Universidad de las Ciencias Informáticas

Facultad 3

Análisis y Diseño del módulo de Paginación-Segmentación del subsistema de autoaprendizaje para el Laboratorio de Sistemas Operativos.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor/es: Arisday Ramírez García

Tutor: Ing. Carlos Y. Hidalgo García

Ciudad de la Habana, Cuba

Junio, 2011

Declaración de Autoría

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de diciembre del año 2011

_____

Firma del autor

Arisday Ramírez García

_____

Firma del tutor

(Carlos Y. Hidalgo García)

Agradecimientos

Es largo el sendero recorrido hasta llegar hoy, momento tan especial en mi vida, y así de especial son todas las personas que de una forma u otra han sido participes o inspiradores en toda mi carrera. Comprometo a mi memoria a no dejar de mencionar cada una de estas personas que lo merecen sin duda alguna.

Tienen un lugar insustituible en mi vida, mis padres, a quienes les debo todo lo que soy y lo que pueda ser en mi vida tanto profesional como espiritual.

A mi tata (Olimpia García), por ser la mejor madre del mundo e inspiradora principal de mi vida como profesional, gracias a ella estoy aquí en esta universidad, quien lloró conmigo por estar lejos de mi casa, pero sin dejar de recordarme cada mi minuto que este era mi futuro. Por estar en todo momento difícil y porque sé que nadie más que tu disfruta de mis victorias y alegrías, por hacer de mi la persona que soy hoy, a ti, **MAMÁ**

A mi papa (Gilberto Ramírez), por ser el mejor papá del mundo, por cada minuto de su preocupación por mí, por tantos viajes que tuvo que dar hasta la escuela para tratar de que yo me sintiera bien y continuase mis estudios. Por su interés mostrado en todo este tiempo de superación, preguntándome si era yo la única que me tenía que acostar tarde estudiando o eso le pasaba a todos. Porque sé que soy su orgullo al cumplir esta meta de graduarme, pues como él dice ¨que no sabe a quien yo salí tan inteligente¨. Por darme el privilegio de crecer a su lado, a pesar de no vivir juntos, a ti, **PAPÁ.**

La vida no me dio la oportunidad de un hermano, por eso nací al lado de mi tío, **Alayn Ramírez** quien sin duda alguna ha sido mi cómplice como suelen ser los hermanos. Por sus buenos deseos de que todo salga bien, de que sea una profesional y al fin regrese junto a ellos, a casa.

A mis abuelas, **Edita** y **Emilia**, de las cuales no han faltado su preocupación y bendiciones, de las cuales sé que soy su punto de apoyo y siempre han querido verme triunfar.

A mi bisabuelo **Mario**, quien la vida no le dio la oportunidad de estar hoy aquí, pero sé que estaría muy orgulloso, porque cada día que pudo verme me preguntaba en grado ya estaba.

A mis tíos **Leonardo, Antonio, Tamara y Carmen**, por sus preocupaciones y ayudas en todo este proceso, orgullosos de que yo sea la primera profesional en la familia, yo se que desearían estar aquí.

En fin a toda la **familia** que yo se que sienten este éxito como suyo.

Agradecimientos

No puedo dejar de mencionar a mi **Pituka** (Maureen), mi cómplice más que amiga una hermana, que nunca dejó de darme fuerza y ánimos para continuar, pendiente a pesar de la distancia de todos mis problemas y de todos mis logros y sé que su deseo es estar aquí.

Cuando comencé la universidad comprendí que esta sería una nueva familia, la UCI me ha dado la oportunidad de tener en mi vida a maravillosas personas.

A la primera persona que sin conocerme me brindó su casa a **Oda** (Odaisa Alfonso) mi compañera de estudio de los años y mi amiga, quien me dijo desde el primer día “no llores más que yo te voy a llevar para mi casa” y a partir de ese momento fue mi casa y me convertí en otra hija para sus padres, gracias mamita y papito, los quiero mucho, han sido una familia.

A mi titi Lisbe (Lisbet Sánchez Moreno) la persona más bella que he conocido, quien me enseñó a ver la vida de otra manera, a ver las cosas buenas, por ser una amiga incondicional en cualquier situación, más que una amiga una hermana, porque no se perdió ni una de mis alegrías, pero tampoco ni una de mis tristezas, gracias mi **Bichi,** te adoro.

A mi **Ropy**, amigo de los años, gracias por tantas fuerzas y ánimos, por siempre sacar de mí una sonrisa en los peores momentos. A pesar de tener tanta responsabilidad sobre ti, nunca faltaste, te quiero mucho.

A mis titis del cuarto **Nani, Pattry, Leane**, **Lili, May** gracias por todo, las valoro como mi familia, son las que han estado a mi lado en las buenas y en las malas, he pasado mi mejor año junto a ustedes, me abrieron la puerta de apartamento hasta el final, las quiero muchísimo.

A mis amigos **Irael**, **Randy**, **Carlos** y **Argilagos** siempre aguantaron mi malcriadez y me ayudaron muchísimo, gracias por todo.

Tendría que mencionar todos los grupos por los que he pasado, amigas y amigos con los que convivido, gracias.

A mi **machi**, que no por ser el ultimo es menos importante (todo lo contrario), desde que entre en la universidad ha sido mi compañero gracias por tu ayuda en este trayecto, por siempre estar pendiente a todo. A sus padres, que me abrieron la puerta de su casa y siempre me apoyaron.

A todos los profesores que siempre me ayudaron tanto. Al tribunal y tutor que colaboraron a que hoy fuera posible estar aquí, en fin a la UCI.

Agradecimientos

Gracias a Dios por ponerme en mi camino a todas estas personas que forman parte de mi vida.

GRACIAS.

Dedicatoria

Dedico este Trabajo a mis padres Olimpia y Gilberto, porque soy la luz de sus días, porque les debo todo lo que soy, por ser mis guías en todo este camino de ser una mejor persona y porque simplemente son mi vida.

Resumen

Resumen:

La vinculación de las Tecnologías de la Informática y las Comunicaciones (TIC) en el proceso de formación de la educación cubana tiene sus mayores avances en la actualidad, con el desarrollo de software educativos, específicamente en la Universidad de las Ciencias Informáticas (UCI). La asignatura de Sistema Operativo (SO) es fundamental en el plan de estudio de la carrera, en este sentido se observa que carece de una fuerte interacción con las TIC en su proceso de aprendizaje, por lo que se hizo necesario el desarrollo de un Laboratorio Virtual de apoyo a la asignatura, así se concibió la necesidad de desarrollar el Análisis y Diseño del modulo de Paginación y Segmentación del subsistema Autoaprendizaje para el Laboratorio Virtual.

En un primer momento se refleja en el trabajo un estudio del arte referente a los laboratorios virtuales, componentes de ayuda para la enseñanza, metodología de desarrollo y lenguajes de modelado. Se muestra un estudio del lenguaje de modelado que se utiliza para aplicaciones educativas, ApEM-L.

En un segundo momento se realiza la construcción de un mapa conceptual, representando el basamento teórico de la investigación, haciendo uso de las herramientas manipuladas en el progreso del trabajo. Siguiendo los requisitos identificados durante las etapas de Ingeniería de Requisitos, se realiza posteriormente todos los artefactos correspondientes al análisis y diseño para el módulo de Paginado y Segmentado del Laboratorio Virtual. Para la aprobación de los artefactos generados se aplican diferentes métricas de validación para el análisis y para el diseño, demostrando la calidad alcanzada en el trabajo realizado.

Palabras Claves: Software Educativo, Laboratorio Virtual, Sistema Operativo, Mapa Conceptual, Lenguaje de Modelado, ApEM-L, Métrica.

Índice de Contenido

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

Introducción.

La informática es una de las ciencias que se encuentra en constante evolución y está presente en la mayoría de los sectores sociales del mundo. Cuba, desde hace algunos años, se ha unido al desarrollo informático mundial. Para alcanzarlo se han trazado en el país diversas estrategias que ayudan a elevar la cultura y conocimiento de esta ciencia. Algunas de ellas son la creación de los Joven Club de Computación y el surgimiento de una universidad de nuevo tipo llamada UCI en la cual se destaca el amplio uso de las TIC en el proceso de formación, por lo que se concibe un nuevo modelo de formación enmarcado en la calidad del aprendizaje, estableciendo un ciclo de formación básico y otro profesional, donde tiene un uso protagónico los Entornos Virtuales de Aprendizaje (EVA), utilizando el Moodle como plataforma de Autoaprendizaje compuesta por los diferentes recursos didáctico y dentro de estos los Objetos de Aprendizajes.

La asignatura SO es fundamental dentro del plan de estudio del 3er año de la carrera de Ingeniería en Ciencias Informáticas, la cual está dividida en tres temas fundamentales que comprenden los principales aspectos referentes al diseño y construcción de los mismos como son: proceso, memoria y entrada / salida de información. En este sentido se observa que la asignatura en la actualidad carece de una fuerte integración con las TIC, lo que ha llevado a que las actividades sean en su mayoría presenciales.

Como parte del perfeccionamiento de la asignatura, se está desarrollando un Laboratorio Virtual del cual se han implementado los simuladores de Proceso y Memoria. Siendo aún necesaria la orientación del profesor para que el estudiante realice las actividades con dichos simuladores, al no contar con un Módulo de autoaprendizaje que le permita al estudiante el acceso a todo el contenido teórico y seguidamente hacer uso de los simuladores. Bajo estas condiciones surge la necesidad de realizar el análisis y diseño para lograr un correcto desarrollo del módulo de Paginación y Segmentación del laboratorio virtual de la asignatura. Para lograra crear un entendimiento común entre clientes y desarrolladores con el fin de identificar las necesidades reales del cliente y traducirlas en modelos, de manera que queden representados claramente qué es lo que debe y no debe hacer el sistema, propiciando una posterior implementación del módulo.

Bajo estas condiciones es identificado el siguiente **problema de la investigación**: ¿Cómo lograr un entendimiento entre clientes y desarrolladores del módulo de Paginado y Segmentado para el subsistema

de autoaprendizaje del Laboratorio Virtual de SO que permita la posterior implementación del mismo? Este problema se enmarca en el **objeto de estudio**: Proceso de Desarrollo de Software.

En este sentido la investigación que se presenta tiene como **objetivo general**: Realizar el Análisis y Diseño del Módulo de Paginado y Segmentado para el subsistema de autoaprendizaje del Laboratorio Virtual de SO, que permita el desarrollo posterior de dicho módulo y cuyo **campo de acción** es: El Análisis y Diseño dentro del proceso de desarrollo de software para aplicaciones educativas.

Se define como **idea a defender:** Realizando el Análisis y Diseño del módulo de Paginado y Segmentado para el subsistema de autoaprendizaje del Laboratorio Virtual de SO se logrará implementar el subsistema de autoaprendizaje.

Para el cumplimiento exitoso del objetivo general se han definido como **objetivos específicos:**

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Obtener los artefactos generados en la etapa de análisis y diseño.
- ✓ Validar la solución propuesta.

Métodos de investigación:

Métodos teóricos:

• Análisis Histórico – Lógico: Para profundizar en los antecedentes de la utilización de las Tecnologías de la Información y las Comunicaciones en los procesos de enseñanza – aprendizaje y sus tendencias actuales.

• Modelación teórica: Se utiliza para representar las características y relaciones fundamentales del objeto, proporcionar explicaciones y servir como guía para la comprensión del fenómeno que se desea transformar.

• Enfoque de sistema o sistémico: Para el estudio del proceso de enseñanza – aprendizaje utilizando los Objetos de Aprendizaje y sus componentes didácticos; así como para la orientación general, al abordar la investigación sobre el uso de las TIC en la enseñanza de la asignatura de SO y la modelación de la concepción didáctica.

• Analítico - Sintético: Se utiliza en la revisión bibliográfica, el estudio de reportes e informes sobre el estado de la infraestructura tecnológica de la UCI, la consulta de documentos rectores de la política de la UCI sobre la Informatización, la tendencia actual al aprendizaje auto gestionado y la introducción de las TIC en el proceso de enseñanza-aprendizaje.

Métodos empíricos:

• Entrevista: Para la recopilación de información especializada o dirigida a directivos, profesores y alumnos que interactúan con las TIC en el proceso de enseñanza – aprendizaje presencial o mixto de la asignatura de SO en la Facultad 3 de la Universidad de las Ciencias Informáticas.

El presente trabajo está conformado por 3 capítulos.

El Capítulo 1. Fundamentación Teórica. Queda expuesto el conocimiento teórico necesario para el desarrollo del mismo, se abordan los elementos relacionados con los sistemas de autoaprendizaje, en esencia los Laboratorios Virtuales. Se refleja un estudio de las diferentes metodologías de desarrollo y lenguajes de modelado y una amplia descripción del lenguaje de modelado utilizado en el trabajo, así como las herramientas involucradas.

El Capítulo 2. Descripción de la Solución. Está conformado por la descripción de la solución propuesta, mediante los diferentes diagramas de diseño generados, guiados por el lenguaje de modelado ApEM-L, mostrando los requisitos funcionales y no funcionales identificados, el diagrama de Caso de Uso, y sus descripciones textuales correspondientes. Conformando todo lo necesario para el diseño del subsistema.

El Capítulo 3. Validación. En este capítulo se aborda lo referente al uso de las diferentes métricas de validación, para demostrar la alta calidez del trabajo. Se utilizan las métricas de validación para la especificación de requisitos, obteniendo un alto grado de especificación. Se utiliza la métrica de calidad de casos de uso para la validación del diagrama de caso de uso. Para la validación del diseño se utilizan las métricas de Tamaño Operacional de Clases y Relaciones entre Clases.

Capítulo 1. Fundamentación Teórica.

1.1 Introducción.

En el presente capítulo se brinda el marco teórico en el que se desarrollará el trabajo, el cual ofrece un fundamento sólido a la solución que propone. Es de vital importancia realizar este estudio preliminar para enmarcar la investigación y arrojar los primeros resultados. Se presenta un estudio de los laboratorios virtuales existentes y de las herramientas utilizadas con el fin de justificar las variantes utilizadas, conjuntamente a las metodologías y lenguaje de modelado. Posteriormente todo lo referido a la Ingeniería de Requisito en cuanto a las etapas de extracción, especificación, análisis y validación de los requisitos y seguidamente la Gestión de Requisitos.

1.2 Sistemas de Autoaprendizaje.

Con el continuo desarrollo de la educación empleando la tecnología, tiene aparición entonces las distintas metodologías y estrategias de aprendizaje que emplean tecnología digital o informática para la interacción del conocimiento. La combinación de la educación con la informática da lugar al surgimiento de la Informática Educativa (E-Learnig), que según la definición de la Comisión Europea es “la utilización de las nuevas tecnologías de multimedia e Internet para mejorar la calidad del aprendizaje facilitando el acceso a recursos y servicios y vinculando entonces el autoaprendizaje al proceso de formación.

El Autoaprendizaje es la habilidad que cada persona posee para dirigir y regular la adquisición del conocimiento a través del estudio de diversos contenidos o de la experiencia en actividades de aprendizaje. Es la forma de aprender a aprender por uno mismo. (Álvarez, 2007)

Ante el desarrollo que existe hoy en día en el proceso de formación donde el uso de las TIC se hace intenso, las universidades deben ir más allá de capacitar a sus estudiantes y docentes en el uso de las nuevas tecnologías, para que puedan desenvolverse en ambientes de aprendizaje autónomo y así dar al alumno la oportunidad de utilizar distintos medios de acceso a la información. (Graells, 2000)

Al conocer estas realidades globales, se asume el reto de diseñar y poner en marcha programas institucionales dirigidos a la formación del profesional del siglo XXI, capaz de desempeñarse en el nuevo entorno mediante la acogida de todas las metodologías y herramientas que le permitan llevar a cabo el autoaprendizaje, donde lo más importante ya no es el conocimiento sino la capacidad para adquirirlo, interpretarlo y utilizarlo. (Londoño, 2004)

Por estas razones surgen los sistemas de autoaprendizaje, no son más que programas informáticos que combinan libros impresos o electrónicos, laboratorio virtual interactivo, laboratorio real (entrenador) y sistema de autoevaluación basado en computador. Dentro de los sistemas de autoaprendizaje tienen lugar los laboratorios virtuales.

1.3 Laboratorios Virtuales.

El proceso educativo de la actualidad se ha visto influenciado por dos fenómenos, por un lado, la introducción progresiva y significativa de nuevos medios informáticos, equipos y recursos audiovisuales, por otra parte, la incorporación de las TIC en la mayoría de los sectores sociales, lo que permite la introducción de recientes técnicas y medios docentes. Por eso, en esta era de la informática, han sido precisamente los laboratorios virtuales herramientas protagonistas en el proceso de aprendizaje, por lo que constituye una valiosa herramienta para resolver el problema existente en la asignatura de SO en la carrera de Ingeniería en Ciencias Informática.

Los primeros laboratorios virtuales empezaron a desarrollarse en 1997 en el Centro de Investigación Académica de la Universidad Estatal a Distancia de Costa Rica y estos fueron los primeros para la enseñanza a distancia a nivel mundial.

Muchos han sido los autores que han dado su definición de lo que es un Laboratorio Virtual, en lo adelante se citan algunos ejemplos.

Según James P. Vary, un laboratorio virtual es un “espacio electrónico de trabajo concebido para la colaboración y la experimentación a distancia, con el objetivo de investigar o realizar otras actividades creativas, y elaborar y difundir resultados mediante tecnologías difundidas de información y comunicación” (Vary, 2000).

Julián Monge Nájera define un laboratorio virtual como “simulaciones de prácticas manipulativas que pueden ser hechas por el estudiante lejos de la universidad y el docente” (Estrada, 2007).

Después de estudiar las definiciones dada por varios autores, se define que un Laboratorio Virtual no es más que un entorno de simulación y experimentación a distancia para realizar el conjunto de prácticas de laboratorio que son necesarias como apoyo al proceso enseñanza-aprendizaje (Urrutia, 2010).

1.3.1 Tipos de Laboratorios Virtuales.

Existen tres tipos de laboratorio virtual:

1. **Laboratorios virtuales software.**

Los laboratorios virtuales de software, son laboratorios desarrollados como un programa de software independiente destinado a ejecutarse en la máquina del usuario, y cuyo servicio no requiere de un servidor Web. (Herías, 2003)

2. **Laboratorios virtuales Web.**

En contraste con el anterior, este tipo de laboratorio se basa en un software que depende de los recursos de un servidor determinado. Estos recursos pueden ser bases de datos, software que requieren ejecutarse en su servidor o la exigencia de determinado hardware para ejecutarse. No son programas que un usuario pueda descargar en su equipo para ejecutar localmente de forma independiente (Herías, 2003).

3. **Laboratorios remotos.**

Los laboratorios remotos son aquellos que permiten operar remotamente cierto equipamiento, bien sea didáctico como maquetas específicas o industrial, además de poder ofrecer capacidades de laboratorio virtual. En general, estos requieren de equipos servidores específicos que les den acceso a las máquinas a operar de forma remota, y no pueden ofrecer su funcionalidad ejecutándose de forma local. (Herías, 2003)

1.3.1.1 Ventajas y Desventajas de los Laboratorios Virtuales.

El uso de los Laboratorios Virtuales tiene sin dudas muchos beneficios para brindar a los usuarios, a continuación se hace mención de alguna de las ventajas que presentan:

- Simula situaciones que en la realidad tendrían escasas posibilidades de realizarlas.
- Se convierten en una ayuda interactiva para el aprendizaje de contenidos difíciles de demostrar en la realidad.
- La simulación en el laboratorio virtual, permite obtener una visión más intuitiva de aquellos fenómenos que en su realización manual no aportan suficiente claridad gráfica.
- Es una herramienta de auto aprendizaje.

- Permite la autoformación reflexiva de los alumnos mediante su trabajo individual, bien como aclaración y complemento de laboratorios experimentales, bien como prácticas.
- Al docente le permite centrarse en la explicación de los fundamentos básicos, disminuyendo el tiempo que el profesor dedica actualmente a la introducción del modo de operación y funcionamiento de los instrumentos.
- Permite una evaluación personalizada y continuada del progreso de aprendizaje del alumno por el profesor o por el propio alumno. De este modo el profesor podrá incidir en los aspectos que crea necesario que cada alumno en particular deba trabajar. (Estrada, 2007)

Acompañado a estos beneficios también se presentan algunos inconvenientes con los cuales se corre el riesgo que el alumno se comporte como un simple espectador. Es preciso que el estudiante realice una actividad ordenada y progresiva, conveniente a alcanzar objetivos básicos concretos, además el alumno no utiliza elementos reales en el laboratorio virtual, lo que provoca una pérdida parcial hacia la visión de la realidad y los resultados son menos atractivos para los estudiantes. (Herreros, 2005.)

Se realizó un estudio sobre los laboratorios que existen con el objetivo de conocer las funciones y modos de trabajo de estos, a continuación se menciona algunos de ellos.

- **Entorno virtual de Física,** realizado en la Universidad de las Ciencias Informáticas con el cual se puede trabajar en la asignatura de Física I, consta con 8 prácticas de laboratorios para el estudio de algunos temas de la asignatura.
- **Sistema Interactivo Didáctico de Enseñanza de la Física (SIDEF),** utilizado en el Departamento de Física de la Facultad de Matemática, Física y Computación de la Universidad Central "Marta Abreu" de Las Villas.
- **Laboratorio Virtual de Química,** la Universidad de Villa Clara “Martha Abreu” también cuenta con un laboratorio virtual de Química, el cual a medida que el estudiante va trabajando y avanzando le muestra los conocimientos que ha ido alcanzando.
- **Laboratorio Virtual de Química General,** de la Universidad de La Habana con el cual se puede aprender Química desde su investigación hasta realizar prácticas virtuales.

- **Laboratorio virtual en Anestesiología,** utilizado en Hospital General Docente "Aleida Fernández Chardiet" Güines, La Habana, donde sirve de apoyo al aprendizaje y la investigación de técnicos y profesionales en el campo de las Ciencias Médicas del territorio.

Posterior al estudio realizado de los laboratorios virtuales, el cual permite conocer las funciones que brinda este tipo de software y su modo de uso, se tiene conocimiento que no existe un laboratorio virtual vinculado a la materia de SO, por lo que surge la necesidad de desarrollar un laboratorio virtual para la asignatura que permita al estudiante el acceso al contenido teórico de manera autónoma, donde realice las diferentes prácticas de simulación necesaria y las evaluaciones correspondientes. (Estrada, 2007)

Al estar la universidad en función del desarrollo informático en todas las esferas, se decide utilizar el laboratorio virtual web, pues le permite al usuario acceder desde cualquier computadora al servidor donde esté la práctica virtual y además podrán acceder a él varios usuarios simultáneamente. En la asignatura SO se están dando los primeros pasos en la realización de un laboratorio virtual, el mismo se va a incorporar en el EVA y tendrá integrado el trabajo de diploma realizado, facilitando que el profesional y el estudiante relacionado a este tema pueda hacer uso del mismo.

Los laboratorios virtuales están compuestos por diferentes recursos didácticos, los cuales engloban a su vez entidades de información denominadas Objetos de Aprendizajes.

1.4 Recurso Didáctico.

Un *Recurso Didáctico* es definido como cualquier material que se elabore con la intención de facilitar al docente su función, comprende cualquier recurso de apoyo al aprendizaje, desde un libro hasta una imagen o video. Estos tienen el objetivo de proporcionar y organizar la información transmitida, ayudando a ejercitar y desarrollar habilidades como el autoaprendizaje, ya que despiertan la motivación y crean un interés hacia el contenido del mismo. Por esta razón se engloba dentro de los recursos didácticos diferentes entidades de información que son denominadas *Objetos de Aprendizajes*. (Yenes, 2007)

Diversos autores coinciden en señalar que en las dos últimas décadas la sociedad ha experimentado su más grande desarrollo a partir de la influencia de las llamadas TIC por lo que investigadores como Castells, denomina a nuestra sociedad actual como la sociedad basada en la información o sociedad-red. De hecho, en los últimos cinco años ha cobrado auge una tendencia en el campo de la tecnología

educativa relacionada con el desarrollo e incorporación de estas entidades de información (Castells., 2000).

Diversas son las definiciones sobre Objetos de Aprendizajes por diferentes autores, en los años 1992-96 es utilizado este término por Wayne Hodgins, el cual crea un grupo de investigación llamado Arquitectura de Aprendizaje (Learning Architecture) y Objetos de Aprendizajes (Learning Objects) en 1994.

Según plantea Merrill, un Objeto de Aprendizaje es un objeto mediático, conjunto de bits de texto, gráficos, video o audio; al cual se le integra una estrategia instruccional. Un objeto de aprendizaje es una unidad mínima de información, digital o no digital, que puede ser re-usada y secuenciada junto con otros objetos de aprendizaje para conformar cursos o unidades que abarquen objetivos de aprendizaje más amplios (David, 2002).

En el año 1996, IEEE crea LTSC (Learning Technology Standards Committee) que en el momento de su creación adopta el término objeto de aprendizaje estableciendo la siguiente definición:

“Cualquier entidad, digital o no digital, que pueda ser utilizada, reutilizada o referenciada durante un proceso de aprendizaje mediado por la tecnología”, ccontenidos de la multimedia, contenidos didácticos, objetivos de aprendizaje, software didáctico y herramientas de software y personas, organizaciones o eventos. (Urrutia, 2010)

En este trabajo se adopta la definición de objeto de aprendizaje dada por la IEEE. Los objetos de aprendizaje son considerados una herramienta educativa muy importante que pueden insertarse en propuestas curriculares y metodologías de enseñanza-aprendizaje.

El concepto de objeto de aprendizaje es visto como un módulo de información con una temática bien definida y una intención de aprendizaje o como un material de apoyo al aprendizaje para que el alumno interactúe con él. Este puede incluir multimedia, instrucciones, programas informáticos de instrucción, todo esto dentro de un aprendizaje soportado con tecnología.

1.4.1 Diseño de Aplicaciones.

Para el desarrollo de cualquier proceso se necesita realizar la modelación del mismo. El proceso didáctico que se desea llevar a cabo, para que los estudiantes reciban contenidos importantes; permite reutilizar objetos de aprendizaje u otro tipo de recurso educativo. El proceso de diseño tiene dos fases esenciales que se encuentran entrelazadas.

- Fase pedagógica: Diseño de la unidad didáctica.
- Fase tecnológica: Se estandariza la unidad como objeto de aprendizaje SCORM. (Medina, 1989)

Para dar cumplimiento a la primera fase se modela el dominio mediante un Mapa conceptual considerado como una herramienta didáctica de creación del aprendizaje ejecutándose de forma secuencial mediante instrucciones.

Los mapas conceptuales son artefactos para la organización y representación del conocimiento. Su objetivo es representar relaciones entre conceptos en forma de proposiciones. Se estructuran en forma jerárquica en la que los conceptos más generales están en la raíz del árbol y a medida que se desciende por el mismo se van encontrando con conceptos más específicos. (Cuevas, 2004)

Para realizar el esquema es necesaria la utilización de una herramienta capaz de representar estas relaciones de la manera más cómoda y entendible posible. Dentro de las más destacadas están:

FreeMind: Es una herramienta que permite la elaboración de mapas mentales o conceptuales. Es útil en el análisis y recopilación de información o ideas generadas en grupos de trabajo. Es la alternativa libre a la aplicación MindManager fabricada por la empresa MindJet. Su principal limitación es que se centra en un único tipo de diagramas, los mapas mentales.

CmapTools: Se trata de una herramienta gratuita para entidades educativas sin ánimo de lucro que permita la construcción de mapas conceptuales. Desarrollada por el Institute for Human and Machine Cognition de la University of West Florida. Su orientación era la captura y representación del conocimiento. (Medina, 1989)

Los mapas conceptuales iniciaron su desarrollo en el Departamento de Educación de la Universidad de Cornell, EUA, durante la década de los setenta y constituye una respuesta a la teoría del aprendizaje significativo desarrollada por Ausubel, en especial, en lo referente a la evolución de las ideas previas que poseen los estudiantes para lograr un nuevo conocimiento. Los mapas conceptuales han constituido desde entonces una herramienta de gran utilidad para profesores e investigadores de temas educativos, psicólogos, sociólogos y estudiantes en general, así como para otras áreas sobre todo cuando se necesita tratar con grandes volúmenes de información (Cuevas, 2004).

Se han publicado diferentes criterios sobre el concepto de los mapas conceptuales, uno de ellos precisamente definido por Novak, su creador, publicado en su texto “Aprendiendo a aprender”, define el

mapa conceptual como una técnica que representa, simultáneamente, una estrategia de aprendizaje, un método para captar lo más significativo de un tema y un recurso esquemático para representar un conjunto de significados conceptuales, incluidos en una estructura de proposiciones. (Cuevas, 2004)

1.5 Ingeniería de Requisitos.

El término de Ingeniería de Requisito (IR), surge por la necesidad de englobar el proceso de desarrollo y gestión de los requisitos durante todo el ciclo de vida de un producto, debido a que el interés por este proceso ha aumentado considerablemente, dado los grandes problemas que trae consigo no realizar una correcta definición de los requisitos del software. Varios autores como Pressman, definen la IR de diferente modo.

"La IR es el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener con un coste reducido el análisis, documentación y evolución continua de las necesidades del usuario y la especificación del comportamiento externo de un sistema que satisfaga las necesidades del usuario" (Sánchez, 2005).

"La IR ayuda a los ingenieros de Software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software" (Pressman, 2005).

Según Napier, "La IR cubre todos los aspectos del descubrimiento, documentación y mantenimiento de los requisitos durante todo el ciclo de vida del software de desarrollo" (Napier y Mathiassen, 2009).

En este sentido se adopta la definición para la IR apoyada en los conceptos de Pressman y Napier ya que el objetivo es realizar todas las actividades que comprenden las diferentes etapas dentro de la IR, como la extracción, análisis, especificación, validación y Gestión de Requisitos del software.

Para el desarrollo de software son de gran importancia los requisitos del mismo, de manera general estos son los que especifican cómo el sistema debe funcionar sin decir cómo debe hacerlo. Expresa claramente lo que el cliente desea obtener y constituyen una especificación de lo que debería ser implementado.

Capítulo 1: Fundamentación Teórica

Existen diversas definiciones según varios autores como Pressman y la IEEE.

Por el autor Roger S. Pressman en el libro de Ingeniería de Software "Un enfoque práctico", "los requisitos comprenden necesidades de información y control, funcionalidad del producto y comportamiento, rendimiento general del producto, diseño, restricciones de la interfaz y otras necesidades especiales (Pressman, 2005).

"La IEEE-Standard Glossary of Software Engineering Terminology, define un requisito como:

- Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
- Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
- Una representación documentada de una condición o capacidad". (Urrutia, 2010)

En la presente investigación se tomará como definición de requisitos la planteada por la IEEE, ya que se expresa de manera amplia y clara los objetivos de los requisitos.

Para realizar la IR existen actividades y técnicas para obtener las necesidades del cliente, que ayudan a entender mejor el problema. Esta se divide en dos etapas, la de desarrollo de requisitos y la Gestión de Requisitos. La primera mencionada está compuesta por las siguientes actividades:

1.5.1 Extracción de Requisitos.

La etapa de extracción de requisitos es de vital importancia para el desarrollo de un software ya que es el momento inicial donde el equipo de desarrollo se encarga de preguntarle al cliente el problema a resolver, los diversos servicios que debe brindar el sistema y las restricciones. Pero no es tan simple ya que se pueden presentar problemas como: alcance, que esto no es más que el límite del sistema esté mal definido, de comprensión, que los clientes no definen claramente qué es lo que necesitan en realidad y de volatilidad que los requisitos cambian con el tiempo (Pressman, 2005).

Por lo que la extracción de los requisitos del sistema no solo implica preguntar al cliente qué es lo que desea, sino que es un proceso exhaustivo que involucra conocer con detalle el dominio de la aplicación, comprender con exactitud el negocio y entender los procesos de trabajo que el sistema apoyará. Para todo este proceso existen técnicas que ayudan a que el producto satisfaga las necesidades del cliente.

Capítulo 1: Fundamentación Teórica

Se presenta un grupo de técnicas que son utilizadas para el proceso de obtención de requisitos:

- **Cuestionarios y lista de verificación**: esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario. Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista.
- **JAD** (Joint Application Development/Desarrollo conjunto de aplicaciones): esta técnica resulta una alternativa a las entrevistas. Es una práctica de grupo que se desarrolla durante varios días y en la que participan analistas, usuarios, administradores del sistema y clientes. Está basada en cuatro principios fundamentales: dinámica de grupo, el uso de ayudas visuales para mejorar la comunicación, mantener un proceso organizado y racional y una filosofía de documentación WYSIWYG (What You See Is What You Get, lo que ve es lo que obtiene), es decir, durante la entrevista se trabajará sobre lo que se generará. Tras una fase de preparación del JAD al caso concreto, el equipo de trabajo se reúne en varias sesiones. En cada una de ellas se establecen los requisitos de alto nivel a trabajar, el ámbito del problema y la documentación. Durante la sesión se discute en grupo sobre estos temas llegándose a una serie de conclusiones que se documentan. En cada sesión se van concretando más las necesidades del sistema
- **Entrevistas**: resulta una técnica muy aceptada dentro de la IR y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. La entrevista, sin embargo, no es una técnica sencilla de aplicar, esta requiere que el entrevistador sea experimentado y tenga capacidad para elegir correctamente a los entrevistados y obtener de ellos toda la información posible en un período de tiempo siempre limitado.
- **Tormenta de ideas:** es también una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas e información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numerosa (máximo 10 personas). Como técnica de captura de requisitos es sencilla de usar y de aplicar. Además, suele ofrecer una visión general de las necesidades del sistema, pero

normalmente no sirve para obtener detalles concretos del sistema, por lo que suele aplicarse en los primeros encuentros.

- **Sketches y Storyboards** (interfaces y estructura de navegación): esta técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno web. La misma consiste en representar sobre papel en forma muy esquemática las diferentes interfaces al usuario (sketches). Estos sketches pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación (storyboard).
- **Comparación de terminología:** uno de los problemas que surge durante la extracción de requisitos, es que usuarios y analistas no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada para obtener un consenso respecto a la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos. (Koch, 2004).

Para realizar esta etapa de la Ingeniería de Requisitos, se seleccionó mediante una guía propuesta para la IR en aplicaciones educativas las técnicas de extracción de requisitos: entrevistas y tormenta de ideas, ya que en este caso se puede contactar en todo momento con el cliente lo que hizo posible numerosas reuniones y trabajos colectivos para un mayor cúmulo de ideas.

1.5.2 Análisis de Requisitos.

En esta etapa de análisis luego de obtener los requisitos, estos se agruparon por categorías y se organizaron en subconjuntos. Se estudia cada uno de ellos, se examinan en su consistencia, complejidad y ambigüedad y se clasifican en base a las necesidades de los clientes. Se recomienda que debe realizarse un análisis íntegro de los requisitos teniendo en cuenta que aquí es donde se conceptúan, investigan y resaltan los problemas que puedan presentarse. Luego de obtener los requisitos se agrupan y se priorizan según el siguiente criterio:

Crítico: se refiere a las funcionalidades básicas del sistema.

Importante: son los que brindan soporte a las funcionalidades básicas del sistema.

Útil: características que le ofrecen comodidad al sistema (tecnología multimedia). (González, 2010)

1.5.3 Especificación de requisitos.

En esta etapa se documentan los requisitos, se especifica el análisis realizado anteriormente y se aplican técnicas y estándares de documentación. Con el objetivo de realizar la definición, análisis y verificación de los requisitos del sistema de una aplicación. Se utilizan diferentes técnicas como:

- **Glosario**: la diversidad de personas que forman parte de un proyecto de software hace que sea necesario establecer un marco de terminología común. Por esta razón son muchas las propuestas que abogan por desarrollar un glosario de términos en el que se recojan y definan los conceptos más relevantes y críticos para el sistema.
- **Lenguaje natural**: Consiste en definir los requisitos en lenguaje natural sin usar reglas para ello, con el fin de que los usuarios puedan comprender sin tener conocimientos técnicos.
- **Plantillas o patrones:** esta técnica tiene por objetivo el describir los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumpliendo usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información. Cuanto más estructurada sea esta, menos ambigüedad ofrece.
- **Escenarios:** la técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La representación del escenario puede variar dependiendo del autor. Esta representación puede ser casi textual o ir encaminada hacia representaciones gráficas en forma de diagramas de flujo. (González, 2010)

Luego del estudio realizado de las diferentes técnicas que se utilizan en esta etapa de especificación de requisitos para el desarrollo de presente trabajo se seleccionó la técnica de Lenguaje Natural para un mejor entendimiento con el cliente.

1.5.4 Validación de requisitos.

La validación de la IR, tiene como objetivo examinar las especificaciones para verificar que los requisitos del sistema han sido establecidos sin ambigüedad, inconsistencias u omisiones, que los errores detectados han sido corregidos y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso del proyecto y del producto. Es uno de los procesos más importantes y de no realizarse se corre el riesgo de implementar una mala especificación, y el costo que conlleva es muy elevado (Pressman, 2005).

Existen diferentes técnicas para realizar esta etapa de validación de requisitos que se presentan a continuación:

- **Reviews** (Revisión): esta técnica consiste en la lectura y corrección completa de la documentación o modelado de la definición de requisitos, con el objetivo de demostrar que la especificación de requisitos realizada define realmente lo que el cliente necesita que haga el sistema.
- **Auditorías:** la revisión de la documentación con esta técnica consiste en verificar los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso.
- **Prototipos:** algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final. (González, 2010)

Se utiliza como técnica de validación de requisitos para este tipo de aplicaciones la técnica Revisión, con el objetivo que quede bien definido las necesidades del cliente.

1.5.5 Gestión de Requisitos.

La Gestión de Requisitos no es más que darle un seguimiento a los requisitos obtenidos para detectar cualquier anomalía que se pueda presentar durante todo el proceso de ejecución del proyecto, ya que estos pueden ser modificados durante todo el ciclo de vida del software.

Existen definiciones de varios autores que se presentan a continuación:

Según Bárbara A. McDonald Landazuri de la Facultad de Informática de la Universidad Politécnica de Madrid Gestión de Requisitos es el tratamiento y control de las actualizaciones y cambios a los mismos. (Landazuri, 2005)

Pressman define la Gestión de Requisitos de la siguiente manera: es un conjunto de actividades que ayudan al equipo de trabajo a identificar, controlar y seguir los requisitos y sus cambios en cualquier momento (Pressman, 2005).

Agusti plantea el siguiente concepto de GR: contribuye al éxito de los proyectos de software, al posibilitar un entendimiento común entre el cliente y el grupo de desarrolladores de los requisitos del cliente que deben

concebirse en el producto final, la comprensión de los problemas que se necesitan solucionar y las posibles vías de resolverlos. (Agust, 2008)

Según los conceptos analizados anteriormente se llega a la conclusión que la Gestión de Requisitos es la forma de establecer un seguimiento y control de las actualizaciones y cambios de los requisitos, es de vital importancia para el desarrollo de software y para ello existen distintas herramientas para realizar la Gestión de Requisitos como:

- **RequisitePro:** es una herramienta potente, fácil de usar e integrada de administración de requisitos que promueve una mejor comunicación. Permite al equipo crear y compartir sus requisitos utilizando métodos basados en documentos potenciados por la aplicación de las capacidades de una base de datos, tales como la trazabilidad y el análisis de impacto (INNOVA, 2007).
- **IRqA:** es una herramienta especialmente diseñada para soportar el proceso completo de IR. En IRqA el ciclo de especificación completo incluye la captura de requisitos, análisis, especificación de sistema, validación y la organización de requisitos es soportada por modelos estándares (Malumbres, 2007).
- **OSRMT:** es una herramienta de software libre, el nombre completo es Herramienta para la Gestión de Requisitos de Código Abierto (Open Source Requeriment Management Tool). Fue diseñada para servir el ciclo de vida completo del desarrollo del software. Es independiente de la plataforma que se vaya a utilizar. Permite la descripción de los requisitos, además de los documentos relacionados con la IR. También comprende las distintas matrices de trazabilidad, funcionalidades, casos de usos y casos de pruebas (Pescador, 2009).

  Para esta etapa de Gestión de Requisitos se definen un conjunto de actividades, estas son:

  Priorizar requisitos: para determinar aquellos que deben cumplir en la primera versión del producto y aquellos que pueden llevarse a cabo en sucesivas versiones.

  Gestión de cambios: es vital gestionar estos cambios de forma efectiva y eficiente. Es adecuado que el documento de requisitos que se esté elaborando, previo a entrar a la línea base, esté sometido a un procedimiento de control de cambios para poder distinguir las versiones iniciales de las versiones aprobadas.

Realizar la trazabilidad: es necesario tener buenas técnicas para separar y especificar correctamente los requisitos, controlar su evolución y soportar los cambios. La trazabilidad es el mecanismo que permite lograr este resultado. (González, 2010)

Para realizar la Gestión de Requisitos se seleccionó la herramienta OSRMT ya que esta permite disminuir el trabajo de los analistas para la gestión de cambios y así poder tener requisitos con mejor calidad, es una herramienta multiplataforma, independiente del SO, posee un código completamente abierto, además de ser la herramienta propuesta en el proceso de mejoras de la UCI. Por los estudios realizados se concluye que las herramientas IRQA y Requisite Pro son propietarias por lo que presentan restricciones que impide que puedan ser utilizadas.

Por todo el estudio que se realizó de IR y sus etapas quedaron definidas todas las técnicas y herramientas que serán utilizadas en el proceso de desarrollo para lograr la calidad en cuanto los requisitos del sistema.

1.6 Metodología de Desarrollo.

El desarrollo de software no es sin dudas una tarea fácil. Como resultado a este problema ha surgido una alternativa desde hace mucho tiempo: la Metodología. Estas imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Mediante un proceso detallado con un fuerte énfasis en planificar, inspirado por otras disciplinas de la ingeniería, también su propósito es establecer un contrato social entre todos los participantes en un proyecto para conseguir la solución más eficaz con los recursos disponibles. Una metodología de desarrollo de software se refiere a un marco de trabajo (framework) que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas de información.

Proceso Unificado de Desarrollo de Software (RUP).

La metodología RUP (Rational Unified Process) es una metodología para la ingeniería de software que va más allá del simple análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software, se caracteriza por ser: guiado por casos de uso, los mismos son el instrumento para validar la arquitectura del software y extraer los casos de prueba; centrado en la arquitectura, es donde los modelos son proyecciones del análisis y el diseño que constituyen la arquitectura del producto a desarrollar; iterativo e incremental, durante todo el proceso de desarrollo se producen versiones incrementales del producto en desarrollo. Esta metodología se divide en

cuatro fases del proceso de desarrollo. Inicio, se hace mayor énfasis en actividades de modelado del negocio y de requisitos, elaboración, el objetivo es determinar la arquitectura óptima, le sigue construcción, en la cual se lleva a cabo la construcción del producto por medio de una serie de iteraciones y por último transición que su objetivo es obtener el producto preparado para su entrega a la comunidad de usuarios. (Sánchez, 2004)

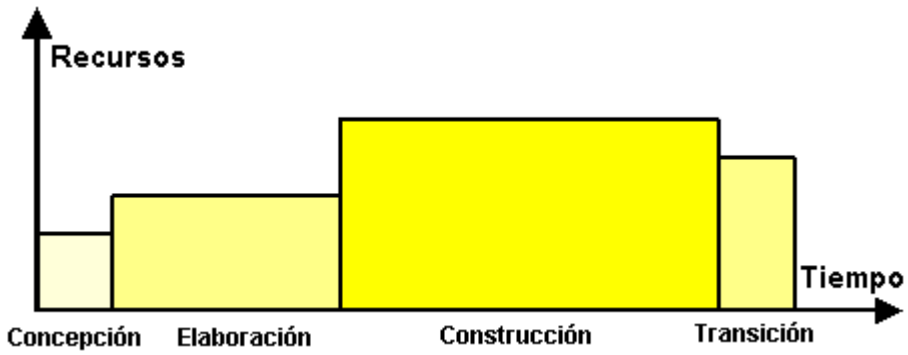

Fig. 1 Fases e Iteraciones de la Metodología RUP. Tomado de (Pressman, 2005)

Cada una de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. (Sánchez, 2004)

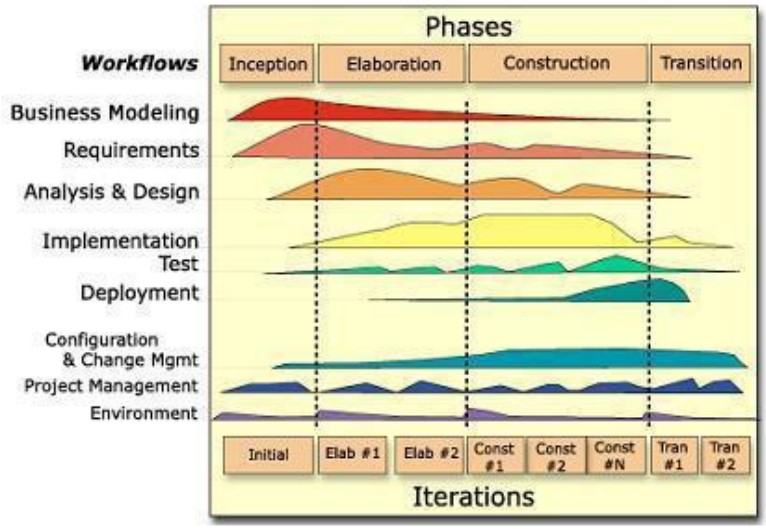

Fig. 2 Fases e Iteraciones de la Metodología RUP. Tomado de (Pressman, 2005)

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, para luego convertirse en un producto entregable al cliente. Esto trae como beneficio la retroalimentación que se tendría en cada entrega o en cada iteración.

En RUP están presente elementos como actividades; que no son más que los procesos que se llegan a determinar en cada iteración, están los trabajadores; que vienen siendo las personas involucradas en cada proceso, los artefactos; que van a ser documentos, modelos, o un elemento de modelo y el flujo de actividades; que es la secuencia de actividades realizadas por trabajadores y que producen un resultado de valor observable. (Sánchez, 2004)

Programación Extrema (XP)

La programación extrema se basa en la simplicidad, la comunicación y el reciclado continuo de código, para algunos no es más que aplicar una pura lógica. Es utilizada para proyectos de corto plazo.

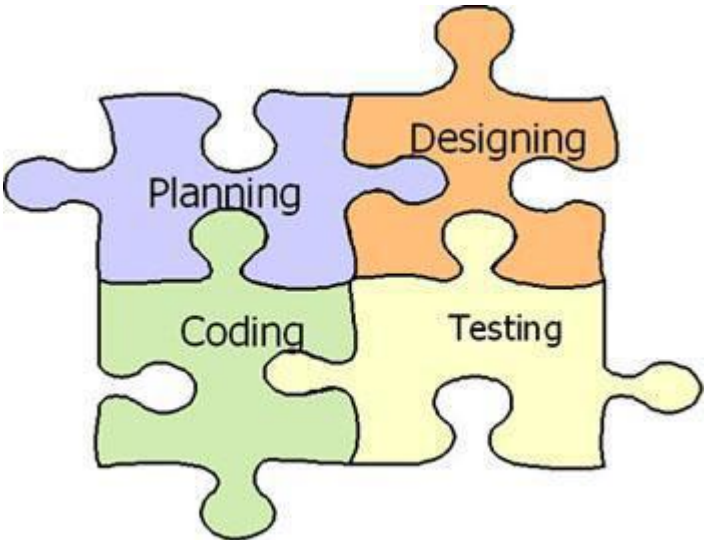

Fig. 3 Metodología Programación Extrema. Tomado de (Sánchez., 2005)

Esta metodología está basada en pruebas unitarias, estas son las pruebas realizadas a los principales procesos, consisten en comprobaciones (manuales o automatizadas) que se realizan para verificar que el código correspondiente a un módulo concreto de un sistema software funciona de acuerdo con los requisitos del sistema; otra característica es la refabricación, se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio y la programación en pares,

la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. (Sánchez, 2005.)

Lo fundamental en este tipo de metodología es:

- La comunicación, entre los usuarios y los desarrolladores.
- La simplicidad, al desarrollar y codificar los módulos del sistema.
- La retroalimentación concreta y frecuente del equipo de desarrollo, el cliente y los usuarios finales.

Después del estudio realizado se puede concluir que XP a pesar de sus ventajas como metodología ágil no se ajusta al proceso actual, en primer lugar, exige que el cliente forme parte del equipo de desarrollo, requisito que no puede ser satisfecho ya que no se cuenta con un cliente que pueda integrarse completamente al equipo de desarrollo, puesto que este pertenece a otras labores productivas, otro motivo que desfavorece el uso de esta metodología es la baja documentación que produce lo que dificultaría el proceso de mantenimiento, pues hay que prever qué pasará luego de entregado el software, cuando el equipo se disuelva, y sea necesario realizar algún cambio o mejora, ya que el equipo de analistas no es el mismo que desarrollará el software, es por esto que se hace necesario mantener cierta documentación. Se decidió no utilizar RUP a pesar de la amplia documentación que este brinda durante todo el proceso de desarrollo, debido a que esta metodología genera artefactos como el modelo de dominio, específicamente en este trabajo se utilizan los mapas conceptuales para entender los conceptos que maneja el usuario y permite representar jerárquicamente el conocimiento en las aplicaciones educativas.

Al no ser utilizadas ninguna de las metodologías mencionadas anteriormente, se realiza un estudio de otras que son utilizadas específicamente para el desarrollo de aplicaciones educativas, que se presentan a continuación:

RMM: Metodología de Administración de Relaciones (Relationship Management Methodology).

La metodología RMM, fue desarrollada por T. Isakowitz, en la Universidad de Nueva York en el año 1995. Se realiza la modelación de las aplicaciones a través de RMDM (Relationship Management Data Model), basado en el modelo Entidad – Relación y posee una herramienta CASE denominada: Relationship Management Case Tool – RMCase.

"RMM (Relationship Management Methodology) contiene el diseño y la construcción de aplicaciones hipermedia en un proceso de siete pasos. Es al mismo tiempo un enfoque "top down" y "bottom up". Durante la fase del diseño Entidad – Relación, entidades y relaciones son identificadas, las cuales se convertirán en nodos y enlaces en la hipermedia resultante. El segundo paso, diseño de cortes (slices), involucra el agrupamiento de atributos de entidades para la presentación. Los cortes (slices) son "unidades de presentación" que aparecen como páginas de una aplicación hipermedia. La separación del contenido y de los aspectos de la presentación no son satisfechos en este paso. RMM especifica la navegación con primitivas de acceso, como enlaces (links), agrupamiento (menus), índices (index) y recorridos guiados (guided tours) (Baumeistier, 2001).

OOHDM:

Metodología de Diseño Hipermedia Orientada a Objetos (Object – Oriented Hypermedia Design Methodology). Desarrollada por Schwabe y Rossi, en la Universidad de Río de Janeiro, Brasil y Universidad Nacional de la Plata, Buenos Aires respectivamente en el año 1996. Adopta la notación y los mecanismos de abstracción de la Programación Orientada a Objetos (POO) y consta de cuatro pasos para su ejecución: diseño conceptual, diseño navegacional, diseño de interfaz abstracta e implementación. Trabaja la representación a través de los siguientes modelos: esquema de clases, esquema de navegación, esquema contextual de navegación y vista abstracta de datos.

"El Modelo de Diseño de Hipermedias Orientado a Objetos: OOHDM (Object – Oriented Hypermedia Design Model) comprende cuatro actividades; estas son: modelo conceptual, diseño de navegación, diseño de interfaces abstractas e implementación. Estas actividades son ejecutadas en un estilo de desarrollo mixto a partir de los modelos incremental, iterativo y basado en prototipos. Este método trata a la aplicación como una vista superior al modelo conceptual. El concepto de contexto de navegación es introducido para describir la estructura de navegación. Es un concepto potente que permite diferentes agrupamientos de objetos de navegación con el propósito de navegar en ellos en diferentes contextos. Una notación especial es utilizada para la representación de la estructura de navegación. (Baumeistier, 2001)

En trabajos tempranos de investigación, OMT se propuso como la notación para el esquema conceptual; un poco más tarde los trabajos ya utilizan UML. Sin embargo, los diagramas de OOHDM no obedecen los patrones UML, sino que utilizan una notación propia para la perspectiva de los atributos en los diagramas

de clase y proponen otros tipos de diagramas para el diseño de la navegación y de las interfaces de usuarios abstractas.” (Baumeistier, 2001).

Luego del estudio realizado de las metodologías para este tipo de aplicaciones, se concluye que debido a que en la universidad no se tiene una basta experiencia en el uso de estas, sería complicado realizar el desarrollo del proceso con las mismas, independientemente que también presentan características desfavorables como es el caso de RMM que establece una herramienta para la modelación y OOHDM que no obedece los patrones UML sino que utiliza una notación propia para la perspectiva de los atributos en los diagramas de clase y propone otros tipos de diagramas para el diseño de la navegación. Al no ser posible utilizar ninguna de las metodologías estudiadas, se estudiará a continuación si será factible guiar el proceso por un marco de trabajo.

Marco de trabajo:

Un **marco de trabajo** para metodología de desarrollo de software consiste en una filosofía de desarrollo de programas de computación con el enfoque del proceso de desarrollo de software. Establece además la base para un proceso de software completo al identificar un número de actividades del marco de trabajo aplicables a todos los proyectos de software, sin importar su tamaño y complejidad. (Pressman, 2005)

A pesar de existir metodologías para aplicaciones educativas en el estudio realizado, estas no cumplen con los aspectos que se necesitan y tienen restricciones como en el caso de OOHDM que impone una anotación específica a usar, RMM establece una herramienta para el modelado. Por todo lo anterior expuesto se decide trabajar mediante un marco de trabajo que se definirá a continuación:

La Ingeniería de Software como disciplina está compuesta por cuatro áreas fundamentales Proyecto, Producto, Persona y Proceso. En este sentido se trabaja con un enfoque hacia el Proceso para modelar un recurso didáctico, el cual no está dirigido por una metodología de desarrollo.

Todo proceso de desarrollo de software define un marco común, dicho marco está compuesto por dos grandes grupos:

- Un primer grupo que comprende las actividades genéricas o comunes.
- Un segundo grupo que comprende las actividades sombrillas.

Cada uno de estos grupos de actividades están compuesto por un conjunto de acciones y estas acciones están conformadas por un conjunto de tareas que permitirán expresar de qué forma va a ocurrir el marco de trabajo.

En un marco de trabajo siempre van a estar presente las 5 actividades genéricas que lo conforman, no es posible desarrollar un proceso de desarrollo de software sin utilizar alguna de estas actividades (Pressman, 2005), estas son:

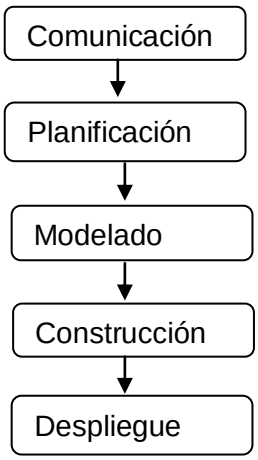

Fig. 4 Actividades Genéricas o Comunes.

Estas actividades comunes siempre van a estar acompañadas por un grupo de actividades que van a garantizar un mejor control y seguimiento y una mejor ejecución del proceso e imprimirán calidad y garantizaran una correcta ejecución del mismo, estas se denominan Actividades Sombrillas. Estas se agrupan en 8 actividades fundamentales, de las cuales, las 6 primeras fueron utilizadas en el presente trabajo debido al alcance del mismo. Estas actividades son:

1. Seguimiento y Control.
2. Gestión del Riesgo.
3. Aseguramiento de la calidad del producto.
4. Revisiones técnicas formales.
5. Medición.

6. Gestión de la configuración.
7. Gestión de la reutilización.
8. Preparación y producción del producto. (Pressman, 2005)

No es posible definir un marco de trabajo bajo un enfoque de calidad que no contenga actividades sombrillas para garantizar su calidad y completitud.

Por todo lo anterior expuesto se muestra como queda estructurado el marco de trabajo propuesto.

Actividades Genéricas	Artefactos generados en cada actividad
Comunicación	▪ Entrevistas con el cliente. ▪ Extracción de requisitos. ▪ Especificación de requisitos. ▪ Validación de requisitos.
Planificación	▪ Planilla de Planificación del proyecto. ▪ Reuniones con el cliente.
Modelado	Diagrama de Caso de Uso. Descripciones textuales de Los caso de uso. Diagrama de Actividades. Diagrama de Clases. Diagrama de Estructura de Navegación. Diagrama de Estructura de Presentación. Diagrama de Secuencia.
Construcción	---

Despliegue	---

Tabla 1: Relaciones entre actividades del marco de trabajo y artefactos realizados.

Este es el marco de trabajo propuesto para el desarrollo de software, que permitirá guiar el proceso y la comunicación con el cliente, donde se definieron las actividades comunes y las actividades sombrillas utilizadas. En este sentido se realizan en el trabajo las actividades de comunicación, planificación y modelado ya que sólo se realiza el análisis y diseño del módulo de Paginado y Segmentado.

Dada las condiciones trabajo, en cuanto a un equipo de trabajo pequeño, motivado, con muy buena relación y comunicación y con el fin de satisfacer al cliente y hacer entrega del trabajo en pequeño tiempo, se definió un marco de trabajo siguiendo un enfoque ágil.

Como mismo existen un conjunto de actividades básicas para el proceso, hay un conjunto de métodos de modelación básicos que cumplen cualquier notación:

- Método de los elementos basado en escenarios.
- Método de los elementos basados en clase.
- Método de los elementos basado en comportamiento. (Pressman, 2005)

Estos son los métodos básicos de modelados que a su vez son métodos de análisis y diseño.

Ha mediado de los años 90 las metodologías y las notaciones estaban juntas. Esta ciencia es muy joven aún, aunque se sustenta en tecnologías de otras ciencias, esto hizo que a finales de los 90 estas dos ramas se separaran, pues una línea debe guiar el proceso de cómo hacer un Software y otra cómo representarlo.

1.7 Lenguajes de Modelado.

Un sistema, tanto del mundo real como en el mundo del software, es bastante complejo, por ello es necesario dividir el sistema en partes o fragmentos si se quiere entender y administrar su complejidad. Estas partes se pueden representar como modelos que describan sus aspectos esenciales. Por tanto, un paso útil en la construcción de un sistema de software es el de crear modelos que organicen y comuniquen los detalles más importante de la vida real con que se relacionan y del sistema a construir.

IDEF0

IDEF0 es una técnica de modelación concebida para representar de manera estructurada y jerárquica las actividades que conforman un sistema o empresa, y los objetos o datos que soportan la interacción de esas actividades. Un modelo IDEF0 se compone de una serie jerárquica de diagramas que permiten, mediante niveles de detalle, describir las funciones especificadas en el nivel superior. En las vistas superiores del modelo la interacción entre las actividades representadas permite visualizar los procesos fundamentales que sustentan la organización (Estrada, 2007).

Lenguaje de Modelado Unificado.

El Lenguaje de Modelado Unificado (UML) es un lenguaje estándar para escribir planos de software. Puede utilizarse para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra gran cantidad de software. Es un lenguaje que ayuda a interpretar grandes sistemas mediante gráficos o texto, obteniendo modelos explícitos que contribuyen a la comunicación durante el desarrollo, ya que al ser estándar, pueden ser interpretados por personas que no participaron en su diseño. En este contexto, UML sirve para especificar modelos no ambiguos y completos. UML es un método formal de modelado. Esto aporta las siguientes ventajas:

- Mayor rigor en la especificación.
- Permite realizar una verificación y validación del modelo realizado.
- Se puede automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa. Esto hace que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño, de más alto nivel, de la estructura de un proyecto (Orallo, 2007).

Lenguaje de modelado para Aplicaciones Educativas (ApEM-L):

El desarrollo de los Software Educativos es una disciplina que se encuentra en plena evolución en la actualidad, el proceso de aprender cómo desarrollar aplicaciones educativas acaba de comenzar. Para ello se hacen uso de nuevos principios y herramientas para asistir a los desarrolladores de este tipo de aplicaciones, donde no siempre estas prácticas tienen éxito debido al uso de técnicas o metodologías inapropiadas.

Capítulo 1: Fundamentación Teórica

Numerosas han sido las propuestas para la modelación de aplicaciones multimedia, mayormente enfocados en la modelación de las relaciones y sincronización de de estas presentaciones.

Generalidades de ApEM-L.

UML es un lenguaje universal que presenta un número elevado de propiedades fundamentales de modelado y está diseñado para ser utilizado con múltiples propósitos, pero hay ocasiones en que no cumple con especificidades de un determinado dominio de conocimiento. Desafortunadamente, UML no soporta todos los aspectos de las aplicaciones multimedia de una manera adecuada e intuitiva. Especialmente, características del lenguaje para el modelado de aspectos de la interfaz de usuario no están explícitamente proporcionadas.

ApEM – L se presenta como una extensión de UML. La concepción del *Lenguaje para la Modelación de Aplicaciones Educativas* (ApEM – L) tuvo los siguientes objetivos:

1. Desarrollar una extensión del lenguaje de modelado UML, tomándolo como base e incorporando a este, a través de sus mecanismos de extensión, los elementos fundamentales del proceso productivo UCI. De esta forma se produjo un lenguaje de propósito particular para la modelación de aplicaciones educativas.

2. Incorporar los elementos más significativos de extensiones como OMMMA – L (2001) y a su vez respetar lo establecido por el estándar OCL (2003), para de esta forma lograr una extensión consistente y escalable en el tiempo.

3. No complicar ApEM – L con elementos que lo convirtieran o abarcaran en un método de desarrollo de aplicaciones educativas, sino sólo el área de la representación y la documentación de este tipo de aplicaciones.

4. No circunscribir ApEM – L a un proceso de desarrollo en específico, sino expresarlo de manera tal que pueda ser utilizado con cualquiera de los existentes, aunque se sugiere la utilización de procesos de desarrollo iterativos incrementales y basados en prototipos, que permitan la modelación de sistemas orientados a objetos. (Ricardo, Julio 2007)

Áreas conceptuales de ApEM-L.

- ***Estructura lógica*:** está compuesta por la vista estática y la vista de arquitectura. La primera de ellas está compuesta por el diagrama de clases y el diagrama de casos de uso. Cualquiera de los modelos presentados por ApEM – L define los conceptos claves de la aplicación que modela, las propiedades internas de estos y sus relaciones. De los diagramas mencionados sólo han sido modificados los siguientes: diagrama de clases y el diagrama de componentes, el resto mantuvo lo establecido por UML. También, aun manteniendo lo establecido originalmente para los diagramas de casos de uso, se adicionaron un conjunto de elementos a la descripción textual de los casos de uso propuesta por UML, para una mejor descripción del contexto productivo de los software educativos.
- ***Comportamiento dinámico*:** realmente esta área no ha sido grandemente modificada, pues sólo se ha hecho una pequeña adición al diagrama de secuencia, salvo que ciertamente se ha enriquecido la semántica original de UML para estos diagramas. El comportamiento de la aplicación está descrito por la vista de comportamiento, la cual está compuesta por los diagramas: de actividad, de secuencia, de colaboración y de estados, donde sólo ha sido modificado el segundo de los listados anteriormente; adicionando una variable de tiempo donde quiera que sea necesario su especificación para un mejor entendimiento.
- ***Gestión del modelo*:** esta área es la que ha sufrido grandes cambios tanto en su carácter semántico como sintáctico, con la incorporación de estereotipos restrictivos en todos los diagramas a partir de nuevos conceptos incorporados a los diagramas de clases originales o básicos de UML. Se crean dos nuevos diagramas: el de estructura de la presentación y el de estructura de la navegación. (Ricardo, Julio 2007)

Vista de ApEM-L

La ***Vista estática*** de ApEM – L está compuesta por dos diagramas, el ***Diagrama de clases*** y el ***Diagrama de Casos de Uso***. El diagrama de Clases está dividido en dos grandes zonas, la de la izquierda dedicada al árbol jerárquico de las ***clases modelo entidad medias*** que representan los recursos mediáticos de la aplicación y en la zona de la derecha del diagrama las clases que controlan la *lógica del negocio* de la aplicación propiamente dicha, esta zona de la derecha vuelve a subdividirse en cuatro zonas. La primera

dedicada a las clases ***vista***, la contigua a esta y en el extremo superior derecho dedicada a las clases ***controladoras***, inmediatamente debajo de esta sección, la destinada a las clases ***modelo***, quedando una banda inferior derecha dedicada en su extremo derecho a las clases ***modelo entidad persistentes*** para el tratamiento de la información persistente de la aplicación; y en el extremo izquierdo las clases correspondientes al Lenguaje de Alto Nivel (HLL en sus siglas en inglés correspondientes a High Level Languege) con el que se programe (Lingo, ActionScript, C#, C++, Object Pascal, PH P, etc.) (Ricardo, Julio de 2007).

Vista de arquitectura.

La vista de arquitectura está compuesta por el ***diagrama de componentes*** y el ***diagrama de despliegue***. El último de los mencionados no sufre cambios en ApEM – L, no así el de componentes donde se incorporan restricciones en los tipos de componentes. Al seguir la arquitectura propuesta por el patrón Modelo-Vista-Controlador-Entidad (MVC-E). En su interior se encontrarán las clases vista, modelos, controladoras, HLL y modelo – entidad, organizadas en componentes que estarían internamente en el paquete Presentación. Solo sería necesario luego un diagrama de componentes a un nivel de abstracción superior que represente cómo se comunican los distintos tipos de paquetes componentes del sistema.

Existen entonces diferencias en cuanto a las vistas estáticas de arquitectura entre UML y ApME-L, específicamente en los diagramas de clases, ya que UML no establece un patrón arquitectónico que rija la concepción del diseño y no contiene la semántica de tratamiento de las clases asociadas a las tecnologías multimedia. Mientras que en las vistas de ApME-L se establece el patrón arquitectónico MVC-E para el diseño de las aplicaciones educativas, se plantea la semántica y los estereotipos restrictivos descriptivos para las clases asociadas a las tecnologías multimedia e hipermedia y se organiza la estructura del diagrama en secciones para la representación lógica de los distintos tipos de clases. (Ricardo, Julio 2007)

Vista de comportamiento.

La vista de comportamiento está compuesta por cuatro diagramas: ***de actividades***, ***de estado***, ***de secuencia*** y ***de colaboración***; siendo estos dos últimos generalizados en diagramas de interacción, pues como plantea la semántica de UML, representan la manera en la que los objetos de la aplicación intercambian mensajes para darle cumplimiento a sus responsabilidades. En ApEM – L sólo ha sido

modificado el diagrama de interacción de secuencia, con un estereotipo descriptivo y por ende decorativo, para denotar el tiempo como variable de sumo interés en aplicaciones de este tipo.

Si se compara UML y ApEM-L en su vista de comportamiento, tenemos que en esta casi no se presentan cambios, los diagramas de secuencia, colaboración y estado se mantienen sin modificaciones de acuerdo a lo planteado por el lenguaje base UML. En el diagrama de actividades no se modifica la semántica del lenguaje base, sino que se extiende al incorporarse el estereotipo restrictivo asociado al tiempo y su representación, siempre que sea necesario, enriqueciendo la descripción funcional de este tipo de aplicaciones. (Ricardo, Julio 2007)

Vista de presentación.

Para las aplicaciones multimedia el diseño de la interfaz de usuario conjuntamente con la planificación de la presentación son aspectos de gran importancia y de sumo interés para los desarrolladores de este tipo de aplicaciones. Esta vista ha sido incorporada completamente al lenguaje base UML, para permitir utilizar la semántica original de dicho lenguaje en la construcción de estructuras lógicas de presentación y navegación, incorporando un conjunto de estereotipos restrictivos y descriptivos para una mejor modelación, construyendo el ***diagrama de estructura de navegación*** y el ***diagrama de estructura de presentación***. (Ricardo, Julio 2007)

- ***Diagrama de estructura de navegación:***

Para el desarrollo del diagrama de estructura de navegación se han definido, sobre el concepto de clase, una segunda clasificación de las mismas, por lo que se enriquece la semántica del concepto original de clases, quedando las siguientes: ***clase menú***, ***clase índice***, ***clase consulta*** y ***clase botón***, además de utilizar las ya definidas ***clases modelo-entidad-media texto*** y ***modelo-entidad-media imagen*** pues estas pueden ser elementos importantes de navegación en el modelo, así como comportarse como clases que visualizan información. La ***clase menú*** (<<Menu>>), es el elemento de composición de una clase vista, desde donde se puede llegar a otras diferentes clases vistas con las cuales se conecta este menú, pues contiene una lista de las opciones de movimiento **siguiente**, pero no los valores de esas opciones, los cuales serán ofrecidos por el resto de los tipos de clases. Dicho menú no tiene que necesariamente llevar a una clase vista directamente, sino que puede ser a través de otro tipo de clase de navegación (guía, consulta o índice). La ***clase consulta*** (<<Consulta>>) es aquel tipo de clase que permite el enlace con una clase vista a través de un valor directo (variable de consulta) asignado a la búsqueda para la

visualización del elemento a mostrar, que se sitúa como identificador en la relación entre las clases. Por último la ***clase índice*** (<<Indice>>), es aquella clase que denota el valor de direccionamiento hacia una clase vista a partir de una opción determinada. La ***clase botón*** (<<Boton>>) es aquella clase que denota la existencia de un elemento interactivo del tipo botón para producir un camino en la navegación hacia una clase vista, utilizando como intermediarias a clases de tipo consulta o índice (Ricardo, Julio de 2007).

- **Diagrama de estructura de presentación.**

Un aspecto esencial en las aplicaciones educativas con tecnología multimedia e hipermedia es la definición de la estructura que tendrán las futuras interfaces de comunicación con el usuario en cuanto a sus componentes y distribución espacial. En el contexto de ApEM – L no se trabajará la distribución espacial, pues las herramientas CASE actuales que soportan el lenguaje no lo permiten tal y como pretende dibujarlo hoy día. Por el contrario se trabajará la estructura que tendrán esas interfaces de comunicación incorporando un conjunto de estereotipos restrictivos y descriptivos que permitirán una organización lógica de los elementos conformantes de dichas interfaces y dejándoles a los diseñadores gráficos la función de decidir dónde y cómo serán en términos visuales dichos elementos. Para la mejor estructuración del modelo, se realiza una segunda clasificación sobre el concepto original de clase, definiendo dos nuevos tipos de estas: la ***clase Estáticos*** y la ***clase Interacción***, las cuales seccionarán los elementos que cumplan con cada una de las características que denotan los propios nombres. La clase *estáticos* agrupará los componentes que sólo tienen como función visualizar información, pero que no permiten interacción con el usuario. Todo lo contrario con los agrupados bajo la clase *interacción,* los cuales serán los elementos de la vista que permiten interacción del usuario con el sistema informático modelado (Ricardo, Julio de 2007).

En esta vista no se puede hacer comparación alguna entre los artefactos que se obtienen, ya que UML no cuenta con una vista de presentación, siendo solamente generado por ApEM-L los diagramas de estructura de navegación y de estructura de presentación.

Por todo lo anteriormente planteado, quedan demostradas las nuevas facilidades que brinda el lenguaje de modelado para aplicaciones multimedia ApEM-L y los aspectos por los cuales se hizo necesario hacer esta extensión de UML a pesar de ser un lenguaje altamente calificado para el desarrollo de software, carece de funcionalidades para el desarrollo de aplicaciones multimedia.

En este aspecto se establece trabajar con el lenguaje de modelado ApEM-L, ya que es el lenguaje de modelado establecido para el desarrollo de aplicaciones multimedia educativas, UML a pesar de ser el estándar mundial de modelado para el desarrollo de software no soporta todos los aspectos necesarios para el modelado de aplicaciones de este tipo, específicamente en el modelado de las interfaces del usuario. Por lo que modelar con UML complicaría la modelación de software para aplicaciones multimedia.

1.8 Herramientas.

1.8.1 Herramienta Case.

Las herramientas CASE son un conjunto de programas y ayudas, que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software. Estas herramientas han desempeñado un papel importante en el desarrollo de aplicaciones informáticas, destinadas a aumentar la productividad; reduciendo los costes de las mismas en término de tiempo y dinero.

Algunos de los componentes de las herramientas CASE permiten:

- Confeccionar la definición de requisitos de los usuarios.
- Mejorar el diseño de los sistemas.
- Mejorar la eficiencia en la programación (por su generación automática de códigos).
- Otorgar a la administración un mejor soporte en la documentación. (Informática., 2008)

Visual Paradigm:

Visual Paradigm es una herramienta CASE profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Produce la documentación completa del sistema en formato PDF, HTML y MS Word. Tiene la capacidad de integrarse con Eclipse/IBM WebSphere, Builder, NetBeans IDE, Oracle JDeveloper, BEA Weblogic y otros. Además soporta un conjunto de lenguajes, tanto en la generación de código e ingeniería inversa sobre Java, C + +, PHP, XML Schema, entre otros (Autores, 2009).

Visual Paradigm ofrece distintas funcionalidades como:

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en Caso de Uso y enfocado al negocio generando un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa en su versión profesional, e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDE.
- Disponibilidad en múltiples plataformas (Windows, Linux).
- La universidad cuenta con la licencia comercial para el trabajo con esta herramienta.

Dentro de las herramientas utilizadas para el desarrollo del trabajo, está el CompendiumLD.

1.8.2 CompendiumLD.

CompendiumLD es una herramienta para la visualización de diseños de aprendizajes, fue desarrollada para ayudar a los profesores y diseñadores a representar visualmente su diseño de aprendizaje. Es una adaptación de la herramienta Compendio, que es una especie de mapas mentales o software de argumentación, que proporciona un conjunto predeterminado de íconos que se utilizan para conformar los mapas conjunto a una descripción de cada nodo y las relaciones entre las partes interesadas. El CompendiumLD tiene otros tipos de nodos que se refieren explícitamente a los diferentes aspectos del proceso de diseño, para representar las actividades de aprendizaje, tareas, funciones entre otras especificidades y los enlaces para relacionar dichas especificidades.

Usar el compendiumLD para la creación de diseños de aprendizaje aporta beneficios. Una herramienta flexible de diseño visual como esta, ofrece al usuario la oportunidad de expresar sus ideas de diseño al detalle. (Brasher, 2008)

En este caso se trabaja con la herramienta para la obtención de **Mapas Conceptuales** que ayudan a la representación de los contenidos que representa el basamento teórico de este módulo del sistema.

El cliente estableció el uso de la herramienta **Compendium LD:** Flexible para la creación de diseños de aprendizajes. Ofrece a los usuarios la oportunidad de grabar y comunicar sus ideas de diseño con un alto nivel de detalle. Se seleccionó esta herramienta, ya que proporciona una serie de íconos para crear mapas que describen los temas y contenidos por los que está compuesto el módulo Paginado y Segmentado. La misma es de gran ayuda para maestros y diseñadores.

1.9 Patrones.

En términos generales, un patrón es un cúmulo de información que proporciona respuesta a un conjunto de problemas similares en un contexto dado. Los patrones hacen la producción de software más flexible al cambio, establecen parejas problema-solución, ayudan a especificar interfaces, facilitan la reutilización del código y permiten una fácil comprensión debido a la documentación estándar que presentan. Existen diversas clases de patrones, entre las cuales se registran: de casos de uso, de análisis, de arquitectura, de diseño, de programación, entre otros.

Los *Patrones de Casos de Usos:*

Extensión Concreta:

La utilización de este patrón se evidencia en los casos de uso donde se tiene algún caso de uso extendido. Un caso de uso extiende a otro cuando sin alterar a este, se incorpora su funcionalidad como parte integral del primero. Se denota con una relación que apunta del caso extendido al caso base y la conexión se hace o bien al principio del flujo de eventos principal del caso base o en alguno de los puntos de extensión que este haya definido.

Inclusión Concreta:

La utilización de este patrón se evidencia en los casos de uso que tienen casos de uso incluido donde la relación es muy íntima, es la modificación del caso base. La inclusión añade funcionalidad al caso base.

Concordancia-Adición:

La aplicación de este patrón se refleja en el caso de uso donde es extendido de varios casos de usos y no puede ser instanciado por sí solo. (Pressman, 2005)

Capítulo 1: Fundamentación Teórica

Patrones de Diseño:

Brad Appleton define un patrón de diseño de la siguiente manera: “Un patrón es una semilla de conocimiento, la cual tiene un nombre y transporta la esencia de una solución probada a un problema recurrente dentro de cierto contexto en medio de intereses y competencias”. Patrones **GRASP**: Patrones de Software para la Asignación General de Responsabilidad. Dicho de otro modo un patrón de diseño describe una estructura de diseño que resuelve un problema de diseño en particular dentro de un contexto específico y en medio de “fuerzas” que pueden tener un impacto en la manera que se aplica y utiliza el patrón.

Alta cohesión.

Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y los mejoramientos.

El patrón alta cohesión presenta semejanzas con el mundo real. Se sabe que si alguien asume demasiadas responsabilidades, sobre todo las que debería delegar, no sería eficiente.

Bajo acoplamiento.

El bajo acoplamiento estimula asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento. Este patrón soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. (Pressman, 2005)

Patrones **GoF:**

Según el libro GoF (Gang-Of-Four Book) existen 3 clasificaciones de patrones (Creación, Comportamiento y Estructurales).

El patrón Singleton clasificado dentro de los patrones GoF como de Creación. Utilizado para la creación de instancias. Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.

El patrón Observer clasificado dentro de los patrones GoF como de Comportamiento. Un efecto lateral muy frecuente, en aquellos sistemas que se fragmentan en un conjunto de clases que cooperan, es la necesidad de mantener la consistencia entre los distintos objetos interrelacionados. Para no recurrir a soluciones fuertemente acopladas (que reducen la posibilidad de reutilización), este patrón define una dependencia de uno-a-muchos entre objetos, para que, cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente.

El patrón Visitor clasificado dentro de los patrones GoF como de Comportamiento, representar una operación que está pensada para ser aplicada sobre los elementos de una estructura de objetos, permitiendo así definir y añadir un nuevo comportamiento sin necesidad de cambiar las clases de los elementos de la estructura de objetos. (Rojas, 2005)

Patrón Arquitectónico:

Para el desarrollo del presente trabajo se toma como propuesta arquitectónica para el desarrollo de aplicaciones multimedia el patrón Modelo – Vista – Controlador– Entidad (MVC-E)

Se estudia en este trabajo una versión del Modelo – Vista – Controlador (MVC) la cual soluciona las necesidades de una aplicación multimedia:

El patrón Modelo – Vista – Controlador – Entidad (MVC-E). El cual tiene sus bases el conocido Modelo – Vista – Controlador el cual originalmente divide una aplicación interactiva en tres áreas: procesamiento, entrada y salida, haciendo uso de tres tipos fundamentales de clases: Clase Modelo, Clase Vista y Clase Controladora.

- Clase Vista: la cual recibe las peticiones del usuario al sistema y muestra la información de salida al usuario.
- Clase Modelo: esta gestiona el procesamiento y almacenamiento de la información persistente de la aplicación.
- Clase Controladora: esta clase recibe las entradas para la gestión de las clases modelos a partir de las clases vistas. Los eventos son traducidos a solicitudes de servicios para el modelo o la vista.

Con las características distintivas del software educativo producido en Cuba, se hace uso de una nueva variante del patrón anteriormente mencionado, donde se descarguen las responsabilidades de la clase modelo concernientes al procesamiento y almacenamiento de la información persistente de las

aplicaciones, incorporando una nueva clase al modelo denominado Modelo Entidad, con dos tipos fundamentales, la clase Modelo-Entidad-Media y Modelo-Entidad-Persistente. La primera de estas con la responsabilidad de agrupar las clases que identifican las medias y su árbol de jerarquía en la aplicación y la segunda tiene como responsabilidad la gestión de la información persistente, que antes sobrecargaba a la clase Modelo del patrón MVC original (Ricardo, Julio de 2007).

Conclusiones Parciales.

A lo largo del presente capítulo han quedado planteados los problemas que presenta la asignatura de Sistema Operativo con respecto a la integración de las TIC, lo cual genera la necesidad de desarrollar el análisis y diseño del módulo de Paginado y Segmentado para el subsistema Autoaprendizaje. Quedaron expuestos todos los elementos teóricos necesarios para la realización del mismo. Se realizó un estudio del arte de los laboratorios virtuales donde se seleccionó en este sentido el laboratorio virtual Web, de la misma manera se realizó un estudio de las metodologías, lenguajes de modelado y herramientas de desarrollo que se podrán utilizar y las explicaciones de estas, donde se definió utilizar un marco de trabajo definido para guiar el proceso, como lenguaje de modelado ApEM-L, Lenguaje de modelado para aplicaciones educativas y las herramientas Visual Paradingm y CompendiumLD para representar el modelado y el mapa conceptual.

Capítulo 2: Descripción de la solución.

2.1 Introducción.

El presente capítulo presenta la realización del mapa conceptual referente al contenido de Paginación-Segmentación el cual conforma el basamento teórico para el desarrollo del presente trabajo. La especificación de requisitos funcionales y no funcionales del sistema con sus clasificaciones. Contiene el Diagrama de Caso de Uso con sus concernientes descripciones textuales (Anexos) y las clasificaciones de los casos de uso, punto de partida para generar los artefactos propuestos para el desarrollo del módulo como solución, sirviendo de apoyo el lenguaje de modelado utilizado.

2.2 Mapa Conceptual de Paginado-Segmentado.

Los Mapas Conceptuales proporcionan un resumen esquemático de lo aprendido, ordenado de una manera jerárquica. El conocimiento está organizado y representado en todos los niveles de abstracción, situando los más generales e inclusivos en la parte superior, y los más específicos y menos inclusivos en la parte inferior. Las relaciones entre ellos se explicitan mediante líneas que unen sus cajas respectivas. Las líneas, a su vez, tienen palabras asociadas que describen cuál es la naturaleza de la relación que liga los conceptos.

Para el contenido de Paginado y Segmentado la Administración de Memoria incluye la Monoprogramación y la Multiprogramación, en ambas se maneja el tema de Protección, funcionando de manera diferente para cada uno, siempre estableciendo Registros Límites. Específicamente la Monoprogramación tiene en cuenta los aspectos de Solapamiento y Relocalización, donde esta última tiene dos tipos de Relocalización, la Dinámica y la Estática. La Multiprogramación define también tipos de Particiones, Fijas y Variables, donde las Particiones Variables manejan la Fragmentación Externa y las Particiones Fijas manejan la Fragmentación Interna. Este tema de Paginado y Segmentado en la Administración de Memoria, maneja también el Intercambio de Trabajo y las Técnicas de Asignación como son Mapas de bits, Paginado, Segmentado, Esquemas Combinados y Listas Enlazadas, esta última utiliza los algoritmos de asignación: Primer Acceso, Próximo Acceso, Mejor Acceso y Peor Acceso. Concluyendo así todo el contenido en cuanto a Paginado y Segmentado. (LONDOÑO, 2004)

Capítulo 2: Descripción de la Solución

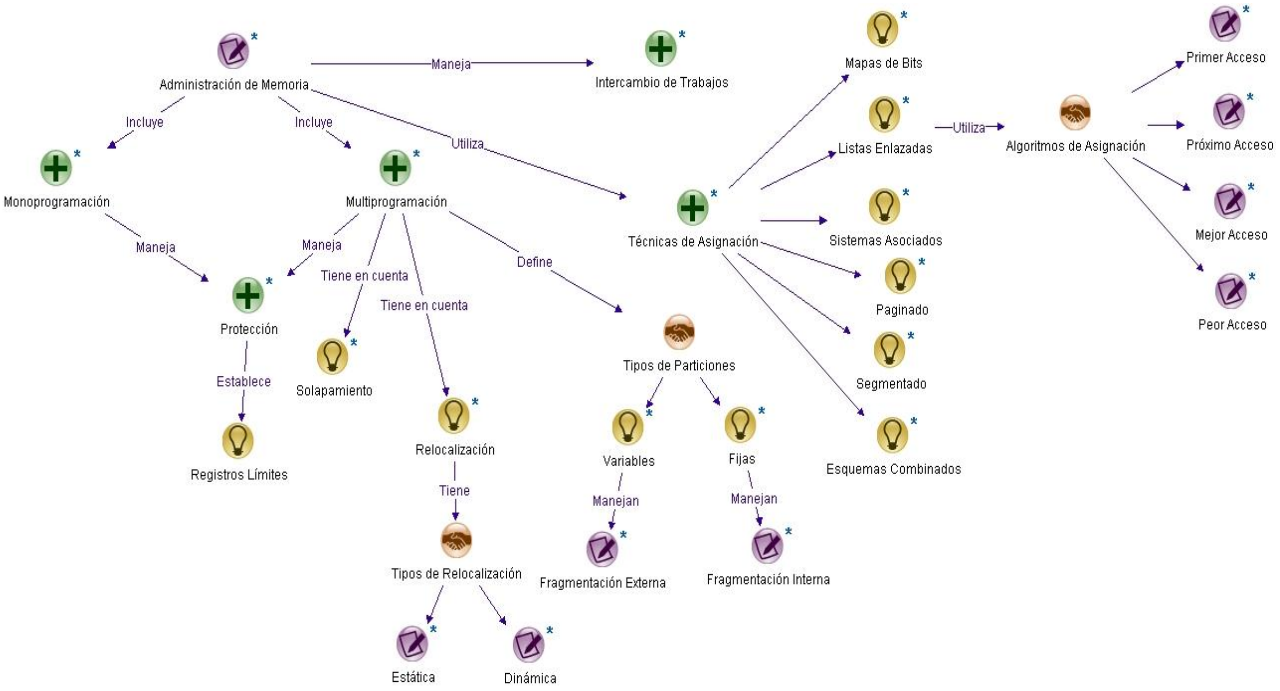

Fig. 5 Mapa Conceptual del contenido Paginado-Segmentado.

2.3 Requisitos Funcionales.

Bajo las condiciones presentadas anteriormente y de acuerdo con el estudio de la IR realizado en el capítulo anterior de las diferentes técnicas que existen para la extracción, especificación y validación de requisitos, quedaron definidas las siguientes técnicas a utilizar:

Se realizaron numerosas reuniones y trabajos de mesas para un mejor entendimiento del cliente con el objetivo de acumular un gran número de ideas y que quedaran explícitos de la mejor forma todas las necesidades del cliente, esto fue posible aplicando las técnicas Entrevistas y Tormenta de Ideas. Para una mejor especificación de los requisitos, de manera que estos sean entendibles para todo el personal (Cliente y Analista), en cuanto a terminología utilizada, se decidió utilizar la técnica del Lenguaje Natural, donde quedaron expresados de una forma clara y entendible todos los requisitos identificados. Posteriormente para la validación de los requisitos, se realizó una revisión cautelosa de todos los requisitos identificados, comprobando que estos estén expresados de manera correcta y que abarcan realmente todas las funcionalidades del sistema que desea el cliente, todo esto mediante la técnica de validación Reviews.

Capítulo 2: Descripción de la Solución

Quedaron definidos los siguientes Requisitos Funcionales (RF) y Requisitos No Funcionales (RNF) con sus clasificaciones asociadas:

Requisitos Funcionales.

RF 1: El sistema debe permitir visualizar el contenido que compone el proceso Exclusión Mutua y Sincronización.

RF 2: El sistema debe permitir modificar el contenido.

RF 3: El sistema debe permitir deshabilitar el contenido.

RF 4: El sistema debe permitir habilitar el contenido.

RF 5: El sistema debe permitir estudiar contenidos.

RF 5.1: El sistema debe permitir reanudar lección.

RF 5.2: El sistema debe permitir seleccionar el contenido a estudiar.

RF5.3: El sistema debe permitir guardar las trazas de navegación.

RF 5.4: El sistema debe permitir culminar su navegación.

RF 6: El sistema debe permitir consultar su evaluación.

RF 7: El sistema debe permitir revalorizar evaluación.

RF 8: El sistema debe permitir rechazar ejercicio.

RF 9: El sistema debe permitir solicitar ayuda.

RF 10: El sistema debe permitir abandonar tema.

Requisitos no funcionales.

Apariencia o interfaz externa:

RNF1: El diseño de la interfaz debe ser sencillo y fácil de usar con reconocimiento visual a través de elementos visibles que identifiquen cada una de sus acciones.

RNF2: La combinación de colores debe ser agradable a la vista del usuario.

Usabilidad:

RNF3: El sistema puede ser usado por cualquier persona que posea conocimientos básicos sobre el funcionamiento interno de un Sistema Operativo.

Rendimiento:

RNF4: La respuesta a solicitudes más complejas de los usuarios del sistema no debe exceder 9 segundos.

RNF5: La aplicación deberá estar disponible las 24 horas del día.

Portabilidad:

RNF6: El sistema debe poder ejecutarse en varios Sistemas Operativos.

Seguridad:

RNF7: El sistema debe establecer un control a la hora de acceder al sistema de forma autorizada y segura para evitar que no entre ningún usuario y modifique cualquier información.

Legales.

RNF8: Cada una de las medias (imágenes, videos, sonidos) que se utilicen en el producto deben tener el permiso legal de sus autores y su aprobación para hacer uso de ellas.

2.4 Diagrama de Caso de Uso.

Como resultado de todos los requisitos planteados anteriormente se generó el siguiente Diagrama de Caso de Uso según lo establecido por el lenguaje de modelado ApEM-L.

Teniendo en cuenta que a este diagrama no sufrió modificaciones algunas por ApEM-L, el Diagrama de Caso de Uso describe las funcionalidades propuestas para el sistema. Un caso de uso representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema. Un caso de uso es una unidad de trabajo significativo. Se utilizaron los patrones de caso de uso: Extensión Concreta y Concordancia-Adicción.

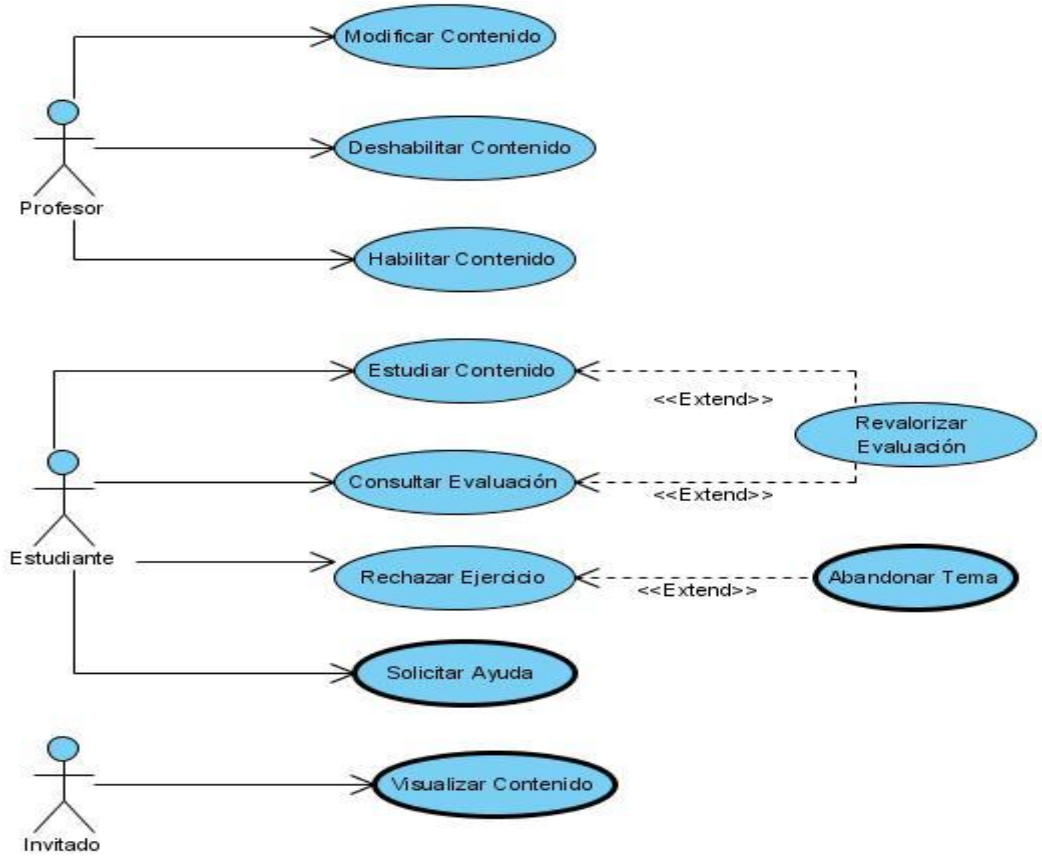

Fig. 6 Diagrama de Caso de Uso.

Las descripciones de casos de uso son reseñas textuales de los mismos. Normalmente tienen el formato de una nota o un documento relacionado de alguna manera con el caso de uso, y explica los procesos o actividades que tienen lugar en el caso de uso. A continuación se muestra cómo quedarían las descripciones textuales de los caso de usos presentados anteriormente, utilizando las modificaciones que se le realizaron a este, al incorporar los elementos significativos del software educativo. Tomando como representación específica la descripción del caso de uso Deshabilitar Contenido.

Descripción textual del caso de uso "*Deshabilitar Contenidos"*	
Actores del caso de uso	Profesor (Inicia)
Propósito	El caso de uso tiene como propósito permitir al profesor

	deshabilitar contenidos o un tema.
Resumen	El caso de uso se inicia cuando el profesor selecciona la opción "Deshabilitar Contenidos", una vez seleccionada la opción deseada el profesor decide qué contenido o tema desea deshabilitar, luego oprime el botón "Deshabilitar" y culmina el caso de uso cuando se presenta el Menú Principal.
Casos de uso asociados	
Referencias	RF 3
Precondiciones	• El actor debe estar autenticado en el sistema.
Poscondiciones	• Los contenidos, los temas o el módulo quedan deshabilitados para los estudiantes. • Menú Principal de la aplicación mostrado.

Curso Normal de los Eventos

Acciones del Actor	**Respuesta del Sistema**
1. El Profesor selecciona la opción "Deshabilitar Contenido".	1.1 El sistema busca los contenidos disponibles a habilitados. 1.2 El sistema muestra el listado de contenidos disponibles a deshabilitar.
2. El Profesor selecciona el contenido a deshabilitar.	
3. El Profesor oprime el botón "Aceptar".	3.1 El sistema verifica si hay estudiantes con acceso al contenido. 3.2 El sistema muestra mensaje de notificación.

	3.3 El sistema deshabilita el contenido sin guardar las trazas de precedencia.
4. El Profesor acepta la notificación.	4.1 El sistema deshabilita el contenido. 4.2 El sistema muestra nuevamente el listado de contenidos disponibles a deshabilitar, concluyendo así el caso de uso.

Cursos Alternos de los Eventos

Acción	**Curso Alterno**
3.	El Profesor oprime botón Cancelar.
3.1	El sistema muestra el Menú Principal concluyendo así el caso de uso.
4.	El Profesor oprime el botón “Cancelar”.
4.1	El sistema no deshabilita el contenido
4.2	El sistema muestra el Menú Principal concluyendo así el caso de uso.

Medias a utilizar	**Tipo de Media**	**Descripción**	**Estado**
	Imagen	Fondo del menú principal.	En construcción
		Icono representativo de cada opción del menú.	En construcción
		Iconos representativos de las opciones estándares en la aplicación	En construcción
		Imágenes a mostrar al tema en cuestión.	En localización.
	Video o Animación		
	Sonido	Sonido de pequeña duración para cuando se pase por encima de las	En localización

		opciones del menú principal.	
		Sonido para cuando se seleccione una opción del menú principal.	En localización
	Texto	Textos de las opciones.	En construcción.
		Texto de fondo.	En construcción
		Lista mostrada con los recursos a deshabilitar.	En construcción.
		Texto del mensaje.	En construcción.
Elementos pedagógicos			

Tabla 1: Descripción Textual del Caso de Uso "Deshabilitar Contenido".

2.5 Diagramas de Actividades.

Se realizaron nueve diagramas de Actividades de acuerdo a las descripciones textuales planteadas. Estos diagramas no sufren modificación alguna en la extensión de UML utilizada. Estos diagramas permiten visualizar mejor los flujos y ayudan al entendimiento del equipo de desarrollo. Como se muestra a continuación.

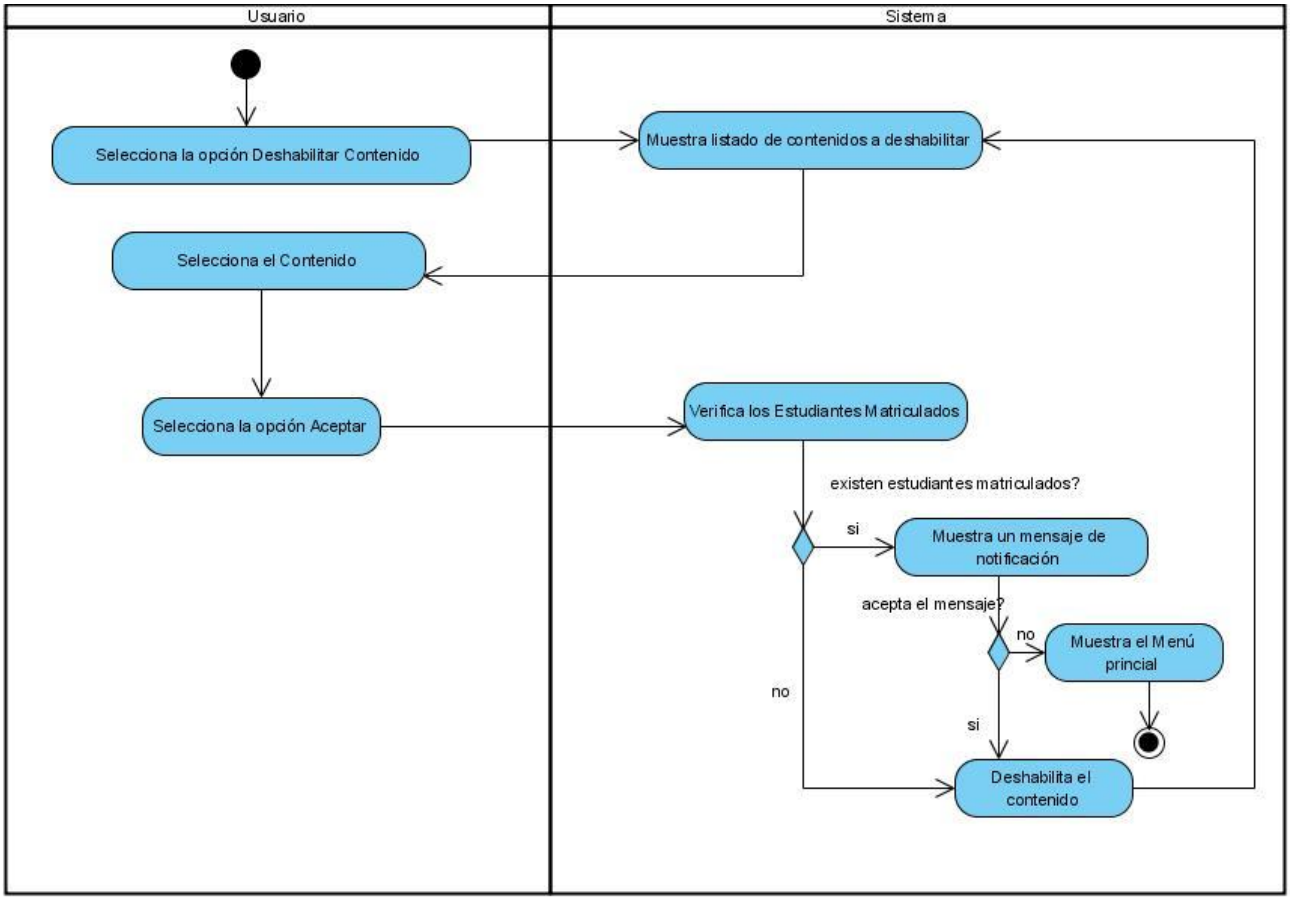

Fig. 7 Diagrama de Actividades del Caso de Uso Deshabilitar Contenido.

2.6 Diagramas de Clases.

Se realizaron nueve diagramas de clases describiendo cada una un conjunto de objetos que almacenan información y se comunican para implementar su comportamiento. La información almacenada se representa como atributos de estas clases y las operaciones a través de los métodos de dichas clases. Guiado bajo el lenguaje de modelado utilizado, estas tienen modificaciones por el modelo arquitectónico propuesto, el patrón MVC-E.

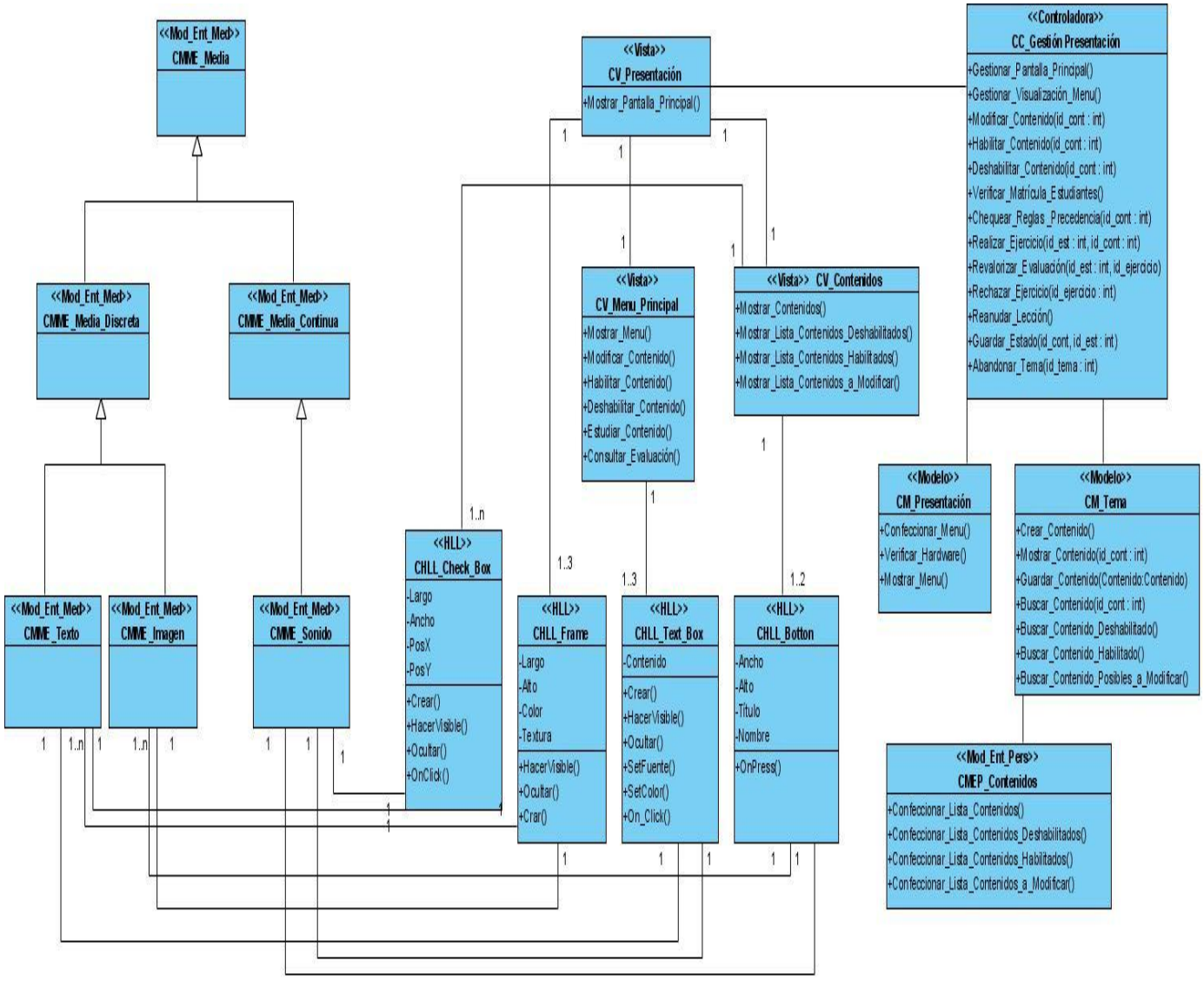

Fig. 8 Diagrama de Clases del Caso de Uso Deshabilitar Contenido.

2.7 Diagramas de Navegación.

Seguidamente quedaron definidos nueve diagramas de Navegación que son los nuevos diagramas incorporados por el lenguaje de modelado para aplicaciones educativas, los cuales tienen como objetivo mostrar las posibilidades de navegación que se hace necesario representar.

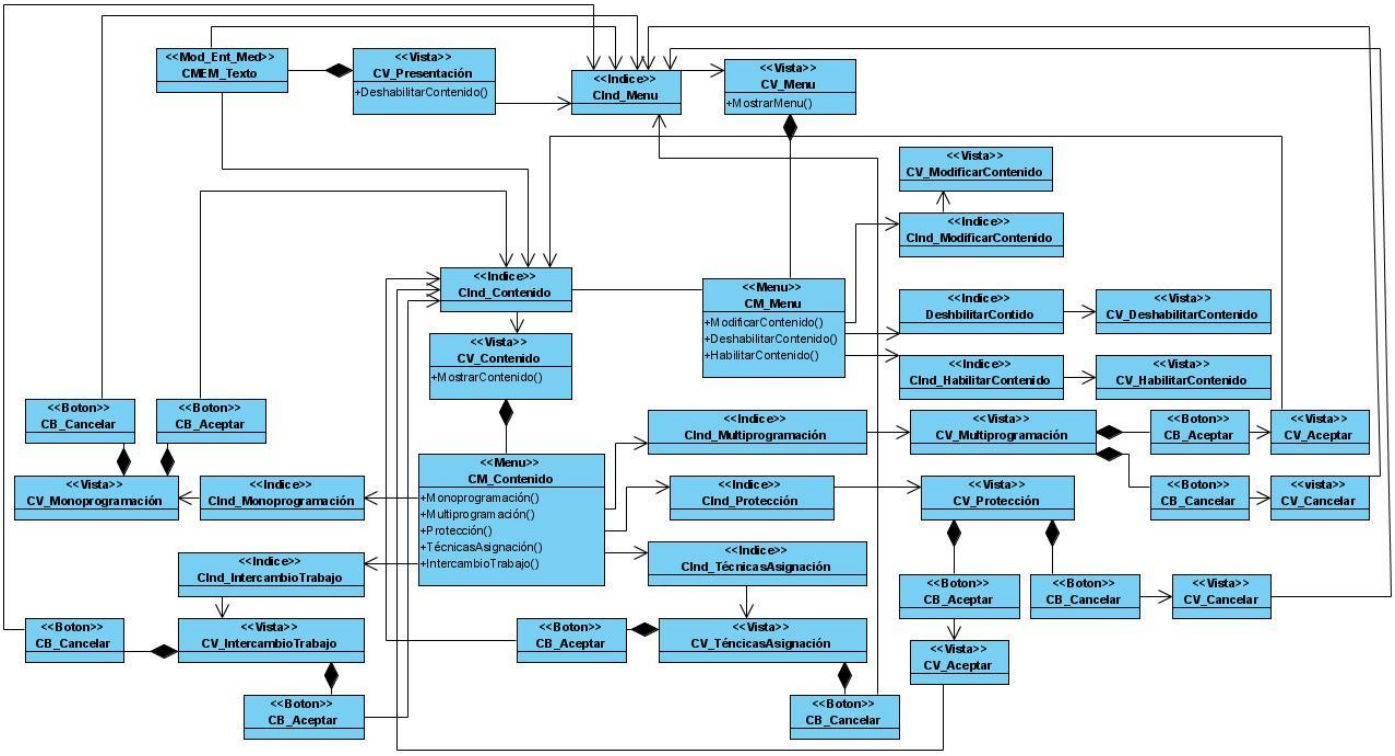

Fig. 9 Diagrama de Navegación del Caso de Uso Deshabilitar Contenido.

2.8 Diagrama de Presentación.

Se generaron nueve diagramas de Presentación, otro nuevo diagrama para el modelado de las aplicaciones multimedia el cual tiene como objetivo producir una versión ruda de la interfaz de usuario. Se dice ruda, por solo contener los elementos que tendrá visualmente la aplicación y su distribución espacial. A continuación se presenta el diagrama de Presentación concerniente al caso de uso Deshabilitar Contenido. Para los restantes diagramas de Presentación se pueden localizar en los anexos.

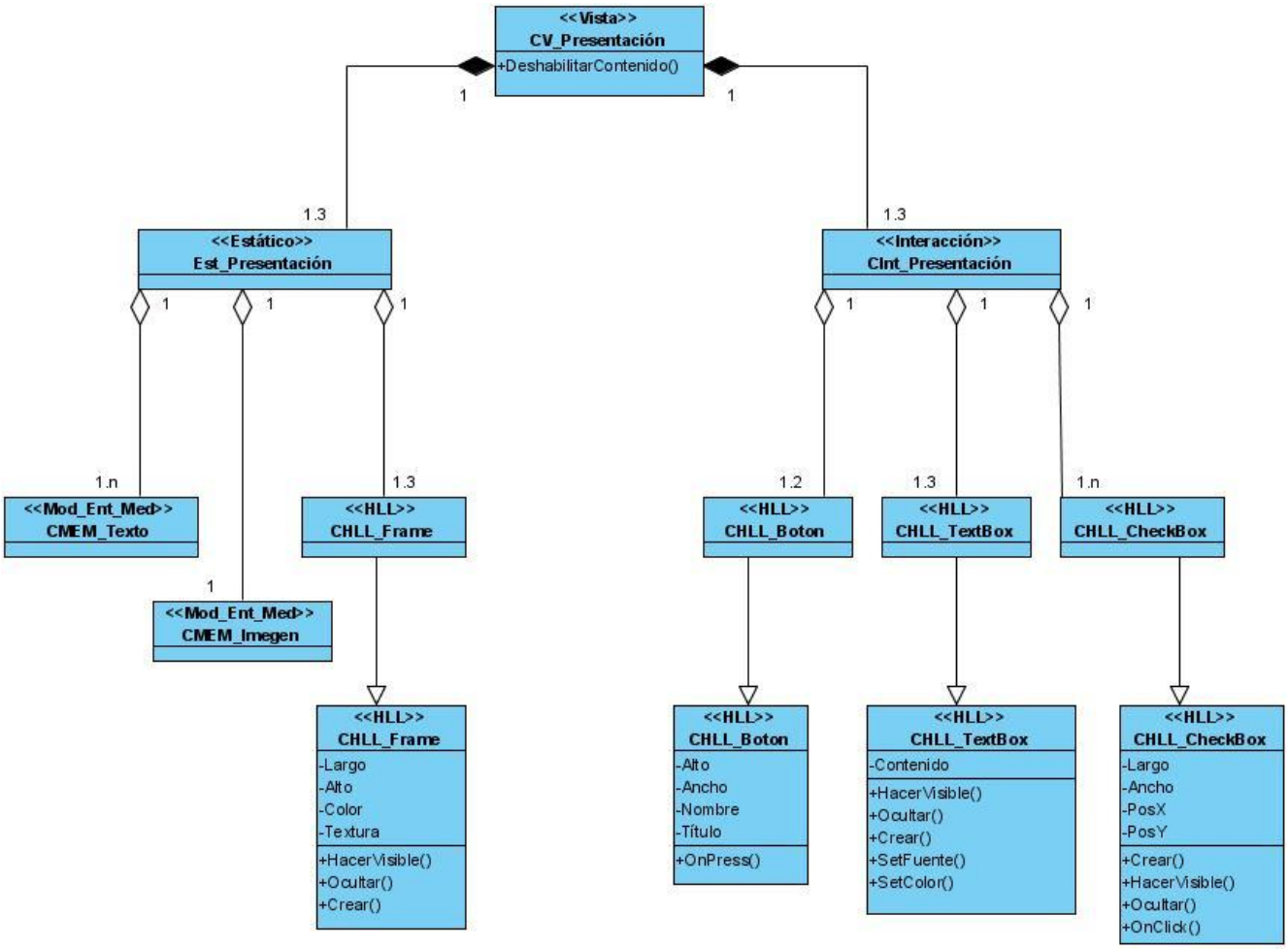

Fig. 10 Diagrama de Presentación del Caso de Uso Deshabilitar Contenido.

2.9 Diagramas de Secuencia.

De la misma manera se realizaron los Diagramas de Secuencia con las pequeñas modificaciones que se le hace por ApEM-L, estos diagramas de secuencia son también diagramas de interacción, los cuales tienen como objetivo destacar la línea de vida y el foco de control que representa el período de tiempo durante el cual un objeto ejecuta una acción. A continuación se muestra el diagrama de secuencia correspondiente al caso de uso Deshabilitar Contenido. Los restantes diagramas de secuencia se pueden localizar en los anexos.

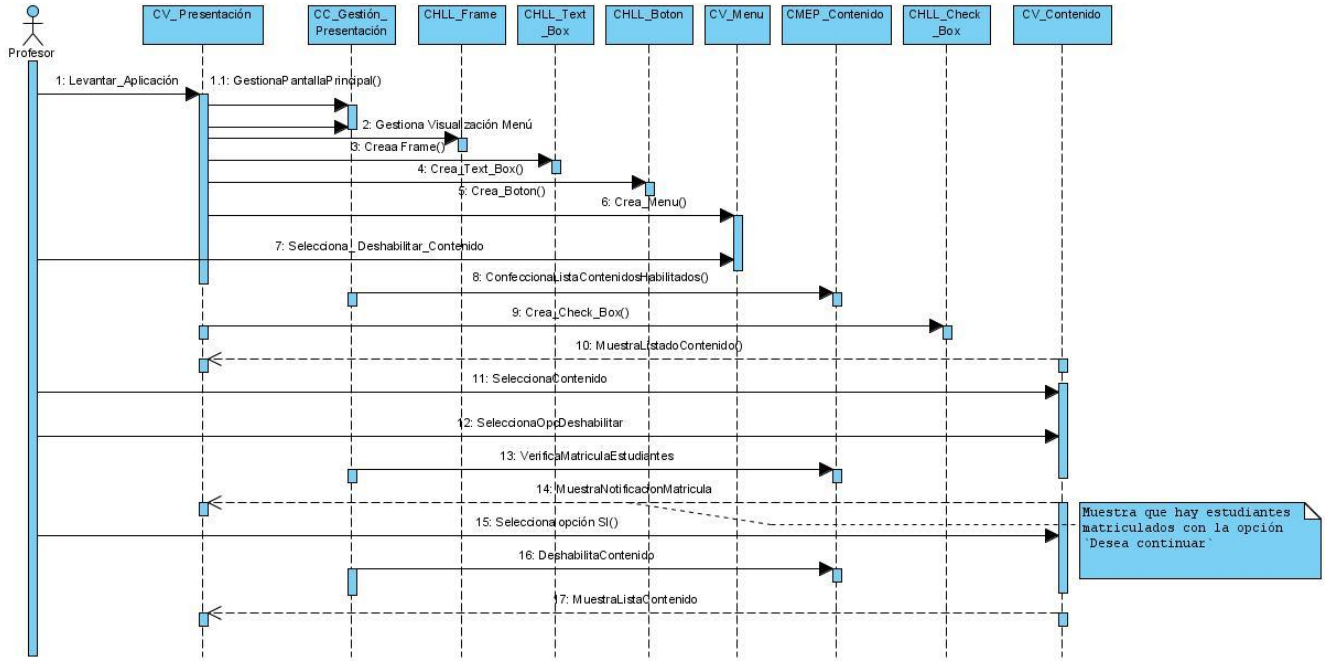

Fig. 11 Diagrama de Secuencia del Caso de Uso Deshabilitar Contenido.

Conclusiones Parciales.

En este capítulo se llevó a cabo la realización del Diagrama de caso de uso según los requisitos necesarios para una aplicación de este tipo, quedando generados los artefactos propuestos por este lenguaje de modelado para aplicaciones multimedia así como una breve descripción de en qué consisten cada uno de ellos. Quedaron diseñados los nuevos diagramas que se incorporan en este modelado que permiten definir una estructura para la navegación y una estructura para la presentación de este tipo de aplicaciones.

Capítulo 3. Validación y Prueba.

3.1 Introducción.

El presente capítulo contiene las métricas de validación aplicadas a la solución propuesta anteriormente, enmarcado en la aplicación de Métricas del Software, específicamente las Métricas de la Calidad de Especificación de Requisitos y de Casos de Uso.

El término de métrica define varios casos de medición. En este sentido, siendo una medida estadística (no cuantitativa) que se aplica a todos los aspectos de calidad de software. En la actualidad están presente un conjunto de métricas en el proceso de desarrollo de software.

3.2 Aplicación de Métricas del Software.

La aplicación de métricas permitirá realizar un refinamiento a la propuesta de solución e indicar la calidad de lo realizado. De manera general el uso de las métricas permitirá: caracterizar, evaluar y mejorar el producto.

La métrica para la calidad de la especificación permitirá evaluar la calidad del modelo de Especificación de Requisitos de Software. El cual es considerado con calidad cuando la especificación es “no ambigua, completa, consistente, ordenada por importancia y estabilidad, verificable, modificable y trazable” (Monzón., 2000).

3.2.1 Métricas para la Calidad de la Especificación de Requisitos.

La especificación de los requerimientos identificados fueron comprobados para evaluar su especificidad (no ambiguos) y estabilidad, a través de la aplicación de la métrica para la calidad de la especificación.

Para la aplicación de la métrica para el factor de especificidad, se debe tener en cuenta lo siguiente:

- Número total de requisitos: Nt.
    - Número de RF: Nf.
    - Número de RNF:Nnf.
- Calidad de la especificación: Q1.
    - Número de requisitos interpretados de la misma manera: Nui.

- Nt

Q1 = Nui / Nt.

El valor óptimo de Q1 es 1, lo que significa que mientras más cercano sea el valor a este, mayor calidad tendrá la especificación de los requerimientos.

Aplicación de la Métrica a la solución:

- Número total de requerimientos (Nt)
  - Número de RF(Nf)
  - Número de RNF (Nnf)
- Nf + Nnf = Nt ⟶14 + 8 = 22

Resultado de la aplicación de la métrica primera revisión:

- Nui / Nt = Q1 ⟶19/ 22= 0.86

El valor de Q1 resultó ser 0.86, siendo identificados 3 requisitos con ambigüedad. Demostrando que la descripción de los requerimientos del módulo de Paginación y Segmentación de autoaprendizaje para Sistemas Operativos ha sido realizada con un alto grado de especificidad (bajo grado de ambiguedad). Considerándose un valor alto según el rango establecido, pero se realiza una segunda revisión para lograr una calidad total en cuanto a la especificación de los requisitos.

Resultado de la aplicación de la métrica segunda revisión:

- Nui / Nt = Q1 ⟶22/ 22= 1

En esta segunda revisión se logran los objetivos propuestos en cuanto a la especificación de los requisitos, evidenciándose en los valores de umbrales siguientes:

Alta (0.90 <= **Q1** <= 1).

Media (0.80 <= **Q1** < 0.90).

Baja (0.7 <= **Q1** < 0.80).

La aplicación de la métrica para el factor de estabilidad tuvo en cuenta lo siguiente:

- Estabilidad (E).
- Requisitos Modificados (R_m).
- Total de Requisitos (R_t).

La estabilidad se determina mediante la siguiente fórmula: $E = \frac{Rt-Rm}{Rt}$

De igual manera el valor óptimo de E es 1, por lo que mientras más cercano a 1 sea el resultado, mayor valor de calidad en cuanto a estabilidad tendrá la especificación de requisitos.

Resultado de la aplicación de la métrica:

E= (R_t- R_m)/R_t

E=(14-0)/14

E= 1

Interpretación del resultado de la aplicación de la métrica.

El valor de E resultó ser 1, demostrando que la descripción de los requisitos del módulo de Paginación y Segmentación de autoaprendizaje para Sistemas Operativos ha sido realizada con un alto grado de estabilidad, según el rango establecido.

Alta (0.90 <= **E** <= 1).

Media (0.80 <= **E** < 0.90).

Baja (0.7 <= **E** < 0.80).

Para la revisión de la métrica aplicada a los requisitos se escogió a tres revisores, el ingeniero Carlos Yasmani Hidalgo García, Dailen Benítez y Yinett Hernández Hernández, el primero cliente de la propuesta presentada y las dos últimas ingenieras con experiencia en el rol de Analista.

3.2.2 Métricas para la validación de Casos de Uso.

En este epígrafe se aplican un conjunto de métricas orientadas a objetos para evaluar las siguientes propiedades de calidad: consistencia, correctitud, completitud y complejidad. Cada uno de estos factores

tendrá asociada una o más métricas, que establecen una medida cuantitativa del grado en que los factores presentan una pobre calidad.

Completitud: Grado en que se ha logrado detallar todos los casos de uso relevantes.

Consistencia: Grado en que los casos de uso del sistema describen las interacciones adecuadas entre el usuario y el sistema.

Correctitud: Grado en que las interacciones actor / sistema soportan adecuadamente el proceso del negocio.

Complejidad: Grado de claridad en la presentación de los elementos que describen el contexto y la claridad del sistema.

Para conseguir una certera validación de los casos de uso del sistema se realizaron dos revisiones, a continuación se presentan los resultados obtenidos:

Primera Revisión.

Atributo	**Factor**	**Métrica Asociada**	**Valor**
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes del usuario encargado de generar/modificar o consultar información?	Métrica 1. Número de roles relevantes omitidos. Umbral<10	Total de roles: 3 Número de roles relevantes omitidos: 0 Representa: 0%.
	Factor 2. ¿Se presenta una descripción resumida de todos los casos de uso?	Métrica 2. Número de casos de uso que no tiene descripción resumida. Umbral<10	Total de casos de uso:10 Número de casos de uso que no tiene descripción resumida: 0 Representa: 0%.

	Factor 3. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	Métrica 3. Número de requisitos omitidos por casos de uso. Umbral<10	Total de requisitos funcionales: 14 Número de requisitos omitidos por casos de uso: 1 Representa: 7.14 %.
		Métrica 4. Número de casos de uso que tienen requisitos omitidos. Umbral<10	Total de casos de uso: 10 Número de casos de uso que tienen requisitos omitidos: 1 Representa: 10 %.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en (crítico, importantes y útiles)?	Métrica 5. Número de casos que no han sido clasificados. Umbral<10	Total de casos de uso: 10 Número de casos que no han sido clasificados: 0 Representa: 0%.
			Para un incumplimiento: 4.3 %
Consistencia	Factor 5. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante	Métrica 6. Número de casos de uso que tienen un nombre incorrecto.	Total de casos de uso:10 Número de casos de uso que tienen un nombre incorrecto: 0

	en el contexto del usuario?	Umbral<10	Representa: 0%.
	Factor 6. ¿Está adecuadamente redactado (en el lenguaje de usuario) el flujo de eventos?	Métrica 7. Grado de adecuación de la descripción del flujo de eventos para un caso de uso. Umbral<10	Total de descripciones: 7 La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción: 0 Representa: 0%.
	Factor 7. ¿La descripción del flujo de eventos se inicia con la descripción de una acción extrema originada por un actor o por una condición interna del sistema claramente identificable?	Métrica 8. Número de casos de uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema. Umbral<10	Total de casos de uso:10 Número de casos de uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema: 0 Representa: 0%.
	Factor 8. ¿Existe una adecuada separación entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados?	Métrica 9. Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos. Umbral<10	Total de casos de uso: 10 Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos: 0 Representa: 0%.

			Para un incumplimiento de 0%
Correctitud	Factor 9. ¿Representa el caso de uso requisitos comprensibles por el usuario?	Métrica 10.Número de casos de uso en que los requisitos representados no son comprensibles por el usuario. Umbral<10	Total de casos de uso: 10 Número de casos de uso en que los requisitos representados no son comprensibles por el usuario: 0 Representa: 0%.
	Factor 10. ¿Las interacciones definidas describen la funcionalidad requerida del sistema?	Métrica 11. Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema. Umbral<10	Total de casos de uso:10 Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema: 0 Representa: 0%.
	Factor 11. ¿Las interacciones introducen mejoras al proceso actual?	Métrica 12. Número de casos de uso que deben ser modificados para mejorar el proceso actual. Umbral<10	Total de casos de uso:10 Número de casos de uso que deben ser modificados para mejorar el proceso actual: 1 Representa: 10 %.
			Para un incumplimiento:

			3.33%
Complejidad	Factor 12. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	Métrica 13. Número de elementos del diagrama que requieren reubicación. Umbral<30	Total de elementos del diagrama:13 Número de elementos del diagrama que requieren reubicación: 1 Representa: 7.7%.
			Para un incumplimiento de: 7.7 %.

Tabla 2 Métricas aplicadas al diagrama de Casos de Usos.

Con la aplicación de las métricas se observa que los resultados que se obtienen no son del todo relevantes en cuanto a los atributos de calidad, representando un 95.7% de Completitud, un 100% de Consistencia, un 97% de Correctitud y un 92,3% de Complejidad. Para mejorar estos resultados se realiza una 2da revisión aplicando nuevamente las métricas.

Segunda Revisión.

Atributo	**Factor**	**Métrica Asociada**	**Valor**
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes del usuario encargado de generar/modificar o consultar información?	Métrica 1. Número de roles relevantes omitidos. Umbral<10	Total de roles: 3 Número de roles relevantes omitidos: 0 Representa: 0%.

	Factor 2. ¿Se presenta una descripción resumida de todos los casos de uso?	Métrica 2. Número de casos de uso que no tiene descripción resumida. Umbral<10	Total de casos de uso:10 Número de casos de uso que no tiene descripción resumida: 0 Representa: 0%.
	Factor 3. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	Métrica 3. Número de requisitos omitidos por casos de uso. Umbral<10	Total de requisitos funcionales: 14 Número de requisitos omitidos por casos de uso: 0 Representa: 0 %.
		Métrica 4. Número de casos de uso que tienen requisitos omitidos. Umbral<10	Total de casos de uso: 10 Número de casos de uso que tienen requisitos omitidos: 0 Representa: 0%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en (crítico, importantes y útiles)?	Métrica 5. Número de casos que no han sido clasificados. Umbral<10	Total de casos de uso: 10 Número de casos que no han sido clasificados: 0 Representa: 0%.
			Para un incumplimiento

			de 0 %
Consistencia	Factor 5. ¿El nombre dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	Métrica 6. Número de casos de uso que tienen un nombre incorrecto. Umbral<10	Total de casos de uso:10 Número de casos de uso que tienen un nombre incorrecto: 0 Representa: 0%.
	Factor 6. ¿Está adecuadamente redactado (en el lenguaje de usuario) el flujo de eventos?	Métrica 7. Grado de adecuación de la descripción del flujo de eventos para un caso de uso. Umbral<10	Total de descripciones: 7 La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción: 0 Representa: 0%.
	Factor 7. ¿La descripción del flujo de eventos se inicia con la descripción de una acción extrema originada por un actor o por una condición interna del sistema claramente identificable?	Métrica 8. Número de casos de uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema. Umbral<10	Total de casos de uso:10 Número de casos de uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema: 0 Representa: 0%.
	Factor 8. ¿Existe una adecuada separación	Métrica 9. Número de casos de uso	Total de casos de uso: 10 Número de casos de uso

	entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados?	complejos que no tienen separación del flujo básico y de flujos alternos. Umbral<10	complejos que no tienen separación del flujo básico y de flujos alternos: 0 Representa: 0%.
			Para un incumplimiento de 0%
Correctitud	Factor 9. ¿Representa el caso de uso requisitos comprensibles por el usuario?	Métrica 10.Número de casos de uso en que los requisitos representados no son comprensibles por el usuario. Umbral<10	Total de casos de uso: 10 Número de casos de uso en que los requisitos representados no son comprensibles por el usuario: 0 Representa: 0%.
	Factor 10. ¿Las interacciones definidas describen la funcionalidad requerida del sistema?	Métrica 11. Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema. Umbral<10	Total de casos de uso:10 Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema: 0 Representa: 0%.
	Factor 11. ¿Las interacciones introducen mejoras al proceso	Métrica 12. Número de casos de uso que deben ser modificados	Total de casos de uso:10 Número de casos de uso que deben ser

	actual?	para mejorar el proceso actual. Umbral<10	modificados para mejorar el proceso actual: 0 Representa: 0 %.
			Para un incumplimiento: 0%
Complejidad	Factor 12. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	Métrica 13. Número de elementos del diagrama que requieren reubicación. Umbral<30	Total de elementos del diagrama:13 Número de elementos del diagrama que requieren reubicación: 0 Representa: 0%.
			Para un incumplimiento de 0 %.

Tabla 3 Métricas aplicadas al diagrama de Casos de Usos

En los resultados de la segunda revisión se observa una diferencia relevante en comparación con la primera revisión realizada, con un alto grado de 100% de cumplimiento en los atributos de calidad. Estos resultados reflejan una mejora en la calidad del artefacto luego de corregir las no conformidades detectadas en la iteración anterior y se considera que el mismo posee un alto grado de funcionalidad. Como se muestra en la siguiente gráfica:

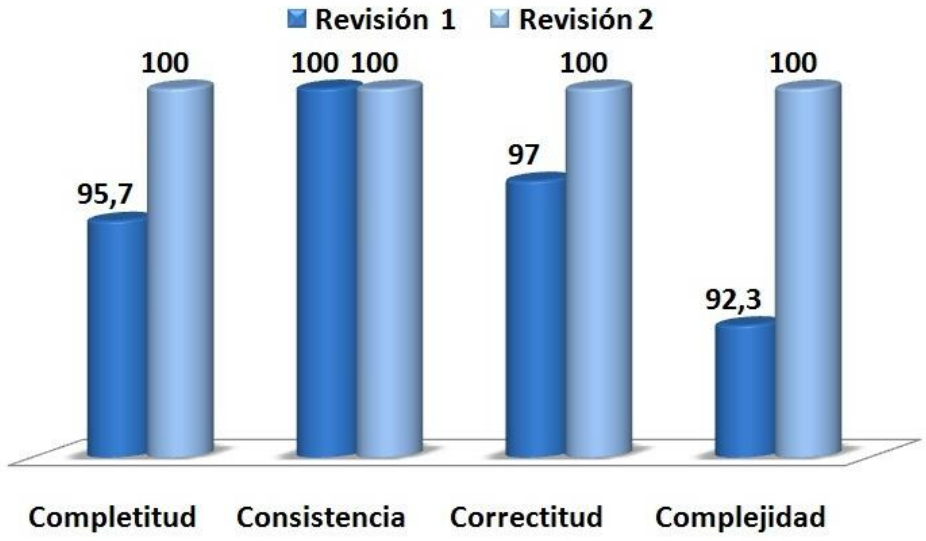

Fig. 12 Gráfica del resultado de las Revisiones de Casos de Usos.

Matriz de Trazabilidad:

Para comprobar que cada requisito esté relacionado con al menos un caso de uso, se realiza la Matriz de Trazabilidad, siendo el mecanismo que permite lograr este resultado.

	CU1	CU2	CU3	CU4	CU5	CU6	CU7	CU8	CU9	CU10
RF1						X				
RF2	X									
RF3		X								
RF4			X							
RF5								X		
RF5.1								X		

RF5.2					X				
RF5.3					X				
RF5.4					X				
RF6			X						
RF7							X		
RF8				X					
RF9			X						
RF10									X

Tabla 4 Matriz de Trazabilidad de Caso de Uso.

Quedaron representados en la tabla anterior la relación que establece cada requisito con uno o más casos de uso, demostrando que todos los requisitos identificados están involucrados en el diagrama de caso de uso para cumplir con las necesidades del cliente.

3.2.3 Métricas para validar el Diseño.

Las métricas del producto son una medida cuantitativa que permite tener una visión profunda de la eficacia del proceso del software. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar estrategias para mejorar la calidad del desarrollo del software.

Métrica de Tamaño Operacional de Clases. (TOC)

La métrica de tamaño operacional de clases (TOC) está condicionada por el número de métodos declarados para las clases del sistema y su resultado es reflejo de los atributos de calidad (AC):

Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

Complejidad de implementación: Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.

Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

Atributo de calidad	Modo en que lo afecta
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 5 Indicadores de la métrica TOC.

Los criterios de evaluación relacionados con el número de clases se encuentran relacionados con los AC, los que se ven afectados de acuerdo a los valores medios calculados en la métrica.

Atributo	Categoría	Criterio
Responsabilidad	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio
Complejidad de implementación	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

Reutilización	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	<=Promedio

Tabla 6 Rango de valores evaluados en TOC.

Para la aplicación de esta métrica se tiene un total de 13 clases, registrándose un total de 58 operaciones como se muestra en la siguiente tabla.

Clase	**Cantidad de Procedimientos**	**Responsabilidad**	**Complejidad**	**Reutilización**
CV_Presentación	1	Baja	Baja	Alta
CV_Menu_Principal	6	Media	Media	Media
CV_Contenidos	4	Baja	Baja	Alta
CC_Gestión_Presentación	13	Alta	Alta	Baja
CM_Presentación	3	Baja	Baja	Alta
CM_Temas	7	Media	Media	Media
CMEP_Contenidos	4	Baja	Baja	Alta
CHLL_Botton	1	Baja	Baja	Alta
CHLL_Text_Box	6	Media	Media	Media
CHLL_Frame	3	Baja	Baja	Alta
CHLL_Check_Box	4	Baja	Baja	Alta

CV_Contenidos_Modo_Disenno	3	Baja	Baja	Alta
CC_Subsistema_Ejercicio	0	Baja	Baja	Alta

Tabla 7: Indicadores relacionados al número de procedimientos.

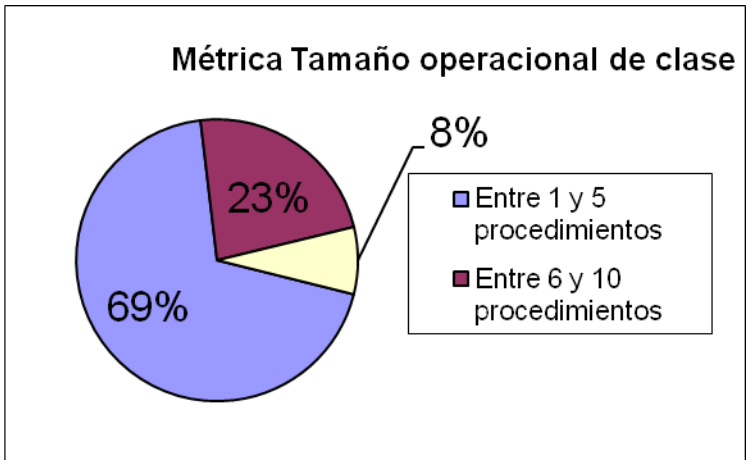

Fig. 13 Gráfica de Procedimientos de las Clases.

Como se observa en la figura 5 el 69 por ciento de las clases modeladas tiene entre 1 y 5 procedimientos lo que significa un tamaño operacional bajo y solamente una clase (la controladora) tiene más de 11 procedimientos. Este resultado permite contar con altos índices de reutilización y bajos valores de responsabilidad y complejidad de implementación. Las siguientes gráficas muestran los resultados detallados de dichos AC.

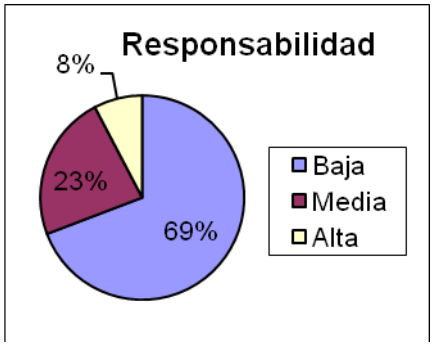

Fig. 14 Gráfica del AC Responsabilidad.

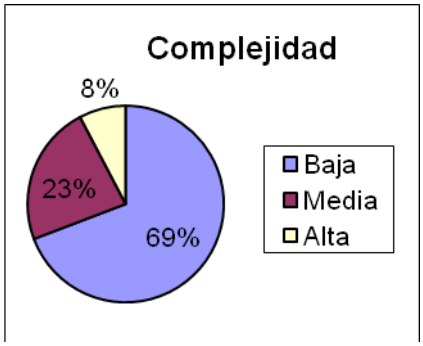

Fig. 15 Gráfica del AC Complejidad.

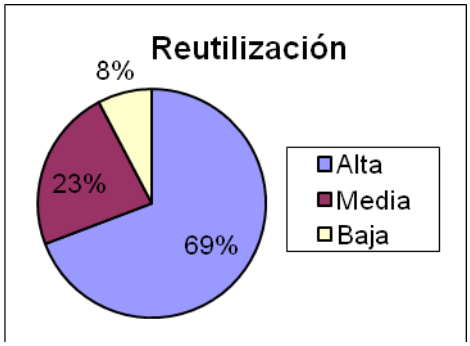

Fig. 16 Gráfica del AC Reutilización.

Métrica de Relaciones entre Clases. (RC)

La métrica de RC está condicionada por el número de relaciones declaradas en las clases del sistema y su resultado es reflejo de los AC:

Responsabilidad: Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.

Complejidad del mantenimiento: Consiste en el grado de esfuerzo necesario a realizar para desarrollar un arreglo, una mejora o una rectificación de algún error de un diseño de software. Puede influir indirecta, pero fuertemente en los costes y la planificación del proyecto.

Reutilización: Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

Cantidad de pruebas: Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (Unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Los criterios de evaluación relacionados con el número de clases se encuentran relacionados con los AC, que se ven afectados de acuerdo a los valores medios calculados en la métrica.

Atributo de Calidad.	**Modo en que lo afecta.**
Responsabilidad.	Un aumento del RC implica un aumento de la responsabilidad asignada a la clase.

Complejidad del mantenimiento.	Un aumento del RC implica un aumento en la complejidad del mantenimiento de la clase.
Reutilización.	Un aumento del RC implica una disminución del grado de reutilización de la clase.
Cantidad de pruebas.	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 8 Tabla de Atributos de calidad de la métrica TOC.

Para los cuales están definidos los siguientes criterios y categorías de evaluación:

Atributo.	Categoría.	Criterio.
Acoplamiento.	Ninguno.	0
	Bajo.	1
	Medio.	2
	Alto.	>2
Complejidad de mantenimiento.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio
Reutilización.	Baja.	>2*Promedio
	Media.	Entre Promedio y 2*Promedio

	Alta.	<=Promedio
Cantidad de pruebas.	Baja.	<=Promedio
	Media.	Entre Promedio y 2*Promedio
	Alta.	>2*Promedio

Tabla 9 Categorías y criterios de los atributos de Calidad.

La siguiente gráfica permite observar que solamente el 15 por ciento de las clases modeladas tiene una cantidad de dependencias superior a dos que es el umbral definido para considerar que un valor es alto; ese por ciento corresponde solamente a dos clases: *CV_Menu_Principal* y *CHLL_Text_Box* las cuales fueron revisadas y se comprobó que todas sus dependencias son necesarias para el correcto modelado de la solución.

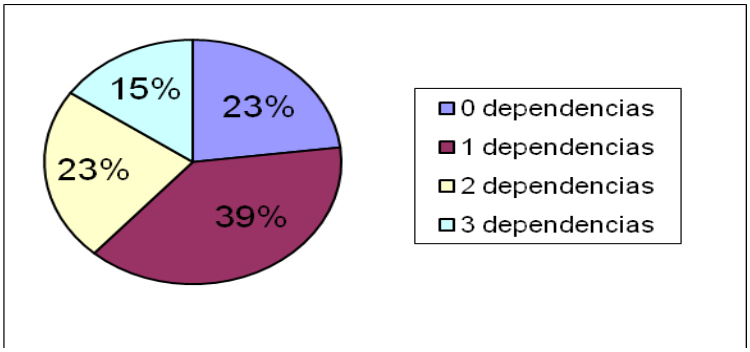

Fig. 17 Gráfica de los resultados de la métrica TOC.

Estos resultados se traducen en bajos índices de acoplamiento, complejidad de mantenimiento y cantidad de pruebas permitiendo una amplia reutilización. Las siguientes figuras muestran detalladamente el comportamiento de estos atributos de calidad.

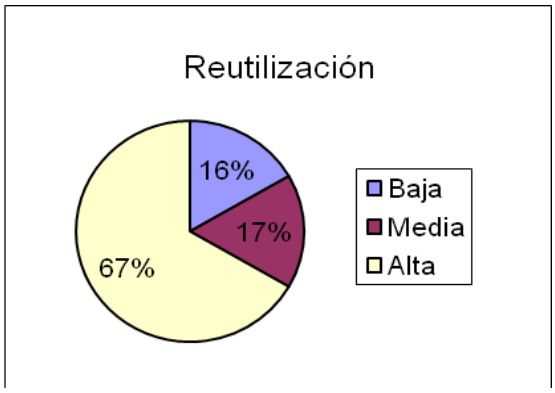

Fig. 18 Gráfica del atributo de calidad Reutilización.

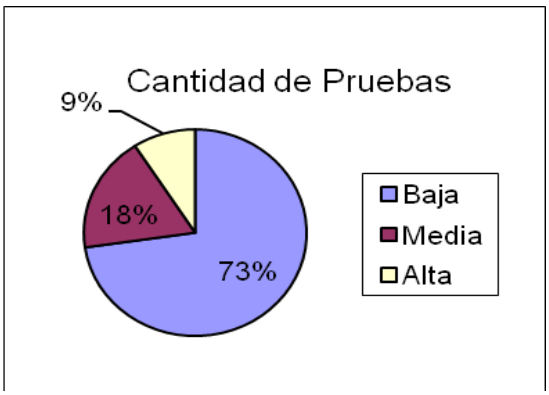

Fig. 19 Gráfica del atributo de calidad Cantidad de Pruebas.

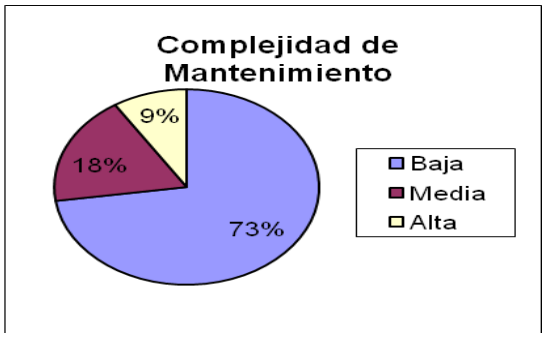

Fig. 20 Gráfica del atributo de calidad Complejidad M.

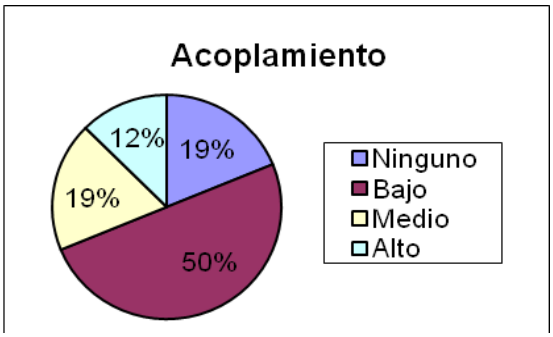

Fig. 21 Gráfica del atributo de calidad Acoplamiento.

Conclusiones Parciales.

Todo lo expuesto anteriormente permitió la validación de la calidad de especificación de requisitos y de casos de uso mediante métricas y el diseño. Mediante la fórmula para la especificación de requisitos queda evidenciado el alto grado de especificidad obtenido. Con la aplicación de las métricas para el diagrama de caso de uso, se comprobó que dicho artefacto fue generado con un alto grado de calidad aplicando las diferentes métricas. El diseño fue validado satisfactoriamente demostrando que la dependencia entre clases es baja lo que permite una mayor reutilización de las mismas.

Conclusiones

Conclusiones.

- Se obtuvieron los conocimientos teóricos necesarios para el desarrollo del trabajo en cuanto a laboratorios virtuales, sistemas de autoaprendizaje y metodologías, que permitieron la correcta selección de las herramientas para el desarrollo de la solución propuesta. Conformando el marco teórico de la investigación.
- Se realizó el análisis y diseño del módulo de Paginación y Segmentación del subsistema de autoaprendizaje para el laboratorio virtual de Sistema Operativo, obteniendo la documentación necesaria para garantizar el futuro mantenimiento del sistema.
- Se aplicaron las métricas de validación necesaria para lograr la calidad en el análisis y diseño, logrando así erradicar la mayor cantidad de errores y de esta forma alcanzar una alta calidad para el módulo de autoaprendizaje del laboratorio virtual de Sistema Operativo.

Recomendaciones

Recomendaciones.

Se recomienda para investigaciones futuras:

- Realizar un estudio de nuevas funcionalidades que puedan incluirse en el subsistema de autoaprendizaje del Laboratorio Virtual de Sistema Operativo.
- Realizar la implementación correspondiente al diseño realizado para una aplicación educativa.

Bibliografía.

Agust, Yadira Pérez. 2008. *Modelado de negocio y gestión de requisitos como etapas imprescindibles para el desarrollo de los sistemas automatizados de información.* Habana : s.n., 2008.

ALBA LUZ LONDOÑO GARCÍA, BEATRIZ ELENA GIRALDO TOBÓN , MARÍA DEL PILAR RÍOS ROLDÁN. 2004. *EL IMPACTO DE LA GESTIÓN DE NEGOCIOS E INFORMÁTICA EN LA INSTITUCIÓN EDUCATIVA DE MARÍA.* YARUMAL : s.n., 2004.

Alfredo J. Simón Cuevas, Boris Piñero Suárez, Danaisy Ruiz Bravo. 2004. *LAS TIC Y LOS MAPAS CONCEPTUALES EN FUNCIÓN DE POTENCIAR LA GESTIÓN DEL CONOCIMIENTO Y EL APRENDIZAJE. .* Habana : s.n., 2004.

autores, Colectivo de. 2009. *Sitio Web oficial Visual-Paradigm.* 2009.

Autores, Colectivo de. 2009. *Sitio Web oficial Visual-Paradigm.* 2009.

autores., Colectivo de. 2009. *Sitio Web oficial Visual-Paradigm. Visual Paradigm Organización.* 2009.

Baumeistier, Hubert, Koch, Nora y Mandel, Luis. 2001. *Towars a UML Extension for Hypermedia Design.* 2001.

Castells., Manuel. 2000. *La Era de la información. .* Madrid : Alianza Editorial, 2000.

Dairelis Pérez González, Grenia Hernández Pérez. 2010. *Guía para realizar Ingeniería de Requisitos a la línea de productos informáticos que utilizan tecnología multimedia.* Habana, Universidad de las Ciencias Informáticas : s.n., 2010.

David, Merrill. 2002. *Position Statement and Questions on Learning Objects Research and Practice.* Utha State University : s.n., 2002.

Estrada, Julián Monge Nájera y Víctor Hugo Méndez. 2007. *Ventajas y desventajas de usar laboratorios virtuales en la educación.* 2007.

—. 2007. *Ventajas y desventajas de usar laboratorios virtuales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración.* 2007.

Bibliografía

Graells, Dr. Pere Marquès. 2000. *IMPACTO DE LAS TIC EN EDUCACIÓN: FUNCIONES Y LIMITACIONES.* 2000.

Herías, Francisco Andrés Candelas. 2003. *Propuesta de Portal de la Red de Laboratorios Virtuales y Remotos de CEA.* 2003.

Herreros, L. Rosado. 2005. *Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física.* 2005. .

INNOVA, Grupo de soluciones. 2007 . *Herramientas y soluciones .* 2007 .

Koch, Escalona y Nora. 2004. *Ingeniería de Requisitos en Aplicaciones para la Web un estudio comparativo.* 2004.

Landazuri, Bárbara A. 2005. *Definición de Perfiles en Herramientas de Gestión de Requisitos.* 2005.

Malumbres, Ana Belén. 2007. *IRqA para realizar una gestión efectiva de los requisitos de sus sistemas.* 2007.

Monzón., Antonio. 2000. *CALIDAD DE LA ESPECIFICACIÓN: ¿SE PUEDEN MEDIR LOS REQUISITOS? .* 2000.

Nannette Napier y Mathiassen, Lars. 2009. *Combining Perceptions and Prescriptions in Requirements Engineering Process Assessment: An Industrial Case Study.* 2009.

Noelia Álvarez Yenes, Patricia González Saiz. 2007. *Todo sobre pedagogía, educación.* Bilbao : s.n., 2007.

Orallo, Enrique Hernández. 2007. *El lenguaje Unificado de Modelado (UML).* 2007.

Pescador, Elena Sánchez. 2009. *Análisis e implementación de una herramienta de Gestión de Requisitos para la Gestión de Servicios.* España : s.n., 2009.

Pressman, Roge. 2005. *Un enfoque práctico.5ta Edición.* 2005.

Pressman, Roger S. *Ingeniería del Software. Un enfoque práctico Sexta Edición. .*

Ricardo, Febe Angel Ciudad. Julio de 2007. *ApEM – L como una nueva solución a la modelación de aplicaciones educativas multimedia en la UCI.* La Habana : s.n., Julio de 2007.

Bibliografía

Rojas, M.C. Juan Carlos Olivares. *Patrones del Diseño.* México : s.n.

Ruz, Fidel Catro. 2002. 2002.

Sánchez, María A. Mendoza. 2004. . *Metodologías De Desarrollo De Software. Metodologías De Desarrollo De Software.* 2004. .

Sánchez., José Ignacio Peláez. 2005. *Lenguajes y Ciencias de la Comunicación en el desarrollo de sistemas.* 2005.

Saúl González Medina, Susana Martínez Ríos del Río. 1989. *Reflexiones sobre el Autoaprendizaje.* California : s.n., 1989.

Urrutia, Lianni Y. Figueredo Jiménez y Misael Rodríguez. 2010. *Simulador para la asignatura de Sistema Operativo.* Habana, Universidad de las Ciencias Informáticas : s.n., 2010.

Vary, James. 2000. *Informe de la reunión de expertos. Informe de la reunión de expertos.* 2000.

Anexos

Glosario de Términos

Multimedia: Consiste en una nueva forma de presentar la información, es una plataforma donde se integran diferentes componentes para realizar diferentes tareas, su gran impacto es al incorporar imágenes, efectos de sonido, video y animación en tercera dimensión para crear presentaciones vivas y de extraordinaria calidad.

Hipermedia: La hipermedia surge como resultado de la fusión de dos tecnologías, el hipertexto y la multimedia. Es la tecnología que permite estructurar la información de una manera no-secuencial, a través de nodos interconectados por enlaces sobre los diferentes formatos de información.

Lenguaje de Modelado: Identifican la acción de establecer un conjunto de signos, reglas, normas y semántica para la representación de la estructura y el comportamiento de los sistemas informáticos, permitiendo de esta forma la homogenización de los términos utilizados por el equipo de desarrollo y la construcción de modelos de representación del futuro software.

Sistema Operativo: Programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permite la normal ejecución del resto de las operaciones.

Clase: Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase).

Arquitectura de software: Consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

Diseño de Software: El proceso de aplicar distintas técnicas y principios con el propósito de definir un producto con los suficientes detalles como para permitir su realización física. El diseño es la primera etapa técnica del proceso de Ingeniería del Software, consiste en producir un modelo o representación técnica del software que se va a desarrollar.

Patrones de diseño: (Design patterns) son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Un patrón de diseño es una solución a un problema de diseño.

Requisito: Condición o capacidad que debe cumplir un sistema**.**

Métricas de Calidad: Es el término que describe muchos y muy variados casos de medición. Siendo una métrica una medida estadística (no cuantitativa como en otras disciplinas ejemplo física) que se aplica a todos los aspectos de calidad de software, los cuales deben ser medidos desde diferentes puntos de vista como el análisis, construcción, funcionalidad, documentación, métodos, proceso, usuario, entre otros.

Validación: Confirmación mediante examen y aportación de pruebas objetivas que se cumplen los requisitos concretos para su uso determinado.