BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

"Metodología para el Uso de Estándares Internacionales en la Creación de Objetos de Aprendizaje"

Tesis profesional presentada por

Priego Meléndez Juan Leobardo

como requisito parcial

para obtener el título de maestro en ciencias

de la Computación

Puebla, Pue. Otoño de 2006

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

Facultad de Ciencias de la Computación

"Metodología para el Uso de Estándares Internacionales en la Creación de Objetos de Aprendizaje"

Tesis profesional presentada por
Priego Meléndez Juan Leobardo
como requisito parcial
para obtener el título de maestro en ciencias
de la Computación
Jurado Calificador
Dr. Vilariño Ayala Darnes
Presidente
Dr. López y López Fabiola
Vocal y Director
Dr. Rossainz López Mario

Secretario

Puebla, Pue. Noviembre de 2006

Tesis profesional sustentada por

Priego Meléndez Juan Leobardo

como requisito parcial para obtener el título de

maestro en ciencias de la Computación.

Aceptada por el

Dr. López y López Fabiola Vocal y Director	Dr. Vilariño Ayala Darnes Presidente
Dr. Rossainz López Mario Secretario	Dr. Jefe del Departamento

23 de Noviembre de 2006

Índice general

1. Introducción **1**

1.1. Objetivos 5

2. Objetos de aprendizaje y estándares **6**

2.1. Objetos de aprendizaje 6

2.2. Uso de los objetos de aprendizaje 8

2.3. *International Organization for Standardization* ISO 15

2.4. *Aviation Industry Computer-Based Training Committee* AICC 16

2.5. *Learning Technology Standards Committee* (IEEE-LTSC) 18

2.6. IMS *Global Learning Consortium* 20

2.7. ARIADNE 21

2.8. *Advanced Distributed Learning* ADL 22

2.9. SCORM 23

3. Metadatos **29**

3.1. Metadatos 29

3.2. Categorías de Metadatos 34

3.2.1. Categoría General 34

3.2.2. Categoría Ciclo de Vida 34

3.2.3. Categoría Meta Información 35

3.2.4. Categoría Técnica 36
3.2.5. Categoría Educativa 37
3.2.6. Categoría Derechos 38
3.2.7. Categoría Relación 38
3.2.8. Categoría Anotación 39
3.2.9. Categoría Clasificación 39
3.3. Análisis comparativo entre estándares internacionales 40

4. Metodología para estandarizar Objetos de Aprendizaje 48

4.1. Estandarizando en Learning Object Metadata 49
4.1.1. Primera fase: Características generales de los OA 49
4.1.2. Segunda fase: Características del ciclo de vida de los OA 51
4.1.3. Tercera fase: Características a cerca de la información de los OA 52
4.1.4. Cuarta fase: Características técnicas de los OA 53
4.1.5. Quinta fase: Características educativas del OA 56
4.1.6. Sexta Fase: Derechos del OA 58
4.1.7. Septima Fase: Especificación de las relaciones del OA con otros OA's mas. 59
4.1.8. Octava Fase: Anotaciones respectivas al OA 60
4.1.9. Novena Fase: Clasificación del OA 61
4.2. Estandarizando en Sharable Content Object Reference Model 1.2 . . . 62
4.3. Estandarizando en Sharable Content Object Reference Model 2004 . . . 66
4.4. Estantarizando en IMS Global Learning Consortium 69
4.5. Estandarizando en DUBLIN CORE 73

5. Modelo de Análisis 77

5.1. Arquitectura General del sistema 79

5.2. Casos de Uso 82
5.2.1. Generación de metadatos 83
5.2.2. Verificador de Metadatos 87
5.3. Diagramas de Clases 90

6. Modelo de Diseño 93

6.1. Diagrama de Clases Jerárquico 93
6.1.1. Generación de Metadatos 94
6.1.2. Nivel de Interfaz 94
6.1.3. Nivel de Control y Lógica 96
6.1.4. Almacenamiento 112
6.1.5. Verificación de Metadatos 112
6.1.6. Nivel de Interfaz 113
6.1.7. Nivel de control o lógica 114
6.1.8. Almacenamiento 118
6.2. Diagramas de Secuencia 118
6.2.1. Diagrama de secuencia: Selección del Estándar 119
6.2.2. Diagrama de secuencia: Edición 121
6.2.3. Diagrama de secuencia: Generación 123

7. Implementación 130

7.1. .NET Framework 131
7.2. Visual Studio .NET 132
7.3. Comparación de .NET con Java 133
7.4. Ventajas de .NET con respecto a Java 134
7.5. Generación de Metadatos 135
7.6. Verificación de un archivo XML 145

8. Pruebas y Resultados **151**

8.1. Plan de Pruebas 152

8.2. Pruebas locales de caja negra 152

8.2.1. Prueba 11: Funcionamiento del OA con metadatos incorrectos sobre Moodle 158

8.3. Pruebas externas de caja Negra 162

8.4. Pruebas de caja Blanca 165

8.5. Pruebas de rendimiento 170

8.6. Pruebas de robustez 170

8.7. Pruebas de funcionalidad 171

A. Apendice **175**

A.1. .NET vs JAVA 175

A.1.1. Clases de marco de trabajo 175

A.1.2. Escribir una vez y depurar en todas partes 176

A.1.3. Seguridad 177

A.1.4. Control de versiones 177

A.1.5. Rendimiento 178

A.1.6. Migraciones mixtas 179

B. Formatos de pruebas realizadas **180**

B.1. Fomularios de pruebas de caja negra 180

C. Articulos y exposisiones **200**

. Bibliografía **201**

Índice de Tablas

3.1. Análisis Comparativo de la Categoría General 43
3.2. Análisis Comparativo de la Categoría Ciclo de Vida 43
3.3. Análisis Comparativo de la Categoría Meta Información 44
3.4. Análisis Comparativo de la Categoría Técnica 45
3.5. Análisis Comparativo de la Categoría Educativa 45
3.6. Análisis Comparativo de la Categoría Derechos 45
3.7. Análisis Comparativo de la Categoría Relación 46
3.8. Análisis Comparativo de la Categoría Observaciones 46
3.9. Análisis Comparativo de la Categoría Clasificación 47

Índice de figuras

2.1. Learning Management System propuesto por ADL 12
2.2. Libros de SCORM 24
2.3. Conjunto de Asset 25
2.4. Objeto de contenido intercambiable 26
2.5. Comportamiento de un objeto de contenido intercambiable SCO 28

3.1. Estructura de un archivo XML 30
3.2. Encabezado del Archivo Manifiesto 30
3.3. Cuerpo del Archivo Manifiesto 31
3.4. Organización del archivo XML respecto a los metadatos 32
3.5. Estuctura del paquete ImsManifest 32
3.6. Categorías principales del estándar LOM 33
3.7. Subdivisión de los elementos de la categoría General 34
3.8. Subdivisión de los elementos de la categoría Ciclo de Vida 35
3.9. Subdivisión de los elementos de la categoría Meta Información 36
3.10. Subdivisión de los elementos de la categoría Técnica 37
3.11. Subdivisión de los elementos de la categoría Educativa 38
3.12. Subdivisión de los elementos de la categoría Derechos 39
3.13. Subdivisión de los elementos de la categoría Relación 40

3.14. Subdivisión de los elementos de la categoría Anotación 41
3.15. Subdivisión de los elementos de la categoría Clasificación 42

4.1. Metadatos requeridos por el estándar SCORM 1.2 63
4.2. Metadatos requeridos por el estándar SCORM 2004 67
4.3. Metadatos requeridos por el estándar IMS Global Learning Consortium 70

5.1. Notación del diagrama de la arquitectura general del sistema 79
5.2. Arquitectura General del Sistema 80
5.3. Caso de uso de la fase de generación de metadatos 83
5.4. Caso de uso de la fase de Verificación de metadatos 87
5.5. Diagrama de Clases General 92

6.1. Diagrama de Clases Jerárquico. Fase Generación 95
6.2. Diagrama de Clases Jerárquico. Fase Verificación 113
6.3. Diagrama de secuencia que realiza la Selección del estándar 120
6.4. Diagrama de secuencia que realiza la Edición de los campos de la clase general 121
6.5. Diagrama de secuencia que realiza la Generación del archivo XML . . . 124
6.6. Diagrama de secuencia que realiza la iniciación del proceso de Verificación126
6.7. Diagrama de Secuencia que realiza la ejecución del proceso de verificación y su terminación 128

7.1. Arquitectura de .NET *Framework*. 132
7.2. Interfaz del proceso de generación de metadatos 136
7.3. Diagrama de actividades general para el generador 137
7.4. Fragmento de código de selección de campos de metadatos 138
7.5. Fragmento de código para generación de encabezados para metadatos . 141

7.6. Diagrama de árbol de un conjunto de etiquetas en XML 145
7.7. Interfaz del proceso de verificación de metadatos 146
7.8. Diagrama de Actividades general para el verificador 147
7.9. Fragmento de código del verificador 149

8.1. Código usado para determinar la variable de lectura 174

Resumen

En la educación a distancia los cursos no pueden ser divididos. Esto impide que sean compartidos, por lo tanto la producción es redundante y sus costos se incrementan. Para solucionar el problema de los cursos en línea indivisibles, se propone formarlos con objetos de aprendizaje (OA). Para que los OA puedan ser intercambiables y reutilizables, al ser creados deberán cumplir con los estándares internacionales tales como SCORM de ADL y LOM de la IEEE [1, 10] . Para ello se propone crear una herramienta la cual guíe al diseñador en la elaboración de metadatos para sus recursos de aprendizaje tales como animaciones, documentos de texto, páginas interactivas, etc y posteriormente, verificar que los metadatos creados cumplan con los estándares, el proceso de verificación podrá emplearse para recursos de aprendizaje externos o ya existentes para comprobar que estén estandarizados.

Al tener los OA creados de acuerdo a los estándares mencionados, se almacénarán en un repositorio o mercado donde podrán ser consultados, o incluso accedidos dependiendo de las políticas del repositorio.

Capítulo 1

Introducción

En el aprendizaje electrónico o *e-learning* los cursos están construidos como una sola entidad la cual no puede ser dividida. Esto impide compartirlos, es decir, las partes que integran un curso en línea no pueden ser compartidas para formar otros cursos; por lo tanto la producción del material didáctico es redundante y sus costos se incrementan. El que los cursos sean producidos como una sola entidad también afecta la calidad de desarrollo, así como accesibilidad, interoperabilidad, durabilidad y reutilización de los materiales y los hace dependientes de las plataformas en los que han sido desarrollados. Estos efectos negativos, lejos de beneficiar o de pasar desapercibidos afectan a la comunidad del aprendizaje electrónico. Es debido a eso, que es importante contar con medidas que disminuyan esos problemas y garanticen la calidad en la producción de contenido de aprendizaje. Si los cursos en línea fueran creados con pequeñas piezas de información digital a modo de que puedan ser unidas para generar un módulo, unidad, tema o un curso completo y que estas piezas fueran reutilizables y modulares de forma que pudieran ser intercambiables entre varios diseñadores de contenido para no volver a crear contenido con esas características se reducirían los efectos negativos que se tienen ahora.

La propuesta es crearlos en base a objetos de aprendizaje los cuales son piezas de conocimiento pequeñas y almacenadas independientemente en una base de datos, estas piezas son auto-contenidas, intercambiables, interoperables, durables y accesibles que pueden ser una imagen, una página *web*, una presentación de diapositivas, documento de texto, archivos de audio en formato MP3 o WAV, una aplicación, etc. Con ellos se reducirían los efectos mencionados, debido a que los OA brindan reutilidad e interoperabilidad, brindando consecuentemente un ahorro importante en los costos de producción y no solamente en ello, si no también, el tiempo empleado para producir contenido educativo, sumando a la lista de beneficios que ofrece la interoperabilidad y la reutilidad, se puede mencionar la calidad de contenido, accesibilidad a la pieza de información y durabilidad de esta.

Los OA proporcionan estos beneficios por que están construidos de tal manera que los hace independientes a las plataformas de cómputo existentes, así como de los sistemas operativos o LMS (*Learning Management Systems*). Pero aún existen inconvenientes al crear los OA, pues deben de ser construidos en base a estándares para poder proporcionar los beneficios mencionados y reducir los problemas que existen, de lo contrario será una opción inútil o una propuesta que no funcione. En el medio de los OA, existen algunas organizaciones que se encargan de crear especificaciones que normalizan su construcción para su correcta implementación y uso adecuado; algunas de estas organizaciones son *Aviation Industry CBT Committee (AICC)*, El *Institute of Electrical and Electronics Engineers* (IEEE), el IMS *Global Learning Consorsium*, el *Advanced Distributed Learning* (ADL) y *Alliance for Remote Instructional and Authoring and Distribution Networks for Europe* (ARIADNE) entre otros. AICC es una asociación internacional basada en la tecnología que se dedica a elaborar pautas para el desarrollo, la entrega y evaluación de tecnologías de formación profesional. El AICC cuenta con

una API (*Aplication Program Interface*)la cual le permite tener interacción al contenido con el sistema de administración de aprendizaje [5].

El IEEE por su parte, es un grupo multinacional que se encarga de desarrollar estándares internacionales para sistemas eléctricos, electrónicos, computacionales y de comunicación. Está organizado en distintos comités y uno de ellos es el comité para los Estándares de la Tecnología de Aprendizaje (LTSC por sus siglas en inglés *Learning Technology Standars Committee*). Una de sus especificaciones más conocida hace referencia a los metadatos (datos acerca de la información contenida en un OA) de los OA (LOM por sus siglas en inglés *Learning Object Metadata*); es decir, lo que intenta LTSC es facilitar el desarrollo y la reutilización de contenidos. Este organismo adoptó el trabajo del AICC [11]. El LTSC tiene más de una docena de grupos de trabajo que desarrollan especificaciones para la industria del aprendizaje electrónico:

- Grupos de trabajo relacionados con datos y metadatos
- Grupos de trabajo relacionados con los LMS y las aplicaciones

El IMS fue formado en 1997 como proyecto dentro de la iniciativa de la infraestructura del aprendizaje nacional el EDUCASE, asociación americana dedicada a la promoción de las tecnologías educativas. Por lo tanto se puede decir que IMS es un consorcio internacional formado por organizaciones educativas, comerciales y gubernamentales. Se refiere a los estándares para los servidores de aprendizaje y contenido de aprendizaje. Su objetivo principal es el desarrollo de especificaciones para facilitar la interoperabilidad entre aplicaciones de aprendizaje distribuido. Actualmente tiene definidas especificaciones en las áreas de meta- información, empresa, empaquetado de contenidos

y evaluación [12].

Por otro lado, ARIADNE es un estándar del aprendizaje electrónico que permite la descripción y clasificación de documentos y recursos educativos en base a un conjunto de metadatos con el objetivo de que puedan ser compartidos de forma organizada por distintas personas en la elaboración de sus cursos [6].

Una de las organizaciones que han tenido más aceptación en la industria mundial del aprendizaje electrónico es ADL. Esta organización es una iniciativa del gobierno de Estados Unidos para facilitar el desarrollo y la entrega de contenido didáctico con el uso de tecnologías ya existentes, así como emergentes. ADL, crea principios y guías de trabajo necesarias para el desarrollo y la implementación eficiente, efectiva y en gran escala, de formación educativa sobre nuevas tecnologías *web* [1]. Este organismo adoptó algunas de las especificaciones del IMS, IEEE LTSC y AICC, las unió y mejoró en su propio estándar: SCORM (pos sus siglas en ingleés *Sharable Content Object Model Reference*).

Las plataformas aprendizaje electrónico más modernas utilizan los estándares mencionados para lograr la interoperabilidad entre los distintos sistemas, sin embargo existe mucho contenido útil disponible bajo diversos entornos, que no está estructurado según los estándares, lo cual provoca serias dificultades para que puedan ser reutilizados. Por tal motivo se hace imprescindible la búsqueda de alternativas que permitan estandarizar dichos contenidos de modo que puedan ser importados y utilizados en las diversas plataformas existentes.

Por otro lado los expertos en contenido no tienen experiencia en el manejo de estándares internacionales para los OA ni en la creación de metadatos para los re-

cursos digitales que han sido creados. Esto, hace necesario proponer una metodología para facilitar la creación de los metadatos para los OA, basados en estándares internacionales. Hay que recordar que una metodología por definición es el análisis de los diversos procedimientos concretos que se emplean en las investigaciones y la discusión acerca de sus características, cualidades y debilidades [23].

1.1. Objetivos

El objetivo general de la Tesis es:

- Diseñar una metodología para el uso de estándares internacionales en la creación de metadatos para los OA.

Para lograrlo, se tienen metas específicas que se deben de cumplir:

- La metodología deberá conducir a un diseñador de OA para que fácilmente pueda incluir los metadatos requeridos por el estándar SCORM y LOM.
- Diseñar una aplicación para verificar que los OA cumplen con los estándares para su correcta implementación

Los resultados parciales de este trabajo han sido publicados en el I Simposio de Enseñanza de Ciencias Básicas 2006 [16], efectúado en la Universidad Autónoma de Yucatán, en la I Semana de la Computacón 2006 [18] de la Universidad Alva Edisón, en el congreso CORE, Research in Computing Science 2006 [17], en el Tercer Congreso Nacional de Ciencias de la Computación 2005 [15] realizado en la Benemérita Universidad Autónoma de Puebla y fue aceptada para participar en el Congreso Internaiconal de Tecnologías Aplicadas a la Educación Electrónica TAEE 2006 de la Universidad Politécnica de Madrid España

Capítulo 2

Objetos de aprendizaje y estándares

2.1. Objetos de aprendizaje

La popularización de la divulgación de contenidos educativos a través de Internet comienza al inicio de la década de los años 90. Poco después, en 1994, Wayne Hodgins marca una pauta importante al introducir el término Objeto de Aprendizaje. La aparición de este concepto ha suscitado desde entonces una creciente expectación, aunque no libre de polémica debido a la falta de acuerdo en las definiciones básicas adoptadas por diferentes autores. Uno de los temas más importantes en la actualidad dentro del ambito del aprendizaje electrónico tiene que ver con la idea de la reutilización para los contenidos didácticos. Es, de hecho, uno de los principios que fundamentan el concepto de objeto de aprendizaje [22] y su importancia es crucial. Es importante resaltar que la reutilización atañe principalmente al diseño de instrucción y no a los formatos digitales o a la estructura de los contenidos. Las características que determinan que un objeto de aprendizaje sea apto en una variedad de contextos educativos son esencialmente pedagógicas. Citando a Polsany [22]:

A Learning Object is an independent and self-standing unit of learning content that is predisposed to reuse in multiple instructional contexts".

Es decir que, un objeto de aprendizaje es una unidad de aprendizaje independiente y auto-contenida que esta predispuesta a reusarse en multiples contextos instruccionales. Dada su importancia, algunos autores como Sicilia [25] incluso sugieren una medida numérica con el fin de poder evaluar sistemáticamente el grado de reutilización de un objeto de aprendizaje dado. Se comprende inmediatamente el ahorro que este planteamiento supone para las instituciones educativas y productoras de contenidos. Si los OA se construyen correctamente y además se almacenan y se catalogan de una manera "inteligente", podrán ser aprovechados con más facilidad, más allá del alcance y del contexto que habrían logrado si hubieran seguido otros planteamientos tradicionales menos flexibles. Detrás de este pensamiento se esconde la filosofía que da lugar a los denominados repositorios de OA. En términos simples, los objetos de aprendizaje son unidades digitales mínimas, que pueden ser utilizadas, reutilizadas, referenciadas, auto-contenidas, intercambiables e interoperables, mediante sistemas de metadatos en aprendizaje electrónico. Estas unidades mínimas se combinan y deben funcionar dentro de un LCMS (*Learning Content Management System* ó Sistema de Gestión de Contenido de Aprendizaje).

Iriarte Navarro, Marco Such et al [13] proponen un procedimiento automatizado que permite construir bibliotecas digitales compuestas por OA a partir de estructuras de contenidos desarrolladas previamente. Por lo cual comenta que la creación de objetos educativos reutilizables no es posible de forma aislada, sino que se requiere de la creación de grupos multidisciplinarios que permitan diseñar las estrategias mas adecuadas para que el profesor tenga la capacidad de usar y construir contenidos educativos de una

forma sencilla.

Otra observación más que realizaron y que es importante resaltar es que un repositorio de OA no resuelve totalmente el problema y que es muy importante estudiar mecanismos de almacenamiento y recuperación de información de dichos objetos para que puedan ser encontrados y utilizados. Para ello es preciso utilizar procedimientos y técnicas que permitan generar objetos de aprendizaje de forma dinámica a partir de los existentes y con determinadas especificaciones del docente. Para el desarrollo de su trabajo se han basado en el estándar SCORM.

2.2. Uso de los objetos de aprendizaje

Morales G. y Agera H. [20] en 2002 tiene como interés principal el desarrollo de Sistemas Tutores Inteligentes capaces de adaptarse a los objetivos, necesidades y preferencias de aprendizaje de cada uno de sus usuarios. Siendo congruentes con las tendencias tecnológicas y las demandas del mercado, adoptaron una estrategia de investigación que, en primer término, considera el desarrollo de un sistema basado en el estándar SCORM capaz de reorganizar dinámicamente los OA que componen un curso. Donde en primera instancia se enfocaron en la representación de contenidos educativos en la forma de objetos reutilizables de aprendizaje, así como en los mecanismos de acceso a dichos contenidos para su ensamblado y presentación. Concretamente, han desarrollado un prototipo de sistema de capacitación en la *Web*, siguiendo el estándar SCORM, que puede ser usado como plataforma para investigaciones y desarrollos posteriores en la línea descrita anteriormente. Los autores comentan que la tecnología de objetos reutilizables de aprendizaje en general ha tenido poca aceptación entre la comunidad de

investigación en sistemas educativos inteligentes.

Otro aspecto importante a considerar en esa investigación es conocer la granularidad apropiada de los OA que permita flexibilidad suficiente para un manejo inteligente de los contenidos, sin comprometer ni la eficiencia del sistema, ni la calidad del contenido educativo presentado al usuario, ni la independencia, reutilidad e interoperabilidad de los OA. A más largo plazo, su investigación considera el uso de agentes inteligentes animados y no animados, como una alternativa atractiva para añadir adaptabilidad e inteligencia a un sistema educativo basado en OA.

Existen varios trabajos de investigación que nos introducen a la construcción de los OA, por ejemplo Velázquez [26] habla de la importancia de la definición de calidad del contenido de un objeto de aprendizaje, Velázquez indica que existen 3 características básicas de un objeto de aprendizaje: Accesibilidad, Reusabilidad/Adaptabilidad e Interoperabilidad. Igual que cualquier otro producto de *software*, los OA deben cumplir con ciertos lineamientos de calidad. La calidad relacionada a los objetos de aprendizaje puede abordarse desde distintos aspectos entre los cuales se puede mencionar

- La calidad en el proceso de desarrollo de los mismos,
- La calidad relacionada en la reusabilidad de los elementos granulares y
- La calidad del contenido del objeto de aprendizaje, por mencionar algunos de los aspectos.

Por otro lado, en el aspecto pedagógico, González [9] comenta la incursión de muchas empresas en el campo de la educación a distancia en las nuevas tecnologías de la co-

municación. Esta incorporación a las nuevas tecnologías provoca un cambio radical en la metodología usada. El desarrollo de contenidos es un aspecto muy importante en la integración a las nuevas tecnologías y se necesitan unas posibilidades de desarrollo que favorezcan el cumplimiento de objetivos pedagógicos

Aguilar [24] se enfoca hacia el dominio de contenidos educativos, proponiendo elementos principales que deban componer al objeto de aprendizaje. La propuesta sugiere desarrollar OA que contengan 5 elementos: Teoría, experimentación, evaluación, colaboración y relación. De esta manera, sus OA se ubican en un tipo de combinación cerrada que impide que los recursos de este objeto o conjuntos de objetos no se pueda acceder de manera individual, sino por conjunto.

La construcción de material didáctico involucra muchas actividades en las que participan varios diseñadores y receptores de contenido o aprendices. Según Sánchez [28] para una formación en línea se requieren dos procesos:

- El proceso de construcción de material didáctico de diferentes tipos y diferentes diseñadores
- El proceso de aprendizaje o acceso al material didáctico construido.

Para facilitar ambos procesos se han creado los sistemas de administración para el aprendizaje llamados LMS. El término LMS se refiere a un conjunto de funcionalidades diseñadas para entregar, medir, reportar y administrar el contenido de materiales, así como lograr la interacción entre alumnos y profesores y administrar el proceso del aprendizaje.

El término LMS puede aplicarse a un simple sistema administrador de cursos o a un ambiente distribuido complejo [4]. En SCORM, el LMS determina que material será entregado y cuando. También determina como medir el progreso y desempeño del alumno a través del contenido de aprendizaje.

SCORM apoya la noción de contenido de aprendizaje formado por objetos de pequeño contenido y reusable para formar unidades de instrucción tales como cursos, módulos, capítulos, etc. [1]

En la figura 2.1 se muestra la arquitectura de un LMS propuesta por ADL y aplicada a SCORM, [4] las flechas de la figura denotan flujo de datos entre los servicios de la arquitectura. Y aunque la especificación de cada módulo no se proporciona, en base a nuestra experiencia podríamos describirlas de la siguiente forma:

- La *instancia API* es una interfaz mediante la cual el cliente tendrá acceso al LMS desde un buscador.
- El *servicio de medición* lleva un registro del desempeño del alumno según sus evaluaciones, su perfil y la secuencia que lleva en sus cursos.
- Mientras tanto, el *servicio de perfil del alumno* proporciona el perfil en relación a los cursos que ha tomado.
- Quien se encarga de la evaluación del alumno y en base a eso también puede dar información para crear su perfil es el *servicio de Administración de cursos.*
- El *servicio de evaluación* pone a prueba al alumno dependiendo de las actividades que ha realizado.

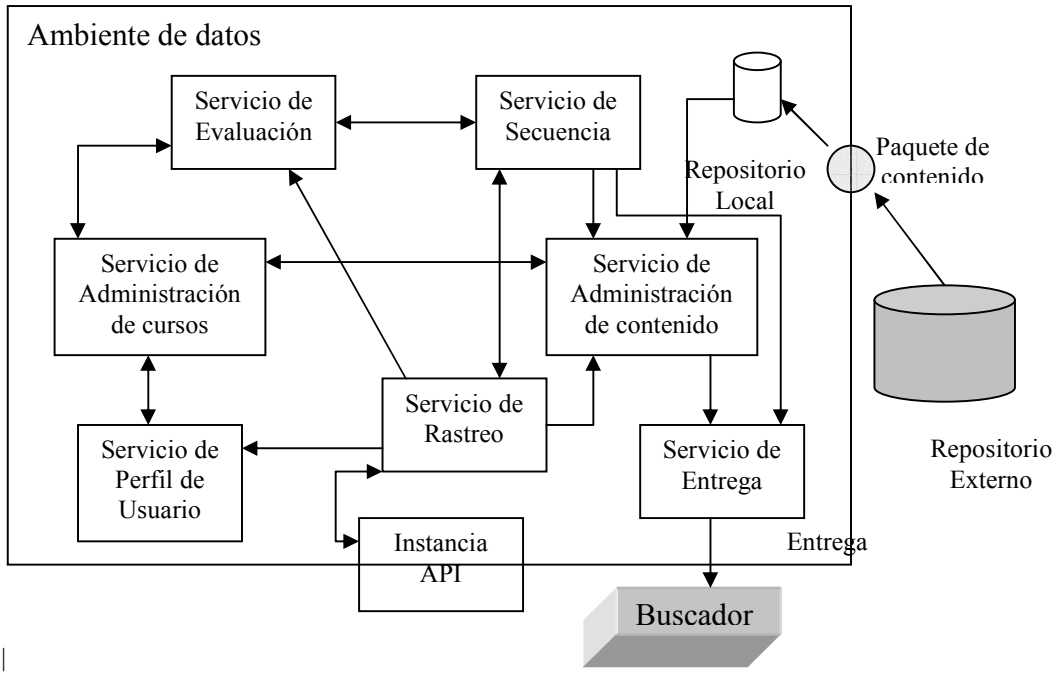

Figura 2.1: Learning Management System propuesto por ADL

- El *servicio de secuencia* se encarga de decidir que contenidos puede acceder el alumno.
- Por otro lado el s*ervicio de administración de contenido* proporciona el acceso al material al que tiene derecho.
- El *servicio de entrega* es el que tiene contacto con el buscador del cliente para proporcionar información del tipo que sea solicitado.
- En cuanto a los *repositorios locales* almacenan material didáctico. Así, quien reúne varios objetos de contenido de repositorios externos para poderlos almacenar en los repositorios internos es el paquete de contenido.
- Los *repositorios externos*, contienen material didáctico de aprendizaje ajeno al LMS.

Gracias a Internet y a las TIC se tiene una evolución rápida en herramientas que facilitan el desarrollo y administración de cursos en línea. Así han surgido organizaciones que comenzaron a trabajar en la elaboración de estándares y especificaciones para el diseño de entornos tecnológicos en lo que se refiere al proceso de enseñanza-aprendizaje con nuevas tecnologías.

Los estándares son acuerdos internacionales o normas establecidas por consenso mundial que contienen las especificaciones técnicas y de calidad que deben reunir productos y servicios para cumplir satisfactoriamente con las necesidades para las que han sido creados y para poder competir internacionalmente en condiciones de igualdad.

En las actuales soluciones del aprendizaje electrónico los contenidos preparados para un sistema no se pueden transferir de forma sencilla a otro. Los estándares en aprendizaje electrónico son el vehículo a través del cual será posible dotar de flexibilidad a estas soluciones. Lo que se persigue con los estándares en el aprendizaje electrónico, es lo siguiente:

- Durabilidad: referida a evitar la obsolescencia de los cursos
- Interoperabilidad: Intercambio de información entre los diferentes LMS
- Reutilidad: hace referencia a que los distintos cursos y OA puedan reutilizarse en diferentes herramientas y distintos LMS.

En el ámbito donde varias empresas compiten en el mercado de la formación en línea, (por un lado creando plataformas, por otro contenido y servicios para integrar en

estas), han aparecido instituciones que pretenden reglamentar una normativa que defina las especificaciones estándares a las cuales deberían ceñirse los entornos de formación. Su propósito no es otro que el de adoptar especificaciones que permitan que contenidos y entornos de aprendizaje distribuido de múltiples autores puedan trabajar juntos.

En este ámbito se utilizan los estándares como el creado por la *Aviation Industry Computer-Based Training Comité* (AICC), el *Learning Object Metadata* (LOM) creado por IEEE y *Learning Technology Standars Committee* (IEEELTSC) y el IMS *Global Learning Consortium*, más aun, el que se está imponiendo es el estándar SCORM desarrollado por *Advanced Distributed Learning* (ADL).

Las plataformas de aprendizaje electrónico más modernas utilizan los estándares mencionados para lograr la interoperabilidad entre los distintos sistemas. Sin embargo existe mucho contenido útil disponible bajo diversos entornos, que no está estructurado según los estándares, lo cual provoca serias dificultades para que puedan ser reutilizados. En contraste algunas de las ventajas que se obtienen en la estandarización son las siguientes:

- Incrementar la cantidad y la calidad de los contenidos
- Posibilitar el intercambio de cursos (compra-venta)
- Personalizar los contenidos y reutilizarlos.
- Asegurar la compatibilidad con diferentes plataformas.
- Garantizar el intercambio de contenidos entre diferentes entornos virtuales de formación.

- Permitir la búsqueda de contenidos por toda la red.
- Fomentar la profesionalización en la elaboración de contenidos.
- Iniciar sistemas de compra-venta de contenidos.
- Aumentar la eficiencia de los contenidos en línea desarrollados, además de facilitar su gestión.
- Garantizar la viabilidad futura de la inversión en el aprendizaje electrónico de una empresa, lo que impide que la empresa no dependa de un único LMS.
- Aumentar la oferta de cursos disponibles en el mercado
- Los productos no quedarán obsoletos a corto plazo

Los distintos estándares que se desarrollan en la actualidad para la industria del aprendizaje electrónico, suelen hacer referencia al:

- Contenido: Estructura del contenido, empaquetamiento del contenido, etc.
- Alumno: Almacenamiento e intercambio de información del alumno, habilidades del alumno, privacidad y seguridad, etc.
- Interoperabilidad: Integración de componentes en un LMS, interoperabilidad entre diferentes LMS, etc.

2.3. *International Organization for Standardization* ISO

En 1947 inicia sus actividades la ISO (*International Organization for Standarization*) [14]. Desde su creación, este organismo internacional no gubernamental ha si-

do el responsable de crear, emitir y certificar todas las normas internacionales de estandarización.

En el campo educativo, la evolución de las Tecnologías de la Información y de la Comunicación y el crecimiento acelerado de Internet en general y del aprendizaje electrónico en particular, han creado la necesidad de que se regulen también sus productos y servicios. Así han surgido organizaciones que comenzaron a trabajar en la elaboración de estándares y especificaciones para el diseño de entornos tecnológicos en lo concerniente al proceso de Enseñanza-Aprendizaje con nuevas tecnologías y en red. Al hablar de estándares en el aprendizaje electrónico hacemos referencia a un conjunto de directivas comunes para las empresas del sector.

Estas reglas especifican a los desarrolladores de tecnología la manera de como deben los construir las plataformas, los contenidos, etc.

2.4. *Aviation Industry Computer-Based Training Committee* AICC

El Comité de Formación Basado en Computadoras de la Industria de la Aviación (por sus siglas en inglés *Aviation Industry Computer-Based Training Committee)* [5], es una asociación internacional basada en la tecnología que se dedica a elaborar pautas para el desarrollo, la entrega y la evaluación de tecnologías de formación profesional. Se compone de los siguientes subcomités

- Comunicación

- Sistemas de bibliotecas electrónicas digitales (DELS)
- Pruebas independientes
- Administración y procesos
- Infraestructura de enseñanza
- Tecnología de la enseñanza

Las publicaciones del AICC referentes a las especificaciones se denominan guías y recomendaciones AICC (AICC AGRs). Estas especificaciones cubren las siguientes áreas:

- AGR 001: Publicaciones AICC
- AGR 002: Estaciones que entregan cursos en línea
- AGR 003: Audio Digital
- AGR 004: Sistema Operativo Windows
- AGR 005: Dispositivos Periféricos CBT
- AGR 006: Instrucción de administración de cómputo
- AGR 007: Intercambio de Cursos en Línea
- AGR 008: Video Digital
- AGR 009: Estándar ICO: Interfaz de Usuario
- AGR 010: Instrucción de Computo-Administración basada en Web

La guía más utilizada es la AGR 010, esta guía hace referencia a la interoperabilidad de plataformas de formación y los cursos resolviendo los problemas de comunicación entre los cursos y el LMS también trata con algunos aspectos para poder subir a un LMS un curso desarrollado por terceros. AICC dispone de un *Test Suit* que permite a las empresas verificar que sus productos cumplen las especificaciones que elabora este organismo otorgándoles certificados [3].

2.5. *Learning Technology Standards Committee* (IEEE-LTSC)

El IEEE (por sus siglas en inglés I*nstitute of Electrical and Electronics Enineers*) [10], es un grupo multinacional que desarrolla estándares internacionales para sistemas eléctricos, electrónicos, computacionales y de comunicación. Está organizado en distintos comités y uno de ellos es el comité para los Estándares de la Tecnología de Aprendizaje (LTSC por sus siglas en inglés *Learning Technology Standars Committee*). Una de sus especificaciones más conocida hace referencia a los metadatos (datos acerca de la información contenida en un objeto de aprendizaje) de los OA (LOM por sus siglas en inglés *Learning Object Metadata*); es decir, lo que intenta LTSC es facilitar el desarrollo y la reutilización de contenidos. Este organismo adoptó el trabajo del AICC. El LTSC tiene más de una docena de grupos de trabajo que desarrollan especificaciones para la industria del aprendizaje electrónico:

- Grupos de trabajo dedicados a actividades generales
    - IEEE 1484.1 Modelo de referencia y arquitectura

- IEEE 1484.3 Glosario

- Grupos de trabajo relacionados con datos y metadatos
  - IEEE 1484.12 Metadatos de OA
  - IEEE 1484.14 Semántica y cambio de compromisos
  - IEEE 1484.15 Protocolos de intercambio de datos

- Grupos de trabajo relacionados con los LMS y las aplicaciones
  - IEEE 1484.11 Instrucción administrativa de cómputo
  - IEEE 1484.18 Plataformas y perfiles media
  - IEEE 1484.20 Definiciones de competencia

Para este estándar, los OA son definidos como una entidad digital o no digital, esto puede ser usado para aprender o enseñar. El LTSC trabaja de forma coordinada con el subcomité de la ISO JTC1SC36 (formado por la ISO y el *International Electrotechnical Commission*) dedicado a la normalización en el ámbito de las Tecnologías de la Información para la educación y el aprendizaje. En este estándar, un metadato para un objeto de aprendizaje describe características para aplicaciones con OA. El propósito de este estándar multi-parte es el facilitar la búsqueda, evaluación, adquisición y uso de los OA para instancias por alumnos, instituciones o procesos de software autónomos. Este estándar multi-parte también facilita las compariciones e intercambio objetos de aprendizaje [12].

2.6. IMS *Global Learning Consortium*

El IMS *global consortium* [12], fue formado en 1997 como proyecto dentro de la iniciativa de la infraestructura del aprendizaje nacional el EDUCASE, asociación americana dedicada a la promoción de las tecnologías educativas; por lo tanto se puede decir que IMS es un consorcio internacional formado por organizaciones educativas, comerciales y gubernamentales. Se refiere a los estándares para los servidores de aprendizaje y contenido de aprendizaje.

Su objetivo principal es el desarrollo de especificaciones para facilitar la interoperabilidad entre aplicaciones de aprendizaje distribuido. Actualmente tiene definidas especificaciones en las áreas de meta-información, empresa, empaquetado de contenidos y evaluación.

Las especificaciones IMS cubren un amplio rango de características que intentan hacerse interoperables entre plataformas que van desde los metadatos, la interoperabilidad en el diseño de instrucción entre plataformas, hasta la creación de cursos en línea para alumnos que tengan alguna discapacidad visual, auditiva, etc. El IMS recoge el trabajo de la IEEE. Su objetivo es la creación de especificaciones que engloben las recomendaciones de la propia IEEE y de la AICC [5] [10]. Elabora las siguientes especificaciones:

- Especificaciones usadas para describir, descubrir e intercambiar contenidos
    - Metadatos
    - Empaquetamiento de contenidos
    - Preguntas y Test
    - Repositorios digitales

- Especificaciones para la interacción con el contenido y el seguimiento
  - Secuencia simple
  - Competencias
  - Diseño del aprendizaje
  - Accesibilidad

- Especificaciones para la interoperabilidad de las aplicaciones
  - Información del alumno
  - Empresa

2.7. ARIADNE

La Alianza para las Redes de Instrucción y de Derechos de Autor a Distancia (Por sus siglas en inglés *Alliance for Remote Instructional and Authoring and Distribution Networks for Europe*) [6], es un estándar del aprendizaje electrónico que permite la descripción y clasificación de documentos y recursos educativos en base a un conjunto de metadatos con el objetivo de que puedan ser compartidos de forma organizada por distintas personas en la elaboración de sus cursos.

La misión de ARIADNE es la de proporcionar mejor calidad de aprendizaje a través de desarrollo de OA, herramientas, protocolos y metodologías que brindan la posibilidad de compartir y reusar recursos para la educación y la enseñanza. [4].

2.8. *Advanced Distributed Learning* ADL

Una de las organizaciones que han tenido más aceptación en la industria mundial del aprendizaje electrónico es ADL (*Advanced Distributed Learning*). Es una iniciativa del gobierno de Estados Unidos para facilitar el desarrollo y la entrega de contenido didáctico con el uso de tecnologías ya existentes, así como emergentes.

ADL, crea principios y guías de trabajo necesarias para el desarrollo y la implementación eficiente, efectiva y en gran escala, de formación educativa sobre nuevas tecnologías *web*. Este organismo adoptó algunas de las especificaciones del IMS, IEEE LTSC y AICC, las unió y mejoró en su propio estándar: SCORM. De lo recogido de las distintas organizaciones se destacan los siguientes elementos:

- El sistema de descripción de cursos en XML (Xtensible Markup Languagede) la IMS (*content packaging*).
- El mecanismo de intercambio de información mediante una API (*Aplication Program Interface*) de la AICC.
- Los objetos de metadatos provistos por el LOM (*Learning Objects Metadata*) de la IEEE.

SCORM divide al mundo de la tecnología del aprendizaje electrónico en dos componentes funcionales. Los principales componentes son: LMS y SCO. Este último se refiere a los objetos de aprendizaje. Otros componentes en el modelo SCORM son herramientas que permiten crear los SCOs y los ensamblan en unidades de aprendizaje

más grandes (un curso por ejemplo) [2].

2.9. SCORM

El SCORM se divide en tres libros los cuales se ilustran en la figura 2.2, El libro 1, SCORM Overview, contiene un resumen de la iniciativa de ADL, la razón de SCORM y un sumario de especificaciones técnicas y guías.

El libro 2, Modelo de Agregación de Contenido (CAM por sus siglas en inglés *Content Agregation Model*), contiene la guía que sirve para identificar y agregar recursos dentro de estructuras de contenido de aprendizaje. Este libro describe una nomenclatura para el contenido de aprendizaje y el paquete de contenido que hace referencia al IMS. El libro 3, Ambiente de tiempo real (RTE por sus siglas en ingles *Run Time Environment*), incluye una guía para lanzar comunicación en un ambiente basado en web. Este libro es derivado de la funcionalidad definida en el Ambiente del tiempo real del AICC [3].

En el libro 2 (CAM) se hace referencia a los SCO describiendo los componentes usados para construir contenidos de aprendizaje. En este libro se comprenden las definiciones de:

- Assets
- SCO

Los Assets, son la forma básica de un recurso de aprendizaje, son representaciones

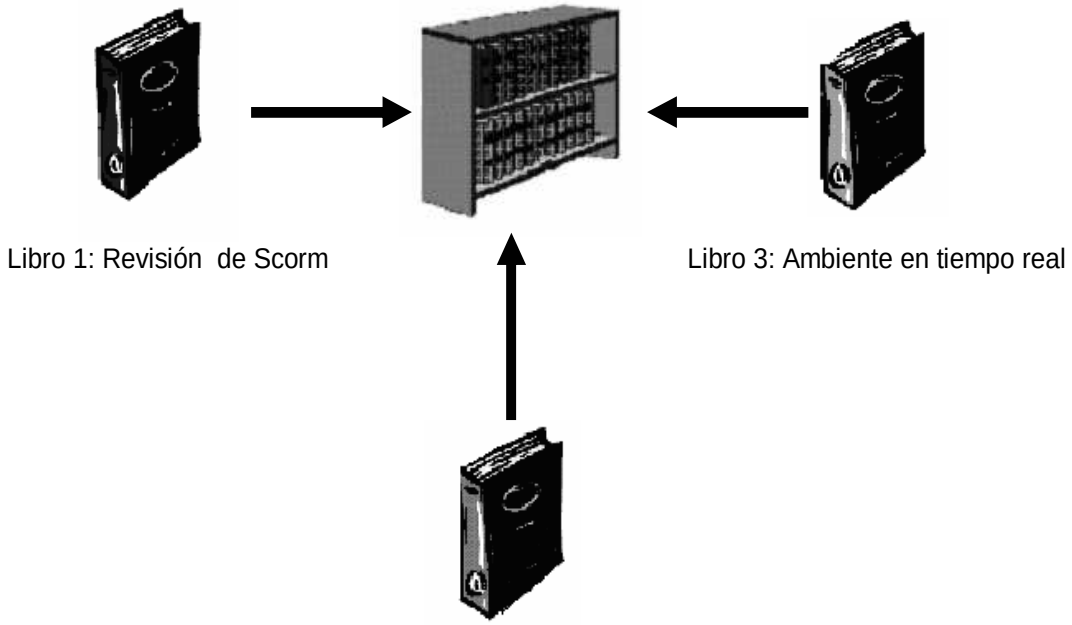

Figura 2.2: Libros de SCORM

electrónicas, tales como texto, imágenes, sonidos y cualquier otra pieza de datos. En la figura 2.3 se muestra la representación de varios Assets juntos. Un Asset puede ser descrito con metadatos para asegurar la facilidad de ser encontrados.

En SCORM un SCO es una colección de uno o más Assets que representan una actividad de aprendizaje. Un Asset representa el nivel más bajo de granuralidad en un recurso OA. La única diferencia entre el Asset y el SCO es que el SCO se comunica con el LMS. En la figura 4 se muestra un ejemplo de un SCO formado por varios Assets.

Para proveer reusabilidad al SCO, este debe de ser independiente del contexto. Por ejemplo un SCO puede ser reutilizado en diferentes cursos de aprendizaje para lograr diferentes objetivos [2].

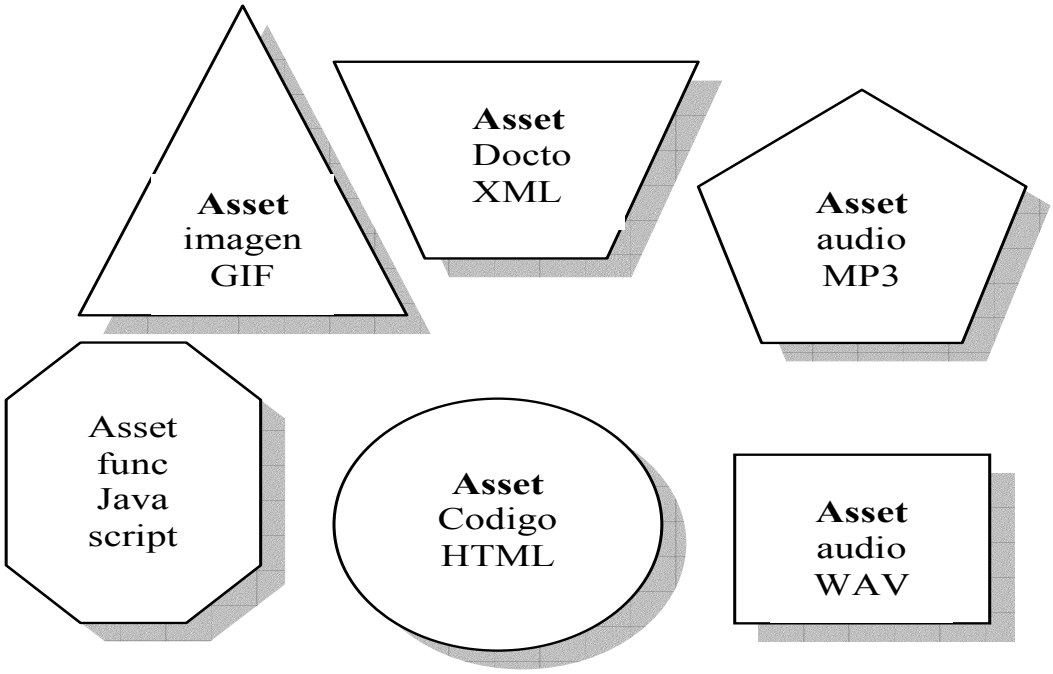

Figura 2.3: Conjunto de Asset

Un SCO es la unidad más pequeña dentro del contexto de aprendizaje que tiene comunicación con el LMS, el tamaño exacto del SCO no esta restringido por SCORM. Un SCO puede ser descrito en base a metadatos, los cuales son información de lo que contiene el SCO, los metadatos sirven para facilitar su descubrimiento en las búsquedas dentro de los repositorios de OA. Un metadato se encuentra en un archivo XML, llamado archivo manifiesto el cual se compone por:

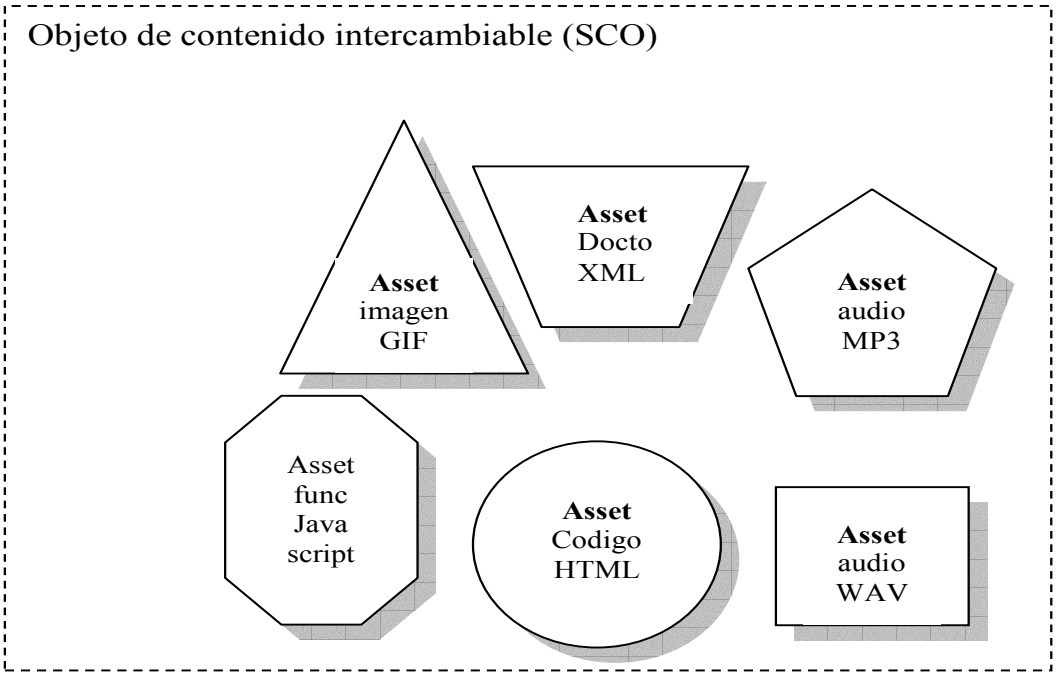

Figura 2.4: Objeto de contenido intercambiable

- Metadatos
- Recursos que ocupan el SCO
- Una organización en la que se encuentran los recursos del SCO

Para construir un objeto de aprendizaje, la propuesta básica es que los contenidos de formación deben de construirse a partir de los SCO. Para que un SCO se considere como un objeto de aprendizaje debe de reunir las siguientes características:

- Reutilizables
- Interoperables

- Duraderos
- Accesibles

Cada SCO es independiente de cualquier otro por lo que no deben de incluir en su interior ninguna referencia a la estructura de otro SCO o curso. La aparición del concepto de los objetos de aprendizaje es anterior a SCORM y es una de las ideas más prometedoras en el mundo del aprendizaje electrónico. Un SCO debe inicializar y terminar su comunicación con el LMS.

Básicamente un SCO es un contenido para ser usado en *web* que se comunica con un LMS después de haber sido lanzado. Es decir, un SCO puede ser una página *web* con un pequeño *script*.

Cuando un SCO es cargado, el *script* encuentra al adaptador API y llama a la función *LMSInitialize* del mismo. Cuando el SCO es descargado, el script llama a la función *LMSFinish* del adaptador API. La figura 2.5 muestra el comportamiento básico de un SCO.

1. Debe de encontrar la instancia API provista por el LMS
2. Usa la instancia API para inicializar la comunicación del SCO con el LMS
3. Opcionalmente, usa la instancia API para conseguir valores
4. Usa la instancia API para terminar la comunicación con el LMS

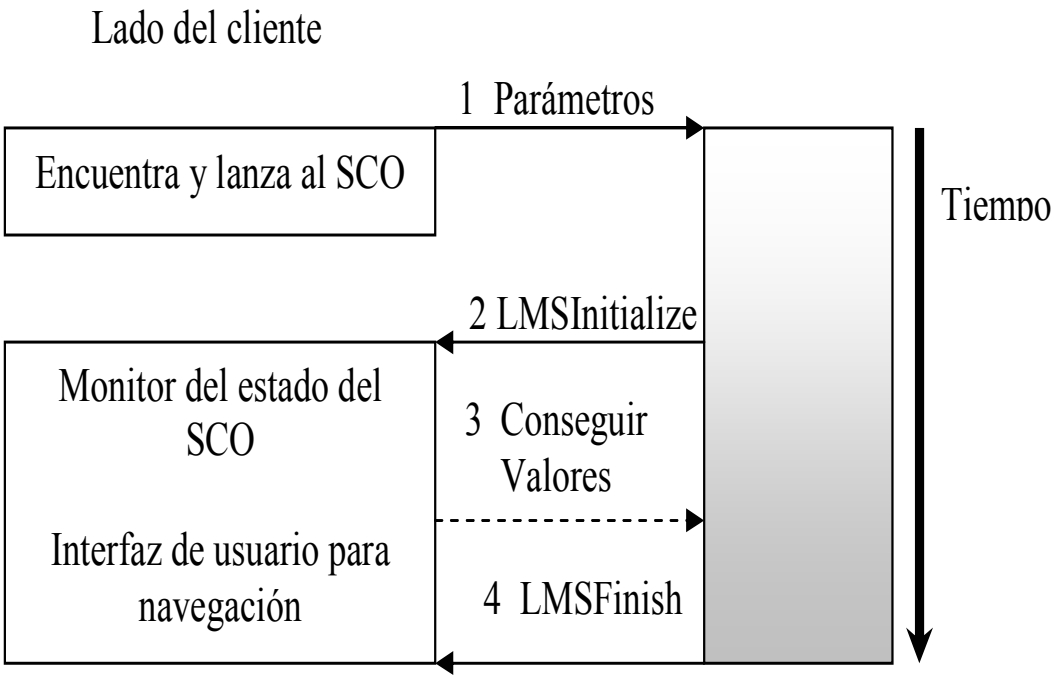

Figura 2.5: Comportamiento de un objeto de contenido intercambiable SCO

Los SCO pueden organizarse en actividades o paquetes. Un mapa que representa la manera de usar contenido a través de unidades de instrucción se le conoce como la organización de contenido. Las actividades dentro del la organización de contenido pueden consistir de otras actividades a su vez o no.

En cuyo caso, se pueden asociar a un recurso, (SCO o Asset). Una actividad en SCORM puede contener más de un SCO para formar una unidad de instrucción o tratar de satisfacer objetivos de aprendizaje de alto nivel. Trabajando en conjunto estos SCO pueden formar cursos completos, unidades, capítulos, etc.

Capítulo 3

Metadatos

3.1. Metadatos

Literalmente los metadatos para los OA son datos a cerca de los OA, que informan a los usuarios sobre las características de los OA existentes de modo que los usuarios sean capaces de entender lo que representan y cómo lo representan. Para ello la información incluida en los metadatos describe: la fecha de los datos, el contenido, la extensión que cubren, el sistema de referencia espacial, el modelo de representación espacial de los datos, su distribución, restricciones de seguridad y legales, frecuencia de actualización, calidad, etc. Los metadatos de un OA son una parte esencial en la administración de un conjunto de ellos, tanto en un repositorio donde soĺo se almacénan o en un mercado donde se realizan transacciones. De igual manera facilitan su búsqueda y localización de los objetos de aprendizaje. La selección de los metadatos que se incorporan a los OA es una decisión importante porque va a determinar su eficaz localización y en consecuencia su capacidad de reutilización por otras personas. Para la construcción de los metadatos se utiliza el lenguaje XML (Xtensible Markup Language) y para su correcta definición existen diversas organizaciones que se encargan de estandarizar la creación de estos elementos.

La estructura del archivo XML se representa en la figura 3.1

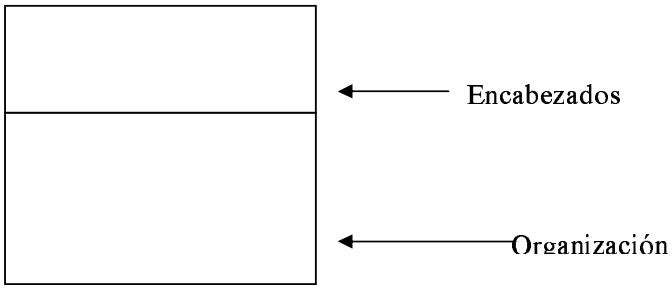

Figura 3.1: Estructura de un archivo XML

En la parte superior se sitúan las etiquetas que indican el tipo de archivo del que se trata y los esquemas que utilizan para definir las etiquetas dentro de él. Por ejemplo, las siguientes líneas indican que el archivo XML estará construido bajo la versión 1.0 de XML en base al ISO-8859-1 y la siguiente línea indica que el esquema que rige el archivo es el definido por el estándar LOM creado por LTSC-IEEE y la dirección en Internet en la que se encuentra.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<lom xmlns="http://ltsc.ieee.org/xsd/LOM"
    xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
    xsi:schemaLocation="http://ltsc.ieee.org/xsd/LOM
    http://ltsc.ieee.org/xsd/lomv1.0/lom.xsd">
```

Figura 3.2: Encabezado del Archivo Manifiesto

En la parte inferior de la figura, representa la organización del archivo respecto a las categorías de los metadatos. Por ejemplo:

Esta organización se representa con un diagrama de árbol como el mostrado en la figura 3.4: donde el elemento lom es el elemento raíz.

```
<lom>
  <general>
    <title>
      <string language="en">titulo</string>
    </title>
    <language>Español</language>
  </general>
  <lifeCycle>
    <status>
      <value>final</value>
    </status>
  </lifeCycle>
  <metaMetadata>
    <identifier>
      <catalog>catalogo</catalog>
    </identifier>
  </metaMetadata>
  <technical>
      <installationRemarks>
        <string language="es">notas de instalacion</string>
      </installationRemarks>
  </technical>
  <rights>
    <cost>
      <value>yes</value>
    </cost>
    <description>
      <string>descricpcion</string>
    </description>
  </rights>
</lom>
```

Figura 3.3: Cuerpo del Archivo Manifiesto

Algunos estándares que se encuentran en el mercado actualmente son SCORM, versiónes 1.2 y 2004, IMS Global Learning Consortium, LOM, Dublin Core, Edusource, entre otros. Entre otras especificaciones para los OA por parte de los estándares internacionales especifican los esquemas para crear los metadatos en el formato XML. Algunos de estos estándares indican que los metadatos deberán de estar en un paquete junto con el OA. Este paquete será en formato ZIP; en donde el contenido de el serán: el OA, el archivo XML que contenga los metadatos (que deberá llamarse IMSMANIFEST.XML que es denominado Archivo Manifiesto) y una organización. El archivo Manifiesto contiene un inventario estructurado del contenido del paquete. En cuanto a la organización, contiene la estructura del contenido del OA. El paquete en formato

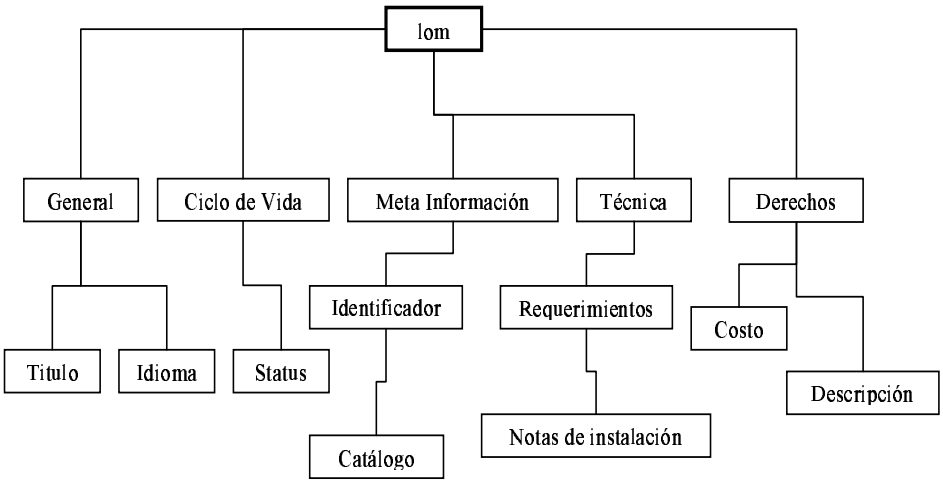

Figura 3.4: Organización del archivo XML respecto a los metadatos

ZIP se representa en la figura 3.5. El archivo manifiesto contiene los metadatos, la organización de los metadatos los recursos del archivo y el OA, denotado como archivo físico.

Figura 3.5: Estuctura del paquete ImsManifest

Para poder utilizar a los OA en un LMS como lo es Moodle necesitamos tener a los OA y los metadatos empaquetados en formato ZIP. Moodle al ser un sistema de gestión de enseñanza, permite ayudar a los educadores a crear cursos de calidad en línea. Estos tipos de sistema de aprendizaje a distancia a veces son también llamados ambientes

de aprendizaje virtual o educación en línea. Moodle tomará al paquete y va a explorar que el archivo manifiesto contenga las cabeceras necesarias que hagan referencia a los esquemas y al OA contenido en el mismo paquete. Si el paquete no esta creado con base a los esquemas de los estándares no podrá utilizar el OA.

Segun el estándar LOM [11] los metadatos se dividen en nueve categorías que son: General, Ciclo de Vida, Meta Información, Técnica, Educativa, Derechos, Relación, Anotación y Clasificación, las cuales se ilustran en la figura 3.6

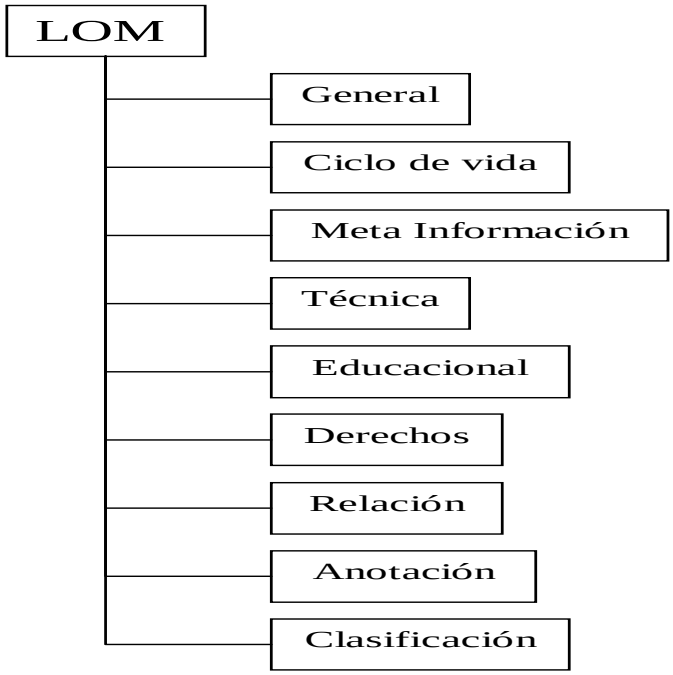

Figura 3.6: Categorías principales del estándar LOM

En el estándar LOM [11] de la IEEE cada metadato es opcional. La lista de elementos requeridos difiere dependiendo del componente al cual describa. Las nueve categorías en las que ha sido dividido el LOM las describimos a continuación.

3.2. Categorías de Metadatos

3.2.1. Categoría General

En la categoría ***General*** se agrupa la información general que describe al OA. La información general, en ocasiones es vista por los buscadores y administradores de OA como la información clave para describir al OA. La figura 3.7, ilustra la subdivisión de la categoría ***General*** donde los elementos son: Identificador, Título, Idioma, Descripción, Palabras Clave, Cobertura, Estructura y Nivel de Agregacón.

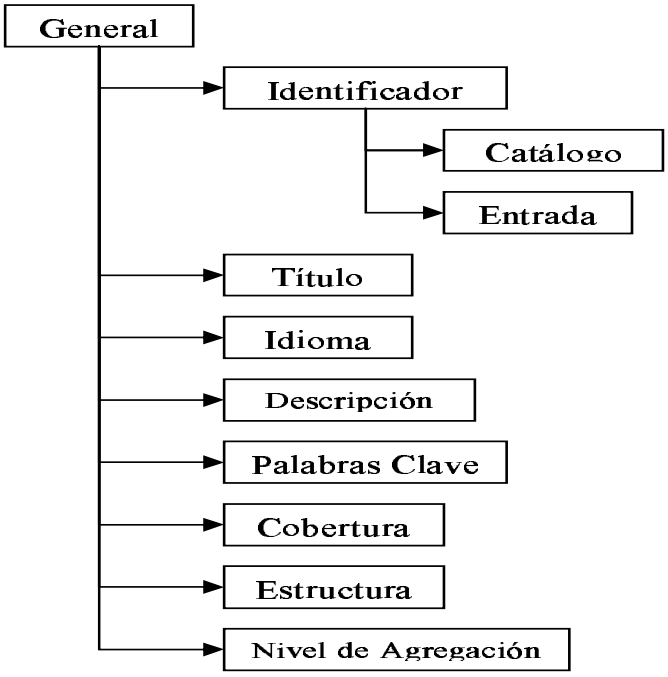

Figura 3.7: Subdivisión de los elementos de la categoría General

3.2.2. Categoría Ciclo de Vida

En ella se describen las características relacionadas con la historia y estado actual del componente, las cuales afectan al componente durante su evolución. En la figura 3.8,

puede verse el esquema de división de esta categoría, en ella se denota que la categoría es ***Ciclo de Vida*** esta subdividida en los elementos Estado y Contribución; y que este a su vez contiene los elementos Función, Entidad y Fecha.

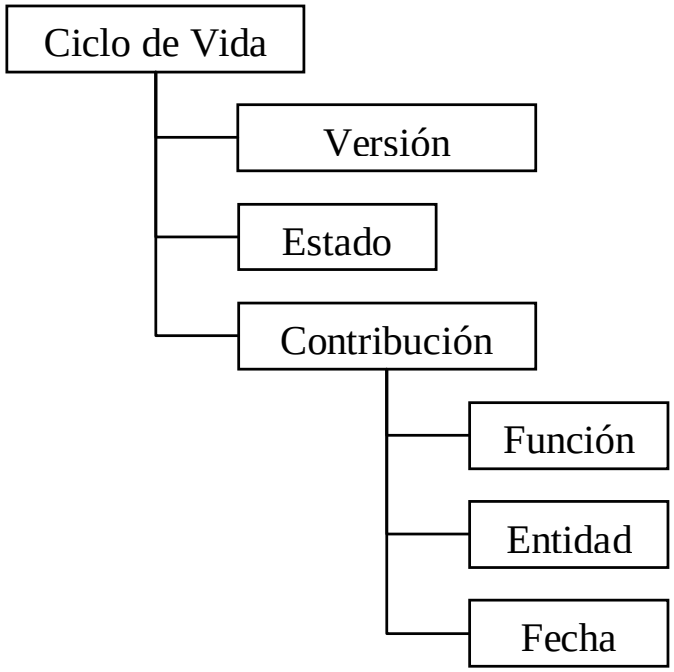

Figura 3.8: Subdivisión de los elementos de la categoría Ciclo de Vida

3.2.3. Categoría Meta Información

Se usa para describir información acerca de los registros de sus metadatos; también describe como la instancia de metadatos puede ser definida. En la figura 3.9, puede verse el diagrama de la categoría de ***Meta Información***, en ella se ilustra que está subdividida en los elementos Identificador, Contribución, Esquema de meta Información e Idioma. Donde el elemento Identificador contiene los elementos Catálogo y Entrada, y el elemento de Contribución contiene los elementos Función, Entidad y Fecha.

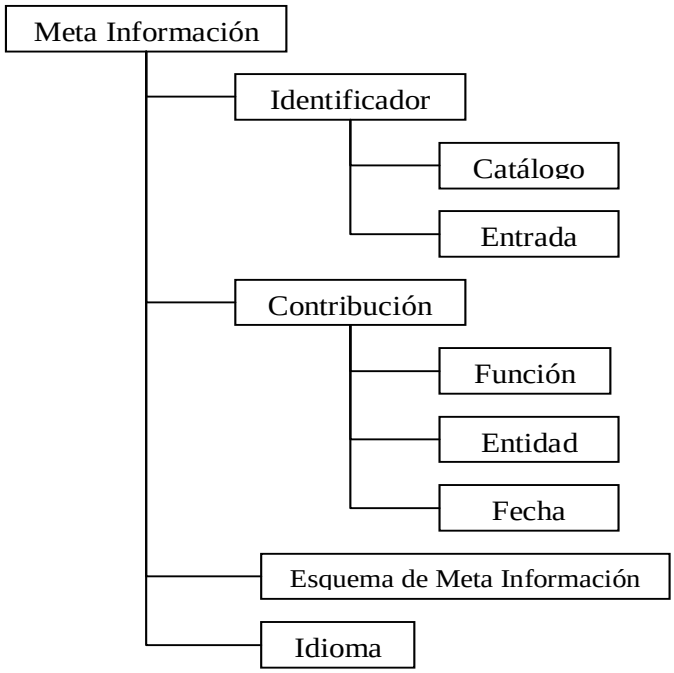

Figura 3.9: Subdivisión de los elementos de la categoría Meta Información

3.2.4. Categoría Técnica

Esta categoría describe las condiciones técnicas requeridas y las características del OA. En la figura 3.10, puede verse el esquema de las subdiviciones de la categoría de ***Técnica***, en ella se muestra que esta subdividida en los elementos Formato, Tamaño, Ubicación, Comentarios de Instalación, Otros Requisitos de Plataforma, Duración y Requerimientos. Donde el elemento Requerimientos contiene el elemento Compuesto y este a su vez a los elementos Tipo, Nombre, Versión Mínima y Versión Máxima.

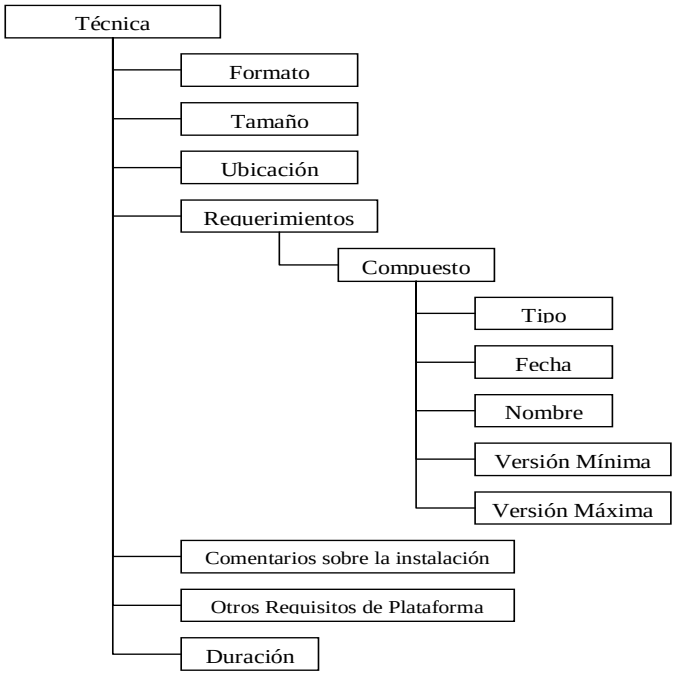

Figura 3.10: Subdivisión de los elementos de la categoría Técnica

3.2.5. Categoría Educativa

Describe las características educativas clave del OA. Esta es la información pedagógica esencial para aquellos implicadas en el desarrollo de entornos de aprendizaje de calidad. El destinatario de estas metainformación incluye al profesorado, gestores educativos, autores y alumnado.

El esquema de la categoría de ***Educativa***, puede verse en la figura 3.11, en donde el elemento Educacional contiene a los elementos Tipo de Interactividad, Tipo de Recurso de Aprendizaje, Nivel de Interactividad, Densidad Semántica, Usuario Final, Contexto, Rango de Edad, Dificultad, Tiempo de Aprendizaje, Descripción e Idioma. Ninguno de los sub elementos del elemento Educaiconal contiene sub elementos a su vez.

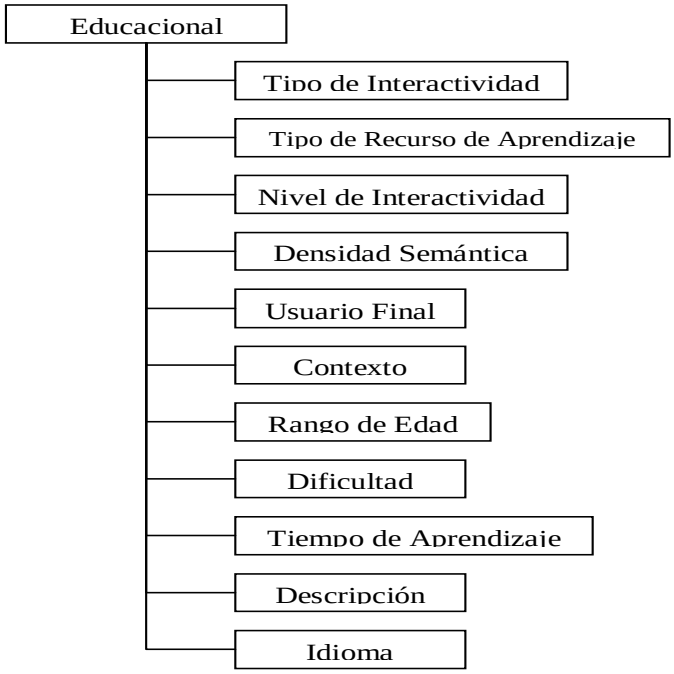

Figura 3.11: Subdivisión de los elementos de la categoría Educativa

3.2.6. Categoría Derechos

Esta categoría describe la propiedad intelectual y las condiciones para utilizar este OA. La intención es incorporar en el futuro las especificaciones sobre derechos de autor y comercio electrónico. Por ello, esta categoría ofrece actualmente un nivel de detalle muy general. El esquema de esta categoría, esta representado en la figura 3.12, en donde el elemento Derechos contiene a los elementos Costo, Descripción y *CopyRight*.

3.2.7. Categoría Relación

La categoría que define la relación entre el OA y otros OAs (si existen) es la de ***Relación***. Para definir relaciones múltiples, pueden establecerse instancias múltiples de esta categoría. Si hay más de un OA principal, entonces cada uno de estos se

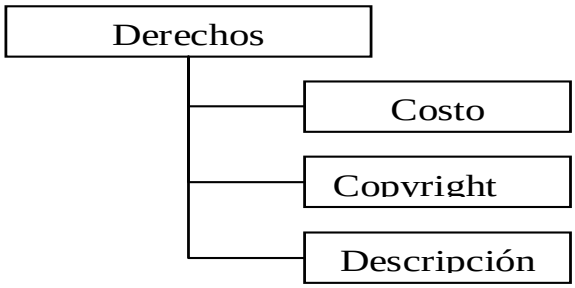

Figura 3.12: Subdivisión de los elementos de la categoría Derechos

referenciará con una nueva instancia de relación.

La categoría Relación representada se puede ver en la figura 3.13, en donde el elemento Relación contiene los elementos Tipo, Recurso, Identificador y Descripción. El elemento Recurso contiene los elementos Identificador y Descripción; el elemento Identificador contiene a su vez de los elementos Catálogo y Entrada.

3.2.8. Categoría Anotación

Esta categoría contiene comentarios sobre el uso educativo del OA; sobre quién generó el comentario y cuándo lo hizo. Cuando se incluyen varios comentarios, se deben utilizar instancias múltiples de esta categoría. En la figura 3.14, puede verse el esquema de la categoría de ***Anotación***. La figura muestra al elemento Anotación que contiene los elementos Entidad, Fecha y Descripción.

3.2.9. Categoría Clasificación

En esta categoría se describe la ubicación del OA dentro de un sistema de clasificación específico. Para definir clasificaciones múltiples, deben establecerse instancias

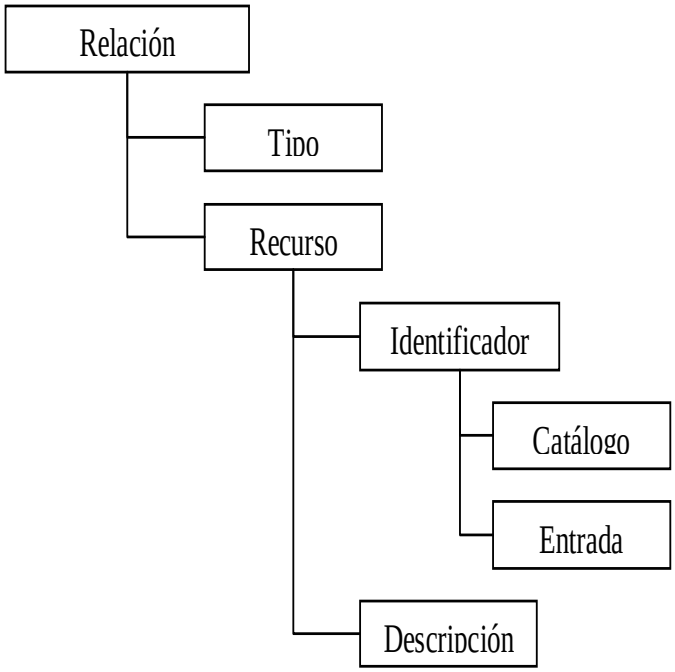

Figura 3.13: Subdivisión de los elementos de la categoría Relación

múltiples de esta categoría. En la figura 3.15, puede verse el esquema de la categoría *Clasificación* donde se observa que esta categoría contiene los elementos Propósito, Nivel Taxonómico, Descripción y Palabras Clave; entre ellos, el elemento Nivel Taxonómico contiene a su vez de los elemtos Fuente y Taxonomía y este a su vez contiene los elementos ID y Entrada.

3.3. Análisis comparativo entre estándares internacionales

Una vez descritos los elementos que comprenden las nueve categorías en que está dividido el estándar LOM, se contempla la manera en que lo aplican algunas organiza-

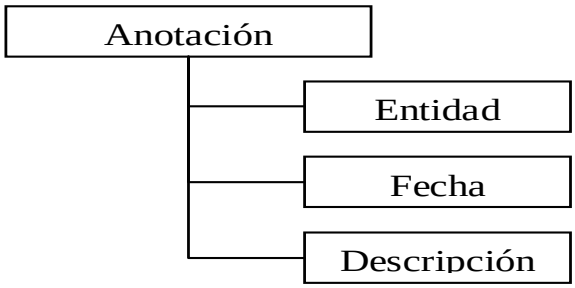

Figura 3.14: Subdivisión de los elementos de la categoría Anotación

ciones. Se Analizan varias propuestas de metadatos de diferentes organizaciones. Empezando con toda la estructura definida por el estándar LOM del IEEE-LTSC 148.1.2.1 el [11] cual indica que todos los metadatos son opcionales. Tomando en cuenta la importancia de los estándares en el desarrollo de aplicaciones y herramientas distribuidas, analizamos el estándar SCORM que es impulsado por el Advanced Distributed Learning de los Estados Unidos de América. El análisis para el SCORM se llevo a cabo en las versiónes 1.2 y 2004. Así mismo, sew analizó la propuesta de metadatos planteada por el IMS *Global Learning Consortium*, Inc [12]. Finalmente también se observa la plantilla del DUBLIN CORE [7] que es una organización dedicada a la promoción y difusión de normas interoperables sobre metadatos y el desarrollo de vocabularios especializados en metadatos para la descripción de OAs que permitan sistemas de recuperación mas inteligentes.

La nomenclatura utilizada para etiquetar los campos en las tablas subsecuentes es la siguiente:

OK.- Campos obligatorios

OP.- Campos opcionales

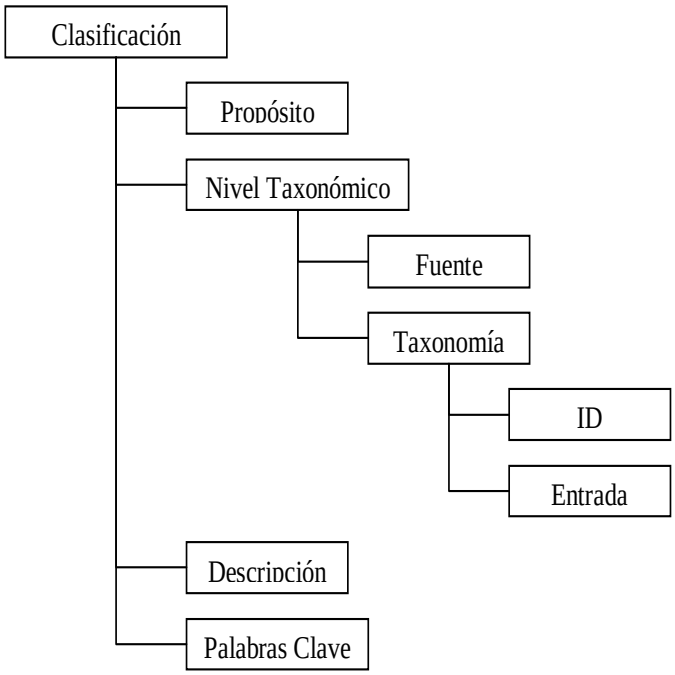

Figura 3.15: Subdivisión de los elementos de la categoría Clasificación

No Usar.- Campos no usados

En los datos de la categoría *General* presentados en la tabla 3.1, no todas las propuestas son uniformes, pero existen elementos en los que coinciden la mayoría de las propuestas, por ejemplo el elemento Título, la Entrada del Catálogo, la Descripción y las Palabras Clave del objeto.

Para la categoría *Ciclo de Vida*, de igual forma no todos los elementos coinciden, el elemento más recurrido por las propuestas es el elemento Estado, en la tabla 3.2 se observa como el IMS [12] es el que tiene mas diferencias entre las otras tres propuestas.

En la tabla 3.3, se encuentra la información referente a la *Meta Información*. En

ELEMENTOS LOM		**SCORM 2004**	**SCORM 1.2**	**IMS**	**DUBLIN CORE**
Identificador		OK	OP	OK	No Usar
Título		OK	OK	OP	OK
Catálogo		OP	OK	OP	No Usar
	Catálogo	OP	OK	OP	No Usar
	Entrada	OK	OK	OK	No Usar
Idioma		OP	OP	OK	OK
Descripción		OK	OK	OP	OK
Palabras Clave		OK	OK	OP	OK
Cobertura		OP	OP	OP	OK
Estructura		OP	OP	OK	No Usar
Nivel de Agregación		OP	OP	OP	No Usar

Tabla 3.1: Análisis Comparativo de la Categoría General

ELEMENTOS LOM		**SCORM 2004**	**SCORM 1.2**	**IMS**	**DUBLIN CORE**
Versión		OK	OK	OP	No Usar
Estado		OK	OK	OK	No Usar
Otros Colab		OP	OP	OP	OK
	Función	OP	OP	OP	No Usar
	Entidad	OP	OP	OP	No Usar
	Fecha	OP	OP	OP	OK

Tabla 3.2: Análisis Comparativo de la Categoría Ciclo de Vida

esta categoría es donde se encuentra un alto grado de discrepancias entre las propuestas.

ELEMENTOS LOM		**SCORM 2004**	**SCORM 1.2**	**IMS**	**DUBLIN CORE**
Identificador		OK	OP	OP	OK
Catálogo		OP	OP	OP	No Usar
	Catálogo	OP	OP	OP	No Usar
	Entrada	OK	OP	OK	No Usar
Otros Colab		OP	OP	OP	OK
	Función	OP	OP	OP	No Usar
	Entidad	OP	OP	OP	No Usar
	Fecha	OP	OP	OP	No Usar
Esquema de meta inf		OK	OK	OP	No Usar
Idioma		OP	OP	OP	No Usar

Tabla 3.3: Análisis Comparativo de la Categoría Meta Información

Para la categoría *Técnica*, la cual tiene reflejados los valores de cada propuesta en la tabla 3.4, existe una mayor coincidencia en los elementos seleccionados; las cuatro propuestas consideran solo el elemento Formato, con excepción de SCORM 1.2 que también considera elemento Ubicación.

Lo que respecta a los datos obtenidos para la categoría *Educativa*, representados en la tabla 3.5, no se obtienen coincidencias. La propuesta del IMS es la que realiza una selección mayor de elementos con un total de tres; los que no muestran interés en el uso educativo son las propuestas de SCORM 1.2 y SCORM 2004.

Los datos de la categoría *Derechos*, representados en la tabla 3.6, el IMS no tiene ningún valor seleccionado y el Dublin Core solo tiene seleccionado el campo de *Copy-Rights*, en comparación con el SCORM 1.2 y el SCORM 2004 que tienen seleccionado los campos *CopyRights* y Costo.

ELEMENTOS LOM		**SCORM 2004**	**SCORM 1.2**	**IMS**	**DUBLIN CORE**
Formato		OK	OK	OK	No Usar
Tamaño		OP	OP	OP	No Usar
Ubicación		OP	OK	OP	No Usar
Requisitos		OP	OP	OP	No Usar
	Tipo	OP	OP	OP	No Usar
	Nombre	OP	OP	OP	No Usar
	Versión Mínima	OP	OP	OP	No Usar
	Versión Máxima	OP	OP	OP	No Usar
Comentarios		OP	OP	OP	No Usar
Otros Requisitos		OP	OP	OP	No Usar
Duración		OP	OP	OP	No Usar

Tabla 3.4: Análisis Comparativo de la Categoría Técnica

ELEMENTOS LOM	**SCORM 2004**	**SCORM 1.2**	**IMS**	**DUBLIN CORE**
Tipo de Inrteractividad	OP	OP	OK	No Usar
Tipo de Recurso de Aplicación	OP	OP	OP	OK
Nivel de Interacción	OP	OP	OP	No Usar
Densidad Semántica	OP	OP	OP	No Usar
Usuario Principal	OP	OP	OK	No Usar
Nivel Educativo	OP	OP	OP	No Usar
Edad	OP	OP	OP	No Usar
Dificultad	OP	OP	OP	No Usar
Tiempo de aprendizaje	OP	OP	OP	No Usar
Descripción	OP	OP	OP	No Usar
Idioma	OP	OP	OP	No Usar

Tabla 3.5: Análisis Comparativo de la Categoría Educativa

ELEMENTOS LOM	**SCORM 2004**	**SCORM 1.2**	**IMS**	**DUBLIN CORE**
Costo	OK	OK	OP	No Usar
CopyRights	OK	OK	OP	OK
Descripción	OP	OP	OP	No Usar

Tabla 3.6: Análisis Comparativo de la Categoría Derechos

En los datos obtenidos de la categoría *Relación*, mostrados en la tabla 3.7 las coincidencias abundan, es decir, todas las propuestas optan porque todos los elementos sean opcionales, excepto para el Dublin Core que marca el elemento recurso.

ELEMENTOS LOM			**SCORM 2004**	**SCORM 1.2**	**IMS**	**DUBLIN CORE**
Tipo			OP	OP	OP	No Usar
Recurso			OP	OP	OP	OK
	Identificador		OP	OP	OP	No Usar
		Catálogo	OP	OP	OP	No Usar
		Entrada	OP	OP	OP	No Usar
	Descripción		OP	OP	OP	No Usar

Tabla 3.7: Análisis Comparativo de la Categoría Relación

En la tabla 3.8 se observan los datos obtenidos de la categoría *Anotación* donde todas las propuestas coinciden de manera uniforme que todos los elementos son opcionales.

ELEMENTOS LOM	**SCORM 2004**	**SCORM 1.2**	**IMS**	**DUBLIN CORE**
Persona	OP	OP	OP	No Usar
Fecha	OP	OP	OP	No Usar
Descripción	OP	OP	OP	No Usar

Tabla 3.8: Análisis Comparativo de la Categoría Observaciones

Por último, se presenta la categoría de clasificación, la cual representa sus datos en la tabla 3.9; para las propuestas de SCORM 1.2 y SCORM 2004 todos los elementos de esta categoría con opcionales, pero para el IMS, la identificación taxonómica es considerada.

ELEMENTOS LOM			**SCORM 2004**	**SCORM 1.2**	**IMS**	**DUBLIN CORE**
Finalidad			OP	OP	OP	No Usar
Nivel Taxonómico			OP	OP	OP	No Usar
	Fuente		OP	OP	OP	OK
	Taxón		OP	OP	OP	No Usar
		Id	OP	OP	OK	No Usar
		Entrada	OP	OP	OK	No Usar
	Descripción		OP	OP	OP	No Usar
	Plabras Clave		OP	OP	OP	No Usar

Tabla 3.9: Análisis Comparativo de la Categoría Clasificación

Capítulo 4

Metodología para estandarizar Objetos de Aprendizaje

La metodología escrita en este capítulo tiene por objetivo el guiar al diseñador de OA en el proceso de estandarización. Le metodología consta de diversas fases durante las cuales se van definiendo los metadatos, el número de fases depende del estándar seleccionado. Esta metodología se complementa con una herramienta de apoyo para le implementación de cada fase y la verificación de los metadatos generados. El desarrollo y uso de la aplicación se describe en capítulos posteriores.

Como se ha mencionado anteriormente, son varios los estándares para OA y aunque la mayoría de los metadatos son similares, su organización y notación varía según el estándar relacionado. Nuestra metodología esta organizada por estándares y para cada uno de ellos se describe su fase correspondiente.

Un OA es una unidad de aprendizaje independiente y auto-contenida que esta predispuesta a reusarse en multiples contextos instruccionales. Dada su importancia, algunos autores como Sicilia [25] incluso sugieren una medida numérica con el fin de poder evaluar sistemáticamente el grado de reutilización de un objeto de aprendizaje

dado. En términos simples, los objetos de aprendizaje son unidades digitales mínimas, que pueden ser utilizadas, reutilizadas, referenciadas, auto-contenidas, intercambiables e interoperables, mediante sistemas de metadatos en aprendizaje electrónico. Estas unidades mínimas se combinan y deben funcionar dentro de un LCMS (*Learning Content Management System* ó Sistema de Gestión de Contenido de Aprendizaje).

4.1. Estandarizando en Learning Object Metadata

4.1.1. Primera fase: Características generales de los OA

En primera instancia, se debe efectuar la selección del estándar bajo el cual se deben crear los metadatos para el OA, ya que esta determinará la organización y metadatos necesarios.

En esta fase, se describen los datos generales de un OA, con el objetivo de poder contar con información suficiente para poder tener una mejor administración del objeto tanto en repositorios como en mercados o buscadores. Con esta información se puede distinguir una OA de otro, así, como también obtener una breve descripción del OA. Los campos incluidos son los metadatos en la figura y presentados a continuación.

1. Identificador.- El elemento *identificador* contiene una etiqueta única global que identifica al OA. Se compone de dos elementos principales los cuales son catálogo y entrada.

    a) La entidad *catálogo* decide el esquema de catalogación de los OA.
    Por ejemplo, por medio de ISBN, en el caso de un libro o de una URL en el caso de una página *web*. Existe una gran diversidad de catálogos para

identificar OA, inclusive, cada repositorio o almacén de OA, puede tener un catálogo propio basado en referencias particulares, por ejemplo, Área, Idioma, etc. El nombre de los catálogos puede ser particularizado tanto como se desee.

b) El campo *entrada* contiene el valor de referencia del OA de acuerdo al tipo de catálogo empleado. Por ejemplo, si el catálogo es ISBN, la entrada puede ser 84-481-384-7; si el catálogo es Áreas, la entrada puede ser Naturales, Sociales, etc.

c) En general, los catálogos deben de ser proporcionados por el administrador de los OA

2. El elemento *título* contiene el nombre dado al OA. Se recomienda que este sea de acuerdo al tema del OA

3. *Idioma*, define la lengua principal utilizada en el OA para comunicarse con el usuario.

4. El elemento *descripción* tiene como función explicar textualmente el contenido del OA.

5. En el elemento de *palabras clave* se definirán las palabras principales que describen el OA y pueden ser usadas para la localización del OA si este se encuentra en un repositorio.

6. La *cobertura* contenpla el entorno espacial y/o temporal de conceptos tales como: tiempo, cultura, geografía o región física del OA.

7. En el campo de *estructura* se describe la estructura intrínseca de organización del OA. Este elemento cuenta con un vocabulario controlado, que es el siguiente:

- Atómico.- El objeto es indivisible
- Colección.- Conjunto de objetos donde no se especifica relación entre ellos
- Red de trabajo.- Conjunto de objetos relacionados
- Jerárquico.- Conjunto de objetos relacionados donde se puede ser representado por medio de una estructura de árbol
- Lineal.- Conjunto de objetos que están completamente ordenados.

8. El *nivel de agregación* define el nivel de complejidad funcional del OA. El vocabulario de este elemento esta controlado y se denota por: 1, 2, 3, 4

   - 1.- Es el menor nivel de agregación. Por ejemplo: fragmentos de datos multimedia
   - 2.- Colección de OA del nivel 1. Ejemplo: una lección
   - 3.- Colección de OA del nivel 2. Ejemplo: un curso
   - 4.- Es el nivel más general. Ejemplo: un conjunto de cursos

4.1.2. Segunda fase: Características del ciclo de vida de los OA

En esta fase se describen las características relacionadas con la historia y estado actual del OA, las cuales lo afectan durante su evolución. Por ejemplo la versión del OA, el estado en el que se encuentra y las contribuciones que ha tenido.

1. La *versión* contendrá información respectiva a la edición del OA. Un OA puede tener varias versiónes o ediciones durante su tiempo de vida. Por ejemplo Versión 1.0, 2.5, 2.9, etc.

2. El campo de *estado* representa el estado o la condición de el OA. El elemento estado contiene vocabulario controlado que es el siguiente:

   a) Borrador

   b) Final

   c) Revisado

   d) No disponible

3. La *contribución* especifíca las personas u organizaciones que han contribuido en la creación del OA (incluidas las ediciones y la publicación). En este elemento se contemplan los elementos: Función, Entidad y Fecha.

   a) La *función* define el tipo de colaboración. Debe incluir exactamente una instancia de autoría. Por ejemplo Creador o Revisor.

   b) La *entidad* contiene la identificación e información sobre las personas u organizaciones que han participado en el OA.

   c) La *fecha* almacéna la fecha de la colaboración.

4.1.3. Tercera fase: Características a cerca de la información de los OA

Esta fase se usa para describir información acerca de los registros de los metadatos del OA. Para describir la información de los metadatos se solicitan datos como el identificador de del esquema que se utiliza para desarrollar los metadatos, el idioma de estos y las contribuciones que se han tomado para describirlos.

1. La información solicitada por el campo *identificador* contendrá la misma definición escrita para el elemento *identificador* y sus sub elementos *Catálogo y Entrada* de la categoría *General*

a) Catálogo

b) Entrada

2. El elemento *contribución* contendrá la misma descripción que el elemento *contribución* y sus sub elementos *Fecha, Entidad y Función* de la categoría *Ciclo de Vida*

3. El elemento *esquema de meta información* se utiliza para contener el nombre y la versión de la especificación autorizada utilizada para crear instancias de metadatos. Si este elemento contiene valores múltiples, la instancia de metadatos debe estar conforme con los múltiples esquemas especificados. Como ejemplo de esquemas autorizados pueden ser los siguientes:

   - *LOM Versión 1.0*
   - *SCORM CAM Versión 2004*

   donde *LOM Versión 1.0*, indica que se ha usado el esquema base de LOM 1.0 para la creación de los metadatos y *SCORM CAM Versión 2004*, indica que se ha seguido las especificaciones del SCORM CAM Versión 2004 para la creación del esquema de metadatos

4. El campo que solicita el idioma especifica la lengua con el cual se ha creado la instancia de metadatos.

4.1.4. Cuarta fase: Características técnicas de los OA

En esta fase se describen las condiciones y características técnicas requeridas del OA. La información que se tiene que complementar en la fase técnica es el formato,

tamaño, ubicación, comentarios de instalación, otros requisitos de plataforma, duración y requerimientos del OA. Donde el elemento de requerimientos contiene el elemento compuesto y este a su vez a los elementos tipo, nombre, versión mínima y versión máxima.

1. El elemento *formato* especifica el tipo de datos técnicos del OA. Identifica el *software* que se necesita para mostrar el OA.

2. El elemento *tamaño* especifica la dimensión del OA digital en bytes. Contiene los dígitos '0'..'9'; la unidad es el byte, no el Mega Byte, Giga Byte, etc. Debe referirse al tamaño real del OA y no al tamaño de una versión comprimida del mismo.

3. En la *ubicación* se define la cadena que se utiliza para acceder al OA. La información introducida en este apartado dependerá de la administración del repositorio o almacén de OA, Puede ser una ubicación por ejemplo: una URL, o un método que lleva a una ubicación por ejemplo: una URI, si el repositorio es local, la ubicación puede hacer referencia a la carpeta local donde se encuentren los OA, por ejemplo C:/temp/

4. Los *requerimientos* describen los requisitos técnicos para utilizar el OA. Si hay muchos requisitos, todos ellos serán necesarios, es decir que el conector lógico será *AND*.

    a) El elemento *compuesto* especifica un solo requerimiento y en elementos que lo componen describe la tecnología apara utilizar el OA. El elemento compuesto contiene a los elementos: Tipo, Nombre, Versión Mínima y Versión Máxima.

1) El elemento *tipo*, especifica la tecnología requerida para utilizar OA. Este elemento cuenta con un vocabulario controlado y este consiste en dos opciones:
   - Sistema Operativo
   - Navegador
2) El campo *nombre*, es referente al de la tecnología requerida para utilizar el OA. Contiene un vocabulario controlado; viene dado dependiendo del valor elejido en el campo *tipo*.
   - Si se selecciono la opción: Sistema Operativo
     - PC-DOS
     - MS-WINDOWS
     - MAC-OS
     - UNIX
     - MULTI-OS
     - NINGUNO
   - Si se selecciono la opción: Navegador
     - any
     - nestcape comunicator
     - ms-internet explorer
     - opera
     - amaya
3) La *versión mínima* especifica la versión tecnológicamente inferior requerida para utilizar el OA.

4) La *versión máxima* especifica la más reciente versión conocida para utilizar el OA.

4.1.5. Quinta fase: Características educativas del OA

En esta fase se describen las características educativas clave del OA. Esta es la información pedagógica esencial para aquellos implicados en el desarrollo de entornos de aprendizaje de calidad. El destinatario de esta metainformación incluye al profesorado, gestores educativos, autores y alumnado.

Para complementar esta categoría es necesario introducir datos referentes a el tipo de interactividad, tipo de recurso de aprendizaje, nivel de interactividad, densidad semántica, usuario final, contexto, rango de edad, dificultad, tiempo de aprendizaje, descripción e idioma.

1. El tipo de *interactividad* especifica el flujo de interacción entre el OA y el usuario. En un OA expositivo, la información fluye principalmente desde el OA hasta el alumno. Los documentos expositivos son, en general, para el aprendizaje a través de la lectura. En un OA activo, la información fluye también del alumno al OA. Los documentos activos son, en general, utilizados para el aprendizaje activo. La activación de vínculos para navegar en documentos hipertexto no está considerada como un flujo de información. Los documentos hipertextuales son documentos expositivos. Por ejemplo, se pueden colocar los siguietnes valores:

   - Activo
   - Expositivo

- Mixto

2. La información que se debe ingresar en el *tipo de OA*, debe especificar de que tipo OA se trata; el vocabulario para este elemento es controlado y se puede elejir el tipo de OA, por ejemplo si es un ejercicio, simulación, cuestionario, diagrama, figura, gráfica, tabla, examen, etc.

3. Para el elemento referente al *nivel de interactividad* se debe definir el grado de interactividad entre el usuario final y el OA, donde: 0 define un nivel de interactividad:

   "Muy bajo", 1 "Bajo", 2 "Medio", 3 Alto", y 4 "Muy alto"

   en donde la selección del valor de este campo es consideración del diseñador.

4. La *densidad semántica* representa el grado de consistencia del elemento, Estos elementos deben definir una medida subjetiva de la utilidad del OA en comparación con su tamaño o duración, donde: 0 define un nivel de densidad semántica:

   "Muy bajo", 1 "Bajo", 2 "Medio", 3 Alto", y 4 "Muy alto"

   en donde la selección del valor de este campo es consideración del diseñador.

5. En el apartado de *usuario final* se especifica el principal usuario para el que se ha diseñado el OA. El vocabulario de este elemento también es controlado por las siguientes opciones:

   - Maestro
   - Autor
   - Estudiante

- Administrador

6. La información que se deberá ingresar en el campo de *contexto* depende del entorno para el que el OA va destinado, por ejemplo, Educación continua, educación superior, primaria, secundaria, entre otros.

7. El *rango de edad* especifica la edad del usuario al que se destina el OA

8. Para editar el campo *dificultad*, seleccionar del menú emergente: 1.-Muy fácil 2.-Fácil 3.-Medio 4.-Difícil 5.-Muy difícil

9. El campo de *tiempo de aprendizaje* permite establecer el tiempo de aprendizaje en Formato DateType: Horas:Minutos:Segundos

10. La *descripción* educativa son comentarios sobre cómo utilizar el OA.

11. El *idioma* define la lengua principal utilizada para comunicarse con el usuario en el OA se especifica en este elemento.

4.1.6. Sexta Fase: Derechos del OA

En esta fase se describe la propiedad intelectual y las condiciones para utilizar este OA. La intención es incorporar en el futuro las especificaciones sobre derechos de autor y comercio electrónico. Por ello, esta categoría ofrece actualmente un nivel de detalle muy general. Los elementos que se deben de complementar en esta fase son: Costo, Descripción y CopyRight del OA.

1. La información referente al *costo* del OA especifica si el uso del OA está sujeto a pago. En ningún momento se determina el costo de este, solamente si esta sujeto

a el.

2. El campo de *Copyright* y otras restricciones es la especificación de que si el uso del OA está sujeto a derechos de autor u otras restricciones. Al igual que en el campo costo, no se especifica cuales son los *CopyRight*s que contiene el OA, solamente si esta sujeto a ellos.

3. La *descripción* contendrá comentarios acerca de las condiciones de utilización de este OA.

4.1.7. Septima Fase: Especificación de las relaciones del OA con otros OA's mas.

En esta fase se define la relación entre el OA y otros OAs (si existen). Si hay más de un OA principal, entonces cada uno de estos se referenciará con una nueva instancia de relación.

En esta fase se debe de describir el elemento hacia el cual es relacionado el OA y si se tiene catalogado, se debe de incluir el catálogo seguido para ello.

1. Para el campo *tipo*, se define la naturaleza de la relación entre el actual OA y el OA principal.

2. La información para el elemento *recurso* especifica el OA principal al que se refiere esta relación.

   a) El *Identificador* debe de contener una etiqueta única global que identifica al OA.

1) El *catálogo* nos sirve para colocar en él, el valor con el cual estamos catalogando a los OA secundarios relacionados con el OA principal, por ejemplo por medio de ISBN, o de una URL. Existe una gran diversidad de catálogos para identificar OA, inclusive, cada repositorio o almacén de OA, puede tener un catálogo propio basado en referencias particulares, por ejemplo, Area, Idioma, etc. El nombre de los catálogos puede ser particularizado tanto como se desee.

2) La *entrada* define el valor del catálogo. Por ejemplo, si el catálogo es ISBN, la entrada puede ser 84-481-384-7, si el catálogo es Áreas, la enrtada puede ser Naturales, Sociales, etc.

b) Para la *descripción* se deberá contener la descripción de las relaciones del OA principal.

4.1.8. Octava Fase: Anotaciones respectivas al OA

En esta fase se ingresarán los comentarios sobre el uso educativo del OA; sobre quién generó el comentario y cuándo lo hizo.

1. En el campo *entidad* se especifica a la persona que ha generado las observaciones.
2. La *fecha* especifica cuando se han generado las observaciones.
3. La información que se debe de colocar para el campo *descripción* deberá contener las observaciones referentes al OA.

4.1.9. Novena Fase: Clasificación del OA

La fase de Clasificación del OA describe la ubicación del OA dentro de un sistema de clasificación específico. Para definir clasificaciones múltiples, deben establecerse instancias múltiples de esta categoría. Para complementar esta fase es necesario proporcionar información referente a el propósito, nivel taxonómico, descripción y palabras clave del OA; entre ellos, el elemento nivel taxonómico contiene a su vez de los elemtos fuente y taxonomía y este a su vez contiene los elementos ID y entrada.

1. El campo *propósito* define la razón por la que se clasifica el OA.

2. El nivel *taxonómico* debe describir una ruta taxonómica en un sistema específico de clasificación. Cada nivel sucesivo aporta mayor precisión a la definición del nivel superior. Pueden haber diferentes rutas, en la misma o en diferentes clasificaciones, que describan la misma característica. Una taxonomía es una jerarquía de términos o frases denominadas taxones.

    a) La *fuente* es un elemento que especifíca el nombre del sistema de clasificación. Este elemento puede utilizar cualquier taxonomía *oficial* reconocida o una taxonomía definida por el propio usuario. Una herramienta de indexación o de interrogación puede ofrecer las entradas de nivel superior de un sistema de clasificación bien definido (LOC, UDC, DDC, etc.)

    b) La *taxonomía* deberá describir un término particular con una clasificación. Una subdivisión de la clasificación, puede ser un nodo el cual tiene una etiqueta o término definido. Una subdivisión de la clasificación puede tener también una designación alfa numérica para identificar su referencia. El elemento Taxonomía, contiene a los elementos ID y Entrada, los valores de los

sub elementos pueden ser utilizados para designar una clasificación determinada.

1) El *ID* especifica al identificador del táxon, como por ejemplo un número o una combinación de letras ofrecida por la fuente taxonómica, por ejemplo *e2587t*.

2) La *entrada* contiene la etiqueta textual del táxon, por ejemplo, *Relación directa al OA*.

3. El campo *descripción* nos sirve para describir en formato de texto libre el OA.

4. *Palabras Clave*.- Contiene las palabras clave del OA.

4.2. Estandarizando en Sharable Content Object Reference Model 1.2

Para estandarizar los OA bajo SCORM 1.2, hay que tomar en cuenta que solo se contemplan ciertos metadatos. Para ello se tienen que cubrir solo algunos aspectos como lo son: la información general, la información referente al ciclo de vida del OA, la Meta información, características técnicas y la información referente a los derechos del OA. En la figura 4.1se muestra un esquema de forma general de los campos de metadatos que requiere SCORM 2004.

General:

Las características generales del OA comprenden la siguiente información:

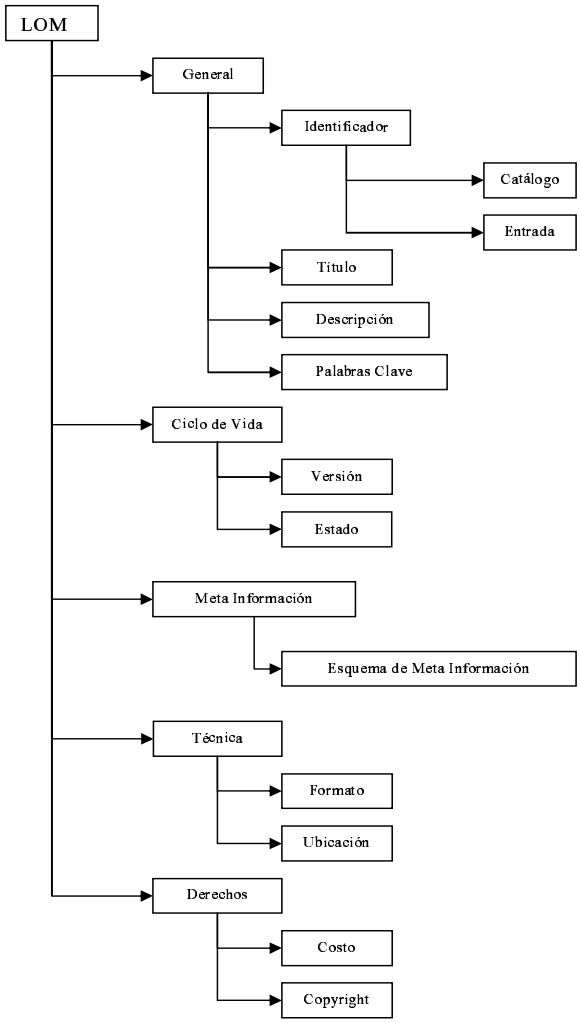

Figura 4.1: Metadatos requeridos por el estándar SCORM 1.2

Un ejemplo de la información del *catálogo de entrada y la entrada del cátalogo*, es si se tiene un OA que este catalogado por URL, la información proporcionada deberia ser:

catálogo: URL

entrada: www.ruta.com.

El título,el cual debe de contener el título del OA.

La Descripción, que es donde se debe de describir al OA.

Las Palabras clave del OA

Ciclo de vida:

Para complementar las características del ciclo de vida del OA se debe de incorporar la siguiente información:

La versión del OA, que es donde se edita la información respectiva del OA. Un OA puede tener varias versiónes o ediciones durante su tiempo de vida. Por ejemplo Versión 1.0, 2.5, 2.9, etc. *El estado* o la condición de el OA. El elemento estado contiene vocabulario controlado que es el siguiente:

1. Borrador
2. Final
3. Revisado
4. No disponible

Meta información:

La información que se presenta en estacategoría y que es requerida por el estándar SCORM 1.2 es *el esquema de metainformación*, el cual indique que esquema de metainformación se ha utilizado, por ejemplo: LomV1.0.

Técnica:

La información que describirá las características técnicas del OA, viene dada en esta categoría. Para el estándar SCORM 1.2 solo se requiere la información correspondiente a *el Formato del OA*, el cual especifica que tipo de formato tiene el OA. El vocabulario es controlado y permite seleccionar una gran cantidad de formatos por ejemplo: word, excel, java, zip, wav, mp3 etc. La otra característica es la *ubicación del OA*, esta determina el lugar en donde se encuentra. Por ejemplo:

Ubicación: www.repositorio.com.

Ubicación: c:/almacen.

Derechos:

Finalmente, el estándar SCORM 1.2 requiere que se proporcione la información correspondiente a los derechos del OA. Esta información esta destinada a la protección legal y de uso del OA. Para complementar esta información, es necesario indicar si el OA esta sujeto a *costo y a derechos de autor*. En los campos destinados a esta área, se especifica si o no el objeto tiene estas caractrarísticas. Estos campos no permiten indicar detalles a cerca del costo o los derechos reservados, simplemente indican su existencia.

Una vez introducida esta información, los metadatos del OA se consideran estandarizados bajo SCORM 1.2.

4.3. Estandarizando en Sharable Content Object Reference Model 2004

Para estandarizar los OA bajo SCORM 2004, solo se contemplan ciertos metadatos. Para ello se tienen que cubrir solo algunos aspectos como lo son: la información general, la información referente al ciclo de vida del OA, la Meta información, características técnicas y la información referente a los derechos del OA. En la figura 4.2se muestra un esquema de forma general de los campos de metadatos que requiere SCORM 2004.

General:

Las características generales del OA comprenden la siguiente información:

La entrada del cátalogo. Un ejemplo se observa si se tiene un OA que este catalogado por URL, la información proporcionada deberia ser:

catálogo: URL
entrada: www.ruta.com.

El título, el cual debe de contener el título del OA.
La Descripción, que es donde se debe de describir al OA.
Las Palabras clave del OA

Ciclo de vida:

Para complementar las características del ciclo de vida del OA se debe de incorporar

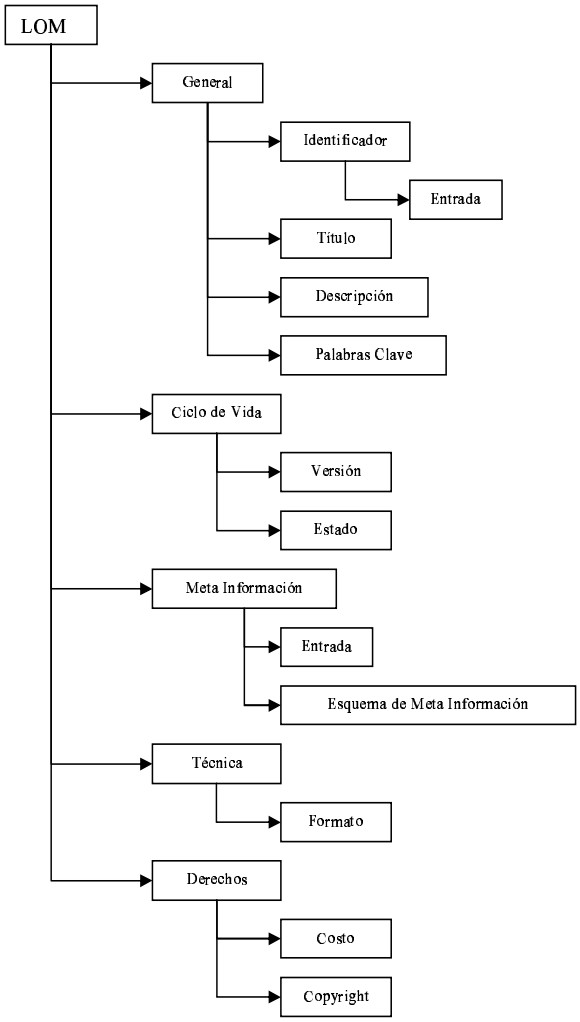

Figura 4.2: Metadatos requeridos por el estándar SCORM 2004

la siguiente información:

La versión del OA, que es donde se edita la información respectiva del OA. Un OA puede tener varias versiónes o ediciones durante su tiempo de vida. Por ejemplo Versión 1.0, 2.5, 2.9, etc. *El estado* o la condición de el OA. El elemento estado contiene vocabulario controlado que es el siguiente:

1. Borrador

2. Final
3. Revisado
4. No disponible

Meta información:

La información que se presenta en esta categoría y que es requerida por el estándar SCORM 2004 es *la entrada del catálogo y el esquema de metainformación.*

La información proporcionada en *la entrada del cátalogo*, es la que va a indicar la forma en que esta catalogado el OA. Por ejemplo, si se tiene un OA que este catalogado por ISBN, la información proporcionada deberia ser:

catálogo: ISBN
entrada: 1245-9874-954-6

El campo de *esquema de meta información* indica que esquema de metainformación se ha utilizado, por ejemplo: LomV1.0.

Técnica:

La información que describirá las características técnicas del OA, viene dada en esta categoría. Para el estándar SCORM 2004 solo se requiere la información correspondiente a *El Formato del OA*, el cual especifica que tipo de formato tiene el OA. El vocabulario es controlado y permite seleccionar una gran cantidad de formatos por

ejemplo: word, excel, java, zip, wav, mp3 etc.

Derechos:

Finalmente, el estándar SCORM 2004 requiere que se proporcione la información correspondiente a los derechos del OA. Esta información esta destinada a la protección legal y de uso del OA. Para complementar esta información, es necesario indicar si el OA esta sujeto a *costo y a derechos de autor*. En los campos destinados a esta área, se especifica si o no el objeto tiene estas caractrarísticas. Estos campos no permiten indicar detalles a cerca del costo o los derechos reservados, simplemente indican su existencia.

Una vez introducida esta información, los metadatos del OA se consideran estandarizados bajo SCORM 2004.

4.4. Estantarizando en IMS Global Learning Consortium

Para estandarizar los OA bajo el IMS, solo se contemplan ciertos metadatos para describir las caracteristicas del OA. Para ello se tienen que cubrir solo algunos aspectos como lo son: la información general, la información referente al ciclo de vida del OA, la Meta información, características técnicas y educativas del OA. En la figura 4.3se muestra un esquema de forma general de los campos de metadatos que requiere IMS.

General:

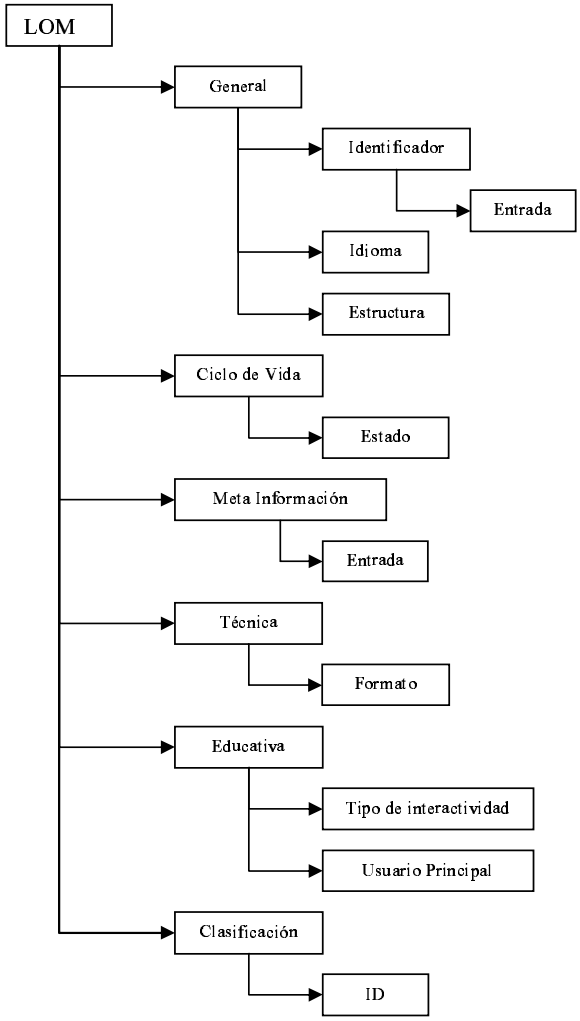

Figura 4.3: Metadatos requeridos por el estándar IMS Global Learning Consortium

Las características generales del OA comprenden la siguiente información:

La entrada del cátalogo. Un ejemplo se observa si se tiene un OA que este catalogado por URL, la información proporcionada deberia ser:

catálogo: URL

entrada: www.ruta.com.

El idioma es otra característica que se debe de definir para considerar los metadatos bajo IMS. Este apartado especificará la lengua bajo la cual se ha creado el OA.

La *Estructura* permite definir la organización del objeto, su vocabulario es controlado y nos presenta 5 opciones:

- Atómico: El objeto es indivisible.
- Colección: Conjunto de objetos donde no se especifica relación entre ellos.
- Red de trabajo: Conjunto de objetos relacionados.
- Jerárquico: Conjunto de objetos relacionados donde se puede ser representado por medio de una estructura de árbol.
- Lineal: Conjunto de objetos que estan completamente ordenados.

Ciclo de vida:

Para complementar las características del ciclo de vida del OA se debe de incorporar la siguiente información:

El estado o la condición de el OA. El elemento estado contiene vocabulario controlado que es el siguiente:

1. Borrador
2. Final

3. Revisado

4. No disponible

Meta información:

La información requerida por IMS, es el catálogo y la entrada del catálogo.

Para el *catálogo de entrada y la entrada del cátalogo*, la información proporcionada aqui es la que va a indicar la forma en que esta catalogado el OA. Por ejemplo, si se tiene un OA que este catalogado por ISBN, la información proporcionada deberia ser:

catálogo: ISBN
entrada: 1245-9874-954-6

Técnica:

La información que describirá las características técnicas del OA, viene dada en esta categoría. Para el estándar IMS solo se requiere la información correspondiente a:

El Formato del OA, el cual especifica que tipo de formato tiene el OA. El vocabulario es controlado y permite seleccionar una gran cantidad de formatos por ejemplo: word, excel, java, zip, wav, mp3 etc.

Educativa:

La informació requerida para describir las características educativas mínimas que solicita IMS son:

El Tipo de interactividad, donde el flujo de interacción entre el OA y el usuario es especificado por este elemento. Existen tres opciones para este campo. En un OA *expositivo*, la información fluye principalmente desde el OA hasta el alumno. Los documentos expositivos son en general, para el aprendizaje a través de la lectura. En un OA *activo*, la información fluye también del alumno al OA. Los documentos activos son en general, utilizados para el aprendizaje activo. La activación de vínculos para navegar en documentos hipertexto no está considerada como un flujo de información. Los documentos hipertextuales son documentos expositivos.

El *Usuario Principal* especifíca para que tipo de usuario se ha diseñado el OA. El vocabulario de este elemento es controlado por las siguientes opciones: Maestro, Autor, Estudiante, Administrador.

El idioma, se refiere al utilizado para comunicarse con el usuario en el OA.

4.5. Estandarizando en DUBLIN CORE

Los metadatos que el estándar Dublin Core toma en cuenta. son: la información general, la información referente al ciclo de vida del OA, características técnicas y educativas del OA.

General:

Las características generales del OA comprenden la siguiente información:

El título del OA, es una de las características generales que requiere Dublin Core, en el se introduce el nombre dado al OA.

El idioma es otra característica que se debe de definir para considerar los metadatos bajo Dublin Core. Este apartado especificará la lengua bajo la cual se ha creado el OA.

La Descripción textual del contenido del OA y las palabras clave de este.

De igual forma se requiere el elemento *Cobertura*, Este elemento define la cobertura espacial y/o temporal de conceptos tales como: tiempo, cultura, geografía o región física del recurso. Ejemplo: '1910, Etapa de Revolución Mexicana'

Ciclo de vida:

Para complementar las características del ciclo de vida del OA se debe de incorporar la siguiente información:

La funcion de *otros colaboradores* para el recurso de arendizaje. y la *fecha* en la que se ha realizado la colaboración.

Técnica:

La información que describirá las características técnicas del OA, viene dada en esta

categoría. Para el estándar Dublin Core solo se requiere la información correspondiente a:

El Formato del OA, el cual especifica que tipo de formato tiene el OA. El vocabulario es controlado y permite seleccionar una gran cantidad de formatos por ejemplo: word, excel, java, zip, wav, mp3 etc.

Educativa:

La informació requerida para describir las características educativas mínimas que solicita son:

Tipo de Recurso de aprendizaje, donde se especifica de que tipo OA se trata; el vocabulario para este elemento es controlado y se puede elegir el tipo de OA.

Derechos:

Finalmente, el estándar Dublin Core requiere que se proporcione la información correspondiente a los derechos del OA. Esta información esta destinada a la protección legal y de uso del OA. Para complementar esta información, es necesario indicar si el OA esta sujeto a *costo y a derechos de autor*. En los campos destinados a esta área, se especifica si o no el objeto tiene estas caractrarísticas. Estos campos no permiten indicar detalles a cerca del costo o los derechos reservados, simplemente indican su existencia.

Una vez introducida esta información, los metadatos del OA se consideran estandarizados bajo Dublin Core.

Capítulo 5

Modelo de Análisis

La creación de un OA se realiza en dos fases:

- La fase pedagógica, que es donde se realiza y decide el contenido del OA de acuerdo a los objetivos de aprendizaje
- La fase tecnológica, que es donde se lleva a cabo la construcción de los metadatos y otras características con las cuales un LMS podrá administrar a los OA más fácilmente.

Cada una de estas etapas comprende una rama de investigación bastante compleja. Nuestro trabajo asume que la fase pedagógica ya se ha realizado, incorporando en ella el diseño del contenido de dicho OA, por lo tanto nos concentraremos en la fase tecnológica, que es donde se incorporan los metadatos. Esta fase permitirá que los OA sean interoperables e intercambiables. De esta forma los OA pueden tener características que le permitan a un administrador de objetos de aprendizaje (como un mercado o un repositorio) organizarlos de la manera más adecuada de acuerdo a sus políticas y a los buscadores encontrarlos y accederlos fácilmente.

Uno de los problemas para hacer de un OA un recurso reutilizable, interoperable e intercambiable, es que a pesar de que el diseñador sea experto en el contenido del OA y por lo tanto pueda llevar a cabo la fase pedagógica, no siempre será experto en el uso de los equipos de cómputo y en la tecnología de OA. Esto plantea la necesidad de construir una aplicación que guíe al diseñador por diferentes etapas para incorporar al objeto de aprendizaje los elementos necesarios para cumplir con los estándares internacionales.

Por otro lado, existen mucho contenido electrónico educativo que no cumple con los requisitos para poder ser un OA y también existen muchos OA que no están estandarizados, lo cual dificulta su distribución, reutilización y disminuye la facilidad de búsqueda de dichos OA. Esta situación se ha convertido en un problema en el intento de conseguir una mejor administración de OA, por lo cual, se vuelve necesario verificar que los OA construidos cumplan con lo establecido por los estándares internacionales, para guiar al diseñador por diferentes etapas para que el OA pueda contener el aspecto tecnológico de acuerdo a los estándares internacionales y con ello, facilitar su distribución e intercambio, ahorrando costos de producción y facilitando la búsqueda de ellos.

Esta verificación, será incluida en la aplicación que guía al usuario (en las fases para incorporar los elementos necesarios para cumplir con los estándares internacionales), con la finalidad de corroborar que los OA contengan como mínimo los elementos que marcan los estándares internacionales y de esta forma, contribuir a facilitar su distribución y por lo tanto, en el ahorro de costos de producción.

Tomando en cuenta que existen varios estándares internacionales en la creación de OA, la aplicación debe de proporcionar al usuario, la oportunidad de poder seleccionar un estándar entre varios, para que bajo sus normas se cree el OA o se verifique.

5.1. Arquitectura General del sistema

En el diagrama de la arquitectura general se utiliza cierta notación tal como se muestra en la figura 5.1:

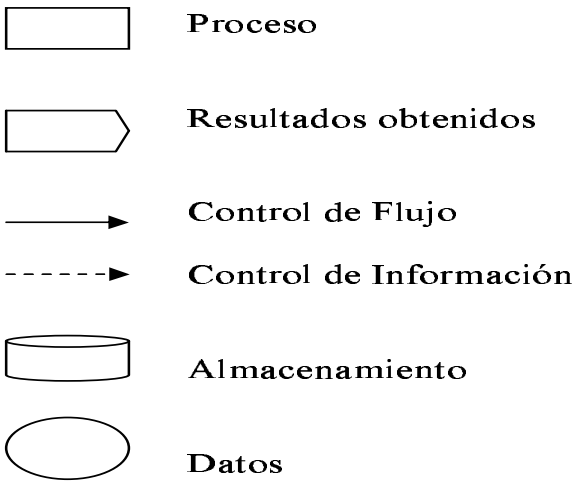

Figura 5.1: Notación del diagrama de la arquitectura general del sistema

La arquitectura general del sistema se describe en la figura 5.2 en donde:

- El proceso *Selector de Estándar*, es bajo el cual, se han de crear los metadatos. La aplicación da la oportunidad de elegir entre los siguientes estándares: LOM, SCORM 1.2, SCORM 2004, IMS y DUBLIN CORE, una vez seleccionado el estándar, se configuran las plantillas que definen el contenido del archivo XML de los metadatos.

- La configuración de las plantillas para la creación de los metadatos repercute directamente en el proceso que indica el *Editor* de los mismos. En este proceso, es donde la aplicación solicita al usuario la información que servirá para generar el archivo XML. El proceso *Editor* se divide en nueve fases las cuales hacen refernecia

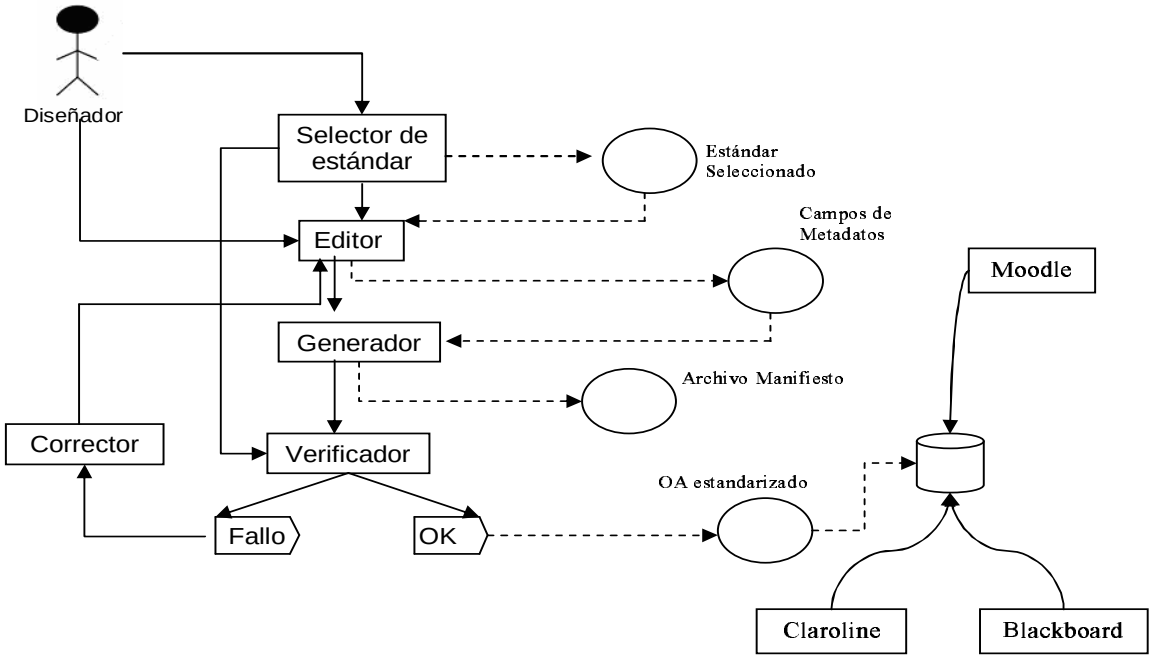

Figura 5.2: Arquitectura General del Sistema

a la división de los metadatos de LOM [11]. Cada una correspondiente a cada área de metadatos en las que los ha dividido el LOM. Al haber llevado a cabo la selección del estándar, estas plantillas de edición se han configurado nuevamente llenando automáticamente algunos campos y activando o desactivando algunos otros.

- Cuando el usuario indica que se ha concluido el proceso *Editor*, la aplicación activa un proceso llamado *Generador*; mientras este proceso permanezca activo, quiere decir que la aplicación se encuentra generando el archivo XML, el cual contiene la información correspondiente a los metadatos que el usuario desea obtener. El *Generador*, comprende el crear un archivo XML apegado a los esquemas de los estándares internacionales; al concluir la generación, tendrá como entrada un parámetro que indique que la generación del archivo ha terminado y terminará el

proceso. Cuando eso suceda, el archivo XML existirá en el directorio que el usuario indique.

- La fase de la *Generación* ha concluido y para asegurarnos que los metadatos cumplen con los estándares internacionales, la aplicación cuenta con un proceso llamado *Verificador*. Un factor importante que hace necesario el proceso *Verificador*, es que existen muchos OA, que no están estandarizados y en el caso de requerirlos para su uso, intercambio, venta, etc, es importante saberlo. Para poder entrar en este estado, previamente el usuario debe hacer la selección de un estándar, para que la aplicación conozca las normas bajo las cuales verificará a los metadatos. Cuando la selección se ha llevado a cabo y el usuario así lo indique, la aplicación ejecutará el proceso *Verificador*, el cual, revisará que cada una de las nueve áreas de los metadatos especificados por LOM, cumplan con los valores que indique el estándar seleccionado. La verificación de cada área, devolverá un resultado, el cual indique si el área correspondiente cumple con las especificaciones del estándar seleccionado. Los resultados de cada área se coleccionan para mostrarlos al usuario cuando la verificación llegue a su fin. La aplicación terminará el proceso verificador cuando reciba un parámetro que indique que la verificación de los metadatos ha terminado.

- Posteriormente presentará el resultado al usuario. Para el caso de haber concluido la verificación de manera no exitosa, la aplicación conducirá al usuario al proceso de *Corrector*.

- El proceso de *Corrector* indica al usuario las modificaciones que necesita llevar a cabo sobre cada una de las áreas de los metadatos y finalmente lo llevará nuevamente al proceso de *Editor*.

- Para el caso de haber concluido exitosamente, el OA, estará estandarizado y podrá ser usado por diferentes LMS

5.2. Casos de Uso

Los desarrolladores crean un modelo de análisis utilizando un modelo de casos de uso como entrada. Cada caso de uso en el modelo se traducirá en una realización de caso de uso en el modelo de análisis. El resultado del flujo de trabajo del análisis es el modelo de análisis, que es un modelo de objetos conceptual que analiza los requisitos mediante su refinamiento y su estructuración.

El modelo de análisis ofrece una especificación más precisa de los requisitos que la que se tiene como resultado de la captura de los mismos. El modelo de análisis se describe usando el lenguaje de los desarrolladores y puede por tanto introducir un mayor formalismo y ser utilizado para razonar sobre los funcionamientos internos del sistema. Además el modelo de análisis estructura los requisitos de un modo que facilita su comprensión, su preparación, su modificación y su mantenimiento. Al comprender la importancia de los metadatos y sabiendo las ventajas que nos pueden ofrecer, la aplicación que se pretende desarrollar tendrá que asegurar que al menos ciertos campos de los metadatos serán cubiertos por el usuario dependiendo del estándar seleccionado para su implementación. Para describir los casos de uso la aplicación general se divide en dos fases, la primera que es donde se generan los metadatos y la segunda, donde se verifican.

5.2.1. Generación de metadatos

El diagrama de casos de uso para esta fase se muestra en la figura 5.3. En él se observan dos actores, el Diseñador y los estándares, el actor tiene acceso a los servicios que son: Selección, Ayuda, Edición y Generación; por otro lado el actor estándares, es de donde se va a tomar la información necesaria para realizar los servicios de manera exitosa.

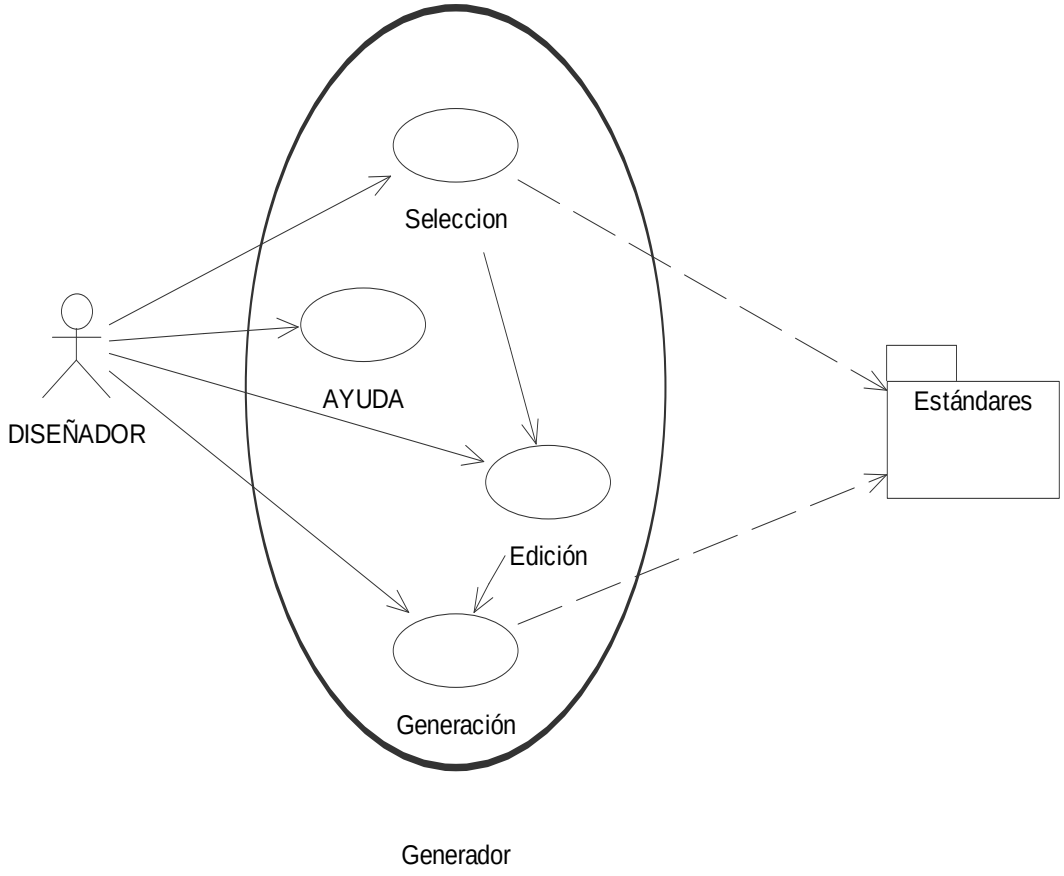

Figura 5.3: Caso de uso de la fase de generación de metadatos

Actor: Diseñador.

Función: El actor *Diseñador* es el usuario principal de la aplicación en la

generación de los metadatos; éste podrá realizar una selección del estándar bajo el cual desee generar sus metadatos, así mismo podrá editar la información que pertenecerá a los metadatos en los campos correspondientes y tendrá acceso al apartado de ayuda en caso de requerirla. Este actor, también podrá indicarle a la aplicación que genera los metadatos en archivo XML cuando considere que ha ingresado la información correspondiente.

- **Caso de uso:** Selección
  - **Actor:** Diseñador.
  - **Objetivo:** Este caso de uso permitirá al usuario seleccionar el estándar bajo el cual se generarán los metadatos.
  - **Relaciones:** Con el paquete *Estándares internacionales*, el cual solamente proporciona la información acerca de que campos de metadatos deberán al menos ser proporcionados.
  - **Alcances:** Este caso de uso permitirá la selección de un estándar entre cuatro, los cuales son: IMS, SCORM, Dublin Core, LOM.
  - **Restricciones:** El actor no podrá incluir otro estándar.

- **Caso de uso:** Ayuda
  - **Actor:** Diseñador.
  - **Objetivo:** Este caso de uso permitirá al usuario obtener ayuda del sistema; para obtener ayuda de los campos o de cada área de clasificación de los metadatos, sólo bastará con dar clic sobre ellos.
  - **Relaciones:** No tiene.

- **Alcances:** Este caso de uso permitirá al usuario obtener información de ayuda acerca de las características de los metadatos (campos y áreas que lo conforman).
- **Restricciones:** La ayuda esta restringida a la aplicación y a los campos de los metadatos pertenecientes a los 4 estándares contenidos.

- **Caso de uso:** Edición.
  - **Actor:** Diseñador.
  - **Objetivo:** Este caso de uso permitirá al actor diseñador editar la información en los campos que formará parte de los metadatos.
  - **Relaciones:** Con el caso de uso de *Selección*, la relación esta basada en que dependiendo de la selección realizada los campos obligatorios de los metadatos cambiarán.
  - **Alcances:** Todos los campos de los metadatos son opcionales por lo tanto, se puede editar el contenido de ellos sin ninguna penalización, algunos campos podrán ser automatizados.
  - **Restricciones:** Los campos marcados como obligatorios no podrán dejarse en blanco, no todos los campos podrán ser automatizados.

- **Caso de uso:** Generación.
  - **Actor:** Diseñador.
  - **Objetivo:** Este caso de uso, permitirá al actor diseñador generar los metadatos en un archivo con formato XML que contenga la información previamente

proporcionada por el usuario.

- **Relaciones:** Caso de uso de *Edición*, debido a que la generación de los metadatos se llevará a cabo con la información provista desde el caso de uso Edición. Con el paquete *Estándares*, la relación se basa en la necesidad de obtener información de los estándares para poder generar el archivo XML de manera adecuada
- **Alcances:** El archivo en XML que contenga los metadatos, podrá ser leído por el LMS que será utilizado como plataforma por el Departamento de Educación a Distancia de la Universidad Autónoma de Puebla.
- **Restricciones:** Los paquetes que utiliza el LMS deberán estar en formato ZIP y este contendrá el archivo XML así como el objeto de aprendizaje. La aplicación no convertirá en formato ZIP a los archivos antes mencionados.

- **Actor:** Estándares.
  - **Acceso:** Selección.
  - **Objetivo:** Contendrá información correspondiente a los cuatro estándares internacionales para la creación de metadatos.
  - **Relaciones:** Caso de uso *Selección*, este caso de uso tendrá comunicacón con este actor para definir los campos obligatorios y opcionales para crear los metadatos adecuados.
  - **Alcances:** Este Actor comprende cuatro estándares internacionales, IMS, SCORM, Dublin Core y LOM
  - **Restricciones:** Sólo proporciona información acerca de los metadatos requeridos por el estándar; no proporciona información completa de los objetos

de aprendizaje dependiendo el estándar seleccionado.

5.2.2. Verificador de Metadatos

El diagrama de casos de uso para esta fase se muestra en la figura 5.4. En el diagrama se muestra al actor *Diseñador*, el cual tiene acceso a los casos de uso Selección, Verifica, Respuesta y Correcciones. También se muestra el actor estándares que es de donde se obtiene la información para realizar los casos de uso.

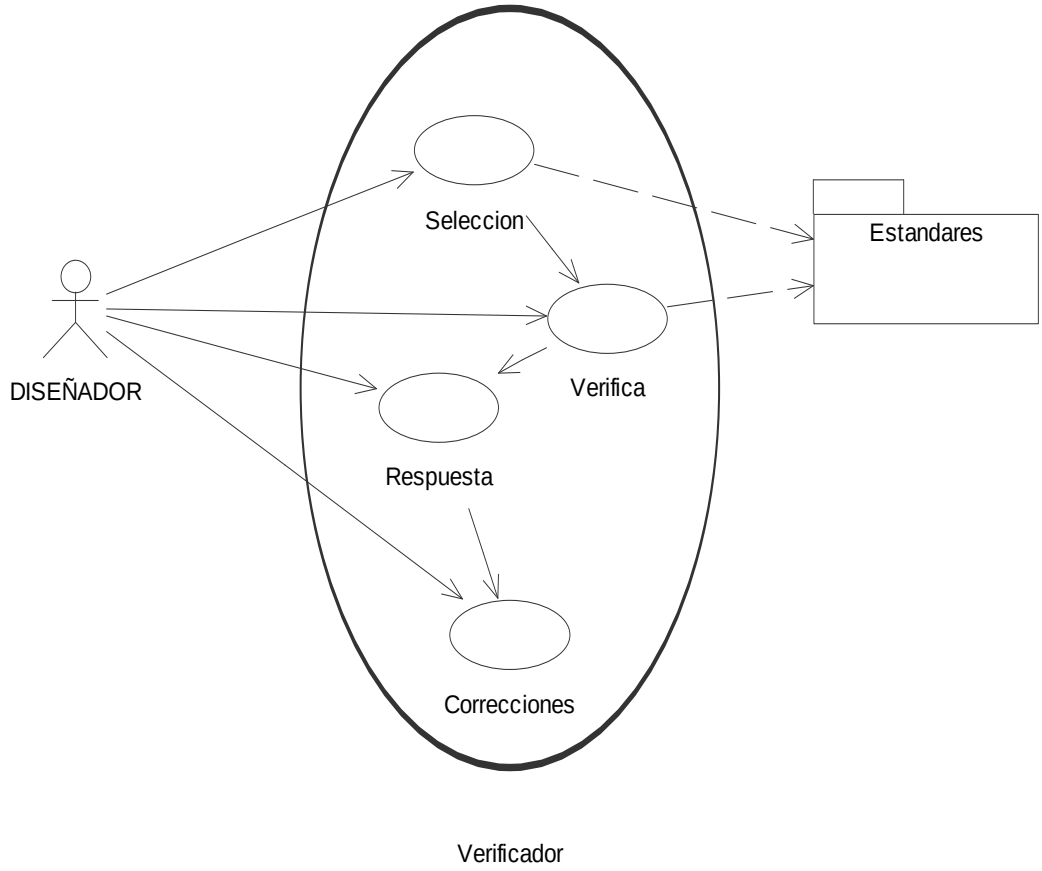

Figura 5.4: Caso de uso de la fase de Verificación de metadatos

Actor: Diseñador.

Función: El actor *diseñador* podrá seleccionar el estándar que requiere para la verificación de los metadatos, al terminar la verificación podrá tener acceso a los resultados producidos y en caso de que el resultado indique que los metadatos no son los adecuados, podrá tener acceso a información que le proporcione las correcciones respectivas.

- **Caso de uso:** Selección
  - **Actor:** Diseñador.
  - **Objetivo:** Permitir al actor diseãdor seleccionar el estándar bajo el cual se verificarán los metadatos.
  - **Relaciones:** Con el actor *estándares internacionales*, los cuales proporcionan la información acerca de que campos deberán de contener los metadatos a revisión.
  - **Alcances:** Este caso de uso permitirá la selección de un estándar entre cuatro, los cuales son: IMS, SCORM, Dublin Core, LOM.
  - **Restricciones:** El actor no podrá incluir otro estándar.

- **Caso de uso:** Verifica
  - **Actor:** Diseñador
  - **Objetivo:** Verificar que los metadatos en cuestión contengan la información mínima requerida por el estándar seleccionado.
  - **Relaciones:** Con el caso de uso *Selección*, la verificación se llevará a cabo en dependencia del estándar elegido en el caso de uso Selección, también tendrá una relación con el actor *Estándares* debido a que deberá basarse en

ellos para poder realizar una verificación correcta. Con el caso de uso *Respuesta*, ésta relación está dada por que el caso de uso Verifica devolverá una respuesta al actor diseñador y eso lo hará en base a este caso de uso.

- **Alcances:** Verificará que el archivo XML que incluye a los metadatos, contenga la información mínima requerida por el estándar seleccionado.
- **Restricciones:** Solamente se verificará la información referida a los metadatos. El caso de uso Verificación solamente tomará en cuenta los archivos XML que sean generados bajo los esquemas de los estándares mencionados.

- **Caso de uso:** Respuesta
  - **Actor:** Diseñador
  - **Objetivo:** Proporcionar al actor diseãdor un reporte claro de la operación llevada a cabo con los metadatos, en el caso de que los metadatos estén correctos se indicará por medio de un mensaje que pueden ser incluidos al objeto de aprendizaje, de lo contrario indicará la revisión de las correcciones.
  - **Relaciones:** Con el caso de uso Verifica, la respuesta se obtendrá con los resultados que arroje el caso de uso Verificación.
  - **Alcances:** Notificará al usuario el resultado de la evaluación de sus metadatos.
  - **Restricciones:** En caso de que los metadatos no sean los correctos, no mostrará correcciones, estas se mostrarán en el caso de uso Correcciones.

- **Caso de uso:** Correcciones
  - **Actor:** Diseñador

- **Objetivo:** Mostrar al actor diseãdor las correcciones necesarias para que los metadatos puedan ser incluidos al objeto de aprendizaje.
- **Relaciones:** Con el caso de uso Respuesta, si el resultado por parte del caso de uso Respuesta es negativo, permitirá al caso de uso Correcciones, mostrar una lista de campos de metadatos que no están debidamente complementados o si el archivo XML, no cumple con los esquemas de los estándares.
- **Alcances:** Notificará al usuario una serie de modificaciones recomendadas para que los metadatos sean adecuados.
- **Restricciones:** Solo mostrará los campos que sean requeridos para que los metadatos sean aceptables, si el archivo XML no cumple con los esquemas de los estándares, recomendará hacerlo nuevamente.

5.3. Diagramas de Clases

La construcción de un diagrama de clases toma en cuenta la identificación de las clases que deben existir en el sistema. Las clases en los lenguajes orientados a objetos pueden verse como una manera comoda de escribir una colección de objetos que tienen las mismas propiedades. Las estructuras o propiedades de la clase se conocen como atributos y el comportamiento como métodos. Un método también es conocida como método, es un servicio proporcionado por la clase que puede ser usado por otras clases. Estas operaciones pueden tener argumentos, cada uno de un tipo y también pueden devolver resultados. Un diagrama de clase representa las clases del sistema con sus relaciones y estructuras de herencia.

El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones. En primera instancia en la figura 5.5 se presenta un diagrama de clases el cual muestra de manera general las clases que se consideran para el modelo de diseño. En el cual se representa por medio de una clase llamada.

La clase SelEstándar representa la actividad que tendrá el usuario al llevar a cabo la selección del estándar de trabajo. La clase SelEstándar está relacionada a las clases Áreas y Verifica; en la clase Áreas es donde se recopilará la información perteneciente a los metadatos de cada Área o categoría especificada en los metadatos.

Generar XML que es la que se encarga de generar el archivo XML que contiene los metadatos, contiene una relación con la clase SelEstándar porque dependiendo del estándar seleccionado es como se creará el archivo XML.

La clase Verifica es la que realizará la verificación de archivos XML que contienen metadatos para saber si están construidos de acuerdo con los esquemas de los estándares internacionales; finalmente, la clase Verifica informa al usuario el resultado de la revisión y lo presenta a través de la clase Respuesta. Para el caso de que los metadatos no cumplan con el estándar predeterminado la clase Correcciones es la encargada de indicarle las modificaciones que tiene que hacer para que los metadatos sean los correctos. La clase Verifica hereda algunos atributos de la clase SelEstándar.

Las clases Respuesta, SelecciónArchivo y Corrector tienen una relación con la clase Verifica. Estas tres clases complementan el proceso Verificador proporcionando información acerca del archivo a verificar y posteriormente muestran los resultados obtenidos.

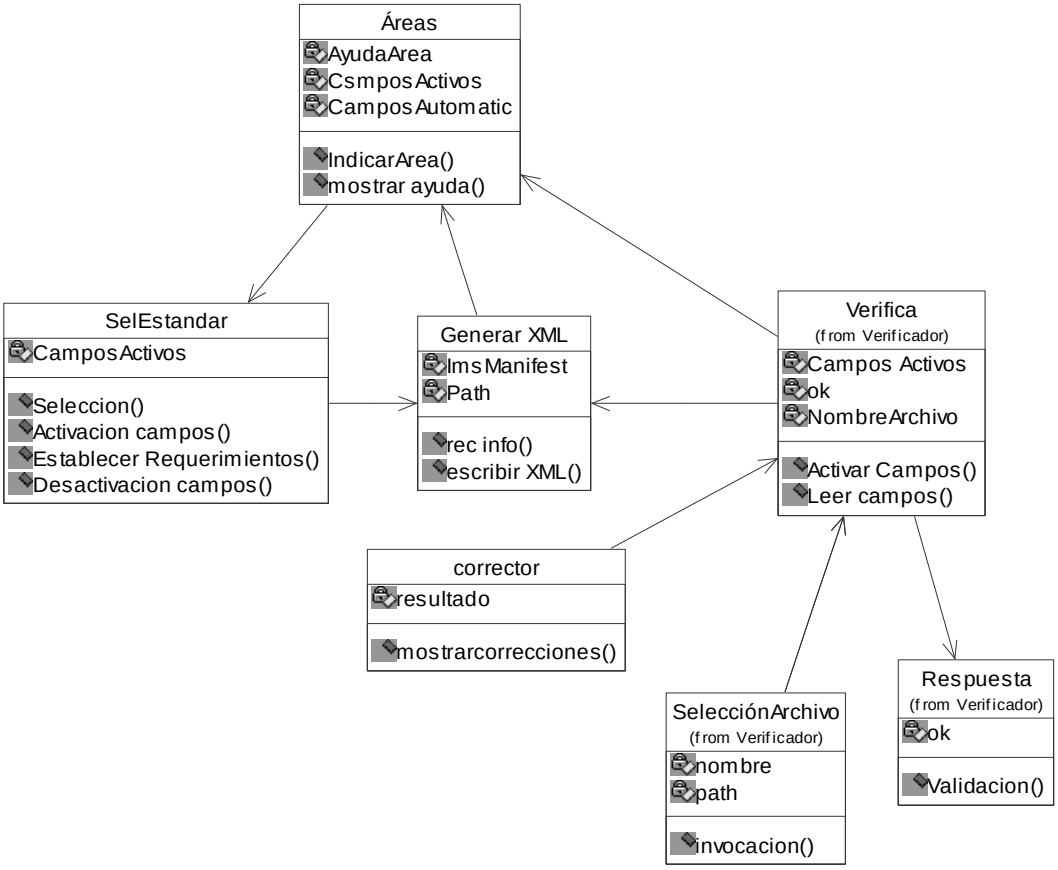

Figura 5.5: Diagrama de Clases General

Capítulo 6

Modelo de Diseño

6.1. Diagrama de Clases Jerárquico

El diagrama de clases jerárquico tiene una mayor aproximación a lo que será el diseño de la aplicación. En el modelo jerárquico se representan en 3 niveles importantes:

- Interfaz con el usuario
- Control y Lógica
- Comunicación y Almacenamiento

En el modelo de análisis, se plantea una posible solución al conflicto de la generación adecuada de metadatos por parte del usuario y asegurar que los metadatos que formen parte de un objeto de aprendizaje que esté destinado a permanecer a un repositorio, sean los requeridos por el estándar utilizado.

Para ello se contemplaron dos fases principales que son:

- La generación de metadatos y
- La verificación de metadatos

De esta forma, el actor tiene la facilidad de crear metadatos en formato XML bajo los esquemas de los estándares internacionales sin necesidad de ser experto en ciertas áreas de cómputo y también sin tener que utilizar directamente el código XML requerido para su construcción y los repositorios, tienen la garantía de que los objetos de aprendizaje incluyen metadatos basados en los estándares internacionales para facilitar su administración.

6.1.1. Generación de Metadatos

El diagrama de clases jerárquico mostrado en la figura 6.1, representa las clases contenidas en el diseño de la fase de la generación del archivo XML que contiene los metadatos.

6.1.2. Nivel de Interfaz

En el nivel de interfaz se indica el ambiente con el que va a interactuar el usuario y la herramienta de verificación. Esto se representa por medio de clases, cada clase para cada actividad (de una manera muy detallada). La clase a la que tiene acceso el actor directamente le llamaremos *Interfaz*, esta clase es una clase compuesta, debido a que desde esta clase son llamadas otras más, a la clase interfaz, también va enlazada a una clase de ctrl, la cual tiene la funcción de mantener el enlace con la clase compuesta que está en otro nivel, el de CONTROL. La clase *Interfaz* es compuesta por lo que cuenta con otras clases distintas:

- Interfaz
- In
- Out

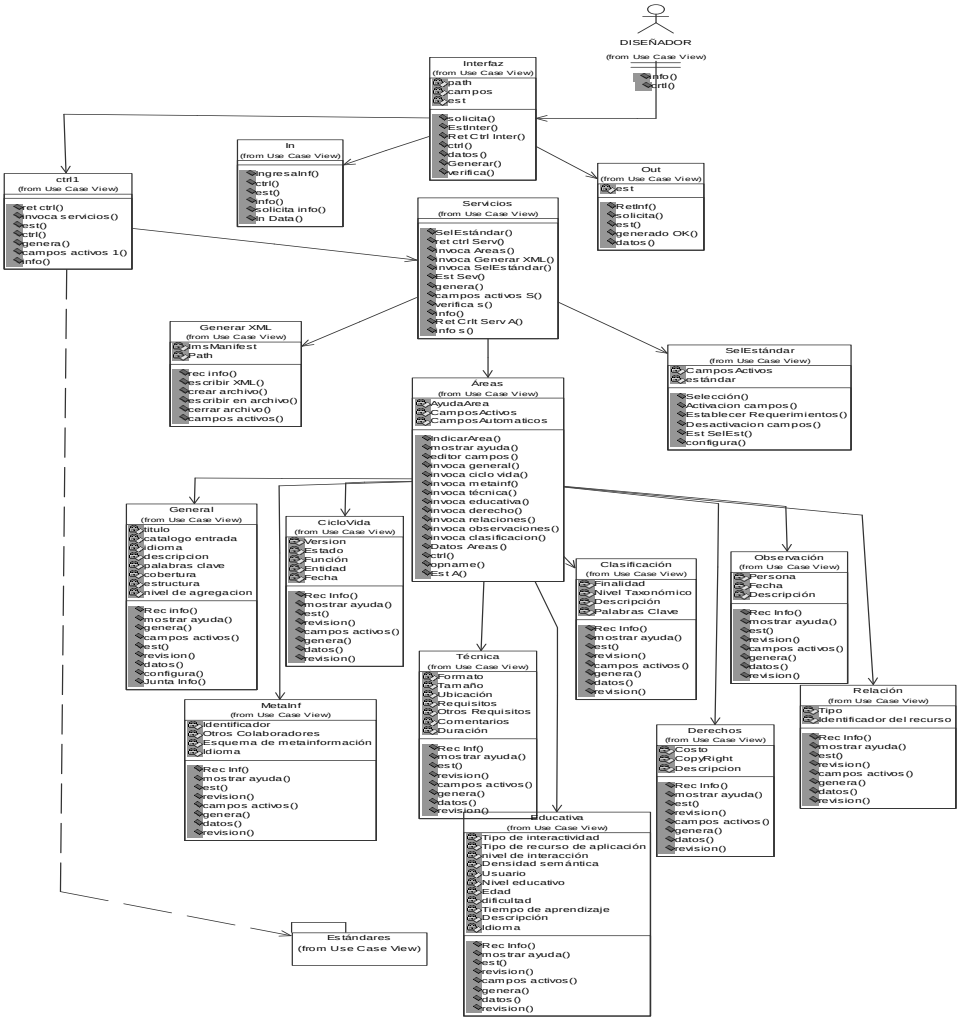

Figura 6.1: Diagrama de Clases Jerárquico. Fase Generación

La clase *Interfaz* es la que tiene el único contacto con el actor. Ésta se compone de las clases de *In* y *Out*. La clase *In* es por donde el actor proporcionará los valores a la aplicación para realizar alguna actividad OA. La clase *Out* es donde la aplicación podrá tener contacto con el actor.

La clase Interfaz contiene las siguientes operaciones:

- solicita.- Esta operación solicita la información requerida para algún proceso
- EstInter.- Esta operación corresponde al estándar seleciconado por el diseñador
- Ret Ctrl Inter.- Permite retornar el control
- ctrl.- Permite proporcionar el control
- datos.- Permite el envio de datos e información
- generar.- Esta operación permite la generación de los metadatos para los OA
- verifica.- Esta operación permite utilizar el verificador del sistema

6.1.3. Nivel de Control y Lógica

Este nivel es accedido por medio de la clase de ctrl1; aquí se encuentra la clase *Servicios* que es una clase compuesta, en ella se describen los servicios principales que ofrece el nivel de control al nivel de interfaz representado por medio de clases, cada una de ellas se compone por métodos que son los que detallan la manera en que se lleva a cabo la función.

La clase *Servicios* es compuesta por lo que cuenta con otras distintas para cada uno de los servicios que ofrece:

- SelEstándar
- Generar XML
- Áreas

Así mismo cuenta también con las siguientes operaciones:

- SelEstándar.- Permite seleccionar un estándar
- Ret Ctrl Serv.- Retorna el control a la clase ctrl1
- invoca áreas.- Llama a la clase áreas
- invoca generar XML.- Llama a la clase generar
- invoca Sel Estandar.- Llama a la clase SelEstandar
- Est Serv.- Permite conocer el estándar seleccionado
- genera.- Permite generar los metadatos
- campos activos S.- Permite activar los campos requeridos
- Verifica s.-Permite enviar el parametro que indica la verificación.
- info.- Permite el envio de información entre clases.
- Ret Ctrl Serv A.- Retorna el control a la clase ctrl1.
- Info s.- Retorna informacion a servicios

Clase: SelEstadar

Esta clase esta encargada de la selección del estándar. Y con ello, la activación de los campos mínimos requeridos para complementar la información de los metadatos.

Atributos:

- Campos Activos: Este atributo representa a los campos que deben de ser activados.
- Estándar: Almacena el valor correspondiente al estándar seleccionado.

Operaciones:

- Selección: En la operación Selección, se lleva a cabo la selección del estándar.
- Activación Campos: Para activar los campos mínimos requeridos por el estándar seleccionado se utiliza la operación Activación de Campos.
- Establecer Requerimientos: Lleva a cabo la aplicación de los criterios establecidos.
- Desactivación Campos: Lleva a cabo la desactivación de los campos que no serán necesarios.
- Est SelEst: Permite reconocer que estándar ha sido seleccionado.
- configura: Configura los campos que deben de ser completados

Clase: Generar XML

Esta clase es la encargada de generar el archivo XML, el cual contendrá los metadatos generados para cierto objeto de aprendizaje.

Atributos:

- ImsManifest: Este atributo refiere al nombre del archivo XML que será generado por la aplicación.

- Path: Este atributo refiere a la ubicación en el disco duro en donde será almacenado el archivo XML.

Operaciones:

- Rec Info: Esta operación se encarga de recopilar toda la información de los campos, la cual fue proporcionada por el usuario en las cajas de texto, posteriormente se utilizará para generar el archivo que contendrá los metadatos
- Escribir XML: Esta operación se encargará de escribir la información en un formato XML en un archivo externo.
- Crear archivo: Esta operación se encarga de crear el archivo físico en el disco duro.
- Escribir archivo: Esta operación efectúa la escritura del archivo XML.
- Cerrar archivo: Cuando el archivo ha sido terminado, debe de cerrarse utilizando esta operación.
- Campos activos: Establece que campos deben de estar activos dependiendo del estándar seleccionado.

Clase: Áreas

En esta clase es donde se llevará a cabo la administración de las 9 áreas en las que son divididos los metadatos. En esta clase se gestiona la manera en que se proporciona la información por parte del diseñador hacia la aplicación. La clase áreas, tiene ligada una clase por cada área de metadatos.

Atributos:

- Ayuda Área: La clase lleva a cabo la presentación de información de ayuda para el diseñador.
- Campos Activos: Los campos activos serán un atributo que permitan incluir información dentro del esquema de metadatos.
- Campos Automáticos: Este atributo contendrá la información correspondiente a los campos que puedan ser complementados automáticamente dependiendo de la información presentada por el diseñador.

Una observación importante es que estos atributos serán heredados a las clases de cada categoría.

Operaciones:

- Activación campos.- Indica que campos deben de estar activos.
- Datos Areas.- Permite el paso de parametros entre las áreas.
- Desactivación campos.- Indica que campos deben de estar desactivados.
- Est A.- Indica que estándar ha sido seleciconado.
- Est SelEst.- Permite conocer que estándar se ha seleccionado.
- Establecer Requerimientos.- Permite conocer que requerimientos son necesarios para los metadatos requeridos .
- Indicar área: Esta operación se encargará de enfocar hacia la clase del área correspondiente.

- Seleccion.- Envia la selección de algún estándar.
- Configura.- Configura los campos de metadatos.
- ctrl.- Retorna el control a otra clase.
- Mostrar ayuda: Esta operación permitirá a la clase mostrar ayuda para auxiliar al diseñador en la manera de utilizar la información.
- Editor Campos: Permite introducir datos en los campos de metadatos.
- invoca general.- Ejecuta la clase General.
- invoca ciclo vida.- Llama la clase Ciclo Vida.
- invoca metainf.- Llama la clase Metainf.
- invoca técnica.- Llama la clase Técnica.
- invoca educativa.- Llama la clase Educativa.
- invoca derecho.- Llama la clase Derechos.
- invoca relaciones.- Llama la clase Relación.
- invoca observaciones.- Llama la clase Observación.
- invoca clasificación.- Llama la clase Clasificación.
- Datos.- Información que puede ser recuperada y enviada a la clase interfaz

Clase: General

Esta clase es la encargada de administrar la información correspondiente al área de metadatos *General*, en ella se introduce la información correspondiente a dicha área.

Atributos: Los siguientes atributos almacenan los valores correspondientes a los metadatos

- Título
- Catálogo de entrada
- Idioma
- Descripción
- Palabras clave
- Cobertura
- Estructura
- Nivel de agregación

Operaciones:

- Rec Info: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla
- Mostrar Ayuda: Esta operación se encarga de mostrarle la ayuda necesaria al diseñador cuando la solicite
- Genera: Genera los metadatos en XML correspondientes al área general

- Campos Activos: Reconoce que campos deberán estar activos para poder ser editados
- est:Permite reconocer que estándar ha sido seleccionado.
- Revisión: Verifica que los campos esten complementados correctamente
- datos.- Permite el envío de datos entre clases.
- configura.- Configura los campos de metadatos para esta fase.
- junta info.- Recopila información referente a esta área.

Clase: Ciclo Vida

Esta clase es la encargada de administrar la información correspondiente al área de metadatos Ciclo de vida, en ella se introduce la información correspondiente a dicha área

Atributos: Los atributos listados almacenan los valores de los metadatos

- Versión
- Estado
- Función
- Entidad
- Fecha

Operaciones:

- Rec Info: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla
- Mostrar Ayuda: Esta operación se encarga de mostrarle la ayuda necesaria al diseñador cuando la solicite
- Campos Activos: Reconoce que campos deberán estar activos para poder ser editados
- est:Permite reconocer que estándar ha sido seleccionado.
- Genera: Genera los metadatos en XML correspondientes al área general
- Revisión: Verifica que los campos esten complementados correctamente
- datos.- Permite el envío de datos entre clases.

Clase: MetaInf

Esta clase es la encargada de administrar la información correspondiente al área de metadatos Meta información, en ella se introduce la información correspondiente a dicha área.

Atributos: Estos atributos almacenan los valores correspondientes a los metadatos

- Identificador
- Otros colaboradores
- Esquema de meta información
- Idioma

Operaciones:

- Rec Inf: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla
- Mostrar Ayuda: Esta operación se encarga de mostrarle la ayuda necesaria al diseñador cuando la solicite
- Campos Activos: Reconoce que campos deberán estar activos para poder ser editados
- Revisión: Verifica que los campos esten complementados correctamente
- Genera: Genera los metadatos en XML correspondientes al área general
- datos.- Permite el envío de datos entre clases.
- est:Permite reconocer que estándar ha sido seleccionado.

Clase: Técnica

Esta clase es la encargada de administrar la información correspondiente al área de metadatos Técnica, en ella se introduce la información correspondiente a dicha área.

Atributos: Estos atributos almacenan los valores correspondientes a los metadatos

- Formato
- Tamaño
- Ubicación

- Requisitos
- Otros requisitos
- Comentarios
- Duración

Operaciones:

- Rec Inf: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla
- Mostrar Ayuda: Esta operación se encarga de mostrarle la ayuda necesaria al diseñador cuando la solicite
- Campos Activos: Reconoce que campos deberán estar activos para poder ser editados
- est:Permite reconocer que estándar ha sido seleccionado.
- Genera: Genera los metadatos en XML correspondientes al área general
- datos.- Permite el envío de datos entre clases.
- Revisión: Verifica que los campos esten complementados correctamente

Clase: Educativa

Esta clase es la encargada de administrar la información correspondiente al área de metadatos Educativa, en ella se introduce la información correspondiente a dicha área.

Atributos: Los siguientes atributos almacenan los valores correspondientes a los metadatos

- Tipo de interactividad
- Tipo de recurso de aplicación
- Tipo de interacción
- Densidad semántica
- Usuario
- Nivel educativo
- Edad
- Dificultad
- Tiempo de aprendizaje
- Descripción
- Idioma

Operaciones:

- Rec Info: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla
- Mostrar Ayuda: Esta operación se encarga de mostrarle la ayuda necesaria al diseñador cuando la solicite

- Campos Activos: Reconoce que campos deberán estar activos para poder ser editados
- est:Permite reconocer que estándar ha sido seleccionado.
- Genera: Genera los metadatos en XML correspondientes al área general
- datos.- Permite el envío de datos entre clases.
- Revisión: Verifica que los campos esten complementados correctamente

Clase: Derechos

Esta clase es la encargada de administrar la información correspondiente al área de metadatos Derechos, en ella se introduce la información correspondiente a dicha área.

Atributos: Los atributos de esta clase, al igual que el de las clases correspondientes a cada área, almacenan los valores correspondientes a los metadatos

- Costo
- Copyright
- Descripción

Operaciones:

- Rec Info: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla

- Mostrar Ayuda: Esta operación se encarga de mostrarle la ayuda necesaria al diseñador cuando la solicite
- Campos Activos: Reconoce que campos deberán estar activos para poder ser editados
- est:Permite reconocer que estándar ha sido seleccionado.
- Genera: Genera los metadatos en XML correspondientes al área general
- datos.- Permite el envío de datos entre clases.
- Revisión: Verifica que los campos esten complementados correctamente

Clase: Relación

Esta clase es la encargada de administrar la información correspondiente al área de metadatos Relación, en ella se introduce la información correspondiente a dicha área.

Atributos: Los atributos almacenan los valores correspondientes a los metadatos

- Tipo
- Identificador del recurso

Operaciones:

- Rec Info: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla

- Mostrar Ayuda: Esta operación se encarga de mostrarle la ayuda necesaria al diseñador cuando la solicite
- Genera: Genera los metadatos en XML correspondientes al área general
- est:Permite reconocer que estándar ha sido seleccionado.
- Revisión: Verifica que los campos esten complementados correctamente
- datos.- Permite el envío de datos entre clases.
- Campos Activos: Reconoce que campos deberán estar activos para poder ser editados

Clase: Observación

Esta clase es la encargada de administrar la información correspondiente al área de metadatos Observación, en ella se introduce la información correspondiente a dicha área.

Atributos: Estos atributos almacenan los valores correspondientes a los metadatos

- Persona
- Fecha
- Descripción

Operaciones:

- Rec Info: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla

- Mostrar Ayuda: Esta operación se encarga de mostrarle la ayuda necesaria al diseñador cuando la solicite
- Genera: Genera los metadatos en XML correspondientes al área general
- est:Permite reconocer que estándar ha sido seleccionado.
- Revisión: Verifica que los campos esten complementados correctamente
- datos.- Permite el envío de datos entre clases.
- Campos Activos: Reconoce que campos deberán estar activos para poder ser editados

Clase: Clasificación

Esta clase es la encargada de administrar la información correspondiente al área de metadatos Clasificación, en ella se introduce la información correspondiente a dicha área.

Atributos: Estos atributos almacenan los valores correspondientes a los metadatos

- Finalidad
- Nivel Taxonómico
- Descripción
- Palabras Clave

Operaciones:

- Rec Info: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla
- Mostrar Ayuda: Esta operación se encarga de mostrarle la ayuda necesaria al diseñador cuando la solicite
- Genera: Genera los metadatos en XML correspondientes al área general
- est:Permite reconocer que estándar ha sido seleccionado.
- Revisión: Verifica que los campos esten complementados correctamente
- datos.- Permite el envío de datos entre clases.
- Campos Activos: Reconoce que campos deberán estar activos para poder ser editados

6.1.4. Almacenamiento

Este nivel contiene un Paquete que representa a la información almacenada correspondiente a los estándares internacionales y del cual se va a obtener la información para realizar las actividades necesarias.

6.1.5. Verificación de Metadatos

El diagrama de clases jerárquico, mostrado en la figura 6.2, presenta las clases contenidas en el diseño de la fase de la verificación del archivo XML que contiene los metadatos. Las clases del nivel de Interfaz son las mismas utilizadas para la fase de *Generación de Metadatos*, las clases que difieren en este diagrama jerárquico es en el nivel de *Control ó Lógica*

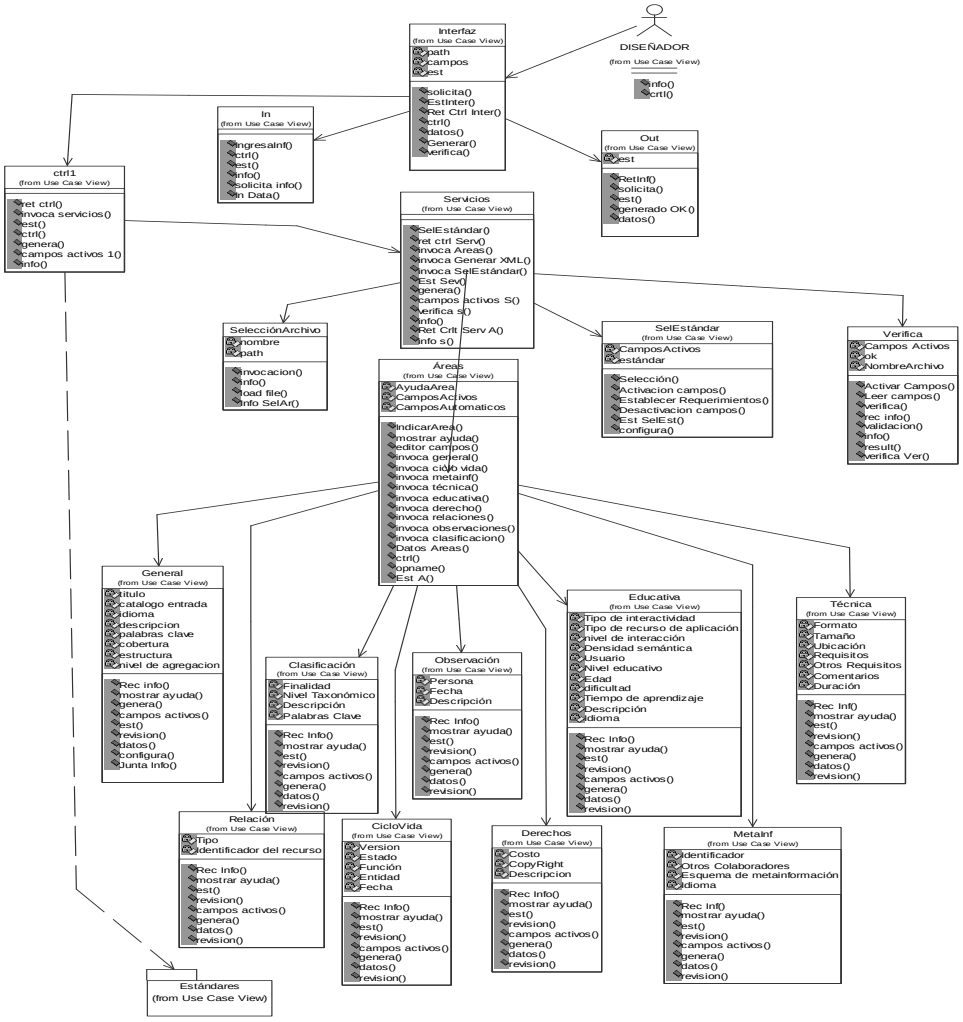

Figura 6.2: Diagrama de Clases Jerárquico. Fase Verificación

6.1.6. Nivel de Interfaz

La clase a la que tiene acceso el diseñador directamente es la clase *Interfaz*, esta clase es una clase compuesta, también va enlazada a una clase de ctrl, la cual tiene la funcción de mantener el enlace con la clase compuesta que esta en otro nivel, el de CONTROL O LÓGICA. La clase de *Interfaz* es compuesta por

- La clase *In*

- La clase *Out*

La clase Interfaz contiene las siguientes operaciones:

- solicita.- Esta operación solicita la información requerida para algún proceso
- EstInter.- Esta operación corresponde al estándar seleciconado por el diseñador
- Ret Ctrl Inter.- Permite retornar el control
- ctrl.- Permite proporcionar el control
- datos.- Permite el envio de datos e información
- generar.- Esta operación permite la generación de los metadatos para los OA
- verifica.- Esta operación permite utilizar el verificador del sistema

6.1.7. Nivel de control o lógica

En este nivel es accedido por medio de la clase de ctrl1; aquí se encuentra la clase *Servicios* que es una clase compuesta, en ella se describen los servicios principales que ofrece el nivel de Control al nivel de Interfaz representado por medio de clases.

La clase *Servicios* es compuesta por lo que cuenta con otras distintas para cada uno de los servicios que ofrece:

- SelecciónArchivo
- Áreas

- Verifica
- SelEstándar

Así mismo cuenta también con las siguientes operaciones:

- SelEstándar.- Permite seleccionar un estándar
- Ret Ctrl Serv.- Retorna el control a la clase ctrl1
- invoca áreas.- Llama a la clase áreas
- invoca generar XML.- Llama a la clase generar
- invoca Sel Estandar.- Llama a la clase SelEstandar
- Est Serv.- Permite conocer el estándar seleccionado
- genera.- Permite generar los metadatos
- campos activos S.- Permite activar los campos requeridos
- Verifica s.-Permite enviar el parametro que indica la verificación.
- info.- Permite el envio de información entre clases.
- Ret Ctrl Serv A.- Retorna el control a la clase ctrl1.
- Info s.- Retorna informacion a servicios

Selección Archivo

La clase Selección Archivo, permite al diseñador indicarle a la aplicación la ubicación del archivo que desee verificar

Atributos:

- Nombre: Especifica el nombre del archivo a revisar.
- path: Especifica la ubicación del archivo

Operaciones:

- Invocación: Con esta operación se puede cargar en memoria el archivo deseado.
- Activar Campos.- Efectua la activación de los campos requeridos.
- Leer Campos.- Permite llevar a cabo la lectura de los campos escritos.
- Verifica.- Realiza una verificación de los campos leidos.
- Rec info.- Recupera información de diversas clases.
- Validación.- Valida que los campos sean correctos.
- Info.- Permite el paso de información entre clases.
- Result.- Permite enviar el resultado de la verificación de los metadatos.
- Verifica Ver.-Permite enviar el resultado de la verificación de los metadatos a la clase verifica.
- load file.- Permite cargar un archivo.
- info SelAr.-Permite el paso de la información hacia la clase de selección de estándar.
- info: Permite el envio de información entre clases.

Clase: SelEstadar

Esta clase ya ha sido descrita anteriormente en el diagrama de clases del generador

Clase: Verifica

El desarrollo de la verificación inicia principalmente en esta clase, esta clase está relacionada con las clases de cada área las cuales realizarán la revisión de su categoría correspondiente y devolverán el resultado a la clase Verificación; posteriormente con una colección de resultados se obtendrá el resultado final

Atributos:

- Ok: Este atributo será verdadero siempre y cuando se cumplan los metadatos mínimos indicados por el estándar.
- Nombre Archivo: Aquí se representa el nombre del archivo en revisión.
- Campos activos: Se encarga de establecer los campos que se deben de verificar.

Operaciones:

- Activar campos: Esta operación activa los campos mínimos registrados dependiendo el estándar seleccionado
- Leer campos: Esta operación se encarga de la lectura de los campos; dependiendo del área que tenga que verificar realizará la invocación de otra clase
- verifica: Permite la verificación de los campos de metadatos

- rec info: Esta operación es la encargada de la lectura del archivo XML en el área correspondiente para así poder evaluarla
- validacion: Declara que el resultado de la verificación es correcto
- info:Permite el envio de información entre clases.
- Result.- Devuelve un resultado
- Verifica ver.- Permite el paso de la verificación

Las clases *General, Ciclo de Vida, Meta Información, Técnica, Educativa, Derechos, Relación, Observación y Clasificación*, contienen la misma descripción que las del diagrama de clsaes jerárquico para la generación de metadatos.

6.1.8. Almacenamiento

En este nivel existe un paquete que contiene a la información correspondiente a los estándares internacionales, de este paquete es de donde se va a obtener la información para efectúar las actividades necesarias.

6.2. Diagramas de Secuencia

En los diagramas de secuencia se muestran las interacciones de objetos mediante transferencia de mensajes entre ellos o subsistemas. Un diagrama de secuencia representa una interacción como un gráfico bidimensional, la dimensión vertical es el eje de tiempo que avanza hacia debajo de la página. La dimensión horizontal muestra los roles de clasificador que representan objetos individuales en la colaboración. Cada rol de clasificador se representa mediante una columna vertical-línea de vida. Durante

el tiempo que existe el objeto, el rol se muestra por una línea discontinua. Durante el tiempo que dura una activación de un procedimiento en el objeto, la línea de vida se dibuja como una línea doble.

Cuando se dice que un subsistema recibe un mensaje, quiere decir que es un objeto de una clase del subsistema el que recibe el mensaje. Cuando un subsistema envía un mensaje, realmente es un objeto de una clase del subsistema que lo envía.

6.2.1. Diagrama de secuencia: Selección del Estándar

En la figura 6.3, se muestra el diagrama de Secuencia para el caso de uso selección del estándar.

1. En primer lugar el actor solicita por medio de la clase Interfaz (que es la clase que interactua con el actor) la selección de un estándar. Hecho esto la clase Out regresa a la clase Interfaz.

2. La clase Interfaz, otorga el control a la clase In para que el actor realice la selección del estándar, cuando lo hace, devuelve el control a la clase Interfaz con el argumento que contienen el estándar seleccionado.

3. Interfaz da el control y el argumento que indica que estándar ha sido seleccionado a la clase ctrl1 (que es la que tiene contacto con el nivel de lógica), esta a su vez invoca a la clase Servicios.

4. La clase Servicios invoca a la clase adecuada para satisfacer las peticiones del diseñador e invoca a la clase SelEstándar.

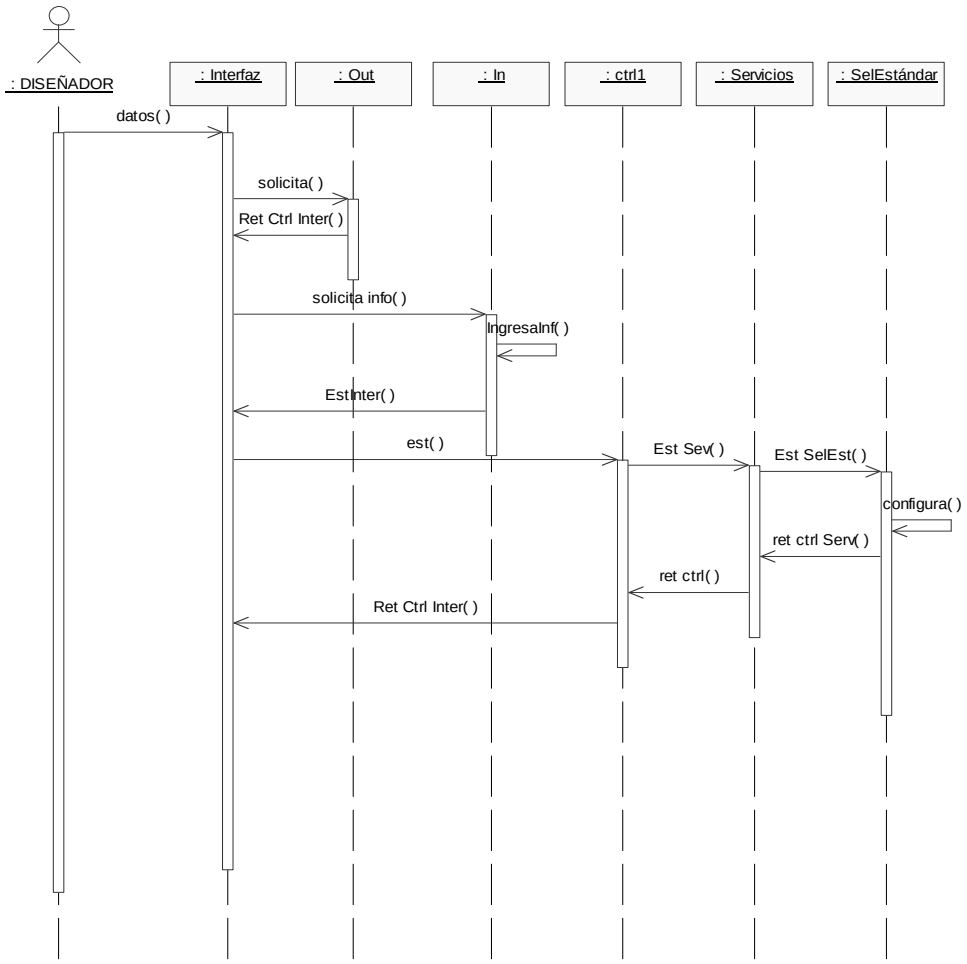

Figura 6.3: Diagrama de secuencia que realiza la Selección del estándar

5. En la clase SelEstándar se lleva a cabo la configuración de las plantillas que se deben de adoptar dependiendo el estándar seleccionado, cuando esta actividad termina, devuelve el control a la clase Servicios y esta a su vez a ctrl1 y para concluir el proceso de selección del estándar, la clase ctrl1 devuelve el control a la clase Interfaz.

6.2.2. Diagrama de secuencia: Edición

En la figura 6.4, se muestra el diagrama de secuencia el cual representa el proceso de edición de la información de los campos referentes a los elemento de los metadatos. Este diagrama muestra un ejemplo de edición exclusivamente de la clase General.

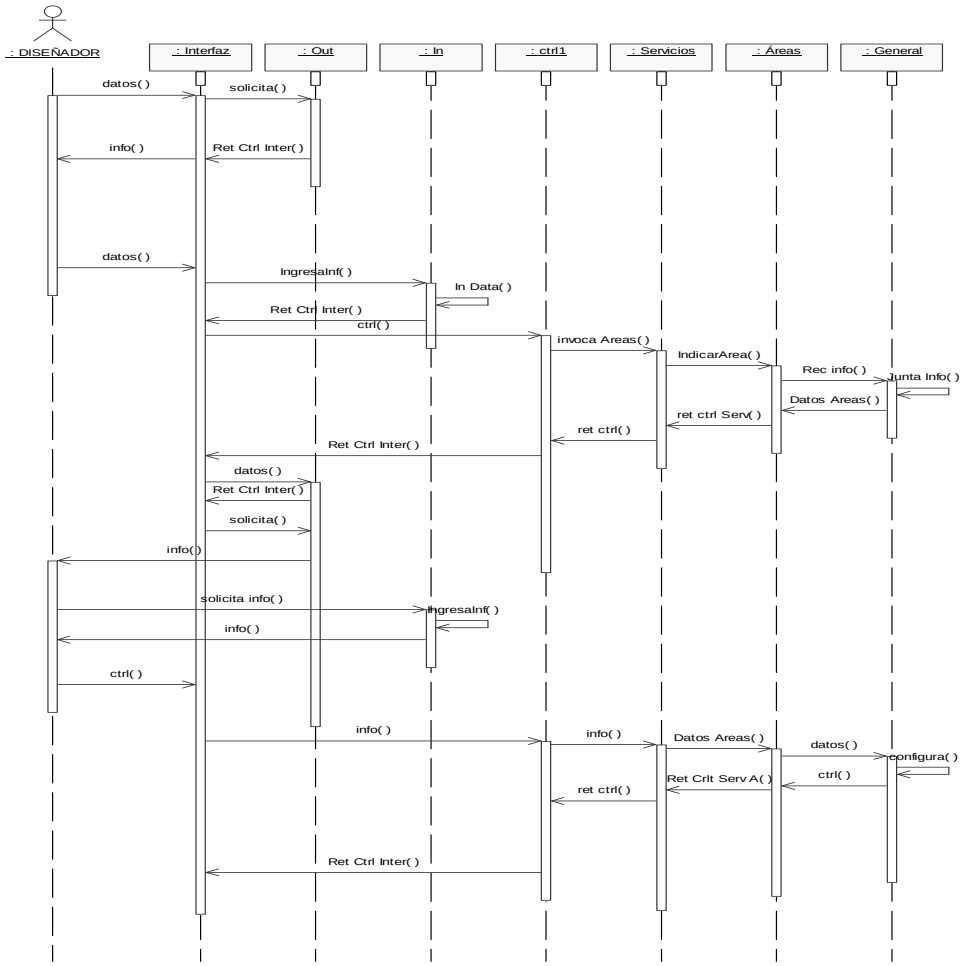

Figura 6.4: Diagrama de secuencia que realiza la Edición de los campos de la clase general

1. EL actor es quien inicia la secuencia para esta actividad a través de la clase interfaz.

2. La clase que tiene el control es la clase Interfaz y esta solicita al actor a través de la clase Out que seleccione la categoría de metadatos que desee complementar, al terminar la clase Out retorna a la clase Interfaz.

3. Interfaz da el control a la clase In en espera de que el diseñador indique la selección de la categoría a editar. Cuando lo hace, devuelve el control a la clase Interfaz con el argumento que indica la selección de la categoría.

4. Interfaz otorga el control a la clase ctrl1 con el argumento de la selección de la categoría, la clase ctrl1 al tener acceso al nivel lógico, invoca a la clase Servicios y le pasa el argumento de la categoría seleccionada.

5. La clase Servicios invoca la clase que satisfaga las necesidades del diseñador, invoca a la clase Áreas, la cual en base al argumento, seleccionará la clase del área correspondiente. Para el caso del diagrama (que modela la clase General) invoca a la clase General para recopilar la información correspondiente a esta área

6. La clase General (que ha sido invocada) reconoce los elementos que necesita y devuelve una solicitud como argumento a la clase Áreas, ésta a su vez devuelve los argumentos solicitados a la clase Servicios y ésta a la clase ctrl1 para que los haga llegar hasta la clase interfaz.

7. Interfaz, dependiendo del argumento, solicita al diseñador por medio de la clase Out, los datos al diseñador; el diseñador los introduce por medio de la clase in que posteriormente regresa a la clase Interfaz.

8. Interfaz envía el control a la clase ctrl1 con los datos como argumentos, esta clase a su vez invoca a la clase servicios con los datos como argumentos.

9. La clase Servicios seleccionará la clase más adecuada para satisfacer la solicitud del diseñador.

10. La clase Servicios invoca a la clase Áreas para que a través de ésta, se seleccione la clase correspondiente a la categoría deseada.

11. Para el caso del diagrama se está poniendo como ejemplo la selección de la categoría general, por lo tanto se invoca a la clase correspondiente (la clase general) con los argumentos de los datos que se necesitan y estos se registran.

12. Al terminar, la clase General devuelve el control a todas las clases subsecuentes, desde la clase Áreas hasta llegar a la clase Interfaz y con esto concluye el proceso de edición.

6.2.3. Diagrama de secuencia: Generación

En la figura 6.5, se presenta el diagrama correspondiente a la generación del archivo XML que contendrá los metadatos.

1. De la misma forma que en los diagramas anteriores se da inicio con el control del actor e interactuando con la clase Interfaz. Ésta da el control a la clase In y espera que el diseñador ingrese las instrucciones para generar el documento. El diseñador indica la generación del archivo XML y este se pasa como argumento a la clase Interfaz.

2. Interfaz da el control con el argumento a la clase ctrl1 y esta al tener contacto con el nivel lógico, invocará a la clase Servicios.

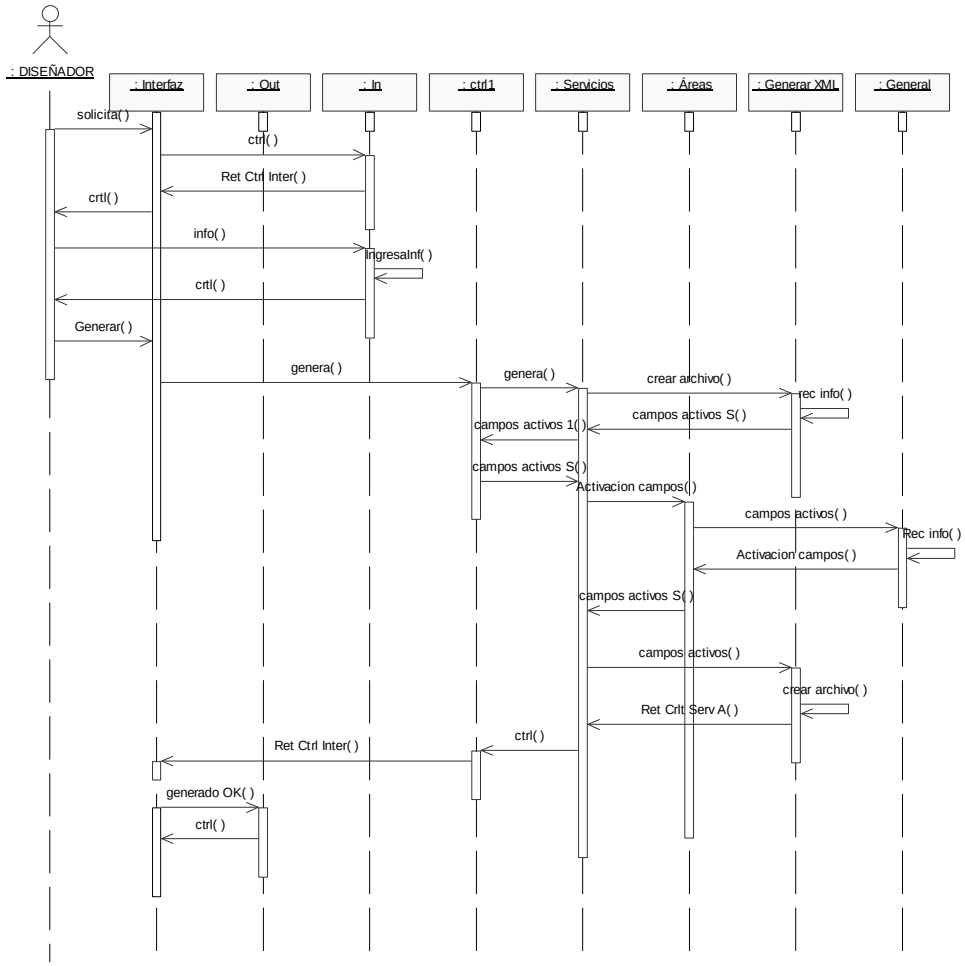

Figura 6.5: Diagrama de secuencia que realiza la Generación del archivo XML

3. Servicios ha sido invocada y esta invocará a la clase Generar XML pasandole el argumento al mismo tiempo.

4. Generar XML inicia la creación del archivo y manda una solicitud la clase Servicios indicando que necesita la información dada por el diseñador registrada en las clases correspondientes a las áreas.

5. Servicios pasa el argumento a la clase Áreas y esta a su vez a las clases de las categorías correspondientes (debido a que se plantea un ejemplo de la actividad sobre

la categoría General, en el diagrama solo se muestra la actividad correspondiente, para las demás categorías la actividad se lleva a cabo de la misma forma).

6. Las clases de las categorías obtienen los datos y los devuelven a la clase Áreas y esta a su vez a la clase Servicios para que Servicios los entregue a la clase Generar XML

7. Cuando Generar XML obtiene los datos, la aplicación entra en un estado llamado Generación, durante el cual se obtendrá el archivo físico de XML. Al obtenerlo indica a la aplicación que ha finalizado la generación y esta sale del estado Generación.

8. La clase Genera XML regresa a Servicios un argumento que indica que la generación ha terminado y esta a ctrl1 para que esta devuelva el control y el argumento a la clase Interfaz.

9. La clase Interfaz indica al diseñador por medio de la clase Out, que la generación del archivo a concluido. Posteriormente la clase Out regresa a la clase Interfaz.

Diagrama de secuencia: Verificación

La figura 6.6, muestra la forma en como el caso de uso de verificación es iniciado. Tomando como primera parte la solicitud del servicio y proporcionanado los datos que la aplicación necesita para ejecutarlo.

1. El actor es quien tiene el control e ingresa la silicitud de verificar metadatos a través de la clase In.

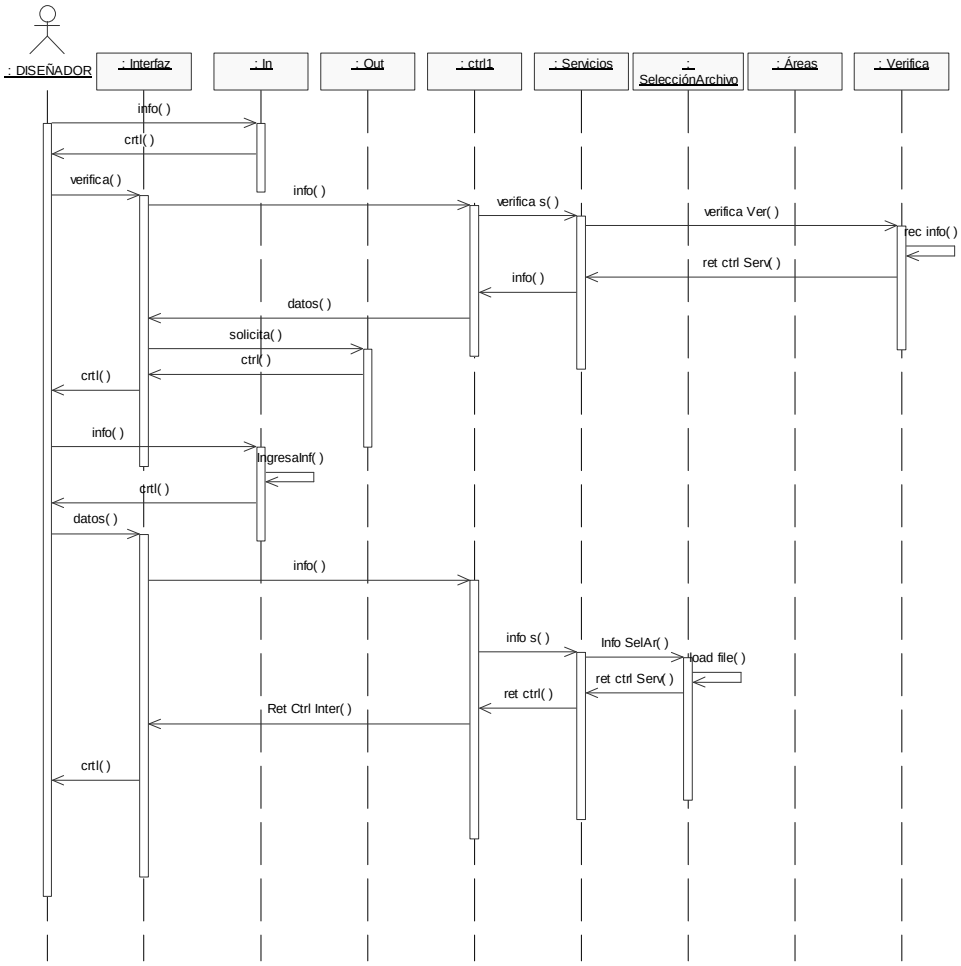

Figura 6.6: Diagrama de secuencia que realiza la iniciación del proceso de Verificación

2. La clase In se encarga de pasar esa solicitud a la clase interfaz que es la que llama a la clase ctrl1 con la información correspondiente y esta a su vez los envía a la clase Servicios.

3. Servicios envía el argumento a la clase Verifica, esta realiza una lista de la información que necesita para verificar al archivo y la envía a la clase Servicios, esta a su vez a la clase ctrl1 para que esta los solicite al diseñador a través de la clase interfaz.

4. Interfaz, por medio de la clase Out solicita al actor el nombre y la ubicación del archivo.

5. El actor tiene el control y por medio de la clase In, proporciona la información requerida.

6. Cuando el diseñador proporcione la información por medio de la clase In esta lo devuelve a la clase interfaz para poder ser entregado a la clase verifica.

7. Interfaz da el control a la clase ctrl1 y le pasa el argumento, esta clase a su vez invoca a la clase Servicios y le entrega el argumento que contiene la ubicación y el nombre del archivo.

8. La clase Servicios es quien invoca a la clase SelecciónArchivo que es quien se encarga de cargar el archivo en memoria.

9. Una vez hecho esto, se devuelve el control a Servicios y a ctrl1 que a su vez a Interfaz y finalmente al actor.

10. Interfaz Solicita al actor que seleccione el estándar bajo el cual se realizará la verificación. El diagrama de secuencia para este proceso se puede observar en la figura 6.3.

La figura 6.7, muestra la forma en como el proceso de verificación es ejecutado y finalizado. En el, se muestra como se proporcionan a la clase Verificación los datos necesarios para que pueda ejecutar el servicio y una vez ejecutado, la forma en que muestra el resultado al diseñador.

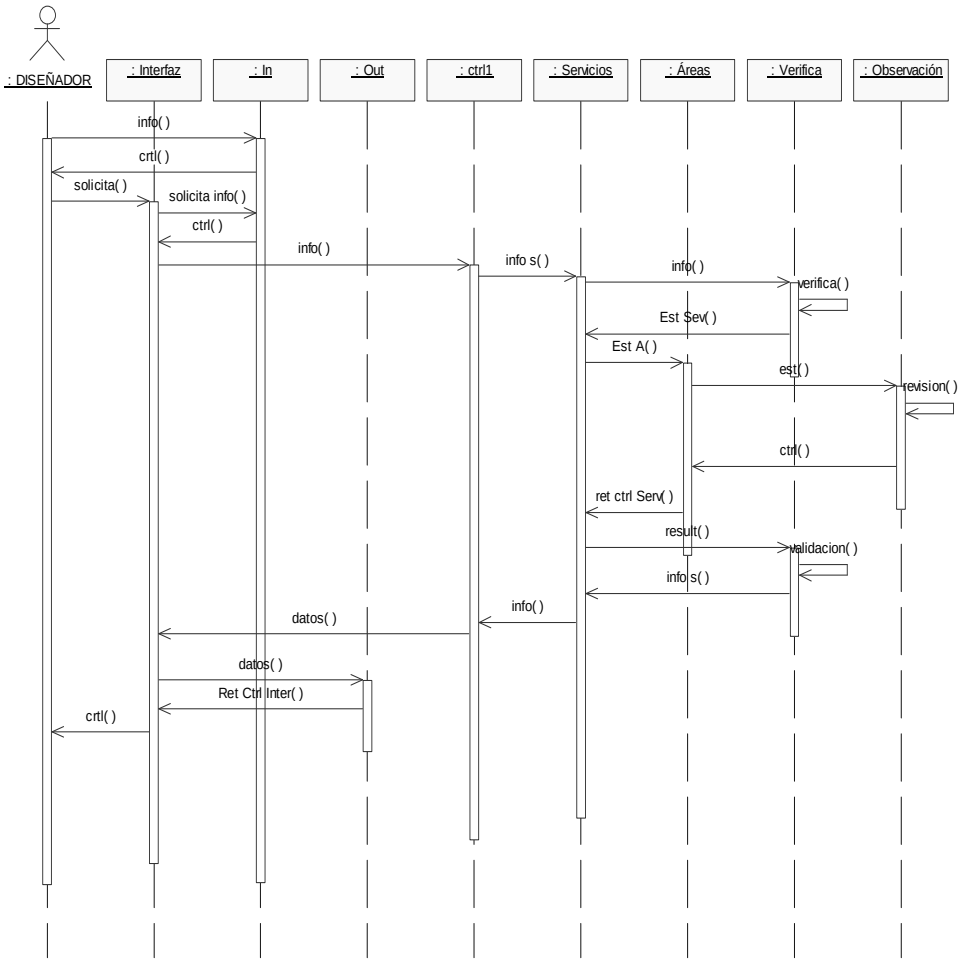

Figura 6.7: Diagrama de Secuencia que realiza la ejecución del proceso de verificación y su terminación

1. Cuando se ha seleccionado el estándar, el actor envía a la clase Interfaz través de la clase In los datos del archivo y el estándar. La clase Interfaz envia a la clase ctrl1, para que los haga llegar hasta la clase Verifica a través de servicios.

2. La clase Verifica recibe los datos y se configura dependiendo de la información recibida en los datos y envía el parámetro estándar a La clase Servicios para que este lo mande a la clase áreas.

3. La Clase Áreas recibe e invoca a cada una de las clases de cada categoría con el argumento estándar. Estas realizarán una verificación de la categoría correspondiente y la devolverán a la clase Áreas, la cual realizará una colección de resultados.

4. Cuando Áreas tenga todos los resultados, los envía a la clase Servicios y este a su vez a la clase Verifica. La clase Verifica toma su decisión y la envía a la clase ctrl1 a través de la clase Servicios para que ctrl1 la lleve a la clase Interfaz y esta por medio de la clase Out la despliegue la información al diseñador.

Capítulo 7

Implementación

Para llevar a cabo la implementación de la aplicación, se ha seleccionado la tecnología .NET y sobre ella Visual C++. Microsoft .NET es un concepto que define un modo de escribir software y un conjunto de herramientas para el desarrollo del software que pone en práctica dicho concepto.

El concepto .NET surge para simplificar la escritura de sistemas, para que se puedan conectar e interoperar entre ellos de forma segura en todo momento y desde cualquier lugar y dispositivo. Asimismo los XML *Web Services* constituyen la tecnología principal que permite habilitar dicha capacidad. Esta tecnología es al mismo tiempo una metodología y una capa de transporte para el paso de información entre componentes de equipos, redes y sistemas operativos diferentes. Actualmente ya existe un número considerable de empresas que se conectan con sus socios a través de XML *Web Services*. Asimismo, Microsoft continúa agregando compatibilidad con los XML *Web Services* a toda su línea de productos. Desde el punto de vista de un desarrollador, .NET facilita la escritura de sistemas capaces de conectar entre sí utilizando Microsoft Visual Studio .NET, .NET Framework y los XML *Web Services*.

7.1. .NET Framework

.NET es mucho más que los XML *Web Services*. En el corazón de .NET se encuentra .NET *Framework*, basado en *Common Language Runtime* y las bibliotecas de clases. Estos dos componentes proporcionan el motor de ejecución y las API de programación necesarias para la creación de aplicaciones .NET.

.NET *Framework* es un conjunto de clases utilizadas para generar aplicaciones que se ejecutan en *Common Language Runtime*. Estas bibliotecas de clases ofrecen una amplia compatibilidad con una variada gama de tareas, entre las que se incluyen el acceso a datos, seguridad, E/S de archivos, manipulación XML, mensajería, reflexión de clase, XML *Web Services*, ASP.NET y servicios Microsoft Windows. Para facilitar la creación de lenguajes para .NET *Framework*, Microsoft ha desarrollado la especificación en lenguaje común (CLS). CLS describe las características que debe ofrecer cada lenguaje para poder utilizar .NET *Framework* y *Common Language Runtime*, así como para interoperar con componentes escritos en lenguajes diferentes. Cuando un lenguaje implementa la funcionalidad necesaria, se dice que es compatible con .NET. Los lenguajes compatibles con .NET admiten los mismos tipos de datos, utilizan las mismas clases de .NET *Framework* y utilizan un *Common Language Runtime* idéntico para administrar la ejecución. Por lo tanto, cada lenguaje compatible con .NET se podría considerar como un ciudadano .NET de primera clase. La figura 7.1 muestra cómo se sitúan los distintos componentes de .NET sobre el sistema operativo. .NET *Framework* está formado por los cuadros que representan a ASP.NET, Windows Forms, ADO.NET y XML y la biblioteca de clases base.

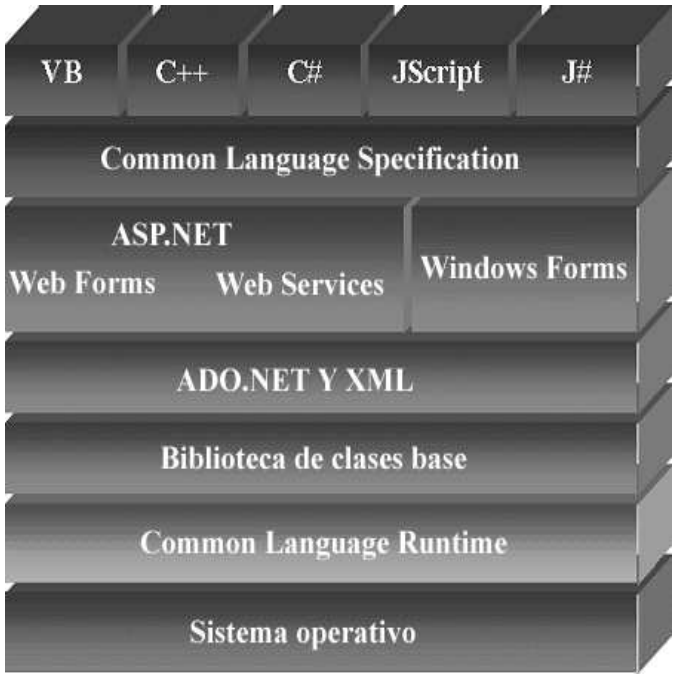

Figura 7.1: Arquitectura de .NET *Framework*.

7.2. Visual Studio .NET

Para habilitar el desarrollo de .NET, Microsoft volvió a escribir sus herramientas de desarrollo y las empaquetó en un producto nuevo conocido como Visual Studio .NET. Visual Studio .NET permite desarrollar aplicaciones en uno o varios lenguajes. Se utiliza el mismo entorno de desarrollo integrado (IDE) para el diseño, desarrollo, depuración e implementación de componentes .NET. También permite la creación de aplicaciones de consola, aplicaciones para Windows, archivos DLL, aplicaciones Web, XML Web Services y aplicaciones para dispositivos de bolsillo. Ofrece algunas características exclusivas de alta productividad, tales como: IntelliSense; diseñadores visuales de Web Forms y Windows Forms, esquemas XML y datos; un depurador de varios lenguajes que alterna sin problemas entre códigos escritos en lenguajes diferentes; una estrecha integración con .NET *Framework*; Ayuda dinámica, que proporciona una ayuda contex-

tual continua mientras se escribe; Lista de tareas, que muestra los errores del compilador y las tareas pendientes; características de diseño de arquitecturas como la integración con Visio y un explorador de servidores para obtener acceso visual a bases de datos, servicios de Windows, contadores de rendimiento y componentes de aplicaciones del lado del servidor.

7.3. Comparación de .NET con Java

Antes de dar inicio a la programación, se tenía contemplado utilizar el lenguaje JAVA, sin embargo, realizamos una comparación con la tecnología .NET y los resultados concluyeron que .NET se acoplaba mejor para nuestros objetivos. Aunque para los desarrolladores de Java expertos, .NET puede resultar bastante parecido a la plataforma Java ya que los dos permiten crear lenguajes de forma estructurada, disponen de lenguajes que compilan a código intermedio y ofrecen una amplia biblioteca con varias API para el desarrollo de aplicaciones. No obstante, .NET encierra una serie de objetivos diferentes a los de la plataforma Java. Desde el punto de vista conceptual, Java representa dos cosas: la plataforma Java (el tiempo de ejecución y las API) y el lenguaje Java. El propósito de esta plataforma es ofrecer compatibilidad con aplicaciones escritas en lenguaje Java y compiladas a código de bytes de Java. Aunque se han intentado compilar otros lenguajes a código de bytes de Java, los esfuerzos han quedado reducidos a meros ejercicios académicos. El ideal de Java ha sido siempre lograr un lenguaje único en varias plataformas. Del mismo modo, .NET representa dos cosas diferentes: .NET *Framework* (el tiempo de ejecución y las API) y la plétora de lenguajes de programación compatibles. El propósito de .NET *Framework* es ofrecer compatibilidad con aplicaciones escritas en cualquier lenguaje. El objetivo de .NET es lograr una única plataforma compartida por varios lenguajes.

7.4. Ventajas de .NET con respecto a Java

Además de la libertad a la hora de elegir un lenguaje, .NET ofrece una serie de ventajas con respecto a Java. A continuación se describen algunas de las ventajas más significativas:

- **Escribir una vez y depurar en todas partes**.- Aunque el ideal de Java resulta atractivo, por su lema: "escribir una vez y ejecutar en cualquier parte", la mayoría de las aplicaciones actuales se aplican a un único sistema operativo.

- **Seguridad**.-.NET ofrece más seguridad. Además de las API utilizadas para la criptografía, las cookies de seguridad y la autenticación, existe un mecanismo enriquecido y extensible para la asignación de distintos niveles de permisos a conjuntos de usuarios diferentes en función del origen del código. Todo este proceso se administra a través de una nueva tecnología, llamada "seguridad de acceso al código", que funciona con código cargado desde Internet o el equipo local.

- **Rendimiento**.- La característica de .NET que realmente destaca sobre las de Java es la capacidad para definir el rendimiento y el comportamiento de las aplicaciones. Las unidades principales de creación de aplicaciones se han diseñado para ofrecer escalabilidad y un rendimiento elevado. Además, la arquitectura de la aplicación .NET se ha diseñado para que ésta sea más rápida, *"El rendimiento de la aplicación .NET de Microsoft demostró claramente ser superior al de la aplicación Java: en un escenario no almacenado en caché lograba responder a 5.000 usuarios virtuales a una velocidad 10 veces mayor y en un escenario almacenado en caché el rendimiento era el doble de rápido"* [27].

7.5. Generación de Metadatos

La primera fase que se llevará a cabo en la implementación es la fase que genera los metadatos para los OA, en ella, se elabora una plantilla general la cual, representa a los metadatos definidos por el estándar LOM, esta plantilla sufrirá modificaciones cuando el usuario realice la selección de un estándar con una especificación diferente a la de LOM. Las opciones se encuentran en una lista que contiene lo siguiente:

- LOM
- SCORM 2.1
- SCORM 2004
- IMS
- DUBLIN CORE

Cada uno de los estándares situados en la lista anterior, contienen especificaciones diferentes para la creación de metadatos, inclusive, existen diferencias entre el SCORM 1.2 y el SCORM 2004. LOM define más de 50 elementos de metadatos e indica que todos son opcionales, cada una de las especificaciones marcan una cantidad de elementos de metadatos como obligatorios y el resto como opcionales, por lo que al seleccionar una especificación sus campos obligatorios deben de ser llenados satisfactoriamente. Con la finalidad limitar la información de los metadatos y para hacerlo más fácil al usuario, los elementos que sean marcados como opcionales quedarán bloqueados para evitar su edición, pero no se ocultan de la plantilla general, con la finalidad de que el usuario pueda observar que campos no permite cierta especificación y en caso de que el requiera de alguno de ellos seleccione otra especificación o en su defecto seleccione el estándar

LOM, que permite complementar todos los elementos de metadatos. En la figura 7.2, se muestra la interfaz de usuario para la generación de los metadatos.

Cada pantalla comprende una categoría de metadatos, para lo cual se definen los siguientes menús: General, Ciclo de vida, Metainformación, Técnica, Educativa, Derechos, Relación, Observación y Clasificación. Para facilitar el entendimiento al usuario inexperto en la tecnología de OA y en la creación de metadatos, por cada selección que realice, en el panel de ayuda situado del lado derecho de la plantilla general, se ofrece una descripción general del elemento o estándar seleccionado, contando en los casos más complejos con pequeños ejemplos de elaboración para su correcto entendimiento.

Figura 7.2: Interfaz del proceso de generación de metadatos

El boton XML es el que permite generar el archivo XML que contienen la estructura de los metadatos para algún OA. El botón salir termina la aplicación.

En el diagrama de actividades presentado en la figura 7.3 se muestra la forma en la cual se llevan a cabo las funcionalidades del generador.

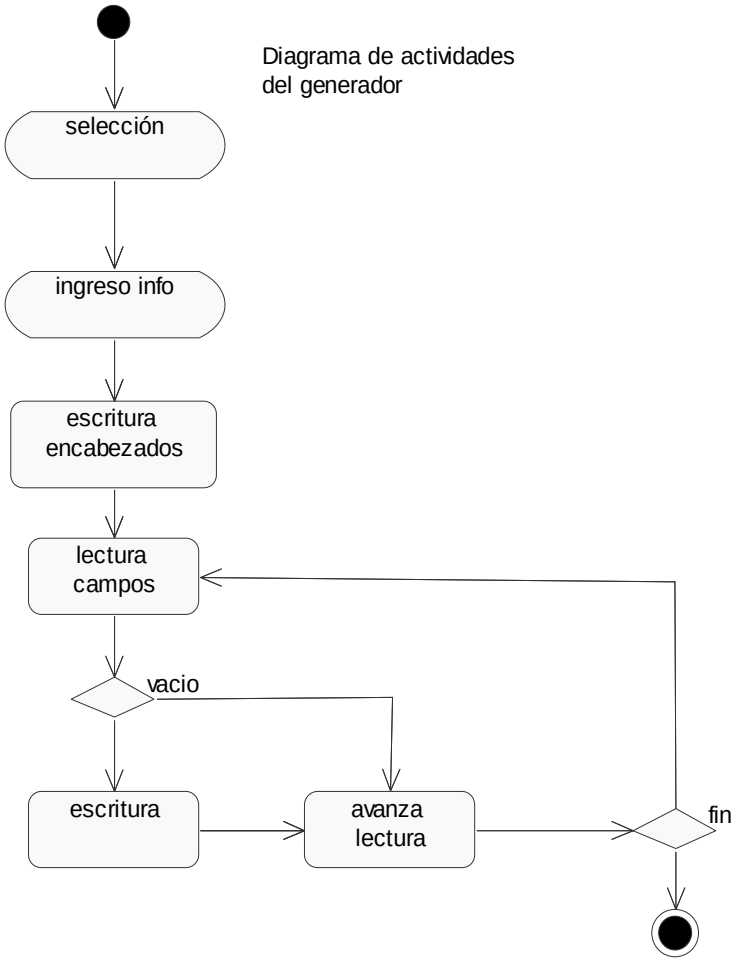

Figura 7.3: Diagrama de actividades general para el generador

La actividad selección del estándar, consta de un listbox el cual contiene cinco opciones correspondientes a los cinco estándares comprendidos por el generador, los cuales son LOM, SCORM 1.2, SCORM 2004, IMS, DUBLIN CORE, al seleccionar alguno de ellos el programa entrara a un selector de casos, el cual se encarga de activar o desactivar campos de metadatos, dependiendo la categoría y el estándar actual. El código

que se muestra en la figura 7.4, es un fragmento del que se utilizó en el programa en la selección de campos de metadatos.

```
int x=0;
if (sender == combo){
        x=(combo->SelectedIndex);
        switch (x){
                case 0:
                        AyudaText->Text=("Estándar LOM de la IEEE-LTSC");
                                        TexTitulo->Enabled=1;
                                        TexCatag->Enabled=1;
                                        CVTexVer->Enabled=1;
                                        CVTexFecha->Enabled=1;
                break;
                case 1:
                        AyudaText->Text=("Estándar SCORM 1.2 ");
                                        TexTitulo->Enabled=1;
                                        TexEstruct->Enabled=0;
                                        TecTexFormato->Enabled=1;
                                        TecTexTama->Enabled=0;
                break;
                case 2:
                        AyudaText->Text=("Estándar SCORM 1.3 ");
                                        catalogoentrada->Enabled=1;
                                        TexCover->Enabled=0;
                                        EdTexTipoInter->Enabled=0;
                                        EdTexEdad->Enabled=0;
                                        DTexCosto->Enabled=1;
                                        DTexDescription->Enabled=0;
                break;
                case 3:
                        AyudaText->Text=("Estándar IMS ");
                                        TexTitulo->Enabled=0;
                                        catalogoentrada->Enabled=1;
                                        ClasTexFin->Enabled=0;
                                        groupBox10->Enabled=0;
                break;
                case 4:
                        AyudaText->Text=("Estándar Dublin Core ");
                                        TexTitulo->Enabled=1;
                                        catalogoentrada->Enabled=0;
                                        ObTexPerson->Enabled=0;
                                        ObTexFecha->Enabled=0;
                break;
                }}
```

Figura 7.4: Fragmento de código de selección de campos de metadatos

Como se puede observar cada uno de los casos corresponde a una especificación diferente, en donde los campos son activados con TexTitulo-Enabled=1; y desactivado con TexTitulo-Enabled=o; Cada una de las plantillas creadas para la captura de metadatos, contiene definidos un campo de texto el cual albergará la información que el usuario necesite.

En esta actividad también se considera la selección de área, que es la que permite al usuario elegir el área donde desea ingresar información de los metadatos. No hay que olvidar que los campos para ingresar la información son activados y desactivados dependiendo de la selección del estándar previamente realizada, por defecto, el estándar que aparece seleccionado el LOM, pues es el que no condiciona los campos de metadatos. Las áreas son definidas por medio de tabs, los cuales contienen la información de cada una de ellas. Los tabs que almacenan a los campos de texto son creados por medio de una línea de código que es como se muestra a continuación:

```
private: System::Windows::Forms::TabPage * TabGeneral;
private: System::Windows::Forms::TabPage * TabDerechos;
```

Las cajas de texto se definen de esta forma:

```
private: System::Windows::Forms::TextBox * TexCatag;
private: System::Windows::Forms::TextBox * TexEntrada;
```

Algunos campos contienen una lista que despliega la información que puede ser ingresada por el usuario. Estas listas vienen en forma de ComboBox y se definen como sigue:

```
private: System::Windows::Forms::ComboBox * TexAgregacion;
```

La caja de texto que muestra la información de ayuda para el usuario se define de la siguiente manera:

private: System::Windows::Forms::RichTextBox * AyudaText;

Cada elemento definido para las plantillas debe de ser inicializado para que el compilador lo reconozca. La inicialización de componentes se lleva a cabo de la siguiente manera:

this-CVTexFecha = new System::Windows::Forms::TextBox();
this-CVTexFecha-BackColor = System::Drawing::SystemColors::Window;
this-CVTexFecha-Location = System::Drawing::Point(24, 136);
this-CVTexFecha-Name = S ÇVTexFecha"; this-CVTexFecha-Size =
System::Drawing::Size(280, 20); this-CVTexFecha-TabIndex = 5;
this-CVTexFecha-Text = S; this-CVTexFecha-Enter += new System : Even Handler(this, CVTexFechaEnter);

Es en estos elementos donde la información será requerida para que posteriormente sea ingresada al archivo XML.

En Microsoft .NET Framework existen dos conjuntos diferentes de clase que proporcionan operaciones de lectura y escritura orientadas a XML. Dichas clases se denominan de forma global: lectores y escritores de XML. La clase para los lectores es XmlReader mientras que XmlWriter proporciona una interfaz de programación básica para los escritores. Un escritor de XML representa un componente que proporciona un método rápido, siempre hacia delante, de generar datos XML a flujos o archivos. Lo que es más importante, un escritor de XML garantiza por diseño que todos los datos que genera cumplen con las recomendaciones de XML 1.0 y de los espacios de nombres de W3C.

Los escritores XML se basan en una clase abstracta XmlWriter que define la interfaz de .NET Framework para la escritura de XML [8].

LA actividad para escribir encabezados, se encarga de asentar la informacón en el archivo XML, con las características del archivo XML en fabricación. Con estos encabezados, el LMS Moodle, podrá identificar y localizar al archivo físico de OA, para poder utilizarlo. Esta actividad cuenta con un identificador y un apuntador al OA físico. El código mostrado en la figura 7.5, denota la manera en que se crean los encabezados de los archivos de metadatos. En él, se inicia el elemento raíz llamado *manifest*, posteriormente se indica de que esquema del cual se obtendrá la gramática para generar al archivo XML. En la línea siguiente, se le asigna un identificador al OA y se especifica el tipo y nombre del OA físico.

```
writer->WriteStartElement("manifest");
writer->WriteAttributeString("xmlns", http://www.imsproject.org/xsd/imscp_rootv1p1p2");
writer->WriteStartElement("resources");
writer->WriteAttributeString("identifier", " RES-9FAE4F92-386A-B16-141-CBA23127");
writer->WriteAttributeString("type", "contenidoweb");
writer->WriteAttributeString("adlcpscormtype", "sco");
writer->WriteAttributeString("href", "animacion.swf");
```

Figura 7.5: Fragmento de código para generación de encabezados para metadatos

La actividad de lectura de campos, revisa campo por campo de la plantilla seleccionada con la finalidad de ingresarlos en el archivo de XML. Si el campo está vacío, simplemente avanza al siguiente, si no, lo incrusta en el archivo en construcción. Posteriormente, verifica que ya no existan campos para leer, si los campos se han terminado, entonces finaliza, de lo contrario regresa al proceso de lectura de campos. La actividad escritura tiene la función de escribir el campo ingresado por el usuario en formato

XML. Finalmente la actividad avanza lectura revisa el siguiente campo de la aplicación.

El proceso generador, efectúa su actividad utilizando la clase XmlWriter, ésta, elabora un fragmento de código, el cual, escribe etiquetas de XML dependiendo del estándar seleccionado y también, de los elementos de metadatos complementados, en un archivo físico dependiendo los esquemas de los estándares.

En la etapa del generador se efectúa la creación de un archivo XML con una sintaxis y una gramática correcta, es decir, que cumpla con los esquemas establecidos por los estándares.

Si partimos de que una gramática es una especificación formal de un lenguaje de programación que permite definir la sintaxis o estructura de los programas. La sintaxis correcta definida por el XML 1.0 de W3C para los archivos creados que comprenden los metadatos, la proporciona la clase XmlWriter, y en base a los esquemas definidos por los estándares se puede crear la estructura correcta.

Una gramática es una especificación precisa y fácil de entender para los programas en un lenguaje de programación, además, impone una estructura al programa que es útil para su traducción en código objeto [29]. La clase XmlWriter garantiza que las etiquetas se encuentren bien escritas; pero, para que gramáticalmente estén correctas deben de seguir las indicaciones establecidas por los esquemas de los estándares, estos esquemas son los que van a validar si la gramática con la que están escritas las líneas de código del archivo XML es correcta o no. El lenguaje XML reconoce dos tipos de programas basados en etiquetas, los llamados *archivos bien formados* y los *archivos validos*. Como se ha comentado anteriormente, la clase XmlWriter garantiza que las

etiquetas se encuentren bien escritas, a los archivos XML con esas características son los conocidos como *archivos bien formados*. Por ejemplo las siguientes líneas de código generan una parte de un *archivo bien formado*:

```
// Iniciar el elemento
writer-WriteStartElement("Elemento ");
//ingresar valor al elemento
writer-WriteString(TexLang-Text);
// Cerrar el elemento
writer-WriteEndElement();
```

Eso nos devuelve una etiqueta que esta escrita de forma adecuada pero aun no se puede considerar gramáticalmente correcta:

<elemento> Valor de información </elemento>

Para los Metadatos estandarizados, no basta con ser considerados como *archivos bien formados*, estos archivos necesitan ser validados por algún esquema y el que deben de seguir es el establecido por los estándares internacionales. Los esquemas definen los nombres que deberán de adoptar cada una de sus etiquetas de los archivos, (el texto que incluyan las etiquetas para describirse viene dado por el usuario), en otras palabras, definen la gramática con la cual deben de ser escritos. Si las etiquetas del archivo XML son construidas como lo indica cierto esquema, este archivo XML se considera válido por el mismo. El ejemplo de un esquema definido por los estándares se sitúa en el apéndice A. El siguiente fragmento de código genera una etiqueta valida por los esquemas de los estándares internacionales:

```
// Iniciar el elemento language
writer-WriteStartElement("language");
writer-WriteString(TexLang-Text);
// Cerrar el elemento language
writer-WriteEndElement();
```

La etiqueta resultante de ese fragmento de código es:

<language> Valor definido por el usuario </language>

La definición de cada una de las etiquetas se realiza con un fragmento de código similar al comentado anteriormente. Los elementos que no deban contener información por el usuario quedarán de la siguiente forma:

<language> </language>

Después de escrito el archivo XML se cierra y genera con estas instrucciones:

```
// Vaciar y cerrar writer-Flush();
writer-Close();
```

La creación de las etiquetas está dada en forma de árbol como el diagrama que se muestra en la figura 7.6:

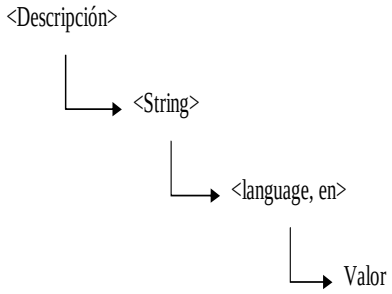

Figura 7.6: Diagrama de árbol de un conjunto de etiquetas en XML

7.6. Verificación de un archivo XML

La fase correspondiente a la verificación de metadatos, utiliza un lector de XML, que proporciona Visual C++ .NET, aunque la interfaz es más sencilla, no representa menor procesamiento de datos que el generador. Si tomamos en cuenta que un analizador sintáctico reconoce que una oración está bien escrita comparando si cumple con las regrlas gramáticales [29], entonces, el verificador, se puede considerar un analizador sintáctico, la razón es que este verificador leerá las líneas de código (oraciones) del archivo XML y este determinará a las líneas bien escritas dependiendo de la gramática utilizada para ese archivo XML; la gramática está dada por el esquema del estándar seleccionado.

La pantalla correspondiente al proceso verificador que se denota en la figura 7.7 muestra una lista para poder seleccionar el estándar bajo el cual se desea comprobar los metadatos y 2 botones, el primero permite ejecutar el proceso de verificación y el segundo cierra la aplicación. La verificación se lleva a cabo con el estándar seleccionado y si los metadatos cumplen con las expectativas esperadas por el estándar, se envía un mensaje que así lo indique, de lo contrario arroja en sus resultados una lista de correcciones, de los elementos requeridos. En el código escrito para realizar la función de verificar, se declaran variables de cadena que contienen un valor igual al de las etiquetas

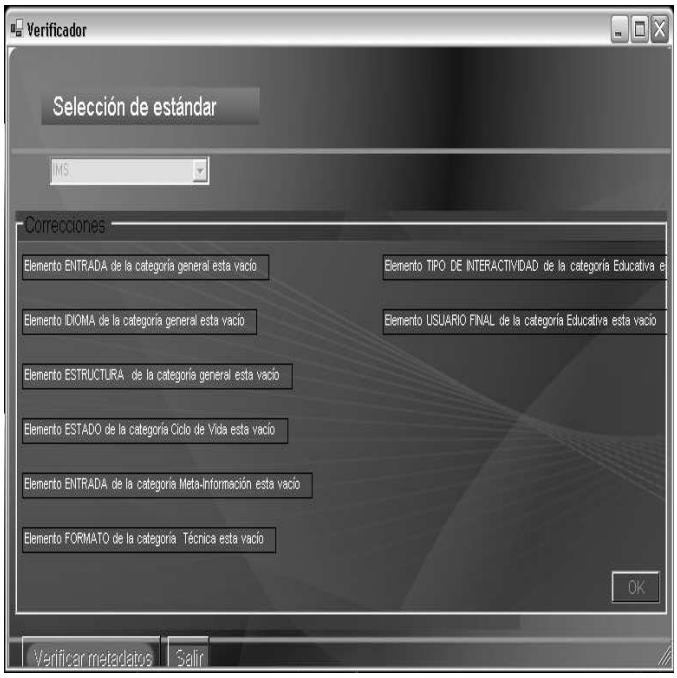

Figura 7.7: Interfaz del proceso de verificación de metadatos

establecidas por el estándar en el archivo XML.

El diagrama de actividades mostrado en la figura 7.8 muestra el funcionamiento general para llevar a cabo la fase del verificador.

En inicio se realiza la selección del estándar bajo el cual se verificarán los metadatos del archivo XML. Esto se efectúa por medio de un listBox, el cual contiene en sus valores a los siguientes estándares:

- LOM
- SCORM 1.2
- SCORM 2004

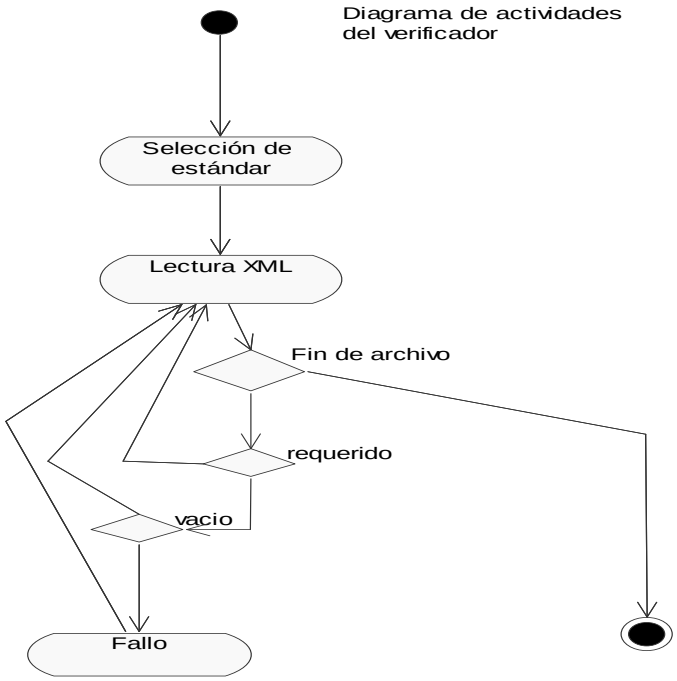

Figura 7.8: Diagrama de Actividades general para el verificador

- IMS
- DUBLIN CORE

La lista de los estándares elegibles tiene que ser definida para que pueda ser reconocida por el programa, esto se hace de esta forma:

private: System::Windows::Forms::ComboBox * VerEst;

y también tiene que inicializarse

System::Object* mcTemp1[] = new System::Object*[5];
mcTemp1[0] = S"LOM 1.0"; mcTemp1[1] = S"SCORM 2004";

```
mcTemp1[2] = S"SCORM 1.2"; mcTemp1[3] = S"IMS";
mcTemp1[4] = S"DUBLIN CORE";
this-VerEst-Items-AddRange(mcTemp1);
this-VerEst-Location = System::Drawing::Point(160, 128);
this-VerEst-Name = S"VerEst";
this-VerEst-Size = System::Drawing::Size(184, 23);
this-VerEst-TabIndex = 9;
this-VerEst-Text = S"LOM 1.0";
```

Posteriormente, realiza la lectura del archivo XML. Esto lo hace una línea a la vez para determinar que esta bien escrito; esa línea en lectura, será dividida en los diferentes elementos que la forman y se identificará elemento por elemento. Después de heber leido el elemento, confirma si no se ha llegado al final del documento, si es asi, entonces, terminará el proceso, si no, procederá a la identificación de la etiqueta leida, es decir, identificará si se trata de un inicio o fin de elemento, de un comentario o de algún elemento de texto. Este proceso es muy interesante pues dependiendo del estándar seleccionado, el verificador, contendrá una lista de que elementos que son los que no pueden quedar vacíos. El lector del archivo estará avanzando a través del documento y cuando encuentre un elemento que exija el estándar seleccionado, con su información correspondiente, lo marcara como leido en su lista. Si el elemento se encuentra, pero no tiene información, lo dejará sin marcar en su lista y al final de la lectura, se comprobará con las listas de que todos los elementos son incluidos con su respectiva información, de lo contrario, la lista reportará que elemento falta y a que área pertenece.

Para ejemplificar lo anterior, la figura 7.9, contiene un fragmento de código en el cual, el primer *while* es el que mantendrá al lector del archivo en ejecucución hasta que

encuentre el fin de archivo. La línea de código que contiene al *switch*, permite evaluar a un elemento de una línea que se este leyendo en ese momento y dependiendo de que tipo de nodo se trate, lo canalizará a un caso específico. En el código solo se muestran nodos de documento, declaración y elemento del un archivo XML.

```
//leer los nodos del xml
while (rdr->Read()){
//hacer algo con los datos
switch (rdr->NodeType){
        case XmlNodeType::XmlDeclaration:
                label4->Text="Declaración de XML";
        break;
        case XmlNodeType::Document:
                label4->Text="Nodo de documento";
        break;
        case XmlNodeType::Element:          //cuando esta leyendo un elemento
                                            //aqui lee los elementos del archivo XML
                                            //elemento=rdr->Name;
                elemento=rdr->Name;

                y=(VerEst->SelectedIndex); //selección del estándar
                switch (y){
                        case 0: label4->Text="Seleccion de lom";
                        break;

                        case 1: label4->Text="Seleccion de scorm 2004";
                                if (area==1){
                                        if (myCompIntl->Compare(entrada, elemento)==0){
                                                MessageBox::Show(S"elemento entrada"");
                                                lectura=1;// elemento leído
                                        }
                                        If (myCompIntl->Compare(descripion,
                                                elemento)==0){
                                                MessageBox::Show(S"elemento
                                                descripion",S"Mensaje...");
                                                 lectura=1;// elemento leído
                                        }
                                if (area ==3){
                                        if (myCompIntl->Compare(catalogo, elemento)==0){
                                                 MessageBox::Show(S"elemento
                                                catalogo",S"Mensaje...");
                                                lectura=1;// elemento leído
                                        }
                                if (rdr->AttributeCount > 0)
                                {   Console::Write("  ");
                                        while (rdr->MoveToNextAttribute())
                                        Console::Write(" {0}={1}", rdr->Name, rdr->Value);
                                        Console::WriteLine();
                                }
                        break;
```

Figura 7.9: Fragmento de código del verificador

Un lector de XML opera sobre un flujo de datos de sólo lectura, pasando de un nodo al siguiente con una única dirección, siempre hacia delante. El lector XML mantiene un apuntador interno que señala al nodo actual, así como a sus atributos y texto.

XmlReader es una clase abstracta que pertenece al espacio de nombres System.Xml;

define el conjunto de funciones que da a conocer un lector de XML y que permite acceso a un flujo de XML que es de sólo lectura, siempre hacia adelante. Esta clase proporciona un método para analizar datos XML que minimizan el uso de los recursos. También utiliza un modelo de extracción, lo que significa que podrá llamar a una función para obtener el siguiente nodo cuando este preparado [19].

Capítulo 8

Pruebas y Resultados

El objetivo de la fase de pruebas de un programa es el de detectar todo posible mal funcionamiento antes de liberar al producto al cliente. Lamentablemente no es posible obtener una plantilla de pruebas perfecta, en consecuencia, necesitamos un criterio para elegir los casos de prueba [21].

Las pruebas aplicadas al sistema se dividen generalmente en dos:

- **Pruebas en tiempo de desarrollo**.- también conocidas como pruebas informales o fase de prueba informal, son las que hace el desarrollador mientras está creando el sistema, tienen como objetivo comprobar que el programa compile y ver que todo está yendo como debiera, normalmente se realizan varios cientos de estas pruebas que básicamente consisten en compilar periódicamente durante el desarrollo y ejecutar para ver el resultado.

- **Pruebas después del desarrollo**.-Cuando se considera que un módulo está terminado se realizan estas pruebas, el objetivo de estas es buscar fallos a través de un criterio específico, estos criterios se denominan "pruebas de caja negra y de caja blanca". Los formatos de las pruebas se encuentran en el apéndice A.

8.1. Plan de Pruebas

Para obtener una buena fase de pruebas, debe de establecerse un plan de pruebas al cual apegarse. El plan de pruebas que se propone es el siguiente:

Plan de Pruebas: EP-Global (Examen de pruebas global)

1. Pruebas de caja negra locales
    a) Clases de Equivalencia
2. Pruebas de caja negra externas
3. Pruebas de caja blanca o pruebas de cobertura
4. Pruebas de rendimiento
5. Pruebas de robustez

Este plan de pruebas corresponde a las pruebas después del desarrollo.

8.2. Pruebas locales de caja negra

Las pruebas de caja negra son aquellas que se enfocan directamente en el exterior del módulo, sin importar el código, son pruebas funcionales en las que se trata de encontrar fallas en donde no se atiene a su especificación, como puede ser interfaz con el usuario, apariencia de los menús, control de las teclas, etc. Este tipo de pruebas no es aplicable a los módulos que trabajan en forma transparente al usuario.

Para realizar estas pruebas se hace referencia a las dos fases del programa:

- Fase de Generación
- Fase de Verificación

Para realizar estas pruebas existe una técnica algebraica llamada *clases de equivalencia*, consiste en tratar a todos las posibles entradas y parámetros como un modelo algebraico y utilizar las clases de este modelo para probar un amplio rango de posibilidades.

Clases de equivalencia.

Para la generación de estas clases no se puede armar un modelo, pero se pueden seguir las siguientes pautas como guía utilizable para la creación de cada clase. Por ejemplo: Cuando una entrada es booleana, existen solo dos clases, verdadero o falso. Para una entrada que está comprendida dentro de un rango, existen tres clases, por debajo, dentro y por encima del rango.

Fase de Generación.

La primera fase que se probó es la fase de Generación. En ella se efectúaron 12 pruebas para cubrir el cien por ciento de la funcionalidad de la aplicación. Las pruebas fueron las siguientes:

Prueba1: Creación de metadatos.

La primera prueba es para asegurar que el generador de metadatos de la aplicación, crea los metadatos conforme a los esquemas establecidos por los estándares y de esta

forma poder obtener un OA con los metadatos estandarizados.

Para ello, se introdujo la información solicitada por a aplicación comprobando que la aplicación permita al usuario ingresar la información en texto libre y que los campos con vocabulario controlado no permitan al usuario ingresar otro tipo de información. Se generaron los metadatos y se realizó una revisión de la estructura del archivo de metadatos para comprobar que esta construido de acuerdo a los estándares. La revisión arrojó como resultado un archivo en formato XML idéntico al marcado por los estándares.

El formato que se utiliza para docuentar las pruebas es el siguiete, teniendo un formulario para describir las actividades y objetivos de las pruebas y otro para describir los resultados obtenidos.

Formulario de pruebas:

PRUEBA No 1 Creación de metadatos

Descripción: La prueba consiste en introducir información a la aplicación para generar metadatos de un OA específico.

Objetivo de la prueba: Probar que la aplicación crea metadatos para un OA específico.

Alcance: Determinar si las etiquetas de los metadatos son generadas correctamente.

Actividades

1. Ingresar información a la aplicación

2. Generar metadatos

3. Comparar visualmente metadatos generados con metadatos estandarizados por LOM existentes.

Resultados esperados

- Para el ingreso de información:
    - El sistema deberá permitir en los campos de texto la introducción de información en formato libre por parte del usuario; la introducción de información en los campos de selección será restringida al vocabulario presentado por la aplicación.
- Para la generación de metadatos:
    - El sistema deberá generar un archivo de formato XML que contenga la información proporcionada por el usuario. Esta información deberá generarse sintáctica y semánticamente correcta. El archivo de formato XML generado deberá crearse en la ubicación definida por el usuario.
- Para la comparación de metadatos:
    - El archivo XML generado deberá se idéntico al archivo XML estándar.

Responsables

Por el departamento de Postgrado

Priego Meléndez Juan Leobardo.

Por la Universidad del Valle de Puebla

Profa. Elaine Allard

Formulario de resultados:

PRUEBA No 1 Creación de metadatos

Descripción: La prueba consiste en introducir información a la aplicación para generar metadatos de un OA específico.

Fecha: 3 de Julio 2006

Observacíones:

Ninguna

Resultado de la prueba

La revisión arrojó como resultado un archivo en formato XML idéntico al marcado por los estándares

Funcionarios que participaron en la prueba

Por parte de la Facultad de Idiomas de la Universidad del Valle de Puebla:

Profa. Elaine Allard

Por parte del Depto de Postgrado de la Benemérita Universidad Autónoma de Puebla:

Nombre Priego Meléndez Juan Leobardo

Las siguientes pruebas se efectuarón creando metadatos bajo los estándares SCORM 1.2, SCORM 2004, DUBLIN CORE. e IMS.

Para introducir la información a la aplicación primero se selecciona el estándar bajo el cual se quieren construir los metadatos, comprobando que la aplicación permita al usuario ingresar la información en texto libre y que los campos con vocabulario controlado no permitan al usuario ingresar otro tipo de información. Posteriormente

se generaron los metadatos y se efectuó una revisión de la estructura del archivo de metadatos. La revisión arrojó como resultado un archivo en formato XML idéntico al marcado por cualquiera de los estándares.

Prueba 6: Funcionamiento del OA en base a LOM sobre Moodle.

El objetivo de la sexta prueba es hacer que el material electrónico pueda ser utilizado en Moodle. Moodle requiere que los OA estén empaquetados en formato ZIP para poder interpretarlo, este paquete se debe de crear de acuerdo con las especificaciones del IMSMANIFEST, que también comprende el estándar SCORM, el paquete deberá de contener el archivo manifiesto, los esquemas de los estándares y los archivos físicos que corresponden al material electrónico.

Al tener el paquete construido, éste deberá ser cargado en Moodle como una actividad SCORM, el paquete se carga y se almacenará en el repositorio local del LMS. Cuando se trato de leer el OA empaquetado, fue leído sin ningún problema. Se construyeron más OA, pero realizando pruebas con los diferentes estándares para comprobar que podían ser leídos en Moodle. El resultado fue el mismo: Éxito

Para el siguiente bloque, se realizaron pruebas para el caso de que los metadatos no cumplieran con los estándares.

8.2.1. Prueba 11: Funcionamiento del OA con metadatos incorrectos sobre Moodle

El objetivo de esta prueba es determinar si el OA puede ser leído en Moodle cuando este no tenga información de metadatos. Al tener el paquete construido, se carga en Moodle como una actividad SCORM, y se almacenará en el repositorio local del LMS. Cuando se trato de leer el OA empaquetado, fue leído sin ningún problema.

Prueba 12: Funcionamiento del OA con encabezados XML incorrectos sobre Moodle.

El objetivo de esta prueba es determinar si el OA puede ser leído en Moodle cuando este contenga los encabezados del archivo XML incorrectos. Se generarán un archivo XML de metadatos el cual contenga la información de los encabezados diferente a la de los archivos estandarizados, posteriormente se crea el paquete IMSMANIFEST, al tener el paquete construido, se carga en Moodle como una actividad SCORM y se almacenará en el repositorio local del LMS, posteriormente se intentará leerlo desde Moodle.

El resultado que se obtuvo fue muy interesante, al descubrir que los metadatos no son imprescindibles para poder correr los OA en Moodle, pero, los encabezados del archivo XML, si lo son ya que estos son los que permiten al LMS localizarlo y poder relacionar el archivo XML con el archivo físico del OA. Por lo tanto en esta fase se concluye que para poder leer los OA en Moodle, no necesita tener los metadatos construidos de manera estandarizada, pero si, la definición de los encabezados para XML y el identificador que relaciona el archivo XML con el OA físicamente.

Fase de Verificación.

Una vez concluidas las pruebas de caja negra para la fase de Generación. Se procedió a efectúar las pruebas para la fase de verificación. En ella se efectúaron siete pruebas para cubrir el cien por ciento de la funcionalidad de la aplicación. Considerando que la aplicación contiene 2 entradas (una que es la que proporciona el usuario directamente y la otra son los metadatos), la siguiente lista muestra las clases de equivalencia que se utilizarán para las pruebas de caja negra:

- Etiquetas con cadena vacía.- Un ejemplo de este tipo de etiquetas en los metadatos esta representado por: donde se establece un campo vacío.

  ⟨⟩⟨|⟩

- Etiquetas con una cadena consistente de únicamente un espacio vacío.- Un ejemplo de este tipo de etiquetas en los metadatos esta representado por: en donde se

  ⟨⟩ ⟨|⟩

  establece un campo que contiene un espacio vacío.

- Etiquetas sintácticamente correcta.- Un ejemplo de este tipo de etiquetas en los metadatos esta representado por: en donde se establece un campo sintácticamente

  <entry> Entrada de datos </entry>

  correcto. Según el estándar LOM [11], el campo entry esta sintácticamente correcto.

- Etiquetas con cadena que empieza o termina en un espacio en blanco.- Un ejemplo de este tipo de etiquetas en los metadatos esta representado por: en donde se

  < entry>Entrada de datos< /entry>

  establece un campo que inicia con un espacio en blanco.

- Etiquetas sintácticamente correcta: valores semánticamente válidos e inválidos.- Un ejemplo para etiquetas sintácticamente correctas pero semánticamente incorrectas esta representado por la siguiente línea:

  En donde los elementos catalog y entry contienen su apertura y fin de elemento,

  <catalog>catalogo<entry></catalog>Entrada de datos</entry>

  considerándose así sintácticamente correcto pero la forma en que están anidados presenta una semántica incorrecta. La siguiente línea muestra un ejemplo de etiquetas sintácticamente y semánticamente correcta según el estándar LOM [11].

  <general><title> TITULO DEL OBJETO DE APRENDIZAJE</title></general>

Las pruebas aplicadas fueron las siguientes:

Prueba 13: Confirmar metadatos no estandarizados.

La siguiente prueba a realizar consiste en leer los metadatos que no son escritos en base a los esquemas de LOM y notificarlo. El objetivo de esta prueba es determinar si la aplicación detecta si los metadatos no son validos dependiendo de los esquemas definidos por el estándar LOM.

Prueba 14: Confirmar metadatos estandarizados.

La siguiente prueba a realizar consiste en leer los metadatos y corroborar que se encuentran escritos en base a los esquemas establecidos por los estándares. El objetivo de esta prueba es determinar la aplicación es capaz de detectar si los metadatos son validos o no dependiendo de los esquemas definidos por los estándares.

Las siguientes pruebas a realizar consisten en leer los metadatos y corroborar que cumplen con la información mínima requerida por algún estándar.

El objetivo de estas pruebas es determinar si la aplicación es capaz de identificar los campos de metadatos requeridos algún estándar y detectar si estos no están vacíos.

Se realizaron las pruebas correspondientes contemplando los demás estándares (de la 16 a la 18), obteniendo como resultado el mismo para todas las pruebas: El verificador identifico los metadatos y determino que tenían los mínimos solicitados por cada estándar.

Los primeros análisis se realizaron con archivos que no cumplen con los estándares. El verificador lee cada uno de los campos de metadatos escritos sobre el archivo manifiesto y reporta los elementos que están mal escritos. Para confirmar el resultado de esta prueba se pasan a revisión archivos que si cumplen con los esquemas; el verificador, no emite ningún aviso de que el archivo contenga anomalías, por lo cual, la tercera prueba se considera con éxito.

Para la fase de pruebas remotas, se obtuvieron OA externos para poder determinar

si son aptos para ser almacenados en el repositorio local. Se obtuvieron OA utilizados en la plataforma AVA de la Universidad Virtual de Guadalajara para realizar pruebas.

8.3. Pruebas externas de caja Negra

Prueba 23: Verificación de metadatos externos.

La siguiente prueba a realizar consiste en leer los metadatos de OA externos y determinar si se encuentran escritos de acuerdo a los esquemas de LOM. El verificador efectúará una lectura del archivo revisando las etiquetas correspondientes a los elementos y notificará al usuario si alguna no cumple con los esquemas.

El objetivo de esta prueba es determinar si el verificador de la aplicación es capaz de detectar que los OA se encuentran escritos bajo los esquemas de LOM. Se efefctúo esta prueba para los diferentes estándares comprendidos en la aplicación, al realizar las pruebas, el proceso de verificación, reporto que no cumplían con los esquemas de los estándares internacionales y al tratar de cargarlos en Moodle, tampoco pudieron ser leídos.

Prueba 28: Creación de metadatos para OA externos bajo.

La siguiente prueba a realizar consiste generar los metadatos de OA externos y conseguir que estos puedan ser leídos en Moodle.

El objetivo de esta prueba es determinar si el generador de la aplicación es capaz de

crear metadatos para algún OA externo y que este OA dentro de un archivo manifiesto pueda ser leído en Moodle.

Para generar los metadatos, se reutilizo el archivo previamente analizado por el verificador, el archivo manifiesto se volvió a crear utilizando el generador de la aplicación y posteriormente los OA pudieron ser cargados sin problemas por Moodle.

Las pruebas de verificación y generación de metadatos para contenido electrónico desarrollado en otras entidades, han sido terminadas satisfactoriamente.

Se llevarón a cabo las pruebas en base a todos los estándares comprendidos por la aplicación. Las pruebas siguientes consisten en ejecutar un buscador de OA sobre material estandarizado y sobre material no estandarizado. Para ello se fabricó un pequeño buscador de OA, el cual consiste en solicitar al usuario palabras clave del OA a buscar y posteriormente analizar los archivos colocados en el repositorio para determinar cuales son los que reúnen las características solicitadas por el usuario. El buscador solo es de prueba y se ejecutará solamente en el repositorio local.

Prueba 32: Aplicación de un buscador de metadatos sobre OA estandarizados.

Esta prueba consiste en buscar OA con metadatos estandarizados en un repositorio local con un buscador de OA piloto. El buscador solamente funciona manualmente por ser piloto.

El objetivo de esta prueba es determinar si el buscador de OA obtiene tiempos de

respuesta más cortos cuando navega en OA's estandarizados que cuando lo hace con OA no estandarizados.

El buscador de OA, tienen en su registro las especificaciones de los metadatos definidas por el estándar bajo el cual opera, por lo tanto primero realiza la búsqueda de cierta información para obtener los resultados apropiados. Al omitir en su búsqueda mas de 50 campos de metadatos, el buscador se vuelve mas certero en sus resultados y al proporcionarle un OA con la información que se precisa en su búsqueda, la entidad buscadora optimiza sus tiempos de rastreo. Si no tienen éxito en la primera búsqueda, ejecuta una búsqueda completa.

Para los OA con metadatos estandarizados el buscador encuentra la información suficiente para valorar al OA ejecutando menos cantidad de código que para OA que no cuentan con metadatos estandarizados. Debido a que el buscador solo se trata de una pequeña aplicación personalizada, no se ejecuto sobre un repositorio completo que contenga gran cantidad de OA, solamente se ejecuto sobre un OA a la vez, pero fue suficiente para observar el comportamiento de este programa y con ello obtener los resultados de cada corrida sobre los archivos y concluir que ejecuta mas líneas de código cuando se trata de OA con metadatos que no están estandarizados. Si este proceso fuera efectúado a un repositorio con gran cantidad de OA y tuvieran que ser revisados todos los OA existentes en ese repositorio, aquellas líneas de código extra que lee cuando se trata de metadatos no estandarizados, se dispararían dependiendo de la cantidad de OA a verificar y también de los criterios de búsqueda introducidos por el usuario. Existen varias organizaciones que realizan especificaciones de metadatos para los OA, estas organizaciones se basan en el estándar IEEE, sin embargo, la aplicación propuesta, solo contempla cuatro de ellas y para el caso de un repositorio que se basa en las

especificaciones definidas por otra organización, los resultados no son óptimos.

8.4. Pruebas de caja Blanca

Las pruebas de caja blanca son mucho mas amplias, normalmente se denominan pruebas de cobertura o pruebas de caja transparente, al total de pruebas de caja blanca se le llama cobertura, la cobertura es un número porcentual que indica cuanto código del programa se ha probado. Básicamente la idea de pruebas de cobertura consiste en diseñar un plan de pruebas en las que se vaya ejecutando sistemáticamente el código hasta que haya corrido todo o la gran mayoría de el, esto que parece complicado es mas aún cuando el programa contiene código de difícil alcance, como por ejemplo manejadores de errores o çódigo muerto". Entiéndase por código muerto a aquellas funciones y/o procedimientos que hemos incluido por encontrarse en recopilaciones pero que estas nunca son ejecutadas por el programa, estas funciones no necesariamente deberán ser removidas pero si probadas por si algún día en revisiones futuras son incluidas.

Hay que tener en cuenta dos puntos importantes, en primer lugar las pruebas de caja blanca no reemplazan, solo complementan a las de caja negra y de aceptación y en segundo lugar, las pruebas de cobertura deben ser realizadas una ves terminado el software y no deben ser confundidas con las pruebas informales que realiza el programador en momentos de desarrollo, dado que si bien estas van cubriendo distintos fragmentos de cada módulo, normalmente no son eficaces por no tener un diseño apropiado. Puede que habiendo logrado una cobertura de caja negra del 100 por ciento , midamos una cobertura de sentencias del 100 por ciento . En este caso, pasaremos a la fase siguiente: prueba de condiciones. Las pruebas de caja blanca se enfocan al código.

Prueba 39: Lectura del elemento de documento de un archivo XML.

Esta prueba consiste en leer con la aplicación un archivo de metadatos y observar el comportamiento que toma el programa al leer el elemento de documento de este; el elemento de documento es importante debido a que especifica que esquemas es el que se va a usar. El código que se medirá se presenta a continuación: Las pruebas se efectúaron

```
while (rdr->Read())
 {
        switch (rdr->NodeType)
        {
                case XmlNodeType::Document:
                 label4->Text="Nodo de documento";
           break;
           }
}
```

con una entrada como la siguiente: Se obtuvo un reconocimiento satisfactorio.

```
<manifest xmlns= "http://www.imsproject.org/xsd/imscp rootv1p1p2">
```

Prueba 40: Lectura del elemento de tipo elemento de un archivo XML.

Esta prueba consiste en leer con la aplicación un archivo de metadatos y observar el comportamiento que toma el programa al leer un elemento de este; este elemento si bien es tan común como los demás, para este trabajo tiene gran relevancia debido a que es el que determina las etiquetas de los metadatos. El código que se medirá se presenta a continuación:

```
while (rdr->Read())
 {
        switch (rdr->NodeType)
         {
                case XmlNodeType::Element:
                label4->Text="Nodo de elemento;

          break;
          }
}
```

Las pruebas se efectúaron con una entrada como la siguiente:

```
<catalog>catalogo</catalog>
<date>
   <datetime> </datetime>
</date>
```

Se obtuvo un reconocimiento satisfactorio.

Las pruebas anteriores obtuvieron un reconocimiento satisfactorio, determinando en cada una de ellas el elemento del tipo indicado. La prueba 40, hacia referencia a la identificación de un elemento de tipo elemento, El caso para este elemento contiene especificaciones especiales para identificarlo.

En este caso:

1. Lee los elementos del archivo XML.

2. Utilizando las variables de control, las comprar para identificarlos y ubicarlos en el área correspondiente y en base a esta área, asignarle un valor a la variable de control SelArea. Para con ella identificar el estándar seleccionado.

3. Cuando este leyendo algún elemento lo comparará con las variables de control para verificar en que elemento se encuentra y en que área.

4. Cuando reconoce que elemento es, el caso el elemento encenderá la variable lectura que es la que indica que se tienen un elemento de tipo elemento y que el siguiente elemento que encuentre el verificador deberá ser un elemento de tipo texto para asegurar que el elemento de tipo elemento engloba a un elemento de tipo texto, al encontrar el elemento de tipo texto la variable de lectura se apagara de forma que le indique al verificador que ha leído el elemento que se tiene abierto y que este lo indica el estándar seleccionado y finalmente se dejará la variable lectura cerrada para que cuando el verificador encuentre el elemento de tipo elemento final la comprueba que en verdad esta cerrada y así pueda determinar que el elemento en cuestión contienen información, si la variable de lectura se encuentra abierta, entonces no hubo ningún elemento de texto y por lo tanto ese elemento esta vacío dejando fuera de lo que marca la especificación del estándar seleccionado

Prueba 45: Apertura de la variable lectura en el elemento de tipo elemento.

El objetivo de esta prueba es determinar en que momento la variable lectura se enciende para indicar que el elemento que se espera es un elemento de texto. El código utilizado se muestra a continuación en la figura 8.1. El código utilizado se muestra a continuación, en el cual se muestran tres de los casos a intervenir dependiendo de la variable de selección para el estándar, en cada uno de los casos mostrados, se enciende la variable lectura, pues es la que indica que se esta leyendo el principio de un elemento de tipo elemento.

Prueba 46 y 47.

Las pruebas 46 y 47 se refieren al control de la variable lectura. La prueba 46, se enfoca al código necesario para hacer la lectura de los elemento de tipo texto asi como para cerrar la variable de lectura, con la finalidad de indicar que el elemento de tipo elemento es sucedido con un elemento de tipo texto. Y de esta forma suponer que el elemento de metadato contiene información introducida por el usuario. La prueba 47 hace referencia al elemento de tipo fin de elemento para denotar que se ha concluido la lectura del campo de metadato. En este elemento la variable de lectura deberá estar cerrada para indicar que el campo de metadato esta completo, de lo contrario, es decir, si la variable de lectura continua abierta, denotará que el campo de metadato no cuenta con la información solicitada por el estándar y por lo tanto el OA no se considerará estandarizado. En este caso, se encenderá la variable NoEstandar, que es la que indica el incorrecto llenado del campo de metadatos.

8.5. Pruebas de rendimiento

Una vez realizadas las pruebas de caja blanca, las cuales hacen referencia al código, se efectúarán las pruebas de rendimiento. Las pruebas conocidas con el nombre de pruebas de rendimiento son aquellas que determinan los tiempos de respuesta, el espacio que ocupa el módulo en disco o en memoria, el flujo de datos que genera a través de un canal de comunicaciones, etc.

El resultado deseado es obtener un margen de tiempo inferior a la unidad. El resultado fue de 0.5 decimas de segundo y este resultado no tuvo variantes para las demas pruebas. Por las pruebas anteriores enfocadas al rendimiento, podemos decir que el sistema tiene excelentes tiempos de respuesta.

8.6. Pruebas de robustez

Pruebas de robustez, comúnmente denominadas robustness test"son las encargadas de verificar la capacidad del programa para soportar entradas incorrectas.

Prueba 54: Prueba de Robustez.

Esta prueba esta diseñada para la fase de generación y consiste en introducir información especial sobre los campos de texto, con la finalidad de determinar las posibles fallas del sistema ante el ingreso de datos incongruentes.

El objetivo de esta prueba radica en detectar fallas del sistema cuando este se expone a uso incorrecto.

Debido a que la información que proporciona el usuario es considerada como texto libre, esa información no repercute de manera directa al funcionamiento del sistema, con excepción del campo que requiere el nombre del OA. Ya que si este es introducido de manera diferente al nombre del OA, existirá un problema de error al intentar leerlo en Moodle.

8.7. Pruebas de funcionalidad

Las pruebas de funcionalidad son aquellas que permiten establecer un paámetro que indique que tan funcional es el sistema para el usuario fianl. Para realizar estas pruebas, se conto con la ayuda del departamento de sistemas del CETE (centro de tecnologias educativas) de la Benemérita Universidad Autónoma de Puebla, como también de profesores de cursos en línea de este centro.

El objetivo de estas pruebas es el de conocer que tan funcional es la aplicación para el usuario final y de la misma forma, identificar pocibles mejoras en la interfaz y comportamiento de la aplicación. Las espectativas de la aplicación fuerón cubiertas satisfactoriamente. Se aplico un instrumento de medición de tipo cuestionario para recolectar las observaciones, el instrumento de medición es el siguiente:

Comentarios de la aplicación de metadatos

1. La interfaz de la aplicación en general le permite conducirse a través de ella:
    - Muy bien
    - Bien

- Normal
- Mal
- Muy Mal

2. El generador de Metadatos es:

- Muy Claro
- Claro
- Normal
- Confuso
- Muy Confuso

3. el ingreso de los Metadatos es:

- Muy Fácil
- Fácil
- Normal
- Complicado
- Muy Complicado

4. La descripción de los campos de metadatos es:

- Muy Buena
- Completa
- Suficiente
- Incompleta
- Deficiente

5. El verificador de Metadatos es:

    - Muy Claro
    - Claro
    - Normal
    - Confuso
    - Muy Confuso

6. La presentación de los errores detectados por el verificador es:

    - Muy Clara
    - Clara
    - Normal
    - Confusa
    - Muy Confusa

7. Comentarios:

Las pruebas realizadas arrojarón resultados interesnates, tomando en cuenta que la mayoría de las pruebas tuvierón denotan que la aplicación creada es funcional y eficiente y que algunas de ellas dejarón ver algunos detalles que había que corregir. Estos detalles fuerón corregidos y la aplicación será entregada al departamento de sistemas del CETE para su uso.—

```
case XmlNodeType::Element:
	elemento=rdr->Name;
	if (myCompIntl->Compare("general", elemento)==0)
	{area=1;}
	SelArea=(VerEst->SelectedIndex);
	switch (y)
	{
		case 0:
			label4->Text="Seleccion de lom";
		break;
		case 1:
			label4->Text="Seleccion de scorm 2004";
			if (area==1)
			{
				if (myCompIntl->Compare(entrada, elemento)==0)
				 {
					lectura=1;
					elementoactual=elemento;
				}
			}
			if (area==2)
			{
				if (myCompIntl->Compare(version, elemento)==0)
				{
					lectura=1;
					elementoactual=elemento;
				}
			}
		break;
		case 2:
			label4->Text="Seleccion de scorm 1.2";
			elemento=rdr->Name;
			if (area==1)
			{
				if (myCompIntl->Compare(catalogo, elemento)==0)
				{
					lectura=1;
					elementoactual=elemento;
				}
			}
			if (area==2)
			{
				if (myCompIntl->Compare(version, elemento)==0)
				{
					lectura=1;
					elementoactual=elemento;
				}
			}
			 if (rdr->AttributeCount > 0)
			 {
				Console::Write("  ");
				while (rdr->MoveToNextAttribute())
					Console::Write(" {0}={1}", rdr->Name, rdr->Value);
				Console::WriteLine();
			 }
		break;
	}
break;
```

Figura 8.1: Código usado para determinar la variable de lectura

Apéndice A

Apendice

A.1. .NET vs JAVA

A.1.1. Clases de marco de trabajo

Dado que Java es una plataforma múltiple, el conjunto de clases de marco de trabajo se ha limitado tradicionalmente a las clases disponibles en todas las plataformas. Esto afecta al ámbito y a la variedad de las clases. El ámbito se limita al denominador común inferior de equipos y sistemas operativos disponibles, al tiempo que se reduce la variedad de las propias clases. Las clases de Java integradas son, en general, muy sencillas. Los desarrolladores suelen crear su propio marco de trabajo sobre éstas o buscan marcos de trabajo de terceros (por ejemplo, para el análisis de documentos XML). Para analizar documentos XML en Java antes del lanzamiento de JDK 1.4, era necesario utilizar una biblioteca de otro fabricante o escribir un analizador propio, lo que equivalía a más de 100 líneas de código. .NET *Framework* se crea sobre la plataforma Windows. Asimismo, se encuentra disponible una versión de Common Language Runtime y la biblioteca de clases de .NET Framework para *FreeBSD*. Las clases permiten obtener acceso al ámbito completo de lo que se encuentra disponible en esta plataforma. Además, las

posibilidades que ofrecen estas clases son mayores que las de Java: .NET proporciona un marco de trabajo de programación en lugar de una base sobre la cual crear un marco de trabajo de programación. La fidelidad de .NET Framework es superior a la de la plataforma Java. Los programas de .NET cuentan con toda la eficacia de Windows, a diferencia de lo que sucedía con el denominador común inferior de todas las plataformas en Java. "lo verdaderamente importante es que trabajar con .NET significa poder disfrutar de todas las herramientas compatibles con estándares que ofrece Microsoft y que, probablemente, necesitan el 90 por ciento de los desarrolladores. En cambio, Java supone todavía componer una solución a partir de varias piezas recopiladas de distintas fuentes."[**?**]

A.1.2. Escribir una vez y depurar en todas partes

Aunque el ideal de Java resulta atractivo, por su lema es que es "escribir una vez y ejecutar en cualquier parte", la mayoría de las aplicaciones actuales se aplican a un único sistema operativo. Esto se debe a dos razones técnicas. En primer lugar, debido a que el ámbito y la variedad del marco de trabajo de Java son limitados, los desarrolladores suelen utilizar clases de propietario para obtener acceso a las características que se encuentran disponibles para la plataforma de destino. En segundo lugar, debido a que las incompatibilidades persistentes entre las diferentes implementaciones de Java han exigido que se lleven a cabo continuos esfuerzos de plataforma cruzada, los desarrolladores deben probar su código en todas las plataformas con las que deseen obtener compatibilidad. Por lo tanto, algunos desarrolladores suelen llamar a Java, en clave de humor, la plataforma "escribir una vez y depurar en todas partes".

A.1.3. Seguridad

.NET ofrece más seguridad. Además de las API utilizadas para la criptografía, las cookies de seguridad y la autenticación, existe un mecanismo enriquecido y extensible para la asignación de distintos niveles de permisos a conjuntos de usuarios diferentes en función del origen del código. Por ejemplo, todo el código de Microsoft se puede considerar seguro. Se puede otorgar (o denegar) acceso de lectura y escritura sobre determinados recursos (tales como la pantalla o el directorio temporal) al código que proceda de una URL específica. Asimismo, se pueden marcar algunas aplicaciones para especificar que necesitan determinados permisos (por ejemplo, el acceso al sistema de archivos). Si estos permisos no se encuentran disponibles, la aplicación no se cargará. Puesto que la comprobación de seguridad se realiza en el momento de la carga, no es necesario que el desarrollador escriba en la aplicación lógica para la detección de errores de seguridad. Todo este proceso se administra a través de una nueva tecnología, llamada "seguridad de acceso al código", que funciona con código cargado desde Internet o el equipo local.

A.1.4. Control de versiones

El sistema de control de versiones de .NET es más robusto que el de Java. En Java, si se instalan dos versiones de la misma clase, el tiempo de ejecución sólo cargará la primera clase que encuentre en la ruta de acceso de clases, sin tener en cuenta la versión. En cambio, .NET se ha diseñado para ofrecer una amplia compatibilidad con todo tipo de versiones. Con la ejecución página tras página, se pueden cargar y ejecutar a la vez varias versiones diferentes de la misma clase. Cada aplicación puede seleccionar su mecanismo de enlace: enlazar a una versión específica de una biblioteca de clases, la más reciente, o bien, a una versión que se encuentre en una ubicación

específica. Asimismo, se pueden aplicar directivas por equipos para reemplazar o volver a definir más adelante el control de versiones. Este control de versiones se aplica también a los lenguajes de programación. Por ejemplo, en el lenguaje C=, los métodos son no virtuales de forma predeterminada y, por tanto, los desarrolladores deben declarar explícitamente métodos reemplazados. Esto significa que los componentes no pueden alterar accidentalmente el comportamiento de las clases base. C= no exige al desarrollador que declare explícitamente las excepciones que puede generar un método: el código cliente seguirá funcionando aunque cambie la lista de excepciones.

A.1.5. Rendimiento

La característica de .NET que realmente destaca sobre las de Java es la capacidad para definir el rendimiento y el comportamiento de las aplicaciones. Las unidades principales de creación de aplicaciones se han diseñado para ofrecer escalabilidad y un rendimiento elevado. Por ejemplo, con el acceso a datos desconectado de ADO.NET, se pueden realizar en memoria un número mayor de funciones de datos, lo que en JDBC (Java Database Connectivity) exige un recorrido de ida y vuelta a la base de datos. Además, la arquitectura de la aplicación .NET se ha diseñado para que ésta sea más rápida. Por ejemplo, las páginas Web ASP.NET ofrecen un rendimiento altamente mejorado en comparación con las páginas Active Server (ASP), las páginas Java Server Pages (JSP) y los lenguajes de secuencias de comandos incrustados HTML. Si lo comparamos con la plataforma Java, el rendimiento de .NET resulta mejor no sólo en escenarios de un único usuario, sino también en escenarios multiusuario. "El rendimiento de la aplicación .NET de Microsoft demostró claramente ser superior al de la aplicación Java: en un escenario no almacenado en caché lograba responder a 5.000 usuarios virtuales a una velocidad 10 veces mayor y en un escenario almacenado en caché el rendimiento

era el doble de rápido"[27].

A.1.6. Migraciones mixtas

Los componentes escritos en lenguajes compatibles con .NET diferentes pueden interoperar fácilmente entre ellos, lo que permite que se realicen migraciones mixtas desde Java. En una migración de este tipo, unos componentes de la aplicación se actualizan a Visual J= .NET y otros se convierten a C=.

Debido a lo anterior, se opto por utilizar la tenología desarrollada por .NET.

Apéndice B

Formatos de pruebas realizadas

B.1. Fomularios de pruebas de caja negra

PRUEBA No 1 Creación de metadatos.

Descripción:

La prueba consiste en introducir información a la aplicación para generar metadatos de un OA específico.

Propósito de la prueba:

Probar que la aplicación crea metadatos para un OA específico.

Actividades:

1. Ingresar información a la aplicación.
2. Generar metadatos.
3. Comparar visualmente metadatos generados con metadatos estandarizados por LOM existentes.

Resultados esperados

1. Para el ingreso de información:

   - El sistema deberá permitir en los campos de texto la introducción de información en formato libre por parte del usuario; la introducción de información en los capos de selección será restringida al vocabulario presentado por la aplicación.

2. Para la generación de metadatos:

   - El sistema deberá generar un archivo de formato XML que contenga la información proporcionada por el usuario. Esta información deberá generarse sintáctica y semánticamente correcta. El archivo de formato XML generado deberá crearse en la ubicación definida por el usuario.

3. Para la comparación de metadatos:

   - El archivo XML generado deberá se idéntico al archivo XML estándar.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA No 2 Creación de metadatos bajo el estándar SCORM 1.2

Descripción:

La prueba consiste en introducir información a la aplicación para generar metadatos en base al estándar SCORM 1.2 de un OA específico.

Propósito de la prueba:

Probar que la aplicación crea metadatos en base al estándar SCORM 1.2 para un OA específico.

Actividades:

1. Ingresar información a la aplicación
2. Generar metadatos
3. Comparar visualmente metadatos generados con metadatos estandarizados por SCORM 1.2 existentes.

Resultados esperados

1. Para el ingreso de información:
    - El sistema deberá permitir en los campos de texto la introducción de información en formato libre por parte del usuario; la introducción de información en los capos de selección será restringida al vocabulario presentado por la aplicación.
2. Para la generación de metadatos:

- El sistema deberá generar metadatos que contengan la información proporcionada por el usuario. Esta información deberá generarse sintáctica y semánticamente correcta determinada por SCORM 1.2. El archivo de formato XML generado deberá crearse en la ubicación definida por el usuario.

3. Para la comparación de metadatos:

   - El archivo XML generado deberá se idéntico al archivo XML generado en base a SCORM 1.2.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA No 3 Creación de metadatos bajo el estándar SCORM 2004

Descripción:

La prueba consiste en introducir información a la aplicación para generar metadatos en base al estándar SCORM 2004 de un OA específico.

Propósito de la prueba:

Probar que la aplicación crea metadatos en base al estándar SCORM 2004 para un OA específico.

Actividades:

1. Ingresar información a la aplicación
2. Generar metadatos
3. Comparar visualmente metadatos generados con metadatos estandarizados por SCORM 2004 existentes.

Resultados esperados

1. Para el ingreso de información:
    - El sistema deberá permitir en los campos de texto la introducción de información en formato libre por parte del usuario; la introducción de información en los capos de selección será restringida al vocabulario presentado por la aplicación.
2. Para la generación de metadatos:

- El sistema deberá generar metadatos que contengan la información proporcionada por el usuario. Esta información deberá generarse sintáctica y semánticamente correcta determinada por SCORM 2004. El archivo de formato XML generado deberá crearse en la ubicación definida por el usuario.

3. Para la comparación de metadatos:

   - El archivo XML generado deberá se idéntico al archivo XML generado en base a SCORM 2004.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA No 4 Creación de metadatos bajo el estándar IMS

Descripción:

La prueba consiste en introducir información a la aplicación para generar metadatos en base al estándar IMS de un OA específico.

Propósito de la prueba:

Probar que la aplicación crea metadatos en base al estándar IMS para un OA específico.

Actividades:

1. Ingresar información a la aplicación
2. Generar metadatos
3. Comparar visualmente metadatos generados con metadatos estandarizados por IMS existentes.

Resultados esperados

1. Para el ingreso de información:
    - El sistema deberá permitir en los campos de texto la introducción de información en formato libre por parte del usuario; la introducción de información en los capos de selección será restringida al vocabulario presentado por la aplicación.
2. Para la generación de metadatos:

- El sistema deberá generar metadatos que contengan la información proporcionada por el usuario. Esta información deberá generarse sintáctica y semánticamente correcta determinada por IMS. El archivo de formato XML generado deberá crearse en la ubicación definida por el usuario.

3. Para la comparación de metadatos:

   - El archivo XML generado deberá se idéntico al archivo XML generado en base a IMS.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA No 5 Creación de metadatos bajo el estándar DUBLIN CORE

Descripción:

La prueba consiste en introducir información a la aplicación para generar metadatos en base al estándar DUBLIN CORE de un OA específico.

Propósito de la prueba:

Probar que la aplicación crea metadatos en base al estándar DUBLIN CORE para un OA específico.

Actividades:

1. Ingresar información a la aplicación
2. Generar metadatos
3. Comparar visualmente metadatos generados con metadatos estandarizados por DUBLIN CORE existentes.

Resultados esperados

1. Para el ingreso de información:
    - El sistema deberá permitir en los campos de texto la introducción de información en formato libre por parte del usuario; la introducción de información en los capos de selección será restringida al vocabulario presentado por la aplicación.
2. Para la generación de metadatos:

- El sistema deberá generar metadatos que contengan la información proporcionada por el usuario. Esta información deberá generarse sintáctica y semánticamente correcta determinada por DUBLIN CORE. El archivo de formato XML generado deberá crearse en la ubicación definida por el usuario.

3. Para la comparación de metadatos:

   - El archivo XML generado deberá se idéntico al archivo XML generado en base a DUBLIN CORE.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA N 6 Funcionamiento del OA en base a LOM sobre Moodle.

Descripción:

El OA será empaquetado en el se encuentra en el IMSMANIFEST Content Package e invocado desde Moodle.

Propósito de la prueba:

Determinar si el OA con los metadatos estandarizados por LOM puede ser leído en Moodle.

Actividades:

1. Crear el Content Package
2. Cargar el Content Package en Moodle
3. Invocar al OA para ser leído en Moodle.

Resultados esperados:

1. Cargar el Content Package en Moodle:
    - El Content Package deberá ser reconocido por la plataforma Moodle.
2. Invocar al OA para ser leído en Moodle:
    - El OA y los metadatos contenidos en el Content Package deberán ser leídos por Moodle a modo de que este, presente el contenido del OA al usuario.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA N 7 Funcionamiento del OA en base a SCORM 1.2 sobre Moodle.

Descripción:

El OA será empaquetado en el IMSMANIFEST Content Package e invocado desde Moodle.

Propósito de la prueba:

Determinar si el OA con los metadatos estandarizados por SCORM 1.2 puede ser leído en Moodle Alcance: La prueba solo intentará cargar el OA en Moodle, son modificar nada; la prueba solo comprenderá al estándar SCORM 1.2. Estrategia: Observación.

Actividades:

1. Crear el Content Package
2. Cargar el Content Package en Moodle
3. Invocar al OA para ser leído en Moodle.

Resultados esperados

1. Cargar el Content Package en Moodle:
    - El Content Package deberá ser reconocido por la plataforma Moodle.
2. Invocar al OA para ser leído en Moodle:
    - El OA y los metadatos contenidos en el Content Package deberán ser leídos por Moodle a modo de que este, presente el contenido del OA al usuario.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA N 8 Funcionamiento del OA en base a SCORM 2004 sobre Moodle.

Descripción:

El OA será empaquetado en el IMSMANIFEST Content Package e invocado desde Moodle.

Propósito de la prueba:

Determinar si el OA con los metadatos estandarizados por SCORM 2004 puede ser leído en Moodle Alcance: La prueba solo intentará cargar el OA en Moodle, son modificar nada; la prueba solo comprenderá al estándar SCORM 2004. Estrategia: Observación.

Actividades:

1. Crear el Content Package
2. Cargar el Content Package en Moodle
3. Invocar al OA para ser leído en Moodle.

Resultados esperados

1. Cargar el Content Package en Moodle:
    - El Content Package deberá ser reconocido por la plataforma Moodle.
2. Invocar al OA para ser leído en Moodle:
    - El OA y los metadatos contenidos en el Content Package deberán ser leídos por Moodle a modo de que este, presente el contenido del OA al usuario.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA N 9 Funcionamiento del OA en base a IMS sobre Moodle.

Descripción:

El OA será empaquetado en el IMSMANIFEST Content Package e invocado desde Moodle.

Propósito de la prueba:

Determinar si el OA con los metadatos estandarizados por IMS puede ser leído en Moodle Alcance: La prueba solo intentará cargar el OA en Moodle, son modificar nada; la prueba solo comprenderá al estándar IMS. Estrategia: Observación.

Actividades:

1. Crear el Content Package
2. Cargar el Content Package en Moodle
3. Invocar al OA para ser leído en Moodle.

Resultados esperados

1. Cargar el Content Package en Moodle:
    - El Content Package deberá ser reconocido por la plataforma Moodle.
2. Invocar al OA para ser leído en Moodle:
    - El OA y los metadatos contenidos en el Content Package deberán ser leídos por Moodle a modo de que este, presente el contenido del OA al usuario.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA N 10 Funcionamiento del OA en base a DUBLIN CORE sobre Moodle.

Descripción:

El OA será empaquetado en el IMSMANIFEST Content Package e invocado desde Moodle.

Propósito de la prueba:

Determinar si el OA con los metadatos estandarizados por DUBLIN CORE puede ser leído en Moodle Alcance: La prueba solo intentará cargar el OA en Moodle, son modificar nada; la prueba solo comprenderá al estándar DUBLIN CORE. Estrategia: Observación.

Actividades:

1. Crear el Content Package
2. Cargar el Content Package en Moodle
3. Invocar al OA para ser leído en Moodle.

Resultados esperados

1. Cargar el Content Package en Moodle:
    - El Content Package deberá ser reconocido por la plataforma Moodle.
2. Invocar al OA para ser leído en Moodle:
    - El OA y los metadatos contenidos en el Content Package deberán ser leídos por Moodle a modo de que este, presente el contenido del OA al usuario.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por el Centro de Tecnologias Educativas: Lic. Javier Bautista

PRUEBA N 11 Funcionamiento del OA con metadatos incorrectos sobre Moodle

Descripción:

Obtener metadatos no estandarizados y con ellos construir el paquete IMSMANIFEST, al tener el paquete construido, se carga en Moodle como una actividad SCORM, y se almacenará en el repositorio local del LMS.

Propósito de la prueba:

Determinar si el OA puede ser leído en Moodle cuando este no tenga información de metadatos.

Actividades

1. Crear metadatos no estandarizados
2. Crear el Content Package
3. Cargar el Content Package en Moodle
4. Invocar al OA para ser leído en Moodle.

Resultados esperados

1. Cargar el Content Package en Moodle:
    - El Content Package no deberá ser reconocido por la plataforma Moodle.
2. Invocar al OA para ser leído en Moodle:
3. El OA y los metadatos contenidos en el Content Package no deberán ser leídos por Moodle.
4.

Responsables

Por el departamento de Postgrado Priego Meléndez Juan Leobardo.

Por la Universidad del Valle de Puebla Profa. Elaine Allard

Apéndice C

Articulos y exposisiones

Bibliografía

[1] ADL, *Advanced distributed learning*, http://www.adlnet.org/ (2005).

[2] ———, *Content agregation model*, www.adlnet.org (2005).

[3] ———, *Run time environment*, www.adlnet.org (2005).

[4] ———, *Scorm an overview*, www.adlnet.org (2005).

[5] AICC, *Aviation industry cbt (computer based training) committee*, http://www.aicc.org/ (2005).

[6] ARIADNE, *Foundation for the european knowledge pool*, http://www.ariadneeu.org/ (2005).

[7] Dublin Core, *Dublin core*, http://dublincore.org/ (2005).

[8] Dino Esposito, *Programación en xml para microsoft .net*, ISBN: 84-481-3814-7 (2006).

[9] M. Gonzalez, *How to develop contents for on line training based on learning objects*, ComuNET (2005).

[10] IEEE, *Institute of electrical and electronics engineers*, http://www.ieee.org (2005).

[11] LTSC IEEE, *Ieee 1484.12.1-2002*, http://ltsc.ieee.org/wg12/par1484-12-1.html (2002).

[12] IMS, *Ims global consortium*, http://www.imsglobal.org/ (2005).

[13] L. Iriarte, *Generación de una biblioteca de objetos de aprendizaje (oa) a partir de contenidos preexistentes*, Revista Redes Murcia (2005).

[14] ISO, *International organization for standardization*, http://www.iso.org/ (2005).

[15] F. López J. Priego, *Estándares en el aprendizaje electrónico*, Tercer congreso nacional de ciencias de la computación, Benemérita Universidad Autónoma de Puebla (2005).

[16] ———, *Construcción de objetos de aprendizaje estandarizados*, I simposio de enseñanza de ciencias básicas 2006, Universidad Autónoma de Yucatán (2006).

[17] ———, *Construcción y verificación estandarizada de metadatos para objetos de aprendizaje*, Congreso CORE, Research in computing Science 2006, ISSN: 1665-9899 CORE 2006 (2006).

[18] ———, *Objetos de aprendizaje para e-learning*, I semana de la computacón 2006 de la Universidad Alva Edisón (2006).

[19] A. Olsen J. Templeman, *Microsoft visual c++ .net*, ISBN: 84-481-3339-0 (2006).

[20] R. Morales, *Capacitación basada en objetos reutilizables de aprendizaje*, Boletn IIE. (2002).

[21] Glenford J. Myers, *Software engineering: a practitioner's approach by roger s. pressman*, (2005).

[22] R. Polsany, *Use and abuse of reusable learning objects*, Journal of Digital Information (2003).

[23] RAE, *Diccionario de la real academia de la lengua pspañola*, www.rae.es (2005).

[24] J. Muñoz S. Pomares J. Aguilar, *Guías de diseño para el desarrollo de objetos de aprendizaje*, Workshop Learning Objects Technology in the Fifth Mexican International Conference on Computer Science (2004).

[25] M. Sicilia, *Reusabilidad y reutilización de objetos didácticos: mitos, realidades y posibilidades*, Revista Redes Murcia (2005).

[26] C. Velazquez, *La importancia de la definición de calidad del contenido de un objeto de aprendizaje*, ENC 2005 (2005).

[27] VeriTest, *Prueba comparativa del rendimiento de microsoft(r) .net pet shop realizada por veritest(r)*, http://www.gotdotnet.com/team/compare/veritest.aspx (2002).

[28] V.Sánchez, *Prototipo de patrimonio público de recursos educativos basado en una red institucional y un repertorio distribuido de objetos de aprendizaje*, ENC 03 (2003).

[29] Fabiola Lopez y Lopez, *Introducción a compiladores*, BUAP (2005).