UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"

ESQUEMA ARQUITECTURAL DE SOFTWARE PARA EDUCACIÓN A DISTANCIA

ING. JOHANNA C. PÁEZ SUÁREZ

Barquisimeto, Diciembre 2006

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA
MAESTRÍA EN CIENCIA DE LA COMPUTACIÓN
MENCIÓN: INGENIERÍA DE SOFTWARE

ESQUEMA ARQUITECTURAL DE SOFTWARE PARA EDUCACIÓN A DISTANCIA

Trabajo presentado para optar al grado de
Magíster Scientiarum

Por: ING. JOHANNA C. PAEZ SUÁREZ

Barquisimeto, Diciembre 2006

ESQUEMA ARQUITECTURAL DE
SOFTWARE PARA EDUCACIÓN A DISTANCIA

Por: ING. JOHANNA C. PÁEZ SUÁREZ

Trabajo de grado aprobado

_____	**_____**
Jurado 1	**Jurado 2**

_____
Jurado 3

Barquisimeto, _____ de _____ de 200__

DEDICATORIA

A mis Padres, por ser siempre
el motor que impulsa y apoya todas mis metas.

A mis hermanas, para que el cumplimiento
de este logro sea un estímulo más para su superación.

A mi amor, por que me comprende,
me ayuda y respeta mi espacio.

A mi familia, por su apoyo incondicional.

AGRADECIMIENTO

Ante todo a Dios, por ser siempre mi guía, y
darme fuerzas y sabiduría para culminar esta meta.

A mis padres, a mis hermanas, y mi familia
por todo su apoyo y colaboración

A la Ing. Luz Estrella Mendoza, por ser siempre
mi apoyo, mi motivación y mayor empuje
en el cumplimiento de esta meta.

Al Ing. Ramón Valera, por todos sus
conocimientos y aportes a esta investigación.

Al Prof. Edison Sira y al Prof. Julio Veliz,
por sus aportes a esta investigación.

Y a todas las personas que de una u otra
forma contribuyeron al logro de esta meta.

INDICE GENERAL

Página

INDICE DE FIGURAS

Figura **Página**

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"

DECANATO DE CIENCIAS Y TECNOLOGÍA

MAESTRÍA EN CIENCIA DE LA COMPUTACIÓN

MENCIÓN: INGENIERÍA DE SOFTWARE

ESQUEMA ARQUITECTURAL DE SOFTWARE PARA EDUCACIÓN A DISTANCIA

Autora: Ing. Johanna Páez
Tutor: M.Sc. Ing. Ramón Valera

RESUMEN

La presente investigación tiene como objetivo central diseñar una arquitectura de software, que permita publicar objetos de contenidos, siguiendo las normas de empaquetamiento de información proporcionado por el modelo estándar SCORM y basada en lenguaje de patrones. Dicha arquitectura muestra una posible solución a instituciones que posean plataformas de formación (LMS) que no están desarrolladas bajo el estándar SCORM para que puedan ofrecer o poner a disposición sus cursos a otras plataformas de formación que sean capaz de interpretar el objeto de contenido, a través de la API desarrollada bajo el estándar SCORM, brindando así una herramienta ante el problema de interoperabilidad entre plataformas de diferentes fabricantes. Para el diseño de la arquitectura se utiliza la metodología recomendada por Yacoub y Ammar (2004) en su libro "Patterns-Oriented Analysis and Desing" la cual dicta que para el desarrollo de cualquier sistema orientado a patrones, se debe comenzar por determinar los requerimientos funcionales y no funcionales del sistema. Una vez que se conoce que se necesita, se realiza una búsqueda de información referente patrones que puedan ser posibles soluciones a los problemas que se presentan, luego se realiza una selección de los patrones más convenientes y se procede a realizar los diagramas que soporten y permitan visualizar la interrelación que existirá entre los patrones seleccionados y que darán solución al problema presentado. Esta investigación está concebida bajo la modalidad de un estudio monográfico documental, para la cual se utilizó la técnica de revisión bibliográfica sobre los temas principales: Estándares de Interoperabilidad entre plataformas (SCORM) y Patrones. El alcance de la investigación está delimitado por el diseño de la propuesta, dejando para posteriores investigaciones el desarrollo e implementación de la misma.

Palabras Claves: e-Learning, Arquitectura de Software, Lenguaje de Patrones, SCORM, Plataformas de Formación (LMS).

INTRODUCCION

Uno de los problemas más significativos que confrontan las sociedades actuales, derivados de la incorporación del avance tecnológico, es su adecuación a la tecnología en informática y computación existente en el mercado. En efecto el domino de la comunicación de datos se ha convertido en una parte fundamental de los medios de comunicación de las sociedades modernas.

Internet ha transformado la forma en que la gente se comunica, trabaja, hace compras y pasa su tiempo libre. Con el uso de Internet (www – World Wide Web), la gente puede encontrar e intercambiar una amplia gama de información de manera sencilla, y dado su potencial de llegar a una gran cantidad de personas en cualquier momento y en cualquier lugar, Internet se está transformando en una de las herramientas más versátiles, accesibles y económicas utilizadas por las personas y las empresas de todo el mundo.

Cada unas de las ventajas proporcionadas por Internet son aprovechadas en el ámbito educativo, para proporcionar a las personas una alternativa de aprendizaje adaptada a sus necesidades, es decir, si posee un horario de trabajo complicado, si no tienen las posibilidades de trasladarse al lugar geográfico donde se dicta un cierto curso que es bastante importante para su desarrollo profesional, pueden desde la comodidad de su hogar o cualquier sala de navegación acceder al aprendizaje de manera fácil, adaptado a sus posibilidades de tiempo, su ritmo de aprendizaje, y muchas otras variantes que pueden afectar el desarrollo profesional de una persona.

Esto ha provocado que numerosas organizaciones se hayan embarcado en la implementación y difusión de sistemas de enseñanzas, lo que conlleva a la necesidad de establecer recomendaciones y estándares que permitan su uso eficiente. La estandarización de las tecnologías aplicadas a la formación, pretende facilitar la

reutilización de recursos y la interoperabilidad entre sistemas y software heterogéneos.

En este orden de ideas, la reutilización e interoperabilidad de recursos que presenta el estándar, sería la adecuación de la plataforma de formación, agregándole la implementación de una API que sea capaz de importar e interpretar los contenidos de aprendizajes que vienen encapsulados en un objeto denominado: objeto de contenido intercambiable (SCO) según lo establece el modelo estándar SCORM.

La presente investigación propone una forma de distribuir cursos de plataformas que no están desarrolladas bajo el estándar SCORM, permitiendo que los contenidos sean encapsulados en un SCO cumpliendo con las normas de empaquetamiento de información del modelo SCORM para que queden a disposición de plataformas de formación que si estén desarrolladas bajo este estándar, sin tener que realizar ninguna modificación a las plataformas ya existentes. De esta manera el estudio se encuentra dividido en capítulos, los cuales se detallan a continuación:

Capitulo I El Problema. Se inicia con el planteamiento de la situación que genera el trabajo investigativo, así mismo se describen los objetivos del mismo, la justificación e importancia, alcances y limitaciones, representando estos aspectos las bases que conducen la investigación.

Capitulo II Marco Teórico. Se presentan los antecedentes y las bases teóricas y filosóficas que soportan el conocimiento del tema en estudio.

Capitulo III Marco Metodológico. Se describen los aspectos metodológicos que conducen la realización del estudio diagnóstico. Seguidamente se presenta la metodología de Yacoub y Ammar (2004), la cual muestra un camino a seguir en el desarrollo de aplicaciones y sistemas orientados a patrones.

Capitulo IV Resultados. Dado el estudio diagnostico y la metodología de trabajo presentada en el capitulo anterior, se presentan todas las fases de la metodología Yacoub y Ammar (2004), que fueron necesarias para producir el diseño de la arquitectura de software propuesta, la cual esta dirigida a la publicación de objetos de aprendizaje SCO partiendo de contenidos existentes en LMS que no posean el formato estándar determinado por SCORM, para que puedan ser utilizados por otras plataformas que estén desarrolladas bajo el estándar de interoperabilidad de LMS de diferentes fabricantes denominado SCORM.

Finalmente el Capitulo V Conclusiones y Recomendaciones. Ofrece un conjunto de conclusiones obtenidas del estudio y algunas recomendaciones derivadas de la investigación.

CAPÍTULO I

EL PROBLEMA

Planteamiento del Problema

La tecnología está presente en todos los ambientes del desarrollo social, lo que ha producido notables modificaciones en la forma de proceder de los individuos. Actualmente las computadoras, en buena medida están al acceso de la mayoría de las personas, y por supuesto el uso de Internet se ha incrementado a todos los niveles. Esto ha originado a su vez un cambio en los procesos de producción, distribución, organización del trabajo y fundamentalmente un cambio en el sector servicios cuya base primordial lo constituye la información. Al respecto Viso (2004) señaló:

> …ninguno de estos avances ha causado un impacto tan hondo en la sociedad como la transmisión de datos informáticos por medios electrónicos, permitiendo la interconexión de ordenadores para conformar una red capaz de transferir información por medio de los mensajes de datos, sin importar la distancia geográfica. El ejemplo paradigmático de esta nueva forma de comunicación es Internet. (p. 19)

Esta situación ha provocado que se hable de un nuevo orden social, caracterizado por la importancia dada a la información y que ha sido llamado Sociedad de Información, donde se describen los cambios sociales, económicos y de otra naturaleza que han surgido de la transición de una economía industrial, fundamentada en la fabricación o en la producción en masa, a otra basada en la información, concebida ésta como actividad y como bien, así como la principal fuente de riqueza. Tal como lo menciona García (2002) en los siguientes términos:

> Este es el panorama de este nuevo fenómeno científico tecnológico en las sociedades modernas. Por ello ha llegado a sostenerse que la informática es hoy una forma de Poder Social. Las facultades que el fenómeno pone a disposición de Gobiernos y de particulares, con rapidez y ahorro consiguiente de tiempo y energía, configuran un cuadro de realidades de aplicación y de posibilidades, donde es necesario el derecho para regular los múltiples efectos de una situación nueva y de tantas potencialidades en el medio social. Los progresos mundiales de las computadoras y la fusión del proceso de información con las nuevas tecnologías de comunicación, así como la investigación en el campo de la inteligencia artificial, ejemplifican el desarrollo actual definido como la "era de la información" o "sociedad de información". (p.23)

La sociedad de información es conocida también como sociedad global o globalizada, sociedad informatizada, sociedad del conocimiento, sociedad cibernética, estado telemático, sociedad digital o simplemente cibersociedad. Sus rasgos más característicos de acuerdo a lo señalado por Viso (2004) se pueden resumir en: "La información como base de la economía, una vida globalizada, un mundo digitalizado, la eliminación de fronteras y un aumento - significativo en el desarrollo del campo de los servicios." (p. 21)

El heredero tecnológico más importante del recién finalizado siglo XX es Internet; cuyo crecimiento vertiginoso es el resultado de la convergencia de la informática y las telecomunicaciones, impulsados a su vez por el desarrollo de la microelectrónica digital. Esta realidad tecnológica constituye un hecho social sin precedentes, y como tal está planteando nuevos retos para la sociedad.

La educación no escapa del impacto que proporciona Internet a la sociedad. Con la comercialización de Internet en 1995 se habla de que la enseñanza a distancia entra en su cuarta generación, la cual esta caracterizada por la generación del multimedia interactivo y de las comunicaciones mediadas por ordenadores. Es también la generación de los campos virtuales o plataformas de formación que basan la educación en la conjunción del uso de ordenadores personales con capacidad

multimedia y un soporte para la distribución electrónica de contenidos basados en Internet y la comunicación tanto asincrónica como sincrónica. García (2002)

Las plataformas de formación o también denominas LMS por sus siglas en inglés Learning Managenemt System, son las encargadas de almacenar, administrar y distribuir la información, no solamente académica sino también administrativas, lo que permite que profesores (facilitadores), alumnos (participantes) y administradores compartan un espacio de trabajo.

El desarrollo de estas aplicaciones (LMS) requiere de las herramientas ofrecidas por la Ingeniería de Software (IS), según Sommerville (2002) ésta es "la disciplina que comprende todos los aspectos de la producción de software, desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de este después de que se utiliza". Para ello generalmente se hace uso de las Arquitecturas de Software (AS) que es una de las herramientas ofrecidas por la IS, la cual es considerada por Garlan y Perry (1995) como la estructura de los componentes de un programa o sistema, sus interrelaciones, y los principios y reglas que gobiernan su diseño y evolución a lo largo del tiempo. Por lo tanto la arquitectura de software es una descripción fundamental de un sistema, a través de sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

La AS ofrece a los ingenieros o desarrolladores de estas aplicaciones modelos ya establecidos y probados para que de una forma selectiva, desarrollen soluciones apropiadas al problema de formación a distancia, ya que con la popularización de Internet como medio para la formación, ha provocado que numerosas organizaciones se hayan embarcados en la implementación y difusión de sistemas de enseñanza. Esto ha generado la aparición de una gran multitud de sistemas y recursos educativos, lo que conlleva a la necesidad de establecer recomendaciones y estándares que permitan su uso eficiente.

La estandarización de las tecnologías aplicadas a la formación pretende facilitar la reutilización de recursos y la interoperabilidad entre sistemas y software heterogéneo, para esto se deben lograr los siguientes dos (2) objetivos: Que un curso de cualquier fabricante pueda ser cargado en cualquier LMS de otro fabricantes y; Que los resultados de las actividades de los usuarios en el curso puedan ser registrados por el LMS.

Como respuesta a estas necesidades se han desarrollados varias normas, y las más relevantes se enumeran a continuación: **AICC** (Aviation Industry CBT Comitee ó Comité de la Industria de la Aviación – CBT (Computer Based-Training ó Entrenamiento Basado en Computadoras): éste comité fue creado para desarrollar una normativa para sus proveedores de formación basada en computadores, para garantizar la armonía de los requerimientos de los cursos, así como la homogeneización de los resultados obtenidos de los mismos. **IEEE Learning Technologies Standards Comittee** (LTSC) es un organismo que promueve la creación de una norma ISO y se encarga de preparar las normas técnicas, prácticas y guías recomendadas para el uso informático de los componentes y sistemas de educación y de formación, para facilitar su desarrollo, despliegue, mantenimiento e interoperabilidad. **IMS Global Learning Consortium, Inc** es un consorcio que está formado por miembros provenientes de organizaciones educacionales, empresas públicas y privadas y su principal función es desarrollar y promover especificaciones abiertas para facilitar las actividades del aprendizaje en línea. **ADL (Advanced Distributed Learning)** con su Modelo Referenciado de Objetos de Contenido Compartible ó SCORM por su siglas en inglés (Sharable Courseware Object Reference Model) como lo dice su nombre, es un modelo estándar para intercambiar objetos de contenido (SCO – Sharable Content Objects), la cual nace como producto de una serie de requisitos y lineamientos que definen un modelo para agregar contenidos de aprendizaje en sistemas basados en Internet, y transportarlos para que sean reusables entre las distintas plataformas de formación realizadas por diferentes fabricantes, siempre y cuando las LMS están desarrolladas con lineamento del

entorno de ejecución de SCORM, el cual está dividido en tres grandes aspectos: Gestión del Entorno de Ejecución: que son las normas que indican el proceso del lanzamiento de los objetos de contenido y la gestión de la comunicación entre la plataforma y los SCO; API: describe la sintaxis y la utilización de todas las funciones de interfaz entre la plataforma y los SCO y Modelo de datos del entorno de ejecución de SCORM: que describe exhaustivamente todos los datos y su modelo de comportamiento para la construcción del entorno de ejecución de SCORM y para la gestión de los SCO por parte de la plataforma.

Según Maurer (2004), Este organismo recogió lo mejor de las anteriores normativas, como el sistema de descripción de cursos XML de la IMS, el mecanismo de intercambio de información mediante una API de la AICC, entre otras.

Cada una de estas organizaciones crean un marco de referencia para sus proveedores de cursos, estableciendo normas que garantiza la interoperabilidad, pero para ello orientan el desarrollo de la LMS basado en su normativa, es decir, se debe modificar la plataforma para que cumpla con sus lineamientos. Ahora, ¿que sucede con las plataformas existentes? ¿deben ser modificadas para cumplir con el estándar?. O, ¿se puede diseñar una forma para poder intercambiar la información sin necesidad de modificar las LMS existentes?. Basado en la búsqueda de las respuestas a estas interrogantes, se plantea el objetivo de la presente investigación, para ello se hace uso de los patrones que es otra de las herramientas proporcionada por la IS, los cuales le permiten al desarrollador optimizar sus esfuerzos, para producir un software de calidad, y de la metodología recomendada por Yacoub y Ammar (2004) que proporciona una guía para llevar a cabo un diseño de software orientado a patrones.

El concepto de patrones fue estudiado al comienzo por el arquitecto Christopher Alexander y fue desarrollado desde un punto de vista de la arquitectura y la planificación urbana, para Alexander (1977), un patrón es como una regla que consta de tres partes donde se expresa la relación entre un cierto contexto, un problema y

una solución. Además, si el problema tratado es frecuente, la solución aportada es ampliamente reconocida como buena, está basada en la experiencia y su éxito queda totalmente asegurado.

Su introducción y aceptación en el campo de la ingeniería del software, y concretamente dentro del diseño orientado a objetos, se produjo a partir del gran éxito obtenido por el libro de Gamma (1995). En éste, las ideas originales de Alexander orientadas a la arquitectura y el urbanismo fueron adaptadas al desarrollo de software, definiendo un patrón de diseño como "una descripción de clases y objetos que se comunican entre sí dispuestos para resolver un problema de diseño general en un contexto particular" Gamma (1995). Sin embargo, la idea de un patrón de diseño no se limita únicamente al ámbito concreto del diseño detallado orientado a objetos, pudiéndose encontrar patrones de proceso, de análisis, de arquitectura, específicos de lenguajes de programación (idioms), de interacción, presentación, navegación, etc.

El establecimiento de estos patrones posibilita el aprovechamiento de la experiencia acumulada en el diseño de aplicaciones. Un diseñador experimentado producirá diseños más simples, robustos y generales, y más fácilmente adaptables al cambio. Por otra parte, un buen diseño no debe ser específico de una aplicación concreta, sino que debe basarse en soluciones que han funcionado bien en otras ocasiones. Es por esto que se hará uso del lenguaje de patrones para describir dicha arquitectura. Tal como menciona Buschmann (1996),

> Los patrones le ayudan a construir sobre la experiencia colectiva de ingenieros de software experimentados. Estos capturan la experiencia existente y que ha demostrado ser exitosa en el desarrollo de software, y ayudan a promover las buenas prácticas de diseño. Cada patrón aborda un problema específico y recurrente en el diseño o implementación de un sistema de software.

Es por ello que describir esta arquitectura de software haciendo uso del lenguaje de patrones, garantiza que resuelve un problema, es un concepto probado, la solución no es obvia, describe una relación no solo entre módulos sino entre estructuras y mecanismos y tiene un componente humano significante (ya que el software sirve a las personas).

Según Rebollo (2004) menciona en su trabajo la necesidad de estandarizar los contenidos utilizando SCORM en los siguientes términos:

> ADL reconoce la necesidad de disponer de un estándar, de un modelo de referencia para especificar el contenido instruccional y también las cuestiones referentes a su almacenamiento, presentación al usuario y distribución a través de Internet. Para ello, definen el modelo SCORM, que no es otra cosa que un modelo coordinado para dotar al e-learning con una colección de métodos estándares que puedan ser ampliamente aceptados e implementados.

En atención a lo expuesto, cabe preguntarse ¿será el modelo SCORM el más adecuado para publicar objetos de contenido a partir de una LMS que no lo posea y así puedan ser utilizados por aquellas LMS que si lo posean? ¿Se puede describir dicha arquitectura a través del lenguaje de patrones? ¿Existen dichos patrones? En este orden de ideas se plantean los siguientes objetivos, cuyo alcance permitirá conocer el diseño de la propuesta realizada.

Objetivo General:

Diseñar una arquitectura de software para publicar objetos de contenido intercambiables para plataformas de formación (LMS) que no estén diseñadas bajo el modelo SCORM, haciendo uso de la metodología de Yacoub y Ammar (2004) orientada a patrones.

Objetivos Específicos:

1. Revisar la bibliografía existente referente a la Arquitectura de Software, Estándares para la Interoperabilidad de LMS y Lenguajes de Patrones.
2. Conocer la funcionabilidad del estándar SCORM.
3. Seleccionar los patrones adecuados para el diseño de la arquitectura.
4. Diseñar la arquitectura de software que cumpla con los requerimientos para la publicación de objetos de contenidos intercambiables (SCO) para LMS que no están desarrollada bajo el estándar SCORM usando lenguaje de patrones y UML.

Justificación e Importancia.

Debido a que los Centros Educativos, entre los que se cuenta a las Universidades, ya no constituyen los únicos proveedores de conocimiento, podría afirmarse que el sistema educativo debe adecuarse a los cambios que está exigiendo la sociedad actual y el rol que desempeña el docente debe trazarse hacia la sociedad de la información, donde los centros educativos han perdido, en parte, el liderazgo de la misma. Los profesionales de la docencia, se desempeñan como facilitadores del conocimiento, pero es necesario que se cree un ambiente de sincronización y armonía, entre las nuevas tecnologías de información y comunicación, sus mensajes y el aprendizaje, y son precisamente los entornos educativos los intermediarios ideales para esta sincronización.

El uso de las tecnologías y comunicaciones debe involucrar al docente, al alumno, a la institución y al entorno. Padrón (1998) señaló al respecto que “El éxito de un proceso de aprendizaje depende básicamente de: la frecuencia, la calidad, y el nivel de feed-back que se dan al alumno durante la fase de una tarea de aprendizaje.” Con estas tecnologías, el estudiante tendrá la suficiente libertad para definir su propio estilo de aprendizaje, y contará con una amplia fuente de recursos, lo cual contribuirá a su desarrollo personal y profesional.

Es importante destacar, de acuerdo a lo planteado por Padrón (1998), que ninguna tecnología de información y comunicación reemplaza al docente. Por el contrario, se requiere de un profesional conocedor de su asignatura, pero con una formación tecnológica simultánea, que le permita desarrollar competencias a fin de usar la mejor combinación de estrategias educativas para el logro de aprendizaje innovador.

Hay que resaltar que el éxito de esta nueva estrategia de aprendizaje esta ligada en buena medida a la calidad del software que se presenta en las LMS tanto al estudiante como al docente, es por ello que la Ingeniería de Software nos brinda herramientas que permiten a los desarrolladores garantizar esta calidad. Estas herramientas están representadas en la concepción de una adecuada arquitectura de software orientada a patrones que permitan proporcionar dicha calidad a los usuarios.

Ahora, es primordial que exista interoperabilidad entre las distintas LMS, desde el punto de vista que se puedan intercambiar pequeños componentes instruccionales, autocontenidos y que estos sean reusables por los usuarios de las plataformas y de los sistemas de distribución de contenidos, sin que se requieran cambios drásticos en la arquitectura de las LMS de diferentes fabricantes, según Álvarez (2003) reseñado por Rebollo (2004) la interoperabilidad de las plataformas y su utilización está enmarcada en la búsqueda de economía, pedagogía, tecnología, reutilización, contenidos y productividad.

Por lo tanto, se debe diseñar una arquitectura de software estándar que permita adaptar estos contenidos o crear bloques de contenidos mejor conocido en la literatura como LO (Learning Objects - objetos de aprendizaje), los cuales pueden combinarse de infinitas formas para construir colecciones de objetos que forman lecciones, cursos, módulos, entre otros, sin modificar la LMS existente. Según IEEE, un objeto de aprendizaje, es cualquier entidad digital o no digital, que puede ser utilizada o referenciada durante el aprendizaje basado en el computador.

Para tratar de solucionar el problema de distribución de LO entre plataformas, durante la década pasada surgieron numerosas iniciativas, de las cuales se escogió SCORM debido a que es la más recientemente desarrollada y además que su filosofía recoge lo mejor de muchas de las soluciones anteriores a su aparición.

El estándar SCORM se describirá en los capítulos posteriores, debido a que se utiliza como punto central en esta investigación, ya que se debe conocer la forma como este empaqueta los contenidos en un objeto de aprendizaje, el cual se denomina en el estándar como SCO (Sharable Content Objects - objeto de contenido intercambiable), para diseñar la propuesta que tiene como objetivo tomar los contenidos (lecciones, cursos, módulos, entre otros) de aquellas plataformas que fueron creadas antes de que surgiera el modelo SCORM y que sin tener que modificarla o agregarle una capa más a la plataforma, estos puedan ser encapsulados en un SCO y ponerlos a disposición de cualquier otra LMS que esté desarrollada con el estándar SCORM y que desee ofertar dichos contenidos.

Para el diseño de esta arquitectura se utiliza la metodología de Yacoub y Ammar (2004), la cual recomienda una serie de pasos a seguir para diseñar, desarrollar e implementar software orientado a patrones. Los patrones son soluciones probadas y según Canal (2000) en su texto afirma que:

> La idea que subyace a estos patrones es la constatación de que, aunque la orientación a objetos facilita la reutilización de código, la reutilización efectiva sólo se produce a partir de un buen diseño, basado en el uso de soluciones (los patrones) que han probado su utilidad en situaciones similares. Los patrones no son bibliotecas de clases, sino mas bien un esqueleto básico que cada diseñador adapta a las peculiaridades de su aplicación. Los patrones se describen en forma textual, acompañada de diagramas y pseudocódigo. (p. 26)

La idea expuesta por Canal (2000), conduce a plantear que el uso de patrones es de gran ayuda a la hora de diseñar una aplicación o un componente de software, debido a que los patrones son soluciones probadas y reutilizadas en innumerables

problemas, que presenten las mismas características en el mismo contexto. Es por esto que, por si solos representan la experiencia de diseñadores expertos, que garantizan al desarrollo de cualquier aplicación la calidad y usabilidad deseada, es decir, que sean la medida justa para resolver el problema.

El diseño de esta arquitectura contribuye con la globalización de la información, esto hace que plataformas como por ejemplo, la plataforma SABER utilizada en la Universidad Centroccidental “Lisandro Alvarado” en Barquisimeto, pueda ofrecer sus cursos en línea para que sean ofertados por otras plataformas que soporten el estándar SCORM, como por ejemplo: Angel 6.1, BlackBoard 6, Moodle 1.4, entre otras. Es decir, los participantes de esas plataformas no tienen opción para inscribirse en los cursos ofrecidos por la plataforma SABER de esta universidad, pero se podría tener la opción de exportar estos cursos a otras instituciones que utilicen cualquiera de las plataformas que soporten el modelo estándar SCORM.

Alcance y Limitaciones

Esta investigación está orientada al diseño de una arquitectura de software que sea capaz de crear y publicar objetos de contenidos intercambiables (SCO) bajo las normas de empaquetamiento de información del modelo estándar SCORM (Sharable Courseware Object Reference Model - Modelo Referenciado de Objetos de Contenido Compartible), el cual permite interoperabilidad o el intercambio de contenido entre plataformas de formación de diferentes fabricantes.

El modelo SCORM, es un estándar internacional que surge como iniciativa de ADL (Advanced Distributed Learning) para garantizar el acceso a la educación y a los materiales de calidad ajustables a las necesidades individuales así como su disponibilidad. En éste se reunieron una serie de requisitos y lineamientos que definen un modelo para agregar contenidos de aprendizaje en sistemas basados en Internet, y transportarlos a distintas plataformas. Este estándar proporciona

herramientas, que permiten la modificación de las plataformas de formación por medio de la agregación de una API que sea capaz de interpretar y presentar a sus usuarios los cursos que vienen empaquetados en forma de objetos de contenidos intercambiables.

Pero el objetivo de esta propuesta es diseñar una arquitectura de software, que en lugar de modificar la plataforma, sea capaz de tomar el contenido que se desea distribuir de una LMS que no soporte el modelo SCORM, lo lleve a un modo estándar y luego lo empaquete en un solo objeto bajo los lineamientos de SCORM, al cual se le denomina SCO (Objeto de Contenido Intercambiable), para que luego este objeto esté disponible para ser interpretado por cualquier otra LMS que esté desarrollada con las normas ofrecidas por SCORM y que su contenido pueda ser presentado a sus usuarios.

Para el diseño de esta arquitectura se utiliza la metodología presentada en el libro "Pattern-Oriented Analysis and Desing" de Yacoub y Ammar (2004) la cual proporciona una guía para diseñar, desarrollar e implementar software orientado a patrones. Esta metodología establece que se debe llevar a cabo los siguientes pasos:

- Análisis de requerimientos del sistema.
- Revisión bibliográfica de los patrones.
- Selección de los patrones.
- Diseño del diagrama de nivel de detalle de patrones.

Cada uno de los pasos mencionados anteriormente serán descritos en el capitulo III del presente trabajo, los cuales contribuyen y facilitan la utilización de patrones en el diseño de software.

Esta investigación comprende la propuesta de diseño de la arquitectura y se dejará la parte de desarrollo e implementación para investigaciones posteriores.

Además la creación y publicación del SCO generado por esta arquitectura se realiza en un solo sentido, es decir, se toma la información de la LMS que no soporta SCORM, se empaqueta el contenido para crear el SCO y se publica para que este pueda ser interpretado por una LMS que si lo soporte.

Esta interoperabilidad o distribución de contenido está limitado debido a que la información empaquetada podrá ser publicada en la LMS que soporte SCORM, pero la LMS que no soporta SCORM no podrá publicar contenido que este desarrollado por una LMS que si lo soporte.

CAPÍTULO II

MARCO TEÓRICO

Antecedentes

Los antecedentes de investigación están constituidos por la evolución de la educación a distancia a lo largo del tiempo, además la exposición de algunos trabajos relacionados a la investigación, con respecto a la creación e implementación de una plataforma de formación y la utilización de SCORM, así como también la evolución los estándares que le anteceden en su meta por unificar y transportar la información de una plataforma de formación a otra.

Con respecto a la educación a distancia, algunos estudios han demostrado que en los últimos años se ha observado que la evolución y desarrollo vertiginoso de los medios de comunicación e Internet, conlleva a que el conocimiento haya pasado a ser el principal activo de las sociedades avanzadas. Hoy en día el mundo está caracterizado por una exposición constante al cambio, por una continua sobreabundancia de nueva información y por las oportunidades ofrecidas desde las nuevas Tecnologías de la Información y Comunicación.

Es por ello, que en investigaciones realizadas por la Universidad Nacional Autónoma de México (UNAM, 2005), se plantea que históricamente la educación a distancia "se originó como producto de los grandes cambios económicos y sociales que se fueron dando en la segunda mitad del siglo XIX, abriendo paso al estudio por correspondencia." (p. 3), constituyendo el inicio de esta modalidad de impartir

educación. Los medios utilizados por la UNAM, para ofrecer educación a distancia, se fundamentaron “en el uso del texto impreso distribuido a los estudiantes por correo postal, muchas veces el mismo material de estudio que se utilizaba en la enseñanza presencial”, así como la introducción de guías para ayudar al estudiante, la realización de actividades complementarias a cada lección, cuadernos de trabajo, de ejercicios y de evaluación, todo esto con el objetivo de generar una relación entre el estudiante y el centro o autor del texto.

En el estudio realizado por la UNAM (2005) se indicó que “la segunda etapa de la educación a distancia tiene lugar a partir de 1960, principalmente con la creación de la Open University Británica, la cual tuvo como objetivo brindar educación a los adultos que no pudieron recibirla”. Igualmente, se señaló que los medios utilizados en esta etapa, fueron el teléfono, la televisión, y los recursos audiovisuales (como diapositivas, audiocasettes, videocasettes, etc.). Esto originó un cambio fundamental en los programas de educación a distancia, por cuanto se modificó sustancialmente la propuesta inicial de cursos por correspondencia, al incorporar a la misma los medios señalados anteriormente. Posteriormente, “su desarrollo se encuentra ligado a la evolución de las tecnologías de la información y la comunicación, sobre todo a partir de 1970.” (p.3) Caracterizándose por la introducción de las telecomunicaciones con otros medios educativos, tales como la computadora, y la modalidad de formación a distancia interactiva, en la cual, el CD-ROM se instituye como el medio dominante para almacenar y difundir la información de carácter educativo.

En la Investigación realizada por la UNAM (2005), se indica que con la llegada de Internet, “surgió a partir de los años 90's, la Enseñanza Abierta y a Distancia, la Teleformación o Formación Virtual basada en Internet”, afirmándose que “es la última generación de sistemas interactivos abiertos que se caracterizan por la introducción de las redes de computadoras como ambiente de aprendizaje individual y colectivo.” Señalando más adelante que la educación a distancia se caracteriza por:

> … su interactividad en los procesos de comunicación, que permite el acceso a los recursos universitarios, a recursos de otras instituciones educacionales o no educacionales y la comunicación entre estudiantes, entre tutores y estudiantes y entre estudiantes y expertos. Una de las principales características, de este modelo educativo, es que propicia en los estudiantes, la organización de sus tiempos de trabajo, así como la construcción y autorregulación de sus procesos de aprendizaje. Por otra parte permite a los docentes (que en el caso de la educación en línea son llamados Tutores o Asesores), evaluar y retroalimentar el conocimiento, además de llevar a cabo el control y avance de cada uno de los estudiantes. (p. 5)

Actualmente, la educación en línea se apoya en las herramientas que ofrece Internet como son: correos electrónicos, salas de charlas virtuales (chat), foros, al igual que en medios para propiciar el aprendizaje, tales como señala la UNAM (2005): "el audio, video, materiales impresos y digitalizados", a fin de "establecer la interacción entre: docente-estudiante, estudiante-estudiante." Sin embargo, la falta de una buena planificación de estos recursos, incide en gran parte en dicha interacción, y los estudiantes podrían perder el interés y la motivación.

Todo esto nos exige afrontar el reto educativo de la sociedad actual desde unos objetivos y exigencias diferentes a las que tradicionalmente se demandaba de la formación, siendo ahora relevante desde el punto de vista de "servicio de actualización de conocimientos". La obsolescencia sistemática de los conocimientos, hace que se valoren especialmente características como la Inmediatez (aquí y ahora), la Flexibilidad (a la medida, con distintos formatos y ritmos), la Actualización (con fecha de caducidad), la Concreción (con ahorro de tiempo) y la Aplicabilidad (capaz de generar cambios en el entorno) de los aprendizajes, como lo menciona Diez (2002).

Actualmente, y gracias a las tecnologías de información y comunicación (TIC), la formación a distancia ofrece experiencias educativas, que permiten a un número elevado de estudiantes acceder a actividades de formación, enmarcadas en ambientes

de aprendizaje que responden a las características y necesidades de formación, según lo plantea Peiró (2001).

Es por ello, que el e-Learning se ha convertido en un tema de gran importancia en la mayoría de centros de aprendizajes, institutos de educación y hasta en las universidades del todo el país, ya que permite el albergue de un sinnúmero de personas con características diferentes (edad, situación económica, situación familiar, entre otras), para que estas puedan realizar su deseo de superación, sin límites de horarios, ni espacio físicos y lo mejor de todo, es que lo pueden realizar a su ritmo personal.

Ahora bien, se hará referencia a otras investigaciones realizadas en el mismo campo.

En el VIII Congreso de Educación a Distancia CREAD MERCOSUR/SUL 2004, realizado en Córdoba – Argentina, profesoras de la Facultad de Cs. Exactas Físicas y Naturales – UNSJ, presentaron un trabajo realizado por un grupo de profesionales de esta universidad denominado “Procesos unificados para modelar sistemas de Educación a Distancia”, en el cual se muestra un procedimiento a seguir para modelar cualquier proceso de un sistema de educación a distancia y a su vez evaluar la pertinencia o no de su automatización.

En el mencionado trabajo, se hizo uso de la metodología RUP (Rational Unified Process) para modelar el sistema de educación a distancia, en el cual el equipo de trabajo realizó una analogía entre un Sistema de Educación a Distancia y cualquier Sistema de Información, utilizando los enfoques de la Ingeniería de Negocios y la Ingeniería de Software, “que proveen perspectivas disciplinadas para garantizar tareas y responsabilidades dentro de una organización de desarrollo”, Contreras y Laciar (2004). Para especificar y visualizar los componentes del sistema de software hicieron uso de UML, además éste permite especificar modelos de negocios y otros sistemas.

Utilizando la metodología RUP, detallaron su trabajo en tres grandes actividades, como lo son: Modelado del Negocio, Requerimientos y Análisis y diseño. En la primera de las actividades Modelado del negocio, realizaron una evaluación del estado actual del negocio, evaluaron los objetivos de la organización, realizaron una investigación de los sistemas de educación a distancia existentes en otras universidades, identificando los stakeholders ó los actores involucrados en el sistema. Como actores fueron identificados los clientes (los participantes), los docentes, los competidores y otros stakeholders. A partir de esta investigación realizaron la descripción de la estructura de un Sistema de educación a distancia, determinaron sus límites, problemas y estimaron la capacidad de adaptabilidad a los cambios.

Haciendo uso de UML se muestra la figura 1, que detalla el desarrollo de las actividades anteriores.

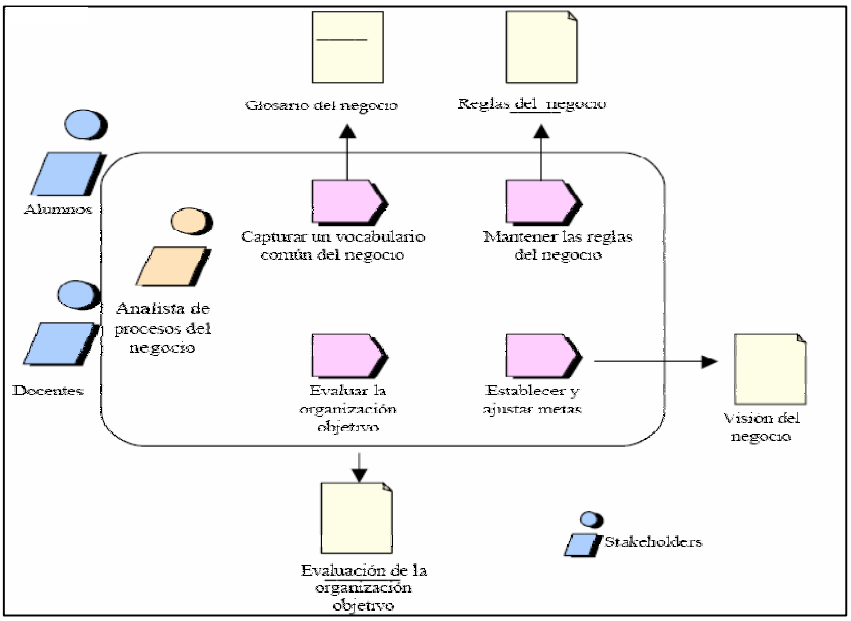

Figura 1: Workflow del Modelado del Negocio. Contreras y Laciar (2004)

Con respecto a la segunda actividad de la metodología, establecieron las necesidades de los stakeholders, identificaron los requerimientos del sistema y de software, el cual lo representaron a través de la figura 2.

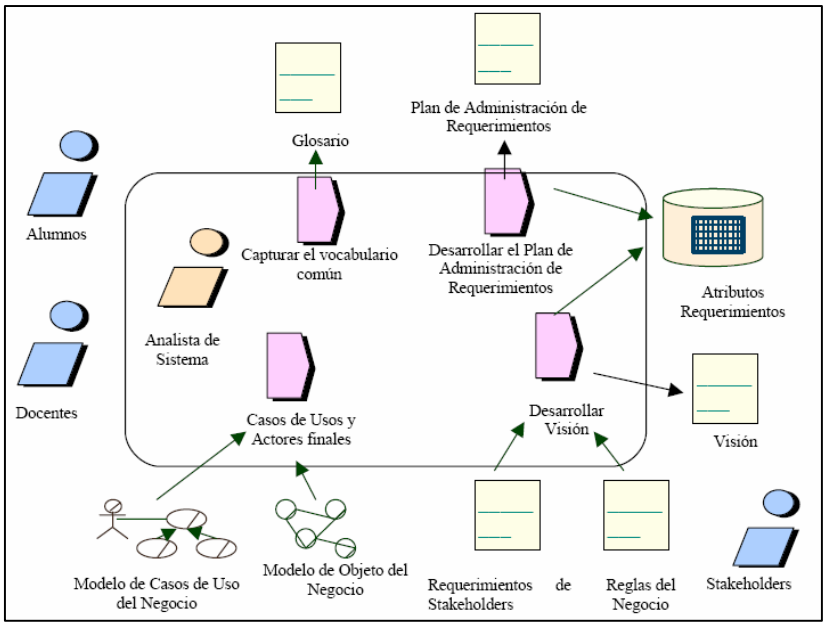

Figura 2: Workflow de Requerimientos. Contreras y Laciar (2004)

Para la tercera actividad, definieron la arquitectura candidata, los componentes de dicha arquitectura como lo son: Vista de despliegue, Vista de implementación, Vista general de navegabilidad y la estructura del sistema; la cual se puede observar en la Figura 3.

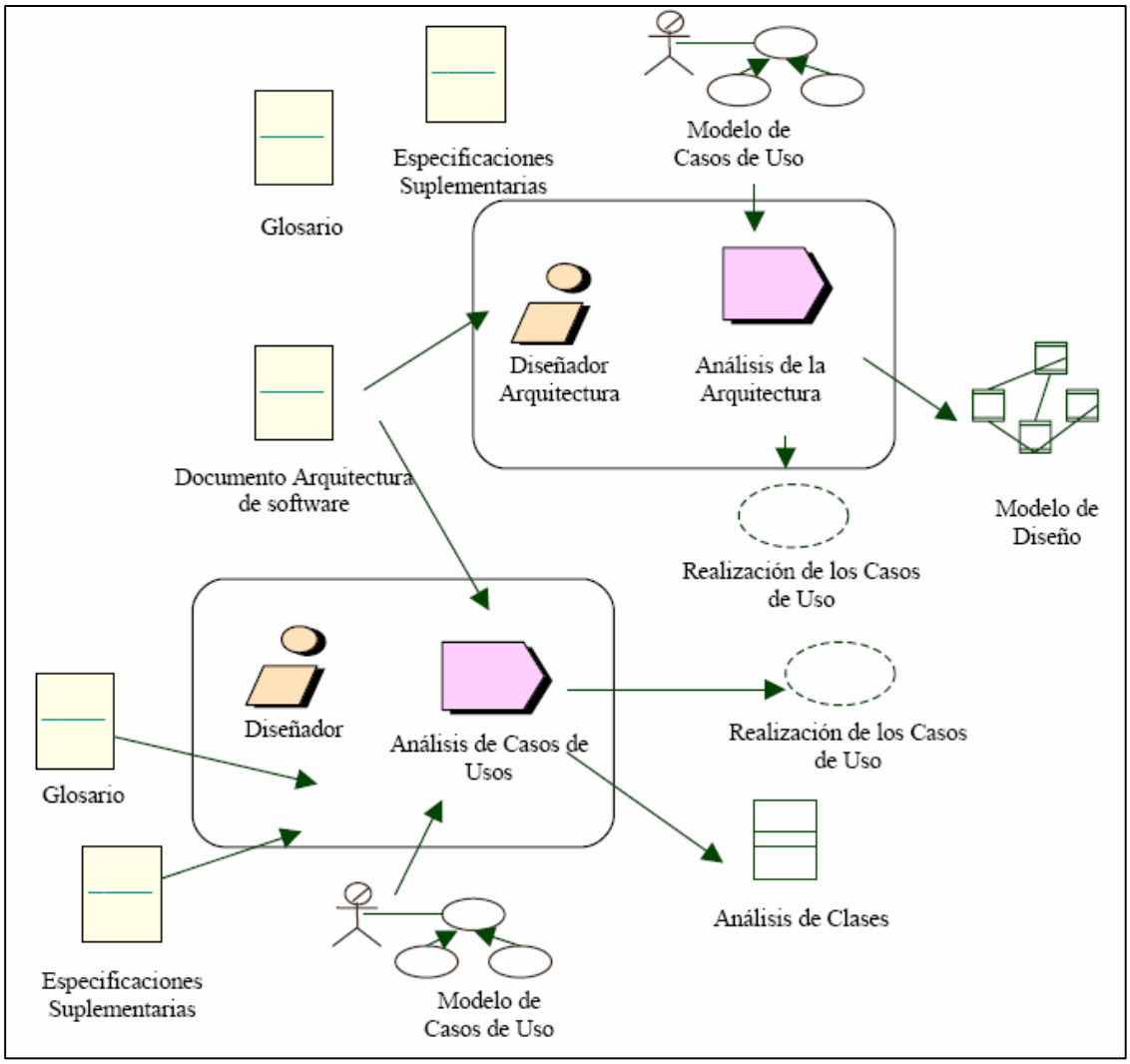

Figura 3: Workflow de Análisis y Diseño. Contreras y Laciar (2004)

Otra investigación realizada fue la del centro de formación virtual de Renfe - España, en el cual desean implementar una capa intermedia de middleware entre la plataforma de e-learning y los contenidos. Este modelo se basa en la arquitectura de SCORM y en la extensión del vocabulario que define el estándar. Dicha propuesta propone dotar a los contenidos y/o plataformas de un conjunto extensible de servicios, la cual se aloja en el cliente web, para que monitorice las comunicaciones entre contenidos y plataforma y aporte en cada caso las funcionalidades necesarias. Se pretende resolver problemas de compatibilidades o construir nuevas funciones no

estándar de una manera coherente, flexible y sistemática con el objetivo de minimizar el esfuerzo de desarrollo, potenciar la reutilización del código y abaratar los costos de producción, aportando las máximas facilidades para los autores, desarrolladores y los administradores del sistema. El middleware se intercala entre los RUNTIME API de SCORM y los contenidos. Se mantiene la comunicación estándar LMS-API, pero el middleware se puede comunicar con la lógica propietaria para realizar tareas específicas, como se muestra a continuación:

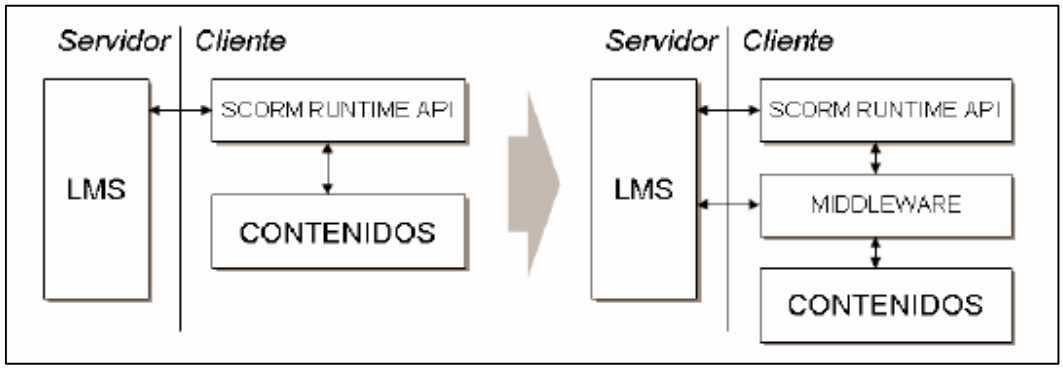

Figura 4: Implementación de la capa intermedia Middleware. Manclús (2004)

La implementación de esta capa intermedia permite extender los mecanismos de comunicación entre contenidos y plataforma, solucionar problemas de compatibilidad e incluir información y funcionalidades ampliadas. Según Manclús (2004), "El middleware es modular y actúa en distintos ámbitos, extendiendo el estándar, accediendo a los metadatos de los contenidos y ofreciendo soporte para el desarrollo de módulos de utilidades". Por lo cual, esta idea la han implementado en el Centro de Formación de Renfe – España en corto tiempo de desarrollo y ha generado la solución requerida.

De manera formal, Martínez (2003) en su estudio relacionado con el e-Learning, señala que un buen sistema debe facilitar:

- Transferencia automática de datos personales, de gestión, formativos, … entre la plataforma educativa y las aplicaciones de de gestión de recursos humanos de la institución;
- Soporte y gestión de cursos de varios orígenes;
- Gestión de datos completos de la actividad formativa, tales como itinerarios, actividades realizadas o información de evaluación;
- Importación y exportación de todos los datos de gestión y de seguimiento

A continuación se presentan varias organizaciones que han dedicados sus esfuerzos en la definición de estándares en e-learning para facilitar la comunicación entre plataformas de formación de distintos fabricantes, los cuales comenzaron a surgir en la década pasada y entre ellas destacan cuatros: AICC, IEEE, IMS y ADL.

1. AICC:

El AICC (Aviation Industry CBT Committee) es el primer organismo con reglas para el intercambio de cursos para la Enseñanza Asistida por Ordenador (EAO – Computer Aided Instruction), comenzando sus actividades en 1992. Su finalidad era la creación de cursos de formación para la industria de la aviación. Desarrollaron varios modelos para la formación virtual que han quedado reflejadas en varias guías denominadas AGR (AICC Guidelines and Recommendations – Guías y Recomendaciones de AICC).

El objetivo de esta organización es conseguir una formación eficiente, sostenible y a un bajo costo ajustado a resultados. Su principal aportación es la propuesta de CMI (Computer Managed Instruction – Instrucción Administrada por Computadoras), que proporciona un conjunto de reglas para el intercambio de contenidos entre plataformas de formación y para la gestión y el seguimiento de los resultados de aprendizaje. Se compone de los siguientes subcomités:

- CMI – Instrucción Administrada por Computadoras

- Comunicaciones
- DELS – Sistema de Bibliotecas Electrónicas Digital
- Laboratorio de Pruebas Independientes
- Dirección y Procesos
- Infraestructura de Adiestramiento
- Tecnología de Adiestramiento.

Las especificaciones del AICC cubren nueve áreas principales, que van desde los objetos de aprendizaje (learning objects - LO) hasta las plataformas de formación (Learning Management Systems - LMS). Cuando una compañía dice que cumple con las especificaciones AICC, significa que cumple con al menos una de estas guías y recomendaciones (AICC Guidelines and Recommendations, AGRs). Las especificaciones del AICC cubren las siguientes áreas:

- AGR 001: Publicaciones de AICC
- AGR 002: Estación de entrega de Cursos
- AGR 003: Audio Digital
- AGR 004: Sistemas Operativos
- AGR 005: Dispositivos Periféricos de CBT
- AGR 006: Instrucción Administrada por Computadoras
- AGR 007: Intercambio de Cursos
- AGR 008: Video Digital
- AGR 009: Estándar de Iconos e Interfaces de Usuarios
- AGR 010: Instrucción Administrada por Computadoras basada en Web

En estas guías resuelven dos de los problemas fundamentales:

- **La carga sin problemas en un LMS de cursos creados por terceros.** Este objetivo se consigue definiendo el curso como una entidad totalmente independiente de la plataforma, y creando un sistema de archivos que describen el curso para que pueda ser interpretado por cualquier plataforma.

- **La comunicación entre el LMS y el curso**, de tal modo que el curso pueda obtener información necesaria sobre el usuario, y después transmitir los resultados de las interacciones y evaluaciones realizadas por el mismo a la plataforma, de manera que esa información pueda ser almacenada en la LMS y pueda utilizarse para obtener estadísticas de los participantes.

Este segundo objetivo es logrado mediante la definición de un mecanismo de comunicación entre el curso y la plataforma, y un conjunto de datos mínimos que el curso debe transmitir a la plataforma y viceversa. La AICC describe dos mecanismos, uno más sencillo y extendido basado en el protocolo http, y otro mediante la aplicación de una API (Application Program Interface - Interfaz del Programa de Aplicación).

De todas las guías y recomendaciones de la AICC, la AGR 010 es la más seguida, ya que hace referencia a la interoperabilidad de las plataformas de formación, y además es la que utiliza el modelo SCORM para la distribución de los contenidos entre plataformas de diferentes fabricantes.

2. IEEE-LTSC

El IEEE Learning Technologies Standards Committee (LTSC) promueve la creación de una norma ISO. Mejora el trabajo de la AICC a través de la utilización de metadatos. Trabaja de forma coordinada con ISO/IEC JTC1 SC36: comité creado por ISO (International Standard Organization) y el IEC (International Electrotechnical Commission) para la normalización en el ámbito de la tecnología de la información para la formación, la educación y el aprendizaje.

El objetivo de este grupo es desarrollar los estándares técnicos, recomendaciones prácticas y guías para componentes de software y herramientas y métodos de diseño. Su aportación más significativa es LOM (Learning Objects Metadata), que define

elementos para describir los recursos de aprendizaje. LTSC tiene más de una docena de grupos de trabajo (working groups o WG) y grupos de estudio (study groups o SG) que desarrollan especificaciones para la industria del e-learning. Los siguientes grupos de trabajo son parte de las actividades generales de la IEEE LTSC:

- IEEE 1484.1 Modelo de Referencias y Arquitecturas
- IEEE 1484.3 Glosario de Términos.

Los siguientes grupos de trabajo son parte de las actividades relacionadas con los datos y los metadata:

- IEEE 1484.12 Metadatos de Objetos de Aprendizajes
- IEEE 1484.14 Semántica e Intercambio Obligatorio
- IEEE 1484.15 Protocolo de Intercambio de Datos

Los siguientes grupos de trabajo son parte de las actividades relacionadas con los LMS y las aplicaciones:

- IEEE 1484.11 Instrucción administrada por Computadoras
- IEEE 1484.18 Perfiles de Multimedia y Plataformas
- IEEE 1484.20 Definición de las Competencias

3. IMS Global Learning Consortium, Inc.

El IMS Global Learning Consortium está formado por más de 250 miembros que provienen de instituciones educativas y empresas públicas y privadas. Tienen como objetivos ayudar a definir las especificaciones técnicas para la interoperabilidad y fomentar la implementación de las especificaciones en productos y servicios reales.

Este consorcio pone en práctica las recomendaciones de AICC y de IEEE, utilizando XML para describir aspectos clave de cursos, lecciones, asignaturas, alumnos y grupos, de tal modo que cualquier LMS pueda leer su archivo de

configuración IMSMANIFEST.XML, para interpretar y cargar el curso dentro de la plataforma. IMS elabora las siguientes especificaciones:

a. Especificaciones usadas para describir, descubrir e intercambiar contenidos:

- **Metadata de Objetos de Aprendizajes (LOM - Learning Object Metadata):** Esta especificación entrega una guía sobre cómo los contenidos deben ser identificados y/o sobre cómo se debe organizar la información de los alumnos de manera de que se puedan intercambiar entre los distintos servicios involucrados en una plataforma. La especificación para metadata de la IMS consta de tres documentos: el Modelo de Información de Metadata para recursos de aprendizaje de IMS (IMS Learning Resource Metadata Information Model), Las Especificaciones obligatorias de XML para recursos de aprendizajes de IMS (IMS Learning Resource XML Binding Specifications), y la Guía de implementación y buenas prácticas de metadata de recursos de aprendizaje de IMS (IMS Learning Resource Metadata Best Practices and Implementation Guide).

- **Empaquetamiento de Contenidos (Content Packaging):** Esta especificación provee la funcionalidad para describir y empaquetar material de aprendizaje, ya sea un curso individual o una colección de cursos, en paquetes portables e interoperables.

- **Interoperabilidad de preguntas y pruebas (QTI - Question and Test Interoperability):** El IMS QTI propone una estructura de datos XML para codificar preguntas y test en línea. El objetivo de esta especificación es permitir el intercambio de estos tests y datos de evaluación entre distintas plataformas.

- **Repositorios Digitales (Digital Repositories):** El IMS está en el proceso de creación de especificaciones y recomendaciones para la interoperación entre repositorios digitales.

b. Especificaciones para la interacción con el contenido y el seguimiento:

- **Secuencia simple (Simple Sequencing):** Esta especificación define normas que describen el flujo de instrucciones a través del contenido según el resultado de las interacciones de un alumno con el contenido.

- **Definición de Competencias (Competency Definitions):** El IMS (al igual que la IEEE) están en el proceso de crear una manera estandarizada de describir, referenciar e intercambiar definiciones de competencias. En esta especificación, el término competencia es usado en un sentido muy general, que incluye habilidades, conocimiento, tareas, y resultados de aprendizaje.

- **Diseño del Aprendizaje (Learning Design):** Este grupo de trabajo del IMS investiga sobre las maneras de describir y codificar las metodologías de aprendizaje incorporadas en una solución e-learning.

- **Accesibilidad (Accesibility):** Este grupo de trabajo promueve el contenido de aprendizajes accesibles a través de recomendaciones, guías, y modificaciones a otras especificaciones. Para la IMS el término tecnología accesible se refiere a la tecnología que puede ser usada sin tener acceso pleno a una o más canales de entrada y salida, usualmente visuales y auditivas.

c. Especificaciones para la interoperabilidad de las aplicaciones:

- **Empaquetamiento de la Información del Alumno (LIP - Learner Information Packaging):** Esta especificación define estructuras XML para

el intercambio de información de los alumnos entre sistemas de gestión de aprendizaje, sistemas de recursos humanos, sistemas de gestión del conocimiento, y cualquier otro sistema utilizado en el proceso de aprendizaje.

Basado en las especificaciones planteadas por la AICC, IEEE-LTSC y IMS, surge como iniciativa de ADL el modelo SCORM, el cual es parte fundamental de esta investigación, ya que reúne lo mejor de cada una de las normativas anteriores, para proporcionar un marco de trabajo y una referencia de implementación detallada, que permita intercambiar en forma dinámica contenidos de aprendizajes y a los sistemas hablar con otros sistemas, logrando así que las plataformas de formación posean interoperabilidad, reusabilidad de contenidos y adaptabilidad a los cambios. En la siguiente sección de este capítulo se mostrará de forma más detallada el modelo estándar SCORM.

Bases Teóricas

Para establecer las bases teóricas de esta investigación se muestra algunos conceptos y definiciones que apoyan el desarrollo de ésta. Se comienza por definir lo que actualmente se considera como Educación a Distancia, su evolución a partir de la popularización de Internet, los requerimientos de dichos cursos, los actores involucrados y sus interrelaciones. Posterior a esta definición se muestra la conceptualización de arquitectura de software, la cual no es más que una visión global de cualquier sistema de software, describiéndola a través de sus componentes y las relaciones entre ellos.

Luego se incorpora el concepto de plataformas de formación y de objetos de aprendizaje, para apoyar y comprender mejor lo que significa SCORM y como funciona. SCORM es un modelo que permite la estandarización de las plataformas

de formación para resolver el problema de intercambio de contenidos entre distintas plataformas de diferentes fabricantes.

Después de describir a SCORM, se presentan los conceptos de patrón, lenguajes de patrones y su clasificación, ya que se hará uso de ellos para cumplir con el objetivo principal de esta investigación. Para representar esta propuesta se utiliza como herramienta principal el lenguaje unificado de modelado – UML.

La Educación a Distancia y las Plataformas de Formación (LMS – Learning Management Systems)

La educación a distancia se ha entendido de diferentes formas. En su significado más simple tiene que ver con la idea de un alumno y un profesor, separados por el tiempo y el espacio, que utilizan ciertos medios para comunicarse y aprender. Estos medios han ido evolucionando a lo largo del tiempo. Poco a poco se han ido añadiendo medios cada vez más sofisticados, como: la radio, la televisión, el vídeocassete, el cassette, y más recientemente todos los medios derivados de Internet. La introducción de Internet ha venido a acelerar la exploración de nuevas formas de aprendizaje a distancia.

Eastmond (1995) plantea que la conectividad creada por las herramientas de comunicación en línea, ofrecen nuevas posibilidades de aprendizaje para la colaboración y el trabajo en grupo, en un formato completamente nuevo. Afirma que “el ambiente online crea oportunidades para compartir ideas y recibir retroacción de otras personas en un ambiente sin amenazas”.

García (2000), ha revisado diferentes definiciones y conceptos en este terreno. A partir de esta amplia revisión encuentra que las características de la educación a distancia son:

- Separación entre el profesor y el alumno: ambos sujetos no comparten un mismo espacio físico.
- Utilización de medios técnicos para facilitar a los alumnos el acceso a los conocimientos y para las comunicaciones.
- Organización de apoyo a los alumnos mediante tutorías.
- Los alumnos pueden aprender de manera flexible e independiente, lo que no necesariamente significa aprender en solitario.
- Comunicación bidireccional entre los profesores y los alumnos y de los alumnos entre sí.
- Enfoque tecnológico en las decisiones referidas a la planificación, el desarrollo y evaluación de las acciones de educación a distancia.
- Comunicación masiva e ilimitada con alumnos en contextos geográficamente dispersos.

Partiendo de estas características, García (2000), define la educación a distancia como "un sistema tecnológico de comunicación bidireccional (multidireccional), que puede ser masivo, basado en la acción sistemática y conjunta de recursos didácticos y el apoyo de una organización y tutoría, que separados físicamente de los estudiantes, propician en estos un aprendizaje independiente (cooperativo)".

La incorporación de las nuevas tecnologías de la información y la comunicación con fines educativos y formativos ha dado lugar a lo que se denomina como teleformación. Ésta se refiere a cualquier oferta de formación a distancia, que reúna las condiciones expuestas anteriormente, pero que incorpore algún medio tecnológico para facilitar algunas de las funciones de aprendizaje como: leer, compartir, observar, simular, discutir, entre otros.

Actualmente la Educación a Distancia o teleformación es la forma de instrucción a través de Internet. Se pueden encontrar diferentes denominaciones en inglés como Web-based training, Web-based instruction, online learning o más recientemente

e-learning. En cualquiera de sus acepciones, se trata de una modalidad de formación que permite utilizar las potencialidades de la red para acercar la formación a sus posibles usuarios. Internet se está convirtiendo no sólo en una vía de formación sino en un auténtico mercado para la formación. McCormack y Jones (1998) plantean que:

> …la Formación con Internet es un ambiente creado en la Web en el que los estudiantes y educadores pueden llevar a cabo tareas de aprendizaje. No es sólo un mecanismo para distribuir la información a los estudiantes; también supone tareas relacionadas con la comunicación, la evaluación de los alumnos y la gestión de la clase

Por otra parte, Palloff y Pratt (2001), definen la educación online como:

> …un enfoque de enseñanza y aprendizaje que utiliza Internet para comunicar y colaborar en un contexto educativo. Incluye la tecnología que sirve de suplemento a la formación tradicional mediante los componentes basados en Internet y los ambientes de aprendizaje donde el proceso educativo se experimenta online

Es por ello, que cada uno de los recursos ofrecidos por Internet aportan herramientas útiles que facilitan la educación a distancia, la teleformación o el e-learning, como se ha observado existen diferentes términos y todos llevan a la conclusión de que son una forma de facilitar conocimientos a personas que, bien sea por su situación geográfica (alumnos en zonas diferentes), bien sea por sus condiciones de trabajo (personas con poco tiempo para atender una enseñanza sujeta a horarios), bien sea por propia opción personal, eligen una formación más acorde con sus posibilidades.

Para llevar a cabo un curso de e-learning se deben tomar en cuenta una serie de requerimientos que son generales para cualquier curso. Se debe presentar una integración tecnológica que simulan la estructura, servicios, empleo y acceso a un centro de aprendizaje virtual. Según Guerra y Estivales (2002), mencionan que el

e-learning se puede configurar desde cuatros visiones o espacios como lo son: una Visión Institucional o corporativa, en la que coexisten un cuerpo docente, políticas o reglas de administración de cursos, imagen corporativa, e individualización institucional (objetivos, misión, visión, modelo y estrategia educativos). Otra sería la Visión Docente, que es el espacio de trabajo para profesores, tutores y ayudantes, en donde se facilitan la elaboración, planificación e implementación de cursos. Una tercera visión es la Visión del Alumno, un espacio organizado de acuerdo a actividades y frecuencia de uso. Este espacio colaborativo mantiene absoluta confidencialidad y privacidad de toda la información de cada alumno, como identificación personal, lugar de acceso o calificaciones. Por último, se presenta la Visión de Administrador, en donde se facilitan las actividades asociadas al mantenimiento del sistema y administración de cursos impartidos, el conjunto de todos estos elementos es lo que se le llama plataforma de formación.

Alguna de las principales características que debe tener un sistema de e-learning o plataforma de formación deben ser: interfaz de acceso y administración Web, soporte multimedial, servicios Web-mail y Web-chat, plantillas de completación y edición de contenidos, herramientas de seguimiento y control de progreso de estudio, herramientas de evaluación y autoevaluación, generación de informes estadísticos de progreso de estudio, planificación dinámica de actividades, administración, seguimiento y control de planificaciones y utilización de recursos de las NTIC complementarios, gestión y administración de participantes, administración de contenidos, entre otras. Guerra y Estivales (2002).

Los principales actores o participantes de este sistema se pueden categorizar o clasificar de la siguiente forma: docente o facilitador, estudiante o participante y administrador.

Para concluir se puede agregar, que todo sistema de e-learning presenta una serie de elementos que se resumen en la Figura 5.

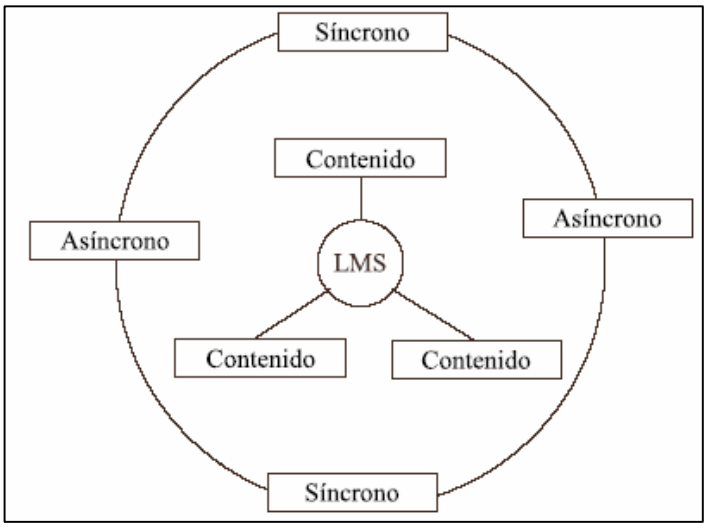

Figura 5: Componentes del e-learning. Foix y Zavando (2002) citado por Rebollo (2004)

En la Figura 5, se puede apreciar que en todo sistema de e-learning, se identifican tres elementos. En primer lugar, una plataforma de formación o LMS, la cual da soporte a todas las actividades formativas y de gestión que tienen lugar durante los aprendizajes. En segundo lugar, los contenidos para el estudio o cursos. Y por último, las herramientas de comunicación, tanto síncrona como asíncrona, que permiten el contacto entre los participantes del curso. Rebollo (2004)

Arquitectura de Software

La construcción de un sistema de e-learning esta soportado por una arquitectura de software (AS), por lo que existen diversas definiciones, pero las más conocidas y representativas de lo que significa la AS se presentan a continuación. Se puede comenzar con la definición de Clements (1996) donde describe que la AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. Esta definición permite entender a la AS como la descripción

de un sistema macro, basándose en los componentes o elementos que la integran y como estos se relacionan e interactúan entre sí, es una visión general del sistema con un alto nivel de abstracción que le permite al ingeniero visualizar a la solución general al problema.

Otra de las definiciones de la AS, es la realizada por Bass (1998) en la cual este autor se refiere a ella como la estructura a grandes rasgos del sistema, estructura consistente en componentes y relaciones entre ellos. Esta estructura se vincula con el diseño, pues la AS es después de todo una forma de diseño de software que se manifiesta tempranamente en el proceso de creación de un sistema; pero este diseño ocurre a un nivel más abstracto que el de los algoritmos y las estructuras de datos. En el que muchos consideran un ensayo seminal de la disciplina, Garlan y Shaw (1994), sugieren que dicha estructura incluye una organización a grandes rasgos y control global; protocolos para la comunicación, la sincronización y el acceso a datos; la asignación de funcionalidad a elementos del diseño; la distribución física; la composición de los elementos de diseño; escalabilidad y rendimiento; y la selección entre alternativas de diseño, es por ello que representa un diseño ajustable o cambiante hasta que proporcione la mejor solución del problema, y una visión general del sistema antes de que este sea desarrollado.

Garlan (2000), establece en una definición tal vez bastante amplia que la AS constituye un puente entre el requerimiento y el código, ocupando el lugar que en los gráficos antiguos se reservaba para el diseño. La definición oficial de AS proporcionada por IEEE Std 1471-2000, adoptada también por Microsoft, dice así: La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

En conclusión, se puede observar que todas las definiciones de la AS radican en

que es una estructura del sistema, representada o descrita a través de sus componentes y como estos se relacionan entre sí.

SCORM - Shareable Content Object Reference Model (Modelo de Referencia para Objetos de Contenidos Intercambiables)

ADL (Advanced Distributed Learning), es un programa del Departamento de Defensa de los Estados Unidos y de la Oficina de Ciencia y Tecnología de la Casa Blanca, formada en 1997 para desarrollar 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 recogió lo mejor de las iniciativas mencionadas anteriormente en la sección de antecedentes del presente capítulo, como lo son el sistema de descripción de cursos en XML de la IMS, y el mecanismo de intercambio de información mediante una API de la AICC; las combinó y surgió lo que es ahora, el Modelo de Referencia para Objetos de Contenidos Intercambiables (SCORM - Shareable Content Object Reference Model). Este modelo proporciona un marco de trabajo y una referencia de implementación detallada que permite a los contenidos y a los sistemas usar SCORM para comunicarse con otros sistemas, logrando así interoperabilidad, reusabilidad y adaptabilidad.

SCORM divide el mundo de la tecnología del aprendizaje en dos componentes fundamentales: la plataforma de formación (LMS - Learning Management System) y los objetos de contenido intercambiables (SCO - Sharable Content Objects). Los SCO constituyen su visión particular de los objetos de aprendizaje reusables y estandarizados. La plataforma es cualquier elemento que almacena información sobre los estudiantes, que es capaz de lanzar o publicar objetos, de comunicarse con los SCO y que puede interpretar las instrucciones que le indican la secuencia correcta entre un conjunto de SCO. Otros componentes en el modelo SCORM son

herramientas que crean los SCOs y los ensamblan en unidades de aprendizaje más grandes (como por ejemplo un curso). Su plataforma puede ser representada por el modelo mostrado en la Figura 6.

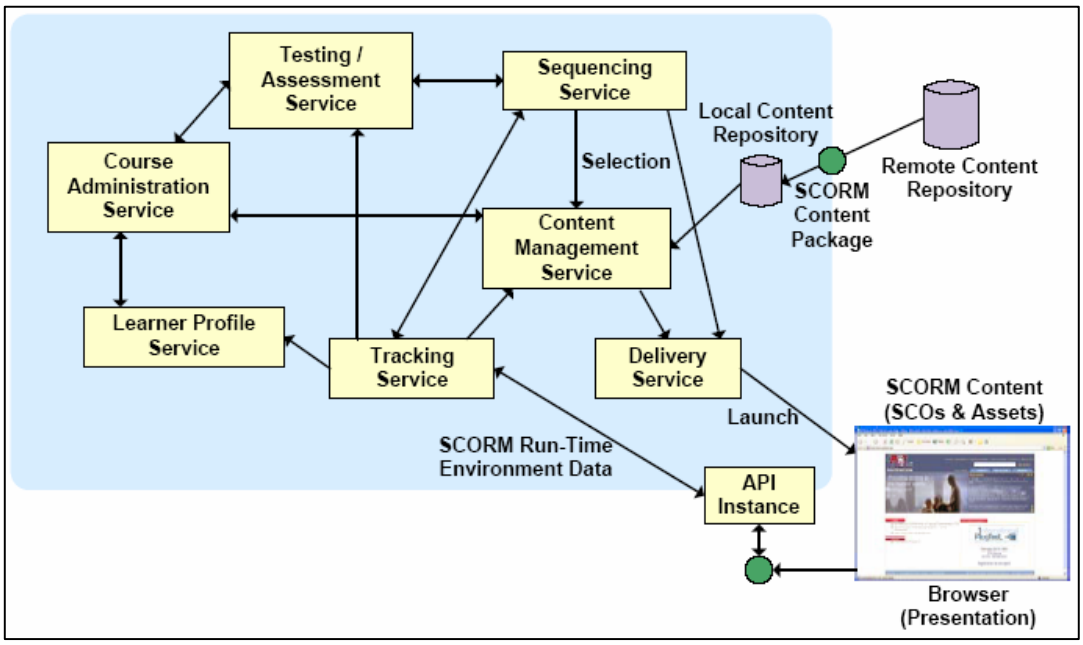

Figura 6: Modelo de Plataforma de Formación - SCORM. ADL (2004)

El modelo mostrado en la Figura 6 de una plataforma de formación basada en SCORM, contiene las siguientes ventajas:

- **La disponibilidad de un Sistema de Gestión de Aprendizaje** o LMS basado en Web para colocar diferentes contenidos que se han desarrollado por varios autores usando herramientas de diversos vendedores.

- **La disponibilidad de diversos LMS producidos por diferentes vendedores** para colocar un mismo contenido.

- **La disponibilidad de múltiples productos o entornos LMS** basados en Web para acceder a un repositorio común de contenidos.

Las especificaciones de SCORM están organizadas como libros separados. La mayoría de estas especificaciones son tomadas desde otras organizaciones. Estos libros técnicos se agrupan bajo dos tópicos principales: el Modelo de Agregación de Contenido (Content Aggregation Model) y el Entorno de Ejecución (RunTime Environment). Como lo muestra la Figura 7, SCORM ha sido dividido en tres libros técnicos que reúnen los diversos aspectos de la especificación, más un cuarto libro que ofrece una visión general sobre el estándar.

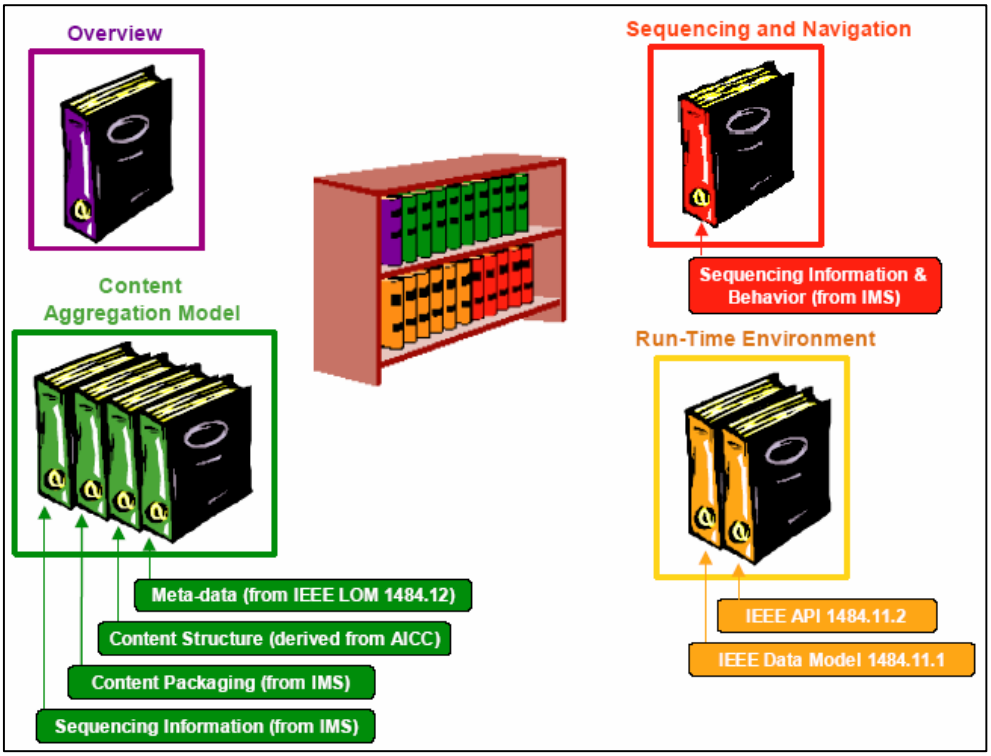

Figura 7: Descripción detallada de SCORM. ADL (2004)

En el Libro 1, **Vista Global de SCORM (SCORM Overview):** contiene una descripción general de la iniciativa de ADL, un análisis de SCORM, y un resumen de las especificaciones técnicas contenidas, así como los elementos que la componen y su terminología.

En el Libro 2, **Modelo de Agregación de Contenidos de SCORM (CAM - SCORM Content Aggregation Model):** contiene una guía para identificar y agregar recursos dentro de un contenido de aprendizaje estructurado. Este libro presenta una nomenclatura para el contenido de aprendizaje, describe el SCORM Content Packaging (empaquetamiento de contenidos) y hace referencia al IMS Learning Resource Metadata Information Model, el cual está basado en el IEEE LTSC Learning Object Metadata (LOM) Specification, que fue el resultado de un esfuerzo en conjunto entre el IMS Global Learning Consortium y la Alliance of Remote Instructional Authoring and Distribution Networks for Europe (ARIADNE).

El objetivo del modelo de agregación de contenidos de SCORM es proporcionar un medio común para elaborar ó construir contenidos educativos desde diversas fuentes compartibles y reutilizables. Especifica cómo un contenido educativo puede ser identificado, descrito y agregado dentro de un curso o una parte de un curso, y cómo puede ser compartido por diversos LMS o por diversos repositorios.

Un paquete de contenidos agrupa una serie de objetos cuya organización se describe en un manifiesto. Un paquete puede representar un curso, una lección, un módulo o una colección de objetos que no se asimila con ningún nivel concreto. El manifiesto es un archivo de XML de nombre “imsmanifest.xml”. Además, el paquete puede agregar información para proporcionar instrucciones a la plataforma sobre cómo manipular su contenido. Algunos de estos elementos se emplean luego en el modelo SCORM RTE.

En el Libro 3**, Entorno de Ejecución de SCORM (RTE - SCORM RunTime Environment)**: se incluye una guía para la ejecución de contenidos, comunicación, seguimiento, transferencia de datos y gestión de errores en un ambiente basado en Web. Este libro es derivado del CMI001 Guidelines for Interoperability ó Guía de Interoperabilidad de la AICC.

Este es el entorno operativo o de ejecución de SCORM y su objetivo es proporcionar un medio para la interoperabilidad entre los SCO, y las LMS. Un requerimiento de SCORM es que el contenido educativo sea interoperativo a través de múltiples LMS, sin tener en cuenta las herramientas que se usen para crear o usar los contenidos.

Los tres componentes del entorno de ejecución de SCORM son:

- **Lanzador o Publicador**: Es el elemento que define el método común para que las plataformas puedan lanzar o publicar un SCO basado en Web. Este componente define los procedimientos y las responsabilidades para el establecimiento de la comunicación entre el contenido a mostrar y la LMS. El protocolo de comunicación está estandarizado a través del uso común de la API.

- **API (Application Program Interface)**. Proporciona un conjunto de funciones predefinidas para que la plataforma pueda comunicarse y controlar a los SCO que lanza o publica. El objeto queda enlazado a la plataforma cuando se lanza o se publica, este enlace se rompe cuando ya no se necesita el objeto. Entre sus funciones también se encuentra, permitir que los objetos lean y escriban información en la plataforma y comprobar los errores que se produzcan durante el proceso.

- **Modelo de Datos**: Está formado por una lista estandarizada de elementos, es decir, un vocabulario que se emplea para intercambiar información. Por

ejemplo, la nota obtenida por un participante al realizar un test contenido en un SCO.

Y por último, el libro 4, **Secuencia y Navegabilidad de la Información (SN - Sequence Information & Navigation)**: describe las reglas que marcan la secuencia de navegación entre distintos objetos de aprendizaje.

Patrón

En secciones anteriores se ha mencionado algunas definiciones de patrones, y como este término fue tomado del campo de la arquitectura y el urbanismo y adaptado a la ingeniería de software por el autor Erich Gamma en su libro, donde muestra como las ideas originales de Alexander (1977) se incluyen al diseño orientado a objetos, definiendo un patrón de diseño como “una descripción de clases y objetos que se comunican entre sí dispuestos para resolver un problema de diseño general en un contexto particular” Gamma (1995).

Es por esto que, un patrón es una herramienta que facilita el desarrollo de software, ya que es una regla que consta de tres partes básicas, un problema, una solución y el contexto donde se desarrolló; y si el problema es frecuente, proporciona la certeza de que la solución propuesta es ampliamente buena y es garantía para resolver ese problema.

Los patrones facilitan la reutilización de diseños y arquitecturas de software que han tenido éxito, además capturan la experiencia y la hacen accesible a los no expertos, se pueden agrupar y el conjunto de sus nombres forma un vocabulario que ayuda a que los desarrolladores se comuniquen mejor. Ayudan a los desarrolladores a comprender un sistema más rápidamente cuando está documentado con los patrones

que usa. Facilitan la reestructuración de un sistema tanto si fue o no concebido con patrones en mente. Pavón (2004)

Lenguaje de Patrones

Los patrones se pueden agrupar formando colecciones que constituyen un vocabulario para comprender y comunicar ideas. Cuando estas colecciones se conforman con habilidad, para formar un todo cohesionado que revele las estructuras y las relaciones de sus componentes ó para cumplir un objetivo compartido, entonces se esta hablando de lo que Alexander (1977) en el urbanismo bautiza como lenguaje de patrones. Un lenguaje de patrones define una colección de patrones, reglas y pautas para combinarlos con un estilo que se podría denominar arquitectónico, pues define una forma de construir estructuras a todos los niveles de escala y en todos los grados de diversidad. En realidad, un lenguaje de patrones se compone de un léxico de patrones y una gramática que establece cómo unirlos para formar estructuras sintácticas. Idealmente, los buenos lenguajes de patrones son generativos, es decir, capaces de generar todas las posibles frases a partir de un vocabulario de patrones rico y expresivo. En palabras de Alexander (1977): “No solamente nos dicen qué reglas hay que seguir en las soluciones sino que nos muestran cómo construir soluciones que satisfagan las reglas”.

Clasificación de los Patrones

Buschmann (1996), define una taxonomía para los patrones, el cual los reúne en tres grandes grupos, estos en cada grupo varían respecto a su nivel de detalle y abstracción, tal como se muestra a continuación:

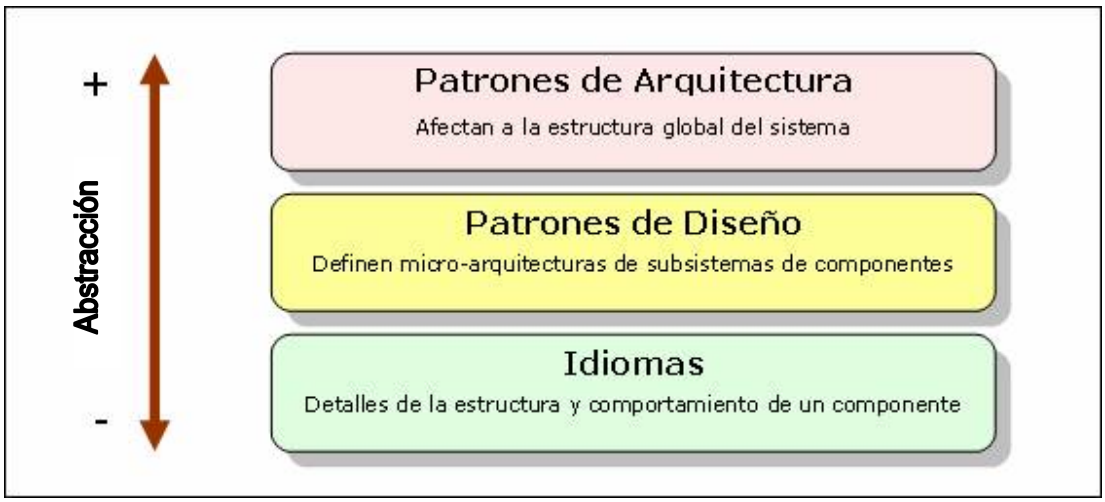

Figura 8: Patrones según el nivel de Abstracción, adaptado de POSA.
Tomado http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_2864.asp

- **Patrones de Arquitectura ó Arquitecturales**

Estos expresan un paradigma fundamental para estructurar un sistema software y proporcionan un conjunto de subsistemas predefinidos, que especifica sus responsabilidades, e incluye reglas y guías para organizar las relaciones entre ellos. Algunos ejemplos: Tuberías y filtros, Cliente/Servidor, Maestro-Esclavo, Control centralizado y distribuido

- **Patrones de diseño**

Los patrones de diseño, están compuestos de varias unidades arquitecturales más pequeñas y describen el esquema básico para estructurar subsistemas y componentes resolviendo un diseño general dentro de un contexto particular. Como por ejemplo: Proxies, Factorías, Adaptadores, Composición, Broker.

- **Patrones elementales (idiomas)**

Un patrón de idiomas, es un patrón de bajo nivel, específico a un lenguaje de programación. Un patrón de idiomas describe cómo llevar a cabo aspectos particulares de componentes o las relaciones entre ellos, usando las características de lenguaje de programación utilizado. Por ejemplo: Modularidad, Interfaces mínimas, Encapsulación, Objetos, Acciones y Eventos, Concurrencia.

Ejemplo de Patrones

A continuación se muestra un ejemplo de un caso práctico, tomado de los autores Yacoub y Ammar (2004) donde se describe un sistema de distribución de contenidos a través de la Web, basado en patrones y a través de UML.

Entre los requerimientos funcionales propuestos para esta aplicación, se requiere un subsistema de distribución de tareas a través de máquinas de trabajo. Por ejemplo, entre las tareas se encuentra que la aplicación debe convertir documento de formato TIFF a formato PDF, otro de los requerimientos es que las máquinas de trabajo deben operar de manera paralela y que cada procesamiento es una tarea de conversión.

Entre otros requerimientos, se encuentran que:

- La aplicación proporcione una interface que acepte las descripciones de la tarea del servidor y reciba los mensajes de notificación de las máquinas de trabajo con respecto a la terminación de una tarea.

- Proporcionar los mecanismos de comunicación necesarios para establecer la comunicación con todas las máquinas de trabajo.

- Debe ser extensible para soportar varias tecnologías de comunicación y protocolos sin afectar la implementación del resto de la aplicación.

- Debe comunicarse con máquinas de trabajo heterogéneas.

- Proveer un mecanismo de distribución para las tareas entre las máquinas de trabajo.

Cada uno de estos requerimientos, les ayuda a los autores Yacoub y Ammar (2004) a revisar la bibliografía existente de patrones y a seleccionar los más adecuados para la solución de su problema, ya que son usados en el diseño del subsistema de distribución. Para mostrar los patrones seleccionados a continuación se muestra el siguiente diagrama.

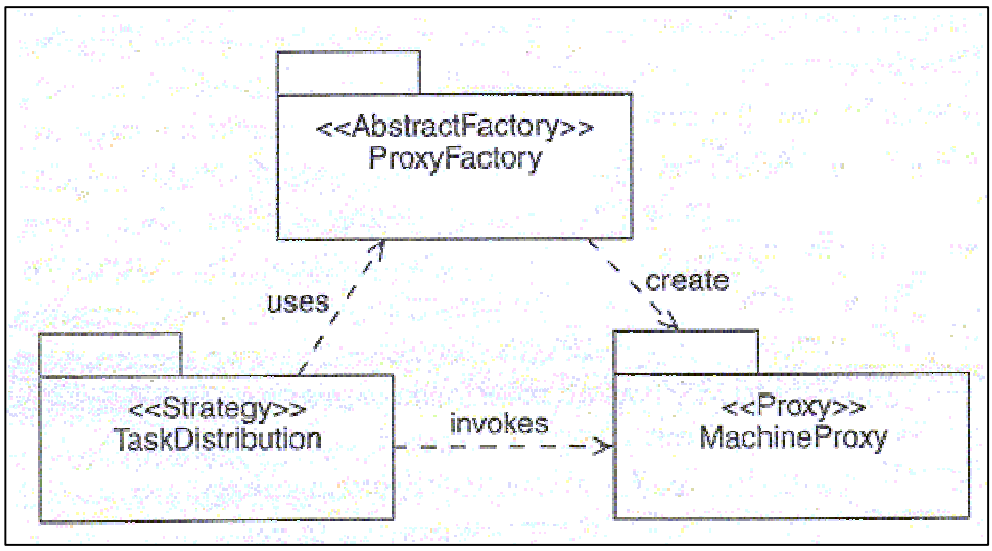

Figura 9: Diagrama de Patrones para el Subsistema de Distribución.
Tomado: Yacoub y Ammar (2004)

En la Figura 9, se muestra como los autores seleccionaron tres patrones de diseño para crear las instancias de los patrones seleccionados e identificar las

relaciones entre esas instancias. En la etapa de análisis seleccionaron los patrones de diseño: Proxy, AbstractFactory y Strategy. Crearon una instancia MachineProxy de tipo patrón Proxy para implementar el sustituto de las máquinas remotas. Crearon una instancia TaskDistribution de tipo patrón Strategy, para proveer la encapsulación necesaria del policía usado para seleccionar la máquina, a la cual se le va asignar la tarea. Y por último, también crearon una instancia ProxyFactory de tipo patrón AbstractFactory para administrar la creación de objetos Proxy para varias máquinas de trabajo.

Luego que se obtiene un primer nivel del diagrama orientado a patrones, se procede a establecer un siguiente nivel, donde se muestra un diagrama inicial de clases para el subsistema de distribución, el cual es refinado a medida que se van elaborando las subsiguientes etapas de la metodología.

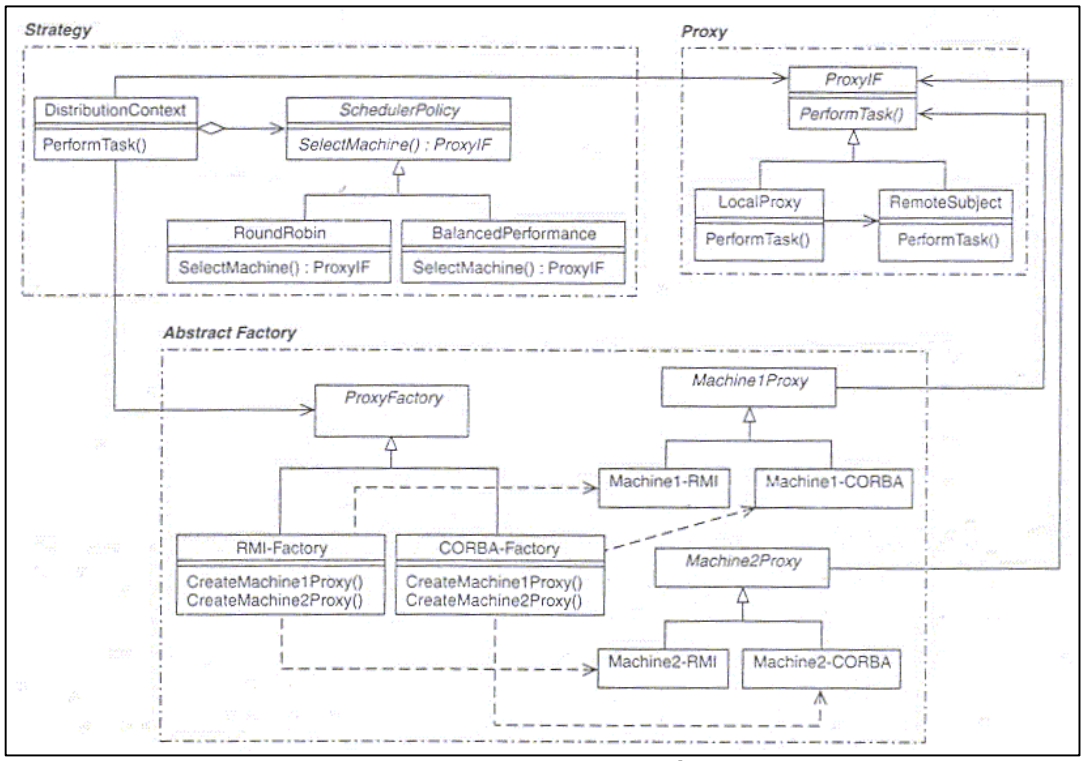

Figura 10: Diagrama de clases del Subsistema de Distribución. Tomado: Yacoub y Ammar 2004)

UML (Unified Modeling Language – Lenguaje Unificado de Modelado)

Así como, en la construcción de un edificio se realizan planos previo a su realización, en Ingeniería de Software se deben realizar diseños o modelos previos a la codificación de un sistema, para facilitar la comunicación y el aporte de conocimientos de cada uno de los integrantes de un equipo multidisciplinario, que participa en la realización de una aplicación de software. Estos integrantes pueden ser: los analistas, diseñadores, especialistas de área y desde luego los programadores.

Entre unos de los diseños previos a la realización de software se encuentran las arquitecturas de software. Para que una arquitectura de software se transforme en una herramienta útil dentro del desarrollo y mantenimiento de los sistemas de software, es necesario que cuente con una manera precisa de representarla. UML permite que se represente de manera semi-formal la estructura general del sistema, con la ventaja de que este mismo lenguaje puede ser usado en todas las etapas de desarrollo del sistema y su representación gráfica puede ser utilizada para comunicarse con los usuarios.

UML es un lenguaje gráfico de modelamiento que usa conceptos de orientación a objetos. Este lenguaje tiene una sintaxis y una semántica bien definidas, sirviendo además para todas las etapas de desarrollo. Booch y otros (1999)

En UML se utilizan para el modelamiento de un sistema diferentes elementos y relaciones, que tienen una semántica y sintaxis definidas. Estos elementos se agrupan en diagramas preestablecidos que corresponden a diferentes proyecciones del sistema.

Los elementos básicos de UML, aquellos que representan principalmente las partes estáticas del sistema, son:

- Clases
- Casos de uso

- Componentes
- Nodos
- Paquetes

Las relaciones que se utilizan para establecer conexiones entre los elementos son:

- Dependencia
- Asociación
- Generalización
- Realización

Cada uno de estos elementos y relaciones tiene una representación gráfica y puede complementar su información utilizando lo que se conoce como especificación. La especificación de un elemento o relación generalmente no es visible en la representación gráfica, o sólo lo es parcialmente, y corresponde a los datos o propiedades adicionales que completan o detallan la semántica del elemento o relación, y por lo tanto del sistema en general.

Los elementos y relaciones se agrupan en diagramas que representan diferentes aspectos del sistema. Los diagramas de UML son:

- **Diagrama de clases:** Presenta las clases, junto con sus atributos, operaciones, interfaces y relaciones. También presenta el agrupamiento de clases en paquetes y las relaciones entre ellos.

- **Diagrama de objetos:** Muestra instancias de clases (objetos) con valores en sus atributos y relaciones.

- **Diagrama de casos de uso:** Los escenarios de uso del sistema, incluyendo los roles de los usuarios.

- **Diagramas de interacción:** Comprende los diagramas de secuencia y de colaboración. Presenta objetos y relaciones entre ellos desde el punto de vista dinámico.

- **Diagrama de estado:** Representa los posibles estados, eventos y transiciones entre las clases u objetos.

- **Diagrama de componentes:** Organización y dependencia entre componentes físicos.

- **Diagrama físico (deployment):** La distribución y comunicación de los componentes en los dispositivos de hardware.

CAPITULO III

MARCO METODOLÓGICO

En este capítulo se describe las técnicas utilizadas para llevar a cabo la investigación, apoyándose en las bases teóricas definidas y descritas en el capítulo anterior relacionadas con el objeto de estudio.

Según Hernández y otros (1999), “La investigación es la herramienta para conocer lo que nos rodea y su carácter es universal”, lo cual nos permite profundizar sobre un determinado tema y llegar a nuestra propias conclusiones, las cuales pueden servir de bases para otros investigadores en el mismo tema y así con el pasar del tiempo, llegar a grandes avances y descubrimientos para mejorar la calidad de vida de los seres humanos.

Para realizar una investigación se necesita una serie de herramientas o técnicas que permitan organizar y sistematizar los conocimientos generados, es por ello que existen diferentes tipos de investigación y existe una gran variedad de autores que muestran una clasificación de ellas y una forma ordenada de hacerla. “Este proceso está compuesto por una serie de etapas, las cuales se derivan unas de otras” Hernández y otros (1999), es por ello que para llevar a cabo una investigación, no se debe omitir ninguna etapa y mucho menos alterar su orden. Toda investigación debe seguir un método para garantizar su confiabilidad, validez y objetividad. La presente investigación se realizó utilizando la siguiente metodología.

Tipo de Investigación

Considerando las características del problema planteado, el objeto de estudio en esta investigación y con los objetivos antes delimitados, esta investigación se inscribe dentro de la modalidad de Investigación monográfica documental según el Manual de Trabajo de Grado de la Especialización y de Tesis de Maestría y Doctorales de la Universidad Centroccidental Lisandro Alvarado, concebido como:

> Se entenderá por investigación monográfica documental el estudio descriptivo o diagnóstico de una situación inherente a la especialidad que lleva a la descripción o evaluación de los elementos que configuran el ámbito del problema. Es entendida como un estudio de tipo teórico – práctico, con el propósito de ampliar y profundizar el conocimiento de su naturaleza. Se basa principalmente en fuentes bibliográficas, documentales y estudios comparados de análisis de problemas que ocurren en la práctica.

La metodología aplicada permitió evaluar varios aspectos relacionados al proceso de búsqueda y selección de documentos pertinentes a la investigación, para efectuar la recopilación de la información necesaria, se emplearon técnicas propias de la investigación documental, como lo es la búsqueda bibliográfica de las diversas fuentes de información existentes sobre el tema, la cual y a través de una lectura general de las mismas, permitió explorar los datos necesarios, además de proporcionar los elementos teóricos precisos para la mejor comprensión del problema objeto de la investigación.

En tal sentido la revisión bibliográfica, permitió conocer conceptos, características e importancia de la educación a distancia, su evolución y transformación para llegar a lo que hoy día, se conoce como e-learning y su representación desde el punto de vista de software, como lo son las plataformas de formación (LMS), además de estudiar los diferentes estándares creados para permitir la interconexión y el intercambio de información entre las LMS de diferentes fabricantes, por medio del cual se seleccionó al modelo SCORM como punto central

de esta investigación por ser un estándar que combina lo mejor de cada uno de sus antecesores.

Partiendo de las bondades que posee el modelo SCORM, su funcionabilidad y operabilidad se diseña una arquitectura de software basada en patrones que permita crear y publicar contenidos de aprendizajes en forma de objetos (SCO) de plataformas de formación (LMS) que no soportan SCORM, para que estos queden disponibles para ser utilizados en plataformas que si soporten el estándar.

Diseño de la Investigación

En cuanto al diseño de la investigación, éste obedece a la modalidad de diseño bibliográfico, pues se revisaron diferentes fuentes documentales bibliográficas para el logro de los objetivos. Esta investigación corresponde a la modalidad de Monográfica Documental, puesto que se analiza una situación desde el punto de vista teórico para generar una propuesta de una arquitectura de software basada en el modelo estándar SCORM y orientada al lenguaje de patrones.

Para diseñar dicha arquitectura de software a través de lenguajes de patrones se utilizó la metodología recomendada por Yacoub y Ammar (2004), la cual establece las siguientes actividades para llevar a cabo en este proceso.

- **Análisis de Requerimientos del Sistema:**

  En esta fase se revisó y analizó al modelo estándar SCORM, cuales son sus elementos principales, como se interrelacionan, para establecer los requerimientos funcionales y no funcionales de dicha arquitectura.

- **Revisión Bibliográfica de los Patrones:**

  En esta fase se busca toda la información referente a patrones y lenguajes de patrones que puedan ser útiles para la siguiente fase. Se clasifican de acuerdo a

su nivel de detalle y abstracción, según la taxonomía definida por Buschmann (1996) mostrada en el capítulo anterior. Para este paso existen una gran variedad de autores lo cual facilita la búsqueda, dentro de los cuales destacan: el Libro: "Dessign Patterns" de Gamma y otros (1995) el cual es un libro básico y muy popular que presenta 23 patrones que son muy útiles durante el diseño de software es conocido como los patrones de la "pandilla de los cuatros" por ser escrito por cuatro autores o patrones GoF (Gang-of-Four), el Libro "Pattern-Oriented Software Architecture" (POSA) volumen 1 y 2 de Buschmann (1996 y 2000 respectivamente) que promueven la discusión sobre patrones en cuestión arquitectura a gran escala, el libro UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado, de Craig Larman (2003). Así como estos existe una gran variedad de autores relacionados con los patrones y lenguaje de patrones que pueden aportar una gran parte en el diseño y desarrollo de software.

- **Selección de los Patrones:**

En esta fase se determina cuales son los patrones que más se adaptan al diseño de la arquitectura de software partiendo del modelo SCORM, tomando en cuenta el análisis de requerimiento elaborado y la revisión documental realizada en la fase anterior.

- **Diseño del Diagrama de nivel de detalle de Patrones:**

Una vez seleccionado los patrones, se hará uso del lenguaje de modelado unificado (UML), para representar a través de sus diagramas la arquitectura de software propuesta mediante los patrones seleccionados y sus relaciones.

Técnicas e Instrumentos de Recolección de Datos:

La técnica de recolección de información según Balestrini, (2001): "Son los métodos usados para extraer resultados que le den forma a la investigación,

cumpliendo con lo establecido en el sistema de variables y respondiendo de forma certera a los objetivos planteados". (p.147). En función de los objetivos definidos en el presente estudio, donde se propone diseñar una arquitectura de software partiendo del modelo estándar SCORM y basada en lenguaje de patrones, se emplearon una serie de instrumentos de recolección de información, orientados de manera esencial a alcanzar los fines propuestos.

Para el cumplimiento de esta estrategia, necesariamente hay que aplicar la técnica de revisión documental, que permita abordar y desarrollar los requisitos teóricos de la investigación; así como la técnica del resumen, subrayado, fichaje, presentación de cuadros, ilustraciones y diagramas, que permiten de manera teórica apoyar la investigación.

CAPITULO IV

RESULTADOS

Según Rebollo (2004), en los sistemas de e-learning, se identifican tres elementos, una plataforma de formación que da soporte a todas las actividades educativas y de gestión realizadas durante los aprendizajes (LMS), los contenidos para el estudio o curso y las herramientas de comunicación sincrónica o asincrónica, que permiten el contacto entre los participantes del curso.

El núcleo central de los sistemas de e-learning es la plataforma de formación, básicamente, se trata de una aplicación (o un conjunto de aplicaciones) cuyas funciones principales son:

- Gestionar Usuarios, en el cual se llevan a cabo los procesos de inscripción, control de acceso, control de los aprendizajes e itinerarios formativos, y generación de informes.

- Gestionar Cursos, incluye el control de la actividad de los estudiantes y los profesores, así como también su interacción con el material formativo, tests, evaluaciones y tiempos de acceso.

- Gestionar Servicios de Comunicación, como por ejemplo: correo electrónico, foros de discusión, salas de charlas virtuales (chat) o soporte para videoconferencia, los cuales sirven de apoyo y soporte al material formativo para el aprendizaje.

En esta investigación se plantea el diseño de una arquitectura de software, que permita crear y publicar objetos de aprendizajes (SCO) en plataformas de formación que no soportan el estándar SCORM, de tal manera que puedan ser aprovechados en plataformas LMS que si soporten el estándar.

Para el diseño de esta arquitectura se comenzó por conocer detalladamente el funcionamiento del estándar SCORM, se revisó los servicios que éste proporciona, la forma de empaquetamiento de la información y el proceso que este utiliza para crear el objeto de aprendizaje (SCO), con la idea de plantear los requerimientos que deben ser satisfechos por la arquitectura a diseñar. Una vez que se han establecido los requerimientos se puede diseñar la arquitectura, para lo cual se usa la metodología presentada por Yacoub y Ammar (2004), donde se presentan cuatro fases para el desarrollo de cualquier aplicación orientada a patrones.

Antes de comenzar a desarrollar cada una de las fases utilizadas en esta metodología, que es la pauta a seguir para el desarrollo del presente capítulo, se muestra una serie de conceptos de SCORM que se utilizan más adelante en los requerimientos del sistema.

El estándar SCORM.

El Modelo de Contenido de SCORM esta representado por: el Recurso (Asset) que es la forma más básica de representación del contenido. Un Asset, puede ser un texto, una imagen, un sonido, un objeto de evaluación o cualquier otra entidad que pueda mostrarse en un Web Browser.

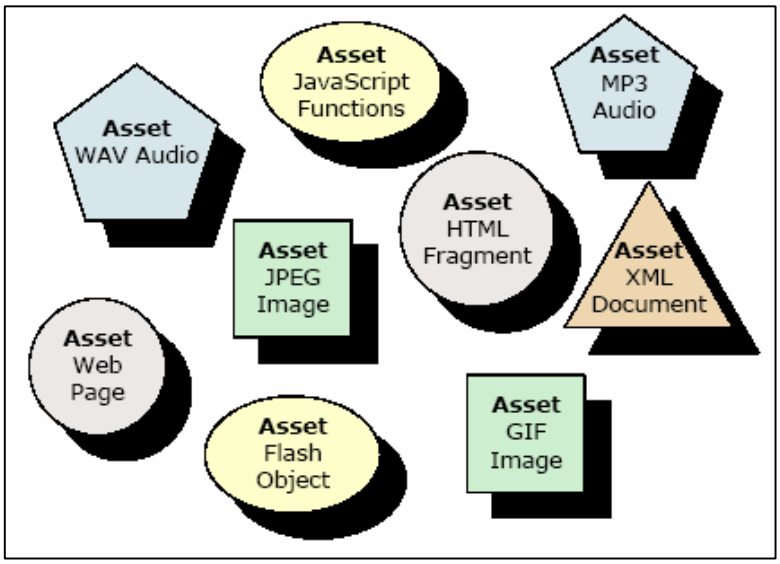

Figura 11: Ejemplos de Recursos (Asset). Tomado: http://www.adlnet.org (SCORM_CAM)

Ahora, un SCO es una colección de uno o más recursos que representan un recurso ejecutable, capaz de comunicarse y de ser lanzado o publicado por una plataforma de formación. La diferencia entre un SCO de un único recurso y el propio recurso es la capacidad que posee el SCO para comunicarse con las plataformas de formación, ya que utiliza IEEE ECMAScript API, que es el Entorno de Servicios de Comunicación para Contenidos. Por medio de la API que se encuentra en las LMS desarrolladas bajo el esquema de SCORM, el SCO es capaz de localizar el punto de entrada a la plataforma, iniciar la comunicación con ella, operar sobre la plataforma leyendo y escribiendo datos sobre ella y terminar la comunicación cuando ya no se necesite.

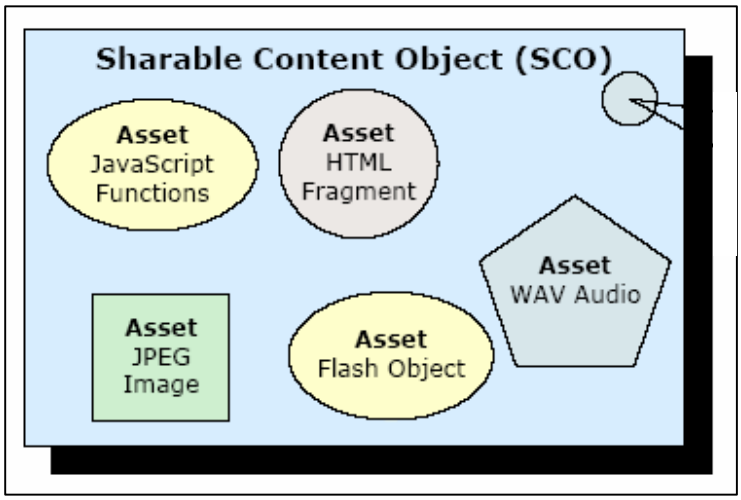

Figura 12: Ejemplo de un SCO. Tomado: http://www.adlnet.org (SCORM_CAM)

En la terminología utilizada en SCORM, la Organización de Contenidos, no es más que un mapa que simboliza el uso de los distintos contenidos, ya sean Recursos ó SCO; por medio de un conjunto de unidades de instrucción denominadas Actividades, las cuales pueden estar conformadas a su vez por más actividades, sin que existan límites en el nivel de anidamiento, ni ninguna relación preestablecida, tal como se muestra en la figura 13.

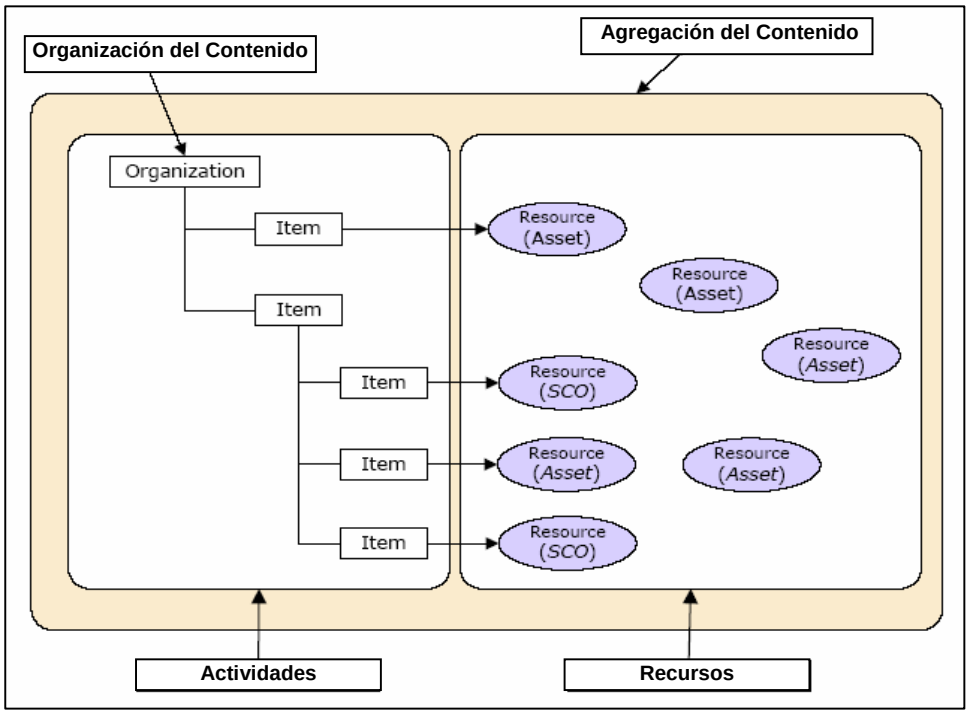

Figura 13: Organización del Contenidos y Actividades. Tomado: http://www.adlnet.org (SCORM_CAM)

Desarrollo de la Metodología.

La metodología recomendada por Yacoub y Ammar (2004), presenta una serie de actividades que se detallan a continuación:

1.- Requerimientos del Sistema.

Los detalles del estándar SCORM considerados en los párrafos anteriores, permiten delimitar los requerimientos funcionales y no funcionales que serán satisfechos por la arquitectura de software propuesta y que se pueden resumir de la siguiente manera:

Requerimientos Funcionales:

- **Atender las solicitudes de los clientes de manera concurrente y sincronizada para publicar objetos de contenidos de aprendizaje en formato SCORM.**

- **Establecer un mecanismo para estandarización de los cursos depositados en LMS que no soportan SCORM:** Uno de los objetivos de la presente investigación es lograr que los cursos de cualquier LMS sean intercambiables, es por eso que se usa el estándar SCORM, para ello el contenido que se desea publicar se debe llevar a un formato estándar, para luego convertirlo en un objeto de aprendizaje en formato SCORM.

- **Empaquetar los Objetos de Aprendizaje:** Para ello se utiliza de forma estricta las especificaciones de empaquetamiento de contenido proporcionado por la IMS y SCORM, que expresa de manera estandarizada la forma de crear la estructura y el comportamiento de una colección de contenidos de aprendizaje.

  Este paquete de contenido esta conformado por dos componentes:
  - Un documento en formato XML que describe la estructura del contenido y los recursos, llamado manifiesto (imsmanifest.xml).
  - Los archivos físicos (o URL) con el contenido real del paquete. El componente de archivos físicos está representado por los archivos actuales referenciados en el componente de recursos. Estos archivos pueden ser archivos locales que están referenciados dentro del paquete de contenidos, o por archivos externos que son referenciados por medio de una URL.

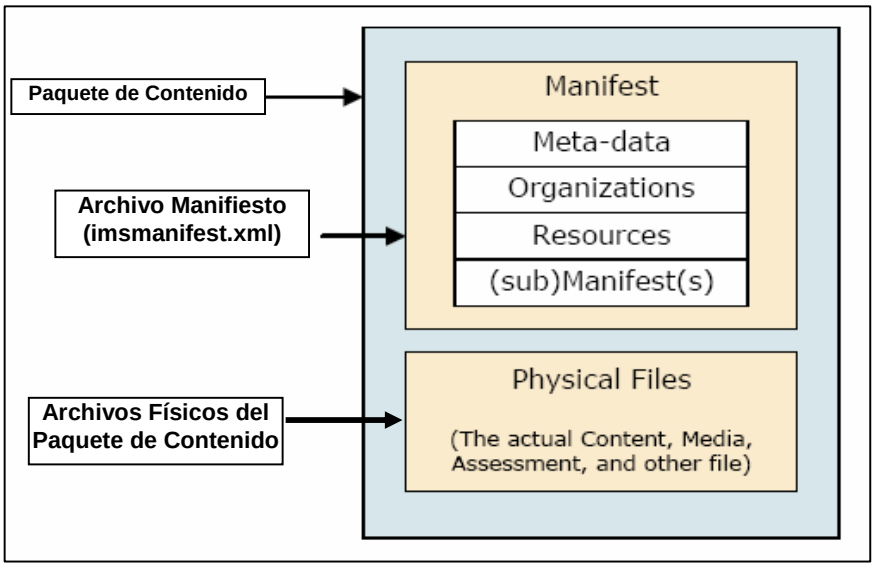

Figura 14: Diagrama Conceptual del Modelo de Empaquetamiento.
Tomado: http://www.adlnet.org (SCORM_CAM)

Un paquete de contenido representa una unidad de aprendizaje que puede ser parte de un curso, pero que tiene relevancia instruccional y que puede distribuirse independientemente. El paquete puede estar compuesto por una porción de un curso, un curso completo o una colección de cursos. Una vez que el paquete llega a su destino debe ser capaz de agregarse o desagregarse. Un paquete debe funcionar por si solo, es decir, debe contener toda la información necesaria para poder usar el contenido cuando este es desempaquetado.

El manifiesto y todos los archivos de contenidos se agrupan en un único archivo comprimido en formato PKZip v2.04g (.zip), que en SCORM se le denomina PIF (Package Interchange File – Archivo del Paquete de Intercambio). El manifiesto contiene la información necesaria para describir lo que contiene el paquete. Está conformado como se observa en la figura 14, por cuatro (4) elementos que se describen a continuación.

- **Metadatos (Metadata):** esta sección contiene la información que describe el paquete de contenido, como por ejemplo: el estándar que utiliza, la versión y el lenguaje del contenido. Además la búsqueda y el descubrimiento de la información del paquete de contenido.

- **Organización (Organizations):** en esta sección se representa la organización de los contenidos y su descomposición en actividades (items). Cada actividad está enlazada a los recursos que utiliza a través de su identificador (identifier), los cuales se encuentran en la siguiente sección. Este elemento incorpora también las instrucciones de secuenciación y navegación.

Para la organización del contenido no existe ninguna regla o norma, es decir, ni IMS ni SCORM establecen una taxonomía o nomenclatura para establecer la organización del contenido, mucho menos una jerarquía con respecto a la información a publicar, como por ejemplo: un curso contiene módulo y ese módulo tiene lecciones y estas poseen actividades, esta nomenclatura no esta predefinida. La organización del contenido es decisión del diseñador y/o del desarrollador, es decir, el contenidos se organiza a través de actividades con sus respectivos recursos.

- **Recursos:** aquí se describen los recursos externos (a través de URL) y locales que utiliza el paquete. Los recursos locales se encuentran comprimidos en el mismo PIF. Si el recurso necesita comunicarse con la plataforma, debe estar empaquetado como un SCO. En otro caso, puede ser un Recurso (Asset).

Los recursos de un manifiesto pueden describir los recursos externos, así como se hace con los archivos físicos localizados en el paquete. Estos archivos pueden ser: archivos de multimedia, archivos de texto, u otros pedazos de datos presentados en forma digital. Estos recursos pueden representar las agrupaciones conceptuales y las relaciones entre los archivos, dentro del componente de recursos. La combinación de recursos generalmente se categoriza como contenido. Los

recursos se envían a diferentes puntos dentro del componente de las organizaciones que mantienen la estructura de los recursos. Un recurso puede estar formado por múltiples componentes. En SCORM, estos componentes son los recursos simples. Si el Recurso fue construido para comunicarse con una LMS entonces el Recurso es un SCO, sino es simplemente un recurso. La colección de componentes de recursos es una organización. Esta colección de recursos y la organización de ellos definen la Organización del Contenido.

- **SubManifiestos:** Los recursos complejos suelen estar formados por una jerarquía de entidades, cada una de las cuales tiene su propio manifiesto (cursos, lecciones, módulos, entre otros). En ese caso, al construir el objeto agregado, es necesario indicar la dependencia existente entre los distintos componentes del recurso de aprendizaje.

Para la secuenciación de los contenidos de un paquete y la presentación de los mismos en un Web browser, en el manifiesto se incluye las reglas y estrategias de las definiciones de las actividades (dentro de las mismas actividades), y de igual manera se incluye la organización de las actividades.

Requerimientos no funcionales.

- **Escalabilidad:** el diseño debe contemplar el uso óptimo de los recursos.
- **Disponibilidad:** la disponibilidad de la aplicación debe ser continua y garantizada para todos los clientes de sistema y en cualquier momento.
- **Seguridad:** se debe garantizar que no se pierda el contenido del paquete durante la creación del SCO.
- **Desempeño:** La aplicación debe ofrecer un buen desempeño del sistema ante una alta demanda de clientes.

2.- Revisión Bibliográfica y Selección de los Patrones.

En esta fase de la metodología se revisaron diferentes autores reconocidos dentro del área de conocimientos referentes a Patrones. Entre los cuales destacan los patrones referenciados por Gamma (1995), el cual es un libro básico en el tema de patrones y muy popular donde describe veintitrés (23) patrones que son muy útiles durante el diseño de objetos, Buschmann (1996) el cual establece una clasificación para los patrones, determina que existen patrones arquitecturales, de diseño y de idiomas, dependiendo del nivel detalle y abstracción de la aplicación. Luego, Buschmann y otros (2000), presentan una serie de patrones arquitecturales y de diseño para el desarrollo e implementación de aplicaciones concurrentes y para trabajo en red.

Luego de la revisión bibliográfica realizada referente a patrones, se presenta y se muestra una breve explicación de los patrones seleccionados para el diseño de esta propuesta.

- **Reactor.**

El patrón arquitectural **Reactor** permite a las aplicaciones que manejan eventos, demultiplexar y despachar solicitudes requeridas por uno o más clientes. También es conocido por Despachador o Notificador.

Por ejemplo: cuando se tiene un servidor que maneja eventos para un servicio de logging distribuido. Las aplicaciones cliente remotas usan este servicio de logging para registrar información sobre su estado dentro del sistema distribuido. Este estado comúnmente incluye: notificaciones de error, depuración de su sistema, y diagnósticos de su ejecución. Los registros de logging de estas aplicaciones son enviados al servidor central de logging y registrados desde varios dispositivos de

salida, como por ejemplos una consola, una impresora, un archivo, o un administrador de base de datos en red.

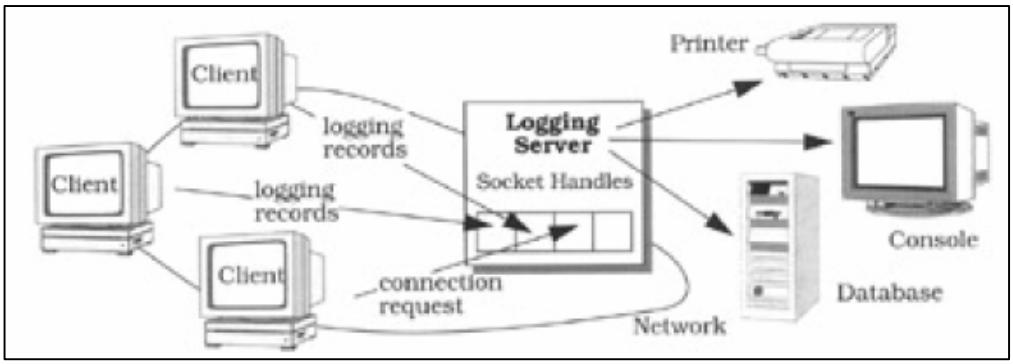

Figura 15: Ejemplo del patrón Reactor. Tomado: Buschmann (2000)

Los clientes se comunican con el servidor de logging a través de un protocolo orientado a la conexión (TCP). Los clientes realizan el servicio de logging a través de asociaciones que consisten en direcciones IP e identifican al cliente y el servicio por el número de puerto en TCP.

Un servicio de logging puede accederse simultáneamente por múltiples clientes, donde cada uno mantiene su propia conexión con el servidor de logging. Cuando un cliente solicita un requerimiento de servicio, este es indicado al servidor por un evento CONNECT. Para procesar un requerimiento de logging de un cliente es procesado por un evento READ, el cual le da instrucciones al servicio de logging para leer la nueva entrada de uno de los clientes conectados. Los registros de logging y las solicitudes de conexión emitidos por los clientes pueden llegar concurrentemente al servidor de logging.

Una forma de implementar el servidor de logging es utilizando algún tipo de modelo multi-hilos. Por ejemplo, el servidor podría usar un modelo de conexión por hilo de ejecución asignando un hilo de control dedicado para cada conexión y un

proceso de registro de logging para cada uno de los clientes que lo soliciten. Usando un modelo multi-hilo puede incurrir en las siguientes obligaciones:

- El hilo puede ser ineficiente y no escalable para el contexto de intercambio, sincronización y movimiento de datos entre CPUs.

- El hilo puede requerir un esquema de control de concurrencia compleja a lo largo de todo el código del servidor.

- Los hilos no están disponibles en todos los sistemas operativos, ni todos los sistemas operativos proporcionan la semántica de hilos portátiles.

- Un servidor concurrente puede ser optimizado alineando su estrategia a los recursos disponibles, como el número de CPUs, en lugar de, al número de clientes concurrentes.

Estas desventajas pueden hacer al modelo multi-hilo ineficaz y demasiado compleja para desarrollar un servidor de logging. El servidor de logging debe asegurar la calidad adecuada para todos los clientes que se conectan, de manera eficaz y justa. En particular, no debe servir a un solo cliente y dejar esperando a los demás.

Por lo tanto, una solución para este problema puede ser una implementación del patrón Reactor, el cual es una aplicación manejada por eventos que recibe solicitudes de múltiples servicios simultáneos, y los procesa de manera sincronizada y consecutiva.

Integra los mecanismos del demultiplexor y permite despachar los eventos de servicios que procesa. Desacopla el demultiplexor de eventos y los mecanismos de despacho de una aplicación específica procesando los eventos a lo largo del servicio.

En detalle: para cada servicio ofrece una aplicación que: introduce un manejador de eventos separado, que procesa ciertos tipos de eventos para cierta fuente de eventos. El manejador de eventos se registra con un patrón arquitectural reactor, el cual usa un demultiplexor de eventos sincrónicos para esperar la indicación que pueda ocurrir de una o más fuentes de eventos. Cuando ocurre la indicación del evento, el demultiplexor de eventos sincrónicos notifica al reactor, el cual de forma sincronizada despacha el manejador de eventos asociado al evento para que este ejecute la solicitud de servicio.

A continuación se muestra el diagrama de clases que representa al patrón arquitectural Reactor.

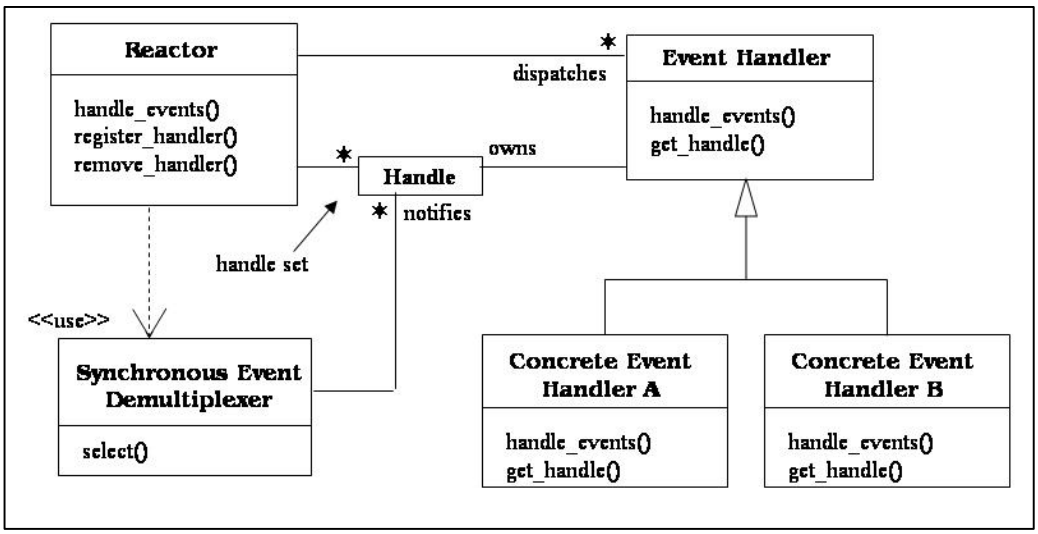

Figura 16: Diagrama de Clases del Patrón Reactor. Tomado: Buschmann (2000)

- **Acceptor-Connector**

El patrón de diseño Acceptor-Connector desacopla las conexiones y los servicios inicializados para el procesamiento de servicio en un sistema de red. Este está contemplado dentro del desarrollo de aplicaciones o sistemas en red, en la cual el

protocolo orientado a conexión es usado para la comunicación de servicios entre pares conectados vía transporte endpoints.

Las aplicaciones en los sistemas orientados a conexión en red, a menudo contienen una gran cantidad de código de configuración para realizar las conexiones e inicializar los servicios. Este código de configuración es independiente del procesamiento que ejecuta los servicios de intercambio de datos. El acoplamiento del código de configuración y código del procesamiento de servicio puede ser inconsistente, debido a que no resuelve los siguientes cuatros puntos importantes:

- Debe ser fácil cambiar los roles de conexión para soportar las conductas de diferentes aplicaciones. Los roles de conexión determinan si la conexión es iniciada activamente o si es pasivamente aceptada. En contraste a esto, en una configuración entre pares los roles de conexión, determina si la aplicación juega un papel de cliente, o de servidor, o de ambos, es decir, de cliente y servidor.

- Debe ser fácil adicionar nuevos tipos de servicios, implementar dichos servicios y el protocolo de comunicación sin afectar el código de configuración del establecimiento de las conexiones existente y el servicio de inicialización.

- En general, el establecimiento de conexiones y las estrategias de inicialización de servicio cambian menos frecuentemente que los protocolos de comunicación y servicios en una aplicación.

- Para los sistemas en red a gran escala, en lo posible se debe reducir el establecimiento conexiones latentes, usando los mecanismos que ofrece los sistemas operativos avanzados, como por ejemplo: el mecanismo de conexión asincrónico.

Ahora, para resolver este problema se debe desacoplar la conexión e inicialización de servicios entre pares en una aplicación en red, para procesar estos servicios entre pares se realiza después que ellos sean conectados e inicializados; de la siguiente manera: Se encapsula los servicios de la aplicación dentro del manejador de servicios entre pares. Cada manejador de servicio implementa una mitad del servicio de end-to-end en una aplicación de red. Se conecta y se inicializa el manejador de servicio entre pares usando dos (2) factores: un aceptor y un conector. Ambos factores cooperan para crear la asociación entre dos (2) manejadores de servicios entre pares y ellos dos (2) conectan al transporte endpoints. Cada encapsulado puede ser transportado por el manejador.

El factor aceptor establece las conexiones pasivamente a través del manejador de servicios entre pares, asociadas al evento de solicitud de conexión emitida por el manejador remoto de servicios entre pares. Igualmente, el factor conector establece las conexiones activamente a través del manejador remoto de servicios entre pares con el manejador de servicios entre pares.

Después que se establece conexión, los factores aceptor y conector inicializan a sus manejadores de servicios entre pares y pasan a ellos sus respectivos manejadores de transporte. El manejador de servicios entre pares ejecuta el proceso específico de la aplicación, usando sus manejadores de transporte para intercambiar datos vía transporte endpoints. En general, el manejador de servicios no interactúa con los factores aceptor y conector después que ellos son conectados e inicializados. Este proceso es mostrado en el siguiente diagrama de clases del patrón de diseño Acceptor-Connector.

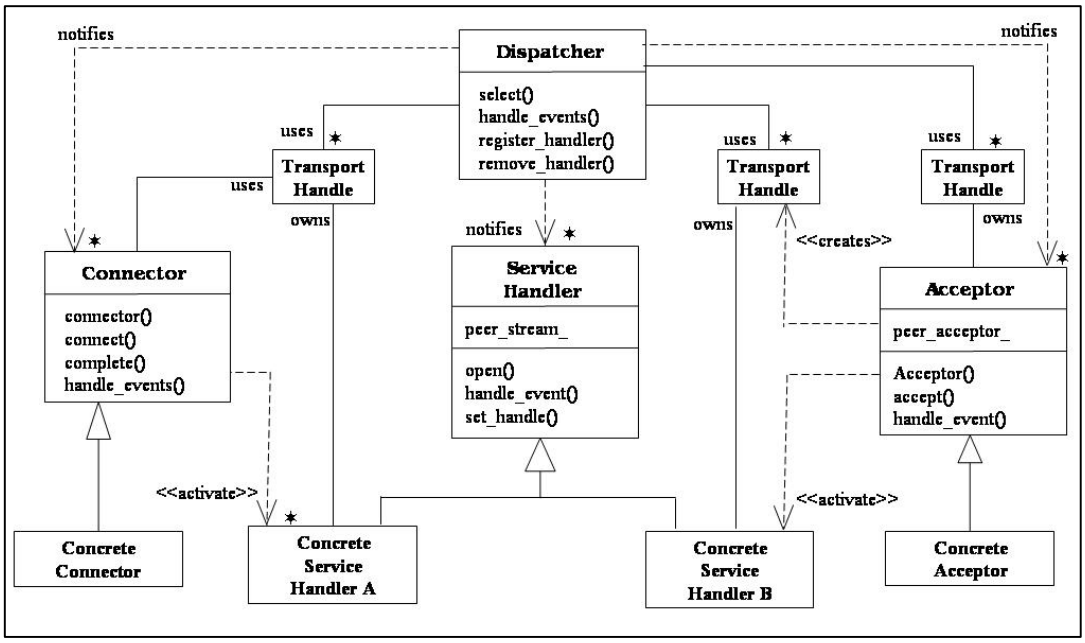

Figura 17: Diagrama de Clases del Patrón Acceptor -Connector. Tomado: Buschmann (2000)

- **Monitor Object.**

El patrón de diseño Monitor Object sincroniza métodos concurrentes de ejecución, para asegurar que solamente un método del objeto se ejecute en un momento determinado. También permite que métodos del objeto fijen una secuencia de ejecución cooperativamente. Este patrón también es conocido con el nombre de Thread-safe Passive Object. El contexto de implementación de este patrón esta caracterizado por múltiples hilos de control accediendo al mismo objeto concurrentemente.

Muchas aplicaciones contienen objetos cuyos métodos son invocados concurrentemente por múltiples hilos cliente. Estos métodos modifican a menudo el estado del objeto. Cada aplicación coexistente debe ejecutarse correctamente, por consiguiente, es necesario sincronizar y fijar el acceso a los objetos.

En la presencia de este problema debe tomarse en cuenta cuatro puntos:

- Para separar lo concerniente del estado del objeto y protegerlo de los cambios descontrolados, los programadores orientados a objeto están acostumbrados a acceder al objeto a través de los métodos de la interface. Esto es realmente sencillo extender su modelo de programación orientada a objeto para proteger los datos del objeto de los cambios concurrentes descontrolados, conociendo las condiciones mínimas. El método de interface de un objeto debe definir los límites de la sincronización, por consiguiente, un solo método debe se activado en un momento determinado dentro del mismo objeto.

- Las aplicaciones concurrentes son más difícil de programar si los clientes pueden adquirir explícitamente y soltar los mecanismos de sincronización de bajo nivel, como por ejemplo: semáforos, variables condicionadas, entre otros. Los objetos deben ser responsables de asegurar la calidad de cada uno de sus métodos que requieran sincronización, éstos son ejecutados transparentemente sin que requiera la invocación de un cliente explícitamente.

- Si los métodos de un objeto se deben bloquear durante su ejecución, estos deben ser capaces de abandonar su hilo de control voluntariamente, para que los métodos llamadores de otros hilos cliente puedan acceder al objeto. Esta propiedad ayuda a prevenir el bloqueo del objeto y la realización de los mecanismos de concurrencia, disponibles entre el hardware y la plataforma de software.

- Cuando un método abandona su hilo de control voluntariamente, este debe dejar el estado del objeto en estable, es decir, debe soportar las variantes específicas del objeto. Similarmente, un método sólo debe reasumir su ejecución dentro de un objeto cuando el objeto está en un estado estable.

Cada uno de estos problemas se solucionan sincronizando los accesos a los métodos de un objeto para que solamente se ejecute un método a la vez. Detalladamente, cada objeto es accedido concurrentemente por múltiples clientes hilos, para ello se define un monitor object. Los clientes pueden acceder a las funciones definidas por un monitor object sólo a través de los métodos de sincronización. Para prevenir condiciones mínima del estado interno del objeto, sólo un método sincronizado puede ejecutarse dentro del monitor object. Para realizar accesos concurrentes en serie al estado del objeto, cada monitor object contiene un monitor locking. Los métodos sincronizados pueden determinar la situación en la cual el objeto se suspendió o reasumió su ejecución, basándose en uno o más monitor conditions asociadas con un monitor object, gráficamente se representa por el siguiente diagrama de clases.

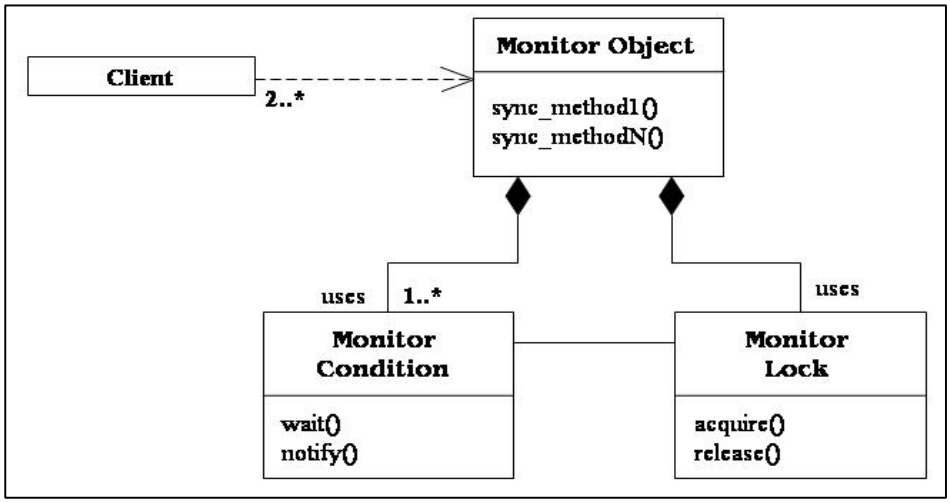

Figura 18: Diagrama de Clases del Patrón Monitor Object. Tomado: Buschmann (2000)

- **Wrapper Facade.**

El patrón de diseño Wrapper Facade encapsula las funciones y los datos proporcionados por las APIs existentes no orientadas a objeto, dentro de clases

orientadas a objetos más concisas, robustas, portables, mantenibles y cohesivas. Su contexto se encuentra en aplicaciones desarrolladas y mantenibles, que poseen accesos a mecanismos o datos proporcionados por APIs existentes no orientadas a objetos.

Las aplicaciones son a menudo APIs escritas en sistemas operativos no orientados a objetos o bibliotecas del sistema. Estas APIs acceden a red y a mecanismos de programación basados en hilos, así como intefaces de usuarios o bases de datos de bibliotecas de programas. Aunque este modelo es común, causa problemas a los diseñadores de aplicaciones, si no se fortalece los siguientes puntos:

- El código conciso es a menudo más robusto que el código verboso porque es más fácil de desarrollar y mantener. Usando lenguajes orientados a objetos pueden ofrecer soportes a un nivel más alto, como por ejemplos constructores, destructores, excepciones y recolectores de basura, reduciendo así las probabilidades de errores comunes en programación. Sin embargo, los diseñadores que programan en bajo nivel con funciones basadas en APIs directamente, tienden a reescribir mucho software verboso y propenso a errores.

- El software que es portátil o que puede ser portable fácilmente entre diferentes sistemas operativos, compiladores y plataformas de hardware, ayudan al aumento de productos en una porción del mercado. Aunque, reusando las APIs de bajo nivel existentes se puede reducir el esfuerzo de desarrollo de software, las aplicaciones desarrolladas bajo este esquema son a menudo no portátiles.

- Mejorando la mantenibilidad del software se ayuda a reducir los costos del ciclo de vida. Los programas escritos directamente por API no orientadas a objetos y de bajo nivel, generalmente son difícil de mantener, sin embargo,

por ejemplo existen desarrolladores de C y C++ que incluyen en su desarrollo portabilidad, insertando directivas compilación condicionada a la fuente de la aplicación. Desafortunadamente, si se dirige a las variaciones específicas de las plataformas por la vía de la compilación condicional se incrementa la complejidad del desarrollo físico de software. Para una instancia de una plataforma específica, los detalles comienzan a extenderse a lo largo de los archivos fuentes de la aplicación.

- Los componentes cohesivos son más fáciles de entender, mantener y reforzar. Sin embargo, raramente las APIs de bajo nivel se agrupan en componentes cohesivos porque los lenguajes como C, por ejemplo carecen de características como clases, namespaces, o paquetes. Por lo tanto, es difícil reconocer la magnitud de las APIs de bajo nivel. Además, programando con APIs no cohesivas y autosuficientes, se extiende el código a lo largo de la aplicación, lo que lo hace un trabajo duro, mientras que utilizando un plugin en sus nuevos componentes, se apoyan las diferentes políticas y mecanismos.

En general, las aplicaciones desarrolladas utilizando APIs no orientadas a objetos, proporcionan un modelo pobre en desarrollo de software, las cuales deben mantenerse y evolucionar con el tiempo. Para solventar cada uno de los problemas mencionados anteriormente, se debe evitar acceder a las APIs no orientadas a objetos directamente. Para cada conjunto de funciones y datos relacionados con una API no orientada a objeto, se debe crear uno o más patrones Wrapper Facade para encapsular estas funciones y datos dentro de métodos más concisos, robustos, portables y mantenibles provistos por el patrón Wrapper facade orientado a objeto.

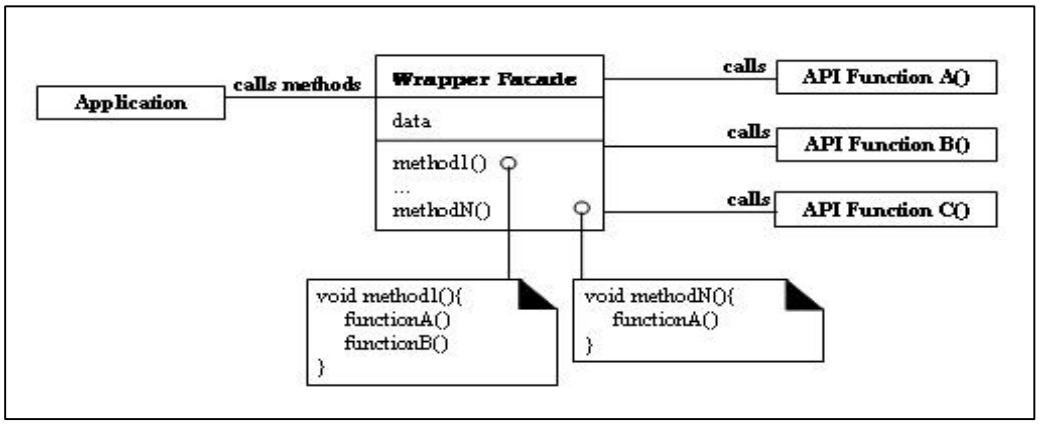

Figura 19: Diagrama de Clases del Patrón Wrapper Facade. Tomado: Buschmann (2000)

- **Strategy.**

El patrón de diseño Strategy define una familia de algoritmos, encapsulando cada uno y haciéndolos intercambiables. Su utilidad se presenta cuando:

- Son necesitadas diferentes variantes de un mismo algoritmo.

- Existen varias subclases y sólo difieren en su comportamiento.

- Una clase define varios comportamientos en múltiples sentencias condicionales.

Por ejemplo, se tiene inicialmente un algoritmo encapsulado en la clase "ClassXYZ", pero se necesita hacer el diseño de la aplicación más flexible, porque se requiere tener la posibilidad de aplicar diferentes algoritmos en tiempo de ejecución.

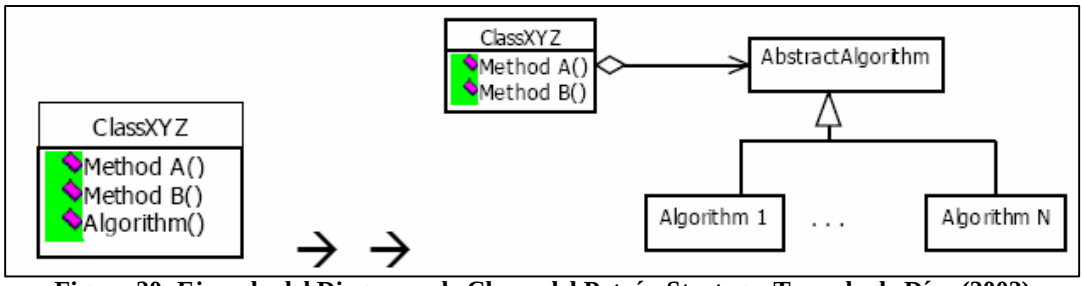

Figura 20: Ejemplo del Diagrama de Clases del Patrón Strategy. Tomado de Díaz (2002)

La solución consiste en desacoplar el algoritmo que se encuentra encapsulado en la "Clase XYZ", definiendo un interfaz general (AbstractAlgorithm) para el grupo de algoritmos se desea codificar, y encapsular cada uno de estos en una clase (Algorithm 1, Algortithm 2, …, Algorithm N), haciéndolos intercambiables.

El patrón Strategy es representado por el siguiente diagrama de clases general para cualquier aplicación.

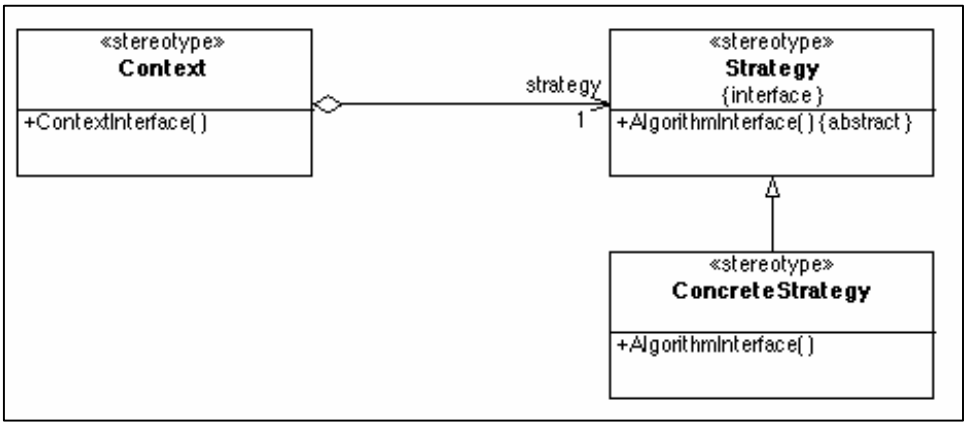

Figura 21: Diagrama de Clase del Patrón Strategy. Gamma (1995)

- **Component Configurator.**

El patrón de diseño Component Configurator permite a una aplicación enlazar o desenlazar un componente de implementación en tiempo de ejecución sin tener que

modificar, recompilar o reenlazarse estáticamente a la aplicación. El component configurator promueve el soporte de reconfiguración de componentes dentro de diferentes procesos de la aplicación, sin tener que parar y recomenzar la ejecución de la aplicación. Este patrón también es conocido como Service Configurator.

Este patrón esta inmerso en aplicaciones o sistemas en las cuales sus componentes necesiten inicializar, suspender, resumir o terminar de forma flexible y transparente.

Las aplicaciones que son conformadas por componentes, pueden proporcionar mecanismos para configurar estos componentes dentro de uno o más procesos. La solución a este problema es influenciada por los siguientes puntos:

- Los cambios de funcionabilidad de un componente o los detalles de una implementación son comunes en muchos sistemas o aplicaciones. Por ejemplo, los mejores algoritmos o arquitecturas pueden ser descubiertos en aplicaciones maduras. Sin embargo, debería ser posible modificar la implementación de un componente por cualquier punto durante el desarrollo o la ejecución de ciclo de vida de la aplicación.

  La modificación de este componente debe tener un impacto mínimo en la implementación de otros componentes que este use. Similarmente, debe ser posible iniciar, suspender, resumir, terminar e intercambiar dinámicamente componentes dentro de una aplicación en ejecución. Estas actividades deberían tener un impacto mínimo en otros componentes que son configurados dentro de la aplicación.

- Los desarrolladores, a menudo, no pueden saber la forma más efectiva de colocar o distribuir los componentes dentro de procesos al momento que desarrolla la aplicación. Si los desarrolladores comprometen una

configuración particular de un componente pueden impedir la flexibilidad, reducen la ejecución global del sistema e incrementan innecesariamente la utilización de recursos.

Además, la decisión de configuración inicial de un componente puede ocasionar con el tiempo un uso no óptimo del componente. Por ejemplo, las actualizaciones de plataforma o el incremento del trabajo, pueden requerir la redistribución de ciertos componentes para otros procesos o hosts. En algunos casos, puede ser útil postergar la configuración de los componentes tanto como sea posible en el desarrollo o la ejecución del ciclo de vida de las aplicaciones.

- La ejecución de tareas administrativas como configurar, iniciar, y controlar los componentes deben ser sinceradas e independientes del componente. Estas pueden manejarse, a menudo, eficientemente por un administrador central en lugar de ser distribuidas a lo largo de la aplicación. Estas deberían ser automatizadas por ejemplo por algún mecanismo scripts.

Para dar solución a estos problemas se debe desacoplar los componentes de su implementación y realizar la configuración de estos componentes independiente a los procesos de la aplicación. Este proceso se puede llevar a cabo de la siguiente forma: un componente define una interface uniforme para configurar y controlar un tipo de servicio en particular de la aplicación y la funcionabilidad que esta proporciona. Los componentes concretos llevan a cabo esta interface de una manera específica de la aplicación. Las aplicaciones o administradores pueden usar la interface del componente para comenzar, suspender, resumir o terminar dinámicamente sus componentes concretos, así como para obtener información en tiempo de ejecución referente a la configuración del componente concreto. Los componentes concretos son empaquetados dentro de una unidad apropiada de configuración, como por ejemplo: una biblioteca enlazada dinámicamente (DLL). Este DLL puede ser

dinámicamente enlazada o desenlazada dentro o fuera de una aplicación bajo el control de Component Configurator, los cuales utilizan un repositorio para almacenar todos los componentes concretos configurados dentro de la aplicación.

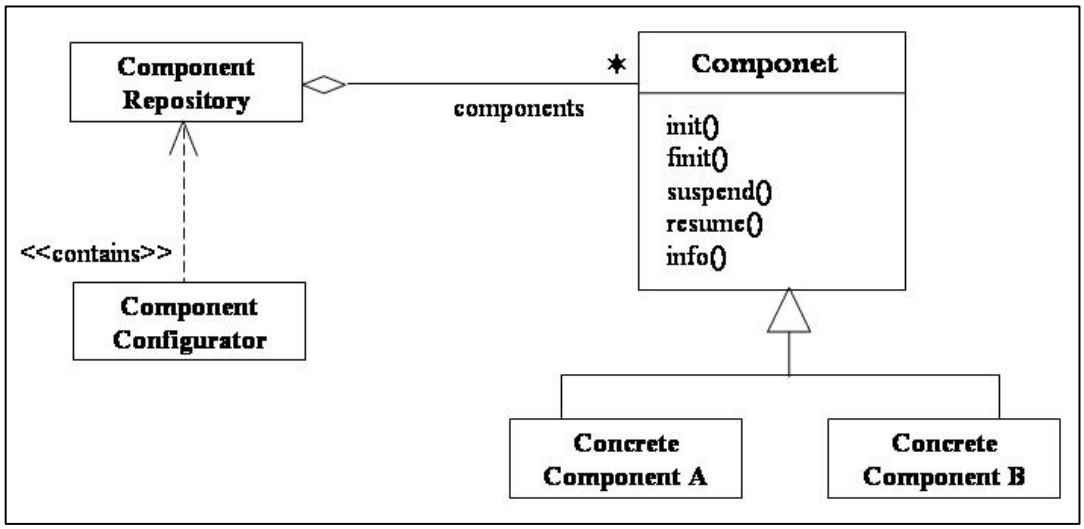

Figura 22: Diagrama de Clase del Patrón Component Configurator. Tomado: Buschmann (2000)

- **Composite.**

El patrón de diseño Composite, permite componer objetos en una estructura de árbol, donde cada nodo no importa si es un nodo compuesto o uno simple, es decir, donde los objetos compuestos se tratan de forma similar a un objeto simple. Por ejemplo, en la arquitectura .Net la aplicación de este patrón permite la creación de un XmlDocument (Documento Xml).

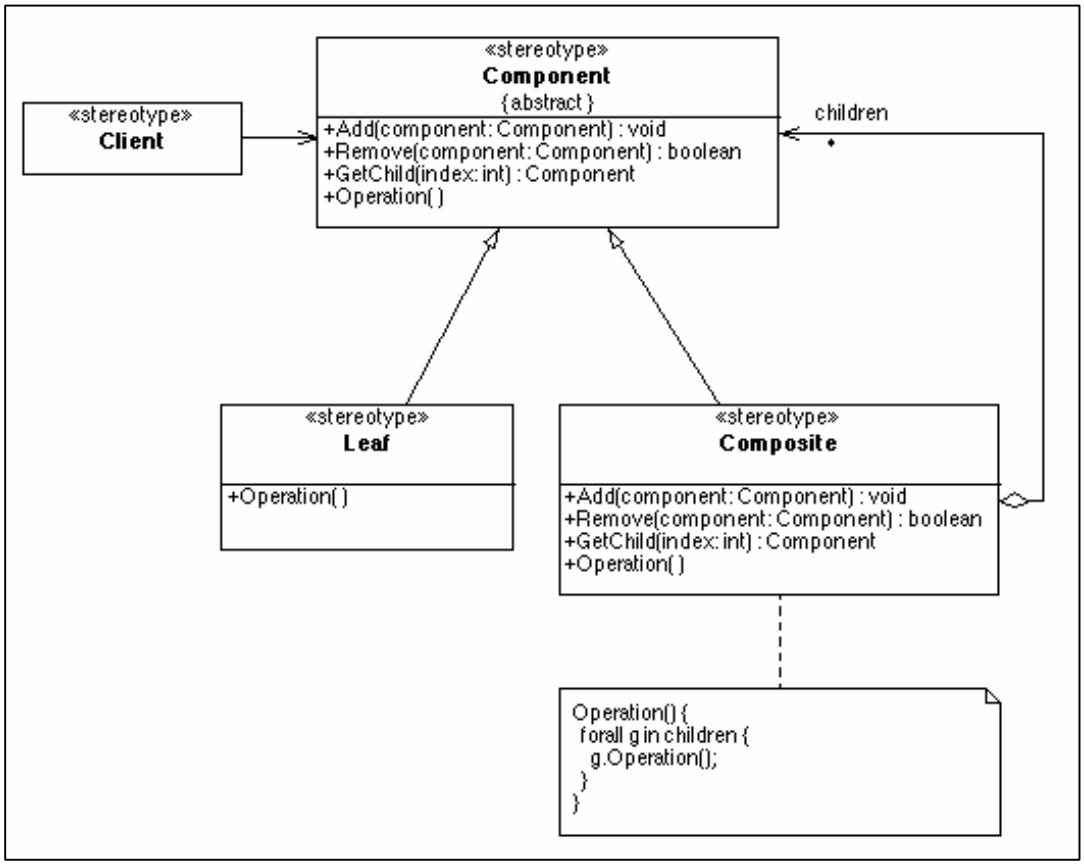

Figura 23: Diagrama de Clases del Patrón Composite. Tomado: Gamma (1995)

3.- Elaboración de la propuesta a partir de la selección de patrones.

Continuando con el desarrollo de la metodología se procede a la elaboración de la propuesta a partir de los patrones seleccionados. Para ello, Buschmann y otros (2000), indican que para la construcción de un sistema de software efectivo se deben relacionar, en la mayoría de los casos más de dos (2) patrones, debido a que desafortunadamente, no existe un patrón único que resuelva el problema.

Se pueden presentar soluciones a problemas enfocados desde diferentes ámbitos, pero la solución real de un sistema involucra por lo general más de un problema, es

por esto que existe el lenguaje de patrones, que no es más que un conjuntos de patrones relacionado para solventar una situación real.

Este lenguaje queda a decisión del diseñador, donde la persona establece como organizar los patrones de forma que sean la solución requerida. Para esto no existen reglas, debe basarse en los requerimientos de la aplicación y en la amplia gama de patrones existentes clasificados por categorías, los cuales se deben relacionar para resolver su problema.

En la figura 24, se muestra el diagrama que representa el lenguaje de patrones creados por la autora de la investigación, para representar la arquitectura de software propuesta que permite la publicación de objetos de aprendizajes (SCO).

Este diagrama de patrones, representa el nivel de abstracción más alto de la propuesta, y la forma como se modelan estos patrones se tomó del libro de Buschmann y otros (2000) que permite mostrar la nomenclatura de Gamma (1995) de representar patrones en su nivel de abstracción más alto, y donde se visualiza la interacción que poseen los patrones en la aplicación. En este diagrama se presenta los patrones arquitecturales sombreados para distinguirlos de los patrones de diseño.

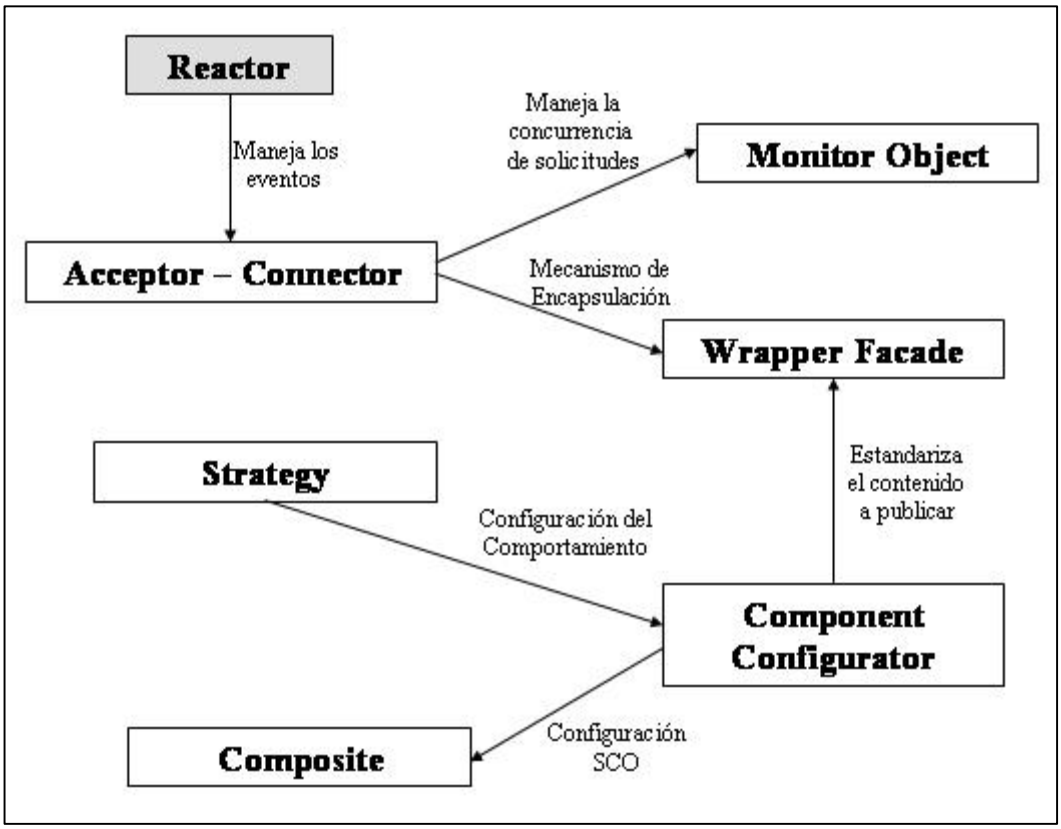

Figura 24: Diseño de la Arquitectura de Software Propuesta. Fuente: Autor de la Investigación

Dado que Buschmann y otros (2000), mencionan que la importancia del lenguaje de patrones radica particularmente en la clasificación apropiada de los patrones arquitecturales, los cuales definen la arquitectura de fondo para la línea del sistema del software completo, se comienza a detallar el diseño de la arquitectura de software por el patrón Reactor, el cual es el patrón arquitectural de la propuesta.

Tomando en cuenta los requerimientos antes expuestos, se observa que la solución requerida involucra una aplicación que contenga un manejador se solicitudes realizadas por múltiples clientes (LMS que si soportan SCORM) y que se debe dar respuesta de manera sincronizada y continua, para ello se propone utilizar el patrón Reactor como patrón arquitectural ya que en su contexto provee una solución a

aplicaciones Web manejado por eventos que recibe solicitudes múltiples clientes simultáneos y los procesa de manera sincronizada y continua.

Pero no solo el patrón arquitectural, provee la solución completa al problema planteado, es por ello que se incorporan a la propuesta algunos patrones de diseños cuidadosamente seleccionados para satisfacer los requerimientos del problema. Con respecto a esto, Buschmann y otros (2000) señalan que cada componente de dicha estructura es complejo por sí solo y que a menudo estos componentes usan patrones de diseño para llevarse a cabo.

Por lo tanto, a continuación se describen las relaciones existentes entre los diferentes patrones de la propuesta.

El patrón arquitectural Reactor muestra en su estructura la utilización de tres tipos de manejadores: un aceptor de eventos, un conector de eventos y un manejador de servicio, los dos primeros manejadores se resuelven utilizando el patrón de diseño Acceptor-Connector, ya que este es capaz de desacoplar la aceptación e inicialización de la conexión del servicio solicitado por el cliente. Ahora para el manejador de servicio se puede complementar con la utilización del patrón de diseño Monitor Object, ya que este permite sincronizar las solicitudes de servicio de cada una de las diferentes LMS, que soporten SCORM y que deseen utilizar el contenido publicado, para que sean atendidas de manera concurrente. De esta manera, se logra resolver la aceptación e inicialización de la solicitud realizada por los clientes y además gestionar el servicio solicitado de manera concurrente y sincronizada.

Luego que se resuelve la conexión de los clientes, y se desacopla de los servicios solicitados, se le agrega a la arquitectura el patrón de diseño Wrapper Facade, el cual funciona en conjunto con los patrones de diseño Strategy y Component Configurator en el proceso que se lleva a cabo para la estandarización del contenido de aprendizaje que se desea publicar.

La estandarización del contenido, se realiza a través de los patrones de diseño Wrapper Facade, Component Configurator y Strategy, donde el patrón Component Configurator obtiene la configuración de la LMS propietaria del contenido, por medio de la familia de algoritmo que posee el patrón Strategy, en el cual se define las características propias de cada LMS. Con el patrón de diseño Component Configurator, se puede reconfigurar un componente en tiempo de ejecución sin tener que parar y recomenzar la ejecución del sistema. Luego actúa el patrón de diseño Wrapper Facade, el cual permite encapsular el contenido estandarizado.

Por último y una vez que se tiene el contenido de aprendizaje en un formato estándar generado del trabajo en conjunto de los patrones de diseño Wrapper Facade, Strategy y Component Configurator, se utiliza el patrón de diseño Composite para crear el objeto de aprendizaje (SCO), siguiendo las normas establecidas por la IMS y SCORM para el empaquetamiento y organización del contenido, para crear el archivo manifiesto descrito anteriormente.

Una vez, que se tiene una visión amplia del sistema, se presenta un Diagrama de Interface, que permite detallar a través de que interface se comunican los patrones seleccionados.

Antes de mostrar las relaciones entre interfaces, primero se debe definir que interface posee cada uno de los patrones utilizados en la Figura 24, el cual es diagrama de alto nivel de la arquitectura propuesta. Las interfaces utilizadas por el patrón arquitectural Reactor son: la clase interface EventHandler y la clase interface Dispacth, las cuales permiten la conexión con el patrón Acceptor-Connector. El patrón de diseño Acceptor-Connector, utiliza la clase interface Acceptor y la clase interface Dispacth, cada una de estas interfaces proporcionan la aceptación e inicialización de conexión de los clientes. El patrón de diseño Monitor Object, posee la clase interface Monitor, la cual proporciona los métodos de sincronización de servicios a cada uno de los clientes.

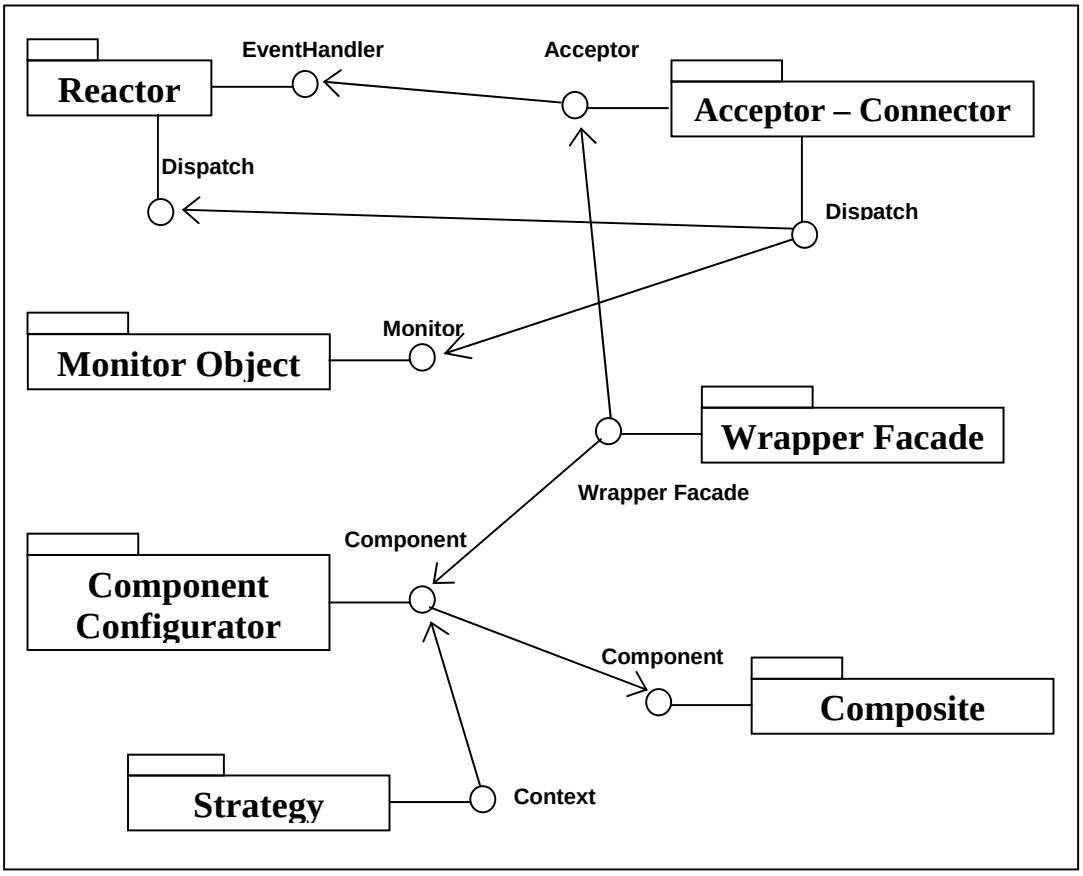

Figura 25: Diagrama de Interface de la Arquitectura de Software Propuesta.

Fuente: Autor de la Investigación

El patrón Wrapper Facade posee una sola interface, la clase interface WrapperFacade, la cual permite encapsular los contenidos una vez que se han estandarizados. El patrón Component Configurator, con su clase interface Component, realiza la configuración del contenido dependiendo de la LMS que lo haya generado, a través de la clase interface Context del patrón de diseño Strategy, que contiene las especificaciones de cada LMS. Por último, el patrón de diseño Composite posee la clase interface Component, que permite la creación del SCO con el contenido estandarizado. De esta manera, se pueden observar cada una de las interfaces que permiten las relaciones entre los diferentes patrones.

En la Figura 26, se muestra un diagrama de clases detallado para la arquitectura de software propuesta, para ello se hizo uso del lenguaje unificado de modelado (UML). En este diagrama se puede observar como interactúan cada uno de los patrones arquitecturales y de diseño, y como intervienen cada una de las clases que los integran. Además, se puede visualizar las relaciones de uso, herencia y dependencia entre las clases que integran cada uno de los patrones.

Para detallar las relaciones existentes en el Figura 26, se puede observar como se relacionan cada uno de los patrones con sus correspondientes clases. Para manejar las solicitudes de evento de conexión, la clase Acceptor del patrón de diseño Acceptor-Connetor hereda de la clase Event Handler del patrón arquitectural Reactor, el método handle_event(), de manera que pueda manejar las solicitudes de las LMS que se conectan e inicializan el proceso de publicación de contenidos.

Ahora, pueden existir varias LMS solicitando el mismo contenido en un mismo momento, lo que hace que la información solicitada se vuelva crítica y se requiera de métodos de sincronización para que no se pierda dicha información, para ello la clase Dispatcher del patrón Acceptor- Connector hereda los métodos de sincronización del patrón de diseño Monitor Object, con lo que se resuelve el primero de los requerimientos del sistema, donde se expresa que se debe atender múltiples solicitudes de los clientes de manera concurrente y sincronizada.

Con esto se logra establecer la conexión entre las LMS de diferentes fabricantes, pero además de conectarse se debe publicar el contenido de la LMS que no posee SCORM, para que pueda ser utilizado por la LMS solicitante, es allí donde actúan los patrones de diseño Wrapper Facade, Component Configurator, Strategy y Composite. Para ello, como se menciona anteriormente se debe primero estandarizar el contenido a publicar para posteriormente crear el objeto de contenido (SCO), que es solicitado por la LMS que si soporta el modelo SCORM.

El proceso de estandarización y publicación de objetos de contenidos (SCO) se realiza de la siguiente manera, el manejador de servicio concreto de la clase Acceptor del patrón Acceptor-Connector depende de los métodos configurados en la clase Wrapper Facade del patrón Wrapper Facade, y estos métodos se definen a través de la clase Component Configurator del patrón Component Configurator, como se observa en la Figura 26. La clase Wrapper Facade depende de la clase Component Configurator, ya que esta le proporciona las características particulares de la LMS no SCORM que es dueña del contenido. Para determinar las características particulares de la LMS no SCORM, la clase Component del patrón de diseño Component Configurator usa la interface Strategy del patrón de diseño Strategy, la cual a través de sus múltiples algoritmos posee la definición de las características particulares de cada LMS no SCORM. Entendiéndose, por LMS no SCORM aquella LMS que no está diseñada basada en el modelo SCORM y posee el contenido a publicar.

En el proceso anterior se obtiene el contenido estandarizado, ahora a través del patrón de diseño Composite se crea el objeto de contenido (SCO) y para ello la clase Component de este patrón de diseño, hereda de la clase Component del patrón de diseño Component Configurator.

De esta manera y con la selección de los patrones descritos anteriormente se da solución al objetivo central de esta investigación, mostrando el diseño de la arquitectura de software que permite publicar contenidos entre diferentes LMS a través de objetos de contenidos intercambiables (SCO). Para ello, la metodología de Yacoub y Ammar (2004) fue de gran ayuda, ya que esta facilitó y simplificó el diseño de una aplicación de software basada en patrones con el desarrollo de cada una de sus etapas. Además, la amplia variedad de patrones reconocidos hace más fácil aún el diseño de la propuesta, ya que con solo entender como funcionan y como se pueden relacionar permite crear una estructura completa, bien diseñada y garantizada que sea la solución requerida.

Una Posible Implementación.

Existen diferentes formas para implementar esta arquitectura, y depende mucho del desarrollador, pero una posible implementación es la siguiente.

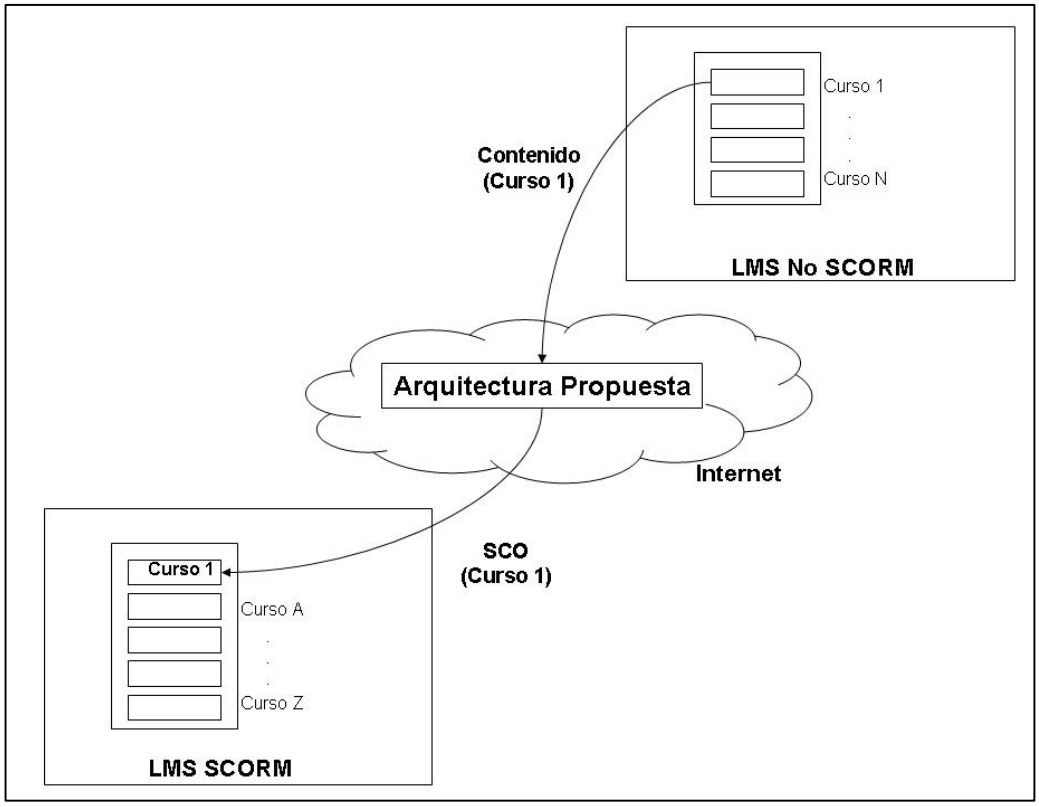

Figura 27: Propuesta de Implementación. Fuente: Autor de la Investigación.

Como se muestra en la Figura 27, la arquitectura propuesta está implementada en un espacio de la Web y puede se accedida por la LMS que contiene el contenido que se quiere publicar. El procedimiento es el siguiente, existen LMS que no poseen el estándar de interoperabilidad y reutilización de recurso, proporcionado por el modelo SCORM, no por ello sus cursos dejan de ser interesantes o relevantes para otras LMS. Pero con la implementación de la arquitectura propuesta, se tiene la posibilidad de que un curso, un módulo o una colección de cursos que son presentados en una LMS

que no posea SCORM sean empaquetados en forma de objetos de contenidos intercambiables (SCO) y puedan ser presentados a usuarios de LMS que si posean el modelo SCORM.

En otras palabras, por ejemplos la plataformas que posee la Universidad Centroccidental “Lisandro Alvarado” (UCLA), llamada “Saber” no está desarrollada con las especificaciones técnicas de SCORM, por lo cual no podría ofrecer sus cursos en líneas, dictado a sus alumnos a otras universidades, que también posean su parte de universidad Virtual y que le interesa algunos de los cursos dictados acá. Es allí, donde entra en juego la arquitectura propuesta, a través de la generación del SCO, del contenido que se desea publicar, ese curso en línea que es presentado a los estudiantes de la UCLA, puede ser llevado a otra universidad que su plataforma de formación si soporte el modelo SCORM, es decir, que posea plataformas como: Angel 6.1, BlackBoard 6, Moodle 1.4, entre otras.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

Conclusiones.

Hoy más que nunca, las Universidades emprenden acciones para refrendar su misión de servicio a la sociedad en un contexto de cambios vertiginosos y profundos. Los alcances, así como el impacto de sus programas sustantivos, se acrecientan conforme se avanza en las actividades de academización, desconcentración, descentralización administrativa, estímulos y vinculación, que conducen a reafirmar su presencia en los distintos sectores de la vida regional, nacional, y en la de otros países.

La propuesta de trabajo de un sistema de educación a distancia (LMS) ofrece a una comunidad que por razones personales y/o académicas encuentran en esta metodología y en su reglamentación, la alternativa para su formación y actualización profesional, con la más alta calidad académica, sus programas de educación profesional, postgrado y educación continua.

Y si aunado a esto, se tiene la posibilidad de importar y exportar cursos entre distintas plataformas de formación (LMS), se hace de la educación a distancia un mundo bastante completo, donde no limitas a los participantes solamente a los cursos ofrecidos por tu plataforma, y sin tener que hacerle cambios a la plataforma puedes ampliar tus ofertas de cursos.

Es allí, donde tiene cabida esta investigación y muestra una arquitectura de software donde sin tener que adaptar la LMS al estándar SCORM, permite crear y publicar cursos para que estén a la disposición de LMS que si lo soporten.

SCORM es el modelo de intercambio de contenidos de aprendizajes, más aceptado a nivel mundial debido a que reúne en un solo procedimiento, la solución al problema de interoperabilidad entre LMS de diferentes fabricantes, permite la distribución de recursos entre plataformas y su reutilización. Por lo tanto, desde el punto de vista de la autora, es el modelo de referencia de intercambio de objetos de aprendizajes, más adecuado para publicar objetos de contenido a partir de una LMS que no lo posea y así puedan ser utilizados por aquellas LMS que si lo posean

La metodología de Yacoub y Ammar (2004) utilizada para llevar a cabo el diseño de la arquitectura, proporciona una herramienta de gran ayuda a la hora de diseñar aplicaciones o componentes de software orientados a la utilización de patrones, es decir, permite tener un horizonte para combinar los requerimientos del sistema con la solución que se quiere aportar.

Además, la gran variedad de patrones existentes y reconocidos a nivel mundial, y cada uno de ellos están muy bien descrito por autores como Gamma (1995), Buschmann (1996), Buschmann y otros (2000), entre otros. Cada uno de ellos señalan, el nombre del patrón, su contexto, la solución que presenta, su implementación y su relación con otros patrones, hacen de está una herramienta de gran ayuda a los diseñadores para conseguir con mayor facilidad el diseño de soluciones complejas sin tener que reinventar la rueda, lo que garantiza la calidad y el éxito de la solución.

Pero si no se cuenta una herramienta de diseño que permita representar en papel, todas esas ventajas ofrecidas por los patrones, difícilmente se podría observar su beneficio. Para ello, se cuenta con el Lenguaje Unificado de Modelado (UML), que

es de gran ayuda, ya que permite de manera estandarizada que los profesionales del área de la Ingeniería de Software se puedan comunicar y transmitir ideas y descubrimientos en el desarrollo de software. Es un lenguaje universal que está bien aceptado a nivel mundial, lo que hace que los profesionales de esta área hablen un mismo lenguaje. Y lo más importante, es que es bastante rico ya que posee una gran variedad de diagramas que permite respaldar y apoyar cada una de las fases del ciclo de vida de un software.

Para finalizar, se puede mencionar que el uso de las herramientas de la Ingeniería de Software, constituyen un elemento importante para llevar a cabo de manera organizada, las actividades con las que se puede conseguir productos de software de calidad y eficiencia.

Recomendaciones.

Tomando en cuenta las conclusiones planteadas, la autora considera conveniente:

- Desarrollar e Implementar la arquitectura de software propuesta en el proyecto de Universidad Virtual existente en la Universidad Centroccidental “Lisandro Alvarado” para ofrecer sus cursos en líneas a otras plataformas que estén desarrolladas con el estándar SCORM.

- Utilizar la metodología presentada por Yacoub y Ammar (2004) para próximas investigaciones en el área de arquitectura de software orientada a patrones, ya que es de gran ayuda y facilita el diseño y desarrollo de la misma.

- Utilizar la filosofía de patrones, ya que presenta una amplia gama de soluciones para el diseño, desarrollo e implementación de soluciones en el área de la Ingeniería de Software.