CDS (Courseware Development System): Una Herramienta para Construir, Administrar y Publicar de Componentes Educativos Reusables Basados en XML

Emilio Ormeño (*), Sergio Ochoa (+)

(*) eormeno@iinfo.unsj.edu.ar
Instituto de Informática, F.C.E.F.y N.,
Universidad Nacional de San Juan.
Argentina.

(+) sochoa@dcc.uchile.cl
Departamento de Ciencia de la Computación
Universidad de Chile, Chile.

Resumen

En los últimos años ha aparecido una gran cantidad de herramientas para desarrollar courseware, las cuales han facilitado notablemente la construcción de estos productos, pero también han traído grandes restricciones. Algunas de las restricciones más serias son producto de la incompatibilidad entre los formatos de material didáctico que maneja cada una. Debido a esto, las instituciones educativas están obligadas a optar por una única herramienta, a reusar sólo el material didáctico desarrollado a través de ella, y a conformarse con la funcionalidad que esta herramienta le brinde. Para contribuir a la solución de estos problemas, este artículo presenta una herramienta para construir, administrar y publicar componentes educativos reusables, los cuales adhieren a especificaciones abiertas. Estos componentes pueden ser utilizados por cualquier herramienta de desarrollo de courseware que soporte las especificaciones involucradas, lo cual hace posible el intercambio de material didáctico entre ellas.

Palabras Claves: Herramienta de Desarrollo, Componentes Educativos, JavaBeans, XML.

1. Introducción

Nos encontramos en un tiempo en que la aplicación de las tecnologías de la información al ámbito educativo está de moda, multiplicándose las iniciativas y experiencias sobre el tema en todo el mundo. Dentro de estas tecnologías, sin duda la que ha irrumpido con mayor fuerza ha sido Internet, y específicamente el World Wide Web (WWW). Sin embargo, actividad no es sinónimo de madurez. Por el contrario, observamos a nuestro alrededor una variada gama de propuestas metodológicas y de herramientas, propia de una etapa de descubrimiento y de una búsqueda inicial de caminos a seguir. En este último tiempo, son muchas las herramientas que han surgido, en relación con la construcción de coursewares que apoyan la educación en distintos escenarios. Estos courseware proveen un ambiente educativo que consta de material didáctico multimedial e interactivo que puede ser accedido a través de Internet, y de un conjunto de herramientas que permiten la comunicación/colaboración entre profesores y alumnos. Normalmente estos courseware embeben los contenidos y las actividades previstas para llevar adelante el proceso de enseñanza-aprendizaje de un curso.

Las herramientas para construir coursewares son conocidas como *herramientas de courseware* [Lan01], y entre las más populares están: *Lotus Learning Space* [Lls01], *WebCT* [Gol96, Web01], *TopClass* [Top01], *Virtual-U* [Har99, Vir01], *WCB* [Wcb01], *CourseInfo* [Cou01], y *ToolBook II* [Too01]. Estas herramientas han facilitado notablemente el desarrollo de courseware, pero también han traído consigo diversas limitaciones. Varias de ellas son producto de la incompatibilidad entre los formatos de material didáctico que maneja cada una. Esto ha obligado a las instituciones educativas a tener que optar por una única herramienta, a reusar sólo el material didáctico desarrollado a través de ella, y a conformarse con la funcionalidad que esta herramienta le brinde.

Debido a las limitaciones antes mencionadas, en los últimos 2 años han surgido algunos proyectos que intentan construir material didáctico que adhiere a especificaciones abiertas [Ros99], como una forma de superar estas limitaciones. El material didáctico obtenido está orientado a áreas específicas del conocimiento (física, matemática, ingeniería, etc.), y no cubre las necesidades generales de un courseware (capítulos, secciones, ejemplos, etc.). Por otra parte han surgido recomendaciones como IMS (Instructional Management System) [Ims01] y LOM (Learning Object Metadata) [Iee01] que establecen un formato de material didáctico intercambiable. Si bien esto es un gran logro, en la práctica aún las herramientas de courseware siguen sin soportar estos formatos. En cambio han incluido otros formatos usados en la industria del software, como por ejemplo: JavaBeans [Doh00], ActiveX[Cha96], y XML[Har01]. Éstos han tenido éxito en la compatibilización de formatos de software, por lo que podrían ser usados para resolver (en parte) el problema planteado.

Siguiendo esta línea, en la próxima sección se presenta una herramienta llamada CDS (Courseware Development System), que permite la construcción, administración y publicación material didáctico, a través de componentes educativos reusables. Estos componentes son piezas de código que respetan un formato estándar, y que permiten la construcción por medio del ensamble de éstos. Puede concebírselos como un bloque lego [Orf98]. El rango de componentes que maneja CDS va desde una imagen, hasta un courseware. Cada uno de éstos puede ser utilizados por cualquier herramienta que soporte las especificaciones involucradas.

La construcción basada en componentes tiene un sin número de ventajas [Pres00, Brow98], algunas de ellas son:

- *Aumenta la capacidad para exportar contenido/funcionalidad a otros formatos.* El hecho de que respeten formatos estándares hace que sea relativamente simple generar traductores entre diferentes formatos.
- *Reduce el esfuerzo de desarrollo.* Debido a que la construcción se lleva a cabo a través del ensamble de estos módulos, que muchos casos ya están instanciados, el tiempo y el costos de desarrollo se reduce notablemente. Algunos, como Pressman [Pres00], sostienen que incluso se mejora la calidad del producto final.
- *Mejora la capacidad de reingeniería.* Debido a que los componentes tienen límites y objetivo bien definidos, la reingeniería se reduce al reemplazo y la reorganización de éstos.
- *Aumentan la capacidad de prototipación.* Debido a que permiten construcción y reingeniería rápida, la velocidad de prototipación aumenta notablemente. Esto es muy importante para el desarrollo de courseware [Ude00].

La sección 2 describe la arquitectura de la herramienta propuesta, como así también ciertas características técnicas de la misma. En la sección 3 se describe el sistema administrador de componentes educativos, que representa la mayor parte de la funcionalidad de CDS. Luego, en la sección 4 se presentan las características de estos componentes. Finalmente, en la sección 5 se discuten las implicancias de este trabajo, y las conclusiones a la que hemos arribado.

2. La Herramienta CDS (Courseware Development System)

Esta herramienta trabaja bajo una arquitectura Cliente/Servidor (Figura 1). El servidor se almacena el courseware (repositorio de componentes), y los clientes interactúan con él. Los clientes pueden ser: un browser, o CDS. Los browsers sólo pueden acceder al courseware e interactuar con éste, según lo previsto en su diseño. En cambio CDS, puede acceder, copiar o actualizar cualquier componente del courseware. Los componentes que maneja CDS representan las actividades y contenidos que forman parte de la mayoría de los courseware, y están almacenados en forma

organizada, a través de un repositorio. Este repositorio puede contener cero o más courseware, donde cada uno representa un conjunto de componentes educativos (capítulos, secciones, ejemplos, etc.), y de vínculos entre ellos, cuya composición permite la obtención del courseware.

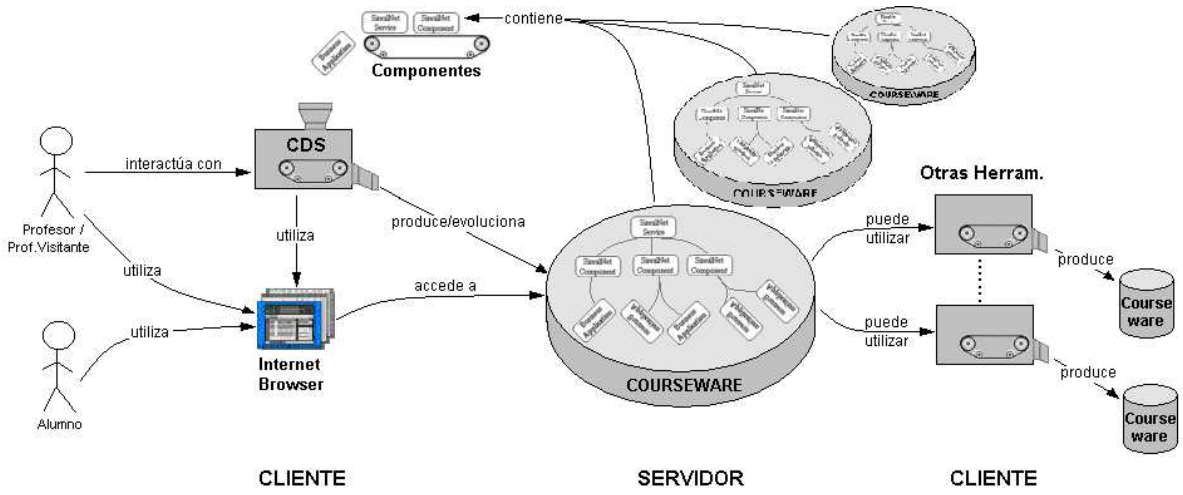

Figura 1. Escenario de Trabajo de CDS

Cada uno de los componentes educativos ha sido implementado como un documento XML que es traducido a javabean o a documento HTML, dependiendo del cliente que interactúa con él (una aplicación java o un browser). Se escogió el formato JavaBean como formato de implementación porque es el único que adhiere a una especificación abierta, excepto, la recién publicada versión 3.0 de CORBA [Omg01]. Además, se escogió XML como formato de representación de estos componentes en ambientes WEB, debido a que es de amplia aceptación, está basado en un estándar, y que permite una buena la manipulación de la información [Mcl00].

Para poder acceder a los componentes, la herramienta CDS define tres roles: *profesor*, *profesor visitante*, o *estudiante*. Estos roles vinculan a un usuario, con un courseware, asignándole los derechos de acceso a éste, y a todos los componentes que forman parte de él. Todos los usuarios interactúan con el repositorio de componentes a través de una aplicación cliente, que como se mencionó antes, puede ser el módulo administrador del sistema, o un browser. Además, éstos deben autenticarse para poder accederlo.

El módulo administrador, puede ser utilizado sólo por usuarios con rol de profesor o profesor visitante. Este módulo es una aplicación hecha en java, que le permite a un usuario consultar, crear, copiar, modificar y borrar componentes educativos en aquellos courseware, dónde él figure con rol de profesor. En el caso de profesores visitantes, el módulo administrador sólo les permite consultar y copiar componentes, alentando así el reuso del material didáctico entre distintos profesores.

Por otra parte, el acceso al repositorio de componentes a través de un browser sólo permite consultar la información de los courseware, y utilizar la funcionalidad diseñada en cada uno, como por ejemplo: comunicarse a través de un foro de discusión o un chat, realizar un examen o un autotest en línea, etc. Los usuarios con rol de profesor o profesor visitante, pueden revisar cualquier componente que forme parte del courseware (Figura 2), mientras que los estudiantes sólo pueden acceder al componente *courseware*.

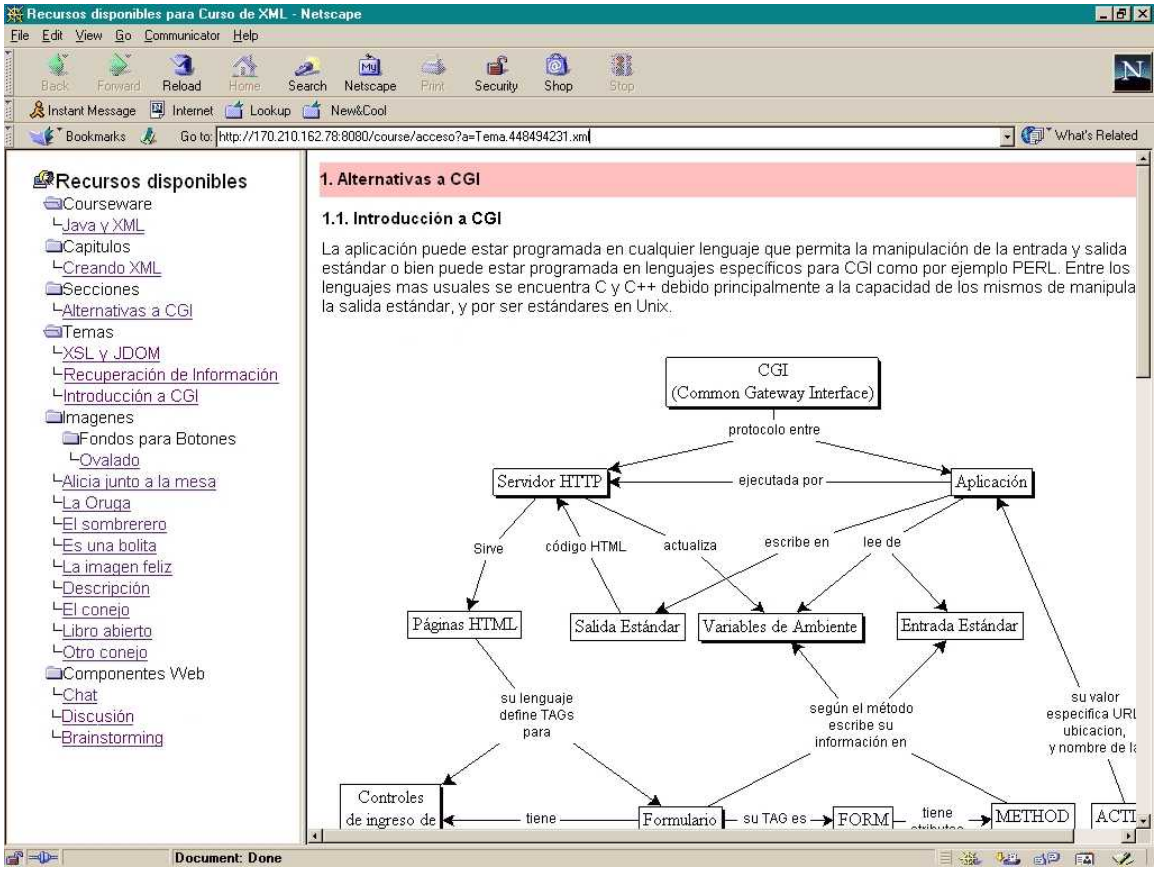

Figura 2. Acceso al Repositorio a través de un Browser, con rol de Profesor o Profesor Visitante

A la hora de crear o modificar un courseware el profesor trabaja con el módulo de administrador del sistema, que provee soporte para integrar componentes a través de operación simples como: “crear una instancia”, “cortar/copiar y pegar” o “configurar”. Los componentes que puede utilizar el profesor a la hora de construir un courseware, pueden ser nuevas instancias de los componentes educativos predefinidos en CDS, o bien copias de componentes ya instanciados, que forman parte de otros coursewares. Por ejemplo, el profesor podría querer crear un tema vacío para luego ir creándole el contenido, o bien, podría hacer una copia un tema que pertenece a otro courseware, para luego pegarlo en el suyo.

La herramienta CDS provee dos tipos de componentes educativos: *contenidos* y *actividades*. Los componentes de tipo contenido son: courseware, capítulo, sección, tema, ejemplo, ejercicio, motivación y resumen. Cada uno de estos componentes soporta contenidos con material multimedial, y su función es almacenar el material didáctico que forma parte del courseware.

Los componentes de tipo actividades son: chat, foro de discusión, brainstoming, test, y presentador distribuido. Estos componentes pueden ser instanciados y pegados como parte del contenido de cualquiera de los componentes anteriores, y su objetivo es dar soporte a las actividades de enseñanza-aprendizaje de un curso. En la medida que se desarrollen nuevos componentes que adhieran a la especificación javabean, éstos pueden ir agregándose a la herramienta CDS, para permitir su utilización. Los componentes creados por esta herramienta pueden ser reusados por cualquier otra, que soporte la especificación antes mencionada. A continuación se describe el servidor de la herramienta propuesta.

2.1. Descripción del Servidor de CDS

Técnicamente, el servidor de este sistema requiere: un *Servidor HTTP* con capacidad para ejecutar servlets, y un *Framework de Publicación Múltiple* como Cocoon [Asf01]. Además, necesita tener instalado 2 servlets: *Acceso* y *Guardar*, los cuales han sido específicamente diseñados para trabajar en este escenario; y el *Repositorio de Componentes* (Figura 3).

El framework de publicación, Cocoon en la implementación actual de CDS, recibe solicitudes http, y entrega información en distintos formatos (html, xml, pdf, etc.), según la hoja de estilo asociada al documento XML solicitado.

El servidor web de Cocoon debe estar protegido del acceso público, de forma tal que sólo el servlet *Acceso* conozca la clave y usuario para poder activarlo. Este servlet es la puerta de entrada al servidor, y es el encargado de la autentificación. Cualquier usuario que desee acceder a algún componente, deberá hacerlo a través de este servlet.

Cuando un usuario desea acceder directamente al componente *curso1.xml*, dado que el framework de publicación está protegido, se desplegará una ventana de identificación. Después de que el usuario se identifica, el servlet Acceso procede a corroborar el usuario y clave con la información provista por los documentos *personas.xml* y *repositorio.xml*. Si la información está correcta, el servlet le entrega la solicitud de acceso al framework de publicación múltiple (Cocoon), para que le proporcione al usuario el componente solicitado, en función de su perfil.

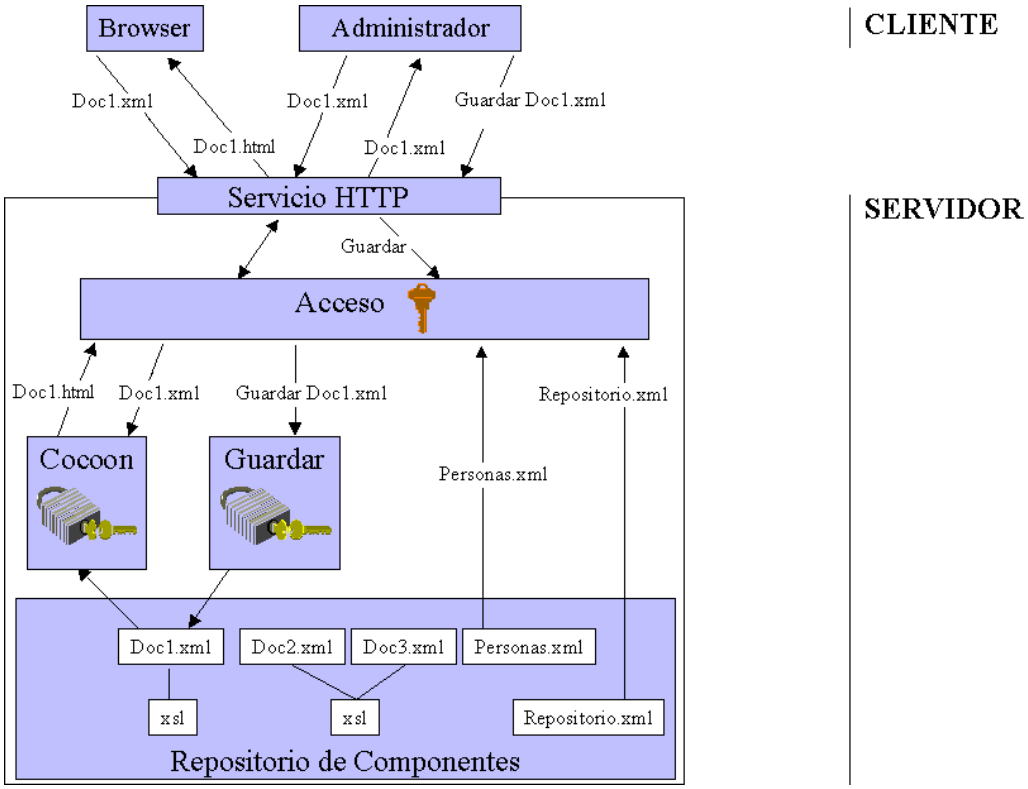

Figura 3. Arquitectura de CDS.

Cabe aclarar que el documento *repositorio.xml* contiene todas las carpetas y enlaces a los documentos XML, que representan a los componentes dentro del repositorio. Cada carpeta tiene,

entre sus atributos, el nombre del usuario que puede accederla. Por su parte, el documento *personas.xml* contiene los usuarios y los grupos de los que son miembro. Por ejemplo:

```
<Personas>
 <Persona usuario="ggonzalez" clave="ggonzalez" apellido="González" nombre="Guillermo">
   <Miembro grupo="Curso de XML" perfil="alumno" />
   <Miembro grupo="Curso de Informática" perfil="alumno" />
 </Persona>
```

En el elemento <Miembro> se dice a qué grupo pertenece el usuario y con qué perfil entra en ese grupo. Cada courseware tiene grupo de usuarios, y de esa manera se les asignan los derechos.

Por otra parte, el *repositorio de componentes educativos* contiene un conjunto de carpetas que representan distintos coursewares. Cada courseware contiene un conjunto de componentes y de relaciones entre componentes que forman parte de él, y una lista de usuarios con su respectivo rol. Ese rol permite que se le asignen derechos a los usuarios, para acceder al courseware.

El servlet *Guardar* es otro componente que forma parte del servidor, y sirve para actualizar el estado de los componentes que están en el repositorio. Para ello se vale de las capacidades que tiene cada componente para generar una representación XML de sí mismo. Esta representación XML es almacenada en el repositorio como una instancia de un componente. Los componentes también tienen la funcionalidad inversa, o sea, transformar la representación XML en JavaBean, por lo que la bidireccionalidad de los datos está asegurada.

Como se mencionó antes, la aplicación cliente puede ser un browser o el módulo administrador del sistema. El primer caso no merece mayor explicación, pero si el segundo. En la siguiente sección se describe el módulo administrador del sistema, y la interacción entre éste y el servidor.

3. El Módulo Administrador del Sistema

Este módulo consta de un entorno de desarrollo de coursewares que facilita el diseño y administración de componentes educativos que conforman un curso. Estos componentes tienen aquí una representación en forma de JavaBean, y en el repositorio una representación como documento XML. Esta herramienta permite la construcción de courseware integrando componentes educativos predefinidos. De esa manera, le permite al profesor y/o asistentes, abstraerse de los detalles de implementación, y construir un producto reusando y adaptando un conjunto de módulos predefinidos (componentes). Esto generalmente reduce el esfuerzo (tiempos y costos) asociado a esta tarea, y aumenta la calidad del producto final [Pre00, Boe99]. En la siguiente figura se puede ver el aspecto general de este módulo.

El Administrador está compuesto por cinco partes, bien diferenciadas (Figura 4): (1) *manejador del repositorio*, (2) *panel de estructura*, (3) *panel de visualización de componentes*, (4) *barra de herramientas*, y (5) *barra de menú*. En las siguientes 5 secciones, se describe la función y las características de cada una de ellas.

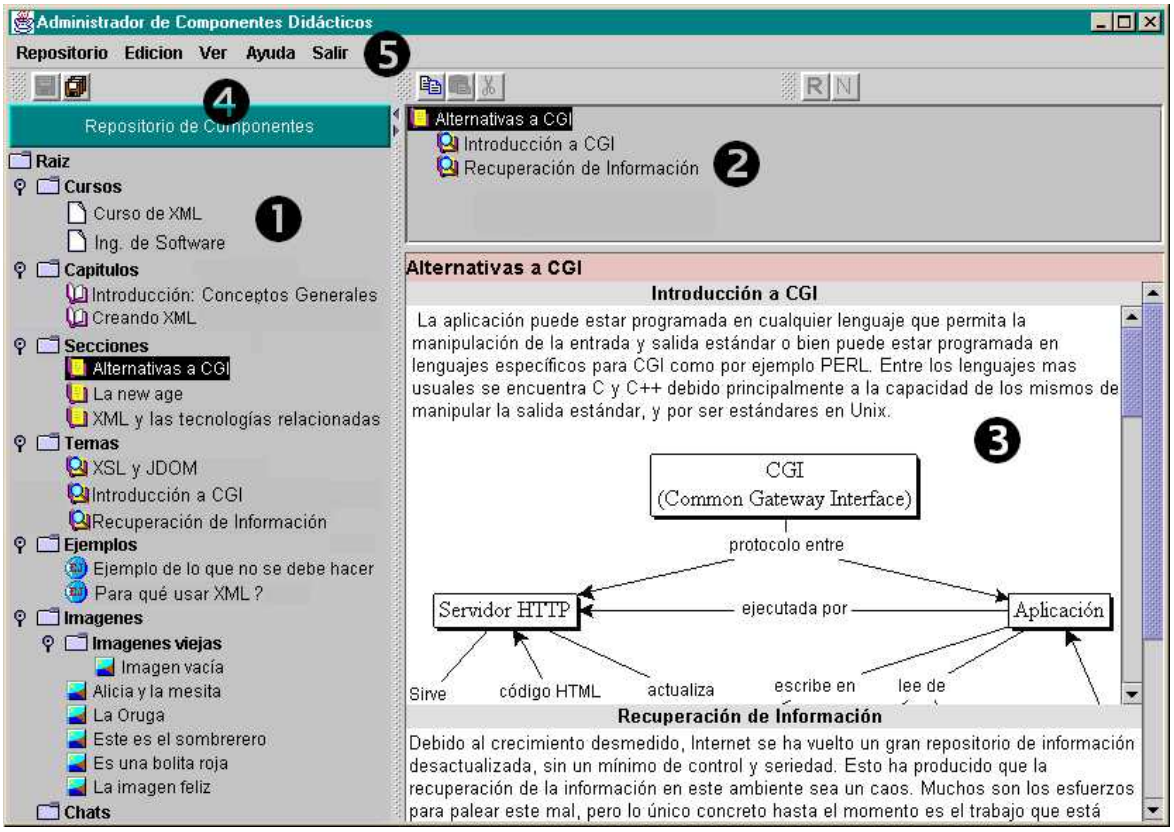

Figura 4. Aspecto del Módulo Administrador del Sistema.

3.1. Manejador del Repositorio

El manejador del repositorio es el encargado de visualizar los componentes disponibles en la forma de un árbol de carpetas y enlaces. Este árbol tiene una carpeta raíz, y a su vez, una carpeta para cada tipo de componente que el sistema puede administrar (Figura 5).

Figura 5. Categorías de Componentes

El manejador del repositorio es también el encargado de instanciar cada documento XML, al que hace referencia cada uno de sus componentes *Enlace.* Además, también se encarga de guardar, a través del servlet *Guardar*, las modificaciones que pudieran haberse realizado. Cada componente *Carpeta* puede posee un menú Pop-up que permite la creación de nuevas instancias de enlaces, y por ende, nuevos componentes. En pocas palabras el componente *manejador del repositorio*, es una mezcla de sistema de almacenamiento y paleta de componentes.

Representación XML del Manejador del Repositorio

Este manejador también es tratado como un componente por el administrador, y como tal, es en realidad un documento XML con una representación JavaBean. Un ejemplo de esta representación, es el siguiente documento XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Repositorio SYSTEM "DTD/Repositorio.dtd">
<Repositorio>
	<Carpeta nombre="Raiz" clase="Raiz" grupo="Curso de XML">
		<Carpeta nombre="Cursos" clase="Curso" perfil="alumno">
			<Enlace nombre="Curso de XML" clase="Curso" archivo="Curso1.xml"/>
		</Carpeta>
		<Carpeta nombre="Capitulos" clase="Capitulo" perfil="docente">
			<Enlace nombre="Generalidades" clase="Capitulo" archivo="cap1.xml" />
		</Carpeta>
		<Carpeta nombre="Secciones" clase="Seccion" perfil="docente">
			<Enlace nombre="Alternativas a CGI" clase="Seccion" archivo="s1.xml"
		/>
		</Carpeta>
		<Carpeta nombre="Temas" clase="Tema" perfil="docente">
			<Enlace nombre="JDOM" clase="Tema" archivo="a-213.Tema.xml" />
		</Carpeta>
		<Carpeta nombre="Ejemplos" clase="Ejemplo" perfil="alumno">
			<Enlace nombre="Why XML?" clase="Ejemplo" archivo="Ej1.xml" />
		</Carpeta>
		<Carpeta nombre="Imagenes" clase="Imagen" perfil="docente">
			<Enlace nombre="Alicia" clase="Imagen" archivo="Alicia.Imagen.xml"/>
			<Enlace nombre="La Oruga" clase="Imagen" archivo="Oruga.xml"/>
		</Carpeta>
		<Carpeta nombre="Chats" clase="cscl.ChatUser" perfil="docente">
		</Carpeta>
	</Carpeta>
</Repositorio>
```

El componente *Carpeta*, tiene 3 atributos: *nombre*, *clase*, *perfil*. El atributo nombre, contiene el nombre de la carpeta. El atributo clase, representa la clase de los enlaces que contiene la carpeta; y el atributo perfil, indica el perfil de usuario requerido para acceder a esa carpeta. El componente Carpeta que tiene por nombre 'Raíz', es la carpeta contenedora de todas las demás, y posee un atributo *grupo* que indica el nombre del grupo (lista de usuarios) donde deberá verificar los derechos de acceso al courseware.

Por otra parte, el componente *Enlace*, también tiene 3 atributos: *nombre*, *clase* y *archivo*. El atributo nombre, contiene el nombre del enlace. El atributo clase, representa la clase del enlace; y el atributo archivo, contiene el nombre del archivo o documento XML al que se hace referencia. En la implementación actual, el atributo *clase* es tomado por el manejador del repositorio para poder instanciar el componente, debido a que ese atributo contiene el nombre de la clase del JavaBean. Este atributo también sirve para que el manejador del repositorio obtenga, mediante introspección, el icono correspondiente a la clase.

3.2. Panel de Estructura

El panel de estructura es el encargado de visualizar y de proveer un acceso directo a los subcomponentes que forman parte de un componente mayor. En la Figura 4 se puede ver, a través de este panel, los componentes que constituyen una sección titulada "Alternativas a CGI". En este caso se muestran los 2 temas que forman parte de ella. Si un usuario hace clic sobre uno de estos componentes, éste será mostrado en el panel de visualización de componentes (3).

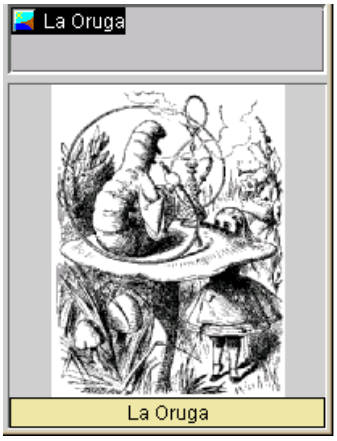

Figura 6. Componente con Estructura Trivial

Obviamente, en componentes que no poseen una estructura interna, el panel de estructura no provee demasiada información. Por ejemplo, en la figura 6 se puede observar la información que el panel de estructura y el de visualización de componentes, muestran a cerca de un componente *Imagen.*

3.3. Panel de Visualización de Componentes

Como ya se ha visto en explicaciones anteriores, este panel es el encargado de visualizar y proveer acceso al contenido de un componente instanciado. Las operaciones que pueden hacerse sobre éste dependen del rol del usuario. Un usuario de tipo *profesor* podrá consultar, crear, copiar, modificar y borrar cualquier componente. En cambio los *profesores visitantes*, sólo podrán consultar y copiar componentes.

3.4. Barra de Herramientas, y Menú

Las barras de herramientas proveen un acceso directo a las funciones del componente que se está visualizando en el panel de visualización de componentes. Estas barras se crean, activan o desactivan en forma automática en función de las acciones publicadas por los JavaBeans.

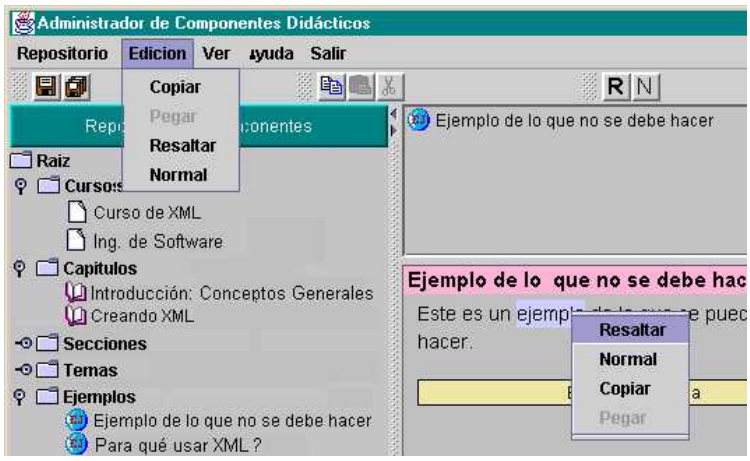

Figura 7. Acciones Permitidas sobre un Componente

En la figura 7 se puede observar que, dentro de un componente de tipo *Ejemplo*, una vez que el usuario ha resaltado una porción del texto, automáticamente a través del menú tendrá acceso a las 3 funciones permitidas sobre ese JavaBean: *resaltar*, *normal*, y *copiar.* Además, también se activarán las operaciones de *guardar componente*, *guardar courseware*, *copiar*, *resaltar*, *normal*, que forman parte de la barra de herramientas.

4. Componentes Educativos de CDS

En el caso de la herramienta CDS, como se mencionó antes, hay dos tipos de componentes: de *contenido* y de *actividad*. Los componentes de tipo contenido son: courseware, capítulo, sección, tema, ejemplo, ejercicio, motivación y resumen. Éstos representan el material didáctico que forma parte del courseware. Por otra parte, los componentes de tipo actividad son: chat, foro de discusión, brainstoming, test, y presentador distribuido. Estos componentes pueden ser instanciados y pegados, como parte del contenido de cualquiera de los componentes anteriores. Su objetivo es dar soporte a las actividades de enseñanza-aprendizaje de un curso. Para que estos componentes puedan ser

accedidos a través de la Web, los de tipo contenido son transformados en documentos HTML, y los de tipo actividad en applets.

El módulo administrador del sistema, como entorno de desarrollo, debe proveer formas de agregar cualquier otro tipo de componente. Pero dado que éstos son, en última instancia, documentos XML, los JavaBeans que los visualizan deben cumplir con ciertas características para que puedan ser administrados. A continuación se enumeran las características más relevantes, sin entrar en detalles de implementación. Estas características son:

- El JavaBean debe heredar de *java.awt.Component*.
- El JavaBean debe implementar la interfaz *Estructurable*.
- El JavaBean puede implementar el método *getJDOM*.
- El JavaBean puede implementar el método *setJDOM*.

La interfaz *Estructurable* provee métodos para que el módulo administrador del sistema pueda conocer la estructura interna de los componentes. Por ejemplo, un componente *Capítulo* debe publicar cuales son sus *hijos* y cuales son sus *hojas* en el árbol JDOM que lo representa [Mcl00, Jdo01]. Esta interfaz también sirve para trasladar eventos (por ejemplo, click derecho sobre el componente o tecla presionada) hacia el módulo administrador. En base a esto, el administrador decidirá si mostrar un menú Pop-up o indicarle al repositorio que se ha producido un cambio en el estado del componente, por lo tanto debe activar la opción *guardar* en la barra de herramientas. Además, la interfaz *Estructurable* tiene un método llamado *getAcciones,* el cual retorna un arreglo con las acciones que se pueden realizar sobre el objeto y su referencia a métodos locales. Por ejemplo, el siguiente fragmento de código Java, muestra las acciones publicadas por el componente *Contenido*.

```
AccionGeneral accion = new AccionGeneral("Resaltar");
accion.putValue("menu","Popup,Edicion");
accion.putValue("toolbar","barraEstilo");
acciones[0] = accion;

accion = new AccionGeneral("Normal");
accion.putValue("menu","Popup,Edicion");
accion.putValue("toolbar","barraEstilo");
acciones[1] = accion;

accion = new AccionGeneral("Copiar");
accion.putValue("menu","Popup,Edicion");
accion.putValue("toolbar","barraEdicion");
acciones[2] = accion;

accion = new AccionGeneral("Pegar");
accion.putValue("menu","Popup,Edicion");
accion.putValue("toolbar","barraEdicion");
acciones[3] = accion;
```

Cada acción debe indicar si tiene o no que ser mostrada en un menú. En ese caso también debe indicar si la acción aparecerá en un menú “Pop-up”, en la barra de herramientas o en ambos. En caso de que deba ser mostrada en una barra de herramientas, se debe proveer el nombre de la barra.

Cada JavaBean posee dos métodos setJDOM y getJDOM, los cuales obtienen e instancian respectivamente los documentos XML, utilizando la librería JDOM. El componente puede o no haberlos definido, dado que a la hora de instanciar el componente, el manejador de repositorio averigua a través de introspección si los tiene o no.

Cuando desde el módulo administrador, se le solicita el componente *'x'* (con doble clic), el manejador del repositorio abre el documento XML vinculado, y luego carga una clase que tenga el mismo nombre que el elemento raíz del documento. Posteriormente, como no desconoce el formato del componente, busca por introspección dentro de la clase, para ver si éste posee un método llamado *setJDOM*. Si lo encuentra, lo invoca utilizando como parámetro al resultado parseado del XML. De ahí en mas, el mismo componente sabe como instanciarse.

Para realizar el proceso inverso, o sea, cuando se solicita guardar un componente que ha sido modificado, el manejador del repositorio busca el método *getJDOM* en el componente. Si lo encuentra, lo invoca, ya que este método devuelve una estructura JDOM con todas las instrucciones de procesamiento que necesita (hoja de estilo, instrucciones para cocoon, etc.). A partir de esta estructura, el manejador del repositorio, genera el documento XML y hace un 'Upload' hacia el repositorio de componentes, utilizando el servlet *Guardar*. Siempre los métodos (setJDOM y getJDOM) de un componente son buscados a través de introspección, para no obligar a los JavaBeans a implementar interfaces para manejar esto.

5. Trabajo a futuro

Las imágenes de las ventanas y páginas de CDS mostradas en este documento, corresponden al primer prototipo del mismo; y si bien este no va ha ser puesto a prueba en un Courseware real, si lo será el siguiente prototipo que está actualmente en elaboración con ciertos cambios en su estructura.

6. Conclusiones

En los últimos años, la incompatibilidad de formatos de la información entre las herramientas de courseware [Lan01], ha llevado a que las instituciones educativas que utilizan este tipo de herramientas, tengan que: optar por una única herramienta de courseware, reusar sólo el material didáctico desarrollado a través de ella, y conformarse con la funcionalidad que esta herramienta le brinde. Para aportar un principio de solución a este problema, este artículo presenta una herramienta de desarrollo de courseware basada en componentes XML, y como tal permite exportar componentes a formatos tan heterogéneos como: JavaBeans, para su manipulación; HTML, para su presentación en un browser; PDF, para su presentación e impresión; etc. Estos componentes pueden ser utilizados por cualquier herramienta de courseware que adhiera a estas especificaciones, asegurando así la portabilidad del material didáctico entre ellas. Además, es posible agregarle funcionalidad a los courseware, a través de los componentes de actividad. Debido a esto, las instituciones que empleen courseware para apoyar la educación, podrán utilizar todas las herramientas de courseware que quieran, siempre que éstas adhieran a las especificaciones antes mencionadas.

El desarrollo de courseware basado en componentes además posee las mismas ventajas que el desarrollo de software basado en componentes [Bro98]. Se espera que a futuro, se genere un mercado de componentes educativos, destinado a comercializar este tipo de componentes [Ros99]. Esto significa que una institución educativa no necesariamente tendrá que desarrollar su material didáctico, sino que también podrá comprarlo a otras instituciones dedicadas a construirlo y comercializarlo. Esto indica que el tiempo que se invierta en explorar este camino, no será tiempo perdido, sino más bien una forma de adelantarse al futuro.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Fondo Nacional de Ciencia y Tecnología (FONDECyT) de Chile, como "Beca de Término de Tesis Doctoral".

Referencias

[Asf01] The Apache Software Foundation. *Cocoon: XML-based Web Publishing. (*2001). URL:xml.apache.org/cocoon2/index.html

[Boe99] Boehm, B. *Managing Software Productivity and Reuse*. IEEE Computer, 32, 9, (September 1999) pp. 111-113.

[Bro98] Brown, A. and Wallnau, K. The Current State of CBSE. IEEE Software. 15, 5, (September/October 1998) pp. 37-46.

[Cha96] Chapell, D. *Understanding ActiveX and OLE: A Guide for Developers & Managers*. Microsoft Press. (1996)

[Cou01] Blackboard, Inc. (May, 2001). "*CourseInfo*". URL: product.blackboard.net/courseinfo/

[Doh00] Doherty, D. and Leinecker, R. *JavaBeans Unleashed*. SAMS Publishing. (2000).

[Gol96] M. Goldberg, S.Salari and P.Swoboda (1996). *World Wide Web Course Tool: An Environment for Building WWW-Based Courses*, Computer Networks and ISDN Systems, 28. http://homebrew.cs.ubc.ca/webct/papers/p29/index.html

[Har99] Harasim, L. *A Framework for Online Learning: The Virtual-U*. IEEE Computer 32, 9, (September 1999). pp. 44-49.

[Har01] Harold, E. and Means, W. *XML in a Nutshel: A Desktop Quick Reference*. O'Reilly & Associates. (2001).

[Iee01] Learning Technology Standardization Committee of IEEE. URL: ltsc.ieee.org/wg12/doc.html (May, 2001).

[Ims01] IMS Global Learning Consortium, Inc. *IMS Content Packaging Specification*. (May, 2001), URL:www.imsproject.org/content/packaging/index.html

[Jdo01] JDOM. *JDOM Specification*. (May, 2001). URL: www.jdom.org/docs/apidocs/

[Lan01] B. Landon. *OnLine Educational Delivery Applications: A Web Tool for Comparative Analysis*". Standing Commitee ON Educational Technology; Centre for Curriculum, Transfer and Technology: Office of Learning Technologies. (May, 2001), URL: www.ctt.bc.ca/landonline/

[Lls01] IBM Lotus Corp. "*Learning Space*". (May, 2001). URL: www.lotus.com/home.nsf/tabs/learnspace

[Mcl00] McLaughlin, B., Loukides, M. *Java and XML*. O'Reilly & Associates (2000).

[Omg01] OMG (Object Management Group). CORBA 3.0. (April, 2001). URL: www.omg.org/technology/corba/corba3releaseinfo.htm

[Orf98] Orfali, R. and Harkey, D. *Client/Server Programming with Java and CORBA*. 2nd Edition. Wiley Computer Publishing. John Wiley & Sons, Inc. (1998).

[Pre00] Pressman, R. *Software Engineering: A Practitioner's Approach*. 5th Edition, McGraw Hill. (2000).

[Ros99] J. Roschelle, C. DiGiano, M. Koutlis, A. Repenning, N. Jackiw, D. Suthers. *Developing Educational Software Components*. IEEE Computer, 32, 9, (September 1999). pp.50-58.

[Too01] Asymetrix. Learning System Inc (May, 2001). "*ToolBook II*". URL: www.asymetrix.com/

[Top01] WBT Systems. *TopClass*. (May, 2001), URL: www.wbtsystems.com/

[Ude00] Uden, L. and Willis, N. (2000). Courseware Engineering Methodology. In Proceedings of On-line Learning Euro 2000. Cheltenham & Gloucester College of Higher Education, UK. (September 5-7, 2000). 6-9.

[Vir01] Simon Fraser University, British Columbia, Canada. *Virtual-U*. (May, 2001), URL: virtual-u.cs.sfu.ca/vuweb/

[Wcb01] Virginia Commonwealth University, U.S.A. *WCB (Web Course in a Box)*. (May, 2001), URL:www.madduck.com/wcbinfo/wcb.html

[Web01] University of British Columbia, Canada. *WebCT*. (May, 2001), URL: homebrew1.cs.ubc.ca/webct/