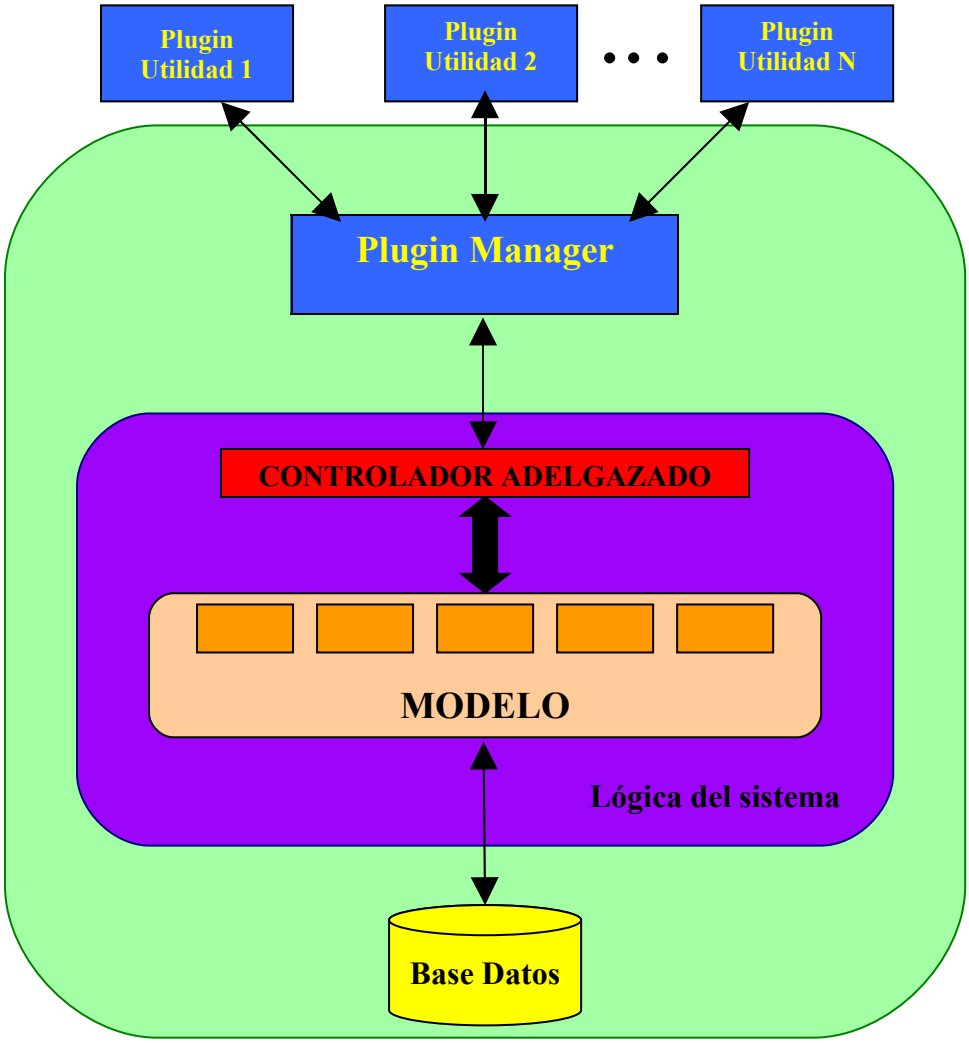
Se crean dos controladores, uno para referenciar a cada módulo de utilidad, a su vez, a cada utilidad se le añade una clase que implementa el interfaz de los plugins, esta clase hará referencia a los métodos del controlador de cada utilidad.



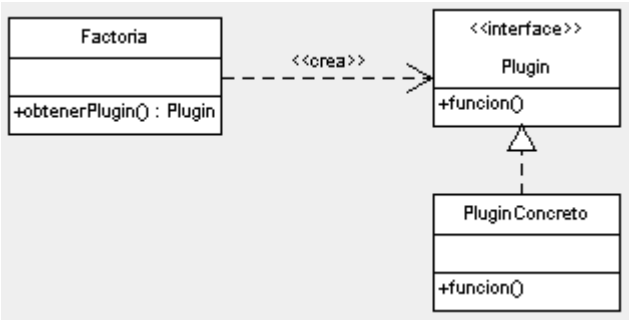
Cada paquete de clases que contiene una utilidad, se comprime en un fichero *Jar*, que se colocará en un directorio determinado (accesible para la aplicación)

Por otro lado, se define una clase manejadora de plugins que, al arrancar, "investiga" en dicho directorio si se dispone de algún

fichero "enchufable" y si es así lo analiza e incluye a las opciones de gestión de los objetos de aprendizaje.

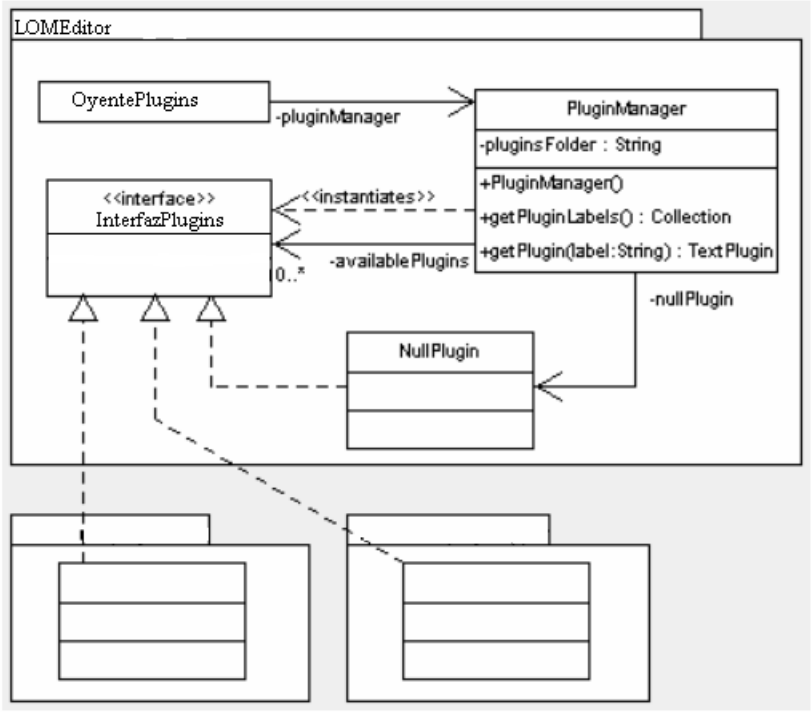


Los plugins, deben implementar un interfaz común:



de modo que desde la aplicación principal, se invocará a un método (que implementan todos los plugins) pasándole ciertos parámetros (en concreto una cadena con las opciones) de modo que cada plugin ejecute la acción que se le demanda desde el interfaz.

Al arrancar la aplicación, el Plugin Manager busca los plugins en el sistema, una vez encontrado uno, se analiza el fichero *Jar*, que ha de contener un manifiesto en el que se especifiquen el nombre del plugin, y las opciones de gestión del objeto que permite. En función a estas opciones, un nuevo menú (dinámicamente) en el interfaz de la aplicación que permita invocar a cada una de estas nuevas funcionalidades.



6 DETALLES DE IMPLEMENTACIÓN

6.Sección 1: Transformación de los módulos de Composición y Evaluación CBR y BD a Plugins

Como ya hemos comentado en el apartedo anterior, para hacer de LOMEditor una herramienta a base de plugins, se parte de una primera versión (LOMEditor 2005) con arquitectura modelo – vista –controlador, en la que la lógica de la aplicación está concentrada en el controlador.

Las funcionalidades de la aplicación están agrupadas en paquetes de clases java, explotaremos esto para crear nuestros plugins.

Separaremos las funcionalidades de Composición y evaluacion CBR (cada una de ellas en un paquete de clases)

Estas utilidades estan comunicadas con la aplicación por medio del controlador. Adelgazamos el controlador sacando todas las llamadas a los paquetes que queremos separar, y creamos 2 nuevos controladores, uno dentro de cada paquete, que será un futuro plugin.
Cada paquete de clases que contiene una utilidad, se comprime en un fichero *Jar*, que se colocará en un directorio determinado (accesible para la aplicación).

Empaquetamiento en Jar, la importancia del manifiesto

Cuando invocamos un plugin lo hacemos a través de la interfaz InterfazPlugins y no nos importa lo complejo que sea el procesamiento llevado a cabo o cuantas clases y paquetes se ven involucrados. Todo ese código debe estar empaquetado en un solo fichero para facilitar la instalación – reducida a copiar el fichero en el directorio designado-.

Para empaquetar usaremos la herramienta Jar que se incluye en el kit de desarrollo de J2SE.

Dentro de un jar debe incluirse un manifiesto -Manifest.mf- con información sobre los contenidos del fichero. Lo que a nosotros nos interesa es que se trata de un documento de texto donde podemos incluir claves –englobadas en secciones- a las que podemos acceder en nuestros programas a través del API de Java –paquetes java.util.jar-.

El manifiesto de nuestros plugins debe incluir obligatoriamente una sección nueva con dos claves -necesarias para registrarlos-. Si no se cumple este requisito, el fichero jar no se registrará.

La modularización de la aplicación, nos va a permitir contar con una serie de ventajas que facilitarán tanto el uso como la escalabilidad del programa:

- En cada momento podemos elegir las funcionalidades a cargar, de modo que si solamente queremos editar un objeto, no debemos cargar ningún plugin y la aplicación sería mucho más sencilla.

- Gracias a la versión de desarrollo podemos añadir funcionalidades a la aplicación, sin tener que conocer a fondo el código de la misma. Simplemente hay que ver la estructura del interfaz, y conocer los datos de los que dispone cada plugin, para poder implementar lo que queramos que haga ese módulo.

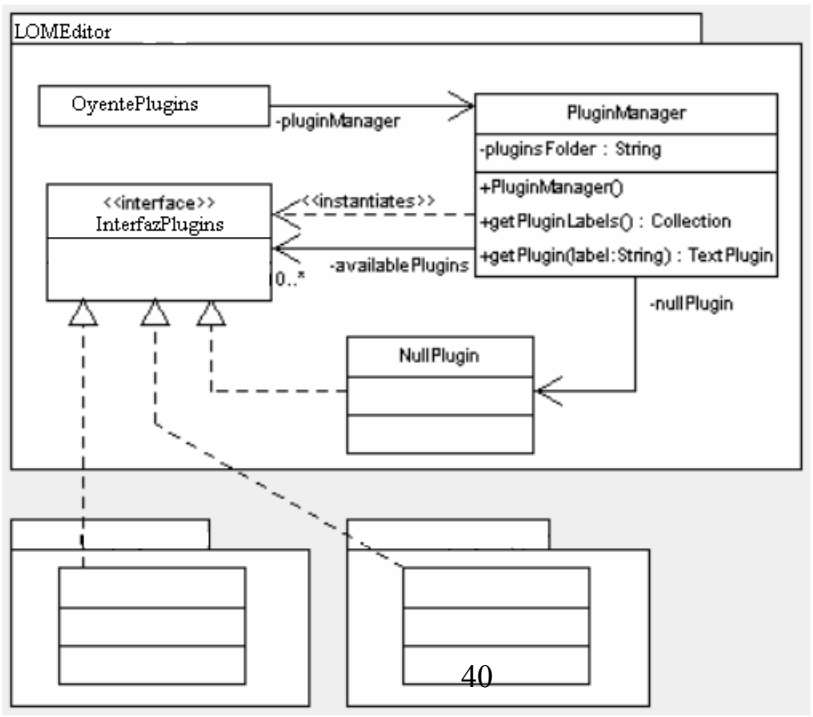
El interfaz a implementar es el siguiente:

```
public interface InterfazPlugins {

      public void lanzar(String opciones) throws
InterfazPluginException;

      public void pasarInterfaz(MainFrame interfaz);

}
```

El método lanzar(), invoca al plugin, pasándole las opciones en el caso de que el plugin tenga mas de una funcionalidad.

El método pasarInterfaz(), sirve para enviarle el interfaz de la aplicación al plugin para que utilice los datos que le hagan falta

Al arrancar la aplicación, el Plugin Manager busca los plugins en el sistema, una vez encontrado uno, se analiza el fichero *Jar*, que ha de contener un manifiesto en el que se especifiquen el nombre del plugin, y las opciones de gestión del objeto que permite. En función a estas opciones, un nuevo menú (dinámicamente) en el interfaz de la aplicación que permita invocar a cada una de estas nuevas funcionalidades.



6.Sección 2: Creación de un nuevo plugin: Plugin Repositorio

Para el desarrollo de nuevos plugins se dispone de una versión de la aplicación, llamada Versión de Desarrollo, en la cual se implementan los plugins, se prueban y posteriormente se empaquetan para ser "enchufados" en la versión que se presenta al usuario.

6.Sección 3: Ampliación de las opciones de edición y creación de OA a IMS Metadata e IMS SS.

Esta parte de la implementacion, ha ampliado el núcleo central de la aplicación, añadiendo a las funcionalidades de LOMeditor 2005, la posibilidad de editar IMS Metadata e IMS Simple Sequencing.

6. Sección 3.1 Validación de objetos de aprendizaje IMS_MD e IMS_SS

Los objetos de aprendizaje son validados por LOMEditor 2006, para asegurar que siguen las gramáticas de los estándares.El proceso de validación incluye:

6. Sección 3.1.1 Apertura y parseo del objeto de aprendizaje

Utilizaremos el paquete JDOM es un API de JAVA para leer, crear y manipular documentos XML y XSD.

Utilizaremos la utilería ZIP del API de Java para descomprimir el objeto de aprendizaje. Se recolectarán los datos necesarios como por ejemplo las rutas al objeto, en la clase Objeto de Aprendizaje.

Con JDom generaremos un documento que contendrá, en forma de árbol, la información de la gramática en la que se basa el objeto de aprendizaje. Se recorre esta gramática y se van rellenando las estructuras que necesitaremos tanto para validar como para las futuras ediciones sobre este objeto de aprendizaje. Esto lo vemos en el siguiente punto

Tendremos una variable que indica el tipo de objeto que se ha abierto, para procesarlo con su gramática correspondiente, recordemos que:

Tipo de objeto → Gramática/s asociada/s

IMS Metadata → imsmd_v1p2p2.xsd

IMS Content Package→ imsmd_v1p2p2.xsd, imscp_v1p1.xsd
IMS Simple Sequencing→ imsmd_v1p2p2.xsd, imscp_v1p1.xsd, imsss_v1p0.xsd

6. Sección 3.1.2 Construcción del modelo de validación

Tras parsear mediante JDom las gramáticas correspondientes, según el tipo de OA, construiremos el modelo de validación.

Cada etiqueta de las gramáticas reflejadas en las xsd es procesada y analizada por un determinado grupo de clases

La clase ManifestValidator almacenará las etiquetas en las clases correspondientes.

ManifesValidator va procesando el manifiesto:

-Si es Metadata, invoca a la clase MetadataValidator, que valida la parte de los metadatos

-Si es CP, MetadataValidator valida a parte de los metadatos y el resto del CP lo valida ManifestValidator

-Si es SS, MetadataValidator valida a parte de los metadatos y el resto del CP lo valida ManifestValidator y la parte de SS lo valida SimpleSequencingValidator.

6.Sección 3.1.3 Algoritmo de validación

Las gramáticas de cada tipo de objeto contienen la estructura que deben seguir los datos del manifiesto. El algoritmo, por tanto, realizará una comprobación por medio de encaje de patrones entre las gramáticas que definen la estructura del manifiesto y el contenido del mismo.

El algoritmo de validación lee el archivo xsd de la gramática correspondiente que define la estructura del manifiesto, y según lo vamos recorriendo vamos almacenando su contenido (parsing del manifiesto).

El algoritmo de validación basa su funcionamiento en dos pilas que irán guardando las etiquetas de los elementos, según se vayan leyendo el archivo xml y la información almacenada previamente sobre las xsd del objeto de aprendizaje.

En cada una de estas pilas se irán guardando las etiquetas de los elementos que se vayan leyendo de cada uno de los archivos comentados anteriormente. Si se lee una etiqueta de cierre del elemento se introducirá en la pila un objeto de tipo String que contenga “TERMINADO”. De esta forma podemos comparar los objetos que sacamos de ambas pilas sabiendo que son hijos de un mismo padre, es decir, que cuando se lee TERMINADO en alguna de las pilas, significará que el elemento no tendrá más hijos. Por ello, si una pila lee terminado antes que la otra, será porque una de ellas tiene más hijos de ese elemento que la otra pila.

Para facilitar la comprensión del algoritmo de validación a continuación se muestra un ejemplo, indicando el estado de las pilas de validación en cada momento. Para no hacer este desarrollo

demasiado complejo, tan sólo se muestra la validación de un objeto sencillo que no contiene metadatos. Por tanto, la validación consistirá en el encaje de patrones entre el fichero imsmanifest.xml y su gramática asociada, en este caso, imscp_v1p1.xsd, que se muestran a continuación.

imsmanifest.xml

```
<?xml version="1.0"?>
<manifest identifier="MANIFEST1"
xmlns="http://www.imsglobal.org/xsd/imscp_v1p1"
xmlns:imsmd="http://www.imsglobal.org/xsd/imsmd_v1p2"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.imsglobal.org/xsd/imscp_v1p1
imscp_v1p1p3.xsd http://www.imsglobal.org/xsd/imsmd_v1p2
imsmd_v1p2p2.xsd ">
  <organizations>
    <organization identifier="TOC1">
      <title>xR 500 Robotic Arm</title>
      <item identifier="ITEM1" identifierref="RESOURCE1">
        <title>Overview</title>
      </item>
    </organization>
  </organizations>
  <resources>
    <resource identifier="RESOURCE1" type="webcontent"
href="robot2/background/robotintro.htm">
      <file href="robot2/background/robotintro.htm"/>
    </resource>
  </resources>
</manifest>
```

imscp_v1p1.xsd

Este fichero almacena la estructura del manifiesto según la especificación IMS Content Package v1.1.4. Previa a la ejecución del algoritmo de validación, la estructura del manifiesto definida en este archivo habrá sido almacenada en las clases implementadas a estos efectos, tal y como se indica en el punto 6.1.2.2 Construcción del modelo de validación, en esta misma sección.

Funcionamiento del algoritmo:

Inicialmente se cargarán las raíces correspondientes a ambos documentos, las pilas contendrán, por tanto:

Figura 6.1.2.1.- Paso 1

Al contener la cima dos Elementos se comprueba que sean iguales y se validan sus atributos. Si la validación de los atributos es correcta, se elimina el elemento de la cima de la pila XML y el de la pila XSD, sólo si se trata de un elemento que puede aparecer una única vez en el documento. Como manifest sólo puede aparecer una vez, según esta especificación, se eliminará el elemento cima de ambas pilas y se introducirá "TERMINADO". A continuación, se introducirán en la pila XML los hijos del elemento manifest y en la pila XSD los posibles hijos del mismo, según la gramática que se encuentra almacenada en el modelo de validación.

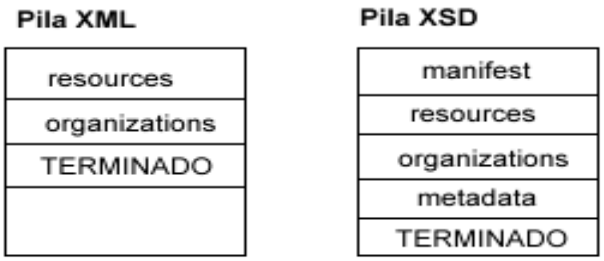


Figura 6.1.2.2.- Paso 2

El siguiente paso en la validación se encontraría con dos elementos en las cimas de ambas pilas, pero de naturaleza diferente. El algoritmo, comprobará entonces que el elemento de la cima de la pila XSD no es obligatorio. Como los submanifiestos no son obligatorios según la gramática utilizada se eliminará de la cima de la pila XSD el elemento manifest.

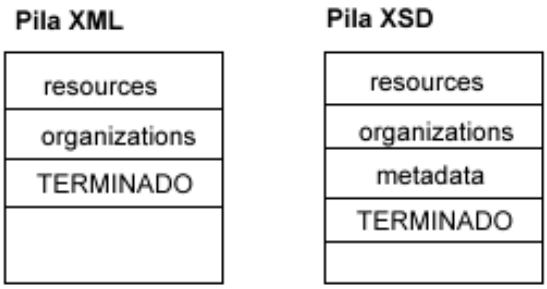


Figura 6.1.2.3.- Paso 3

Llegados a este punto, se procedería a la validación de los elementos resources que se encuentran en la cima de ambas pilas. De nuevo, al tratarse de elementos complejos, con hijos, y del mismo tipo, se procedería a comprobar que los atributos del elemento resources del manifiesto son correctos.

Como en este caso los atributos del elemento resources son correctos, se continuaría evaluando los hijos de este elemento. Con esta finalidad, se introduciría el código "TERMINADO" y los elementos de hijos de resources según la gramática y según el manifiesto. Como el elemento resources sólo puede aparecer una vez dentro del manifiesto se eliminará de la cima de ambas pilas, quedando éstas como indica la siguiente figura.

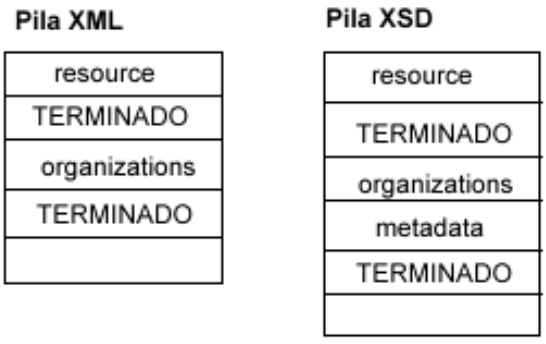


Figura 6.1.2.4.- Paso 4

Al encontrarse con la situación de la figura anterior, el objeto procedería de nuevo a la evaluación de los hijos de los elementos de la cima de la pila, resource, eliminando el elemento en la cima de la pila XML, pero no haciéndolo en la pila XSD ya que este elemento puede aparecer varias veces, según la gramática que se está considerando.

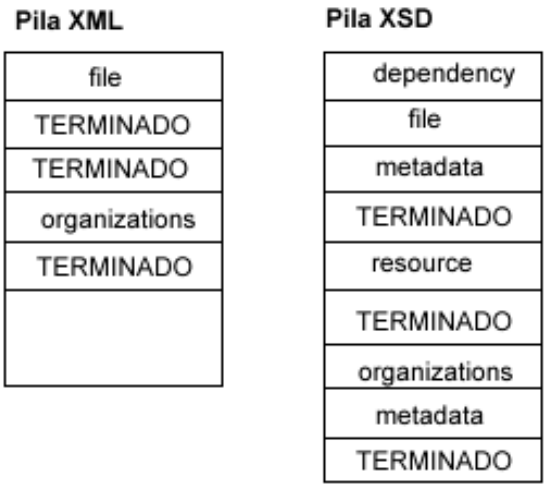


Figura 6.1.2.5.- Paso 5

En este punto el algoritmo, detectaría que dependency no es un elemento obligatorio según la estructura definida mediante la xsd, por lo que sacaría el elemento de la cima de la pila XSD y continuaría evaluando. Después comprobaría la correspondencia entre los dos elementos file que se encontraría en la cima de la pila. Puesto que el elemento file del xml de ejemplo es correcto, después de una serie de pasos nos encontraríamos en la siguiente situación:

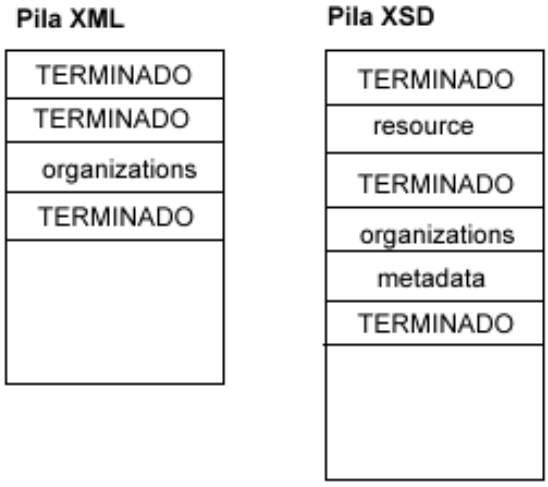


Figura 6.1.2.6.- Paso 6

Los "TERMINADO" que se encuentran en la cima de ambas pilas serían desapilados en este punto.

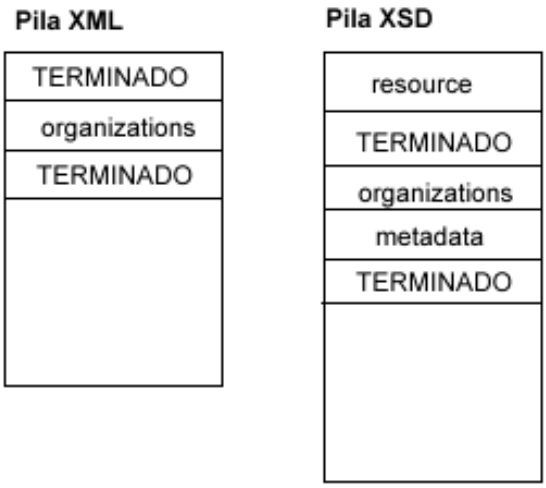


Figura 6.1.2.7.- Paso 7

Al encontrarse en la situación que indica la figura anterior el algoritmo eliminaría el elemento resource de la cima de la pila XSD ya que, aunque es un elemento obligatorio, ya ha aparecido con anterioridad en el fichero xml.

El proceso continuaría hasta que el algoritmo encontrase las dos pilas vacías, ya que en el caso de ejemplo el manifiesto valida la xsd que define su estructura.

6. Sección 3.2 Construcción del modelo de datos

Una vez finalizado el proceso de validación explicado anteriormente, la herramienta se encuentra en disposición de construir el modelo de datos con el que va a trabajar a partir de ahora.

Las acciones que dan lugar a la construcción del modelo de datos, han podido aprovecharse íntegramente de la herramienta LOMEditor2005, por lo tanto únicamente las citaremos:

1. *Modelo de validación*
2. *Documento del manifiesto*
3. *Árbol del objeto de aprendizaje*

6. Sección 3.3 La validación, el modelo de datos y la secuenciación

Para hacer que todo esto sea aplicable a la parte se Secuenciación, veamos cómo hemos añadido la secuenciación a la aplicación ya existente

En primer lugar pensamos incluir la secuenciación por medio de software, es decir, del mismo modo que se reconocía y añadía la etiqueta lom en la aplicación que nuestros compañeros desarrollaron el año pasado. Pero se nos ocurrió otra opción más sencilla y potente que no solo nos permitiría añadirle secuenciación sino cualquier otro conjunto de funcionalidades con tan solo cambiar el documento xml correspondiente con el content package.

De este modo partiendo de las gramáticas que nos proporcionaba IMS en su página web para conseguir secuenciación, logramos con solo unas ligeras modificaciones sintácticas para que el content package reconociese los elementos correspondientes a la secuenciación que estaban descritos en las gramáticas descargadas de la página de IMS que la aplicación manejase secuenciación sin ninguna limitación y sin modificar el código de la aplicación.

Esto nos abre un gran abanico de posibilidades para el futuro, ya que mediante este sencillo procedimiento en un futuro se podría hacer por ejemplo que esta misma aplicación fuese compatible con etiquetas de otros estándares con solo añadirlas de forma adecuada en el content package.

Un ejemplo de esta técnica es el siguiente:

Partiendo del archivo que describe la secuenciación conseguido en la pagina de IMS **imsss_v1p0.xsd** y del content package de los objetos que manejaba el lomeEditor 2005 **imscp_v1p1.xsd** conseguiremos de una forma sencilla que el imscp_v1p1.xsd (el content package) maneje la etiqueta sequencing y nuestra aplicación pueda editarla sin problemas.

Tomamos del **imsss_v1p0.xsd** la parte que nos interesa que es donde se declaran los tipos y los elementos. Extraemos el siguiente fragmento:

```
<xs:element name = "sequencing" type = "sequencingType"
              block = "#all">
              <xs:annotation>
                    <xs:documentation>The root element for all sequencing
tags.  This tag will usually appear as a child element to an IMS CP item
tag.</xs:documentation>
              </xs:annotation>
       </xs:element>
       <xs:complexType name = "sequencingType">
              <xs:annotation>
                    <xs:documentation>The type associated with any top-
level sequencing tag</xs:documentation>
              </xs:annotation>
```

```
            <xs:sequence>
                  <xs:element name = "controlMode" type =
"controlModeType"
                              block = "#all" minOccurs = "0">
                              <xs:annotation>
                                    <xs:documentation>non-exclusive
definition of acceptable control-modes</xs:documentation>
                              </xs:annotation>
                  </xs:element>
                  <xs:element name = "sequencingRules" type =
"sequencingRulesType"
                              block = "#all" minOccurs = "0"/>
                  <xs:element name = "limitConditions" type =
"limitConditionsType"
                              block = "#all" minOccurs = "0"/>
                  <xs:element name = "auxiliaryResources" type =
"auxiliaryResourcesType"
                              block = "#all" minOccurs = "0"/>
                  <xs:element name = "rollupRules" type =
"rollupRulesType"
                              block = "#all" minOccurs = "0"/>
                  <xs:element name = "objectives" type =
"objectivesType"
                              block = "#all" minOccurs = "0">
<!--
                              <xs:unique name = "uniqueGlobalObjective">
                                    <xs:selector xpath =
".//imsss:mapInfo[@writeSatisfiedStatus = 'true' or
@writeNormalizedMeasure = 'true']"/>
                                    <xs:field xpath = "@targetObjectiveID"/>
                              </xs:unique>
-->
                  </xs:element>
                  <xs:element name = "randomizationControls" type =
"randomizationType"
                              block = "#all" minOccurs = "0"/>
                  <xs:element name = "deliveryControls" type =
"deliveryControlsType"
                              block = "#all" minOccurs = "0"/>
                  <xs:any namespace = "##other" processContents =
"strict" minOccurs = "0" maxOccurs = "unbounded"/>
            </xs:sequence>
            <xs:attribute name = "ID" type = "xs:ID"/>
            <xs:attribute name = "IDRef" type = "xs:IDREF"/>
      </xs:complexType>
```

y añadimos cada uno de los nuevos elementos y tipos complejos (complextype) que aparecen en este fragmento a nuestro content package cambiando la sintaxis de forma adecuada, es decir, donde en este fragmento pone <xs ... hay que poner en el content package <xsd ... Como puede verse en el siguiente framento.

```
<!-- ** Element Declarations ** -->
        <xsd:element name = "dependency" type = "dependencyType"/>
        <xsd:element name = "file" type = "fileType"/>
        <xsd:element name = "item" type = "itemType"/>
        <xsd:element name = "manifest" type = "manifestType"/>
        <xsd:element name = "metadata" type = "metadataType"/>
        <xsd:element name = "organization" type = "organizationType"/>
        <xsd:element name = "organizations" type = "organizationsType"/>
        <xsd:element name = "resource" type = "resourceType"/>
        <xsd:element name = "resources" type = "resourcesType"/>
        <xsd:element name = "schema" type = "schemaType"/>
        <xsd:element name = "schemaversion" type =
"schemaversionType"/>
        <xsd:element name = "title" type = "titleType"/>
        <xsd:element name = "sequencing" type =
"sequencingType"/>
        <xsd:element name = "controlMode" type = "controlModeType"/>
```

en esta parte del cp hemos declarado un nuevo tipo de elemento sequencing al que le hacemos corresponder un tipo complejo sequencing type que tendrá la siguiente estructura:

```
<!--**secuencing ** -->

        <xsd:complexType name = "sequencingType">
                <xsd:sequence>
                        <xsd:element ref = "controlMode" minOccurs = "0"/>
                        <xsd:element ref = "limitConditions" minOccurs = "0"/>
                        <xsd:element ref = "auxiliaryResources" minOccurs =
"0"/>
                        <xsd:element ref = "randomization" minOccurs = "0"/>
                        <xsd:element ref = "deliveryControls" minOccurs =
"0"/>
                        <xsd:element ref = "sequencingRule" minOccurs =
"0"/>
                        <xsd:element ref = "rollupRules" minOccurs = "0"
                        maxOccurs = "unbounded"/>
                        <xsd:element ref = "objectives" minOccurs = "0"
                        maxOccurs = "unbounded"/>
                        <xsd:group ref = "grp.any"/>
                </xsd:sequence>
                <xsd:attributeGroup ref = "attr.identifierref"/>
        <xsd:attributeGroup ref = "attr.identifier"/>
        <xsd:anyAttribute namespace = "##other" processContents =
"strict"/>
        </xsd:complexType>
```

Procediendo del mismo modo con todas las xsd obtenidas de la página principal de IMS, a saber; imsss_v1p0auxresource.xsd, imsss_v1p0control.xsd, imsss_v1p0delivery.xsd, imsss_v1p0limit.xsd, imsss_v1p0objective.xsd, imsss_v1p0random.xsd, imsss_v1p0rollup.xsd, imsss_v1p0seqrule.xsd, imsss_v1p0util.xsd conseguiremos que la aplicación

incluya la secuenciación de una forma sencilla y sin necesidad de alterar el código ya existente.

7 CONCLUSIONES Y TRABAJO FUTURO

7.1 Conclusiones

7.1.1 Situación final del proyecto

Toda la funcionalidad prevista en la especificación de requisitos y planificación de alcance de LOMEditor 2006, ha podido ser desarrollada. Esta funcionalidad, adecuadamente probada, es la siguiente:

- Edicion de objetos de aprendizaje de los tipos:
  -IMS Metadata
  -IMS Content Package
  -IMS Simple Sequencing

  La edicion incluye las funcionalidades de la versión anterior de la herramienta, nótese que algunas han tenido que ser adaptadas o modificadas:

  - Apertura de objeto de aprendizaje previa validación, que incluye la visualización del contenido del objeto de aprendizaje.
  - Edición de un objeto de aprendizaje, que comprende la inserción de nuevos elementos, la eliminación de los mismos y de la posibilidad de modificar sus atributos.
  - Salvar el manifiesto (imsmanifest.xml) del objeto de aprendizaje con el que se está trabajando.
  - Guardar un objeto de aprendizaje abierto en forma comprimida.
  - Creación de un nuevo objeto de aprendizaje, partiendo de cero.
  - Ayudas
  - Consultas sobre una base de datos que contiene información de objetos de aprendizaje.

- Cambiar la arquitectura de la aplicación a una arquitectura basada en plugins.

- Ser capaces de generar nuevos plugins para la aplicación, en concreto: creación del Plugin Repositorio, que conecta la aplicación vía web con una lista de repositorios de objetos de aprendizaje.

7.1.2 Trabajo realizado y conocimientos adquiridos

Para llevar a cabo el desarrollo de esta herramienta, ha sido necesario realizar un trabajo coordinado y organizado, para el cual han sido muy útiles los conceptos adquiridos en la asignatura de Ingeniería del Software.

Se realizó en un principio la especificación de requisitos, asi como unas metas de alcance del proyecto. Posteriormente, se decidió el tipo de arquitectura que modulizaría mejor la aplicación y que la ajustaría a la demanda mas actual.

En la gestión de configuración por parte de los miembros del grupo ha sido determinante el uso de internet, pues nos hemos podido comunicar dia a dia mediante e-mails y mediante programas de chat, para trabajar on-line.

Finalmente cada integrante del equipo de desarrollo ha cumplido satisfactoriamente con el trabajo que tenía asignado.

Los conocimientos adquiridos se centran entorno al e-learning, estándares y especificaciones (IMS, SCORM)

Dado que la información principal del objeto de aprendizaje se encuentra en un manifiesto de tipo XML, y la gramática en la que se basa es XSD, nos vimos en la necesidad de profundizar en la aplicación de estas tecnologías.

Así mismo, aprendimos un nuevo tipo de arquitectura, completamente innovador y muy interesante por su actualidad: los plugins.

7.2 Trabajo futuro

7.2.1 Crear un repositorio de objetos de aprendizaje

Sería utíl para la aplicación el hecho de tener un repositorio de objetos de aprendizaje remoto, para compartir objetos de aprendizaje con otras personas o simplemente para almacenar los objetos propios y tenerlos accesibles mediante internet desde cualquier lugar.

Como ejemplo de herramientas que se han analizado y que cumplen estas funcionalidades cabe citar la aplicacion *Door* de libre distribución.

Sería interesante, como futura mejora, que el repositorio estuviese implementado sobre una base de datos XML por la naturaleza de los documentos que componen un objeto de aprendizaje. De esta forma se podrían realizar búsquedas en los documentos XML que forman parte de un OA.

7.2.2 Crear un buscador inteligente de recursos para los objetos de aprendizaje.

En este apartado se podría desarrollar un plugin que clasificase paginas web mediate el *framework de Weka* u otro similar, sería en definitiva un clasificador de documentos. De esta forma un usuario no tendría que buscar en google o cualquier otro buscador web existente sino que podría directamente desde la aplicación realizar búsquedas de recursos específicos que le facilitasen el desarrollo de objetos de aprendizaje.

7.2.3 Mejoras en las funcionalidades de edición de la herramienta

- Multiedición de Objetos de Aprendizaje.
-Personalización de la presentación de los contenidos.
-Composicion de todos los tipos de OAs que abre la herramienta

7.2.4- Adaptación Web de la herramienta.

El futuro de este tipo de herramientas pasa por su adaptación a tecnologías de tipo Web, de forma que sean accesibles de forma universal a cualquiera con solo disponer de un navegador, y que puedan de esta forma interaccionar con repositorios de contenidos conectados a la red. De esta forma se permitirá una reutilización de objetos ya realizados, así como una rápida generación de otros nuevos.

Apéndice A - Manual de usuario de LOMEditor 2006

Índice de contenidos

A.1 Introducción

LOMEditor 2006 es un editor de IMS Metadata, IMS ContentPackage e IMS Simple Sequencing. Con LOMEditor 2006 puedes introducir tus propios contenidos en objetos de aprendizaje, como imágenes, vídeos, páginas HTML, pdfs, etc.

LOMEditor 2006 es una herramienta de gran valor para la comunidad educativa, ya que proporciona un medio para crear y modificar objetos de aprendizaje que siguen los estándares y pueden por tanto ser utilizados en repositorios locales o distribuidos de objetos.

LOMEditor 2006 proporciona las siguientes funcionalidades:

-Empaquetar objetos Metadata, CP, SS

-Desempaquetar y editar objetos Metadata, CP y SS ya existentes, incluso los creados con otras herramientas, siempre que cumplan los estándares IMS

-Acceder a repositorios de objetos de aprendizaje

-Añadir plugins creados por desarrolladores software que implementen futuras funciones de la herramienta

-A demás se mantienen el resto de funcionalidades de la version anterior de la herramienta como Componer objetos de aprendizaje y evaluar la calidad de los objetos.

LOMEditor 2006 implementa las especificaciones IMS para Metadata, Content Package y Simple Sequencing. Estas especificaciones pueden encontrarse en la página web de IMS: http://www.imsproject.org/

Actualmente LOMEditor 2006 soporta las versiones v1.2.2 de la especificación IMS para Metadata, v1.1 de la especificación IMS para Content Package y v1.0 de la especificación IMS para Simple Sequencing

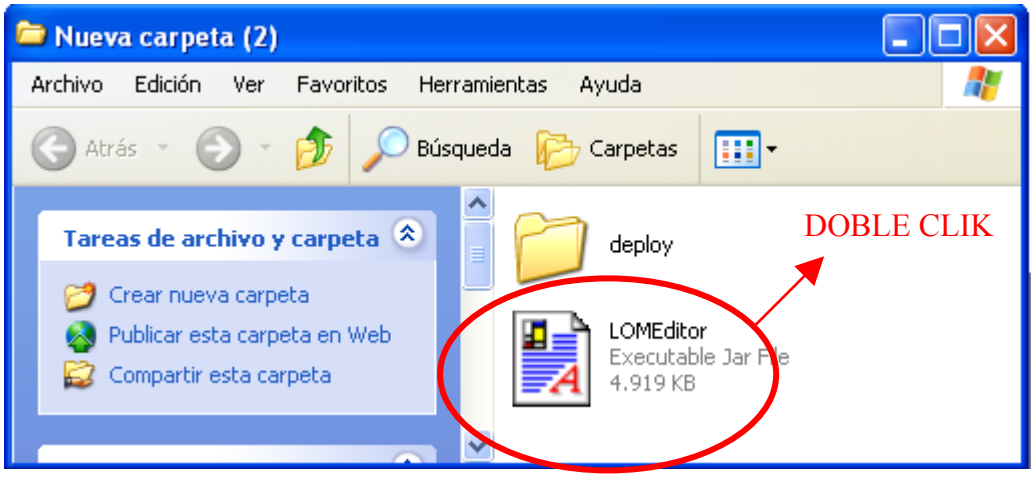
A.2 LOMEditor 2006

LOMEditor 2006 es una aplicación Java, por lo tanto se podrá ejecutar en cualquier platafroma capaz de ejecutar aplicaciones Java. Se proporcionaran ejecutables del LOMEditor 2006 para Microsoft Windows, Macintosh, Linux, Solaris y un ejecutabje Jar. Es necesario tener instalada la maquina virtual de java en su version 1.4.2 o superior.

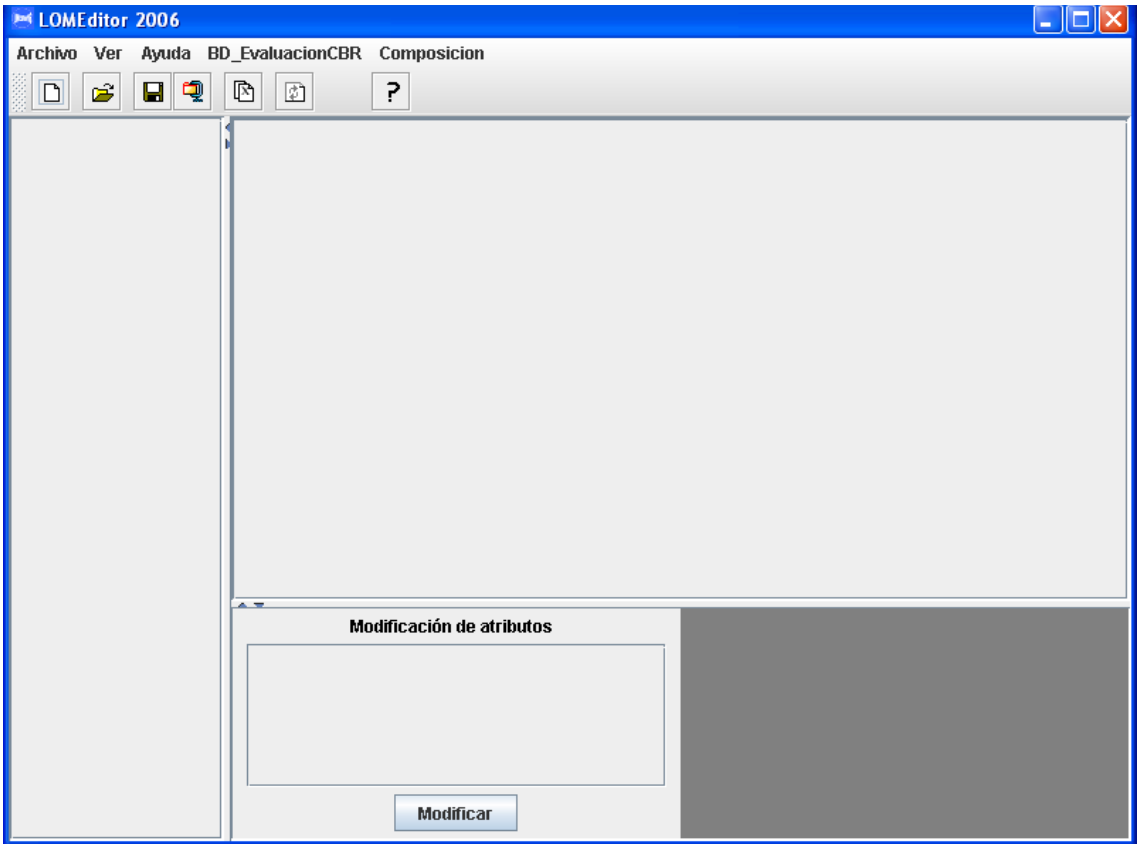
A.2.0 Ejecución de la aplicación usando el Jar como Ejecutable

En el CD que se proporciona con esta memoria, hay una copia de a última versión del software de LOMEeditor 2006, en éste se proporciona un archivo Jar ejecutable (LOMEditor.jar).

Para ejecutar la aplicación, ha de crearse un directorio que se llame "deploy" y que este en el mismo directorio que el archivo ejecutable (este directorio contendrá los plugins, que el usuario desea que se carguen al ejecutar la aplicación). Posteriormentese hará doble click sobre el Jar ejecutable y se ejecutará la aplicación.



Al hacer doble clik sobre el ejecutable, aparecerá la pantalla de la aplicación:



A.2.1 Requerimientos del sistema

En cuanto a hardware, los requerimientos mínimos del sistema son:

-Procesador intel pentium 3 o equivalente, a 800 MHz

-256 Mb de RAM

En cuanto a software

-Microsoft Windows 98, ME, NT4.0, 2000, o XP

- Navegador Microsoft® Internet Explorer 6 o equivalente

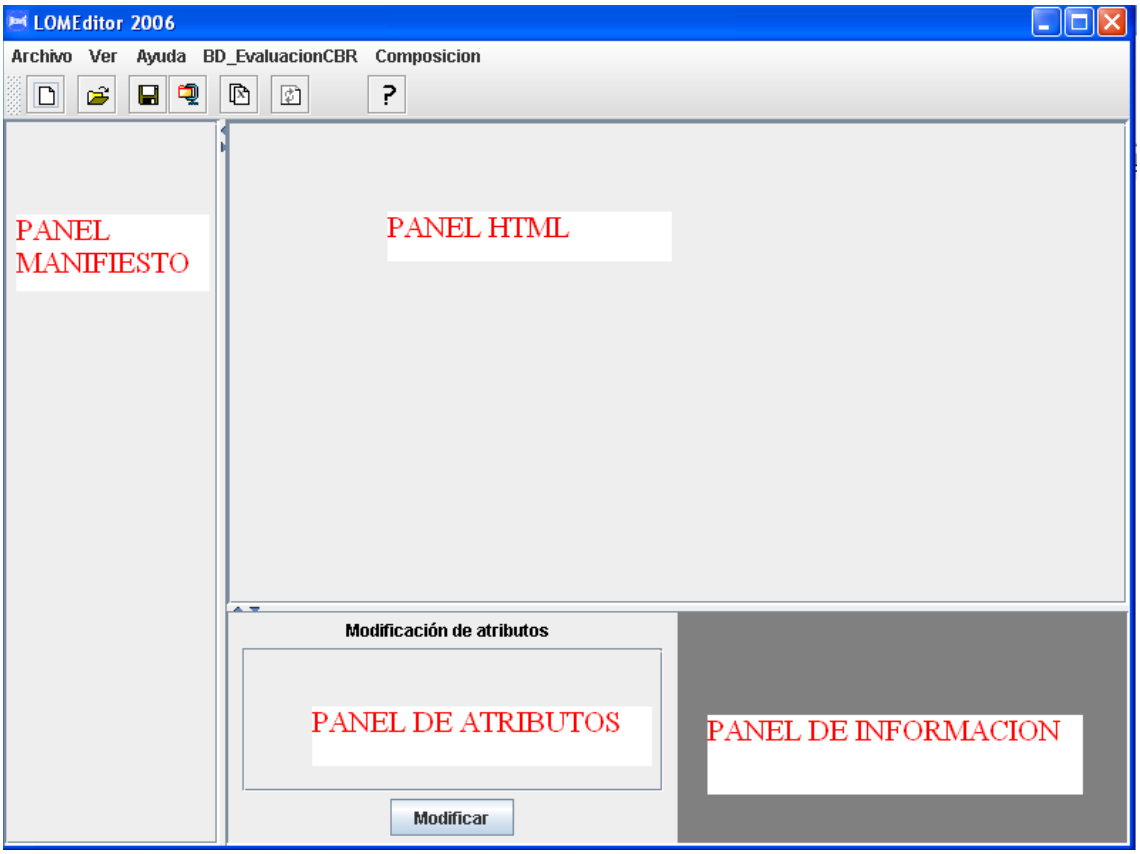
A.2.2 El Workspace de LOMEditor 2006

Básicamente consiste en tres paneles:

-El panel del manifiesto (a la izquierda), en el cual se mostrará el contenido del manifiesto en forma de árbol

-El panel HTML central

-El panel de modificacion de atributos e información del elemento seleccionado actulmente(abajo)



A.2.3 Barra de herramientas

La barra de herramientas del LOMEditor 2006 tiene la forma:

Ecplicamos la acción de cada componente de izquierda a derecha:

 Nuevo (Archivo nuevo) crea un nuevo IMS Metadata, CP o SS

 Abrir (Abrir archivo) abre IMS Metadata, CP o SS

 Guardar (Guardar objeto de aprendizaje) guarda el objeto de aprendizaje

 Guardar (Guardar objeto de aprendizaje)

 Cerrar (Cierra el objeto de aprendizaje)

 Actualizar (Actualiza el objeto de aprendizaje)

 Ayuda

A.2.4 Menú

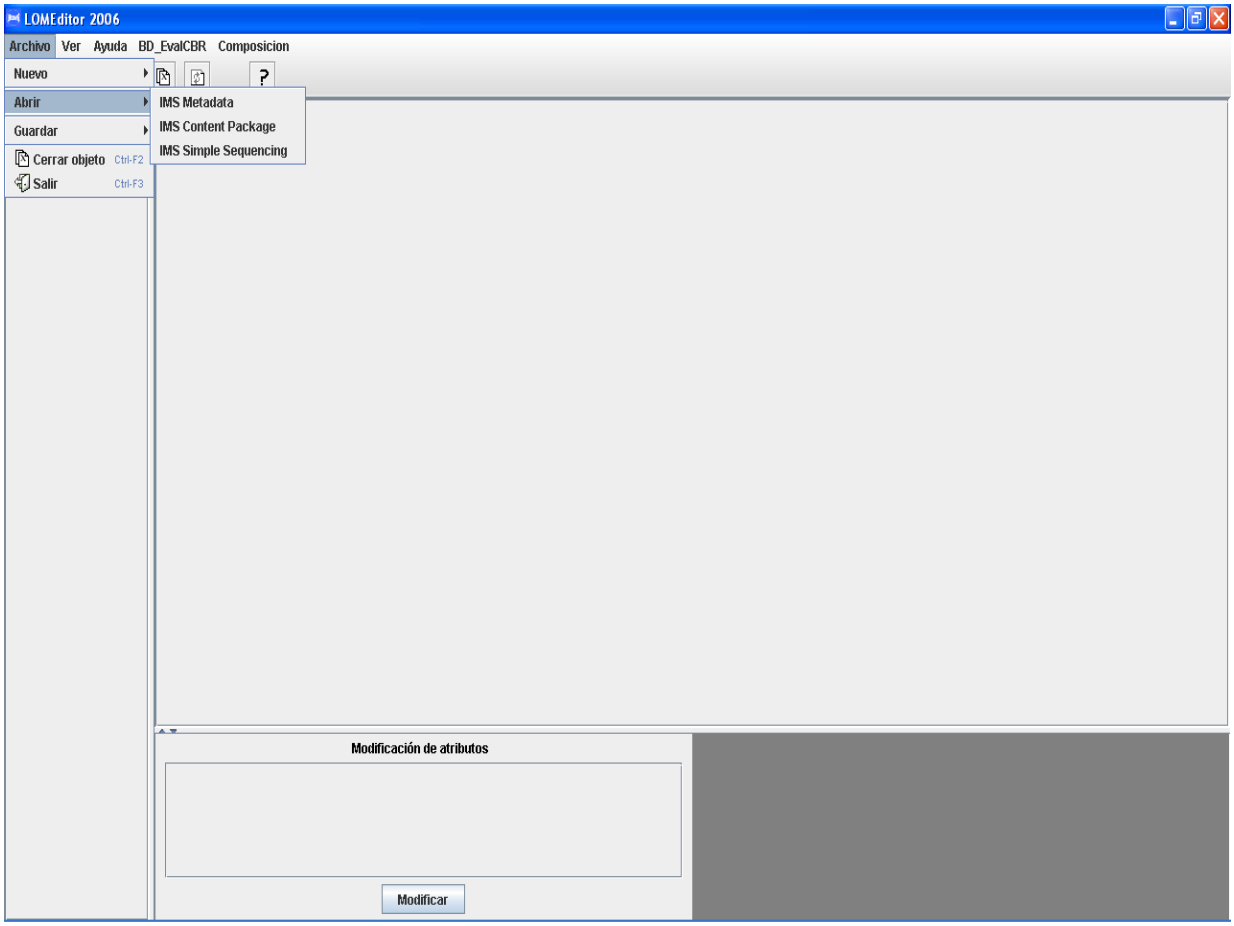
El menú será explicado según vayamos encontrando las opciones del mismo, mas adelante

A.3.Tutorial

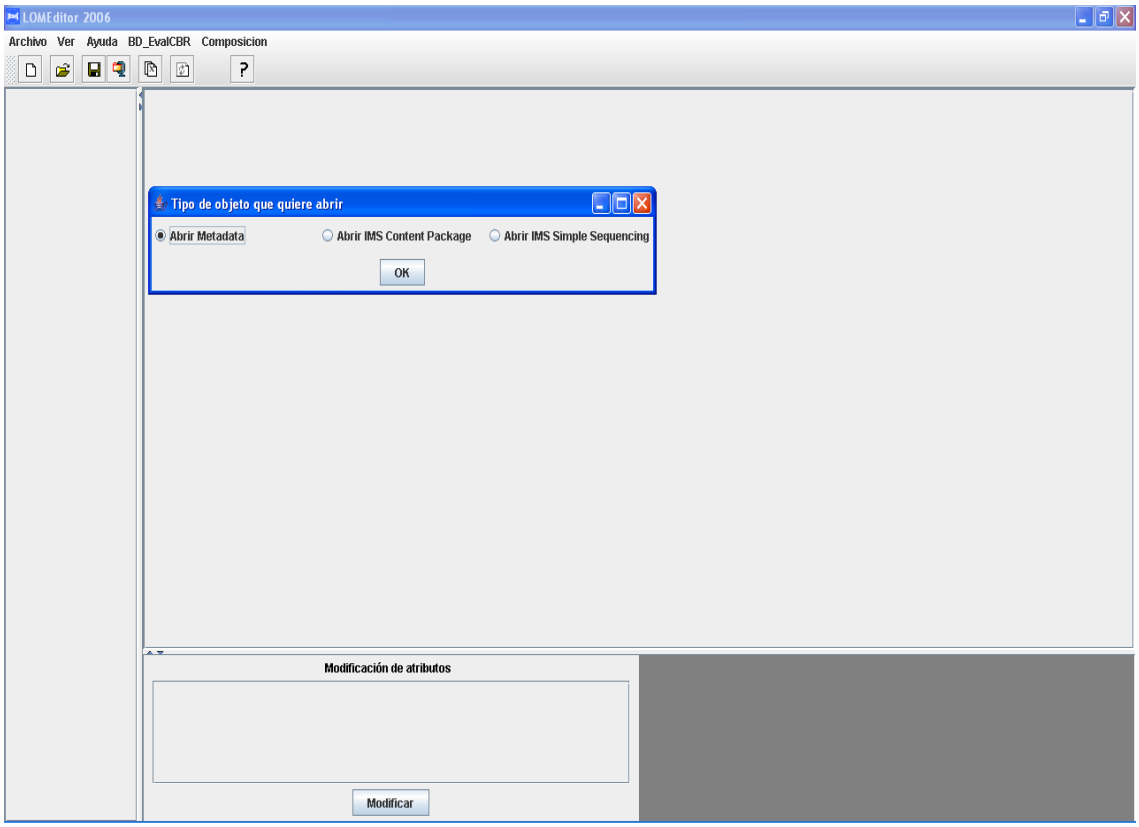
Esta sección explica cómo crear objetos IMS Metadata, IMS Content Package, e IMS Simple Secuencing, cómo editarlos

A.3.1 Abrir un objeto de aprendizaje:

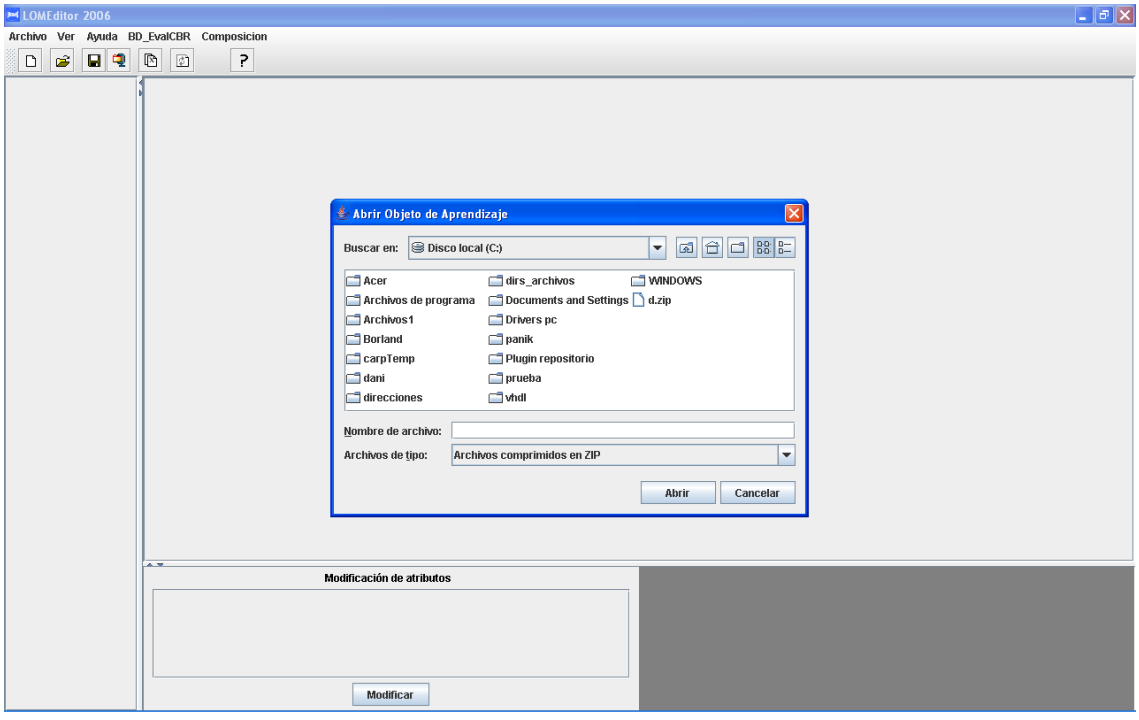
Para abrir un objeto de aprendizaje el usuario pincha en Archivo y en Abrir y se abrirá un menú en el que se elegirá el tipo de objeto a abrir:



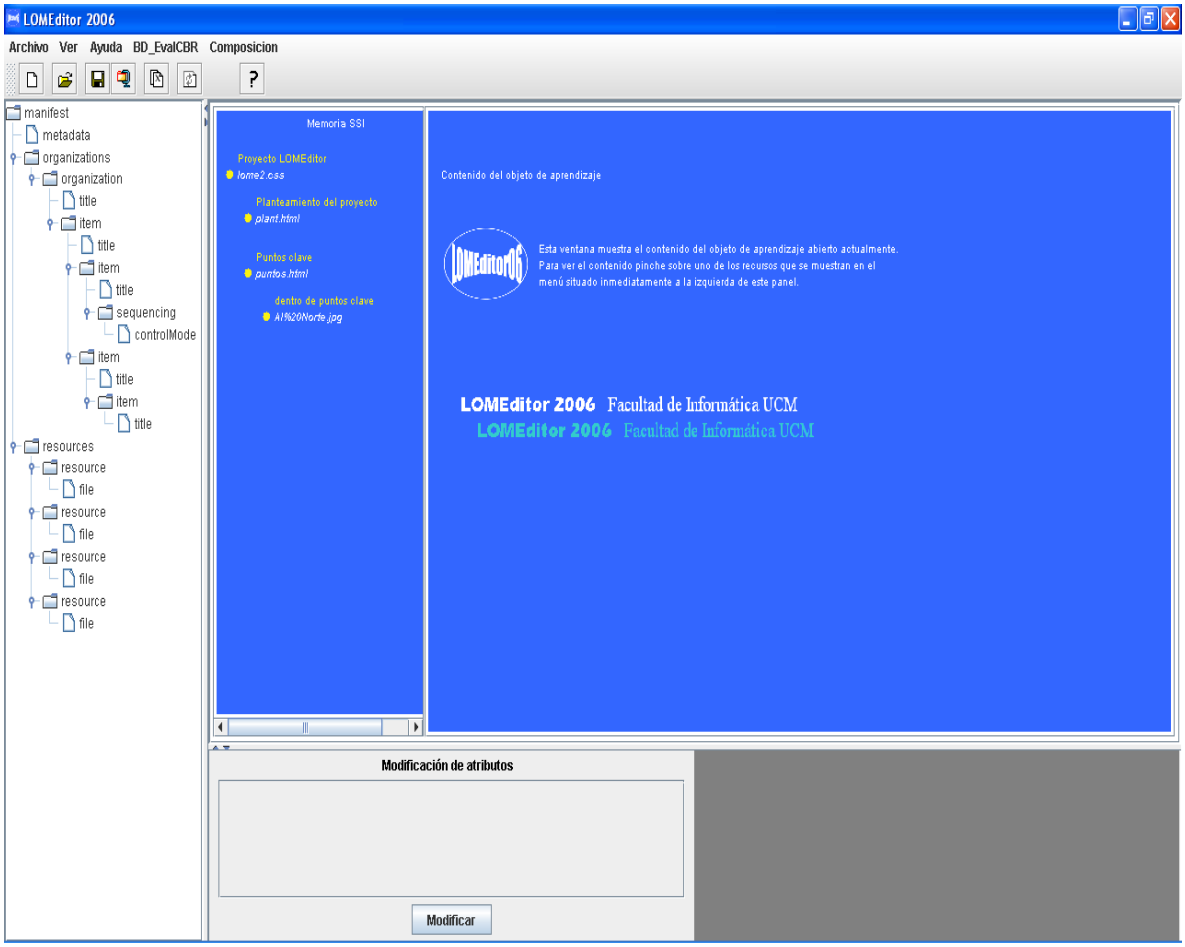
Si se pulsa el botón de la barra de herramientas que corresponde a abrir, aparecerá una nueva ventana para elegir el tipo de objeto a abrir:



Una vez elegido el tipo de objeto, aparecerá un cuadro de diálogo para buscar el objeto y otro para decidir donde se descomprime el mismo:



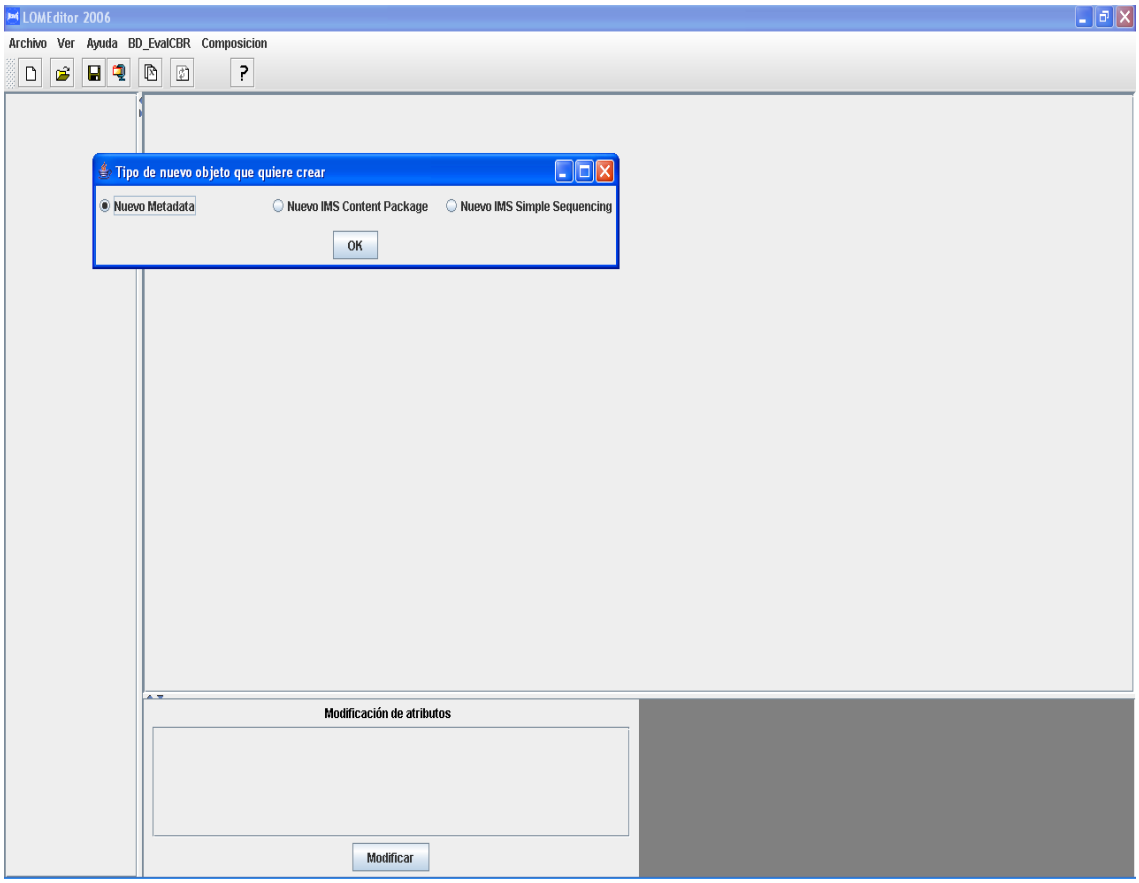
Una vez abierto el objeto se mostrará en el panel de la izquierda el árbol de contenidos de ese objeto:



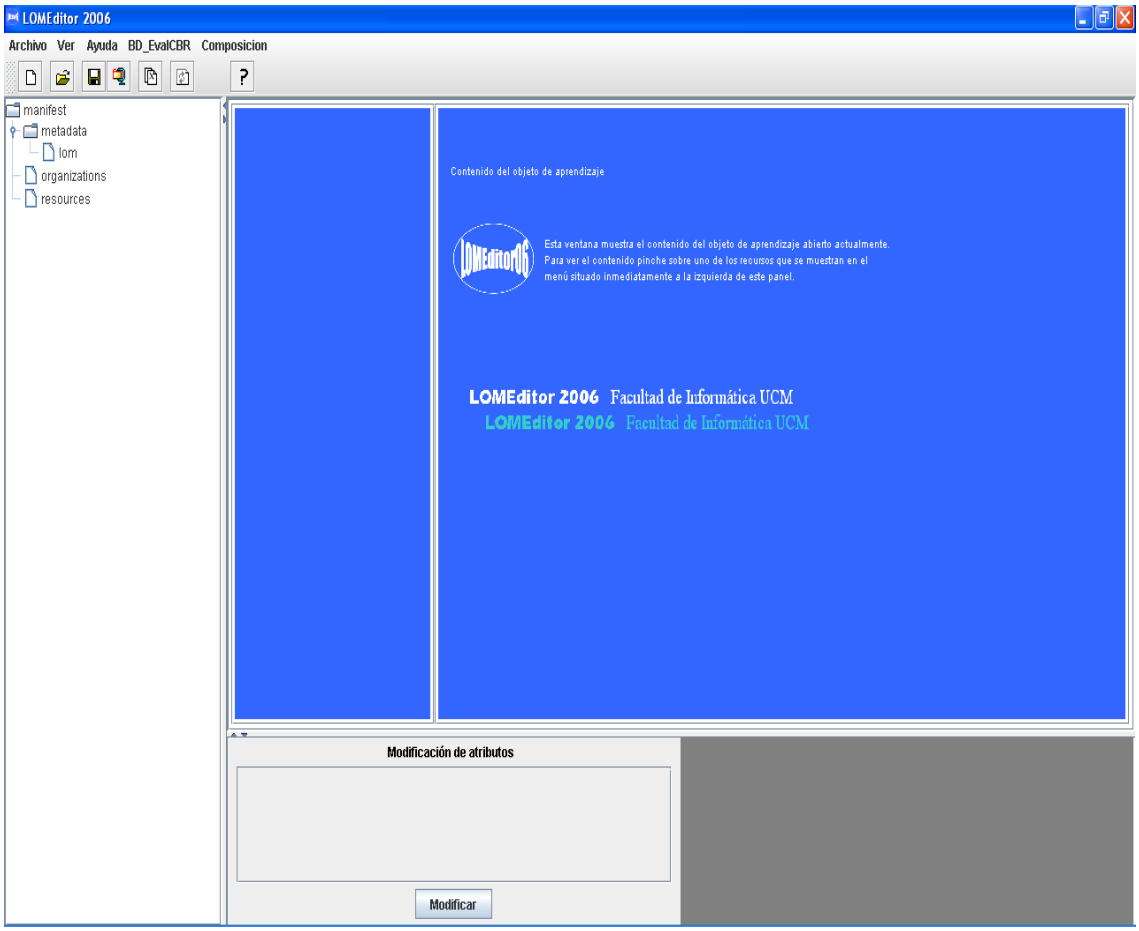
El objeto del ejemplo es un objeto con secuenciación ya que podemos observar que tiene la etiqueta sequencing.

A.3.2 Crear un nuevo objeto:

El usuario puede ir a archivo nuevo y elegir el tipo de objeto a crear o pulsar el botón de nuevo de la barra de herramientas y le aparecerá una ventana en la que deberá elegir el tipo de objeto a crear:

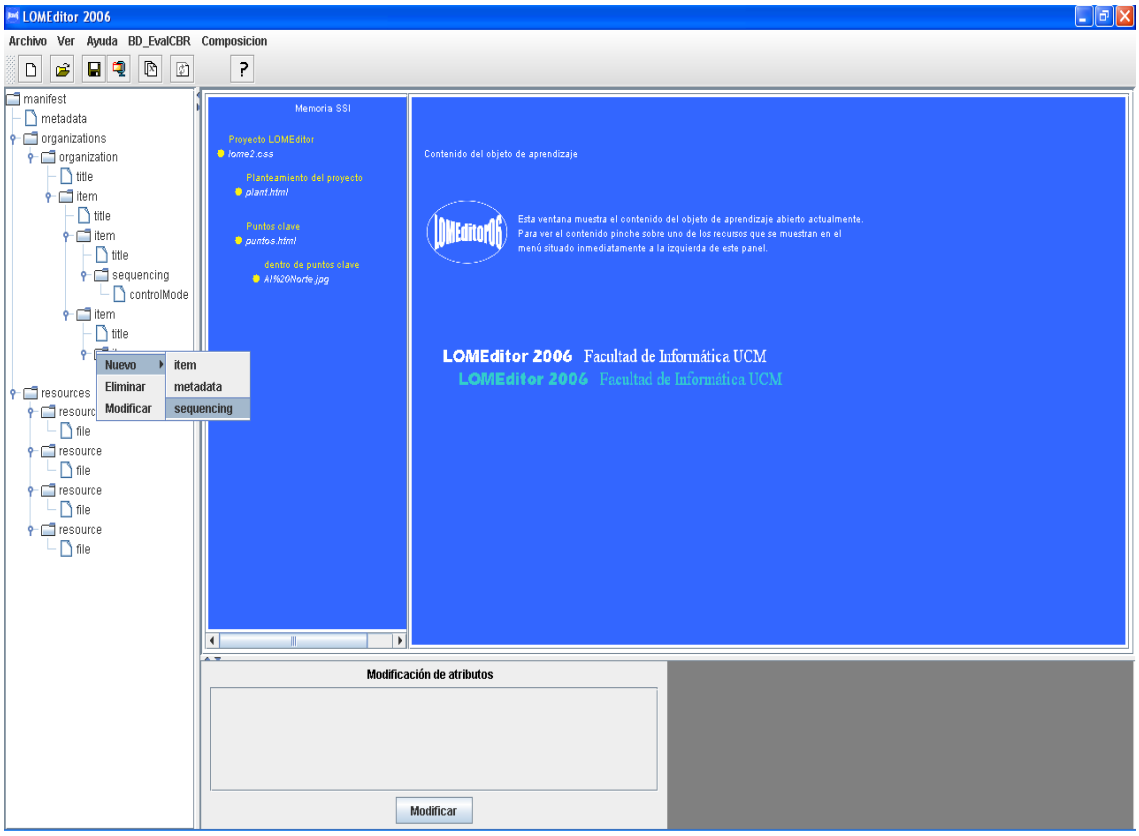


Una vez elegido el tipo de objeto se pulsa ok y aparecerá un cuadro de diálogo para indicar donde queremos guardar el nuevo objeto, se elige la carpeta y se pulsa ok. El sistema muestra el árbol de un objeto vacío, listo para ser completado:

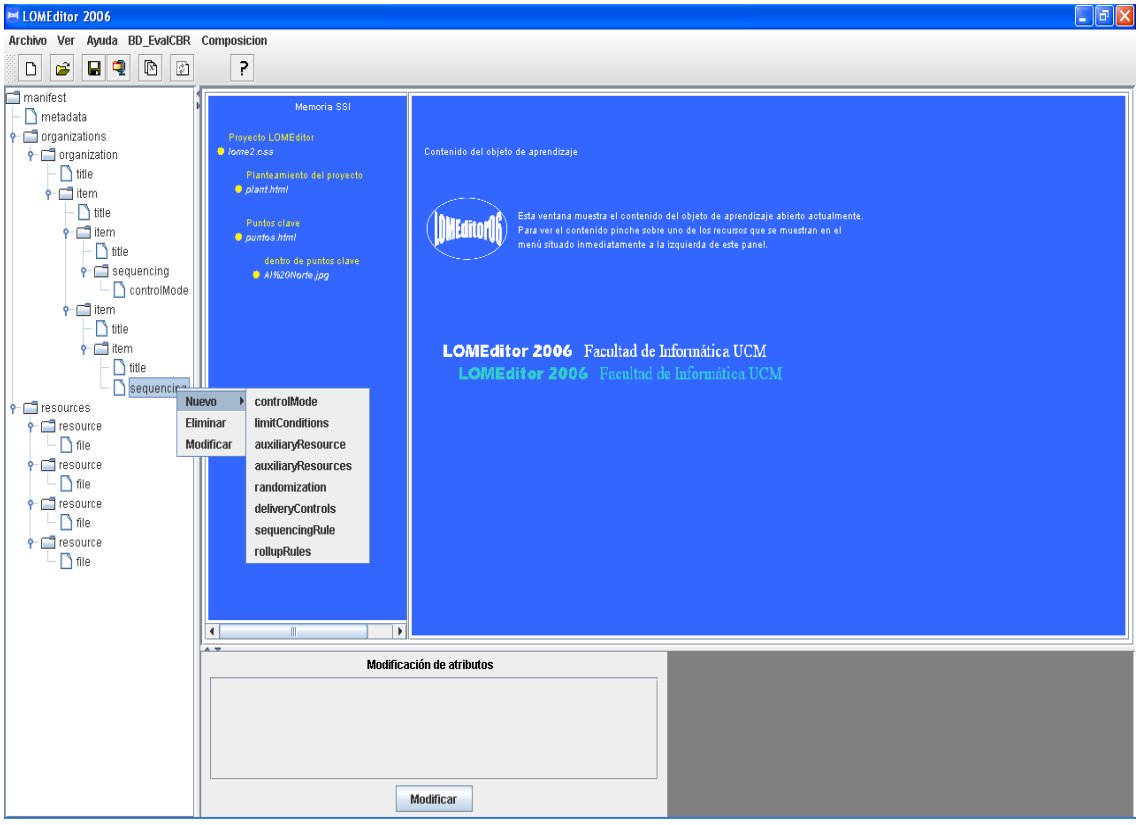


A.3.3 Añadir secuenciación a un objeto de aprendizaje:

Para añadir secuenciación a un objeto, en primer lugar tiene que estar abierto. Luego pulsaremos con el botón derecho en un nodo de organizations al que se quiere aplicar la secuenciación y se desplegará un menú en el que nuevo, nos dará la opción de añadir secuenciación:



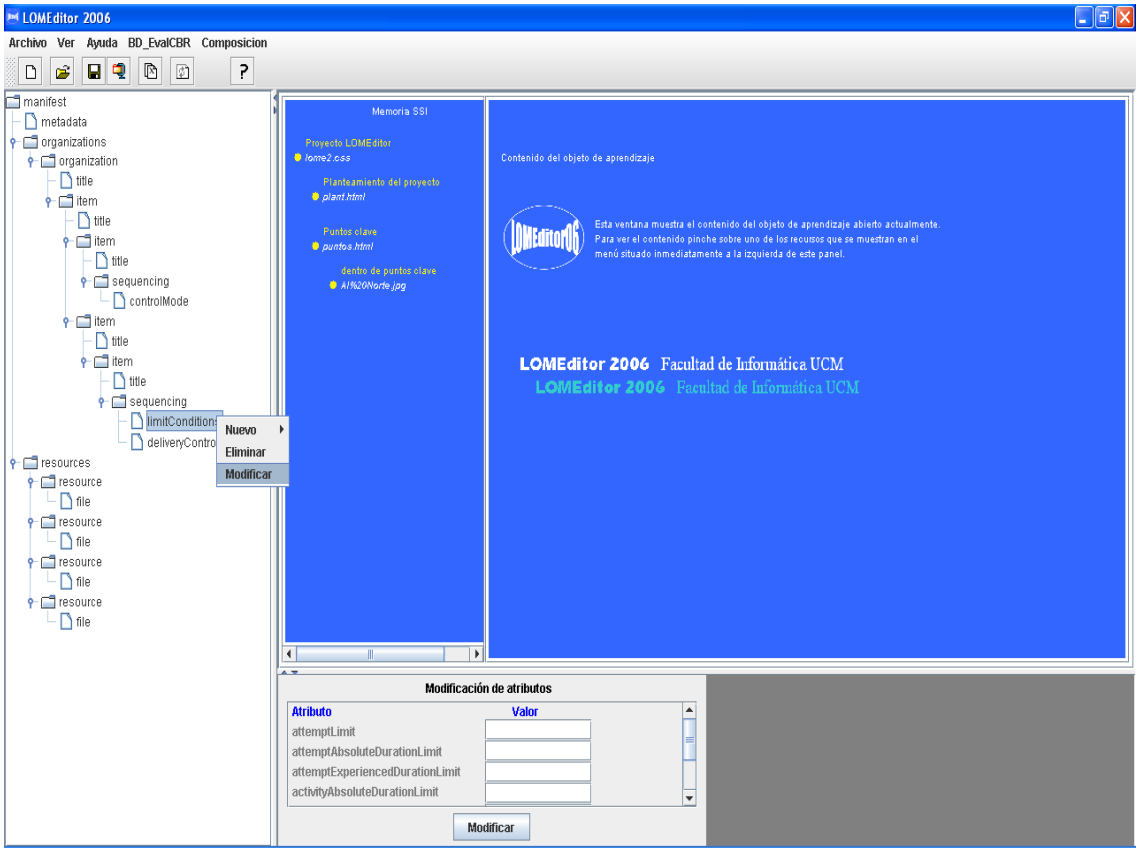
Nos aparecerá un nuevo hijo en ese nodo con el nombre de sequencing. Si pulsamos el botón derecho sobre él y elegimos nuevo, nos dará todos los posibles atributos a definir para la secuenciación:



A.3.4 Modificar atributos de secuenciación:

Para poder modificar los atributos de secuenciación de un objeto, hay que tener un objeto IMS SS abierto.

Una vez abierto el objeto seleccionamos la etiqueta sequencing del nodo al que qeremos modificar sus atributos, y elegimos el atributo a modificar:

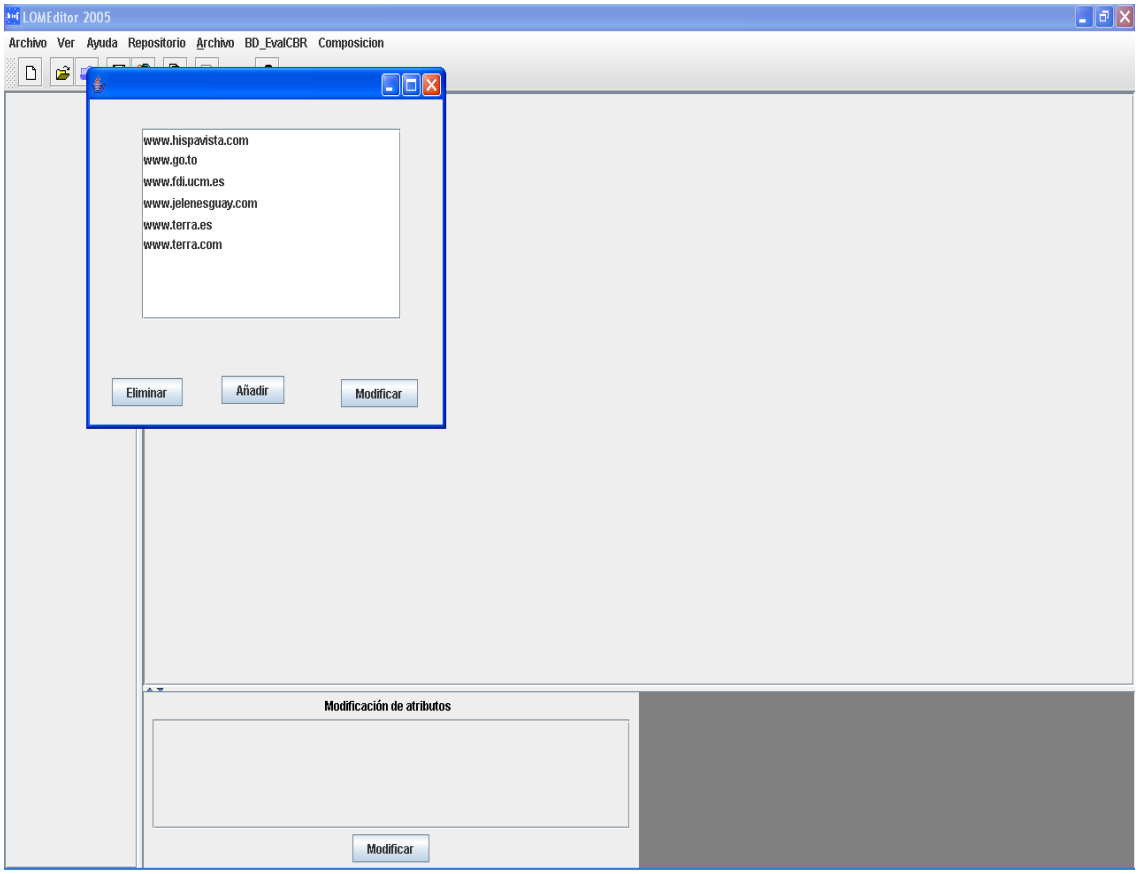


Nos aparecerá en el panel inferior los atributos con un espacio para definir su valor si no estaba definido, o para modificar el valor que tenían.

A.3.5 Acceder a un repositorio web

Para acceder a un repositorio web es necesario tener el plugir repositorio cargado en la aplicación. Para ello, el plugin repositorio ha de estar en la carpeta de plugins.

Con este plugin cargado en el menú tendremos repositorio, en el que nos da la opción de lanzar repositorio y nos aparecerá una ventana con los repositorios que contenga la base de datos:



Pulsando cualquier dirección de las que aparecen, se abrirá un navegador con la página en la que están los objetos de aprendizaje.

A.3.5.1 Añadir una dirección a la lista de repositorios

Con la ventana anterior abierta, pulsamos el botón añadir y nos aparecerá una ventana para introducir el nuevo repositorio, lo introducimos y pulsamos añadir. En este momento la base de datos de repositorios se actualiza y se muestra por pantalla el nuevo repositorio introducido.

A.3.5.2 Eliminar una direccion de la lista de repositorios

Seleccionando una de las direcciones que se nos muestran en la ventana del repositorio, pulsamos el botón eliminar y se borrará de la base de datos y de la ventana visible.

A.3.5.3 Modificar una dirección de la lista de repositorios

Seleccionamos la dirección a modificar y pulsamos el botón modificar. Aparecerá una ventana con la dirección a modificar, se modifica y se pulsa aceptar. La base de datos será modificada y la ventana visible también.

Apéndice B: Etiquetas de secuenciación (elementos de la secuenciación)

<sequencing>

Es la etiqueta raíz que permite generar a todas las demás es. Puede estar dentro de una etiqueta <element> o del elemento <organization>. Esta etiqueta no tiene valores asociados pero si contiene dos atributos:

- ID (identifier): este atributo es opcional, es un identificador único asociado a un elemento de secuenciación.
- IDREF (identifierref): atributo opcional que es una referencia a otro elemento de secuenciación por medio de su identificador único (identifier).

Dentro de la etiqueta <sequencing> podemos encontrar los siguientes elementos (etiquetas).

- <controlMode>
- <sequencingRules>
- <limitConditions>
- <auxiliaryResources>
- <rollupRules>
- <objectives>
- <randomizationControl>
- <deliveryControls>

Todos estos elementos pueden ocurrir como maximo una vez dentro de cada elemento sequencing (Multiplicidad 0 o 1).

A continuación comentaremos el contenido y función de cada uno de los elementos enumerados anteriormente.

<controlMode>

Esta etiqueta contiene la descripción de los tipos de comportamiento de secuenciación para una actividad determinada. No tiene valores asociados con ella y contiene los siguientes atributos:

- choice : este atributo es opcional y su valor por defecto es true. indica que está permitida una petición de secuenciación que apunte a los hijos de la actividad. Tipo de datos XML xs:boolean.
- choiceExit :este atributo es opcional y su valor por defecto es true. Indica que un hijo activo de esta actividad tiene permitido terminar si una petición de secuenciación así lo solicita. Tipo de datos XML xs:boolean.
- flow: este atributo es opcional y su valor por defecto es false. Indica que esta permitido el flujo de una petición dev secuenciación hacia los hijos de esta actividad. Tipo de datos XML xs:boolean.

- forwardOnly: este atributo es opcional y su valor por defecto es false. Indica que los objetivos hacia atrás (en términos del árbol de actividades) no están permitidos para los hijos de esta actividad. Tipo de datos XML xs:boolean.
- useCurrentAttemptObjectiveInfo: atributo opcional cuyo valor por defecto es true. Indica que la información de progreso de un objetivo para un hijo de la actividad solo será usado en reglas de evaluación y "rollup" si esta información fue registrada durante el actual intento de la actividad. Tipo de datos XML xs:boolean.
- useCurrentAttemptProgressInfo: atributo opcional y valor por defecto true. Indica que la informacion de progreso del intento para un hijo de la actividad solo será usada en reglas de evaluación y "rollup" si esta información fue registrada durante el intento actual de esta actividad. Tipo de datos XML xs:boolean.

Multiplicidad: la etiqueta controlMode una vez o ninguna dentro del elemento sequencing.

Un ejemplo de la utilización de este elemento es la siguiente:

```
<item identifier="prueba">

(...)

<sequencing>
        <controlMode choice="false" choiceExit="false"
                flow="true" forwardOnly="true" />

</sequencing>

(...)

</item>
```

<sequencingRules>

Este elemento es el contenedor de una descripción de una regla de secuenciación. Cada una de esta reglas describe el comportamiento de secuenciación de una actividad. Cada una de las actividades puede tener un número ilimitado de reglas de secuenciación no agrupadas que se evalúan en el orden en el que aparecen listadas.

Multiplicidad: puede aparecer cero o mas veces dentro del elemento sequencing

No tiene valores asociados con ella, ni atributos y contiene los siguientes elementos:

- <preConditionRule>

Este elemento contiene la descripción de acciones sobre decisiones del control de la secuenciación para una actividad específica. Es una regla de precondición.

Este elemento a su vez tampoco contiene valores ni atributos asociados con el sin embargo contiene otros dos elementos:

- <ruleConditions>

Este elemento contiene el conjunto de condiciones que se aplicarán tanto a las regla de precondición, como a las de precondición y a las de salida (exit rules).

Multiplicidad: este elemento puede aparecer una o ninguna vez dentro de <preConditionRule> , <exitConditionRule> y <postConditionRule>.

No tiene valores asociados con el, no obstante contiene los siguientes atributos y elementos:

Atributos:

- conditionCombination : es opcional y su valor por defecto es all. Su tipo XML es xs:token Tiene dos valores posibles:
  - all : la condición evalúa a cierto si y solo si todas las condiciones individuales evalúan a cierto.
  - any : la condición evalua a cierto si alguna de las condiciones evalua a cierto.

Elementos:

- <ruleCondition>

Representa la condición que se quiere avaluar.

Atributos:

- referencedObjective: este atributo es opcional y no tiene valor por defecto. Representa un identificador de un objetivo asociado con la actividad, usado durante la evaluación de la condición. Tipo XML xs:string.
- measureThreshold: este atributo es opcional y su valor por defecto es 0.0. Se usa como valor umbral en las evaluaciones de condiciones basadas en medidas. Tipo XML xs:decimal (rango desde -1.0000 hasta 1.0000 con precisión de 4 decimales)

- operator : este atributo es opcional y su valor or defecto es noOp (no operación). Tipo XML xs:token. Puede tener los siguientes valores
    - not: se niega la condición correspondiente en la evaluación de la regla.
    - noOp: la condición no se altera.
- condition: este atributo es obligatorio y su valor por defecto es always. Representa la condición actual de la regla. A continuación se listan los posibles valores para el atributo condition:

    - satisfied
    - objectiveStatusKnown
    - objectiveMeasureKnown
    - objectiveMeasureGreaterThan
    - objectiveMeasureLessThan
    - completed
    - activityProgressKnown
    - attempted
    - attempLimitExceeded
    - timeLimitExceeded
    - outsideAvailableTimeRange
    - always

El segundo elemento de <preConditionRule> es <ruleAction>

Este elemento indica el comportamiento deseado si la regla evalúa a cierto. El conjunto de comportamientos varia dependiendo del tipo de concición (<preConditionRule> , <postConditionRule> o <exitConditionRule>)

Multiplicidad: aperece una vez en <preConditionRule> , <postConditionRule> y <exitConditionRule>

Este elemento contiene los siguientes atributos:

- action : este atributo es obligatorio pero si no se define ninguna acción se ignora. Nos indica el comportamiento de secuenciación que se adoptará cuando la condición de la regla evalúe a cierto.

    - si action esta definida dentro de una <preConditionRule> podrá tener uno de los siguientes valores:

        - skip

        - disabled

        - hiddenFormChoice

- stopForwardTraversal

- si action esta definida dentro de una <postConditionRule> podrá tener uno de los siguientes valores:

  - exitParent
  - exitAll
  - retry
  - retryAll
  - continue
  - previous

- si action esta definida dentro de una <exitConditionRule> podrá tener únicamente el siguiente valor:

  - exit

Los otros dos elementos de <sequencingRules> son <postConditionRule> y <exitCondicionRule> y tienen los mismos elementos y características que <PreConditonRule>.

<limitConditions>

Esta etiqueta refleja los limites temporales y de intentos sobre una actividad.

Multiplicidad: aparece una vez o ninguna dentro del elemento sequencing.

Contiene los siguientes atributos:

- attemptLimit: es opcional y su valor por defecto es 0. Indica el máximo número de intentos para la actividad. Tipo XML xs:nonNegativeInteger.
- attemptAbsoluteDurationLimit: es opcional y su valor por defecto es 0.0. Indica el tiempo que, como máximo, se puede invertir en un intento de realización de una actividad. Tipo XML xs:duration.

<auxiliaryResources>

Este elemento actúa como contenedor de los recursos auxiliares que son necesarios para consumir el objeto de aprendizaje correctamente.

Multiplicidad: puede aparecer una o ninguna vez dentro de la etiqueta sequencing.

Contiene un único elemento <auxiliaryResource> que indica cual es el recurso auxiliar necesario. Puede aparecer una o más veces dentro de la etiqueta <auxiliaryResources>

A su vez <auxiliaryResource> contiene dos atributos:

- auxiliaryResourceID: es el identificador único del recurso auxiliar.
- purpose: describe el propósito del recurso auxiliar.

<rollupRules>

Este elemento contiene todas las rollup rules asociadas con el objeto de aprendizaje.

Multiplicidad: puede aparecer una o ninguna vez dentro de la etiqueta sequencing.

Tiene los siguientes atributos:

- rollupObjectiveSatisfied: es opcional y su valor por defecto es true. Indica que el estado de satisfacción del objetivo asociado con la actividad esta incluido en el rollup para la actividad padre. Tipo XML xs:boolean.

- rollupProgressCompletion: es opcional y su valor por defecto es true. Indica si el estado de intentos completados asociado con la actividad esta incluido en el rollup para la actividad padre. Tipo XML xs:boolean.

- objectiveMeasureWeight: es opcional y su valor por defecto es 1.0000. Este atributo indica el factor de peso (ponderado) aplicado a los objetivos usada durante el rollup de la actividad padre.

Contiene a su vez un único elemento <rollupRule>

Esta elemento es el contenedor de cada regla rollup que se aplica a una actividad. El formato general de una regla se puede expresar de manera informal como " Si (conjunto de actividades hijo), conjunto de condiciones entonces acción". Se permiten múltiples condiciones.

Contiene los siguientes atributos:

- childActivitySet : este atributo es opcional y su valor por defecto es all. Indica que valores de datos se usan para evaluar las condiciones de rollup. El Tipo XML es xs:token. Esta es una lista de los valores permitidos para este atributo:

- all
- any
- none
- atLeastCount
- atLeastPercent

- minimumCount: es opcional y su valor por defecto es 0. Es un atributo que debe utilizarse cuando el childActivitySet sea atLeastCount. La condición de la regla rollup evalúa a cierto si al menos el número de hijos especificados por este atributo tienen una condición rollup a cierto. El Tipo XML es xs:nonNegativeInteger.

- minimumPercent: es opcional y su valor por defecto es 0.0000 . Este atributo que debe utilizarse cuando el childActivitySet sea atLeastPercent. La condición de la regla rollup evalua a cierto si al menos el porcentaje de hijos especificados por este atributo tienen una condicion rollup a cierto. El tipo XML es xs:decimal

Contiene los siguientes elementos:

- <rollupConditions>

Este elemento es el contenedor del conjunto de condiciones que se aplican en una regla rollup simple.
Multiplicidad : ocurre una única vez dentro del elemento <rollupRule>.

Contiene el siguiente atributo:

- conditionCombination: es opcional y su valor por defecto es any. Indica como son combinadas las condiciones rollup. Tipo XML xs:token. Los valores posibles para este atributo son:

  - all
  - any

Y a su vez contiene el elemento <rollupCondition>

Este elemento identifica una condicion que será aplicada en una regla rollup.
Multiplicidad: puede aparecer una o más veces dentro del elemento <rollupConditions>

Contiene los atributos:

- operator: es opcional y su valor por defecto es noOp. Es el operador lógico unario aplicable a una condicion. El Tipo XML es xs:token Sus posibles valores son:

- not
- noOp

- condition : es obligatorio. Indica el elmento de condicion para la regla. El tipo XML es xs:token. Sus posibles valores son:

  - satisfied
  - objectiveStatusKnown
  - objectiveMeasureKnown
  - completed
  - activityProgressKnown
  - attempted
  - attemptLimitExceed
  - timeLimitExceed
  - outsideAvailableTimeRange

No contiene elementos

- <rollupAction>

  Este elemento identifica una condición que será aplicada a una regla rollup.

  Multiplicidad: aparece una única vez dentro del elemento rollupRule.

  Contiene el siguiente atributo:

  - action: es obligatorio este atributo indica el comportamiento deseado cuando una regla evalúa a cierto. El tipo de datos XML correspondiente es xs:token. Los posibles valores son:
    - satisfied
    - notSatisfied
    - completed
    - incomplete

  No contiene elementos.

<objectives>

El elemento objectives es el contenedor del conjunto de objetivos que están asociados con una actividad. Cada actividad debe tener al menos un objetivo primario y puede tener un número ilimitado de objetivos.
Multiplicidad: aparece una o ninguna vez en el elemento sequencing.

No contiene atributos

Contiene los siguientes elementos:

- <primaryObjective>

Identifica el objetivo que interviene en el rollup asociado con la actividad. Si el elemento objectives está definido, entonces el elemento primaryObjective es obligatorio; aunque se puede definir como un elemento vacio (<primaryObjective / >).

Multiplicidad: aparece una y solo una vez en el elemento objectives.

Sus atributos son los siguientes:

- satisfiedByMeasure: es opcional y su valor por defecto es false. Este atributo indica que el elemento <minNormalizedMeasure> (que se describirá más adelante) debe ser utilizado (si el valor de este atributo es true) en el lugar de cualquier otro método para determinar si el objetivo asociado con la actividad se ha satisfecho. El Tipo de datos XML es xs:boolean.

- objectiveID: es opcional. Representa el identificador asociado con la actividad. El valor asignado al elemento objectiveID es un identificador único, y este valor no puede ser una cadena de caracteres vacía ni puede contener espacios en blanco.

  Si un <primaryObjective> contiene un mapa del objetivo (<mapInfo>), entonces el atributo objectiveID es obligatorio. Por el contrario, si un <primaryObjective> no contiene un mapa de objetivo, entonces el atributo objectiveID es opcional.

  Para un conjunto de objetivos dados para una actividad (p.e. un <primaryObjective> y multiples <objective> dentro de un elemento <objectives>), todos los objectiveID definidos deben ser únicos. Los Learning Management Systems utilizarán este valor para inicializar el cmi.objectives.n.id (elemento de modelo de datos). El tipo de datos XML es xs:anyURI.

El elemento primaryObjectives contiene los siguientes elementos:

- <minNormalizedMeasure>

El elemento minNormalizedMeasure identifica la medida de satisfacción mínima para un objetivo. Su valor esta normalizado entre -1 y 1 ambos incluidos. Si este elemento se usa para definir la medida de satisfacción mínima para el objetivo primario, entonces el Learning Management System utilizará este valor para inicializar el cmi.scaled_passing_score (para más información consultar la documentación SCORM RTE Book)

Multiplicidad: aparece una o ninguna vez en los elementos <primaryObjective> y <objective>

El tipo de datos XML es xs:decimal. A diferencia de los elementos que hemos decrito hasta ahora este no es un contenedor de

atributos y elementos (no es un elemento padre) sino que tiene su propio valor. Por ejemplo:

<minNormalizedMeasure>0.6</minNormalizedMeasure>

De esta forma se asigna 0.6 a minNormalizedMeasure sin necesidad de más atributos ni elementos.

- <mapInfo>

Es el elemento que contiene la descripción del mapa del objetivo. Define el mapeado para la información objetivo local de una actividad para y desde un objetivo global compartido. Cada actividad puede tener un número ilimitado de mapas de objetivo.

Multiplicidad: aparece 0 o más veces dentro de los elementos <primaryObjective> y <objective>

Contiene los siguientes atributos:

- targetObjectiveID: este atributo es obligatorio. Es el identificador del objetivo compartido global dirigido por el mapeado. El targetObjectiveID es un identificador único y no puede ser una cadena de caracteres vacía ni debe contener espacios en blanco. El tipo de datos XML es xs:anyURI.

- readSatisfiedStatus: es un atributo opcional y su valor por defecto es true. Indica que el estado de satisfacción para un objetivo local identificado debería ser recuperado (con valor true o false) desde el objetivo global compartido identificado cuando el progreso del objetivo local es indefinido. El tipo XML es xs:boolean.

- readNormalizedMeasure: es un atributo opcional y su valor por defecto es true. Indica que la medida normalizada para un objetivo local identificado debería ser recuperado (con valor true o false) desde el objetivo global compartido identificado cuando la medida del objetivo local es indefinido. El tipo XML es xs:boolean.

- writeSatisfiedStatus: es opcional y su valor por defecto es false. Indica que el estado de satisfacción del objetivo local identificado debería ser transferido (con valor true o false) al objetivo global compartido identificado una vez terminado (Termination(” ”)) el intento de la actividad. El tipo de datos XML es xs:boolean.

- writeNormalizedMeasure: es opcional y su valor por defecto es false. Indica que medida normalizada del objetivo local identificado debería ser transferido (con valor true o false) al objetivo global compartido identificado una vez

terminado (Termination(” ”)) el intento de la actividad. El tipo de datos XML es xs:boolea.

Nota: para los atributos readSatisfiedStatus y readNormalizedMeasure:

- si existen multiples elementos <mapInfo> para un objetivo (ya sea <primaryObjective> o <objective>) entonces solo un elemento <mapInfo> debería tener readSatisfiedStatus a true.

- si existen multiples elementos <mapInfo> para un objetivo (ya sea <primaryObjective> o <objective>) entonces solo un elemento <mapInfo> debería tener readNormalizedMeasure a true.

Nota: para los atributos writeSatisfiedStatus y writeNormalizedMeasure:

- Para una actividad, si varios objetivos (<primaryObjectives> o <objective>) tienen elementos <mapInfo> que comparten el mismo targetObjectiveID, entonces solo uno de los objetivos debería tener el atributo writeSatisfiedStatus a true.

- Para una actividad, si varios objetivos (<primaryObjectives> o <objective>) tienen elementos <mapInfo> que comparten el mismo targetObjectiveID, entonces solo uno de los objetivos debería tener el atributo writeNormalizedMeasure a true.

No contiene elementos.

El segundo de los elementos que puede contener el elemento <objectives> es

- <objective>

  Este elemento identifica los objetivos que no contribuyen al rollup asociado con la actividad. El elemento objective solo puede existir si se ha definido previamente un elemento primaryObjective.

  Multiplicidad: puede a parecer 0 o más veces dentro de un elemento objectives.

  Contiene los siguientes atributos:

  - satisfiedByMeasure: es opcional y su valor por defecto es true. Indica que el elemento <minNormalizedMeasure> debe ser utilizado (si el valor del atributo es true) en el lugar de cualquier otro método para determinar si el objetivo asociado con la actividad esta satisfecho. El tipo XML es xs:boolean.

- objectiveID: este atributo es obligatorio. . Representa el identificador asociado con la actividad. El valor asignado al elemento objectiveID es un identificador único, y este valor no puede ser una cadena de caracteres vacía ni puede contener espacios en blanco.

Para un conjunto de objetivos dados para una actividad (p.e. un <primaryObjective> y multiples <objective> dentro de un elemento <objectives>), todos los objectiveID definidos deben ser únicos. Los Learning Management Systems utilizará este valor para inicializar el cmi.objectives.n.id (elemento de modelo de datos). El tipo de datos XML es xs:anyURI.

El elemento <objective> puede contener los elementos :

- <minNormalizedMesaure>
- <mapInfo>

Los cuales ya han sido descritos anteriormente en el elemento <primaryObjective> y no creemos necesario repetir aqui.

<randomizationControls>

Este elemento contiene la información acerca de cómo los hijos de una actividad deberían ser ordenados durante el proceso de secuenciación.

Multiplicidad: puede aparecer una o ninguna vez en el elemento sequencing.

Contiene los siguientes atributos:

- randomizationTiming: es opcional y su valor por defecto es never. Este atributo indica cuando debe ocurrir la ordenación de los hijos de la actividad. Tipo de datos XML xs:token Puede tener los siguientes valores:
  - never
  - once
  - onEachNewAttempt
- selectCount: es opcional y su valor por defecto es 0. Este atributo indica el número de actividades hijo que deben ser seleccionados del conjunto de actividades hijo asociadas con la actividad. Tipo de datos XML xs:nonNegativeInteger.
- reorderChildren: es opcional y su valor por defecto es false. Este atributo indica que el orden de las actividades hijo es aleatorio. Tipo de datos XML xs:boolean.

- selectionTiming: es opcional y su valor por defecto es never. Este atributo indica cuando debe ocurrir la selección. Tipo XML xs:token. El atributo contiene un valor de entre los siguientes:
    - never
    - once
    - onEachNewAttempt

No contiene otros elementos

<deliveryControls>

Este elemento contiene información acerca de cómo los hijos de una actividad deberían ser ordenados durante el proceso de secuenciación.

Multiplicidad: aparece una o ninguna vez en el elemento sequencing.

Contiene los siguientes atributos:

- tracked: es opcional su valor por defecto es true. Este atributo indica que la información de progreso del objetivo y la información de actividad/intento del objetivo para este intento debe ser registrada (true o false) y los datos contribuirán a el rollup de la actividad padre. Tipo de datos XML xs:boolean.
- completionSetByContent: es opcional y su valor por defecto es false. Este atributo indica que es estado de finalización de un intento de una actividad debe ser determinado por el Sharable Content Object (true o false). El Tipo XML es xs:boolean.
- objectiveSetByContent: es opcional y su valor por defecto es false. Este atributo indica que el estado de satisfacción del objetivo para el objetivo asociado de la actividad que contribuye al rollup será determinado por el Sharable Content Object. El tipo de datos XML es xs:boolean.

No contiene elementos.

Apéndice C – Cómo aumentar la capacidad de la herramienta para el etiquetado de nuevas gramáticas

Para explicar este apartado, nos basaremos en el ejemplo de la secuenciación.
Para añadir la secuenciación a la aplicación ya existente

- NO SE HA MODIFICADO EL CODIGO DE LA APLICACIÓN.
- Únicamente se ha cambiado el documento xml correspondiente con el content package.
- Después, se han realizado unas ligeras modificaciones sintácticas.

A continuación se describen estas modificaciones, con un ejemplo:
Se parte del archivo que describe la secuenciación conseguido en la pagina de IMS, **imsss_v1p0.xsd** y del content package de los objetos que manejaba el lomeEditor 2005 **imscp_v1p1.xsd.**
Se logra que la gramatica resultante maneje la etiqueta sequencing.
Se toma del **imsss_v1p0.xsd** la parte que interesa, que es donde se declaran los tipos y los elementos.

- Declaración del elemento "sequencing"

```
<xs:element name = "sequencing" type = "sequencingType"
block = "#all">
</xs:element>
```

- Declaración del tipo complejo "sequencingType"

```
<xs:complexType name = "sequencingType">
<xs:sequence>
<xs:element name = "controlMode" type = "controlModeType"
                    block = "#all" minOccurs = "0">
</xs:element>
 <xs:element name = "sequencingRules" type =
"sequencingRulesType" block = "#all" minOccurs = "0"/>
<xs:element name = "limitConditions" type =
"limitConditionsType"
block = "#all" minOccurs = "0"/>
<xs:element name = "auxiliaryResources" type =
"auxiliaryResourcesType"
block = "#all" minOccurs = "0"/>
<xs:element name = "rollupRules" type = "rollupRulesType"
```

```
        block = "#all" minOccurs = "0"/>
        <xs:element name = "objectives" type = "objectivesType"
         block = "#all" minOccurs = "0">
        </xs:element>
        <xs:element name = "randomizationControls" type =
"randomizationType"
        block = "#all" minOccurs = "0"/>
        <xs:element name = "deliveryControls" type =
"deliveryControlsType"
         block = "#all" minOccurs = "0"/>
        <xs:any namespace = "##other" processContents = "strict"
minOccurs = "0" maxOccurs = "unbounded"/>
        </xs:sequence>
        <xs:attribute name = "ID" type = "xs:ID"/>
        <xs:attribute name = "IDRef" type = "xs:IDREF"/>
        </xs:complexType>
```

A la hora de introducirlos en el content package hay que realizar el siguiente cambio sintáctico:

- **<xsd:element name = "sequencing" type = "sequencingType"/>**
-

Es decir cambiar xs por xsd para que el content package lo reconozca y añadirlo en la zona de declaración de elementos.

Del mismo modo se procede con el tipo complejo ene la zona de declaracion de complexType.

```
<xsd:complexType name = "sequencingType">
            <xsd:sequence>
                    <xsd:element ref = "controlMode" minOccurs = "0"/>
                    <xsd:element ref = "limitConditions" minOccurs = "0"/>
                    <xsd:element ref = "auxiliaryResources" minOccurs =
"0"/>
                    <xsd:element ref = "randomization" minOccurs = "0"/>
                    <xsd:element ref = "deliveryControls" minOccurs = "0"/>
                    <xsd:element ref = "sequencingRule" minOccurs = "0"/>
                    <xsd:element ref = "rollupRules" minOccurs = "0"
                    maxOccurs = "unbounded"/>
                    <xsd:element ref = "objectives" minOccurs = "0"
                    maxOccurs = "unbounded"/>
                    <xsd:group ref = "grp.any"/>
            </xsd:sequence>
            <xsd:attributeGroup ref = "attr.identifierref"/>
            <xsd:attributeGroup ref = "attr.identifier"/>
```

De este modo la aplicación ya reconocería la etiqueta sequencing y conocería su estructura interna.

Sólo quedaría proceder de la misma forma con el resto de las etiquetas.

Esto permite reconocer todo tipo de nuevas etiquetas sin necesidad de modificar el código de la aplicación (p.e. etiquetas adl)

A continuación se dan unos detalles prácticos sobre la implementación software:

- Los objetos de aprendizaje IMS_MD e IMS_SS se validarán con estas gramaticas modificadas sintácticamente.
- La ruta de las gramáticas CP modificadas sintácticamente es: (relativa a la carpeta LOMEditor-v4.7, que se proporciona en el CD)

  LOMEditor-v4.7/ LOMEditor/CPS_Modificados/

- Las variables de la aplicación que hacen referencia a estos archivos son:
  - En la clase LOMEditor.Mainframe.java:
    - En el método abrirObjetoAprendizaje(), la variable "cpXSD"
  - En la clase LOMEditor.composicion.ControladorComposicion.
    - La variable "cpXSD", que aparece en varios métodos.
  - En la clase LOMEditor.composicion.PluginCpmposición, la misma variable.

Apéndice D – Bibliografía

PÁGINAS WEB CONSULTADAS:

IMS Global Learning Consortium. http://www.imsproject.org

Advanced Distributed Learning. http://www.adlnet.gov

Foro de Estándares de e-learning. http://www.elearningworkshops.com

Observatorio de e-learning. http://madeira.ls.fi.upm.es/o-e-learning/index.jsp?pagina=22

Reload Project. http://www.reload.ac.uk

Eduforge http://eduforge.org

Sun Microsystems http://java.sun.com

Universidad de Milwokee
http://www.uwm.edu/Dept/CIE/AOP/learningobjects.html

Pligins en java (El rincón del programador)
http://www.elrincondelprogramador.com/default.asp?pag=articulos/leer.asp&id=61

ARTÍCULOS CONSULTADOS:

- *Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy* . Por David A. Wiley

- *The nature and origin of instructional objects*. Por Robert Richards

- *Learning object systems as construtivist learning environments. Related assumptions, theories, and aplicatios.* Por Brenda Bannan-Ritland, Nada Debbagh y Kate Murphy

- *Designing resource-based learning and performance support systems.* Por Michael J. Hannafin, Janette R. Hill James E. McCarthy.

- *Learning objects to support inquiry-based online learning.* Chandra Hawley Orrill.

- *Designing learning objects to mass customize and personalize learning.* Por Margaret Martinez.

- *Evaluation of learning objects and instruction using learning objects.* Por David D. Williams.

- *Battle stories from the field: Wisconsin online resource center learning objects project.* Por Kay Chitwood, Carol May, David Bunnow y Terri Langan.

- *A university-wide system for creating, capturing, and delivering learning objects.* Joseph B. South y David W. Monson.

- *Collaboratively filtering learning objects.* Mimi M. Recker, Andrew Walker, David A. Willey

- *Knowledge objects and mental models.* Por M. David Merril

- *The future of learning objects.* H. Wayne Hodgins.

LIBROS CONSULTADOS:

The SCORM Implementation Guide: A Step by Step Approach, Release Date: 11/27/2002, Author: Alexandria ADL Co-Laboratory

Design Patterns, E. Gamma et al., Addison Wesley, 1996.

Agradecimientos

En primer lugar al profesor director del proyecto Antonio Sarasa Cabezuelo por su colaboración y ayuda en el desarrollo del trabajo.

Por su aporte teórico y por la idea desarrollada, agradecemos su trabajo a los autores del LOMEditor 2005: Raul Arriola, Susana de la Iglesia, Fran J. Piquer.

Agradecemos al profesor Baltasar Fernández Manjón su colaboración informativa en el ámbito del trabajo con plugins.