

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS



DESARROLLO DE UN EDITOR GRÁFICO PARA

OBJETOS DE APRENDIZAJE

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADO EN CIENCIAS COMPUTACIONALES

PRESENTA:

VÍCTOR HUGO SANDOVAL SORIANO

ENSENADA B.C. MÉXICO. MARZO DE 2005

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

FACULTAD DE CIENCIAS

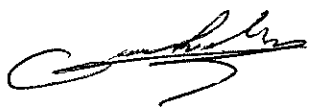
**“DESARROLLO DE UN EDITOR GRÁFICO PARA
OBJETOS DE APRENDIZAJE”**

TESIS PROFESIONAL

QUE PRESENTA

VÍCTOR HUGO SANDOVAL SORIANO

APROBADO POR:



M.I. GABRIEL LÓPEZ MORTEO.
DIRECTOR



DR. GILBERTO LÓPEZ MARISCAL
CODIRECTOR



M.C. LUIS ENRIQUE VIZCARRA CORRAL
PRIMER VOCAL

ENSENADA B.C. MÉXICO. MARZO DE 2005

RESUMEN de la Tesis de **Víctor Hugo Sandoval Soriano** presentada como requisito parcial para la obtención de la Licenciatura en **Ciencias Computacionales**. Ensenada, Baja California, México. Febrero de 2005.

Desarrollo de un Editor Gráfico para Objetos de Aprendizaje

Resumen aprobado por:



M.I. Gabriel López Morteo.

Director de Tesis.



Dr. Gilberto López Mariscal

Codirector de Tesis.

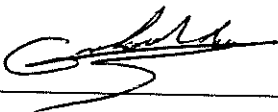
En CICESE hemos desarrollado objetos de aprendizaje para las matemáticas denominados Instructores Interactivos de Diversiones Matemáticas (IIDM), los cuales tienen una estructura interna basada en XML. Mediante el Sistema de Edición de Instructores Interactivos de Diversiones Matemáticas (SEIIDM), la edición de los IIDM se hace transparente para usuarios no experimentados en el uso de XML. El SEIIDM surge a partir de la necesidad de contar con una herramienta electrónica de autoría distribuida, especializada en la creación, modificación, recuperación y almacenamiento de los IIDM. Con el uso del SEIIDM el autor tiene la seguridad de crear un objeto de aprendizaje totalmente compatible con el modelo base de los IIDM, al cual podrá tener acceso de forma local o remota. Con el SEIIDM se pueden realizar modificaciones a elementos existentes sin que estos pierdan su estructura, ya que el editor valida la estructura del documento comparándolo contra el archivo de definición de datos (DTD).

Palabras Clave: objetos de aprendizaje, matemáticas recreativas, ambientes de aprendizaje

ABSTRACT of the Thesis presented by **Victor Hugo Sandoval Soriano**, as a partial requirement to obtain the **degree in Computer Sciences**. Ensenada, Baja California, México. February 2005.

Development of a Graphic Editor for Learning Objects.

Abstract approved by:



M.I. Gabriel López Morteo.

Thesis Director.



Dr. Gilberto López Mariscal

Thesis Codirector.

In CICESE we have developed learning objects for mathematics named Interactive Instructors of Recreational Mathematics (IIRM), which have an internal structure based on XML. By means of the Edition System of Interactive Instructors of Recreational Mathematics (ESIIRM), the edition of the IIRM becomes transparent for users without experience with XML. The ESIIRM arises from the necessity of an electronic distributed tool of authorship, specialized in the creation, modification and recovery of IIRM. With the use of the ESIIRM, authors create a learning object completely compatible with the model of the IIRM. In addition, it can be stored locally or remotely. The edition of legacy IIRM can be done without losing its structure, since the editor validates the structure of the document, comparing it with the document type definition file (DTD).

Keywords: Learning Objects, Recreational Mathematics, Learning Environment.

A Mis Padres

Agradecimientos

A mi Director de Tesis, M.I. Gabriel López Morteo, por su paciencia y apoyo.

A mi Codirector de Tesis, Dr. Gilberto López Mariscal, por su disponibilidad y apoyo.

A mi compañera de cubo, Adriana Miramontes, por su amistad incondicional y su apoyo durante el desarrollo de la tesis.

A cada uno de los miembros de mi comité de Tesis.

A la UABC y CICESE.

A mis maestros de la carrera, que sin su enseñanza no hubiera sido posible esta tesis.

A mis compañeros y amigos de carrera, mil gracias por su amistad y cariño.

Tabla de Contenido

Resumen	II
Abstract	III
Agradecimientos	V
Lista de Figuras	VIII
Capítulo I. Introducción.	1
I.1 Introducción.....	1
I.2 Antecedentes.....	3
I.3 Planteamiento del Problema.....	5
I.4 Objetivo.....	6
I.5 Organización de la Tesis.....	7
 Capítulo II. Metodología.	 8
II.1 Herramientas Utilizadas.....	8
II.2 UML (Unified Modeling Language).....	9
II.3 El Lenguaje de Modelado de Intercambio XML en el SEIIDM.....	10
II.4 Metodología de Desarrollo	11
 Capítulo III. Diseño del SEIIDM.	 13
III.1 Análisis de Casos de Uso de SEIIDM	13
III.2 Diagramas de Secuencia del SEIIDM	19
III.2.1. Diagrama de Secuencia de la Creación de un Nuevo Proyecto..	20
III.2.2. Diagrama de Secuencia de la Recuperación de un Proyecto.....	21
III.2.3. Diagrama de Secuencia del Almacenamiento del Proyecto.....	23
III.2.4. Diagrama de Secuencia de la Edición del Proyecto.....	26
III.2.5. Diagrama de Secuencia de la Selecciona Destino.....	32
III.3 Organización de las Clases que conforman al SEIIDM.....	33
III.4 Diagrama de Clases del SEIIDM.....	36
III.4.1. Diagramas.....	37

Tabla de Contenido (Continuación...)

Capítulo IV. Sistema de Edición de Instructores Interactivos de Diversiones Matemáticas (SEIIDM).	41
IV.1 Arquitectura.....	41
IV.2 Herramientas de Autoría.....	41
IV.3 Descripción del Proceso de Edición del SEIIDM.....	44
IV.3.1. Implementación del SEIIDM.....	44
IV.3.2. Funcionalidad del SEIIDM.....	50
IV.4 Edición del IIDM Colaborativa a través del SEIIDM.....	58
IV.4.1. Acceso a Servidor Remoto WebDAV.....	58
IV.4.2. El Protocolo WebDAV en el SEIIDM.....	59
IV.4.3. Implementación del Sistema.....	64
IV.4.4. Java Web Start.....	70
 Capítulo V. Conclusiones y Discusión	 76
 Referencias	 78
 Apéndice A. Herramienta XML Master (XMas)	 80
A.1 Requerimientos e Inicio de la aplicación XMas.	80
A.2 Panel de Control y Barra de Herramientas de XMas.	81
A.3 ¿Cómo se utiliza la aplicación XMas?.	84

Lista de Figuras

Figura	Descripción	Página
1	Modelo conceptual del IIDM representado como objeto de aprendizaje.....	4
2	Diagrama General de Casos de Uso del SEIIDM.....	13
3	Caso de Uso Creación de un Nuevo Proyecto.....	14
4	Caso de Uso Recuperación y Almacenamiento de un Proyecto.....	16
5	Caso de Uso Manipulación, Edición y Navegación del Proyecto....	18
6	Caso de Uso Destino Remoto.....	19
7	Diagrama de Secuencia para Crear un Nuevo Proyecto.....	20
8	Diagrama de Secuencia para Crear un Nuevo Proyecto (continuación....).....	21
9	Diagrama de Secuencia para Abrir un Proyecto Existente.....	22
10	Diagrama de Secuencia para Abrir un Proyecto Existente seleccionando destino.....	22
11	Diagrama de Secuencia para Abrir un Proyecto Existente y generación de errores.....	23
12	Diagrama de Secuencia para Guardar un Proyecto.....	24
13	Diagrama de Secuencia para Guardar un Proyecto (parte 2).....	25
14	Diagrama de Secuencia para Guardar un Proyecto (parte 3).....	25
15	Diagrama de Secuencia para la Edición del Proyecto Inicialización.....	27
16	Diagrama de Secuencia para la Edición del Proyecto Inicialización (continuación 1....).....	28
17	Diagrama de Secuencia para la Edición del Proyecto Inicialización (continuación 2....).....	29
18	Diagrama de Secuencia para la Edición del Proyecto Elaboración.....	30
19	Diagrama de Secuencia para la Edición del Proyecto Elaboración (continuación....).....	31
20	Diagrama de Secuencia Seleccionar destino cuando se abre un documento remoto o local.....	32
21	Diagrama de paquetes del SEIIDM.....	34
22	Diagrama de paquetes del SEIIDM (continuación....).....	34

Lista de Figuras (Continuación...)

Figura	Descripción	Página
23	Diagrama de paquetes del SEIIDM que contiene los accesos a archivos.....	35
24	Diagrama de paquetes del SEIIDM que contiene los analizadores XML.....	35
25	Diagrama de paquetes del SEIIDM que contiene las clases ResourceChoser.....	36
26	Diagrama de clases del paquete ResourceChoser.....	38
27	Diagrama de clases del editor SEIIDM primera parte.....	39
28	Diagrama de clases del editor SEIIDM segunda parte.....	40
29	Muestra al editor JEdit con contenido XML lo cual requiere conocimiento del lenguaje.....	43
30	Diagrama de paquetes del SEIIDM primera parte.....	48
31	Diagrama de paquetes del SEIIDM segunda parte.....	49
32	Panel de control del SEIIDM.....	50
33	Componentes del panel Archivo del SEIIDM.....	50
34	Ventana de Dialogo que advierte al usuario perdida de información.....	50
35	Ventana de diálogo que advierte pérdida de información cuando no ha sido grabado el proyecto y se desea salir.....	51
36	Componentes del panel Nodo del SEIIDM.....	52
37	Opciones del Panel Acerca del SEIIDM.....	52
38	Ventana que muestra la versión y los créditos del SEIIDM.....	52
39	Barra de Herramientas de acceso rápido con funcionalidades básicas del panel de control.....	53
40	Árbol de navegación mostrando el elemento padre al iniciar la generación del proyecto.....	53
41	Ventana de navegación mostrando opciones generales de inicio de edición.....	53
42	Árbol de navegación mostrando la estructura del documento generado y en donde se puede acceder a cualquier nivel del árbol...	54

Lista de Figuras (Continuación...)

Figura	Descripción	Página
43	Panel de Edición mostrando formato correspondiente a metadatos como lo refleja el árbol de navegación en la parte izquierda.....	55
44	DTD correspondiente reflejando el formato correspondiente a metadatos que muestra las Figuras 43 y 45.....	56
45	Panel de edición con contenido reflejado también en el árbol de navegación del lado izquierdo.....	56
46	Ventana de errores mostrando posición y tipo de error del archivo XML leído.....	57
47	Documento XML generado con el SEIDM.....	57
48	Problema de la pérdida de la actualización. Figura tomada de (Whitehead, 1998). Sinopsis del proceso.....	60
49	Diagrama de clases del paquete ResourceChoser.....	63
50	Navegación de Windows mediante WebDAV solicitando el contenido del directorio Pruebas.....	68
51	Ventana de navegación del SEIDM solicitando el contenido del directorio Pruebas a través del ResourceChoser.....	68
52	Ventana de Internet Explorer solicitando el contenido del directorio Pruebas a través del Web.....	69
53	Ventana mostrando el cargado de la aplicación XMas.....	80
54	Aplicación XMas en ejecución.....	81
55	Barra de Herramientas de la aplicación XMas.....	81
56	Menú Archivo del panel de control XMas.....	82
57	Menú Proyecto del panel de control de XMas.....	82
58	Menú Ayuda del panel de control de XMas.....	83
59	Ventana selectora del DTD y el Elemento Raíz.....	84
60	Ventana de Navegación para realizar la búsqueda y selección del archivo DTD.....	85
61	Ventana selectora del DTD y el Elemento Raíz mostrando la lista de elementos del DTD.....	85
62	Aplicación XMas mostrando las tres secciones que la dividen correspondientes a la edición en Non-Visual Beans.....	86

Lista de Figuras (Continuación...)

Figura	Descripción	Página
63	Aplicación XMas mostrando las tres secciones que la dividen correspondientes a la edición en Visual Beans.....	86
64	Aplicación XMas sección Default Editor presentando las opciones que tiene la elaboración de beans específicos.....	87
65	Aplicación XMas sección Default Editor presentando las opciones de personalización del Editor.....	87
66	Creación de Non-Visual Beans mostrando las opciones de la Área pestaña Bean Specifics en el Área de Edición y Código XML en de Código Fuente proveniente del DTD.....	88
67	Creación de Non-Visual Beans mostrando las opciones de la pestaña Path and Attributes.....	89
68	Opciones generales para la creación de un Visual bean del elemento metadatos.....	90
69	Editor Visual de componentes para los Visual Beans.....	90
70	Partes que conforman el editor visual de componentes.....	91
71	Barra de Herramientas del Editor Visual de componentes.....	91
72	Panel de Edición mostrando el componente Label agregado.....	92
73	Ventana de propiedades del componente Label como color, tamaño, etc.....	92
74	Ventana de selección para ligar el componente a una propiedad del elemento metadatos.....	93
75	Panel de edición mostrando el Label y la caja de texto ligada al título.....	93
76	Panel de edición mostrando todos los elementos que debe contener la creación de la interfaz grafica de los metadatos.....	94
77	Menú Project mostrando la selección de Setup Packages.....	95
78	Ventana para especificar los paquetes y el contenedor de los Beans No Visuales, el contenedor y los Iconos.....	95
79	Ventana para especificar los paquetes de los Beans Visuales y del Editor por Omisión.....	96

Lista de Figuras (Continuación...)

Figura	Descripción	Página
80	Ventana en la cual se definen tanto el directorio de salida del código fuente, como el nombre del archivo JAR a generar.....	96
81	Ventana de Generación de código fuente para los beans visuales y no visuales.....	97

Capítulo I. Introducción

I.1 Introducción.

La tecnología de la información aplicada a la educación de las matemáticas ha generado una gran variedad de software con el propósito de auxiliar a los estudiantes en el aprendizaje de los conceptos matemáticos. Muchos investigadores se han dado a la tarea de tratar de identificar los problemas pedagógicos y psicológicos que están relacionados con la formación de un pensamiento matemático en los alumnos. Bajo estas expectativas, se han realizado esfuerzos para lograr la creación de sistemas computacionales para el aprendizaje que contiene un fuerte componente lúdico, con el fin de capturar completamente la atención del individuo que esta interactuando con ellos y brindar una alternativa tecnológica para el aprendizaje entretenido, o bien para el entretenimiento educativo. (Papert S. 1996; Center for Innovation in Mathematics Teaching. 1997; Klawe M. 1998; Mendoza P. y Galvis A. 1998; Sánchez Santos J. 1998; Gorriz C. y Medina C. 2000)

En los años recientes las organizaciones han adoptado elementos didácticos electrónicos para equiparar a los entrenamientos tradicionales de salón. Estos elementos están compuestos por contenidos educativos, que por lo general emplean elementos multimedios, hojas de cálculo y programas. Ya que la tendencia es reutilizar este tipo de contenidos, se ha acuñado el concepto de **objetos de aprendizaje** para manipularlos como una entidad. Los objetos de aprendizaje (OA) se definen como una colección de recursos o entidades digitales o no digitales que pueden ser utilizadas para el aprendizaje, la educación o el entrenamiento (LTSC, 2002). En este sentido, en CICESE se desarrollaron objetos de aprendizaje para las matemáticas en forma de Instructores Interactivos de Diversiones Matemáticas (IIDM) (López Morteo G. López Mariscal G. and Ibarra Esquer J. 2001; López-Morteo G. López Mariscal G. e Ibarra Esquer J. 2002; López Morteo G. López Mariscal G. & Vizcarra Corral L. 2002; López Morteo G. and López Mariscal G. 2003; López Morteo G. y López Mariscal G. 2004), los cuales son el objeto de estudio de

esta tesis. Parte de la motivación para el desarrollo de OA basados en estándares, recae en la posibilidad de exportar e importar contenido educativo en ambientes de aprendizaje electrónicos; es decir, la reutilización de contenido educativo en contexto distinto a los que fueron creados.

Para formalizar los mecanismos de interacción e intercambio de OA, desde 1998 diferentes organizaciones están trabajando en la iniciativa ADL (*Advance Distributed Learning*, <http://www.adlnet.org/>), la cual es un esfuerzo conjunto del gobierno estadounidense, la industria y los académicos para establecer un nuevo entorno de aprendizaje distribuido, que permite la interacción de herramientas de aprendizaje y el compartir contenidos a escala global. La visión de ADL es proveer acceso a la educación de calidad y entrenamiento, adaptado a las necesidades individuales, entregadas de forma rentable en cualquier lugar y en cualquier momento. La iniciativa del ADL ha definido requisitos de alto nivel para los contenidos, tal como reutilización, accesibilidad, durabilidad e interacción. Promueve el uso del aprendizaje basado en tecnología y proporciona una base económica sana para la inversión.

Posteriormente se creó el proyecto SCORM (*The Sharable Content Object Referent Model*) <http://www.academiccolab.org/projects/scorm.html>, que aplica desarrollos tecnológicos actuales a un modelo de contenido específico produciendo recomendaciones para implementaciones consistentes. Está construido sobre el trabajo de *The Aviation Industry CBT Committee* (AICC), *Instructional Management System* (IMS), *IEEE Learning Technology Standards Committee* (LTSC), *Alliance of Remote Instructional Authoring and Distribution Networks for Europe* (ARIADNE) y otros para crear un modelo de referencia unificado de especificaciones técnicas correlacionadas y guías para encontrar requerimientos de alto nivel para el intercambio y reutilización de contenido educativo basado en el Web.

Tomando en cuenta la importancia que han tomado los objetos de aprendizaje y las necesidades que estos sean reutilizables en el contenido educativo, actualmente no se cuentan con herramientas simples para su creación y edición para los usuarios que no estén familiarizados con XML.

I.2 Antecedentes

Los Instructores Interactivos de Diversiones Matemáticas (IIDM) representan la conjunción de modelos pedagógicos y las matemáticas recreativas para fomentar la creación o ejercitamiento de ciertas habilidades relacionadas con el aprendizaje y uso de las matemáticas (Figura 1). Estos modelos se basan en la necesidad de equilibrar la calidad de la educación, donde se pone un énfasis no en la enseñanza, sino en el aprendizaje; es decir, en los progresos del estudiante más que en los conceptos impartidos por el maestro, centrándose en los procesos mentales del alumno y en su capacidad de avanzar hacia habilidades cognitivas cada vez más complejas, ya sea por si mismo o con ayuda de algún maestro o especialista.

El concepto de interactividad agregado al IIDM se logra al integrar técnicas extraídas de las Ciencias de la Computación para ampliar el alcance del entendimiento del concepto matemático a través de la interacción directa con componentes de software. Estos construyen el entorno de los elementos de la diversión matemática que permite visualizar el concepto matemático, o bien, que representan la solución del problema que se está planteando permitiéndole al individuo modificar sus partes, observar el resultado y obtener una retroalimentación inmediata a sus acciones. De esta manera, el individuo no solamente se ve expuesto a un problema matemático con fines recreativos, sino que interactúa con un sistema que lo guía durante la construcción de una solución, fomentando la generación de un pensamiento reflexivo, el cual es necesario para la generación del pensamiento matemático.

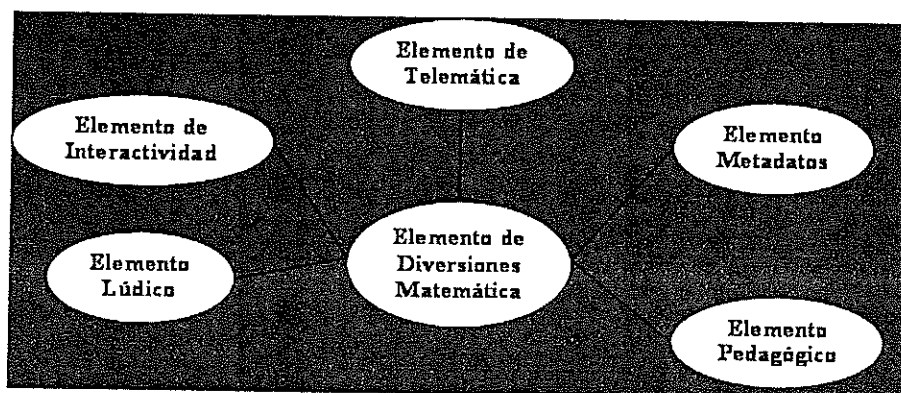


Figura 1: Modelo Conceptual del IIDM representado como objeto de aprendizaje.

Entonces, un IIDM es un componente de software educativo que se basa en conceptos matemáticos, con los cuales los usuarios pueden interactuar de tal forma que experimenten con ellos y desarrollen un pensamiento matemático individual conforme a sus habilidades.

A partir de las primeras definiciones de los IIDM, se desarrolló en CICESE un sistema de Instructores Interactivos de Diversiones Matemáticas, orientado a las matemáticas que se imparten a nivel secundaria llamado FIBONACCI (López Morteo G. López Mariscal G. & Vizcarra Corral L. 2002) en el cual se establecieron las bases para la formalización de los IIDM. De acuerdo a los criterios establecidos por el autor la incorporación de Metadatos facilita las búsquedas de OA.

Como es bien sabido, Internet ha impulsado el área educativa enormemente y día con día son más las instituciones y empresas que están utilizando a este medio para la enseñanza y capacitación a distancia. Esto, a su vez, ha tenido como resultado la creación de plataformas de administración de la enseñanza (*Learning Management System*) encargadas de almacenar, administrar y distribuir la información, no solamente académica, sino también administrativa, permitiendo que profesores, alumnos y administradores compartan un espacio de trabajo. En este sentido, en el Departamento de Ciencias de la Computación de CICESE, se desarrolló un Ambiente de Aprendizaje Electrónico llamado Los Supersabios (<http://azul.cicese.mx/supersabios/>), que es un espacio de trabajo colaborativo para la generación de conocimiento matemático cuyo contenido educativo son los IIDM (López Morteo G. López Mariscal G. and Ibarra Esquer J. 2001; López-Morteo

G. López Mariscal G. e Ibarra Esquer J. 2002; López Morteo G. López Mariscal G. & Vizcarra Corral L. 2002; López Morteo G. and López Mariscal G. 2003; López Morteo G. y López Mariscal G. 2004). Este ambiente fue diseñado para proveer soporte al educando mediante mecanismos de interacción, educando- máquina, educando- educando a través de un espacio interactivo de trabajo. Proponiendo al IIDM orientado a la solución de problemas en conjunción con tecnologías interactivas (java applets, vídeos, animaciones u hojas de cálculo).

Como un antecedente particular, está el editor de objetos de aprendizaje presentado por Myunghee Kang del departamento tecnológico educacional de la universidad de Ewha Womans en Seúl, Korea (Myunghee Kang., 2003). El cual está enfocado en la creación de objetos de aprendizaje electrónicos de cursos en línea. Esta herramienta cuenta con los mecanismos para la creación de objetos de aprendizajes no estructurados, que posteriormente son empaquetados y secuenciados bajo el estándar SCORM. De esta manera, se crea un conjunto de objetos de aprendizaje que pueden ser directamente incorporados a un ambiente de aprendizaje que soporte estos estándares, para que esté administre la serialización de la presentación del contenido, como parte de un curso en línea. Por ello, el enfoque principal de este editor radica en la estructura del curso más que en la estructura del contenido. A diferencia de este proyecto, se pretende en el desarrollo de este trabajo de tesis tener un control de la estructura durante la creación y edición de objetos de aprendizaje del tipo IIDM.

I.3 Planteamiento del Problema.

Como se mencionó anteriormente los Objetos de Aprendizaje se definen como una colección de recursos o entidades digitales o no digitales que pueden ser utilizadas para el aprendizaje, la educación o el entrenamiento (LTSC, 2002). De acuerdo a las normas SCORM y IEEE los objetos de aprendizaje incorporan al contenido educativo a los metadatos en XML. Estos metadatos y los IIDM cuentan con una estructura interna en XML, la cual, es establecida mediante archivos de definición de datos (DTD) y esquemas XML, con el fin de establecer un estándar en la elaboración de objetos de aprendizaje.

Actualmente no existen herramientas simples para la creación y edición de objetos de aprendizaje que estén diseñadas para un usuario final no capacitado en la edición de documentos válidos en XML. Tomando en cuenta herramientas como *Macromedia Flash*, *Authorware de Macromedia*, *JEdit*. Así entonces, los IIDM como objetos de aprendizaje, no cuentan con una herramienta de software de autoría especializada y fácil de usar que permitan la edición simplificada del código XML. De tal manera que sea posible crear su estructura interna y los metadatos asociados, que permita incorporar los IIDM en un ambiente de aprendizaje electrónico mediante procedimientos automatizados.

I.4 Objetivo

Por lo tanto en respuesta a esta problemática, en este trabajo de tesis se propone desarrollar un sistema de autoría de objetos de aprendizaje del tipo IIDM, basado en la estructura de metadatos y el modelo educativo que proveen los IIDM.

El objetivo general de este trabajo es desarrollar un sistema de autoría de objetos de aprendizaje del tipo Instructores Interactivos de Diversiones Matemáticas.

Además del objetivo general se establecieron los siguientes objetivos específicos:

- Crear y editar objetos de aprendizaje del tipo IIDM basados en XML.
- Almacenar los documentos XML localmente en el disco duro, o en un servidor WWW remoto a través del protocolo WebDAV.

Por otra parte el usuario podrá:

- Emplear un sistema basado en WWW para almacenar, buscar y recuperar los objetos de aprendizaje.

I.5 Organización de la Tesis

El resto de este trabajo esta estructurado de la siguiente manera. El capítulo II da una introducción a las metodologías de programación orientadas a objetos. Se muestran las herramientas utilizadas en este trabajo y la metodología de desarrollo que utilizamos.

En el capítulo III muestra Diagramas de casos de Uso, Diagramas de Secuencia, Diagramas de Paquetes y Diagramas de Clases.

En el capítulo IV se presenta el Sistema de Edición de Instructores Interactivos de Diversiones Matemáticas (SEIIDM), su arquitectura, herramientas de autoría, edición del IIDM colaborativa a través del SEIIDM, *Java Web Start* y Descripción del proceso de Edición del SEIIDM.

Y por último se presentan el Apéndice A donde se da una descripción detallada de la utilización de la herramienta *XML Master*.

Capítulo II. Metodología.

Una metodología de acuerdo con (MCDERMID, 1990) contiene fases, cada una esta formada por sub-fases que son responsables de orientar sobre las técnicas más apropiadas en cada paso del proyecto y también ayuda a planear, administrar, controlar y evaluar el estado del sistema de información. Siendo una técnica una manera de crear una actividad en particular en el proceso de desarrollo de sistemas de información y que pueden empezar a ser utilizadas por algunas metodologías. Cada técnica puede involucrar el uso de una o más herramientas que representan algunos dispositivos que son utilizados en el desarrollo de sistemas. Las herramientas generalmente son automatizadas, esto es, son recursos computacionales (software) que asisten durante el desarrollo.

Las técnicas de programación fueron discutidas por primera vez en los finales de la década de los 60's. La primera metodología de proyectos orientada a objetos, que todavía no tenía este nombre, fue descrita por Michael Jackson y Jean-Dominique Warnier en la década de los 70's (AVISON, 1997). Para Babiere (1997) uno de los factores críticos encontrados en el desarrollo de proyectos orientados a objetos es la carencia de definición de una línea metodológica que se seguirá en el proyecto.

II.1 Herramientas Utilizadas.

La construcción del SEIDM se realizó por etapas, donde una primera etapa estuvo compuesta por el diseño y prototipo de la aplicación. Para el diseño nos valimos del lenguaje de modelado UML y para su codificación del lenguaje de programación Java™ 2. En una segunda etapa agregamos soporte para la comunicación entre el editor y los servidores de datos, mejor conocidos como servidores *WebDAV*. Finalmente se agregaron a la aplicación, modelos de estandarización del tipo *Java Web Start* que añade características particulares al editor para el control de versiones, convirtiéndolo en una aplicación independiente al navegador.

Durante el desarrollo de este trabajo de tesis, se investigó sobre editores de XML existentes (Capítulo IV), sus características y las facilidades que brindan para la edición basada en DTD's (Documentos de Definición de Datos). Se empleó la herramienta *XML Master* de IBM que por su arquitectura facilitan la reutilización de código, ayudando en la construcción del soporte para el análisis de documentos XML, además para la creación de *beans* visuales y no visuales de acceso a los nodos de documentos XML.

II.2 UML (Unified Modeling Language).

Como indican su nombre, se trata de un lenguaje de modelado. UML no es un lenguaje de programación. UML no es una metodología o proceso que nos guíe en la construcción de sistemas software. UML simplemente marca unas pautas sobre cómo realizar una serie de diagramas gráficos que modelan la aplicación durante el proceso de desarrollo. La clave a la hora de desarrollar sistemas orientados a objetos está en una buena elección y comprensión del proceso o metodología de desarrollo a utilizar. Los procesos que han demostrado ser más versátiles y eficientes son aquellos que entienden el desarrollo de una forma incremental (sucesivas versiones cada vez más completas hasta llegar al sistema completo) y flexible (no establecer una serie de normas de obligado cumplimiento siempre, sino ajustar el proceso para cada caso concreto)(Perdita Stevens y Rob Pooley. 2002).

La herramienta de software que se empleó para trabajar con UML es POSEIDON (<http://www.gentleware.com/>), y con su ayuda se desarrollaron los diagramas de caso de uso, diagramas de secuencia, diagramas de paquetes y diagramas de clases, los cuales se presentan con detalle en el capítulo III.

II.3 El Lenguaje de Modelado de Intercambio XML en el SEIIDM

El SEIIDM basa su funcionamiento en la manipulación de documentos XML. XML es una tecnología bastante moderna, alrededor de la cual existe un gran interés (justificado por sus grandes posibilidades) pero también un gran desconocimiento. XML no es una herramienta de desarrollo de software como lo puedan ser *Visual Studio* o *Visualage*. Ni siquiera es un lenguaje de programación. Es un lenguaje de etiquetas al estilo HTML, pero mientras HTML mezcla los datos con la información necesaria para presentar esos datos (colores, tipos y tamaños, etc.) XML no contiene ninguna información de presentación. Está orientado a estructurar un documento en donde las etiquetas tienen una representación semántica que refleja el dominio del problema para el que esta diseñado.

Existen tres usos generales, que están siendo explotados por aplicaciones del mundo real como son:

- 1) Construcción de sitios Web creando una capa intermedia de datos, basada en XML, a partir de la cual se pueden generar salidas en HTML, WML, PDF, etc.
- 2) Intercambio de datos entre empresas y aplicaciones a través de Internet, independientemente de las plataformas de hardware, sistemas operativos y lenguajes de programación implicados. Un caso específico de este uso lo constituyen los famosos servicios Web (*Web Services*) para que el intercambio de datos entre aplicaciones Web.
- 3) La configuración de aplicaciones mediante archivos en formato XML de múltiples servidores y software de uso variado.

El lenguaje XML en sí mismo es un lenguaje muy sencillo que se puede aprender a utilizar en unas pocas horas. En cambio si consideramos todo el abanico de tecnologías basadas en XML (lenguajes de restricción, lenguajes de transformación, API para uso de XML desde un entorno de programación, etc.) llegamos a una situación verdaderamente compleja en la cual es imprescindible contar con un buen mapa para no perderse.

II.4 Metodología de Desarrollo.

El SEIIDM sigue el modelo iterativo (Fower Martin & Scout Kendall. 1999; Jacobson. Ivar et.al. 1999; Presuman Roger. 1999) de desarrollo de software empleando UML como el lenguaje de modelado. Se seleccionó este modelo porque un proyecto que utiliza el desarrollo iterativo presenta un ciclo de vida que consiste de varias iteraciones. En cada iteración un conjunto de productos es generado. Cada iteración incorpora actividades, requisitos, análisis, implementación, pruebas e instalación a medida que el proyecto va evolucionando. En otras palabras, cada iteración es como un mini-proyecto en el modelo cascada en donde ocurren (en mayor o menor grado, dependiendo de la fase), todas las fases típicas de un proyecto.

La evaluación final de cada iteración consiste en la verificación del estado del proyecto desde la perspectiva del producto y del cronograma, además que la evaluación final, también analiza lo que deberá ser cambiado en la organización y en el proceso en sí a fin de que la próxima ejecución de un proyecto sea aún mejor. Las iteraciones son planeadas en número, duración y objetivo. Las tareas y responsabilidades de los participantes son definidas y las medidas de progreso son capturadas en documentos o memorias.

Un desarrollo iterativo es, en general, superior al desarrollo lineal o de cascada. Ya que de acuerdo con (Fower Martin & Scout Kendall. 1999; Jacobson. Ivar et.al. 1999; Presuman Roger. 1999):

- Soporta bien los cambios de requisitos.- Requisitos “mutantes” siempre causan estragos en proyectos, causando atrasos, cronogramas extrapolados, clientes insatisfechos y desarrolladores frustrados, sin mencionar el descrédito de la empresa generado junto al cliente.
- Elementos son integrados progresivamente.- No existe “*big-bang*”. El desarrollo iterativo es casi una integración continua. Lo que solía ser un largo,

incierto y problemático período de tiempo, tomando hasta 40% del esfuerzo total del proyecto, ahora puede ser dividido en varias integraciones menores.

- Los riesgos son eliminados más temprano.- Los riesgos son identificados y encarados a medida que van siendo pasadas las primeras iteraciones todos los flujos de trabajo principales del proceso son ejercitados: uso de herramientas, empleo de productos de terceros, desarrollo de las habilidades del equipo, e incluso algunos riesgos percibidos pueden probar no ser riesgos legítimos.
- Permite a la organización aprender y mejorar.- Los miembros de un equipo en un desarrollo iterativo pueden aprender a lo largo del camino probando los productos más tempranamente, escribiendo manuales, identificando necesidades de entrenamiento y soporte adicional. Como comparación, en un desarrollo no iterativo esas mismas personas permanecerían aguardando para comenzar el trabajo.
- Facilidad de Rehúso.- Es más fácil identificar elementos potenciales de código de ser reutilizados a medida que van siendo parcialmente proyectados o implementados, estabilizando código común en las iteraciones subsecuentes.
- Resulta en un producto más robusto.- Por el hecho de que los errores son corregidos en diversas iteraciones. En la fase prueba los errores son detectados antes y los obstáculos de desempeño son removidos en un tiempo en el cual ellos pueden ser enfrentados y solucionados.

Capítulo III. Diseño de SEIIDM.

III.1 Análisis de Casos de Uso del SEIIDM.

El diagrama de casos de uso representa la forma en como un cliente (actor) opera con el sistema en desarrollo, además de la forma, tipo y orden en como los elementos interactúan (operaciones o casos de uso). Un diagrama de casos de uso consta de tres elementos tales como Actores, Casos de Uso y Relaciones de Uso. Un Actor es un rol que un usuario juega con respecto al sistema, donde este no necesariamente representa a una persona en particular, sino más bien la labor que realiza frente al sistema. Un Caso de Uso es una operación/tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso. Y por ultimo las relaciones que pueden ser Asociación siendo esta la relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso), Dependencia o Instanciación donde es una forma muy particular de relación entre clases, en la cual una depende de otra y Generalización siendo este uno de los mas utilizados, cumple una doble función dependiendo de su estereotipo, que puede ser de Uso (`<<uses>>`) o de Herencia (`<<extends>>`).

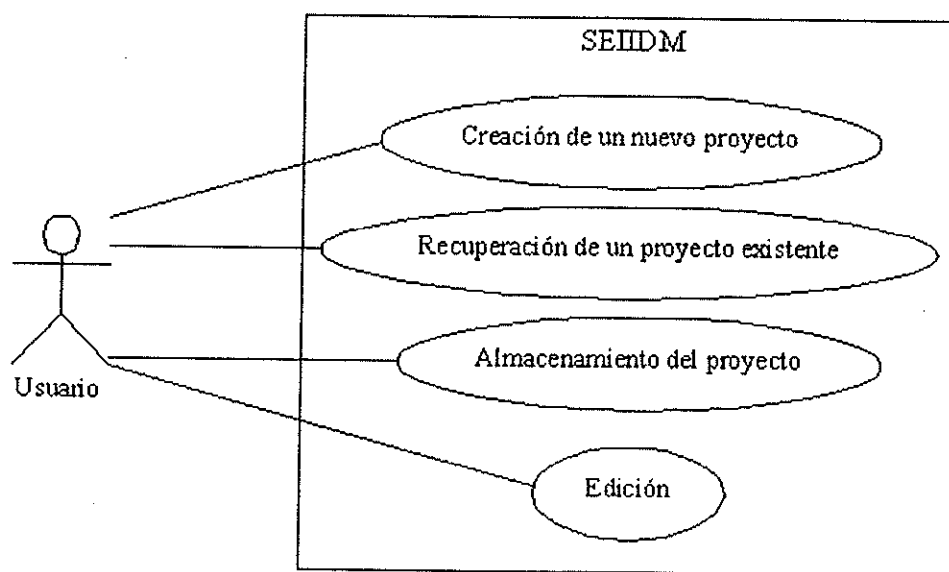


Figura 2: Diagrama General de Casos de Uso del SEIIDM.

Caso de Uso: Creación de un nuevo proyecto (Figura 2 y Figura 3)

Descripción: Permite generar un nuevo proyecto partiendo desde cero. Durante su funcionamiento se involucran dos casos de uso como Presentación de interfaces de edición y Verificación de datos iniciales de la aplicación.

Caso de Uso: Presentación de interfaces de edición. (Figura 3)

Descripción: Se encarga de mostrar las diferentes pantallas que son necesarias para comenzar con la edición del proyecto.

Caso de Uso: Selección de Elemento Raíz. (Figura 3)

Descripción: Este caso de uso tiene la función de obtener del documento DTD el elemento raíz el cual permitirá establecer un punto de inicio durante la edición del proyecto.

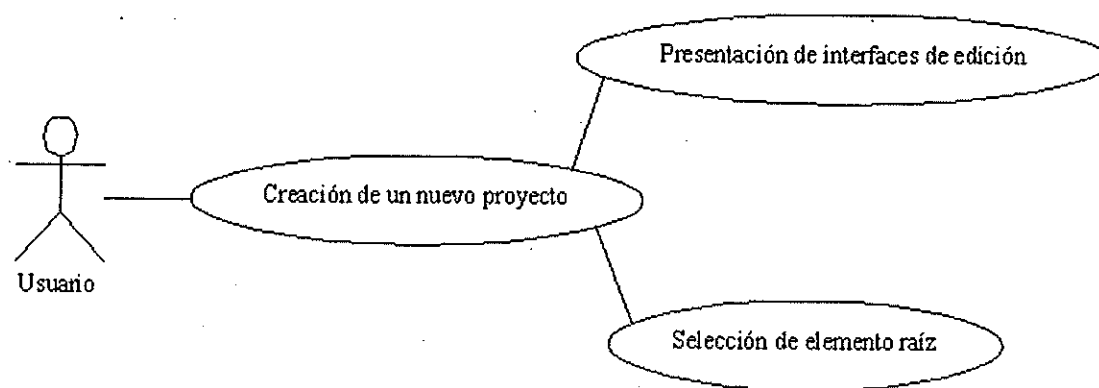


Figura 3: Caso de Uso Creación de un Nuevo Proyecto.

Caso de Uso: Recuperación de un proyecto existente. (Figura 2 y Figura 4)

Descripción: Este caso de uso permite abrir un proyecto que haya sido creado anteriormente y que se encuentre almacenado local o remotamente; para posteriormente presentar las pantallas de edición y así poder modificar su contenido y/o agregar nuevos datos. Para desempeñar su función necesita de dos casos de uso: Leer archivo XML local y Leer archivo XML remoto.

Caso de Uso: Leer archivo XML local. (Figura 4)

Descripción: Este caso de uso se encarga de realizar la búsqueda local del archivo que contiene el proyecto para finalmente obtenerlo.

Caso de Uso: Leer archivo XML remoto. (Figura 4)

Descripción: Este caso de uso se encarga de realizar la búsqueda remota del archivo que contiene el proyecto utilizando el caso de uso Destino Remoto (Figura 6) para finalmente obtener el archivo.

Caso de Uso: Almacenamiento del Proyecto. (Figura 2 y Figura 4)

Descripción: este caso de uso permite guardar el proyecto actual un archivo XML el cual puede ser almacenado remota o localmente, para generar el archivo requiere del caso de uso Creación de la cadena XML. También involucra al caso de uso Guarda Archivo XML.

Caso de Uso: Crear Archivo XML remoto. (Figura 4)

Descripción: Aquí es donde se obtienen las diferentes etiquetas e información que se genera en el proyecto y las agrupa en una sola cadena compatible con el lenguaje XML y correspondiente al estándar del IIDM. Emplea al caso de uso Destino Remoto (Figura 6) para depositar la cadena generada en el archivo remoto.

Caso de Uso: Crear Archivo XML local. (Figura 4)

Descripción: Aquí es donde se obtienen las diferentes etiquetas e información que se genera en el proyecto y las agrupa en una sola cadena compatible con el lenguaje XML y correspondiente al estándar del IIDM.

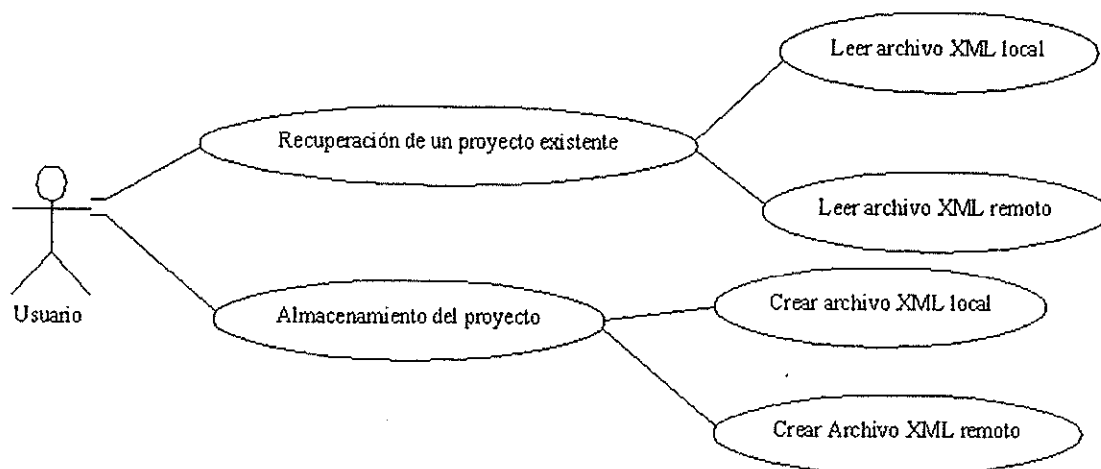


Figura 4: Caso de Uso Recuperación y Almacenamiento de un Proyecto.

Caso de Uso: Manipulación de elementos. (Figura 5)

Descripción: Este caso de uso se encarga de presentar los elementos que están disponibles dependiendo del nivel en que se encuentre un nodo, permitiendo así solo generar elementos validos. Este caso de uso depende de dos casos mas que son Agregar un Elemento e Insertar un Nodo Antes.

Caso de Uso: Agregar un elemento. (Figura 5)

Descripción: Este caso de uso se encarga de insertar un elemento al proyecto y a su vez cambiar la ventana de navegación al estado actual, presentando los elementos validos permitidos.

Caso de Uso: Insertar un Nodo Antes. (Figura 5)

Descripción: Este caso de uso permite ver un nivel atrás del nodo actual presentando los elementos validos que pueden ser agregados al proyecto sin que la ventana de navegación cambie de estado hasta que se pase al modo de insertar.

Caso de Uso: Edición de Datos. (Figura 5)

Descripción: Este caso de uso se encarga de presentar todo el ambiente de edición de

contenido de los diferentes nodos generados con la ayuda de la ventana de navegación y al árbol de navegación mostrando así sus formas correspondientes dirigiendo el trabajo a uno de tres casos de uso que son Capturar datos, Modificación de datos y Borrado de datos.

Caso de Uso: Capturar datos. (Figura 5)

Descripción: Este caso de uso permite habilitar la escritura de los campos pertenecientes a un elemento en edición.

Caso de Uso: Modificación de datos. (Figura 5)

Descripción: Este caso de uso al igual que con Capturar datos, permite la escritura y modificación de la misma en un elemento leído o creado por el usuario.

Caso de Uso: Borrado de datos. (Figura 5)

Descripción: Este caso de uso permite eliminar contenido que fue agregado o modificado mediante los casos de uso Capturar datos y Modificación de datos.

Caso de Uso: Navegación por el documento. (Figura 5)

Descripción: Este caso de uso tiene como función mostrar el árbol de edición generado permitiendo ver el estado actual del proyecto además que se basa en tres casos de uso que son Moverse entre niveles, Eliminar Nodos y Seleccionar Elemento.

Caso de Uso: Moverse entre niveles. (Figura 5)

Descripción: Este caso de uso se encarga de generar los mecanismos de navegación dentro del árbol, permitiendo así poder acceder a cualquier nivel que contenga el árbol.

Caso de Uso: Eliminar Nodos. (Figura 5)

Descripción: Este caso de uso permite eliminar un nodo del árbol, ya sea si es hoja o si contiene una descendencia.

Caso de Uso: Seleccionar Elementos. (Figura 5)

Descripción: Este caso de uso permite elegir un elemento del árbol cuyo contenido será visualizado en el panel de edición.

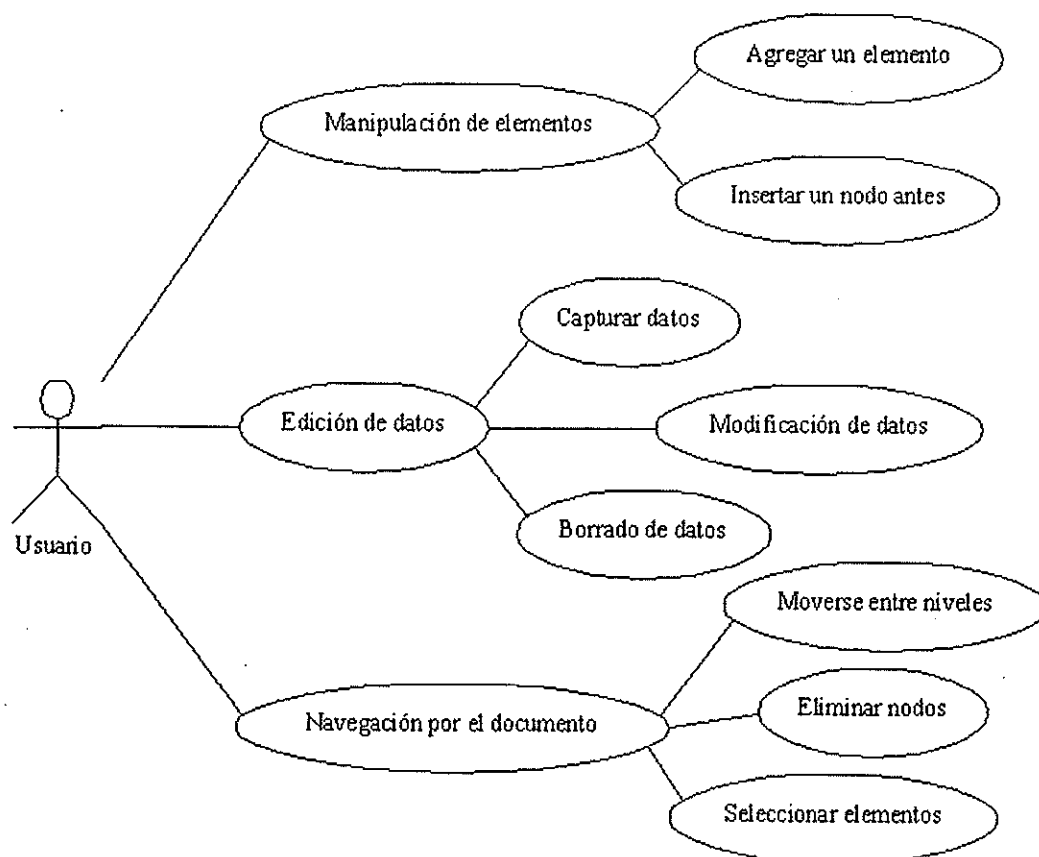


Figura 5: Caso de Uso Manipulación, Edición y Navegación del Proyecto

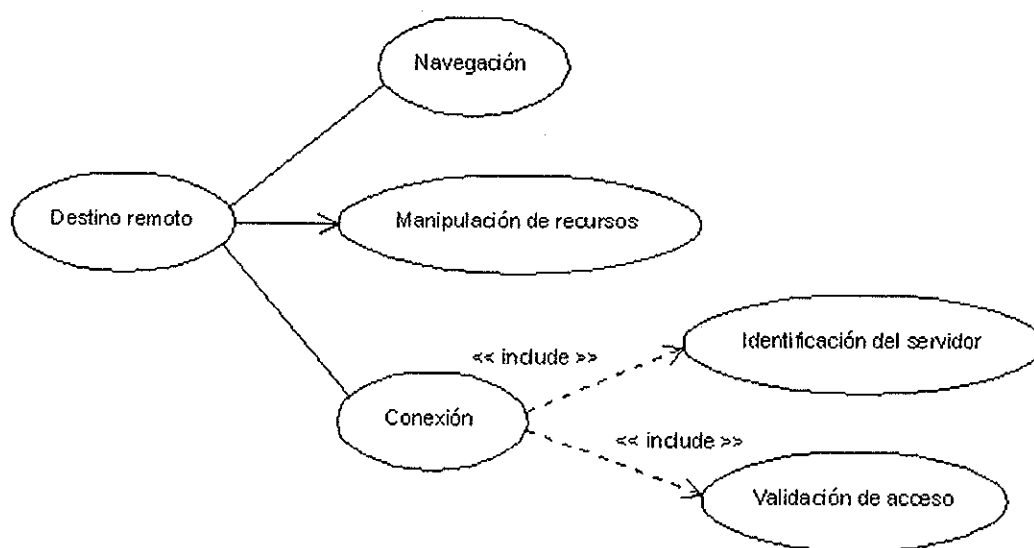


Figura 6: Caso de Uso Destino remoto

III.2 Diagramas de Secuencia del SEIIDM.

El diagrama de secuencia es el núcleo de nuestro modelo dinámico, y muestra todos los cursos alternos que pueden tomar los casos de uso. Los diagramas de secuencias se componen de 4 elementos que son: el curso de acción, los objetos, los mensajes y los métodos (operaciones).

Los diagramas de secuencia, formalmente diagramas de traza de eventos o de interacción de objetos, se utilizan con frecuencia para validar los casos de uso. Documentan el diseño desde punto de vista de los casos de uso. Observando que mensajes se envían a los objetos, componentes o casos de uso y viendo a groso modo cuanto consume el método invocado; los diagramas de secuencia nos ayudan a comprender los cuellos de botella potenciales para así poder eliminarlos.

El diagrama de secuencia de un sistema muestra gráficamente los eventos que originan los actores. La creación de los diagramas de secuencia forma parte de la investigación para conocer el sistema, por lo que es parte del análisis del mismo. La creación de los diagramas de secuencia depende de la formulación de los casos de uso. Los

casos de uso indican cómo los actores interactúan con el sistema. Durante la operación del sistema, los actores generan eventos, solicitando alguna operación a cambio.

III.2.1. Diagrama de Secuencia de la Creación de un Nuevo proyecto.

En la Figura 7 y 8 se muestra el diagrama de secuencia para iniciar la creación de un nuevo proyecto. Esto comienza cuando el usuario presiona el botón de crear un proyecto nuevo, ahí el sistema escucha la acción y se pasa a la verificación del archivo donde se hace una comparación entre si hay un archivo en edición y este fue guardado o si no es así. Si es la primera opción se manda a generar una ventana de dialogo la cual es presentada al usuario indicándole la situación actual y que de no guardar su proyecto este se perderá. Si no es así el flujo se pasa hasta la inicialización (Figura 8), donde se establece el elemento raíz el cual es almacenado en el contenedor del proyecto, se le asigna un nombre al proyecto, y finalmente se habilita la edición pasando el control a la Edición del Proyecto.

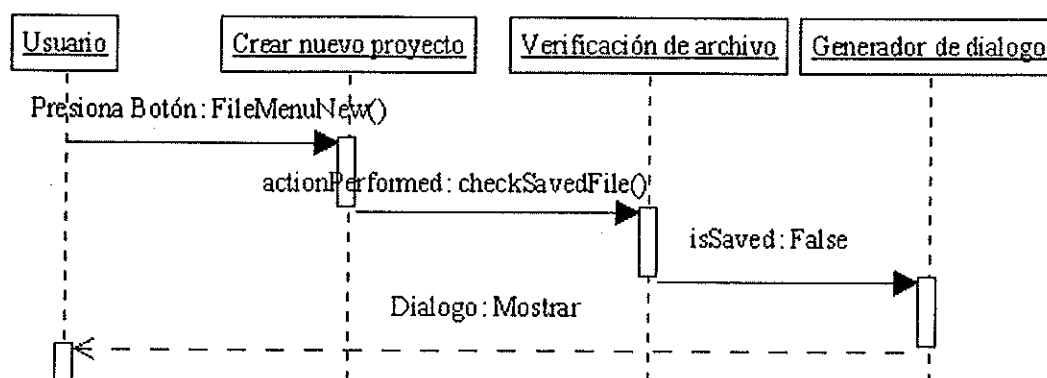
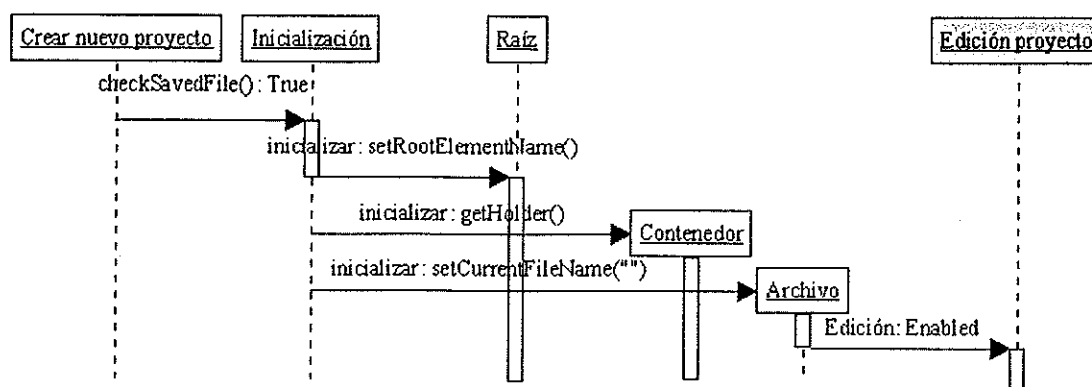


Figura 7: Diagrama de Secuencia para Crear un Nuevo Proyecto.



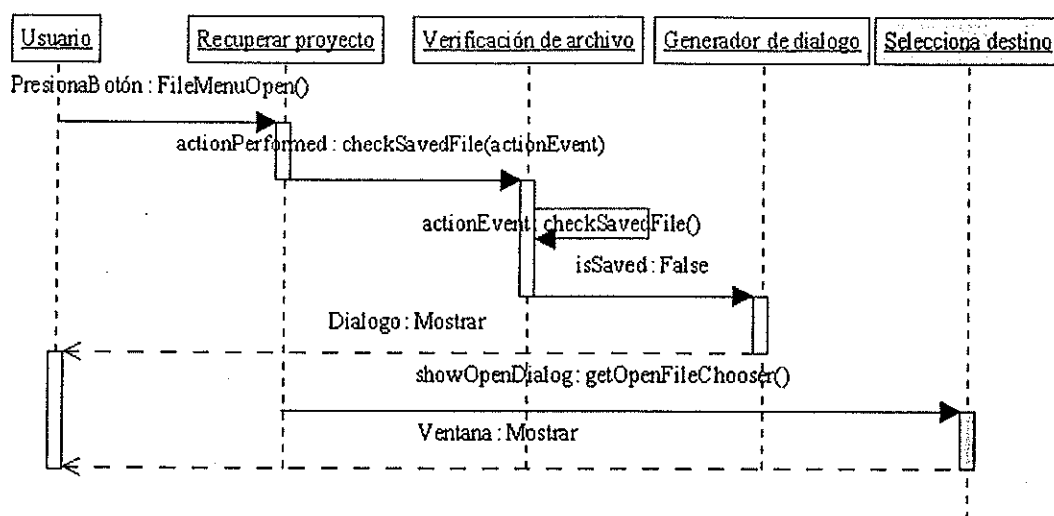
Objeto con su propio diagrama de secuencia

Figura 8: Diagrama de Secuencia para Crear un Nuevo Proyecto (continuación...).

III.2.2. Diagrama de Secuencia de la Recuperación de un Proyecto.

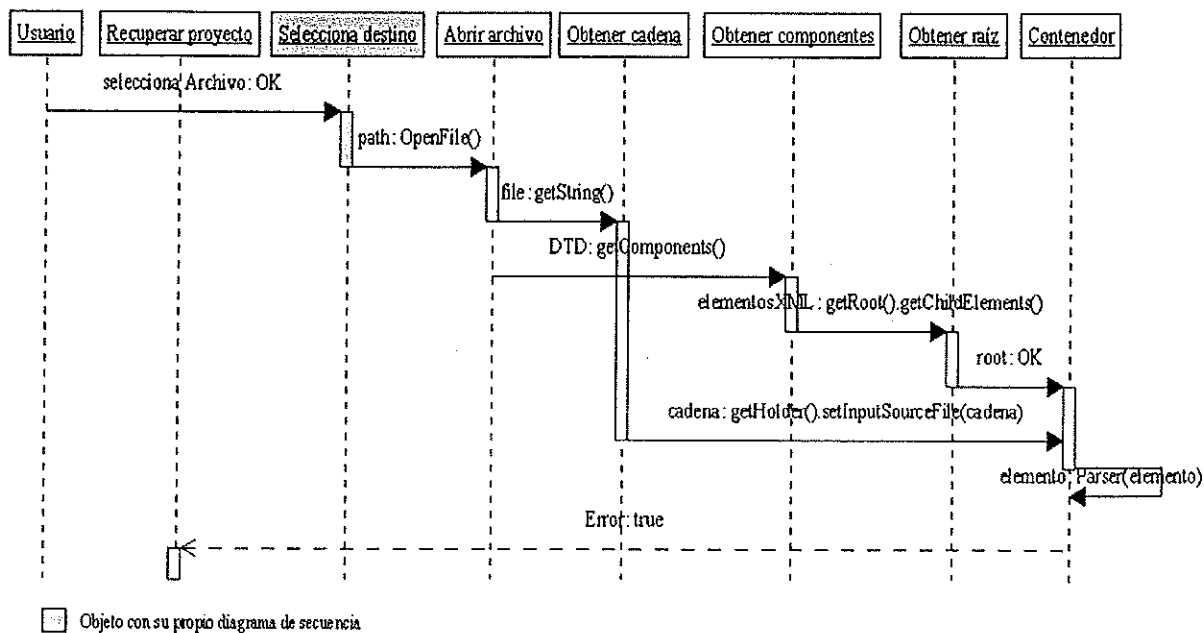
En las Figuras 9, 10 y 11 se muestran el diagrama de secuencia para la recuperación de un proyecto existente. Esto comienza cuando el usuario presiona el botón de Abrir un Proyecto, el sistema escucha la acción y se pasa a la verificación del archivo donde se hace una comparación entre si hay un archivo en edición y este fue guardado o si no es así. Si es la primera opción, se manda a generar una ventana de dialogo la cual es presentada al usuario indicándole la situación actual y que de no guardar su proyecto este se perderá; si no es así, el flujo se pasa a Seleccionar Destino donde se manda llamar una ventana de navegación y esta es mostrada al usuario; ahora el usuario selecciona el destino y el archivo pasando el flujo a Abrir Archivo (Figura 10), donde el contenido es extraído en una cadena y a su vez se mandan a extraer los componentes del DTD. De ahí se obtiene la etiqueta raíz; la cual se almacena en el contenedor del proyecto y a su vez se pasa la cadena del archivo al contenedor donde este, antes de almacenarla, realiza un análisis de cada uno de los elementos. Si uno de ellos no coincide con la estructura del DTD, se activa una bandera de error la cual manda a la generación de la ventana de errores donde se presentan al usuario todos los errores existentes en el archivo que intento leer (Figura 11), explicando

la posición del error y una descripción obligando al usuario a volver intentar la acción. De otra forma si no se presenta un error en el archivo, el contenedor pasa el control a Abrir Archivo indicándole que todo fue un éxito y que los elementos fueron almacenados en el contenedor del proyecto habilitando así la edición del proyecto.



Objeto con su propio diagrama de secuencia

Figura 9: Diagrama de Secuencia para Abrir un Proyecto Existente



Objeto con su propio diagrama de secuencia

Figura 10: Diagrama de Secuencia para Abrir un Proyecto Existente seleccionando destino.

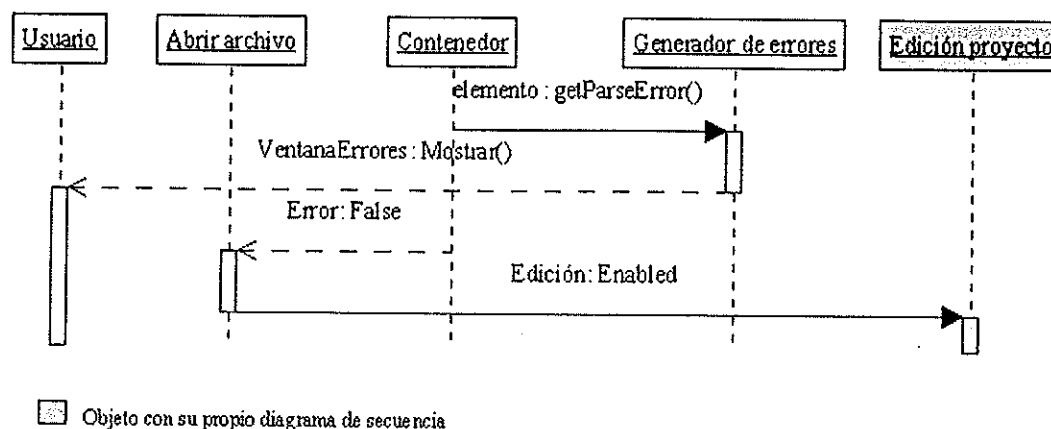


Figura 11: Diagrama de Secuencia para Abrir un Proyecto Existente y generación de errores.

III.2.3. Diagrama de Secuencia del Almacenamiento del Proyecto.

En las Figuras 12, 13 y 14 se muestra el diagrama de secuencia para el almacenamiento de un proyecto en edición. Esto comienza cuando el usuario presiona el botón de Guardar Proyecto; el sistema escucha la acción y se pasa a la verificación del archivo donde se hace una comparación si el archivo ya tiene asignado un nombre o no. Si no tiene un nombre, se manda llamar la ventana de navegación en donde se seleccionara el destino de almacenamiento del documento (Figura 12). La ventana es mostrada al usuario para que indique donde quiere guardar el archivo y con que nombre; una vez realizada la operación por el usuario, se regresa el control a Seleccionar Destino que a su vez manda la trayectoria y el nombre del archivo a generar, esta es convertida en cadena y se pasa a la elaboración del contenido del archivo correspondiente a los elementos almacenados en el contenedor; se le quitan los espacios y si el nombre no existe en la trayectoria este pasa el control la edición del proyecto indicándole que fue un éxito el almacenamiento del proyecto. Si ya hubiera un documento con el nombre seleccionado, se pasa el control a Seleccionar Destino en donde éste a su vez, mostrara un mensaje de archivo existente al usuario indicándole si lo quiere sobrescribir o cambiar de nombre (Figura 13). Después de la acción del usuario, se pasa el control a Seleccionar Destino para realizar la misma secuencia antes descrita. Por otra parte si al presionar el usuario el botón de guardar

proyecto este ya tenia un nombre asignado, no se muestra la ventana de navegación al usuario omitiendo algunos pasos, sino por lo contrario se realizan las operaciones de modificación del archivo XML final el cual será almacenado en la trayectoria predefinida anteriormente terminado con el paso de control a Edición del proyecto (Figura 14).

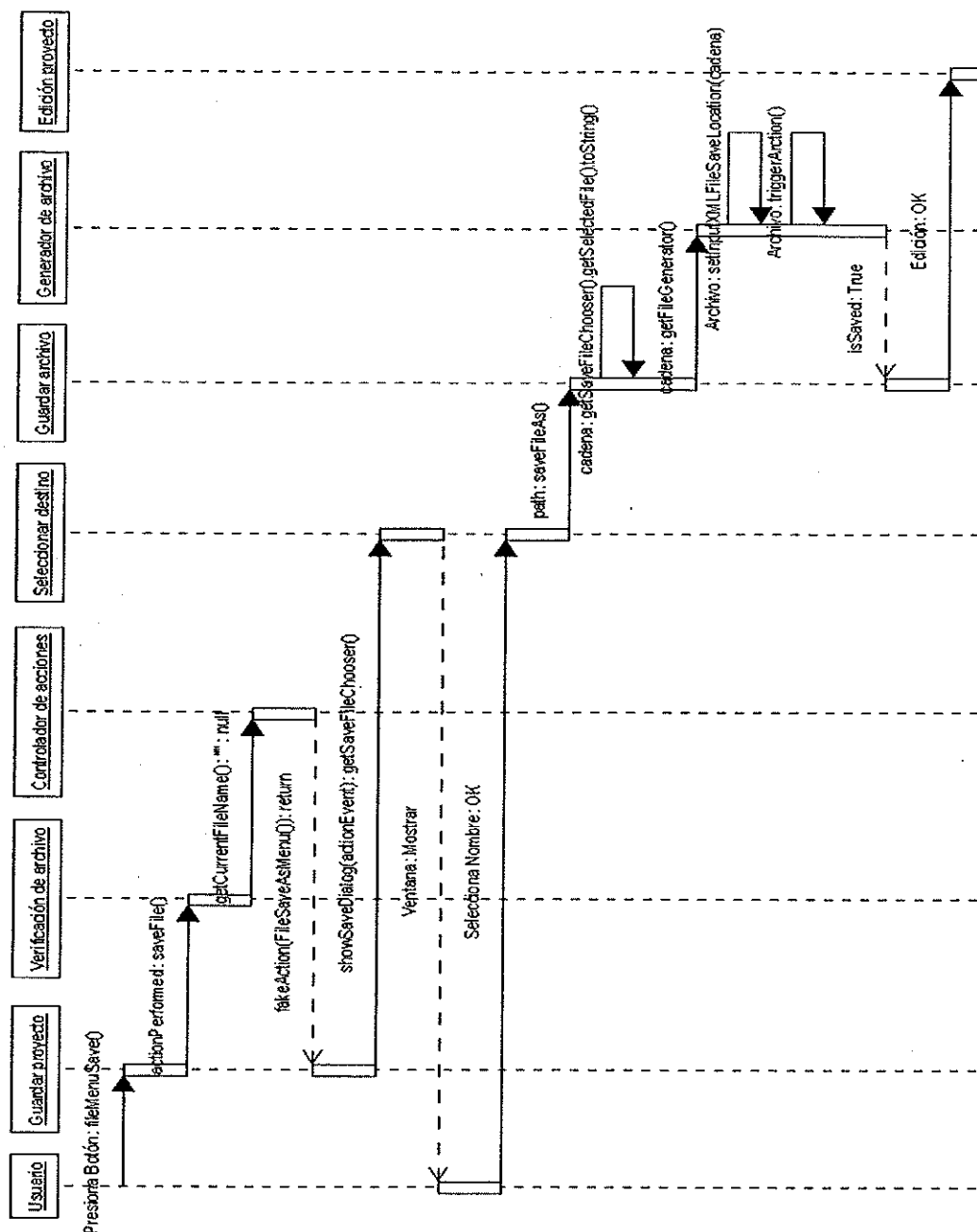


Figura 12: Diagrama de Secuencia para Guardar un Proyecto.

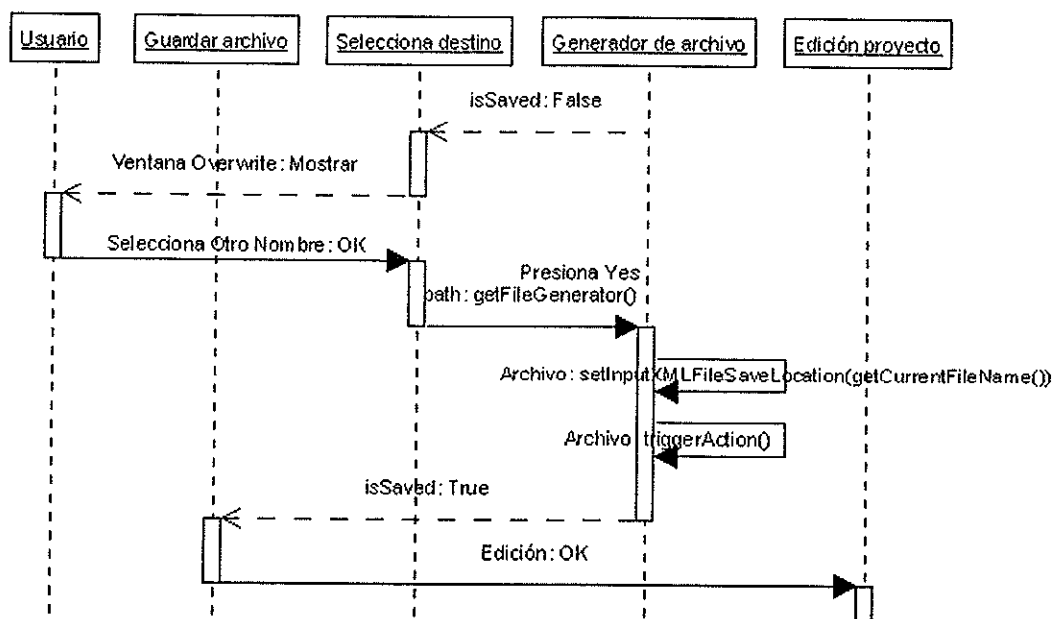


Figura 13: Diagrama de Secuencia para guardar un proyecto (parte 2).

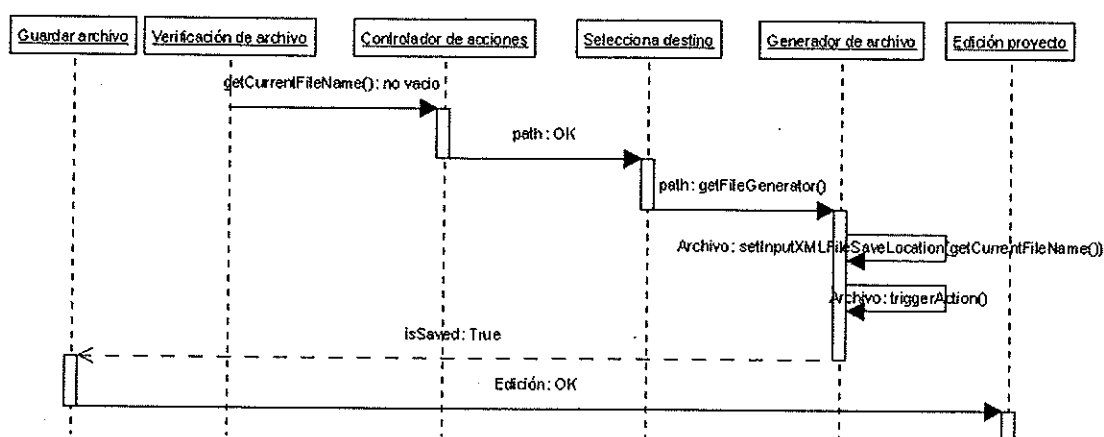


Figura 14: Diagrama de Secuencia para guardar un proyecto (parte 3).

III.2.4. Diagrama de Secuencia de la Edición del Proyecto.

En las Figuras 15, 16 y 17 se muestra el diagrama de secuencia para la edición del proyecto, correspondiente a la inicialización del Editor. Esto comienza cuando el usuario ejecuta la aplicación, pasando el control a la inicialización del panel, este a su vez asigna operaciones por omisión al Editor Panel como son nombre, formato de la barra de menú, panel de contenido, inicialización de la navegación del documento. Después inicia la creación del contenedor donde en primera instancia se deposita el elemento raíz del documento DTD; una vez analizado el DTD se extrae la raíz y se almacena en el contenedor (Figura 15 y 16). Se inicializa también la ventana de navegación y el árbol de navegación asociando los nodos con los iconos que serán mostrados en el árbol de navegación y la ventana de selección de elementos (Figura 17), y finalmente se pasa el control a Editor panel indicando que esta listo pasando finalmente el control al usuario.

En la Figura 18, 19 se presenta el diagrama de secuencia que muestra la edición del proyecto relacionado a las acciones que puede realizar el usuario después de que la aplicación ha sido inicializada y el control esta en poder del usuario. Las opciones que tiene el usuario son Crear nuevo proyecto (Figura 9,10 y 11), Recuperar proyecto (Figura 12, 13 y 14), Guardar proyecto (Figura 15, 16 y 17).

En el caso de crear nuevo proyecto, este manda obtener el documento de entrada perteneciente al contenedor; este inicializa la ventana de navegación y la lista de elementos correspondientes a los elementos encontrados en el contenedor, depositando cada elemento del contenedor en el navegador de documentos por medio de un árbol que en primera instancia contiene solo el elemento raíz, y se prepara el panel de edición para escuchar los eventos que se lleven a cabo con el uso de la ventana de navegación y el árbol, una vez que estos sean mostrados al usuario, y habilitados para que los utilice. Después que el control es pasado al usuario este realiza las operaciones de edición comenzando por seleccionar un nodo de la ventana de selección de elementos esta obtiene sus propiedades mandado llamar al nodo correspondiente, almacenándolo en el contenedor y actualizando los datos mostrados tanto en la ventana de selección de elementos como en el panel de edición conformado por árbol de navegación y el Editor Panel; pasando finalmente el control al

Usuario. El usuario puede repetir el proceso cuantas veces sea necesario una vez que obtiene el control. El puede moverse libremente por el árbol de navegación y dependiendo el nivel le serán mostradas las opciones disponibles, puede eliminar elementos y/o contenido, así como agregar nuevos elementos con ayuda del la ventana de selección de elementos.

En el caso que se recupere un proyecto existente (Figura 19), se siguen los mismos pasos en la inicialización. La única variante será que el contenedor iniciara con más elementos; adecuando la ventana de selección de elementos como el árbol de navegación, al estado actual del proyecto recuperado. El proceso de edición es el mismo. Para el caso de Almacenamiento del Proyecto, el método Guardar Proyecto obtiene los datos del contenedor y este notifica al panel de edición que fue guardado y este finalmente notifica al usuario el éxito de la operación (Figura 19).

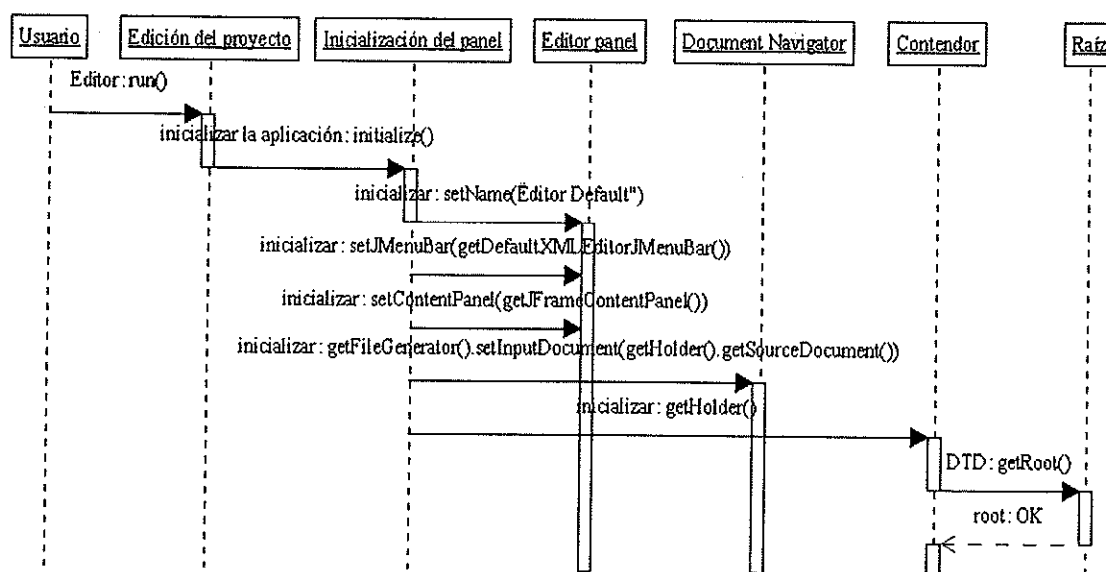


Figura 15: Diagrama de Secuencia para la Edición del Proyecto Inicialización.

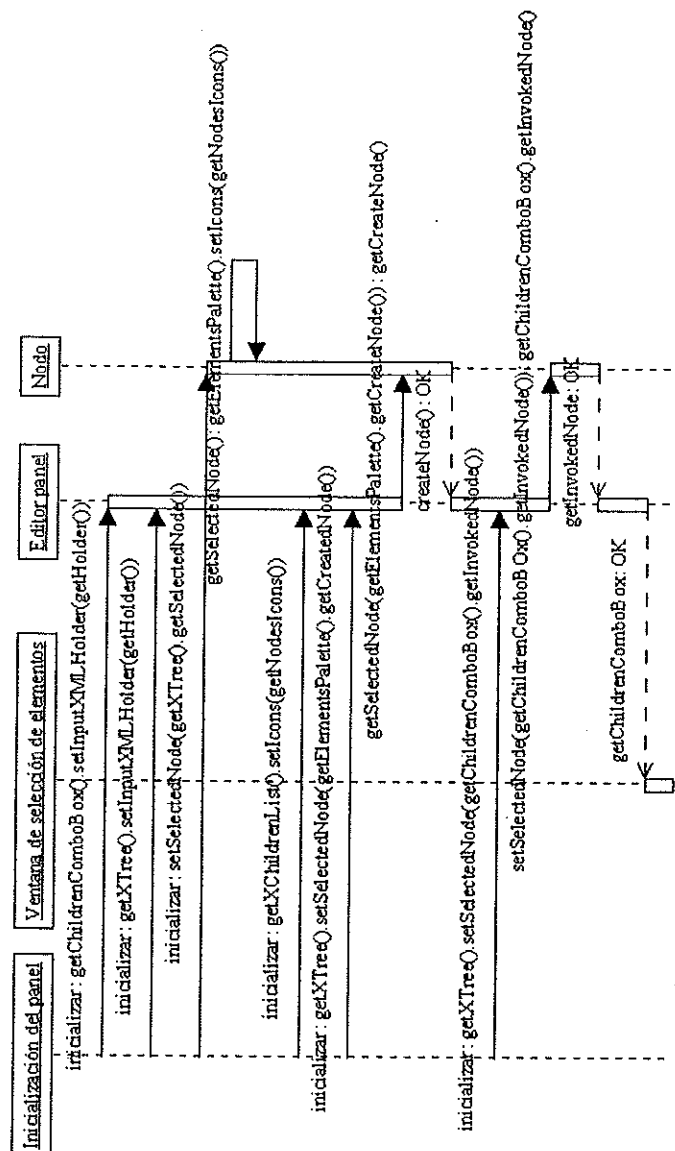


Figura 16: Diagrama de Secuencia para la Edición del Proyecto Inicialización (continuación...).

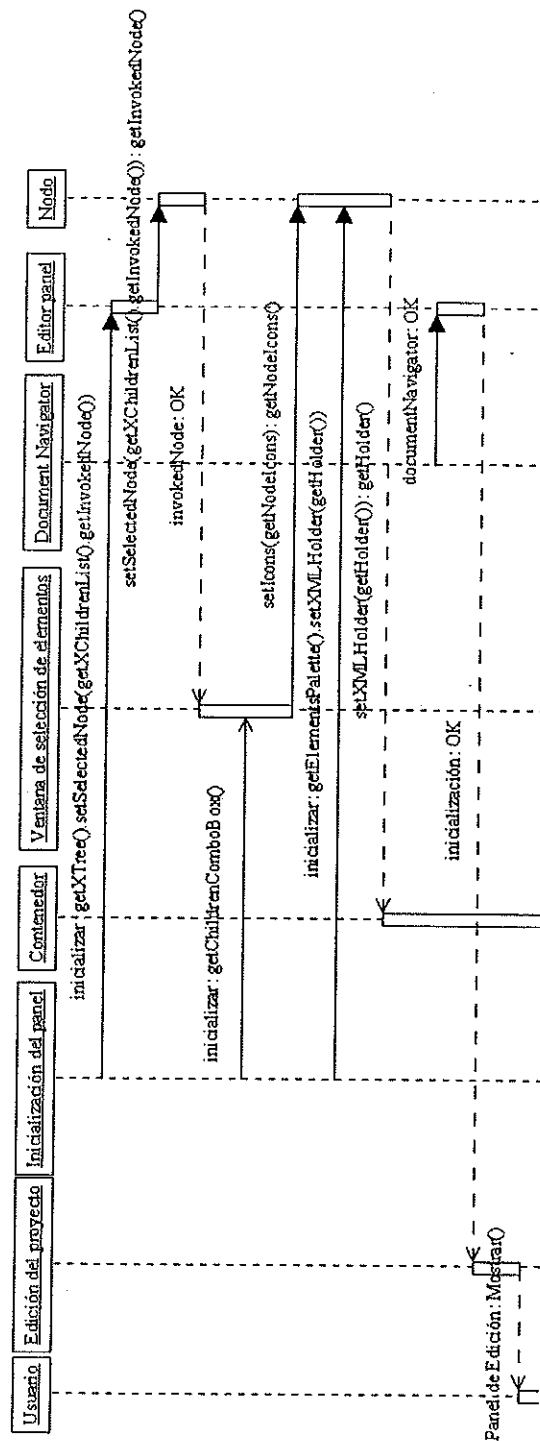


Figura 17: Diagrama de de Secuencia para la Edición del Proyecto Inicialización (continuación...).

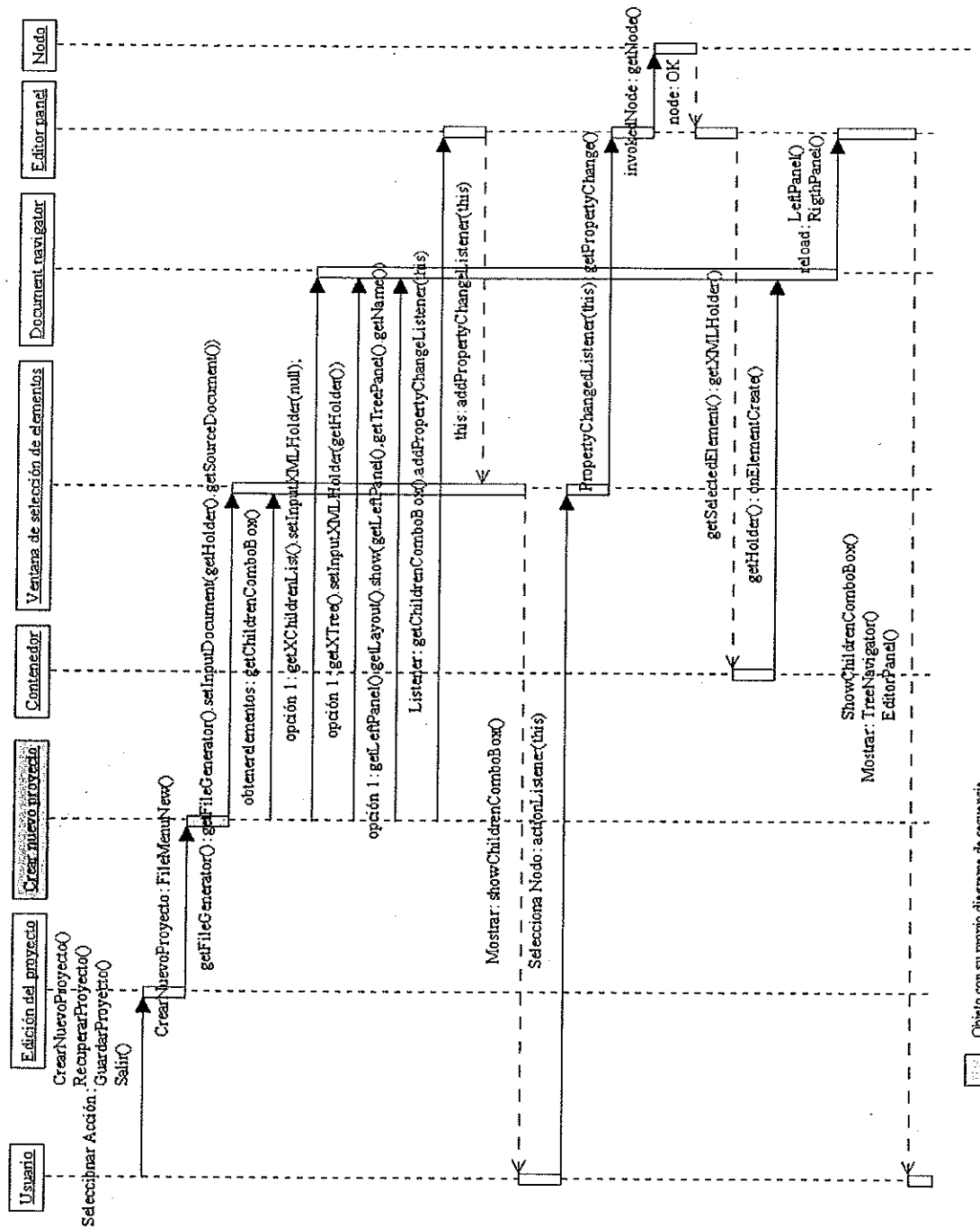


Figura 18: Diagrama de Secuencia para la Edición del Proyecto Elaboración.

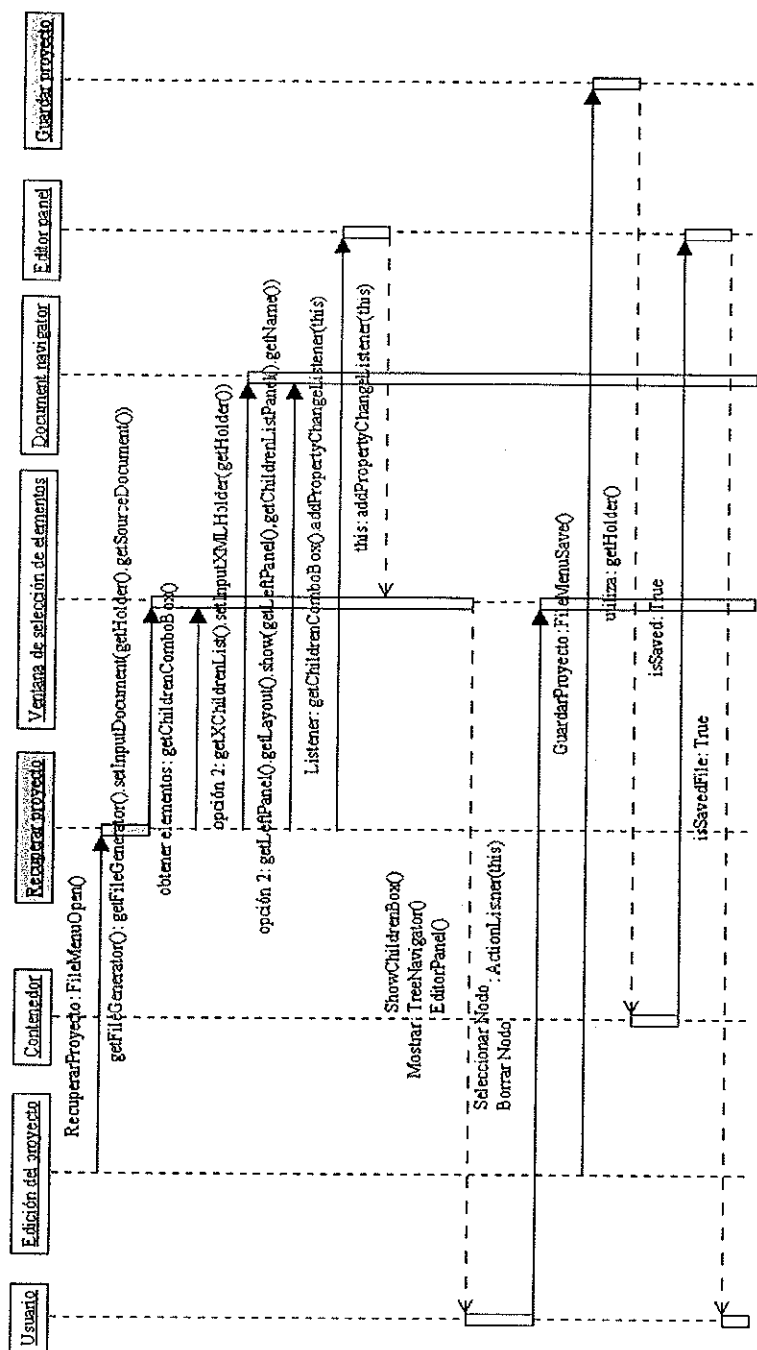


Figura 19: Diagrama de Secuencia para la Edición del Proyecto Elaboración (continuación...).

III.2.5. Diagrama de Secuencia de Selecciona Destino.

En la figura 20 se muestra el diagrama de secuencia perteneciente a Selecciona Destino. Cuando el usuario elige la opción de abrir un documento, la clase SeleccionaDestino crea una instancia de la clase ResourceChooser que muestra la operación de abrir un recurso en su título y botón de acción. Cuando el usuario termina de interactuar con la ventana se verifica que la acción no se haya cancelado, de ser así se verifica el tipo de recurso elegido. Si se eligió trabajar con un documento de servidor de Web se obtiene el objeto WebdavResource que representa a un documento dentro del servidor de Web, enseguida se coloca un seguro contra la escritura en el documento, se obtienen su contenido y su identificador (URL). Si el tipo de documento elegido corresponde a un documento local se obtiene el objeto File que representa a un documento en el sistema de archivo local, se obtiene el contenido del documento y se determina su trayectoria absoluta.

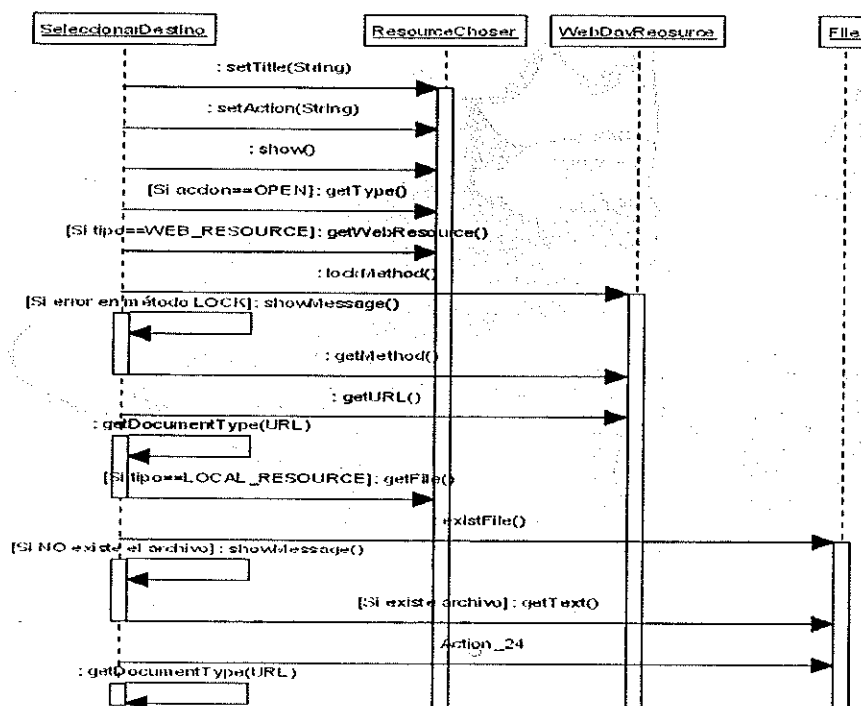


Figura 20: Diagrama de Secuencia Seleccionar destino cuando se abre un documento remoto o local.

III.3 Organización de las clases que conforman al SEIIDM.

Una de las tareas claves para modelar un sistema de software de grandes dimensiones es dividirlo primero en áreas manejables. Aunque estas áreas se llaman dominios, categorías o subsistemas, la idea es la misma: dividir el sistema en áreas que tengan competencias parecidas.

UML introduce la noción de un paquete como el ítem universal para agrupar elementos, permitiendo a los modeladores subdividir y categorizar sistemas. Los paquetes pueden ser usados en cualquier nivel, desde el nivel más alto, donde son usados para subdividir el sistema en dominios, hasta el nivel más bajo, donde son usados para agrupar casos de uso individuales, clases o componentes.

En las figuras 21, 22, 23 y 24 se muestra el diagrama de paquetes correspondiente al SEIIDM, donde se incluyen clases utilizadas en el la herramienta XMas (Apéndice A), para el manejo de documentos XML. También el SEIIDM hace uso del paquete ResourceChoser del proyecto Copper (Figura 25), en el se encuentran las clases necesarias por la ventana de diálogo ResourceChoser la cual permite a los usuarios accesar los recursos de distintos servidores Web, así como los recursos de la máquina local (Capítulo IV).

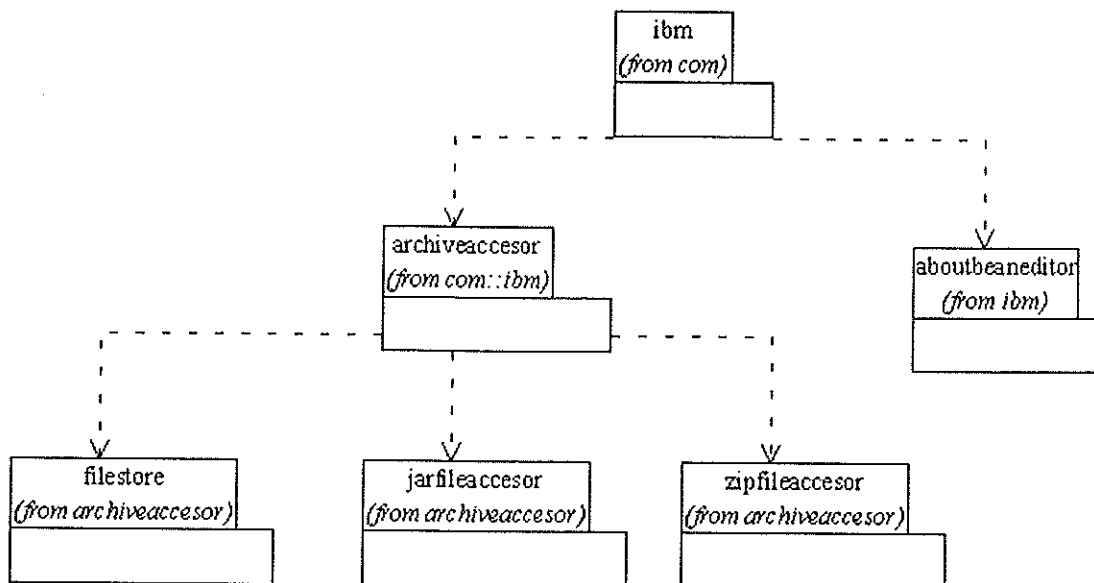


Figura 23: Diagrama de paquetes del SEIDM que contiene los accesos a archivos.

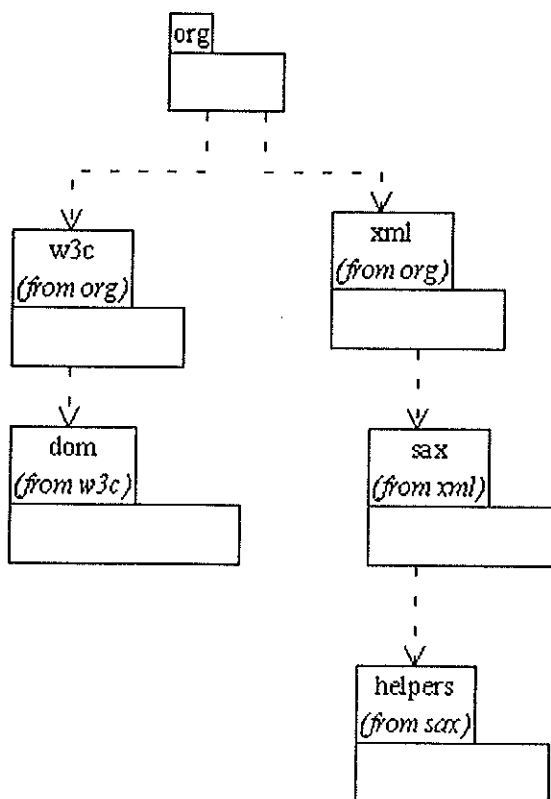


Figura 24: Diagrama de paquetes del SEIDM que contiene los analizadores XML.

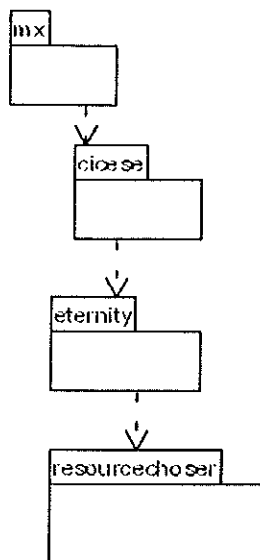


Figura 25: Diagrama de paquetes del SEIDM que contiene las clases ResourceChoser.

III.4 Diagrama de Clases del SEIDM.

Una clase es el descriptor de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y comportamiento. Una clase representa un concepto discreto dentro del sistema que se está modelando. A continuación se presentan las clases principales del SEIDM:

- *Editor*. Esta es la clase principal del SEIDM, es la encargada de representar a la interfaz grafica del editor para la elaboración de proyectos, así como manejar los eventos que ocurran sobre este.
- *Webdavresource*. Es la clase responsable de implementar el protocolo WebDAV, que utiliza la clase Documento al trabajar con los documentos de un servidor de Web.
- *ResourceChoser*. Se encarga de mostrar una ventana de diálogo que permite al usuario acceder tanto los documentos de un servidor de Web como los de la maquina local.

Maneja los eventos que el usuario genere y proporciona un manejador del recurso según la el tipo de recurso elegido.

- *About*. Es la clase encargada de presentar una interfaz grafica con el objetivo de mostrar las condiciones de uso del editor, su versión e información adicional a el.

Los Diagramas de clases de UML se utilizan para documentar la estructura estática del sistema; esto es, que clases hay y como están relacionadas, pero no como interactúan para mostrar comportamientos particulares. Un diagrama de clases puede también mostrar otros aspectos de la estructura estática tales como paquetes.

III.4.1. Diagramas.

La figura 26 muestra el diagrama de clases del paquete ResourceChoser. La clase ResourceChoser se compone de las clases ConnectionsTableModel, WebTableModel y LocalTableModel, las cuales son componentes de interfaz gráfica para manejar listas de servidores de Web que se tengan registrados, documentos almacenados en un servidor de Web y documentos almacenados en la máquina local. Estas clases utilizan las clases WebConnectionInfo, WebResourceInfo y LocalResourceInfo para almacenar la información correspondiente.

La clase ResourceChoser utiliza la clase VDAuthentication para capturar los datos de identificación del usuario, cuando se requieran por parte del servidor de Web. La clase ResourceChoser utiliza los métodos de la clase ConnectionsView para almacenar una lista de servidores de Web.

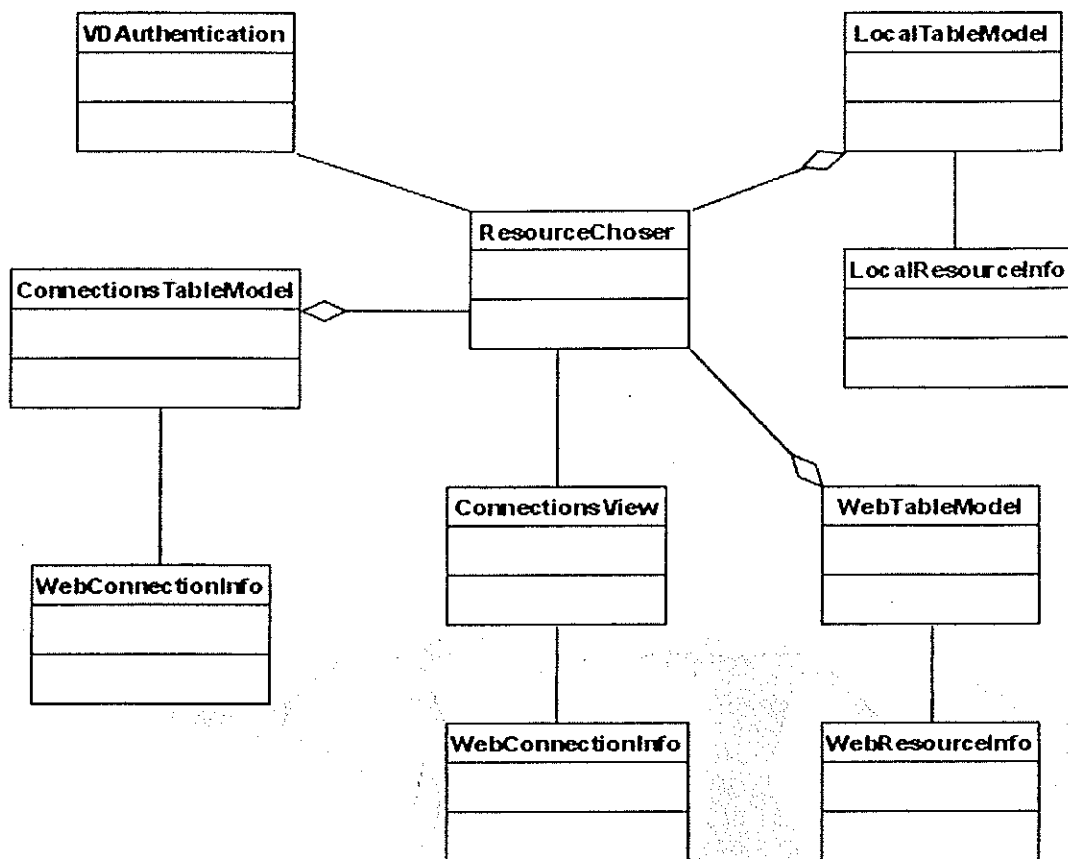


Figura 26. Diagrama de clases del paquete ResourceChoser.

En la Figura 27 se muestra la estructura de clases que utiliza el editor SEIIDM, las clases como XTree, XNavigator, XPanel y XChildrenComboBox permiten generar el ambiente grafico del editor. Las clases DTDEngine, XMLElements, Node, DataSource, XMLHolder, ImageIcon y Component logran hacer el manejo de los documentos XML por generar o que ya estén creados y a través de XMLHolder indicar a las clases XTree y XChildrenComboBox el estado actual del proyecto.

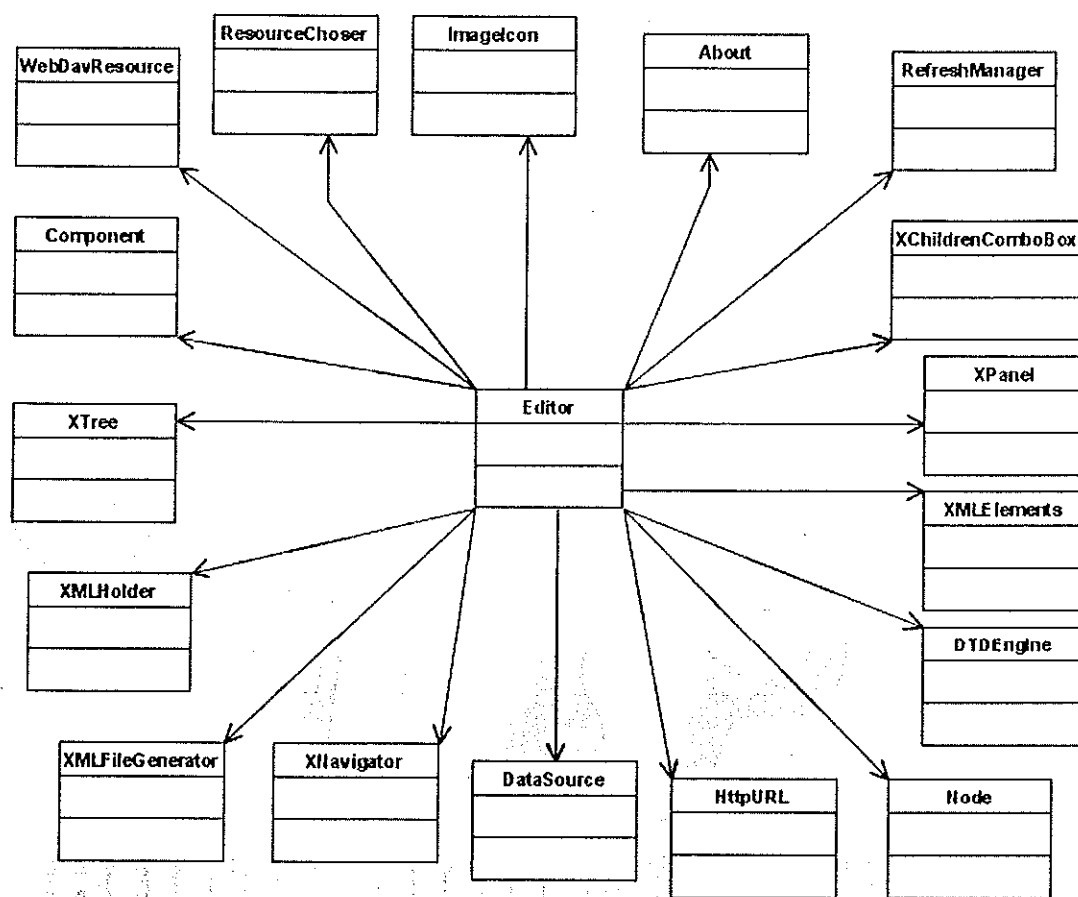


Figura 27: Diagrama de clases del editor SEIIDM primera parte.

Por otro lado en la Figura 28 se presenta el diagrama de clases del editor SEIIDM que corresponde a la estructura de clases graficas que serán presentadas dentro del panel de edición correspondientes al nivel que corresponde, todo este conjunto de clases hacen uso de beans no gráficos individuales en donde estos son ligados al editor por medio de instanciaciones al contenido del mismo. Aquí se pueden agregar tantas clases como sean necesarias para la edición del documento siempre y cuando contengan la estructura adecuada como por ejemplo metadatosEditor es la clase grafica hace uso de un metadatosEditorBeanInfo, metadatosBeanInfo y la clase metadatos.

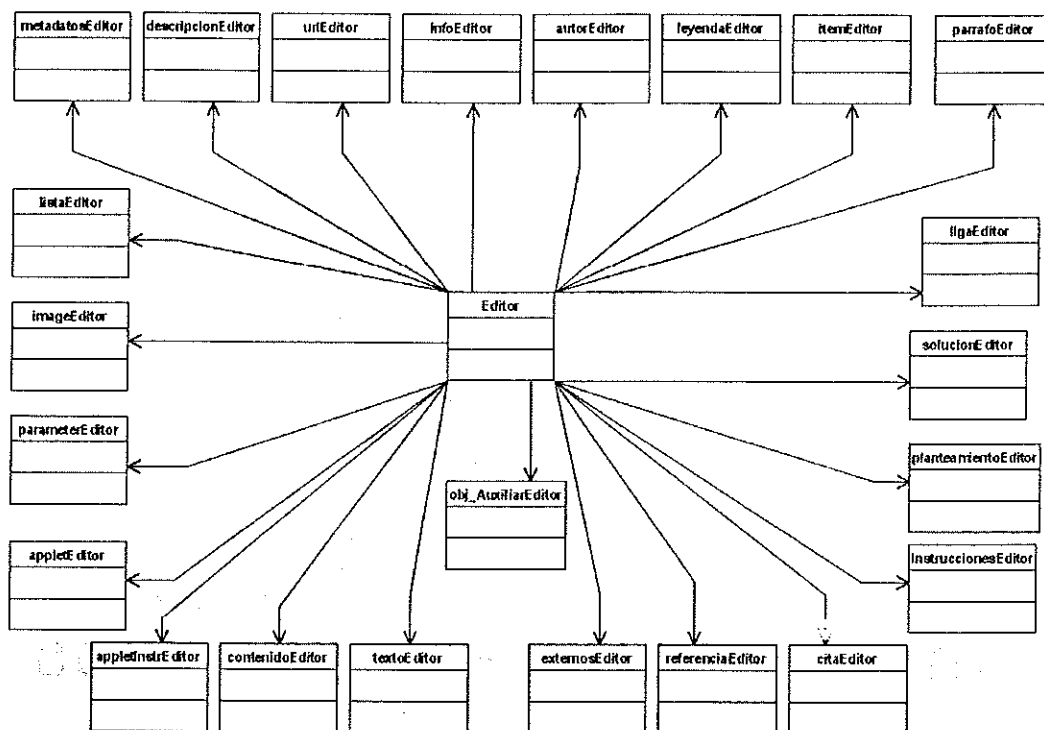


Figura 28: Diagrama de clases del editor SEIIDM segunda parte.

Capítulo IV. Sistema de Edición de Instructores Interactivos de Diversiones Matemáticas (SEIIDM).

IV.1 Arquitectura.

El SEIIDM se compone de dos partes, un cliente que es el editor de XML y un servidor de documentos. El servidor de documentos se utiliza como contenedor de documentos a la vez que atiende las solicitudes de documentos, es decir, el servidor toma físicamente el documento y lo envía al cliente para que el usuario trabaje en él de forma local. El cliente envía el documento modificado para que sea almacenado en el servidor. El servidor también se encarga de manejar los candados sobre los documentos para evitar que estos sean modificados por un tercero mientras se está trabajando en ellos. El servidor de documentos es un servidor HTTP que implementa el protocolo WebDAV (Stein Greg. 2002; Whitehead E. Jim. 2002).

IV.2 Herramientas de Autoría.

Las herramientas de autoría son aquellas que permiten editar contenido sin depender de terceros para su elaboración. Su uso puede ser bastante complejo y es necesario tener un alto nivel de conocimiento de su funcionamiento para lograr hacer una buena edición de documentos.

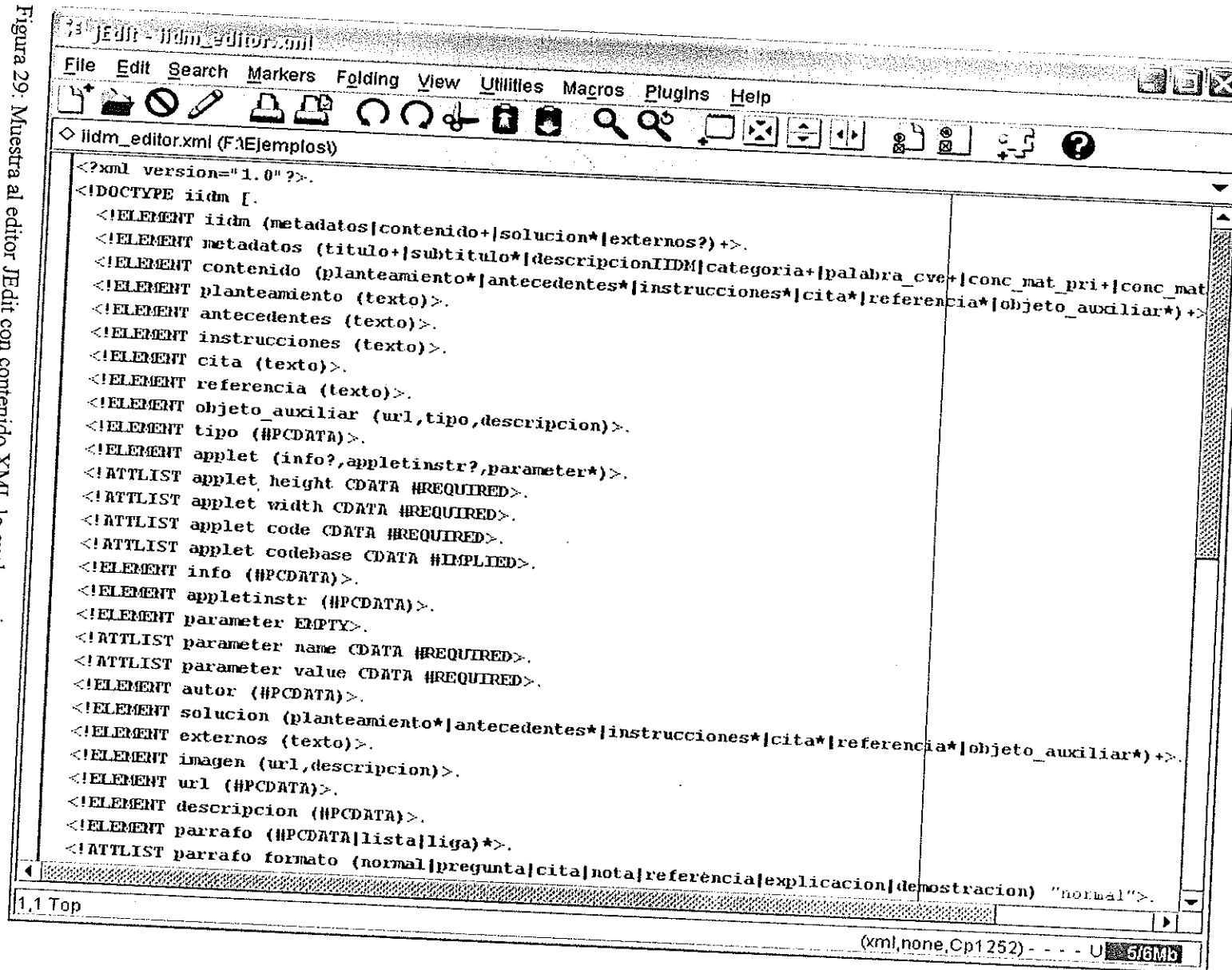
En base a la necesidad de utilizar una herramienta de autoría especializada en XML, se ha creado software como el editor JEdit (URL), el cual permite generar documentos XML validados basándose en un DTD específico: Aunque este editor ofrece mucha ayuda visual al usuario, su uso es sólo para expertos, ya que se requiere nivel bueno de conocimientos de XML y archivos de definición de datos (DTD). La Figura 29 muestra el editor JEdit con un documento DTD.

Otros editores de código abierto, especializados en la edición de documentos XML, mantienen características similares a los del JEdit. En el sentido de que ofrecen ayuda para

la incorporación de las etiquetas definidas en el DTD, así como ayuda visual para observar la estructura del documento XML (como Octave <http://www.octave.org/>), sin embargo, el usuario sigue buscando el código XML, por lo que requiere de un entrenamiento especial para editar el documento cuando se presenta mezclado el contenido con el XML.

Las anteriores consideraciones nos llevaron a considerar la adopción del proyecto XML Master de IBM cuya función es el manejo de XML a partir de un DTD ocultando el XML al usuario (IBM Corp. Sun Microsystems Inc. 1999-2000). Lamentablemente este proyecto carece de documentación que explique con detalle como utilizarlo, sin embargo nos dimos a la tarea de adentrarnos en su funcionamiento. Con su uso, nos dimos cuenta de que la herramienta como tal nos sería de mucha utilidad en el desarrollo de este trabajo de tesis, gracias a que la aplicación hace permisible la reutilización de código, pudimos utilizar mecanismos de análisis de documentos XML, además, entre otras cosas nos facilitó la creación de beans visuales y no visuales. Se utilizó esta herramienta por que cuenta con analizadores XML que han sido probados por desarrolladores de IBM ahorrando costos en tiempo de desarrollo. Se utilizó para la creación de interfaces para cada elemento del DTD con soporte de beans. La explicación y funcionamiento de la herramienta se puede ver en el Apéndice A.

Figura 29: Muestra al editor JEdit con contenido XML lo cual requiere conocimiento del lenguaje.



IV.3 Descripción del Proceso de Edición del SEIIDM.

IV.3.1. Implementación del SEIIDM.

Todo el SEIIDM está implementado utilizando el lenguaje de programación Java (Sun Microsystems Inc. 2003). En su desarrollo se hizo uso de las clases de JDK (*Java Development Kit*) versión 1.4.2. Además se utilizaron clases de otros paquetes que brindan servicios especializados. Entre los que se encuentran; manejo del protocolo WebDAV, manipulación de documentos XML, entre otros.

Tomando en cuenta la importancia de la reutilización de código, al disminuir tiempos y costos en la elaboración y desarrollo de sistemas de software, se planteó como base en la implementación del SEIIDM, elegir proyectos tales como: XML Master (XMas) de IBM (IBM Corp. Sun Microsystems Inc. 1999-2000) y el Sistema COPPER de CICESE (Hiroshi Natsu Cárdenas. 2002). En general, su desarrollo es basado en los mecanismos de manipulación de cadenas XML utilizados en el sistema XMas. La herramienta XMas permite elaborar las interfaces gráficas que caracterizan a cada elemento del archivo de definición de datos (DTD), permitiendo así generar sus “*beans*” visuales como sus no visuales. Durante todo el proceso de desarrollo se utilizan paquetes con clases especializadas simplificando así el proceso de reutilización a futuro.

XMas (XML Master) es un intento por crear un procesador y un editor visual para documentos XML. Estos están basados en *beans* visuales y no visuales de *Xmas Bean Suite*. La aplicación XMas por sí sola utiliza un documento XML como un proyecto de modelo de datos y es creado utilizando los *beans* de *Xmas Bean Suite*.

Como un dato fuente, la aplicación XMas hace uso de un archivo denominado Documento de Definición de Datos (DTD), el cual contiene la información de la estructura XML y el nombre del elemento raíz. De cualquier forma esto es un grupo de *beans* intentando procesar y editar documentos XML, creados de acuerdo al archivo DTD. Este grupo incluye los siguientes elementos:

- *non-visual beans* con una estructura de árbol que permite un fácil acceso a cualquier elemento XML para ser cambiado.

- *Visual beans* representados como *Swing panels*. Cada panel puede ser ligado con *non-visual beans* mencionados arriba y estos intentar editar un elemento XML definitivo.
- El Editor XML, como una aplicación completa intentando editar y mostrar documentos XML correspondientes a un específico archivo DTD.

La aplicación XMas puede simplificar el trabajo por que te permite seleccionar el elemento XML necesario y sus atributos. Entonces de acuerdo a la información recopilada, este genera beans no visuales con una estructura de árbol, el cual permite tener un fácil acceso a cualquier elemento XML y los beans visuales que son parte de un editor consumado de elementos XML. Mientras se ensambla la aplicación a partir de esos beans, no se necesita agrupar totalmente el árbol XML ya que este árbol es generado cuando los beans son creados.

La aplicación XMas es una integración de un ambiente visual que soporta la completa creación de procesadores y editores visuales para documentos XML. Los procesadores y los editores visuales para documentos XML están basados en los siguientes *beans* de *XMas Bean Suite*:

- *Non-Visual beans* que incluye XMLHolder, XMLElement, XMLAttribute y XMLFileGenerator permitiendo abrir y crear un documento XML, salvando un documento procesado, tal como un fácil acceso a cualquier parte del documento. Esto también incluye la notificación al sistema para realizar cambios sobre cualquier selección de elementos XML.
- *Visual beans* son proveídos para los beans XMLElement con una conexión y permitiendo la edición del contenido de texto de los elementos y atributos del XML.

Como XML contiene una estructura de árbol, es más conveniente trabajar con estos utilizando una estructura de árbol por cada bean. El *bean* XMLHolder perteneciente a *XMas Bean Suite* es utilizado para ir almacenando el documento XML y para ir conectando el resto de los *beans* con cada uno de ellos. Este también es responsable del envío de los eventos conectandolo al árbol XMLElement, quien para cada *bean* provee acceso a un

arreglo de elementos XML descrito de la misma forma que en el DTD. Los *beans* XMLElement son conectados según su turno a los *beans* XMLAttribute, los cuales proveen acceso a cada atributo del elemento XML. El bean XMLFileGenerator genera un archivo XML del árbol de objetos DOM, el cual es usado como un modelo básico de datos.

La aplicación XMas permite seleccionar visualmente las necesidades de los elementos XML y sus atributos, entonces conforme a la información recopilada se generan los *beans* no visuales derivados de XMLHolder y XMLElement. Mientras son ensamblados a la aplicación de esos *beans*, no se requiere que se ensamble el árbol XML completamente ya que este árbol es generado en un *bean* derivado de XMLHolder (contenedor). Siendo esto suficiente para conectar el resto de los *beans* no visuales a las propiedades de childXMasElement (ventana de navegación). Siendo no necesario el enlazar los *beans* XMLAttribute con ellos.

El *XMas Bean Suite* incluye un número de *beans* visuales diseñados básicamente con las librerías *Swing*, las cuales son directamente conectadas a los *beans* XMLElement. Estos permiten ir editando los elementos almacenados en él (o sus subelementos y atributos) y manejar los *beans* XMLElement. La aplicación XMas permite contruir paneles para formar esos componentes. Cada panel es proporcionado para editar un definitivo elemento XML acorde a la estructura que es descrita en el DTD.

Los paneles generados son *beans* derivados de la clase JPanel. Estos tienen las propiedades del inputXMLElement, los cuales, son enlazados a su correspondiente *bean* XMLElement con una conexión. Este *bean* puede ser generado dentro de la aplicación XMas o ser fuente de *XMas Bean Suite*.

Por otro lado, la parte medular del SEIIDM es la incorporación de una interfaz para el usuario final llamado, DefaultXMLEditor2, facilitando la edición y la rápida comprensión de su funcionamiento. Este proceso involucra la parte de recuperación de documentos XML, brindando un llamado a "*beans*" y métodos pertenecientes a paquetes utilizados en XMas (Ver Figura 30). Lo más destacable dentro del SEIIDM es la generación de dos paneles donde se construye la estructura de elementos XML mediante la generación del método constructXMLElementStructure(), el cual obtiene el DTD para

generar una tabla hash de cada elemento iniciando con la obtención de elemento raíz, realiza una comparación del documento origen y deposita cada elemento del archivo en un contenedor que permitirá manipularlo. Además se construyó una ventana auxiliar llamada `ChildrenComboBox()`, su función es auxiliar la construcción del documento mostrando los nodos disponibles y en base a esto, el usuario pueda seleccionar sobre cual nodo trabajar. Se presenta también un árbol de navegación ubicado en el panel izquierdo que muestra el estatus actual de la estructura del documento que va generando el usuario. Para la generación de todos estos mecanismos que conforman el editor se utilizaron clases de 7 paquetes.

Otra implementación que agrupa elementos importantes para el funcionamiento del editor, fué la creación de la clase `EditorXML2`, que es una extensión de `DefaultXMLEditor2`. Cuya función es la de inicializar las partes que conformaran el editor correspondiente a la estructura de los elementos que contendrán los beans gráficos del editor, establece la ubicación del DTD y los beans a utilizar mediante una asignación a través de un `setEditorPanel()`.

Se tienen que generar los beans gráficos y no gráficos como `citaEditor` y `citaEditorBeanInfo`, que son parte importante dentro de la aplicación. Aparte en `citaEditor` se establece una clase extendiendo a un `XPanel` donde se definen los parámetros gráficos que se mostrarán al usuario. Por otra parte, `citaEditorBeanInfo` que es un bean, se extiende de `SimpleBeanInfo`, cuya estructura es la nativa de un bean. Aquí se establecen los mecanismos de llenado que utilizará la clase `citaEditor`. Bajo este concepto se desarrollaron 26 clases más con su correspondiente bean.

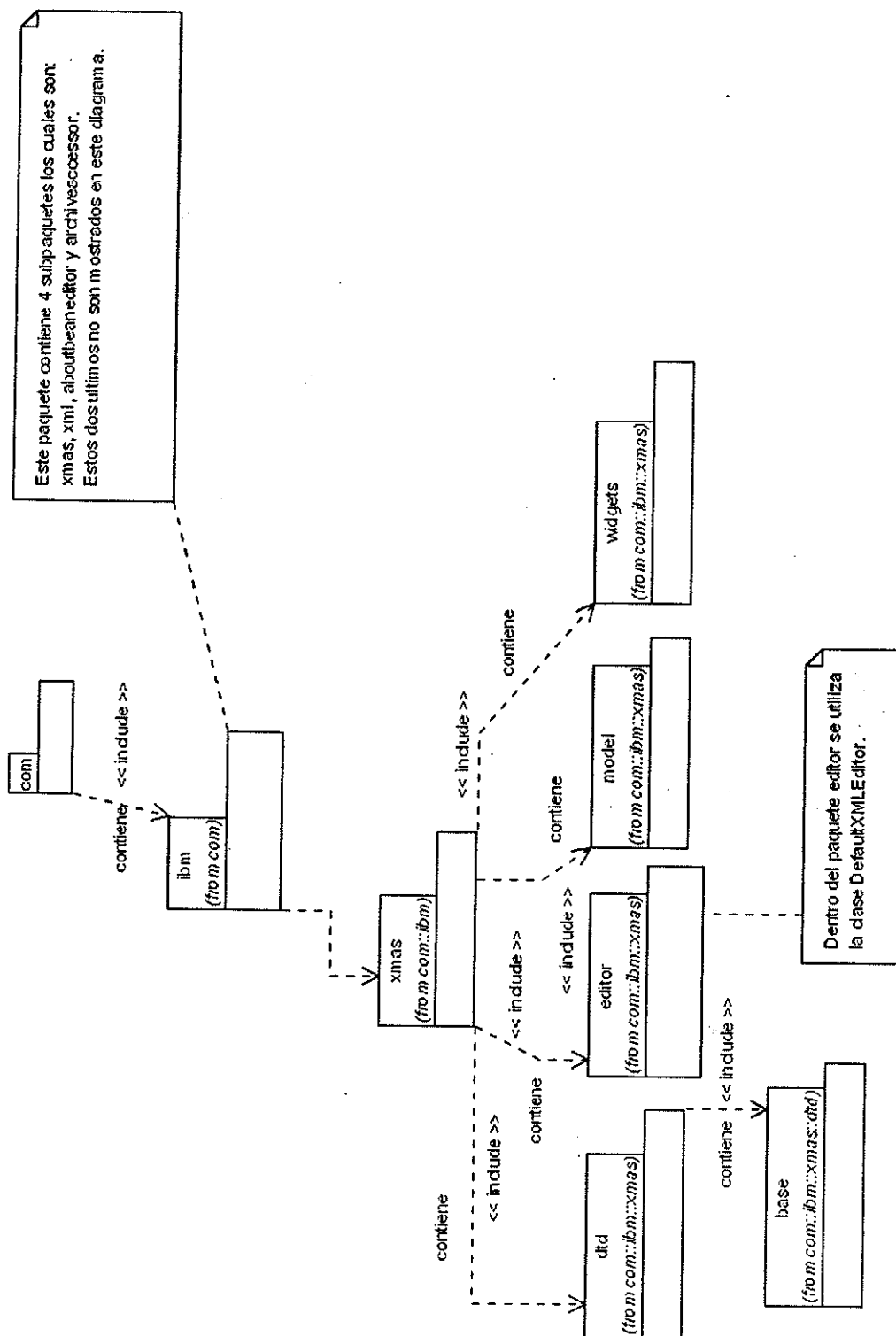


Figura 30: Diagrama de paquetes del SEIIDM primera parte.

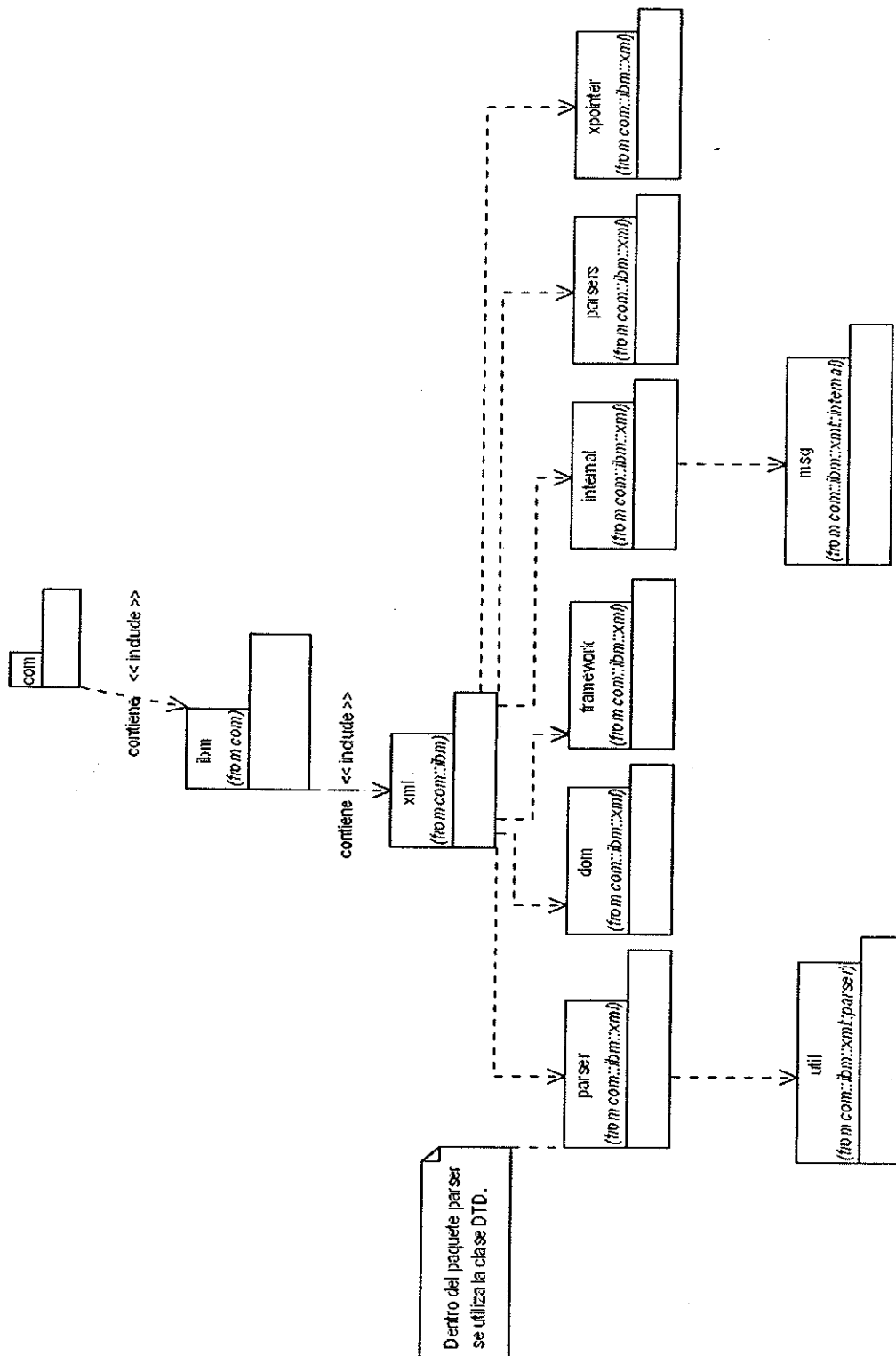


Figura 31: Diagrama de paquetes del SEIIDM segunda parte.

IV.3.2. Funcionalidad del SEIIDM.

El sistema cuenta con 3 grupos dentro del panel de control que se dividen en Archivo, Nodo y Acerca que son parte importante para la generación de documentos (Ver figura 32).



Figura 32: Panel de control del SEIIDM

Archivo es la primera opción del panel de control, contiene los elementos para la generación, recuperación de documentos así como también la salida del sistema, tal como se muestra en la figura 33.

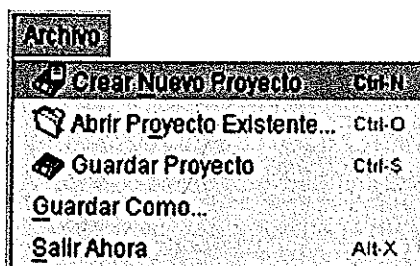


Figura 33: Componentes del panel Archivo del SEIIDM.

La primera opción es Crear Nuevo proyecto. Como su nombre lo indica, es para iniciar un nuevo documento, independientemente si existe uno en edición, se muestra una ventana de diálogo (ver. Figura 34) notificando al usuario la posible pérdida de la información si no lo guarda primero antes de iniciar otro nuevo documento.

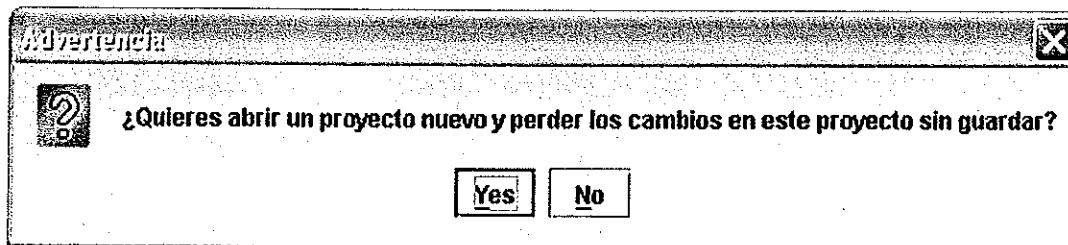


Figura 34: Ventana de Dialogo que advierte al usuario pérdida de información.

La segunda opción es Abrir Proyecto Existente, ésta permite recuperar un proyecto que ya fue editado anteriormente, ya sea en la máquina local o en un servidor remoto. En un inicio se realiza una verificación de existencia de un proyecto en edición, si hay rastro de esto, se muestra una ventana de diálogo similar a la de la figura 34, advirtiéndole al usuario posibles pérdidas de información. Una vez resuelto este problema, se presenta una Ventana de Navegación de Archivos (Figura 35) para que el usuario elija el documento.

La Tercer y Cuarta opción son Guardar Proyecto y Guardar Como..., cuya función es la de salvar el documento que fue generado, para hacer este proceso de salvado también se hace uso de la Ventana de Navegación de Archivo, en el primer caso sólo cuando es por primera vez, en las siguientes ya no es necesario. Para el segundo caso, siempre es utilizada presentando la opción de cambiar de nombre y de unidad de almacenamiento, ya sea local o remoto, por supuesto transparente para el usuario.

Y por último la opción de Salir del Sistema, que realiza una verificación del proyecto existente y si este no ha sido grabado muestra una ventana de diálogo (figura 35) que advierte sobre la pérdida de información si no guarda el proyecto, posteriormente se permite cerrar la aplicación.

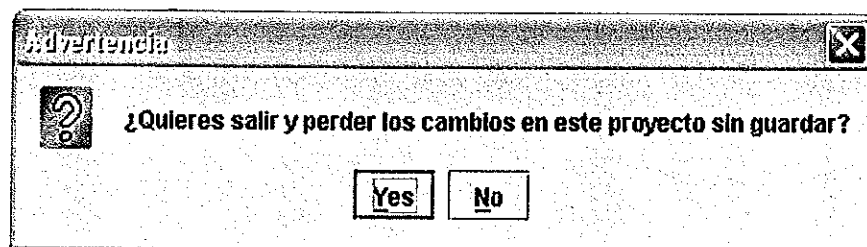


Figura 35: Ventana de diálogo que advierte pérdida de información cuando no ha sido grabado el proyecto y se desea salir.

Nodo es la segunda opción del panel de control, contiene dos elementos que son de utilidad para moverse o borrar nodos existentes en el Árbol de Navegación, las opciones son mostradas en la Figura 36.



Figura 36: Componentes del panel Nodo del SEIIDM.

Como se puede observar en la Figura 36, existen dos opciones, donde la primera es Cambiar uno Arriba, que tiene como finalidad poder moverse un nivel arriba del nivel actual en que se encuentre ubicado dentro del árbol de navegación, si no existe nivel esta opción es deshabilitada. La segunda opción es Eliminar nodo actual, su función es quitar del árbol de navegación un nodo específico no importando que este tenga hijos, por que también serán eliminados.

Acerca es la tercera opción del panel de control, su finalidad es mostrar al usuario la versión actual del SEIIDM, así como también los créditos del personal e instituciones que desarrollaron el Sistema. (Ver Figura 37 y 38).



Figura 37: Opciones del Panel Acerca del SEIIDM.

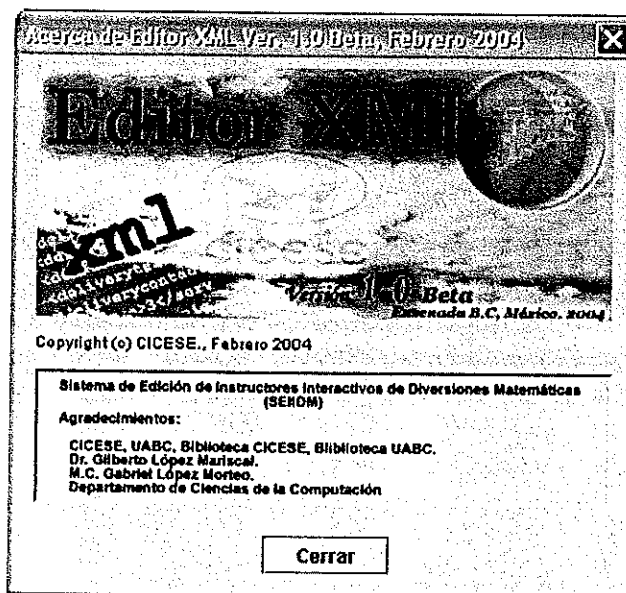


Figura 38: Ventana que muestra la versión y los créditos del SEIIDM.

La barra de herramientas contiene una serie de iconos de acceso rápido que permiten realizar algunas operaciones del panel de control con sólo dar un clic con el ratón ver Figura 39.

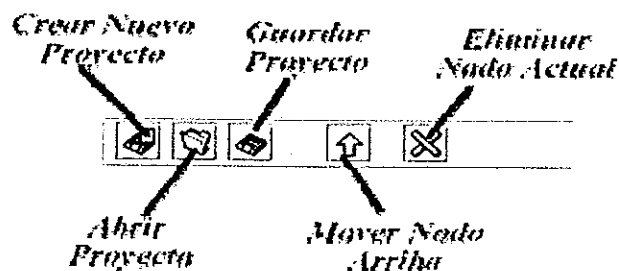
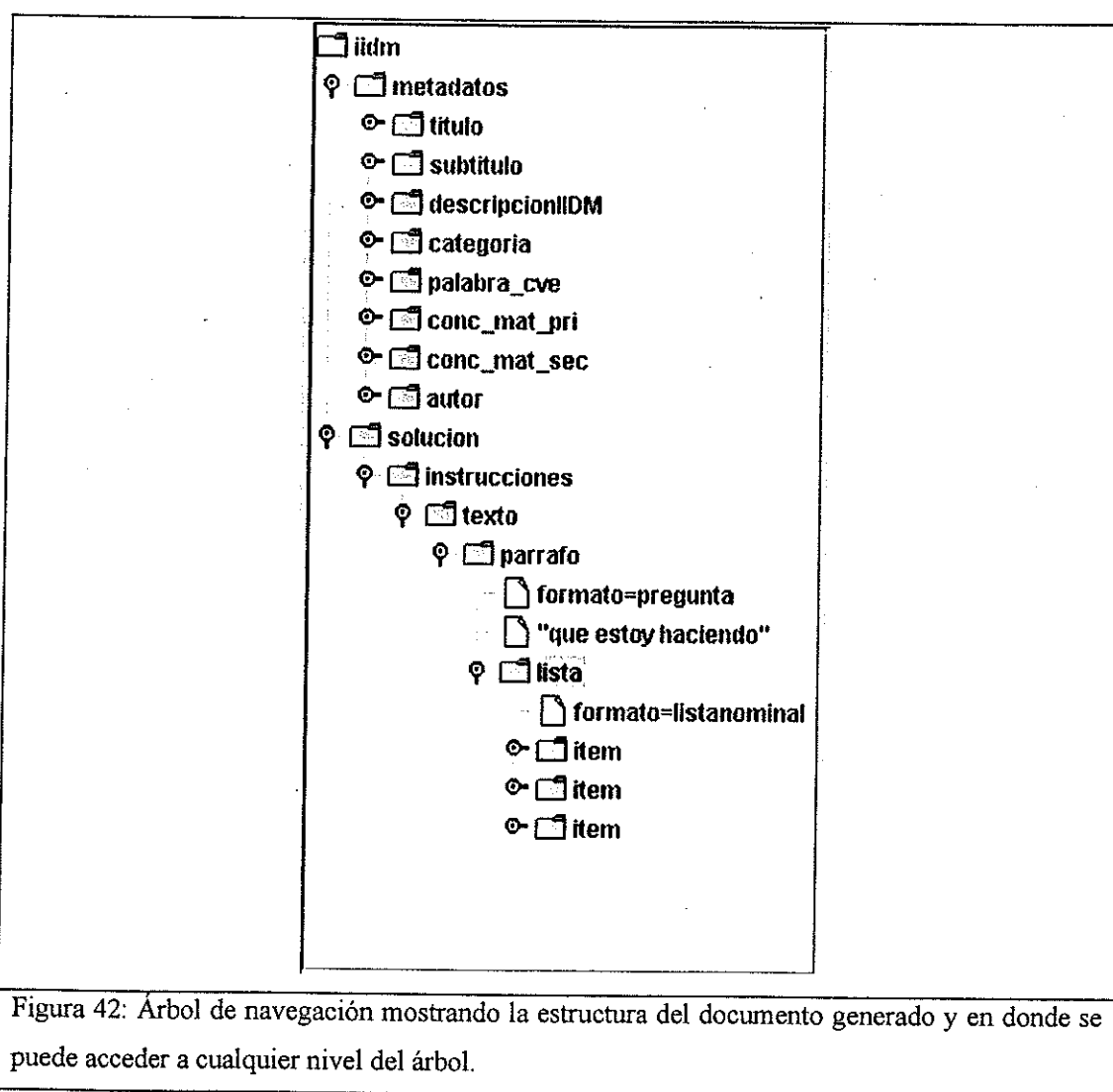


Figura 39: Barra de Herramientas de acceso rápido con funcionalidades básicas del panel de control.

La ventana de navegación y el árbol de navegación son fundamentales durante la creación y modificación del proyecto a generar o generado. En primera instancia, al generar un nuevo proyecto, en el árbol de navegación se muestra el elemento padre, el cual no puede ser eliminado (ver Figura 40) y en la ventana de navegación se presentan los elementos pertenecientes al padre que pueden ser agregados al proyecto como se muestra en la Figura 41.

Figura 40: Árbol de navegación mostrando el elemento padre al iniciar la generación del proyecto.	Figura 41: Ventana de navegación mostrando opciones generales de inicio de edición

Al seleccionar un elemento de la ventana de navegación de la figura 41, éste pasa a formar parte del árbol de navegación, cambiando así el contenido que se presenta en la ventana de navegación. Esto se debe a que cada elemento tiene varios niveles de navegación correspondientes a un elemento en particular. A raíz de ese cambio, sólo se presentan las opciones que son válidas dentro de la secuencia de elaboración del documento. Toda la estructura generada es reflejada en el árbol de navegación, permitiendo eliminar algunos elementos o agregarlos desde cualquier nivel disponible. (Ver Figura 42).



El panel de edición, en conjunción con la ventana de navegación y el árbol de navegación tienen un rol importante, ya que es aquí donde el usuario agrega contenido dependiendo de los elementos seleccionados. Su llenado es sencillo, similar a un formulario de una página Web, que lleva de la mano al usuario durante la elaboración del documento. Todo lo escrito en el panel de edición es reflejado también en el árbol de navegación al momento de cambiar de campo (ver figuras 43 y 45). Dentro del panel de edición se presentan diferentes formas dependiendo el elemento que se este editando.

The image shows a software interface for editing metadata. On the left is a sidebar with a tree view containing 'idm' and 'metadatos' (which is selected). The main area on the right contains a form with the following fields:

- Titulo :** [text input]
- SubTitulo :** [text input]
- Descripción :** [text input]
- Categoria :** [text input]
- Palabra Clave :** [text input]
- Conocimientos Matematicos Primarios**
[text input]
- Conocimientos Matematicos Secundarios**
[text input]
- Autor :** [text input]

Figura 43: Panel de Edición mostrando formato correspondiente a metadatos como lo refleja el árbol de navegación en la parte izquierda.

```

ELEMENT METADATOS
COMMENT: Informaci3n relacionada con la descripci3n del
IDM y que funciona como el v3nculo con los metadatos
para los objetos de aprendizaje.
-->
<ELEMENT metadatos (titulo+|subtitulo*|descripcionIDM|categoria+|palabra_cve+|conc_mat_pri+|conc_mat_sec*|autor*)+>
<ATTLIST metadatos >
<!--
ELEMENT: autor
COMMENT: Relaci3n de autor(es): autores de la divisi3n
matem3tica, autores del texto, autores de im3genes y autores de
aplicaci3n
-->
<ELEMENT autor (PCDATA) >
<ATTLIST autor >
<!--
ELEMENT: conc_mat_sec
COMMENT: Concepto Matem3tico Secundario
-->
<ELEMENT conc_mat_sec (PCDATA) >
<ATTLIST conc_mat_sec >
<!--
ELEMENT: conc_mat_pri
COMMENT: Concepto Matem3tico Principal
-->
<ELEMENT conc_mat_pri (PCDATA) >
<ATTLIST conc_mat_pri >
<!--
ELEMENT: palabra_cve
COMMENT:
-->
<ELEMENT palabra_cve (PCDATA) >
<ATTLIST palabra_cve >
<!--
ELEMENT: categoria
COMMENT: Categori3n: historia, geograf3a, entidad, efem3rides,
problemas, etc
-->
<ELEMENT categoria (PCDATA) >
<ATTLIST categoria >
<!--
ELEMENT: subtitulo
COMMENT: Subtitulo agregado o no al IDM
-->
<ELEMENT subtitulo (PCDATA) >

```

Figura 44: DTD correspondiente reflejando el formato correspondiente a metadatos que muestra las Figuras 43 y 45.

- idm - metadatos - titulo "El principito" - subtitulo "La revolucion de la luna" - descripcionIDM "Novela con contenido historico sobre la verdadera historia del principito en la luna" - categoria "Ninos menores de 3 años" - palabra_cve "PC0001" - conc_mat_pri "Ecuaciones" - conc_mat_sec "Integrales" - autor "Juan Osorio y Sergio Andrade"	Titulo : El principito SubTitulo : La revolucion de la luna Descripci3n : Novela con contenido historico sobre la verdadera historia del principito en la luna Categoria : Ninos menores de 3 años Palabra Clave : PC0001 Conocimientos Matematicos Primarios Ecuaciones Conocimientos Matematicos Secundarios Integrales Autor : Juan Osorio y Sergio Andrade
--	--

Figura 45: Panel de edici3n con contenido reflejado tambi3n en el 3rbol de navegaci3n del lado izquierdo.

La ventana de control de errores es presentada cuando al abrir un proyecto existente este no cumple con las especificaciones del IIDM y es entonces por falta de concordancia se presentan los errores y son notificados al usuario como se muestra en la Figura 46.

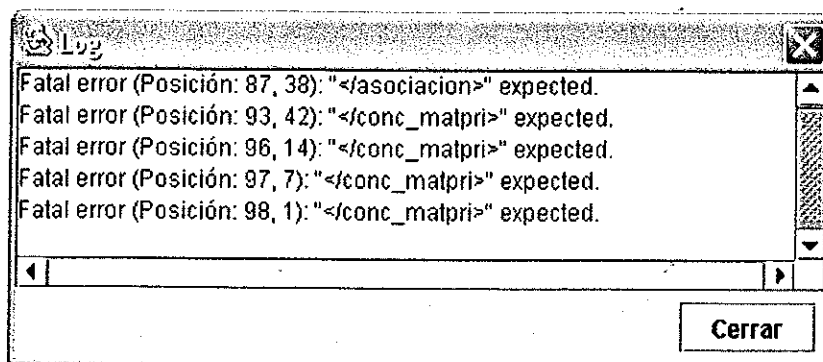


Figura 46: Ventana de errores mostrando posición y tipo de error del archivo XML leído.

Por último el documento XML generado al momento que es guardado el proyecto se puede observar en la Figura 47.

```
- <iidm>
  - <metadatos>
    <titulo>El Principito</titulo>
    <subtitulo>La revolucion de la luna</subtitulo>
  - <descripcionIIDM>
    Novela con contenido historico sobre la verdadera historia del principito en la luna
  </descripcionIIDM>
  <categoria>NI♦os menores de 3 a♦os</categoria>
  <palabra_cve>PC0001</palabra_cve>
  <conc_mat_pri>Ecuaciones</conc_mat_pri>
  <conc_mat_sec>Integrales</conc_mat_sec>
  <autor>Juan Osorio y Sergio Andrade</autor>
</metadatos>
</iidm>
```

Figura 47: Documento XML generado con el SEIIDM.

IV.4 Edición del IIDM Colaborativa a través del SEIIDM.

IV.4.1. Acceso a Servidor Remoto WebDAV.

En el desarrollo de nuevas tecnologías en el mundo de la computación, es cada día más importante la comunicación mediante software a servidores remotos, esto con el fin de dar acceso a la información ahí contenida. En años recientes se han realizado considerables esfuerzos para dar soporte a la colaboración sobre el Web. Estos incluyen trabajos sobre búsquedas sincrónicas, espacios de trabajo colaborativos y ambientes de interacción casual, por mencionar algunos.

Hoy en día, el principal uso del *World Wide Web* (Web) es para buscar información en forma de sólo lectura. Sin embargo, ésta no fue la concepción original del Web, el objetivo del Web era el de servir como medio de colaboración dando capacidades de lectura y escritura sobre los recursos (Whitehead E. James. 1998).

Otro aspecto relevante a la coautoría de documentos sobre el Web es el de la forma en que normalmente se crea o actualiza la información en los servidores. Usualmente la información almacenada en los servidores se actualiza de forma manual por el administrador del sitio. Los documentos son preparados en un ambiente de edición y luego se colocan en el espacio público del servidor Web (Decouchant D. Favela J. y Martínez A. M. 2001).

El modelo actual de cliente está altamente orientado a interacciones pasivas con la información, la mayoría de los clientes y servidores no toman ventaja de los aspectos del protocolo de intercambio de información (*Hyper Text Transfer Protocol*, HTTP), el cual puede soportar interacciones más activas.

El navegador y el protocolo HTTP son utilizados para recuperar información de un servidor de Web, pero la información típicamente es provista sin utilizar HTTP, lo que significa que se escribe en el sistema de archivos pasando por alto al servidor. Esto es, el servidor de Web usualmente mapea un localizado universal de recursos (URL) a un archivo en el sistema local de archivos, estos archivos no son creados o actualizados a través de interacciones HTTP con el servidor, sino que son actualizados con operaciones

de tipo puerta trasera “back door” con el sistema de archivos (Murray S. Mazer y Charles L. Brooks. 1998).

La visión original del Web como un medio de colaboración de lectura y escritura no se perdió. En años recientes, ha surgido un interés para desarrollar aplicaciones de autoría colaborativa sobre el Web. En 1995 aparecen dos herramientas comerciales que soportaban autoría de Web remota: NaviPress de NaviSoft y FrontPage de Vermeer. Estos productos comenzaron a desarrollar un mercado de herramientas de autoría las cuales permiten a un usuario editar documentos HTML (*HyperText Markup Language*) remotamente, tomando ventaja de la habilidad de trabajar a distancia sobre el Internet (Whitehead E. James y Wiggins Meredith. 1998). La experiencia que se obtuvo en el desarrollo de estos primeros sistemas de autoría basados en el Web era clara en un punto, no podían ser creados sistemas de autoría en el Web útiles sin extender HTTP. Tanto NaviPress como FrontPage tuvieron que agregar funcionalidad por encima de HTTP para satisfacer sus necesidades (Whitehead E. James. 1998).

Impulsados por el deseo de contemplar la visión original del Web como medio de colaboración, se encuentran por primera vez un grupo de personas interesadas en autoría distribuida sobre el Web. Posteriormente en Junio de 1996 en la conferencia *América Online*, se identificaron aspectos importantes en las herramientas de autoría y encontraron la necesidad de una extensión estándar al protocolo HTTP (Fielding R. 1997) para dar capacidades de autoría remota. Este grupo de personas vino a formar el grupo de trabajo “*Web Distributed Authoring and Versioning*” mejor conocido como WebDAV.

El protocolo WebDAV, contiene mecanismos de manipulación de archivos remotos que son de utilidad en el EIIDM, así que a continuación se muestra una descripción del mecanismo, como funciona y como fue implementado en el Editor de Instructores Interactivos de Diversiones Matemáticas (EIIDM).

IV.4.2. El Protocolo WebDAV en el SEIIDM.

WebDAV (*Web Distributed Authoring and Versioning*) es un conjunto de extensiones al protocolo HTTP que permite a los usuarios colaborar en la edición y gestión

de recursos Web en servidores remotos. Su objetivo es extender el protocolo HTTP para permitir una arquitectura abierta en el nivel del protocolo, para desarrollar nuevas herramientas de creación distribuida en la Web, haciendo hincapié en la creación de páginas Web en colaboración [RFC 2518]. WebDAV define operaciones sobre las propiedades, las colecciones, los espacios de nombres y la protección de sobreescritura (mecanismo de bloqueo) y, para estas operaciones, define nuevos métodos, encabezados, organismos de entidad de petición y respuesta. Sin embargo las características de tratamiento de versiones incluidas en la propuesta original se ha trasladado al grupo de trabajo de la IETF Delta-V que pretende extender WebDAV y HTTP/1.1 en estas características (Whitehead E. J. 1999).

El mecanismo de bloqueo definido en WebDAV ha sido diseñado para evitar el problema de la pérdida de actualización, empleando las operaciones de bloqueo y desbloqueo. El proceso del problema de pérdida de actualización está esquematizado en la Figura 48. El usuario obtiene el documento para trabajar localmente con él (48a); posteriormente un segundo usuario baja el documento para modificarlo, sin percatarse que otro autor se encuentra trabajando en él (48b); El primer autor actualiza el documento con sus cambios (48c); por último, el segundo autor sube sus modificaciones sobrescribiendo las modificaciones del primer autor (48d).

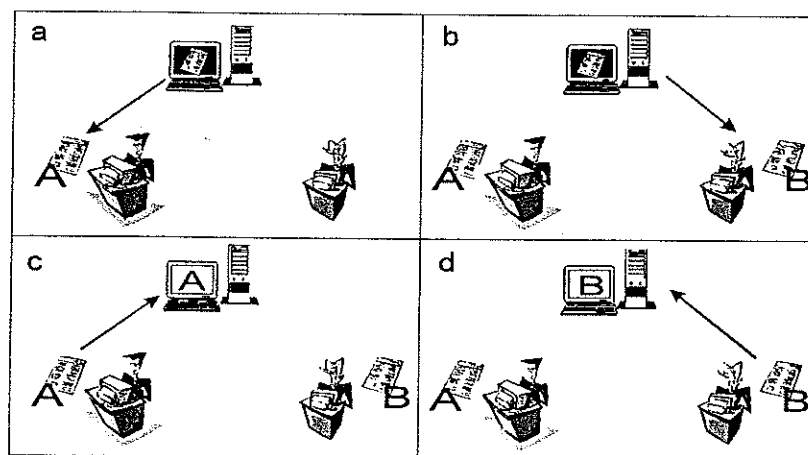


Figura 48. Problema de la pérdida de la actualización. Figura tomada de (Whitehead, 1998). Sinopsis del proceso.

Un bloqueo controla el acceso de escritura a un recurso, limitando las operaciones de escritura HTTP, como *PUT*, *POST*, *DELETE*. Esto significa que solamente alguien que conozca el bloqueo podrá ejecutar estas operaciones. El protocolo WebDAV también define dos ámbitos de bloqueo: *bloqueo exclusivo* y *bloqueo compartido*. Un bloqueo exclusivo garantiza que una persona posee el bloqueo y que nadie más podrá bloquear el recurso (esto no afecta a la lectura del recurso). Un bloqueo compartido permite a un grupo de usuarios crear sus bloqueos sobre un recurso. Los grupos a su vez deben confiar unos en otros para prevenir problemas de acceso. Además, cada bloqueo tiene una duración límite y el usuario no puede olvidar que los bloqueos pueden desaparecer en el servidor debido a errores en el servidor.

En el WebDAV los bloqueos pueden tener un alcance de un sólo recurso, incluyendo todas las propiedades del recurso, o jerarquía de recursos, por ejemplo una colección y todos sus recursos miembros. Un mecanismo para descubrir bloqueos (propiedad del WebDAV) permite a los autores encontrar algún bloqueo en un recurso WebDAV.

La incorporación del protocolo WebDAV al SEIIDM surge de la necesidad de editar contenido XML localmente y almacenarlo remotamente.

La implementación del protocolo WebDAV es incorporada al SEIIDM mediante librerías WeDAV del proyecto Slide, (Apache Software Foundation. 2002) el cual es un subproyecto del proyecto Jakarta (Apache Software Foundation. 2002). Estas herramientas permiten que el cliente realice operaciones de comunicación con el servidor a través del Web, haciendo uso de los métodos de comunicación HTTP. Por su parte el servidor utiliza las librerías WebDAV como medio de transporte en la generación de respuestas a solicitudes del servidor.

El SEIIDM incorpora la reutilización de código del proyecto COPPER, (Hiroshi Natsu Cárdenas. 2002) que es un sistema de edición colaborativa que utiliza recursos del protocolo WebDAV.

A partir de la idea de desarrollo de Software Libre, es viable la reutilización de código de aplicaciones y librerías bajo esta licencia. Esto acelera el proceso de desarrollo, evita el desperdicio de recursos en repetir esfuerzos y aumenta la calidad general del

trabajo al basarse en código que ya ha sido verificado y probado. La tendencia de las empresas de desarrollo de software es practicar la programación por módulos bajo un estándar que permita la reutilización.

El modelo de programación que contiene el proyecto COPPER tiene la ventaja de ser modulado y con esto permitir la reutilización. El módulo que fue reutilizado es el llamado ResourceChoser (Hiroshi Natsu Cárdenas, 2002) el cual contiene los mecanismos de búsqueda tanto en servidores como en maquina local como se muestra en el diagrama de clases del paquete *ResourceChoser* en la Figura 49.

Durante la incorporación del Módulo ResourceChoser se utilizaron las siguientes características:

- Autentificación con WebDAV.
- Sistema de navegación en Servidores WebDAV.
- Sistema de navegación en Máquina Local.
- Sistema de bloqueo.

La clase ResourceChoser se compone de las clases ConnectionsTableModel, WebTableModel y LocalTableModel, las cuales son componentes de interfaz gráfica para manejar listas de servidores de Web que se tengan registrados, documentos almacenados en un servidor de Web y documentos almacenados en la máquina local. Estas clases utilizan las clases WebConnectionInfo, WebResourceInfo y LocalResourceInfo para almacenar la información correspondiente.

La clase ResourceChoser, utiliza la clase VDAuthentication para capturar los datos de identificación del usuario, cuando el servidor Web lo requiera. La clase ResourceChoser utiliza los métodos de la clase ConnectionsView para almacenar una lista de servidores de Web. Este manejador de archivos permite al usuario trabajar con documentos que se encuentran almacenados en la máquina local o con los documentos que se encuentran almacenados en un servidor Web, utilizando las ventajas que ofrece WebDAV. La ventana de diálogo es capaz de presentar un sistema de archivos, resultado de integrar los contenedores de los servidores Web que el usuario tenga registrados. La ventana de diálogo trata de hacer transparente al usuario el pasar de un ambiente remoto a uno local.

Utilizando la ventana de diálogo, el usuario es capaz de navegar por el sistema de archivo de la maquina local así como con los contenedores de servidores Web como si se tratara de un mismo sistema de archivos.

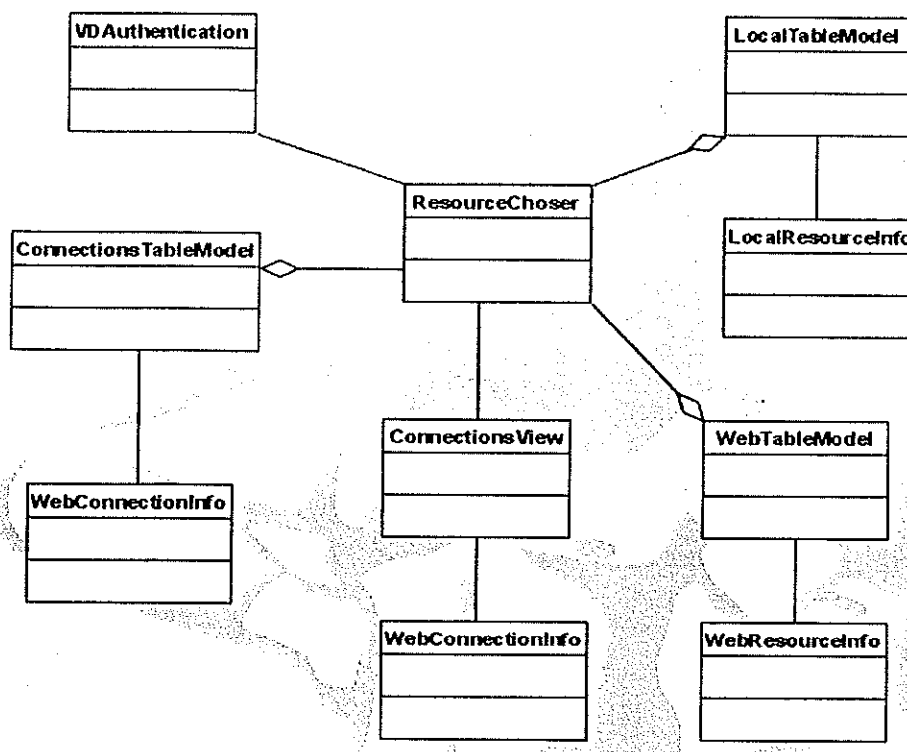


Figura 49. Diagrama de clases del paquete ResourceChoser.

El mecanismo de autenticación definido para el uso con HTTP y WebDAV también es aplicable para el protocolo de control de acceso WebDAV, en particular los mecanismos de autenticación básico y de Resumen (*Basic and Digest Access Authentication*) definidos en [RFC2617].

El esquema de autenticación básica (*Basic Authentication*) está basado en un modelo en el cuál el cliente puede autenticarse el mismo con un identificador de usuario y una contraseña.

El sistema de navegación en servidores WebDAV consiste en establecer una serie de comunicaciones mediante el protocolo WebDAV con el servidor, en donde se pide un listado de los recursos existentes en el directorio raíz del usuario en el servidor WebDAV

que se especificó. Haciendo uso de los métodos de la librería WebDAV se realiza una comparación de cada uno de los recursos que contenga el servidor y se determina si son colecciones (Carpetas) o son recursos (Archivos). Si se trata de una colección la ventana de Navegación ésta realiza las operaciones de acceso a la colección y reestablece nuevamente el listado del contenido actual del URI. Lo anterior, se realiza creando una imagen temporal de los recursos del Web la cual permita acceder a diferentes niveles de navegación sin perder los anteriores.

El sistema de navegación en Máquina Local permite acceder a los recursos del sistema local de la misma forma que con el servidor WebDAV con la ventaja que en esta ocasión no se requiere autenticación para hacer las operaciones de búsqueda en el sistema.

El bloqueo es la habilidad para ponerle candado a un recurso mediante mecanismos de acceso serializado de este recurso. El utilizar un candado garantiza a un cliente editor que otro cliente no podrá modificar el mismo recurso mientras él lo este editando. De esta manera, el cliente puede prevenir problemas de pérdida de actualizaciones.

IV.4.3. Implementación del Sistema.

Durante el proceso de implementación se desarrollaron los métodos para acceder y guardar archivos remotos y locales. Estos métodos están íntimamente ligados con las librerías `org.apache.commons.httpclient`, `org.apache.webdav.lib`, `org.apache.util` y `mx.cicese.eternity`.

Inicialmente, se generó un “listener” para controlar los eventos relacionados con la obtención de archivos, el cual realiza una verificación de un archivo en edición existente. Si este archivo existe, muestra una ventana de diálogo, la cual contiene una advertencia de pérdida de información si no es guardado el proyecto en edición. Si no es así se prosigue al llamado del método `ShowOpenDialog()` (`mx.cicese.eternity.ResourceChoser`), el cual realiza una instancia del recurso correspondiente a la **ventana de navegación** que contiene todos los métodos de la clase `ResourceChoser`.

Posteriormente, se hace el llamado del método `ShowDialog()` (`mx.cicese.eternity.ResourceChoser`) el cual permite mostrar la ventana de navegación. Este llamado regresa un valor con el cual se determina si la selección de un archivo fue aceptada o cancelada. Mientras esta acción es determinada se obtiene el tipo de recurso que se está utilizando. Si el recurso pertenece al Web, se utilizan métodos exclusivos del Web para el proceso de navegación, los cuales contienen funcionalidades de soporte WebDAV definidos dentro de la clase `ResourceChoser`.

Para iniciar la navegación sobre el Web, se requiere la dirección URL del servidor remoto. El servidor debe contar con soporte WebDAV para poder realizar una autenticación de usuario y contraseña permitiendo acceso a los recursos de dicha cuenta. Todo el proceso de autenticación y verificación de los recursos remotos son procesados por la clase `ResourceChoser` (Figura 49). Cada vez que se genera una nueva conexión a un servidor y siempre que ésta sea por primera vez, los datos de la misma son almacenados en un archivo local llamado “`conexion.dat`” el cual facilita las conexiones futuras a dicho servidor, ya que almacena el usuario y la clave de acceso, ésta última si el usuario así lo desea. Una vez que se seleccionó el recurso remoto, se almacena en una variable de tipo `WebDavResource` en la cual se verifica su existencia, se obtiene el URL de dicho recurso y si éste no tiene candado se le activa uno. Se obtiene el contenido del archivo que es depositado en una cadena que a su vez es almacenada en un archivo temporal de la aplicación. Una vez realizada la operación, se obtiene el nombre del documento y se realiza una asignación del nombre actual del documento.

El siguiente paso, es determinar el elemento raíz del documento y almacenar cada elemento en el contenedor del SEIIDM, así como analizar cada uno de los elementos, verificando que coincidan con la estructura establecida en el DTD del SEIIDM y si esto se cumple, son guardados en el contenedor; de lo contrario se despliega una ventana que indica los errores de concordancia que contiene el archivo.

Para el almacenamiento del documento, se presentan dos situaciones que representan un manejo especial cuando se trata de almacenamiento remoto. Cuando es un almacenamiento realizado por primera vez, se realiza una instancia a la ventana de navegación para que permita definir el tipo de recurso a utilizar, en este caso el recurso

será remoto. Una vez seleccionada la trayectoria en la cual se desea grabar el documento se escribe el nombre que llevará, de esta forma se genera una imagen del archivo sin contenido al cual se hace referencia para continuar con la asignación del nombre al documento actual capturando el URL del recurso. Estos recursos son almacenados en un grupo de conexiones remotas que son utilizadas por un manejador de actualizaciones para optimizar la conexión al servidor. Después de verificar que el URL existe se prosigue a la generación del archivo mediante el método `getFileGenerator()` (paquete `com.ibm.xmas.editor`), al cual se le indica el nombre del archivo al que será agregada la información, donde primeramente se grabará localmente en un directorio temporal. Posteriormente será almacenado en el servidor remoto utilizando los recursos de escritura WebDAV y finalmente cerrando la conexión con el servidor cuyo URL corresponda.

Para el caso en el que el almacenamiento se haga después de una modificación, puede ser hecho con el mismo nombre o con uno diferente. Si es con el mismo nombre, se obtiene el recurso del grupo o colección de direcciones cuyo URL coincide para después llamar al método `getFileGenerator()` (`com.ibm.xmas.editor`), para que almacene la información, primeramente en la imagen temporal y para que posteriormente sea agregada al recurso remoto mediante métodos WebDAV. En el caso de que sea con un nombre distinto se siguen los mismos procedimientos que cuando se graba por primera vez.

Tomando en cuenta que la ventana de navegación también controla el acceso a la máquina local cuando el recurso es local se utiliza la trayectoria en donde se encuentra el archivo. En este caso, la autenticación no es requerida ya que se trata de la propia máquina del usuario. Después de haber seleccionado el archivo, este es almacenado en una variable del tipo `File`, la cual es manipulada con el objeto de extraer cada uno de los elementos que contiene el archivo. De igual manera, se obtiene el nombre y se asigna al documento actual. Para esta situación no se trabaja con un archivo temporal como se hizo con el recurso remoto, ya que sería un paso no necesario para la aplicación.

El siguiente paso a seguir, es determinar el elemento raíz del documento y almacenar cada elemento en el contenedor del SEIIDM, analizado antes cada uno de los elementos de tal manera que coincidan con la estructura establecida en el DTD del SEIIDM. Si se cumple con estas condiciones, se hace el almacenamiento en el contenedor,

de lo contrario, al igual que con el recurso Web, se muestra una ventana que indica los errores de concordancia que contiene el archivo.

Por otra parte, para el almacenamiento del documento, al igual que para el recurso Web, se realiza una instancia a la ventana de navegación cuando esto es hecho por primera vez, permitiendo definir el tipo de recurso a utilizar. En este caso, el recurso será local. Una vez seleccionada la trayectoria en donde será almacenado el documento, se escribe el nombre que llevará, de esta forma se genera una imagen del archivo sin contenido a la cual se hace referencia para continuar con la asignación del nombre al documento actual, capturando la trayectoria del archivo. A continuación se comienza con la generación del archivo mediante el método `getFileGenerator()` (`com.ibm.xmas.editor`), al cual se le indica el "path" absoluto del archivo al que se le agregara la información, en donde se grabara directamente el archivo, lo que no sucede con el grabado remoto que utiliza un archivo temporal.

Para el caso en el que el documento es almacenado después de una modificación, se puede usar o no el mismo nombre. Si es con el mismo nombre solamente se obtiene el nombre actual del documento y se le indica al método `getFileGenerator()` (`com.ibm.xmas.editor`), que guarde la información en el archivo con la misma trayectoria. En el caso de que sea guardado con diferente nombre se siguen los mismos procedimientos que cuando se guarda por primera vez.

Las Figuras 50, 51 y 52 muestran las diferentes formas de acceso al mismo recurso del servidor <http://azul.cicese.mx/tesisistas/victor/>: WebDAV de Windows empleando HTTP conocido como Web Folders, WebDAV del SEIIDM y mediante el Navegador IE 6.0.

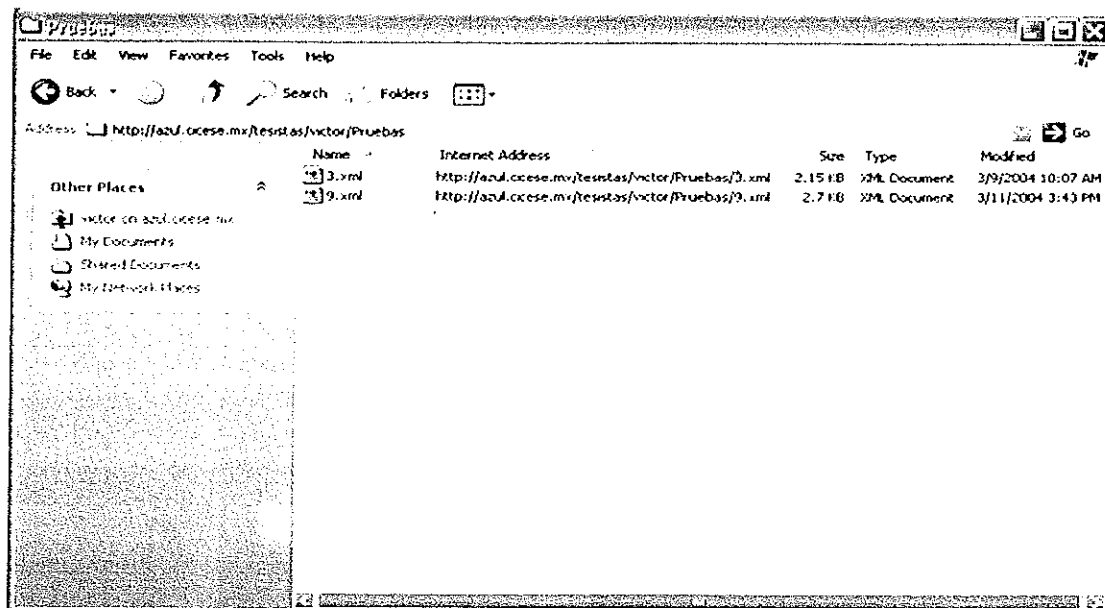


Figura 50: Navegación de Windows mediante WebDAV solicitando el contenido del directorio Pruebas.

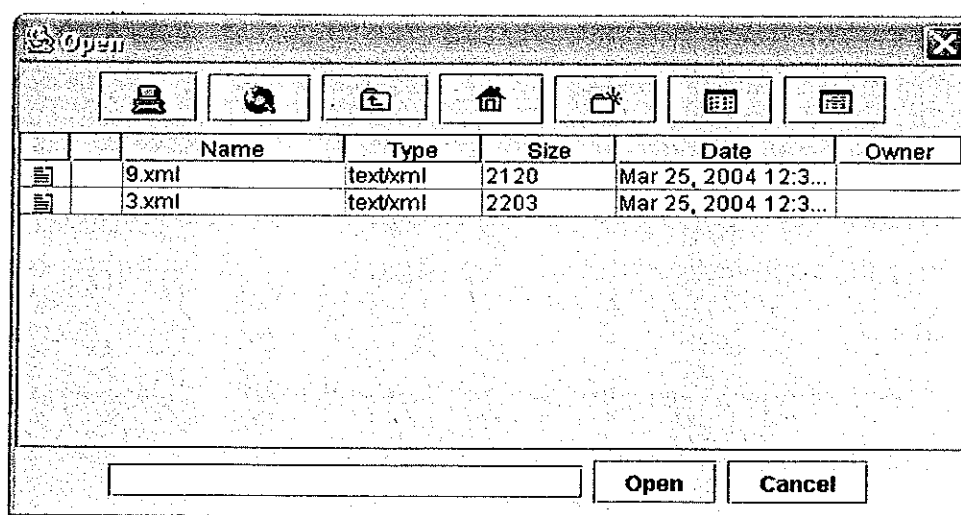


Figura 51: Ventana de navegación del SEIDM solicitando el contenido del directorio Pruebas a través del ResourceChoser.

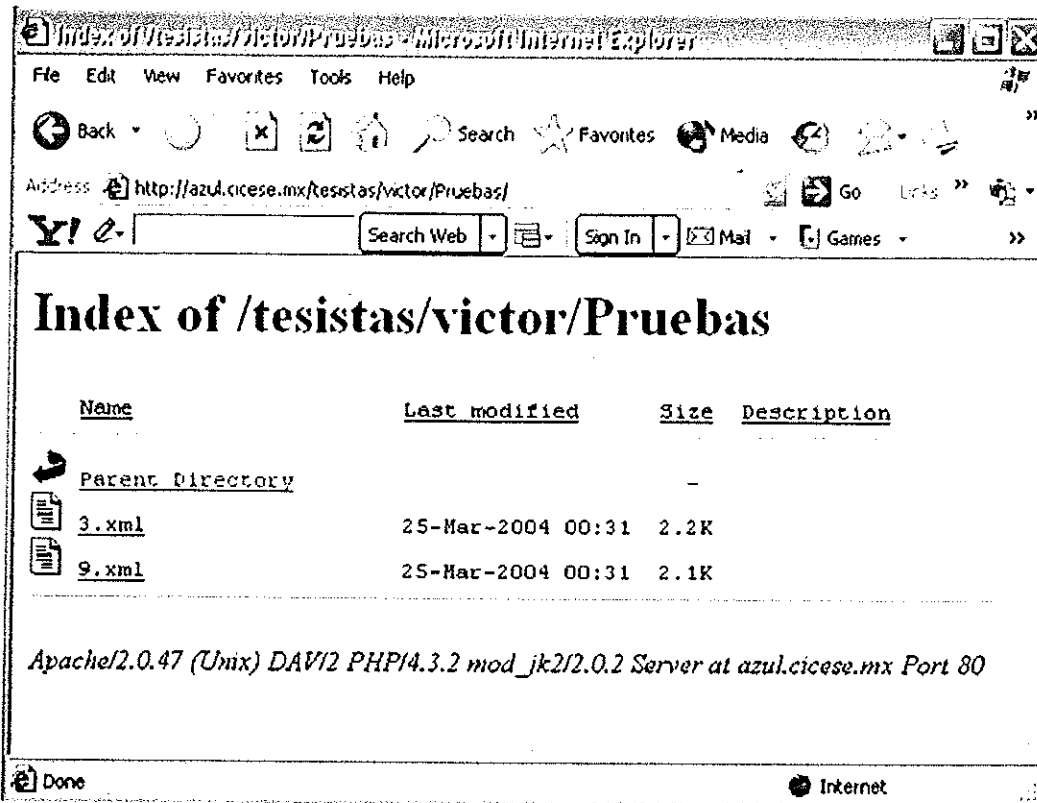


Figura 52: Ventana de Internet Explorer solicitando el contenido del directorio Pruebas a través del Web.

IV.4.4. Java Web Start

Java Web Start es la implementación de referencia de la especificación JNLP (Java Networking Launching Protocol) que define como ejecutar aplicaciones de consola escritas en Java a partir del Web.

Java Web Start revoluciona el concepto tradicional que tenemos de las aplicaciones ya que normalmente cuando se quiere ejecutar una aplicación que no se encuentra instalada en un equipo, se descarga del servidor, se instala en dicho equipo y por ultimo se ejecuta. Java Web Start intenta simplificar al máximo todo este proceso, de modo que el usuario lo único que tiene que hacer es ejecutar una aplicación con solo dar clic a un enlace de su navegador. A partir de ese momento, todo el proceso relacionado con la descarga, instalación y ejecución del programa se realiza de una manera transparente para el usuario.

A pesar de su parecido, una aplicación de Java Web Start no tiene nada que ver con un Applet. Java Web Start solo utiliza el navegador como medio para que el usuario pueda ejecutar las aplicaciones. Una vez que el usuario da click en un enlace de una aplicación, ésta se ejecuta en la máquina virtual del cliente como cualquier otra aplicación.

Java Web Start no forma parte del navegador Web, es una aplicación independiente y por tanto no requiere del navegador para su funcionamiento. Una vez que el usuario da click a un enlace para ejecutar una aplicación, puede continuar navegando o cerrar el navegador sin que esto interfiera en el funcionamiento de la aplicación que ha sido ejecutada. Además, Java Web Start va guardando en un cache interno las aplicaciones que va ejecutando el usuario, de modo que éste pueda ejecutarlas posteriormente sin la necesidad de abrir el navegador o incluso ejecutarlas localmente sin conectarse a ninguna red, ya que al ejecutar una aplicación del cache se buscan automáticamente, actualizaciones del software..

Java Web Start no es la única implementación de la especificación JNLP. Una alternativa muy popular es OpenJNLP (JNLP Group. 2002), una implementación Open Source de la especificación que está desarrollada completamente en java.

Como se mencionó anteriormente, Java Web Start revoluciona el concepto tradicional de aplicaciones. Las ventajas que ofrece tanto a los desarrolladores de las mismas, como a los usuarios son muchas y muy importantes, entre ellas podemos mencionar:

- **Transparencia:** El usuario no necesita pasar por un proceso traumático de descarga e instalación de la aplicación para poder ejecutarla. Únicamente tiene que dar clic a un enlace en su navegador y la aplicación se descarga, se instala y se ejecuta de manera automática. Además, java Web Start se encarga de crear los accesos directos correspondientes en el escritorio y menú de inicio del usuario.
- **Facilidad de Mantenimiento:** Para los desarrolladores y administradores de sistemas, Java Web Start es una bendición. Ahora ya no es necesario copiar la misma aplicación a todos los usuarios de una red cada vez que se realiza una pequeña modificación en la misma, sino con actualizar en el servidor Web es suficiente para que los usuarios puedan utilizar la última versión de la misma.
- **Control de versiones:** Java Web Start se encarga automáticamente de realizar el control de versiones de las aplicaciones. Antes de ejecutar una aplicación, Java Web Start comprueba en el servidor Web que no exista una versión mas avanzada de la misma, en cuyo caso actualiza la vieja versión automáticamente. Esto beneficia tanto a los usuarios que siempre ejecutan la última versión de su software, como a los desarrolladores que no tienen necesidad de distribuir nuevas versiones a los usuarios o crear algún sistema interno de control de versiones.

- **Independencia del servidor Web y del navegador:** Java Web Start puede funcionar en cualquier servidor Web tan sólo añadiendo el tipo MIME correspondiente a los archivos con extensión *.jnlp*. Por otra parte, también funciona en cualquier navegador aunque en algunos habrá que configurar el programa asociado a los archivos con dicha extensión.
- **Independencia del sistema operativo:** Aunque Java Web Start no está disponible para todos los sistemas operativos para los que la plataforma Java se encuentra disponible, OpenJNLP está escrito en Java, que es una iniciativa Open Source, que si es totalmente independiente del sistema operativo.
- **Automatiza la gestión de JREs:** Cada aplicación puede decidir que JRE quiere utilizar para ejecutarse, es más, si ese JRE no existe en el equipo del cliente, Java Web Start se encarga automáticamente de su descarga e instalación en el sistema.
- **Transparencia al desarrollador:** No es necesario modificar las aplicaciones existentes para que aprovechen esta tecnología. Para hacer una vieja aplicación compatible con Java Web Start, tan solo hay que crear un pequeño descriptor XML con las características de la aplicación y colocarla en un servidor Web. Las aplicaciones pueden seguirse ejecutando del modo tradicional sin ningún problema.
- **Ejecución local de las aplicaciones:** Java Web Start, a diferencia de tecnologías como JSP/Servlets no necesita la red para ejecutar las aplicaciones. La red tan solo es un medio para obtener dichas aplicaciones y sus actualizaciones. Una vez descargada una aplicación, esta se ejecuta de manera local y tan solo accede a la red si lo necesita para su funcionamiento.

Como toda tecnología, Java Web Start no está exenta de problemas:

- **Una maquina virtual por aplicación:** Este es quizás el problema más importante. La especificación JNLP establece que cada aplicación se ha de ejecutar en una maquina virtual diferente. Obviamente este es un gran obstáculo para entornos con recursos limitados y donde sea necesario ejecutar múltiples aplicaciones diferentes simultáneamente, obligando a un consumo de recursos y de memoria innecesario.
- **Problemas de flexibilidad:** Java Web Start tiene varias limitaciones de flexibilidad. No se pueden pasar algunos parámetros a la maquina virtual ya que se comprometería la seguridad y la portabilidad (ejemplo: los parámetros que comienzan con `-X`), algunas opciones solo se pueden configurar desde la computadora del usuario (como el tipo de máquina virtual a utilizar, registrar la salida, etc.)
- **No soporta los JRE 1.1 e inferiores:** Java Web Start basa su funcionamiento en el modelo de seguridad de la plataforma Java 2™ por lo que no existe soporte para versiones anteriores.

Una vez que es creada la aplicación y se comprueba que funciona correctamente en modo local, se prosigue a crear el archivo `.jar` que contendrá la aplicación. Para ello, simplemente hay que ejecutar la siguiente línea, teniendo en cuenta que la aplicación utiliza paquetes:

*Jar -cvf editor.jar com dtd images mx *.class*

El siguiente paso es la creación del descriptor JNLP. Este descriptor es un archivo XML que contiene información sobre la aplicación y sobre la forma en que la va a ejecutar Java Web Start. El descriptor de la aplicación es muy sencillo y no representa ningún problema, incluso a los lectores no familiarizados con esta tecnología, sin embargo el número de parámetros y opciones que soporta dicho descriptor es bastante grande.

El archivo JNLP que se generó es el siguiente:

```

<?xml version="1.0" encoding="UTF-8"?>

<jnlp spec="1.0+" codebase="http://azul.cicese.mx/tesistas/victor/EditorXML/" href="EditorXML.jnlp">
  <information>
    <title>Editor XML versión 2.0</title>
    <vendor>Supersabios</vendor>
    <homepage href="http://azul.cicese.mx"/>
    <description>Editor XML v2.0 , Marzo 2004</description>
    <description kind="short">
      Esta aplicación contiene los elementos necesarios para la generación de un XML
      basado en el DTD del IIDM de una manera amigable para el Usuario.
    </description>
    <icon href="images/" />
    <offline-allowed/>
  </information>
  <security> <all-permissions/> </security>
  <resources>
    <j2se version="1.4+" />
    <jar href="lib/editor2.0.jar"/> <jar href="lib/commons-httpclient.jar"/>
    <jar href="lib/jaxp.jar"/> <jar href="lib/webdav.jar"/>
    <jar href="lib/webdavlib.jar"/> <jar href="lib/editor/xmas_beans.jar"/>
    <jar href="lib/editor/xml4j.jar"/> <jar href="lib/editor/ext/ext.jar"/>
  </resources>
  <application-desc main-class="EditorXML2"> </application-desc>
  <application-desc class-path="lib/editor2.0.jar; lib/commons httpclient.jar; lib/jaxp.jar;
lib/webdav.jar; lib/webdavlib.jar; lib/editor/xmas_beans.jar; lib/editor/xml4j.jar; lib/editor/ext/ext.jar">
  </application-desc>
</jnlp>

```

Como se puede observar la mayoría de los campos son autodescriptivos. Quizás los mas interesantes sean la etiqueta <jnlp> cuyos atributos especifican donde se encuentra el archivo JNLP, la etiqueta <jar> que permite especificar los diferentes archivos que componen la aplicación, la etiqueta <j2se> cuyo atributo versión especifica cual es la

máquina virtual que se usará para ejecutar la aplicación y la etiqueta <application-desc> cuyo atributo main-class especifica cual es la clase principal de la aplicación.

Java Web Start puede ejecutar las aplicaciones en dos modos diferentes. El primero, es el modo restringido, en el cual las aplicaciones se ejecutan en un “sandbox” y solo pueden hacer uso de determinados recursos del sistema. El segundo modo, es el de confianza y es el que se aplicó al EIIDM, ya que en este modo las aplicaciones consiguen acceso a todos los recursos del sistema. Para que una aplicación pueda conseguir dicho acceso, previamente ha de tener todos los archivos jar que deben estar firmados digitalmente, de este modo, cuando el usuario ejecute la aplicación, se desplegará un certificado donde se solicita acceso no restringido al sistema, el usuario es el que decide si debe confiar o no en la fuente que emite el certificado.

El proceso de firma de los archivos jar es muy sencillo. Lo primero que hay que hacer es crear una clave de autenticación con la herramienta keytool, por ejemplo:

```
keytool -genkey -keystore MiAlmacendeLlaves -alias almacen
```

Esta herramienta pedirá información acerca de la clave y el emisor del certificado que se despliega al usuario al ejecutar la aplicación tales como: Nombre del emisor, Organización, Ciudad, Estado y País. La clave se almacenara en el almacén de claves que se especifique. Una vez creada la clave, tan solo queda firmar todos los archivos jar, en este caso:

```
jarsigner -keystore MiAlmacendeLlaves editor2.0.jar almacen
```

De esta misma forma se prosigue con los demás archivos jar que utiliza la aplicación SEIIDM. Ó bien la aplicación ya está preparada por completo, tan solo falta configurar el servidor Web añadiendo soporte para el tipo MIME para los archivos JNLP.

Capítulo V. Conclusiones y Discusión.

El SEIIDM puede ser considerado un editor de XML por que permite gracias a la simplificación de la estructura del documento acceder a cualquier nivel jerárquico del documento para editar, incorporar o eliminar nodos compatibles con el IIDM. Gracias a que los IIDM son hechos en XML se puede contar con un estándar durante su creación permitiendo la portabilidad y manipulación por quienes sigan dicho estándar.

El SEIIDM es una herramienta amigable con el usuario y que está diseñada para la creación de IIDM siguiendo estrictamente las especificaciones definidas en el DTD. Con la incorporación del soporte para WebDAV y para Java Web Start, se simplificaron tanto la administración del software, como el proceso de publicación de los IIDM en el sistema de presentación.

El SEIIDM permite el acceso a servidores WWW mediante el protocolo WebDAV para lograr la edición colaborativa de IIDM, logrando además que el almacenamiento o recuperación remota de documentos XML se haga de forma transparente para el usuario, como si se tratase del acceso a un directorio local.

En este trabajo, la reutilización de recursos juega un papel importante y se da de dos maneras distintas. Una de ellas es a nivel desarrollador al reutilizar código de proyectos como XML Master (XMas) de IBM (IBM Corp. Sun Microsystems Inc. 1999-2000) y Sistema COPPER (Hiroshi Natsu Cárdenas. 2002) desarrollado en CICESE. La segunda forma de reutilización se da a nivel autor, porque el SEIIDM permite manipular IIDM creados por terceros, para ser editados con base a las posibilidades de reutilización de estos objetos de aprendizaje. Esto es posible debido a la simplificación de la presentación de la estructura del documento, donde el autor o coautor puede acceder a cualquier nivel de la jerarquía del documento para editar, incorporar o eliminar nodos.

Se eligió trabajar con el protocolo WebDAV, ya que utiliza el puerto HTTP, para lectura y para la escritura de recursos. Así, todo el proceso de administración de documentos en el servidor, se realiza a través de un solo software, lo que simplifica la administración. Es importante recordar que este protocolo contiene mecanismos para el control de versiones mediante la incorporación de candados a los recursos; por otra parte

para poder tener acceso mediante el protocolo FTP, es necesario que el servidor tenga activado el servicio, pero muchos administradores prefieren retirar ese servicio por seguridad, debido a que este protocolo utiliza 2 puertos diferentes para realizar una transacción. Dichos puertos son utilizados para llevar acabo tanto el listado como la transferencia de información. Aunque la transferencia a través del HTTP es menos confiable que la transferencia FTP, no se puede descartar la futura incorporación del protocolo FTP al SEIIDM, aunque su efectividad dependería del nivel de seguridad implementado en el servidor.

Existen otros protocolos de transferencia que son más seguros y que utilizan la encriptación de los datos como es el caso del SSH, SFTP entre otros. Estos protocolos también podrían ser incorporados al SEIIDM en un futuro si así se requiriera sin que esto afecte el funcionamiento del SEIIDM, ya que el diseño modular que se empleó permite hacer la modificación de los mecanismos de transferencia de archivos, sin que esto afecte en ningún sentido a la estructura del editor.

Resulta interesante dominar el lenguaje UML, así como alguna herramienta de software que permita trabajar con él, pero los mejores resultados se obtienen cuando se comprende que lo más importante es seguir un buen proceso de desarrollo orientado a objetos.

Referencias.

- Apache Software Foundation. (2002). The Jakarta Project.
- Apache Software Foundation. (2002). The Jakarta Slide Project.
- AVISON, D. E., FITZGERALD, G. (1997). Information Systems Development Methodologies, Techniques and Tools. London.
- Centre for Innovation in Mathematics Teaching. (1997). Games in the classroom.
- Decouchant D. Favela J. y Martínez A. M. (2001). PIÑAS: A Middleware for Web Distributed Cooperative Authoring. The 2001 Symposium on Applications and the Internet, IEEE Computer Society and IPJ Information Processing Society of Japan. San Diego, CA, In Proc. SAINT'2001.
- Fielding R., G. J., Mogul J., Frystyk H. Berners-LeeT. (1997). Hypertext Transfer Protocol - HTTP/1.1. RFC 2068.
- Fower Martin & Scout Kendall. (1999). UML Gota a Gota, Addison Wesley Longman.
- Gorritz C. y Medina C. (2000). Engaging girls with computers through software games. Comunications of the ACM.
- Hiroshi Natsu Cárdenas. (2002). Sistema de programación en pares distribuido sobre el Web. Centro de Investigación Científica y Educación Superior de Ensenada. Ensenada B.C, México.
- IBM Corp. Sun Microsystems Inc. (1999-2000). XMas (XML Master).
- Jacobson. Ivar et.al. (1999). The Unified Software Development Process. Wesley Publishing Company.
- JNLP Group. (2002). OpenJNLP.
- Klawe M. (1998). When Does The Use Of Computer Games And Other Interactive Multimedia Software Help Students Learn Mathematics?
- López-Morteo G. López Mariscal G. e Ibarra Esquer J. (2002). Enticing learners to mathematics through a ludic collaborative learning environment. Memorias del Segundo Congreso Internacional de Educacion Abierta y a Distancia. Coleccion Innovación Educativa y Nuevas Tecnologías, ANUIES.
- López Morteo G. and López Mariscal G. (2003). An Electronic Ludic Learning Environment for Mathematics based on Learning Objects. Proceedings of ED-MEDIA 2003. World Conference on Eductaional Multimedia, Hypermedia and Telecommunications. Association for the Advancement of Computing in Education., Honolulu, Hawaii.
- López Morteo G. López Mariscal G. & Vizcarra Corral L. (2002). Una concepción lúdica del software educativo para las matemáticas. Educación Abierta y a Distancia. Experiencias y perspectivas. Coordinadores: Pérez Fragoso C. Favela Vara J. y López Mariscal G. ed. Universidad Autónoma de Baja California.
- López Morteo G. López Mariscal G. and Ibarra Esquer J. (2001). Ambiente de aprendizaje basado en instructores interactivs de diversiones matematicas. In Proceedings of International Conference on New Technologies in Science Education., Aveiro, Portugal.
- López Morteo G. y López Mariscal G. (2004). Computer support for learning mathematics: A learning environment based on recreational learning objects. Computers & Education. Elsevier.
- MCDERMID, E. M. (1990). Software Engineering for Information Systems. Maidenhead.
- Mendoza P. y Galvis A. (1998). Juegos Multiplayer: Juegos Colaborativos para la Educación. Colombia, Universidad de los Andes.

- Murray S. Mazer y Charles L. Brooks. (1998). Writing the Web While Disconnected, IEEE Personal Communications.
- Myunghye Kang. (2003). Learning Designer (TM): An e-Learning Design and Development Tool Generating SCORM Learning Objects. M. K. Doo H. Lim. Educational Technology Department Ewha Womans University Seoul, Korea.
- Papert S. (1996). The Connected Family. Bidding The Digital Generation Gap., Longstreet Press.
- Perdita Stevens y Rob Pooley. (2002). Utilización de UML en Ingeniería del Software con Objetos y Componentes., Addison Wesley.
- Presuman Roger. (1999). Ingeniería del Software- Un enfoque práctico., McGrawHill.
- Sánchez Santos J. (1998). Experimentación en compañía: el aprendizaje colaborativo en Ludomática. Colombia, Universidad de los Andes.
- Stein Greg. (2002). WebDav resources.
- Sun Microsystems Inc. (2003). The Source for JavaTM Technology.
- Whitehead E. J. (1999). El futuro del desarrollo de software distribuido en Internet.
- Whitehead E. James y Wiggins Meredith. (1998). WEBDAV: IETF Standard for Collaborative Authoring on the Web.
- Whitehead E. James. (1998). Collaborative Authoring on the Web: Introducing WebDAV, Bulletin of the American Society for Information Science.
- Whitehead E. Jim. (2002). IETF WEBDAV Working Group.

Apéndice A. Herramienta XML Master (XMas).

A.1 Requerimientos e Inicio de la aplicación XMas.

La Aplicación Xmas es una herramienta 100% desarrollada con Java, por tal motivo es indispensable contar con Java en el Sistema Operativo para poder correr la aplicación, la versión recomendable para su correcta visualización es trabajar con J2SDK 1.4.2 o superior, en cuanto al sistema operativo puede ser cualquiera siempre y cuando tenga soporte Java.

En un principio para poder ejecutar XMas es indispensable agregar al *classpath* las trayectorias de las bibliotecas que utiliza la herramienta las cuales serían agregando la siguiente línea:

```
Set classpath = .; lib\xmas.jar; lib\xmas_beans.jar; lib\xml4j.jar; lib\archiveaccessor.jar;  
lib\ext\ext.jar; %CLASSPATH%
```

Después de hacer lo anterior se ejecuta la aplicación desde una consola escribiendo el comando: *java com.ibm.xmas.XMas*. Una vez aceptado el comando se verán dos pantallas como lo muestran las Figuras 53 y 54, donde finalmente es ejecutada la aplicación quedando lista para empezar la utilización de la misma.

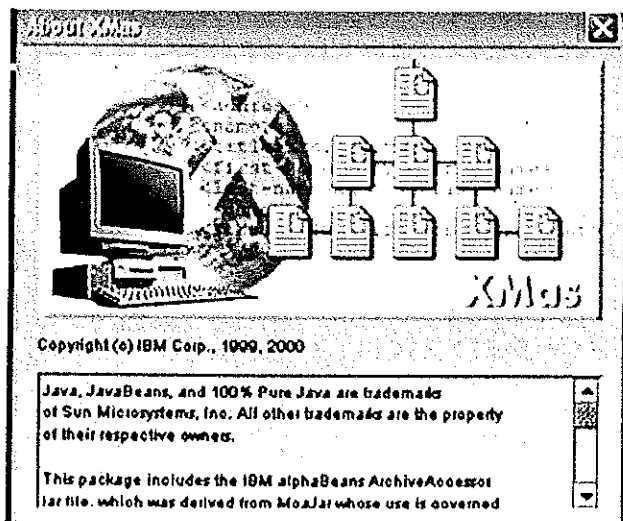


Figura 53: Ventana mostrando el cargado de la aplicación XMas.

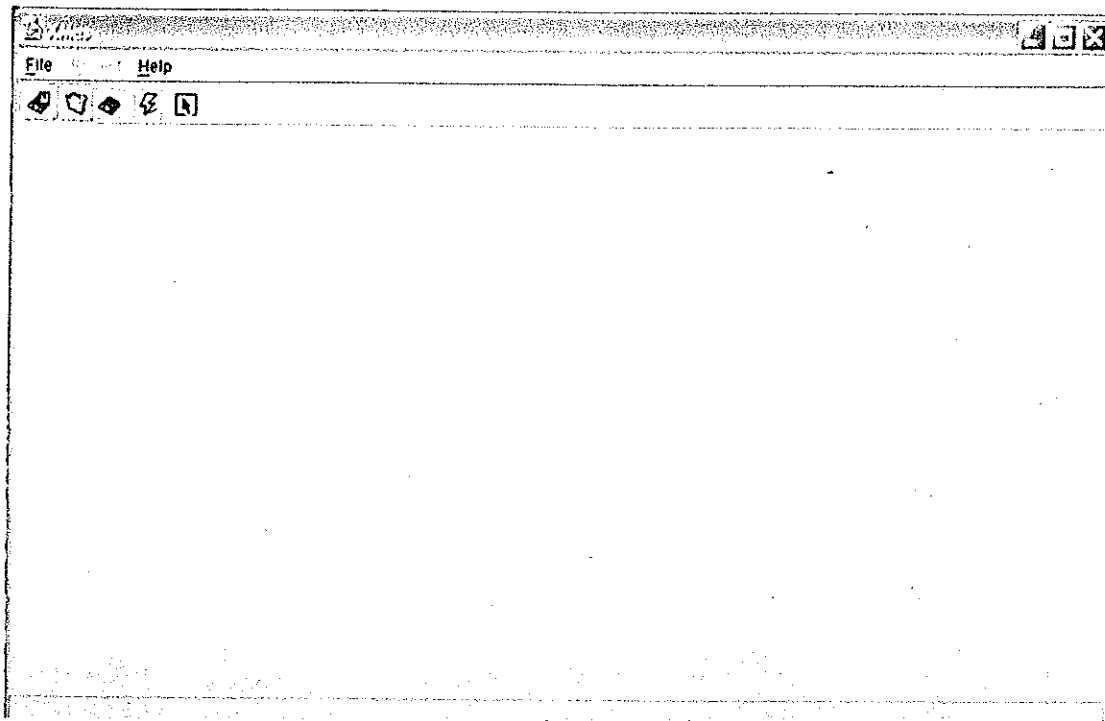


Figura 54: Aplicación XMas en ejecución.

A.2 Panel de Control y Barra de herramientas de XMas.

La barra de herramientas como es mostrada en la Figura 55, presentando una breve descripción del funcionamiento de cada uno de los elementos que la conforman.

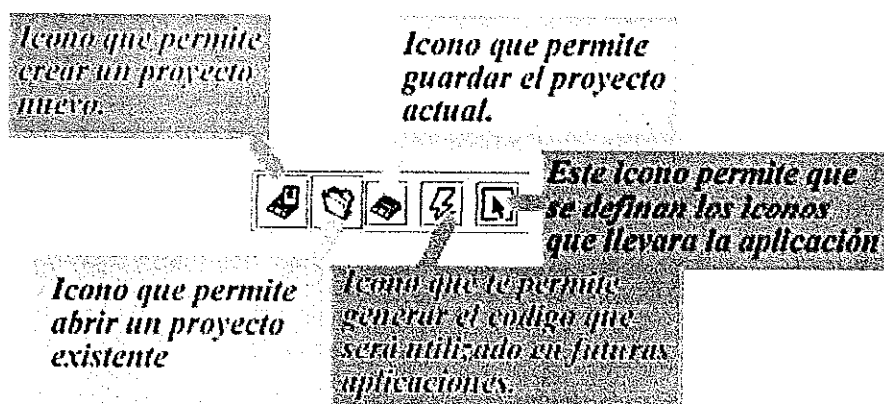


Figura 55: Barra de Herramientas de la aplicación XMas.

Por otra parte también la herramienta XMas cuenta con un panel de control el cual contiene funcionalidades que no son mostradas en la barra de herramientas, esto no quiere decir que sean de menor importancia, de lo contrario representan un rol importante durante la elaboración de un proyecto. En las Figuras 56, 57 y 58 se muestran las partes que conforman el panel de control así como una descripción de los elementos involucrados en cada módulo que se maneja.

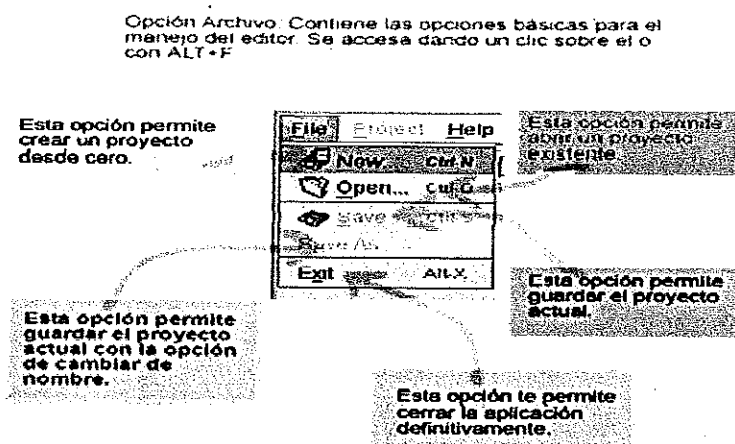


Figura 56: Menú archivo del panel de control XMas.

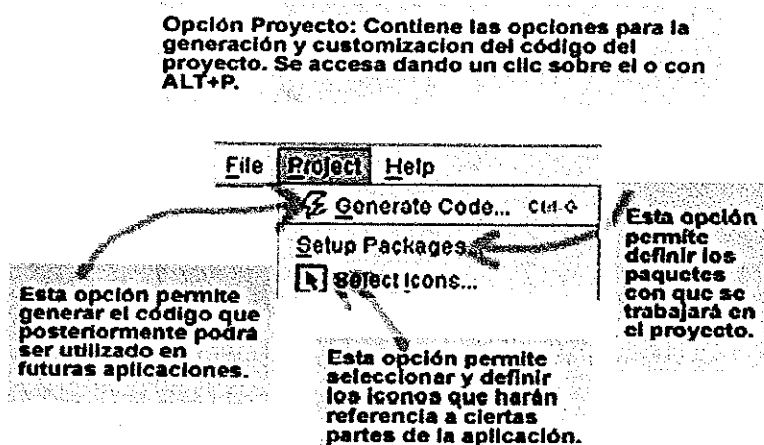
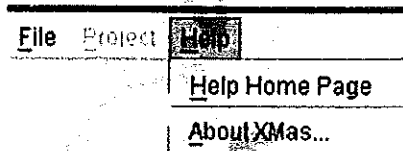


Figura 57: Menú proyecto del panel de control de XMas.

Opción Ayuda: Contiene las opciones para la presentación de ayuda y créditos de la aplicación. Se accesa dando un clic sobre el o con ALT+H.



Esta opción permite abrir la página de ayuda general de la aplicación.

Esta Opción permite abrir una ventana donde se muestran los créditos y la versión del software.

Figura 58: Menú Ayuda del panel de control de XMas.

A.3 ¿Cómo se utiliza la aplicación XMas?

La aplicación XMas una vez ejecutada, presenta la pantalla como en la Figura 54. Esta aplicación permite generar un nuevo proyecto o abrir uno ya existente. Se inicia con la creación de un nuevo proyecto, para eso ir primero a panel de control y dar click en File donde se muestra un panel emergente (Figura 56), después presionar la opción New. Esto también se puede hacer presionando las teclas CTR + N o desde la barra de herramientas presionar el primer icono (Figura 55). Al finalizar este proceso se mostrara una ventana donde se pedirá que se especifique la ubicación de un archivo DTD, con el cual se trabajara durante la creación del proyecto (Figura 59).

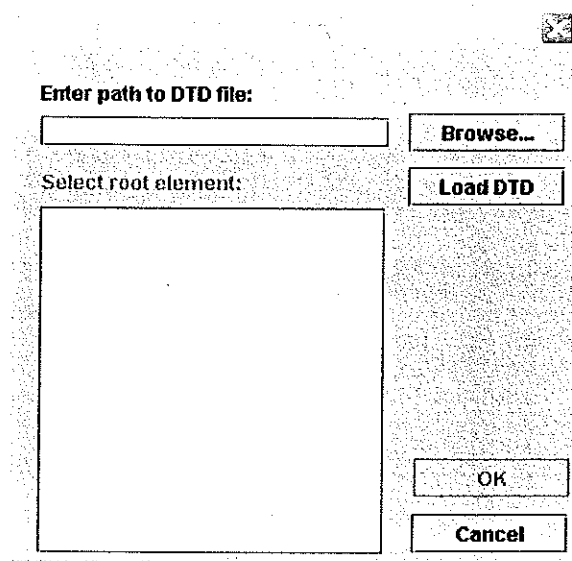


Figura 59: Ventana selectora del DTD y el Elemento Raíz.

Esta ventana te permite realizar una búsqueda en la estructura de archivos local, al presionar el botón "Browse" este realiza una llamada a una ventana de navegación para la búsqueda del archivo DTD (Figura 60). Una vez que el archivo fue encontrado se muestra en la caja de texto la trayectoria del archivo y en la parte inferior aparecerá una lista de los elemento que contiene el DTD, siendo el primer elemento de la lista el Padre el cual debe ser seleccionado e inmediatamente después presionar sobre el botón OK para iniciar la creación del proyecto (Figura 61).

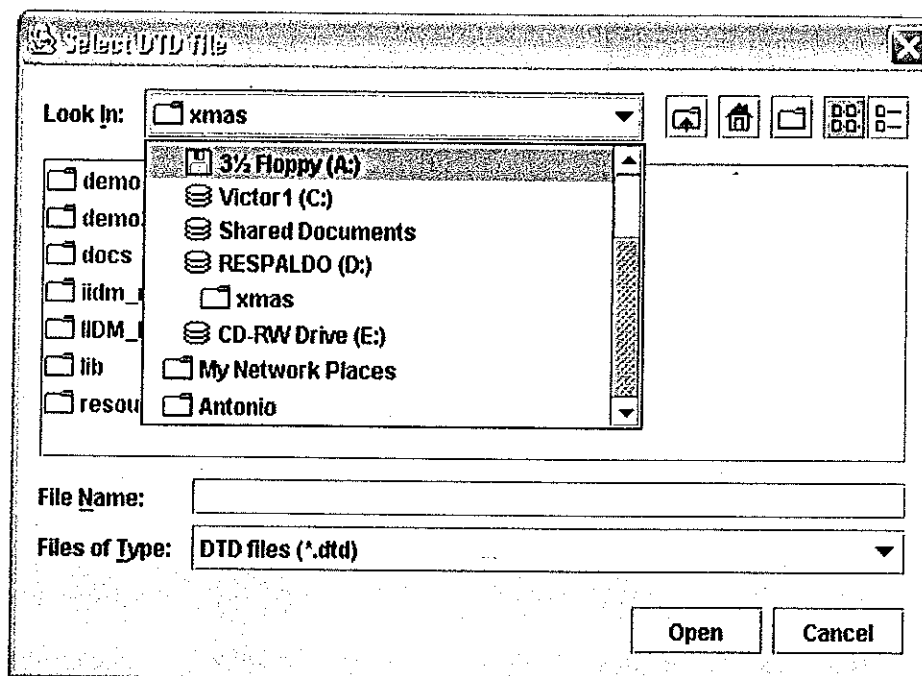


Figura 60: Ventana de navegación para realizar la búsqueda y selección del archivo DTD.

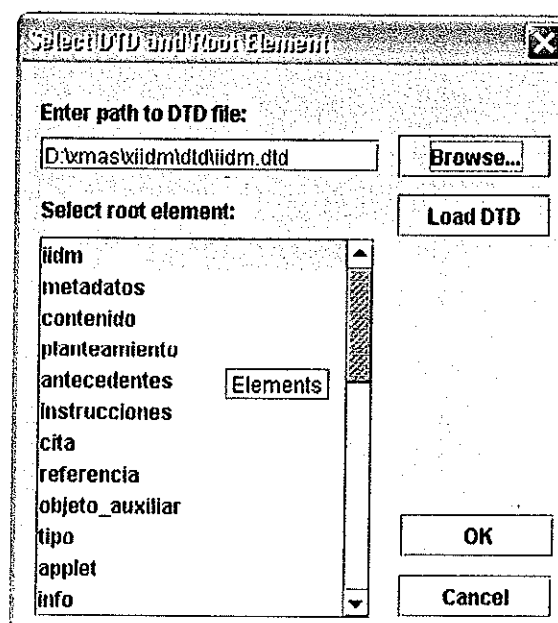


Figura 61: Ventana selectora del DTD y el elemento raíz mostrando la lista de elementos del DTD.

Después de haber concluido con éxito la selección del DTD y el elemento Raíz aparecen tres pestañas que son *Non-Visual Beans* (Figura 62), *Visual Beans* (Figura 63) y

Default Editor (Figura 64 y Figura 65). El resto de la aplicación se divide en tres secciones que son el árbol de elementos, el área del código fuente del DTD y por ultimo el área de edición.

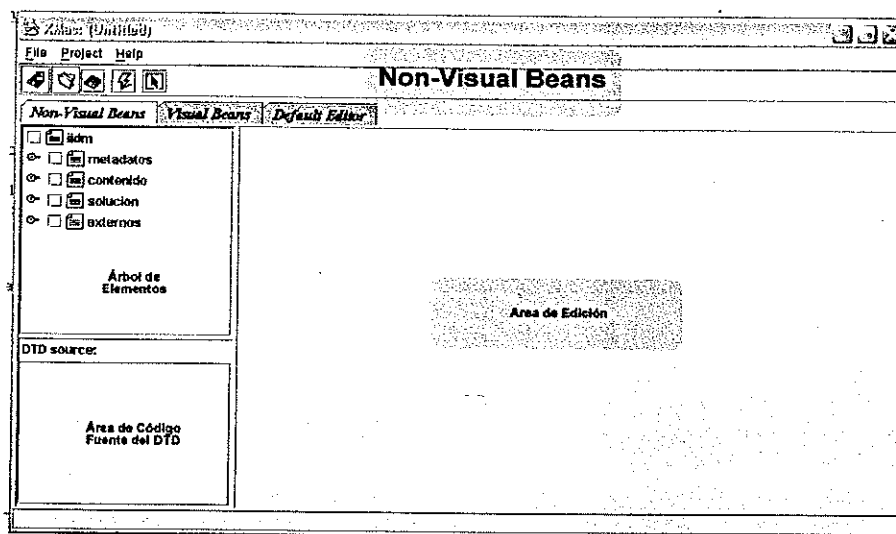


Figura 62: Aplicación XMas mostrando las tres secciones que la dividen correspondientes a la edición en Non-Visual Beans.

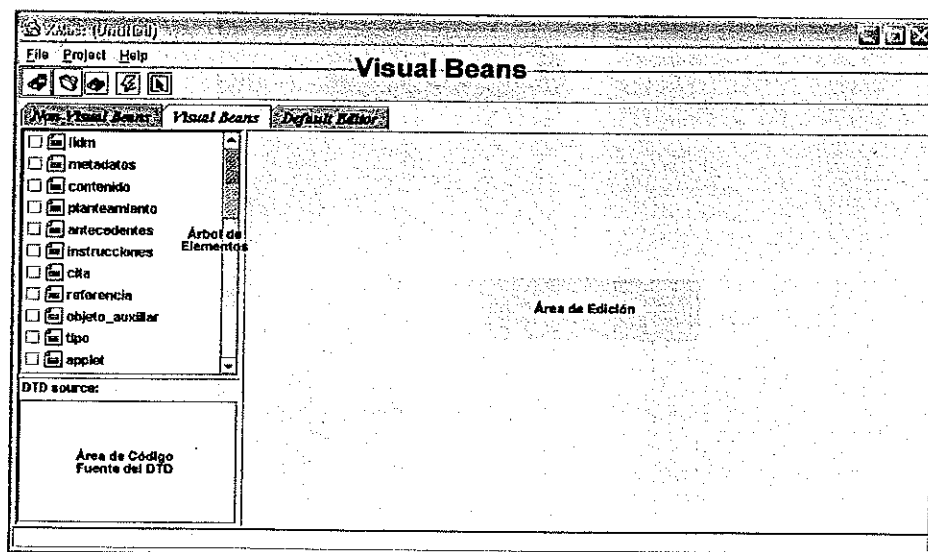


Figura 63: Aplicación XMas mostrando las tres secciones que la dividen correspondientes a la edición en Visual Beans.

En *Default Editor* se manejan dos pestañas, a diferencia con *los Non-Visual Beans* y *Visual Beans* este no maneja 3 secciones sino una extendida a todo el panel. Dentro de

esta sección hay dos pestañas. La primera es *Beans Specifics* (Figura 64), en donde se define el nombre del editor por el cual se reconocerá cuando sea llamado en un ambiente general, también se presentan cuatro campos donde se pueden definir los iconos que identificarán al editor cuando este sea cargado en un IDE. Por otra parte está *Editor Customization* (Figura 65) que presenta las opciones manuales que mostrara por omisión el editor.

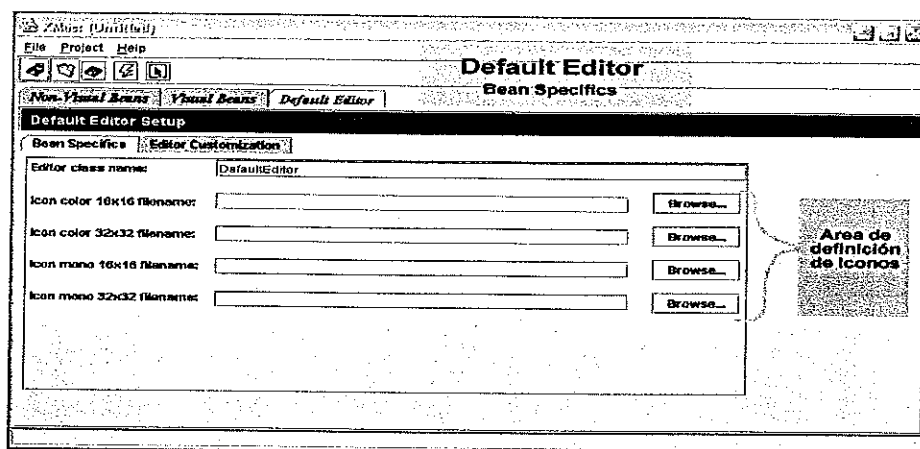


Figura 64: Aplicación XMas sección Default Editor presentando las opciones que tiene la elaboración de beans específicos.

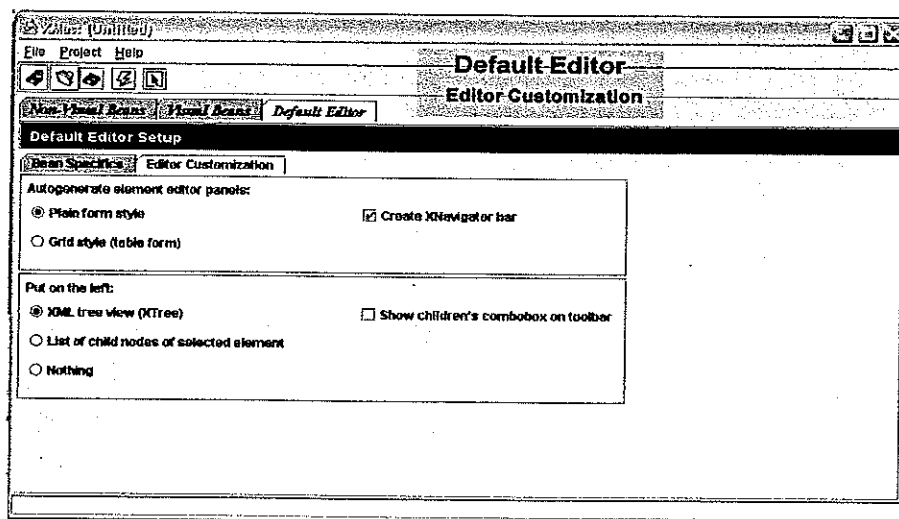


Figura 65: Aplicación XMas sección Default Editor presentando las opciones de personalización del Editor.

Después de conocer las partes que conforman la aplicación ahora empezamos con la edición. Seleccionamos la pestaña de los *Non-Visual Beans* y dentro del Árbol de Elementos (Figura 62), seleccionamos metadatos y en Área de Edición se mostraran dos pestañas que son *Bean Specifics* y *Path and Attributes*. En el Área de Código Fuente se presentan las características y atributos en XML que representan el elemento seleccionado (Figura 66).

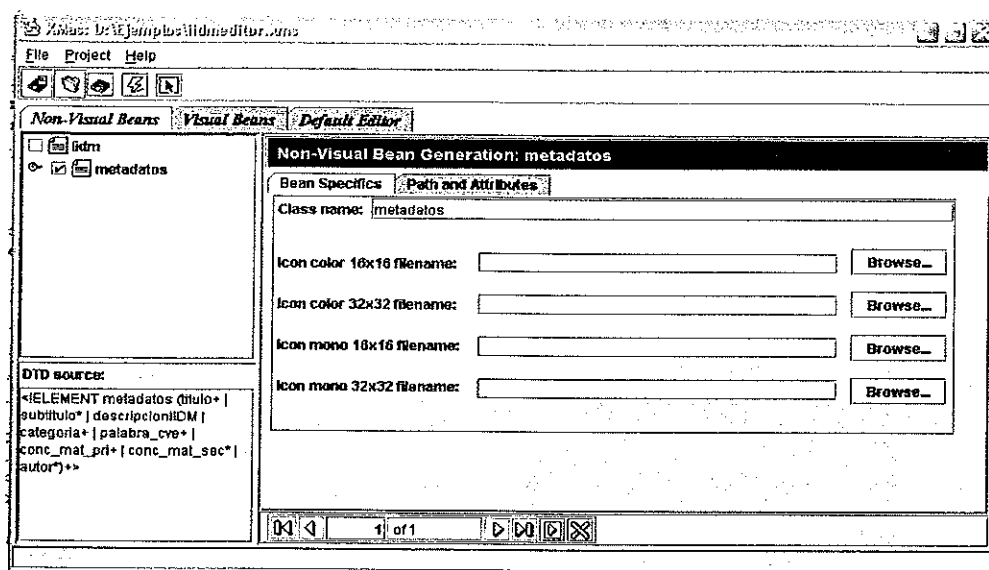


Figura 66: Creación de Non-Visual Beans mostrando las opciones de la pestaña Bean Specifics en el Área de Edición y Código XML en Área de Código Fuente proveniente del DTD.

Dentro de *Bean Specifics* se establece el nombre de la clase que identifica al elemento seleccionado y los iconos representativos al elemento. A continuación se elige la segunda pestaña que es *Path and Attributes* (Figura 67), en donde se muestran tanto el elemento padre como los elementos seleccionados para la edición, ubicándolos dentro de la cadena de inicialización. Si cualquiera de estos elementos contiene atributos específicos, al ser seleccionado dicho elemento se presentan sus atributos, los cuales pueden ser promovidos como propiedades si así se desea (Figura 67). También se puede observar en la parte inferior derecha de la ventana una barra de navegación la cual nos permite agregar mas elementos de un mismo tipo sirviendo así para poder tener mas de una interfaz de un mismo tipo.

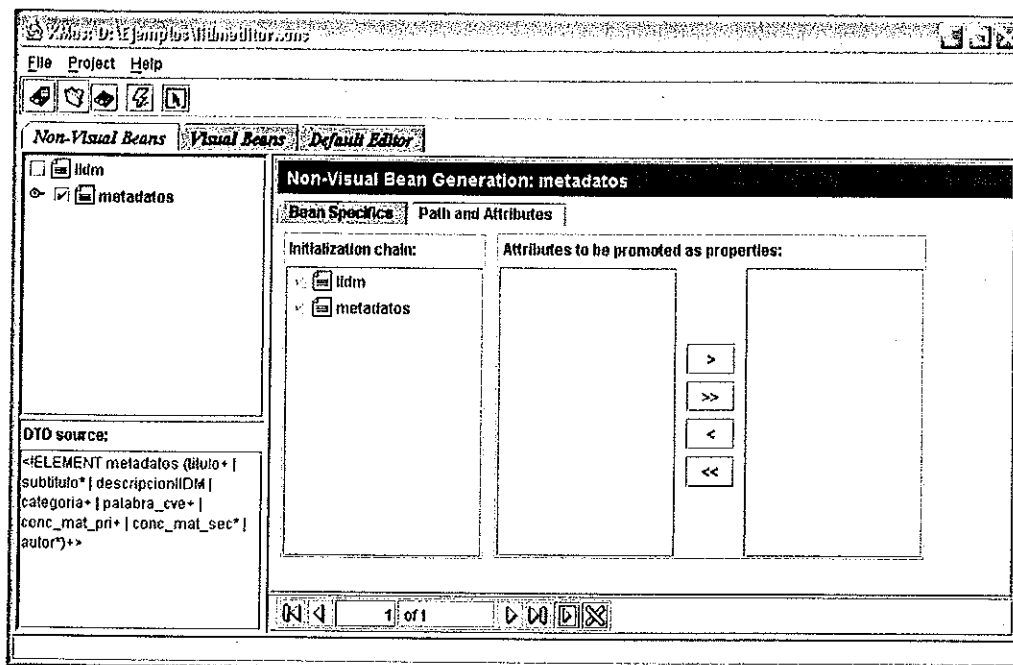


Figura 67: Creación de Non-Visual Beans mostrando las opciones de la pestaña Path and Attributes.

Ahora pasemos a seleccionar la pestaña Visual Beans, aquí se elabora la interfaz que tendrá cada elemento, esto puede ser creado en general para un elemento y sus hijos o específicamente una interfaz para cada uno de ellos. Como se puede observar anteriormente solamente seleccionamos el elemento metadatos, entonces seleccionamos en el Árbol de Elementos correspondiente a Visual Beans el elemento metadatos y nos pasamos al Área de Edición donde aparecen las opciones para la generación de un panel para el elemento del editor, dentro de estas opciones a primera vista se necesita escribir el nombre de la clase que identificara a este bean grafico, al igual que con los Non-Visual Beans se puede relacionar un icono característico a dicho elemento el cual es opcional. Otra opción que se presenta es el poder utilizar esta interfaz por omisión o no, lo recomendable es activar esa opción (Figura 68). La parte esencial concerniente a la generación de la interfaz se encuentra al presionar el botón Visual Composition, como su nombre lo indica aquí permite elaborar un panel grafico personalizado (Figura 69).

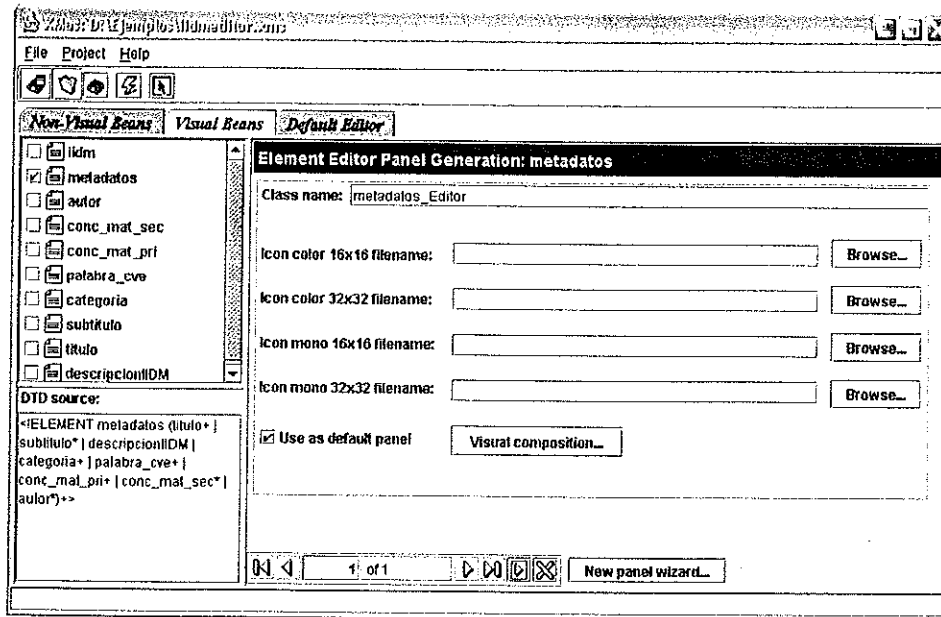


Figura 68: Opciones generares para la creación de un Visual Bean del elemento metadatos.

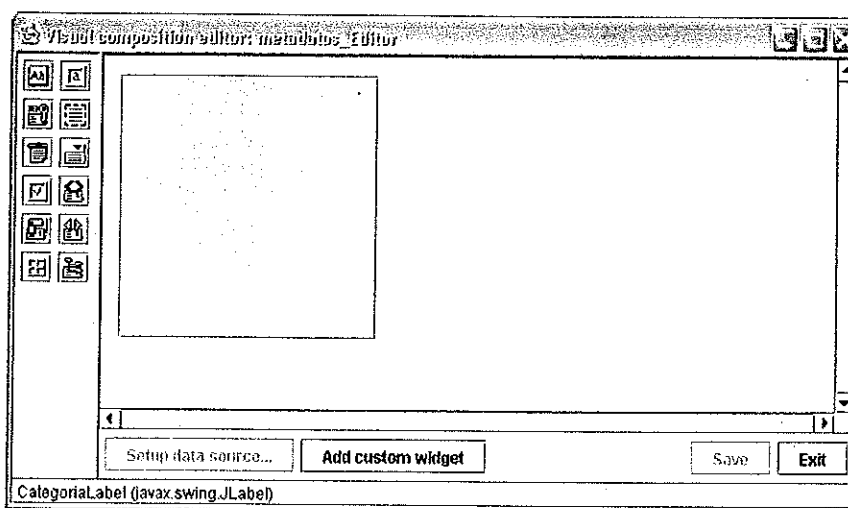


Figura 69: Editor Visual de componentes para los Visual Beans.

Antes de proseguir con la creación del panel tenemos que conocer antes el editor visual de la Figura 69, para eso en la Figuras 70 se muestra una explicación de las partes que lo conforman y en la Figura 71 se explica la barra de herramientas. El Panel de Edición Grafico es nuestra área de trabajo en donde podemos agregar los elementos que se encuentran en la barra de herramientas acorde a las necesidades. Dentro del Área de

opciones se pueden asociar las herramientas a un atributo del elemento en edición así como también se puede guardar el panel y salir cuando se requiera.

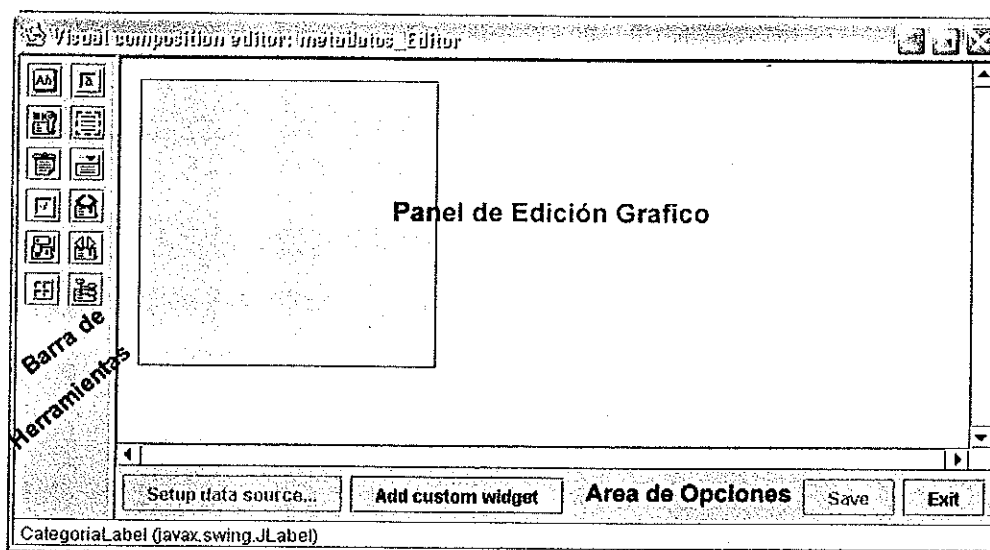


Figura 70: Partes que conforman el editor visual de componentes.

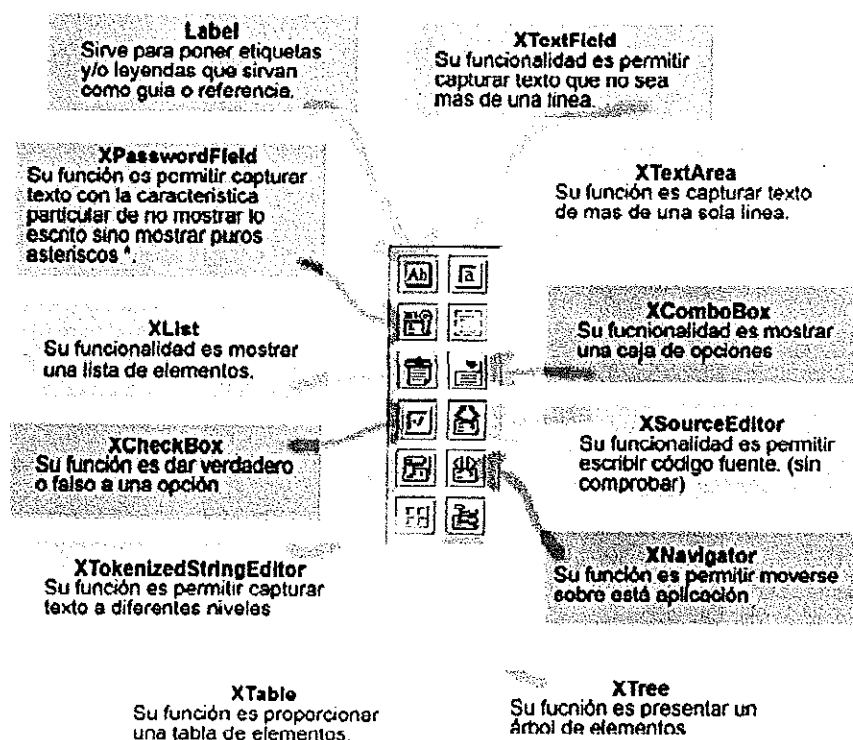


Figura 71: Barra de Herramientas del Editor Visual de componentes.

Ahora construiremos la interfaz para los metadatos utilizando el Editor Visual de Componentes (Figura 69). Primero seleccionamos de la barra de herramientas el icono Label con un clic sobre el y después sobre el panel de edición donde aparecerá el elemento seleccionado (Figura 72), damos clic sobre el label que aparece en el panel de edición con el botón derecho donde aparece una ventana de propiedades en la cual se pueden definir las características de la etiqueta, como son nombre, color de fondo, color de las letras, tipo y tamaño de la letra, Texto que será mostrado, posición inicial en X, Y y finalmente el ancho y el largo de la etiqueta (Figura 73).

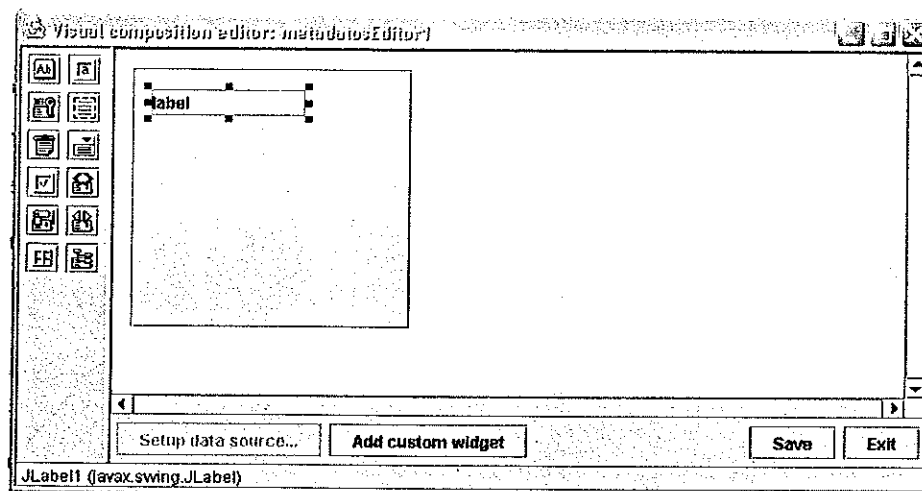


Figura 72: Panel de Edición mostrando el componente Label agregado.

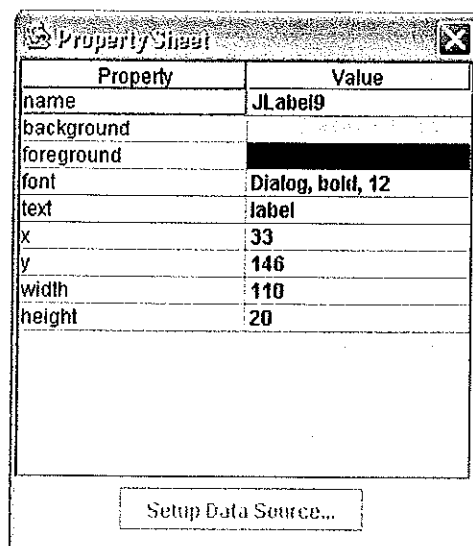


Figura 73: Ventana de propiedades del componente Label como color, tamaño, etc...

Ahora en la ventana de propiedades cambiamos el nombre de la etiqueta sustituyendo el existente por uno que identifique el uso que se le dará a la etiqueta; en este caso sustituimos “JLabel9” por “LTiulo”. Después en el campo text pondremos el nombre que se presentara en la interfaz, para eso sustituimos el “label” por “Titulo” luego cerramos la ventana presionando sobre la cruz roja que aparece en la esquina superior derecha de la ventana. Siguiendo con la elaboración de la interfaz continuamos con la incorporación de una caja de texto relacionándola con el elemento metadatos en la sección titulo, para eso de la barra de herramientas seleccionamos el componente XTextField y de la misma forma que con el Label se deposita sobre el Panel en donde aparecerá una ventana en la cual se especificara a que propiedad del elemento metadatos será ligado (Figura 74) en este caso seleccionamos titulo que finalmente será visualizado como en la Figura 75.

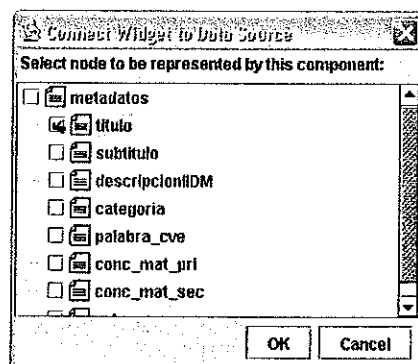


Figura 74: Ventana de selección para ligar el componente a una propiedad de elemento metadatos.

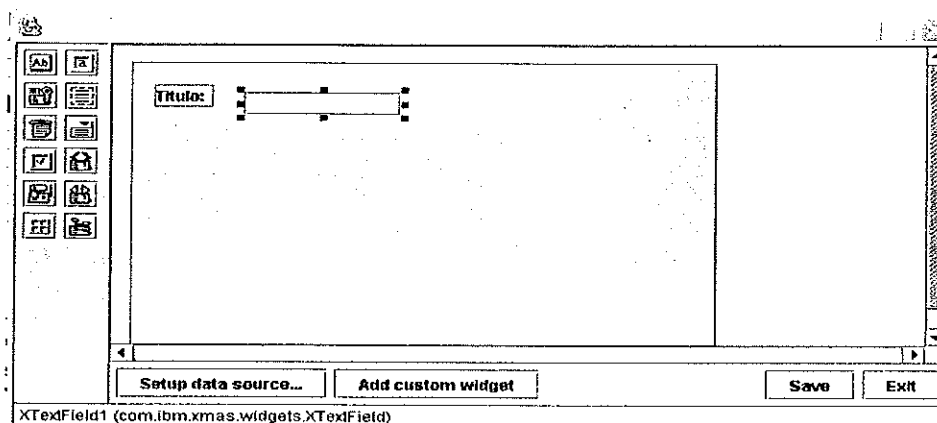


Figura 75: Panel de edición mostrando el Label y la caja de texto ligada al titulo.

De la misma forma que creamos lo de la Figura 75 seguimos con los demás elementos, que son la creación de un Label y su caja de texto ligada a una propiedad de metadatos utilizando cualquier componente de la barra de herramientas que se adecue a las necesidades, quedando como se muestra en la Figura 76.

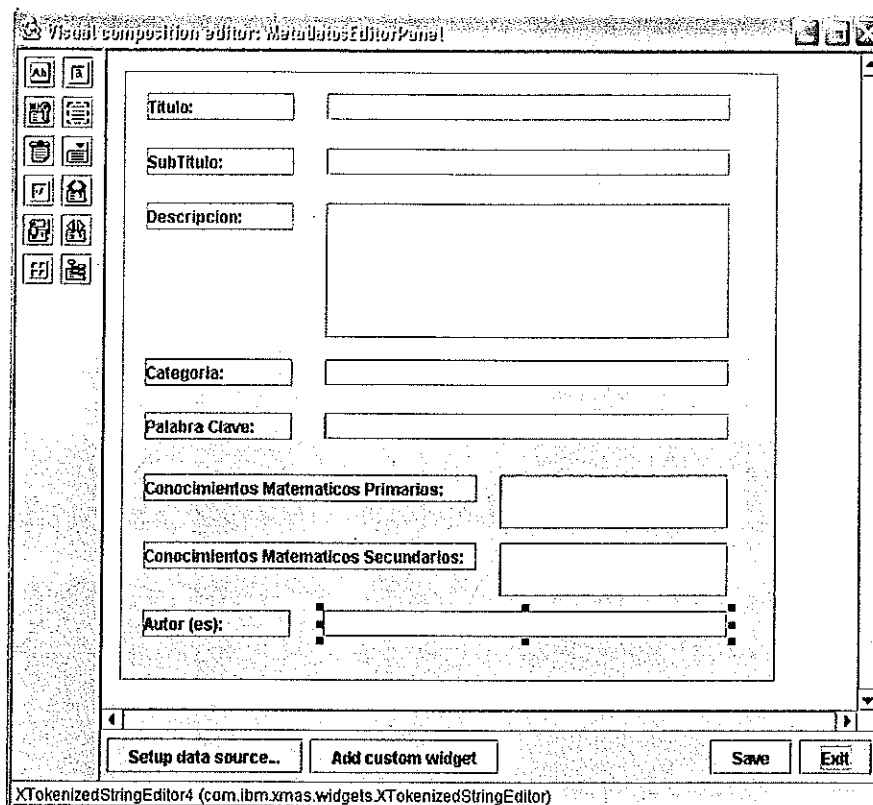


Figura 76: Panel de edición mostrando todos los elementos que debe contener la creación de la interfaz grafica de los metadatos.

Para finalizar la interfaz de los metadatos guardamos la interfaz dando clic sobre el botón “Save” que se encuentra en el Área de Opciones del panel y después sobre “Exit” para salir del panel de edición grafica y continuar la edición de otros elementos.

De una forma similar como establecimos los Non-Visual Beans y los Visual Beans correspondientes a los metadatos, seguimos con la creación de los demás elementos como son contenido, solución y externos.

Después de terminar con la creación de Non-Visual Beans y Visual Beans, debemos definir los paquetes del proyecto a cual pertenecerá, para eso seleccionamos del panel de control "Project" y después seleccionar "Setup Packages..." como se muestra en la Figura 77.

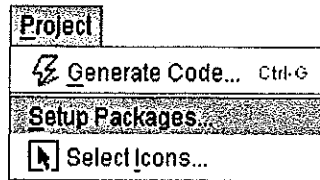


Figura 77: Menú Project mostrando la selección de Setup Packages.

Al haber realizado este procedimiento aparecerá una ventana que nos permite definir los paquetes que utilizaremos así como también definir el contenedor de la aplicación. Como se muestra en la Figura 78 iniciamos con la definición de los Non-Visual Beans, donde en primer plano tenemos que escribir el nombre del paquete de los Non-Visual Beans, después el nombre de la clase contenedora y finalmente de forma opcional podemos relacionarlos con iconos. Pasamos ahora a la siguiente pestaña donde definimos los paquetes para los Visual Beans y para el Editor por Omisión (Figura 79), en ambos casos seleccionamos el mismo paquete de los Non-Visual Beans, para eso seleccionamos los checkbox que indican que sea el mismo.

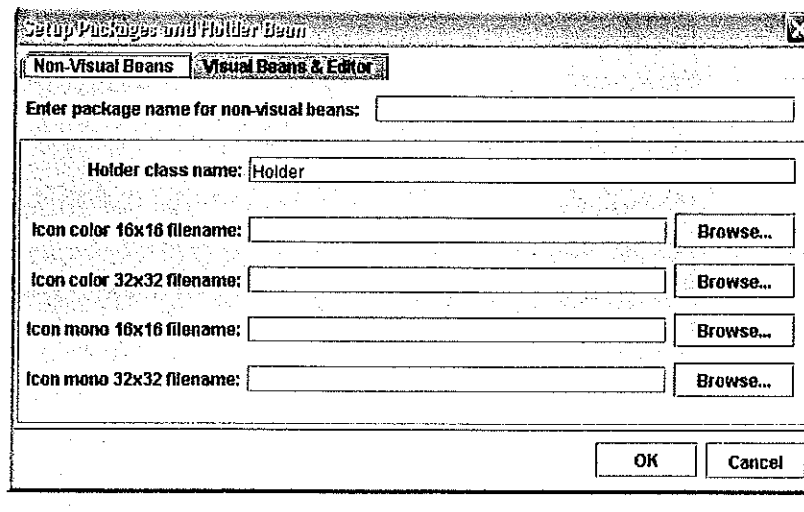


Figura 78: Ventana para especificar los paquetes y el contenedor de los Beans No Visuales, el contenedor y los Iconos.

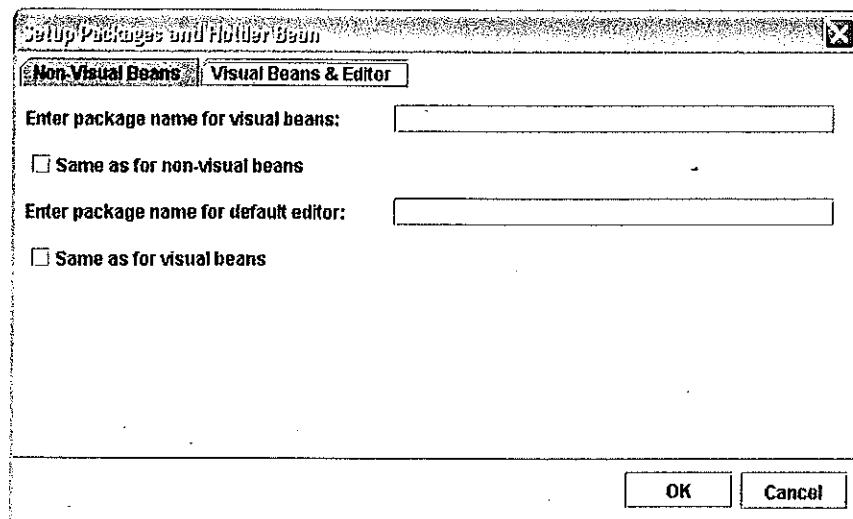


Figura 79: Ventana para especificar los paquetes de los Beans Visuales y del Editor por Omisión.

Continuamos con la generación del código fuente de los Beans que acabamos de definir, para eso seleccionamos del panel de control la opción "Project" y seleccionamos "Generate Code", aparecerá una ventana donde se especifica la trayectoria donde será depositado el código fuente y también especificamos en nombre del archivo JAR que se generará (Figura 80). Una vez especificado lo anterior y haber presionado sobre el botón OK, se abrirá una ventana (Figura 81), donde mostrara el estado durante la generación del código fuente, mostrando así los posibles errores durante su generación, casi siempre estos errores son por las limitaciones de la maquina virtual de java, al manejar muchos recursos del sistema.

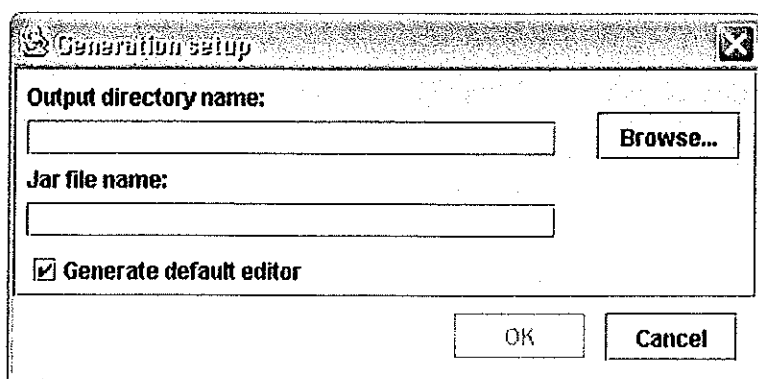


Figura 80: Ventana en la cual se definen tanto el directorio de salida del código fuente, como el nombre del archivo JAR a generar.

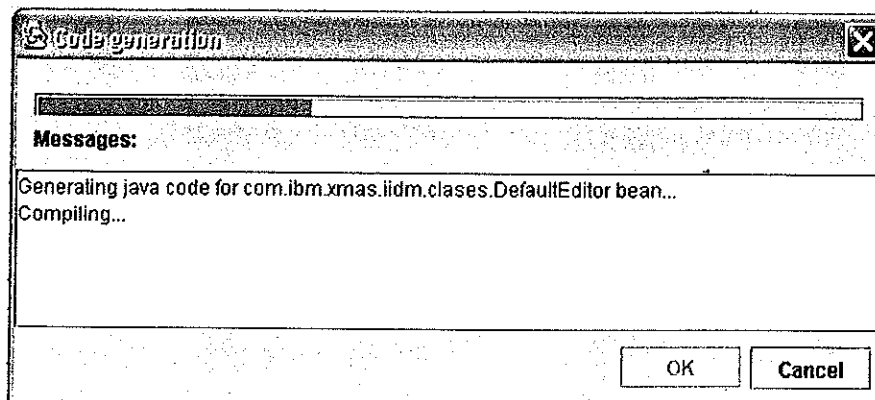


Figura 81: Ventana de Generación de código fuente para los beans visuales y no visuales.

Finalmente si se genero el .jar, este puede ser incorporado a algún ambiente IDE de desarrollo y disponer así de los beans visuales como de los no visuales. Dicha incorporación se puede realizar importando el .jar y a partir de ese momento si todo sale bien, se puede empezar a disponer de los beans en el proyecto si así lo desean.