Universidad de las Ciencias Informáticas

Facultad 3

Análisis y Diseño del Módulo de Entrada y Salida del subsistema de autoaprendizaje para el Laboratorio Virtual de Sistemas Operativos

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

Autora: Lianet Estrada Hidalgo

Tutor: Ing. Carlos Y. Hidalgo García

Ciudad de la Habana, Cuba

Junio, 2011

"…aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos".

Declaración de Autoría

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y se le reconoce 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 junio del año 2011

_____

Firma del autor

Lianet Estrada Hidalgo

_____

Firma del tutor

(Carlos Y. Hidalgo García)

Agradecimientos

A mi mamá por su amor incondicional y siempre estar conmigo en todo momento.

A mi hermana Misladis, por estar siempre presente en todos los momentos más difíciles en estos cinco años.

A mi tía Marta, mi segunda mamá, por su preocupación, apoyo, cariño y siempre estar pendiente de mí.

A mi tío, papá Mana, por preocuparse y brindarme tanto cariño, más que tío es un padre para mí.

A mi prima Yadira, por su preocupación y apoyarme tanto en estos cinco años.

A mi papá, por guiarme por buen camino.

A mi tía Julia, por su preocupación.

A mi primo Julio, por apoyarme y ser mi médico en todos estos años.

A mi primo Nolberto y mi prima Yaima por apoyarme siempre.

A mi amigo Ramsés, por ayudarme tanto en estos años, ser mi profesor y arreglar las cosas que se me rompían.

A mi amiga Aylin por ayudarme en la tesis y soportarme.

A todos los amigos que he tenido en estos cinco años y que de una forma o de otra me han ayudado y apoyado.

A la Revolución y a Fidel por darme la oportunidad de realizar mi sueño.

.

Dedicatoria

En especial a mi mamá por su sacrificio, por creer en mí, porque nunca será suficiente lo que haga por ella.

A papá Mana, que siempre ha sido un padre para mí y lo quiero muchísimo.

A mi tía Marta, por creer y tener tanta fe en mí.

A mi hermana Misladis y mis sobrinas Yenifer y Diana, que son las cosas más lindas que tengo.

A mi papá, por guiarme y apoyarme.

A toda mi familia en general por su apoyo, amor y cariño.

Resumen

Hoy en día las tecnologías informáticas y las comunicaciones juegan un papel fundamental dentro de la educación, pues ha permitido crear condiciones favorables para facilitar el aprendizaje, por lo que han surgido los recursos didácticos en el mundo. Dentro de estos recursos podemos encontrar los Laboratorios Virtuales, siendo una herramienta de mucha ayuda al trabajo docente tanto para el estudiante como para el profesor. Una de estas herramientas es el EVA, dentro del que está montado el sistema de aprendizaje de la asignatura de Sistemas Operativos en la Facultad 3 de la Universidad de Ciencias Informáticas, el cual, no permite el acceso a los servicios brindados por este al estudiante sin la previa autorización del profesor, además, no cuenta con todas las facilidades que pueden hacer que la calidad del aprendizaje sea mayor. Por estas razones el presente trabajo está enmarcado en realizar el diseño del módulo de Entrada y Salida del subsistema de autoaprendizaje para el Laboratorio Virtual de Sistemas Operativos donde el estudiante pueda adquirir y practicar los conocimientos obtenidos por su propio autoaprendizaje.

Para el desarrollo del trabajo se realizó un estudio de conceptos fundamentales como: Laboratorios Virtuales, Recursos Didácticos, Objetos de Aprendizaje y el estándar SCORM. Además se tratan aspectos fundamentales para el desarrollo del análisis y diseño de la aplicación como, el lenguaje y herramienta de modelado, la ingeniería de requisitos, patrones de diseño, arquitectónico y de casos de uso. Finalizando con las métricas de validación aplicadas.

Palabras Claves: Laboratorios Virtuales, Objetos de Aprendizaje, Recursos Didácticos, Sistemas de Autoaprendizaje.

Índice de Contenido

Índice de Contenido

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

La informática es una de las ciencias que se encuentra en completa 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 Universidad de las Ciencias Informáticas (UCI).

La UCI se ha visto inmersa en un aumento de las responsabilidades productivas y la asignación de tareas, ante la inmediata necesidad de convertirse en el motor impulsor del desarrollo de software en Cuba. Con el aumento de la matrícula de estudiantes y profesores vinculados a las labores productivas se concibió un nuevo modelo de formación siguiendo los siguientes principios: centrado en el aprendizaje, establecimiento de un ciclo de formación básico y otro profesional en el cual tienen un uso protagónico los Entornos Virtuales de Aprendizaje (EVA).

El uso del EVA en la UCI para el proceso de enseñanza y aprendizaje es creciente, en el que se destaca la utilización de los software de simulación para llevar a cabo el proceso docente - educativo en las distintas materias. En tal sentido es importante destacar el rol que desempeñan los laboratorios virtuales, siendo una significativa herramienta de apoyo al trabajo docente, tanto para el profesor como para el estudiante.

La asignatura Sistema Operativo 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: Procesos, Memoria y Entrada / Salida.

Se ha podido observar que la asignatura en la actualidad carece de una fuerte integración con el desarrollo de las Tecnologías de la Informática y las Comunicaciones (TIC), lo que ha llevado a que las actividades sean en su mayoría presenciales, entrando en contradicción con el actual modelo de formación propuesto, donde la semipresencialidad, la amplia y adecuada utilización de las TICs son los componentes principales para lograr la integración formación – producción - investigación, donde el estudiante debe ser capaz de realizar un autoestudio y ser el protagonista de su propio proceso de aprendizaje.

Introducción

Dentro de la concepción del Laboratorio Virtual de Sistemas Operativos el contenido referente al tema de Entrada y Salida se encuentra reflejado en un simulador, que permite que el estudiante realice prácticas para reforzar los contenidos recibidos en el aula, esto es lo que conlleva a que cada ejercicio o contenido que el estudiante pueda desarrollar deba ser orientado por el profesor no se tuvo en cuenta en la concepción inicial, la idea es desarrollar un módulo de autoaprendizaje que permita que el estudiante no necesite completamente de la orientación del profesor para poder utilizar los simuladores correspondientes a cada tema y pueda ir directamente al Laboratorio Virtual, utilizar el módulo de autoaprendizaje donde a partir de las orientaciones y ejercicios propuestos, hacer un uso eficiente del mismo para poder comprobar o poner en práctica los conocimientos teóricos correspondientes al tema en cuestión y darle al profesor el papel de asesor o guía metodológico tal y como se establece en el nuevo modelo de formación.

Para lograr un correcto desarrollo del módulo se hace imprescindible realizar un análisis con el fin de identificar las necesidades reales del cliente y se traduzcan al lenguaje de los desarrolladores los resultados de ese análisis teniendo en cuenta que ellos no poseen todos los elementos respecto a la necesidad del cliente y por lo tanto tiene claro lo qué debe y qué no debe hacer el sistema. Es decir, se necesita llegar a un acuerdo entre los involucrados sobre lo que el software debe hacer para así proporcionar a los desarrolladores un mejor entendimiento de los requisitos del software.

Bajo estas condiciones es identificado el siguiente **problema de la investigación**: ¿Cómo lograr un entendimiento entre clientes y desarrolladores del tema referido a Entrada y Salida para el subsistema de autoaprendizaje del Laboratorio Virtual de Sistemas Operativos, 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 la modelación del módulo Entrada y Salida del Subsistema de autoaprendizaje para el Laboratorio Virtual de Sistemas Operativos garantizando el desarrollo posterior de dicho módulo, cuyo **campo de acción** es: Diseño dentro del proceso de desarrollo de software para aplicaciones didácticas multimedia.

Se define como **idea a defender:** Realizando la modelación del tema Entrada y Salida para el subsistema de autoaprendizaje del Laboratorio Virtual de Sistemas Operativos se logrará implementar el subsistema de autoaprendizaje.

Introducción

Para el cumplimiento exitoso del objetivo general se han definido como **Objetivos Específicos:**

- Elaborar el marco teórico de la investigación.
- Desarrollar la fase de diseño.
- Validar los artefactos obtenidos.

De este modo quedaron plasmadas las siguientes **tareas**:

- Realización de un estudio de los sistemas de autoaprendizaje, Laboratorios Virtuales, recursos didácticos, objetos de aprendizaje, estándar SCORM y mapa conceptual.
- Realización de un estudio de las metodologías de desarrollo de software, herramientas CASE, herramientas para modelar el mapa conceptual, lenguajes de modelado y patrones más utilizados.
- Obtención del mapa conceptual.
- Realización de la especificación de requisitos de software, matriz de trazabilidad de los requisitos, diagrama de clases, diagrama de estructura de navegación, diagrama de estructura de presentación y diagrama de secuencia.
- Validación de los artefactos obtenidos.

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.
- 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 Sistemas Operativos I en la Facultad 3 de la Universidad de las Ciencias Informáticas.

- Observación: Para la recopilación de información de las características y comportamiento de los estudiantes al utilizar las TIC en la enseñanza y el aprendizaje con el uso de los Objetos de Aprendizaje en la asignatura de Sistemas Operativos I.

Para presentar los resultados obtenidos durante la investigación, el trabajo consta de 3 capítulos estructurados de la siguiente forma:

Capítulo 1: Fundamentación Teórica: Se describe todo lo referente al estado del arte respecto al tema a investigar, haciendo énfasis en los laboratorios virtuales, sistemas de autoaprendizaje, recursos didácticos, objetos de aprendizaje y el estándar SCORM, también se abordan sobre las metodologías, lenguajes de modelado y herramientas a utilizar, así como los patrones, la ingeniería de requisitos.

Capítulo 2: Se describe el desarrollo de la solución propuesta a partir del diagrama de casos de uso, obtención de requisitos funcionales y no funcionales, los patrones utilizados, la descripción de los casos de uso y una serie de diagramas diseñados para el desarrollo del laboratorio virtual.

Capítulo 3: En este capítulo se validará la solución mediante las métricas de la calidad de especificación de requisitos y casos de uso del sistema, además de realizar la matriz de trazabilidad. También se valida el diseño a través de la métrica Tamaño Operacional de Clase y Relación entre Clases.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se realizará un estudio de los conceptos claves de la investigación como, los laboratorios virtuales y sus ventajas, recursos didácticos, los objetos de aprendizaje y el estándar SCORM. Se abordará sobre las metodologías, los lenguajes y herramientas de modelado para el diseño, la ingeniería de requisitos y los patrones.

1.2 Sistemas de Autoaprendizaje

La gran aceptación por parte de nuestra sociedad de las denominadas tecnologías de la información ha hecho que se haya extendido su uso a muchos campos entre los que se encuentra la educación. En el caso particular de la enseñanza universitaria, el desarrollo de programas de autoaprendizaje no pretende en ningún caso sustituir a las denominadas clases presénciales, sino disponer de un complemento didáctico que ayude a comprender y a profundizar en ciertos conceptos, su objetivo final es facilitar al alumno el aprendizaje individual de la materia, además de incentivar su capacidad para tomar decisiones analizando la información de la que dispone.

En los sistemas de autoaprendizaje se promueve el estudio independiente sin la dirección constante del profesor, para ello el papel de las tecnologías es fundamental. Estos sistemas no solo son utilizados en el ámbito educativo sino también en el ámbito empresarial.

1.3 Laboratorios virtuales

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 pa ra la colaboración y la experimentación a distancia con objeto de investigar o realizar otras actividades creativas, y elaborar y difundir resultados mediante tecnologías difundidas de información y comunicación” (P.Vary, 1999)

Julián Monge Nájera define un laboratorio virtual como “simulaciones de prácticas manipulativas que pueden ser hechas por el estudiante le.jos de la universidad y el docente”. (Monge-Nájera, 1999)

Capítulo 1: Fundamentación Teórica

Después de estudiar las definiciones dada por varios autores, podemos definir un Laboratorio Virtual como: un entorno de simulación y experimentación a distancia para realizar un conjunto de prácticas de laboratorio que sirven de apoyo a las distintas asignaturas.

1.4 Tipos de Laboratorios Virtuales

Existen diversos tipos de laboratorios virtuales, según la bibliografía consultada, se proponen tres clasificaciones que se mencionan a continuación.

1.4.1 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. Es el caso de programas con instalación propia, que pueden estar destinados a plataformas Unix, Linux, M.S. Windows e incluso necesitar que otros componentes de software estén instalados previamente, pero que no necesitan los recursos de un servidor determinado para funcionar(como bases de datos o módulos de software de servidor). También determinados laboratorios virtuales pensados inicialmente como aplicaciones Java accesibles a través de un servidor Web se pueden considerar de este tipo si funcionan localmente y no necesitan recursos de un servidor en concreto.

1.4.2 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.

1.4.3 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. Otro motivo que los hace dependientes de sus servidores es la habitual gestión de usuarios en el servidor. (Ramil, 2007)

Capítulo 1: Fundamentación Teórica

Se trabajará con los laboratorios Virtuales Web pues el grupo de trabajo del proyecto que está desarrollando el laboratorio virtual de la Asignatura de Sistemas Operativo en la Facultad 3 de la Universidad de Ciencias Informáticas tomó esta decisión en los inicios del proyecto, además de permitir que varios usuarios estén conectados a la aplicación y puedan hacer uso del mismo para su beneficio.

1.5 Ventajas de los laboratorios virtuales

El uso de laboratorios virtuales tiene sin duda muchos beneficios: permite al estudiante buscar información, en él se pueden relacionar fenómenos con sus consecuencias, se pueden repetir los eventos o fenómenos cuantas veces se requiera. Se incorporan las tecnologías de la información y comunicaciones en las prácticas educativas y sociales para beneficio de los estudiantes.

El aprendizaje está basado en simulaciones, pero además de esto tiene otras ventajas más significativas como son:

- 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.
- Es una herramienta de autoaprendizaje. (Pereira, 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 necesario que el estudiante realice una actividad ordenada y progresiva, beneficiosa para alcanzar objetivos básicos concretos, además el alumno no utiliza elementos reales en el laboratorio virtual, provocando una pérdida parcial hacia la visión de la realidad y los resultados son menos atractivos para estudiante.

1.6 Diseño de Aplicaciones

Debido a que el proceso de transformación de recursos educativos basados en las TIC es complejo, se requiere del apoyo de especialistas en educación y cómputo así como recursos didácticos que faciliten el proceso de transformación.

El uso de las herramientas informáticas a través de un cómputo permite desarrollar los procesos de diseño, los mismos tienen dos fases:

- Fase pedagógica: donde se diseña la unidad didáctica.

Para esta fase se modela un mapa conceptual asiendo uso de herramientas.

Los mapas conceptuales son un medio para visualizar ideas o conceptos y las relaciones jerárquicas entre los mismos. La elaboración de estos mapas puede servir para representar textos en los que se maneje lógica y ordenadamente cierta información, de ahí que sean considerables como organizadores de contenido de diversas actividades académicas y de la vida práctica. A continuación se muestran algunas de estas herramientas:

Inspiration: Es una de las herramientas de aprendizaje visual más utilizada por los docentes de todo el mundo. Especialmente diseñada para la creación de diagramas en forma de telaraña, mapas de ideas y mapas conceptuales. Permite exportar los mapas creados a formatos gráficos como jpg, gif y bmp. Compatible con los SO Windows y Macintosh.

CmapTools: Se diseñó con el objetivo de apoyar la construcción de modelos del conocimiento representados en forma de mapas conceptuales (MC), pero también se pueden elaborar telarañas, mapas de ideas y diagramas causa-efecto.

CompendiumLD: es una herramienta para ayudar a crear y representar visualmente diseños de aprendizaje. Es una adaptación de la herramienta Compendio, que proporciona un conjunto predeterminado de iconos para la creación de mapas, permitiendo una fácil comprensión y creación de estos.

Beneficios de CompendiumLD

- Aumento de la eficiencia
- Fomentar la creatividad

Se utilizó el CompendiumLD como herramienta de diseño para elaborar el mapa conceptual pues el equipo de trabajo del proyecto que está desarrollando el laboratorio virtual de la asignatura de Sistemas Operativos estableció utilizar la misma, además permite una fácil interacción a la hora de hacer uso de esta herramienta.

- Fase tecnológica: donde se estandariza la unidad como OA.

1.7 Recursos didácticos

Los recursos didácticos no son más que cualquier material que sea construido con el objetivo de ayudar al profesor en su tarea de enseñar y al alumno facilitar su aprendizaje. Estos se pueden considerar como una herramienta para llevar a cabo la tarea formativa siempre que sean utilizados de la manera adecuada.

Dentro de las funciones que desarrollan, podemos encontrar que brindan información a los estudiantes ayudando a enriquecer su conocimiento, pero también permiten a los alumnos ejercitar y desarrollar las habilidades. Debido a la fácil interacción que brinda, despiertan la motivación y el interés del contenido en cuestión. Además los profesores pueden evaluar a los estudiantes en cada momento, manteniendo un seguimiento del comportamiento de este en todo el transcurso del contenido impartido.

Estos recursos han permito mejorar la calidad del aprendizaje y es importante destacar el papel que han jugado los objetos de aprendizaje, pues han ayudado a enriquecer el proceso docente educativo.

1.8 Objetos de Aprendizaje

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 tecnologías de la información y la comunicación (TIC).

Por tal motivo, investigadores sociales como Castells (2000), 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 entidades de información denominadas objetos de aprendizaje (OA).

El concepto tuvo su aparición entre los años 1992 - 1996 donde Wayne Hodgins, el primero en utilizar la metáfora de LEGO, crea un grupo de investigación llamado Learning Architecture and Learning Objects (1994). Luego aparecen distintos grupos y consorcios que trabajan sobre esta cuestión como IMS, ARIADNE, etc.

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

Capítulo 1: Fundamentación Teórica

"Cualquier entidad, digital o no digital, que pueda ser utilizada, reutilizada o referenciada durante un proceso de aprendizaje mediado por la tecnología"

- Contenidos de la multimedia.
- Contenidos didácticos.
- Objetivos de aprendizaje.
- Software didáctico y herramientas de software.
- Personas, organizaciones o eventos.

Llama la atención en esta definición: primero que se toman en cuenta no solo los recursos digitales, segundo las entidades o recursos pueden o no estar soportadas en la tecnología y tercero deja explícito el soporte tecnológico para desarrollar el proceso de aprendizaje lo que apunta al problema cardinal de los LMS.

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

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.

David Wiley, editor de un libro muy conocido acerca de los objetos de aprendizajes propuso una definición más cerrada: "cualquier recurso digital que puede re-usarse para apoyar el aprendizaje."

- Recurso digital
- Puede ser reutilizada
- Tiene un propósito educativo

Un OA es una unidad mínima de información, digital o no digital, que puede ser re-usada y secuenciada junto con otros OA para conformar cursos o unidades que abarquen objetivos de aprendizaje más amplios.

Capítulo 1: Fundamentación Teórica

Mediante el uso de los OA, se pretende asegurar que la producción de los materiales educativos se desarrollen, se organicen y se distribuyan de manera uniforme, pero además que un mismo material educativo pueda utilizarse en contextos instruccionales diferentes.

La creación y disposición de recursos para el aprendizaje supone una visión de acumulación del capital académico que se da por ejercicio de la enseñanza y la participación colaborativa de los docentes. El acento está puesto en la máxima distribución de la información pedagógica producida, o como señala Chan (2002) el valor de los OA está dado por la cantidad de usuarios que consumen la información.

El profesor Emilio Castañeda los define en función de los Objetos de Información y señala como Objetos de Aprendizaje, "No es más que una colección intencionalmente estructurada de objetos de información (OI) utilizando metadatos, con la finalidad de darle un uso para el aprendizaje". Definiendo Objeto de Información (OI), "se define como la más pequeña pieza o unidad de información que puede ser usada y rehusada como entidad independiente y que posee un identificador o nombre único" ejemplos de OI pueden ser: un video, una imagen, un artículo, una definición, un procedimiento, un sonido, otros (ejemplo: anexo.doc, sonido1.wav, imagen.jpg, trabajo.ppt, u otros). (Castañeda, 2002)

Los Objetos de Aprendizaje constituyen un recurso pedagógico – tecnológico para favorecer el proceso de enseñanza – aprendizaje, es decir desde el punto de vista pedagógico, sirven para que los estudiantes aprendan, los profesores preparen tareas/actividades de aprendizaje y los estudiantes diseñen sus propias guías de aprendizaje.

Desde lo tecnológico, en el campo de la informática, están basados en el paradigma de cómputo orientado a objetos, el cual se refiere a crear componentes o módulos que puedan ser desde el punto de vista informático reutilizables en otros programas.

El concepto reutilizable tiene así un triple impacto, desde el proceso informático (programación), desde el diseño de las actividades de aprendizaje (maestro/profesor) y desde el aprendizaje (estudiante).

Objetivos de los OA

- Favorecer el acceso a los contenidos educativos.
- Optimizar los recursos destinados a su producción mediante su reutilización.
- Se aplican 3 estrategias:

- ✓ Modularidad y agregación de los contenidos.
- ✓ Descripción de los contenidos con metadatos normalizados.
- ✓ Interoperabilidad de los contenidos en entornos diferentes.

Promesas de los OA

- Acceso universal a contenidos de aprendizaje.
- Optimización de recursos.
- Reutilizables en distintos contextos.
- Localización de más y mejores contenidos.

Frenos de los OA

- Los derechos de autor, la rentabilidad de la producción de contenidos.
- La carga de trabajo de los metadatos.
- Habilidades técnicas de los docentes.
- Una visión del aprendizaje centrada en la transmisión.
- El conocimiento de cómo un objeto que puede usarse.

1.8.1 Contenido de un OA

Un OA pude incluir contenido multimedia, instrucciones, objetivos de aprendizaje, programa informático de instrucción, programa informático para herramientas, personas, organizaciones, eventos encomendados, y todo eso dentro de un aprendizaje soportado con tecnología.

La incorporación de los objetos de aprendizaje en una propuesta didáctica innovadora e integral requiere considerar aspectos:

- Que sean entidades independientes y completas, abordando un tema o subtema de estudio con independencia del contexto instruccional.
- Que tengan un tamaño estándar.
- Que una secuencia de OA tenga un contexto, creado durante el ensamblado.
- Que sigan un formato instruccional estándar (selección de estrategias instruccionales, uso de jerarquías).
- Que incluyan meta-etiquetas para facilitar su recuperación con vistas a su re-uso.

- Que describa las características pedagógicas de un recurso (tipo de recurso, papel del usuario, contexto de utilización).
- Que especifique las condiciones de utilización del recurso.
- Que describa, si hay razones, la relación entre un recurso y otros.
- Que permita comentarios sobre la utilización pedagógica del recurso.
- Que describa la localización del recurso en un sistema de clasificación.

1.9 SCORM

Cuando hablamos de objetos de aprendizajes, tenemos en mente, el cómo prevenir y posibilitar posibles incompatibilidades, migraciones, exportaciones e integraciones de los contenidos desarrollados a otros entornos físicos o virtuales o sea otras plataformas o proyectos.

Para ello es ideal referirnos al SCORM (Sharable Content Object Reference Model / Modelo Referenciado de Objetos de Contenido Compartible) que se basa en plataformas de tipo LMS (Learning Management System / Plataformas de Administración de la Enseñanza). Se concentra en cuatro palabras claves: Reusabilidad, Accesibilidad, Interoperabilidad, Durabilidad.

1.9.1 ¿Qué es SCORM?

Es un estándar de paquetes de objetos de aprendizaje, es decir, de pequeñas unidades de aprendizaje en soporte digital, como, por ejemplo, páginas web. Un paquete no es otra cosa que una serie de objetos de aprendizaje juntos.

Para que no se pierda la organización que le da el autor, cada paquete va acompañado de un manifiesto, es decir, de un documento donde queda reflejado el contenido y el orden o secuencia que se puede seguir para lograr los conocimientos. El contenido del manifiesto son datos que proporcionan información de los objetos de aprendizaje que contiene el paquete.

Lo que está estandarizado es el manifiesto, que no es otra cosa que un documento XML donde quedan reflejados los metadatos (información sobre la estructura en que se organizan los objetos de aprendizaje). El manifiesto es interpretado por unas hojas de estilo que transforman los metadatos escritos en lenguaje XML a lenguaje comprensible por los humanos.

El paquete SCORM no es nada más que un fichero comprimido en formato Zip, que contiene los objetos de aprendizaje, el manifiesto y las hojas de estilo que permiten interpretarlo.

Hay diferentes estándares sobre metadatos. SCORM es un estándar americano que tiene como característica la facilidad de ser interpretado por diferentes entornos virtuales de enseñanza/aprendizaje como por ejemplo Moodle.

1.10 Principales objetos de aprendizaje

- Los Simuladores: Son objetos de aprendizaje que mediante un programa de software intentan modelar parte de una réplica de los fenómenos de la realidad. Su propósito es que el usuario construya conocimiento a partir del trabajo exploratorio, la inferencia y el aprendizaje por descubrimiento. Los simuladores se desarrollan en un entorno interactivo, que permite al usuario modificar parámetros y ver cómo reacciona el sistema ante el cambio introducido.

- Animaciones: Son un conjunto de imágenes estáticas que se colocan en forma secuencial para dar la impresión de movimiento, al ser rodadas a gran velocidad. Por lo general se utilizan para hacer demostraciones o simulaciones.

- Documentos interactivos: Son documentos en los que la interacción de refiere a la consulta de hipertextos a un sistema de navegación que facilita el acceso a los contenidos.

- Cursos digitales: Son objetos de aprendizaje virtuales complejos y estructurados para la formación, en cualquier área del conocimiento, a través de la Internet o distribuidos en medios electrónicos como los CD-ROMs.

- Aplicaciones multimedia: Son materiales informáticos cuyo propósito es facilitar el autoaprendizaje por parte del usuario. Integran en un solo paquete varios medios (de allí su nombre), tales como textos, imágenes, videos, animaciones y sonidos. Son una poderosa herramienta didáctica, muy útil para la enseñanza de diversas disciplinas, especialmente los idiomas.

1.11 Metodología

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.

Capítulo 1: Fundamentación Teórica

Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente, este proceso es desarrollado detallando 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.

1.11.1 Metodología de Administración de Relaciones (RMM).

El método RMM fue la primera metodología para el diseño de multimedia y fue creada en 1995. Es un método para hipermedia que cubre todo el ciclo de desarrollo, desde el estudio de factibilidad hasta la evaluación del sistema, aunque sólo propone actividades y productos concretos para las fases de análisis y diseño. (León, 2009)

El modelo propone un lenguaje que permite describir los objetos del dominio, sus interrelaciones y los mecanismos de navegación hipermedia de la aplicación. Los objetos del dominio se definen con la ayuda de entidades, atributos y relaciones asociativas. Para moldear los aspectos unidos a la presentación de las entidades es necesario que exista un subconjunto de atributos de una misma entidad destinados a ser presentados de forma agrupada.

La navegación se moldea con la ayuda de primitivas de acceso, enlaces estructurales (unidireccional y bidireccional) que permiten especificar la navegación entre los atributos, y visita guiada condicional, índice condicional y agrupación, que permiten especificar la navegación entre entidades.

1.11.2 Proceso Unificado de Desarrollo de Software (RUP).

RUP es una metodología para la ingeniería de software. Actualmente, unida al lenguaje de modelado UML, es el estándar más usado para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. Además permite producir aplicaciones informáticas más fuertes y flexibles que se ajustan a las necesidades de los usuarios.

Esta metodología está compuesta por características esenciales que son:

Dirigido por los Casos de Uso: Un sistema de software se crea para servir a sus usuarios. Por lo tanto, para construir un sistema exitoso se debe conocer qué es lo que quieren y necesitan los usuarios prospectos. Un caso de uso es una pieza en la funcionalidad del sistema que le da al usuario un resultado

de valor. Los casos de uso capturan los requerimientos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso, el cual describe la funcionalidad completa del sistema. (Pérez, 2010)

Centrado en la arquitectura: El concepto de arquitectura de software involucra los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura es la vista del diseño completo con las características más importantes hechas más visibles y dejando los detalles de lado. El proceso ayuda al arquitecto a enfocarse en las metas correctas, tales como claridad, flexibilidad en los cambios futuros y la reusabilidad. (Pérez, 2010)

Iterativo e incremental: Las iteraciones se refieren a pasos en el flujo de trabajo, los incrementos se refieren a crecimiento en el producto. En cada iteración, los desarrolladores identifican y especifican los casos de uso relevantes, crean el diseño que usa la arquitectura como guía, implementan el diseño en componentes y verifican que los componentes satisfacen los casos de uso. (Pérez, 2010)

La metodología de RUP cuenta con cuatro fases fundamentales:

Fases: (Pérez, 2010)

- Concepción o Inicio: Comprender los requisitos y determinar visión y alcance del proyecto.
- Elaboración: Asignar recursos, especificar las características y definir la arquitectura.
- Construcción: Implementación, construir el producto operacional.
- Transición: Hacerlo operativo para los usuarios, nivel correcto de calidad para entregar.

1.11.3 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.

Es una de las metodologías de desarrollo de software más exitosa en la actualidad utilizada para proyectos de corta duración, poco equipo y poco tiempo de desarrollo. La metodología consiste en una programación rápida o extrema, cuya particularidad es tener como parte del equipo, al usuario final, pues es uno de los requisitos para llegar al éxito del proyecto. (León, 2009)

Características de XP, la metodología se basa en:

- Pruebas Unitarias: se basa en las pruebas realizadas a los principales procesos, de tal manera que adelantándonos en algo hacia el futuro, se pueda hacer pruebas de las fallas que pudieran ocurrir. Es como si se adelantara a obtener los posibles errores. (León, 2009)
- Refabricación: se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio. (León, 2009)
- Programación en pares: una particularidad de esta metodología es que propone 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. (León, 2009)

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 integrase al equipo de desarrollo, otro motivo que desfavorece el uso de esta metodología es la baja documentación que produce lo que dificultaría el proceso de mantenimiento, puesto que hay que prever que pasará luego de entregado el software, cuando el equipo se disuelva, y sea necesario realizar algún cambio o mejora. Es por esto que se hace necesario mantener cierta documentación.

1.11.4 Marco de trabajo

Un marco de trabajo está compuesto por actividades sombrillas y genéricas o comunes, que se ejecutan sobre la base de un conjunto de acciones que se apoyan en un grupo de tareas para poder desarrollar dichas actividades que definen al proceso de desarrollo del software.

Actividades genéricas o comunes: (Pressman, 2005)

- **Comunicación:** Esta actividad del marco de trabajo implica una intensa colaboración y comunicación con los clientes; además, abarca la investigación de requisitos y otras actividades relacionadas.
- **Planificación:** Esta actividad establece un plan para el trabajo de la ingeniería del software. Describe las tareas técnicas que deben realizarse, los riesgos probables, los recursos que serán requeridos, los productos del trabajo que han de producirse y un programa de trabajo.

- **Modelado:** Esta actividad abarca la creación de modelos que permiten al desarrollador y al cliente entender mejor los requisitos del software y el diseño que lograra satisfacerlos.
- **Construcción**: Esta actividad combina la generación del código (ya sea manual o automatizado) y la realización de pruebas necesarias para descubrir errores en el código.
- **Despliegue:** El software (como una entidad completa o un incremento completado de manera parcial) se entrega al cliente, quien evalúa el producto el producto recibido y proporciona información basada en su evaluación.

Las actividades sombrillas garantizan un mejor control, seguimiento y ejecución del proceso de desarrollo del software, estas asegura una alta calidad y correcta realización del mismo.

Actividades Sombrillas (Pressman, 2005)

- **Seguimiento y control**: Permite que el equipo de software evalúe el progreso comparándolo con el plan de proyecto y así tomar las acciones necesarias para mantener el programa.
- **Gestión del riesgo:** Evalúa los riesgos que pudieran afectar los resultados del proyecto o la calidad del producto.
- **Aseguramiento de la calidad del producto:** Define y conduce las actividades requeridas para asegurar la calidad del software.
- **Revisiones técnicas formales**: evalúa los productos del trabajo de la ingeniería del software en un esfuerzo encaminado a descubrir y eliminar los errores antes de que estos se propaguen hacia la siguiente acción o actividad
- **Medición:** define y recolecta mediciones del proceso, el proyecto y el producto para ayudar al equipo a entregar software que satisfaga las necesidades del cliente; se puede usar en conjunto con todas las otras actividades del marco de trabajo o actividades sombrillas
- **Gestión de la configuración del software:** maneja los efectos del cambio a través del proceso del software.
- **Gestión de la reutilización:** define los criterios para la reutilización de productos del trabajo (se incluyen componentes del software) y establece mecanismos para la creación de componentes reutilizables.

- **Preparación y producción del producto de trabajo:** abarca las actividades requeridas para crear productos del trabajo como modelos, documentos, registros, formatos y listas.

Después de haber realizado un estudio de las metodologías para el desarrollo del software, no se tomó una en específico, debido a que ninguna se ajusta a las características de la aplicación que se está desarrollando. XP no cuenta con buena documentación, afectando el mantenimiento de la aplicación. RUP es una metodología con un proceso de desarrollo grande y no es favorable para proyectos que no cuentan con tiempo disponible y RMM no permite trabajar con un lenguaje de modelado si no es el que tiene definido, lo que no favorece al diseño de la aplicación que se va a desarrollar. Por estas razones, se realizó un marco de trabajo como guía del proceso de desarrollo de la aplicación, para esto, están presentes las actividades antes mencionadas, que son necesarias para seguir una secuencia de pasos y tareas a cumplir que permitirán realizar una correcta elaboración de la aplicación. En este trabajo solo se realizaran las actividades de comunicación, planificación y modelado.

Actividades a desarrollar en cada actividad genérica.

Comunicación

- Entrevistas con el cliente.
- Extracción de requisitos.
- Especificación de requisitos.

Planificación

- Plan de trabajo que contiene las tareas que fueron realizadas.

Modelado

- Diagrama de Casos de Uso.
- Diagramas de Actividades.
- Descripciones Textuales de Casos de Uso.
- Diagramas de Clase.
- Diagramas de Navegación.
- Diagramas de Presentación.
- Diagramas de Secuencia.

1.12 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.

1.12.1 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.

UML propone diagramas con la finalidad de presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Un modelo UML describe lo que supuestamente hará un sistema, pero no dice cómo implementarlo. Luego de analizadas estas notaciones de modelado se decide utilizar dos de estos lenguajes, UML que brinda la posibilidad de construir los modelos que define la metodología escogida para desarrollar el software, lo que permite expresar requisitos y representar todos sus detalles, además hace que se obtenga una documentación que es válida durante todo el ciclo de vida de un proyecto. Al mismo tiempo es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad para modelar los artefactos creados durante el proceso de desarrollo de software.

1.12.2 ApEM – L

ApEM – L (Lenguaje de Modelado de Aplicaciones Educativas) está basado en el lenguaje de modelación UML, tomando elementos representativos de extensiones del mismo (Ricardo, 2007), descansa toda su estructura sobre los elementos planteados por el estándar OCL, en su versión actualizada 2.0 del 2003. Consta de tres áreas fundamentales: de estructura lógica, de comportamiento dinámico y de gestión del modelo, y a su vez de cuatro vistas distribuidas en dichas áreas: estática, de arquitectura, de

comportamiento y de presentación; para la modelación de los productos, a través de un conjunto de diagramas distribuidos por cada una de estas vistas. (Ricardo, y otros)

Ventajas: (Ricardo, 2007)

- Puede utilizar para su representación todas las herramientas CASE que existen actualmente para la modelación de UML.
- Es un lenguaje que utiliza el estándar internacional OCL, para la modelación de la programación Orientada a Objetos.
- No modifica la semántica del lenguaje base UML, sino que trabaja en estereotipos restrictivos, por lo que a su vez produce modificaciones descriptivas y decorativas en la representación de los componentes del lenguaje base.

1.12.3 Lenguaje para la modelación Orientada a Objetos de Aplicaciones Multimedia (OMMMA – L).

Fue desarrollado en la Universidad de Paderborn en Alemania, en el año 2001, tomando como base el lenguaje UML. Consta de cuatro vistas fundamentales en su modelación: vista lógica, vista de presentación espacial, vista de comportamiento temporal predefinido y vista de control interactivo. Modifica los diagramas originales de UML de: clases, secuencia y estado. Añade como parte de la vista de presentación espacial un nuevo diagrama: el diagrama de presentación, para la representación espacial de los elementos visuales del futuro software multimedia. Basa su descripción en el patrón de arquitectura MVCMM.

Partiendo del estudio realizado de los lenguajes de modelado, se utilizará ApEM-L, ya que está compuesto por estereotipos restrictivos y descriptivos que permiten modelar de manera más explícita el diseño de las aplicaciones multimedia. Este lenguaje presenta diagramas que visualizan aspectos como videos, imágenes, animaciones, sonidos, entre otros que son importantes para tener una vista más amplia del diseño. Además fue creado en la Universidad de las Ciencias Informáticas para el diseño de este tipo de aplicaciones, lo que brinda la facilidad de poder trabajar directamente con los especialistas funcionales que pueden brindar ayuda al respecto.

1.13 Herramientas CASE

Las herramientas CASE son una base para el proceso de análisis y desarrollo de software. Estas herramientas son el conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores. Facilitan el mejoramiento del ciclo de vida del desarrollo de sistemas informáticos, completamente o en alguna de sus fases.

Existen varias herramientas CASE que se pueden utilizar para el modelado, entre las que podemos mencionar:

1.13.1 Enterprise Architect

Es una herramienta de modelado UML flexible, completa y potente de modelado en UML orientado a objetos para el desarrollo completo del ciclo de vida del software bajo plataforma Windows. Es una alternativa completa y eficaz para realizar todos los diagramas correspondientes al flujo de trabajo de análisis y diseño. Ofrece un completo panel de herramientas básicas tales como diagramas de CU, diagramas de secuencia y diagramas de clases. Contiene una interfaz gráfica muy amigable, fácil de usar y proporciona rapidez una vez que el usuario logra interactuar con sus opciones y elementos.

(Valladares, y otros, 2008-2009)

1.13.2 Rational Rose

Es una de las más poderosas herramientas de modelado visual basado en UML para el análisis y diseño de sistemas orientados a objetos. Es una herramienta con plataforma independiente que ayuda a la comunicación entre los miembros del equipo, a monitorear el tiempo de desarrollo y a entender el entorno de los sistemas. Otras de las ventajas es que los diseñadores pueden modelar sus componentes e interfaces de forma individual y luego unirlos con otros componentes del proyecto. Ayuda a los desarrolladores de software a construir mejores productos en menor tiempo, da un excelente soporte en el manejo de cambios durante el ciclo de vida del proyecto y mejora la comunicación entre los miembros del equipo. (Valladares, 2008-2009)

1.13.3 Visual Paradigm

Visual Paradigm es un producto distinguido que facilita la organización de los diagramas. La herramienta ayuda al equipo de desarrollo de software a agilizar el modelado del software, aumentando al máximo y acelerando el trabajo en equipo y las contribuciones individuales. Es una herramienta multiplataforma. Además posee una buena cantidad de productos o módulos para facilitar el trabajo durante la confección

de un software así como garantizar la calidad del producto final. Permite dibujar todos los tipos de diagramas, código inverso, generar código desde diagramas y generar documentación. (Valladares, 2008-2009)

Visual Paradigm ofrece:

- Entorno de creación de diagramas para UML 2.0.
- Diseño centrado en casos de uso y enfocado al negocio que generan 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 (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. (González, 2010)

Se utilizará como herramienta de modelado el Visual Paradigm 4.1 pues la Universidad de Ciencias Informáticas posee la licencia para su uso. Además permite dibujar todo tipo de diagrama de clases, incluyendo aquellos que son necesarios para modelar aplicaciones multimedia.

1.14 Ingeniería de requisitos

"La Ingeniería de Requerimientos es el uso sistemático de procedimientos, técnicas, lenguajes y herramientas para obtener con un coste reducido el análisis, documentación, 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 ingeniería de requisitos (IR) juega un papel fundamental en la creación del software en la actualidad, pues la misma engloba todo el proceso de desarrollo del software. Además realiza una serie de pasos bien organizado, para realizar con buena calidad el software, dado los requisitos obtenidos por el cliente.

1.14.1 ¿Qué son los requisitos?

Los requisitos dentro del desarrollo del software son esenciales pues estos indican cuales son las funcionalidades que tendrá el sistema, tanto funcional como no funcional. Son una condición o capacidad que el sistema debe tener para satisfacer las necesidades del cliente.

La IEEE Standard Glossary of Software Engineering Terminology defines 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.

1.14.2 Extracción de requisitos

En esta etapa el analista se encarga de preguntarle al cliente, a los usuarios y a los que estén involucrados en los objetivos del sistema o producto como estos se ajustan a las necesidades del negocio o sea el problema que tiene que resolver. Los diversos servicios que el sistema debe prestar y las restricciones. (Pressman, 2005)

Para la construcción del producto es de mucha importancia esta actividad pues es aquí donde se descubren las funcionalidades del sistema para realizar un correcto modelado, es por esto que debe existir mutuo acuerdo entre las partes participantes y una buena comprensión de parte del analista logrando con esto, que la aplicación tenga una buena aceptación por parte del cliente.

1.14.3 Técnica para la captura de requisitos

Entrevistas: resulta una técnica muy aceptada dentro de la Ingeniería de Requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y entender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Actualmente existen muchos tipos de entrevistas y su estructura abarca tres pasos fundamentales: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista). Sin embarco esta no es muy fácil de aplicar, requiere que el entrevistador sea experimentado y tenga capacidad para elegir correctamente a los entrevistados obteniendo de ellos toda la información posible en un período de tiempo corto.

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. (Escalona, 2004)

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 y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo 10 personas). Una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador. (Escalona, 2004)

Mapas Conceptuales: los mapas conceptuales son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la IR pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de este. (Escalona, 2004)

Sketches y Storyboards (interfaces y estructura de navegación): está 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). (Escalona, 2004)

Casos de Uso: aunque inicialmente se desarrollaron como técnica para la definición de requisitos, algunos autores proponen casos de uso como técnica para la captura de requisitos Los casos de uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales) expresados como casos de uso de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos (personas y otros sistemas) que interactúan con el sistema. (Escalona, 2004)

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. (Escalona, 2004)

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

en forma complementaria a otras para obtener consenso respecto a la terminología a ser usada en el proyecto de desarrollo. (Escalona, 2004)

1.14.4 Especificación de Requisitos

En esta fase los requisitos son documentados donde se aplican técnicas y se especifica el análisis realizado anteriormente. El objetivo de este servicio es la formalización, definición, análisis y verificación de los requisitos del sistema de una aplicación, a partir de los requisitos presentados por los clientes.

1.14.5 Técnicas para validar requisitos

Los requisitos son validados para permitir que el sistema o la aplicación informática cumplan con todas las restricciones y características requeridas por los clientes.

Reviews (Revisión): esta técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida. (González & Hernández Pérez, 2010)

Auditorías: la revisión de la documentación con esta técnica consiste en el chequeo de los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir, sólo una muestra es revisada. (González & Hernández Pérez, 2010)

Matrices de trazabilidad: esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo los objetivos que cubre cada requisito, de esta forma, se podrán detectar inconsistencias u objetivos no cubiertos. (González & Hernández Pérez, 2010)

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 de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final. (González & Hernández Pérez, 2010)

1.14.6 Gestión de Requisitos

La gestión de requisitos es la forma de darle tratamiento a la actualización y cambio de los requisitos. Esta etapa comienza una vez terminado el período de desarrollo de requisitos, pues es aquí donde se

determina si estos cumplen con las necesidades del cliente y pueden pasar al diseño. Para esto se determinan un conjunto de actividades que 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. Esta proporciona elementos que ayudan a la comunicación entre los equipos de trabajo que es un factor que presenta gran problema en dicho proceso.

1.15 Patrones

Un patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, estandariza buenos principios y da sugerencias sobre la manera de usarlo en nuevas situaciones, sobre todo en la asignación de responsabilidades. Es decir, es una solución común a un problema común de un determinado contexto. (Jacobson, 2000)

1.15.1 Patrones de Casos de Uso

Los patrones de casos de uso forman reglas, permitiendo que los diagramas sean fáciles de comprender y posean un correcto modelado. Algunos de estos patrones son:

- **Concordancia (Adición)**

En el caso de este patrón alternativo, la subsecuencia común de casos de uso, extiende los casos de uso compartiendo la subsecuencia de acciones. Los otros casos de uso modelan el flujo que será expandido con la subsecuencia. Este patrón es preferible usarlo cuando otros casos de uso se encuentran propiamente completos, o sea, que no requieren de una subsecuencia común de acciones para modelar los usos completos del sistema. (Morales, 2010)

- **Extensión concreta o inclusión**

Extensión: Consiste en la existencia de una relación de extensión entre dos casos de uso. El caso de uso extendido puede ser o no instanciado por el caso de uso base. El caso de uso base puede ser concreto o abstracto. Este patrón se utiliza cuando un flujo puede extender el flujo de otro caso de uso o bien puede ejecutarse dentro de este. (Morales, 2010)

Inclusión: Existe una relación de inclusión del caso de uso base con el caso de uso incluido. Este último puede ser instanciado como él mismo. El caso de uso base puede ser concreto o abstracto. Este patrón se utiliza cuando un flujo puede ser incluido en el flujo de un caso de uso y también puede ejecutarse dentro de este. (Morales, 2010)

- **Múltiples actores**

Roles diferentes: Captura la concordancia entre actores manteniendo roles separados. Consta de un caso de uso y por lo menos dos actores. Es utilizado cuando dos actores juegan diferentes roles en un caso de uso, o sea, interactúan de forma diferente con el caso de uso. (Morales, 2010)

Roles comunes: Puede suceder que los dos actores jueguen el mismo rol sobre el CU. Este rol es representado por otro actor, heredado por los actores que comparten este rol. Es aplicable cuando, desde el punto de vista del caso de uso, solo exista una entidad externa interactuando con cada una de las instancias del caso de uso. (Morales, 2010)

1.15.2 Patrones de Diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas en el desarrollo del software. Estos presentan un conjunto de patrones para la asignación de responsabilidades donde se destacan los (GRASP o General Responsibility Assignment Software Patterns, Patrones Generales de Software para Asignación de Responsabilidades) y otros como los GOF (conocidos como los patrones de la banda de los cuatro GOF, gang of four), todos ellos constituyen principios básicos para la construcción eficaz de un software orientado a objeto.

Patrones GRASP

Experto: Asignar una responsabilidad a la clases experta en información o sea la que contiene la información necesaria para realizar la labor que tiene recomendada.

Creador: Asignar a la clase B la responsabilidad de crear una instancia de clase A si se cumple uno o más de los casos siguientes:

- B agrega objetos de A
- B contiene objetos de A
- B registra instancias de objetos de A
- B utiliza más estrechamente objetos de A.
- B tiene datos de inicialización que se pasarán a un objeto de A cuando sea creado (por tanto, B es un Experto con respecto a la creación de A).
- B es un creador de los objetos A.

Bajo Acoplamiento: Asignar una responsabilidad de manera que el acoplamiento se mantenga bajo entre las clases.

Alta cohesión: Asignar una responsabilidad de manera que la cohesión se mantenga alta en las clases.

Controlador: Asignar una responsabilidad de recibir o manejar un mensaje de evento del sistema a una clase que representa una de las opciones siguientes:

- Representa el sistema global, dispositivo o subsistema.
- Representa un caso de uso en el que tiene lugar el evento del sistema a menudo denominado <nombre del caso de uso> Manejador, <nombre del caso de uso> coordinador, <nombre del caso de uso> Sesión.
- Utilice la misma clase controlador para todos los eventos del sistema en el mismo escenario de caso de uso.
- Informalmente, una sesión es una instancia de una conversación con un actor. Las sesiones pueden tener cualquier duración, pero se organizan a menudo en función de casos de uso.

Patrones GOF Creacionales:

Creación: Dentro del grupo de los patrones de creación hay existen algunos que abordan problemas comúnmente encontrados al momento de decidir, dónde, cómo y cuándo crear objetos. Abstract Factory: El primero de la lista, usado para crear un conjunto de objetos que sean independientes del cliente.

Factory Method (Fábrica Abstracta): Aquí ya no se trabaja con un creador separado de la representación, son las clases derivadas las que deciden la implementación particular. A diferencia del anterior, ya no se tiene el direccionador, se trabaja con el creador directamente.

Prototype (Prototipo): Usado cuando es necesario trabajar con varias instancias de un mismo objeto, pero se busca además cierta independencia entre estos, Se trabaja con clonaciones de un objeto en particular.

Singleton (Instancia única): Cuando se quiere trabajar con solo una instancia de un objeto, este es por cierto el patrón que casi siempre el primero en ser mencionado. Un lugar donde podría ser aplicado es cuando se tienen opciones de menú, y se quiere que al hacer clic sobre el mismo, se muestre solo un formulario asociado.

Patrones GOF Estructurales (Composición):

Adapter (Adaptador): Adapta una interfaz para que pueda ser utilizada por una clase que de ninguna otra manera podría utilizarla.

Composite (Objeto compuesto): Permite tratar objetos compuestos como si de uno simple se tratara.

Decorator (Envoltorio): Añade funcionalidad a una clase dinámicamente.

Facade (Fachada): Proporciona una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.

1.15.3 Patrón Arquitectónico

Los patrones arquitectónicos son los que definen la estructura de un sistema de software, estos componen subsistemas con sus responsabilidades, también tienen una serie de directivas para organizar los componentes del sistema, con el objetivo de facilitar el diseño.

El patrón Modelo – Vista – Controlador MM (MVC-MM)

MVC divide una aplicación interactiva en tres áreas: procesamiento, salida y entrada; trabajando con tres tipos fundamentales de clases: clases vista, clases controladoras y clases modelo. En la variante para

modelar aplicaciones multimedia (MVC-MM) se diversifica la clase modelo incorporando al esquema dos nuevos tipos de clases: la modelo dinámica y la modelo estática, la cual a su vez se subdivide en las clases media y lógica de la aplicación. (Ricardo, 2007)

Principales clases del patrón (MVC-MM): (Ricardo, 2007)

- Clase Vista: Recibe las peticiones del usuario al sistema. Muestra la información de salida al usuario. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- Clase Modelo: Encapsula los datos y las funcionalidades. Gestiona el procesamiento y almacenamiento de la información. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- Clase Controladora: Reciben las entradas para la gestión de las mismas, a partir de las clases vistas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("service requests") para el modelo o la vista. Gestionan el modelo que realizará el procesamiento así como la vista que generará la respuesta al usuario.

El patrón Modelo – Vista – Controlador – Entidad (MVC-E)

Debido a las características del software educativo producido en Cuba, se realizó el análisis de una variante de solución al MVC-MM para las aplicaciones educativas cubanas, donde se descarguen las responsabilidades de la clase modelo concernientes al procesamiento y almacenamiento de la información persistente de las aplicaciones, donde se le agrega una nueva clase al modelo denominado Modelo Entidad, con dos tipos fundamentales de clase:

Modelo-Entidad-Media: es la responsable de agrupar las clases que identifican las medias y su árbol de jerarquía en la aplicación.

Modelo-Entidad-Persistente: tiene como responsabilidad la gestión de la información persistente, que antes sobrecargaba a la clase Modelo del patrón MVC original. Esta variación a su vez sustenta las características actuales de los sistemas multimedia como son: comunicación con bases de datos, archivos XML, o sistemas externos.

1.16 Conclusiones

En este capítulo se abordaron conceptos fundamentales a tener en cuenta para el desarrollo de este trabajo. Se profundiza sobre la ingeniería de requisitos, las técnicas para la captura y validación de los mismos. Otro de los temas estudiados son las distintas metodologías y patrones para el modelado de la aplicación.

Además se analizó cuales serán las tecnologías y herramientas informáticas a utilizar para el desarrollo de dicha aplicación informática, llegando a la conclusión de que el modelado de la aplicación será desarrollado utilizando como lenguaje de modelado ApEM–L, como herramienta CASE el Visual Paradigm y para el diseño del mapa conceptual el Compendium LD.

Capítulo 2: Descripción de la solución

2.1 Introducción

En este capítulo se mostrará el mapa conceptual y se realizará la captura de requisitos. Además se obtendrá el diagrama de caso de uso para representar la relación existente entre los casos de usos y sus actores, junto a la descripción de los mismos. Para realizar el diseño se expondrán una serie de diagramas donde se visualizará como son las relaciones entre las clases y cuáles será el diseño que tendrán los distintos casos de uso con los diferentes aspectos ya planteado en la descripción de los mismos.

2.2 Mapa Conceptual

En el siguiente mapa conceptual muestra como está compuesto el contenido referido al tema de Entrada y Salida en la asignatura de Sistemas Operativos. El mismo está compuesto por tres temas fundamentales Principio del Hardware, Principios del Software y Planificación de Disco.

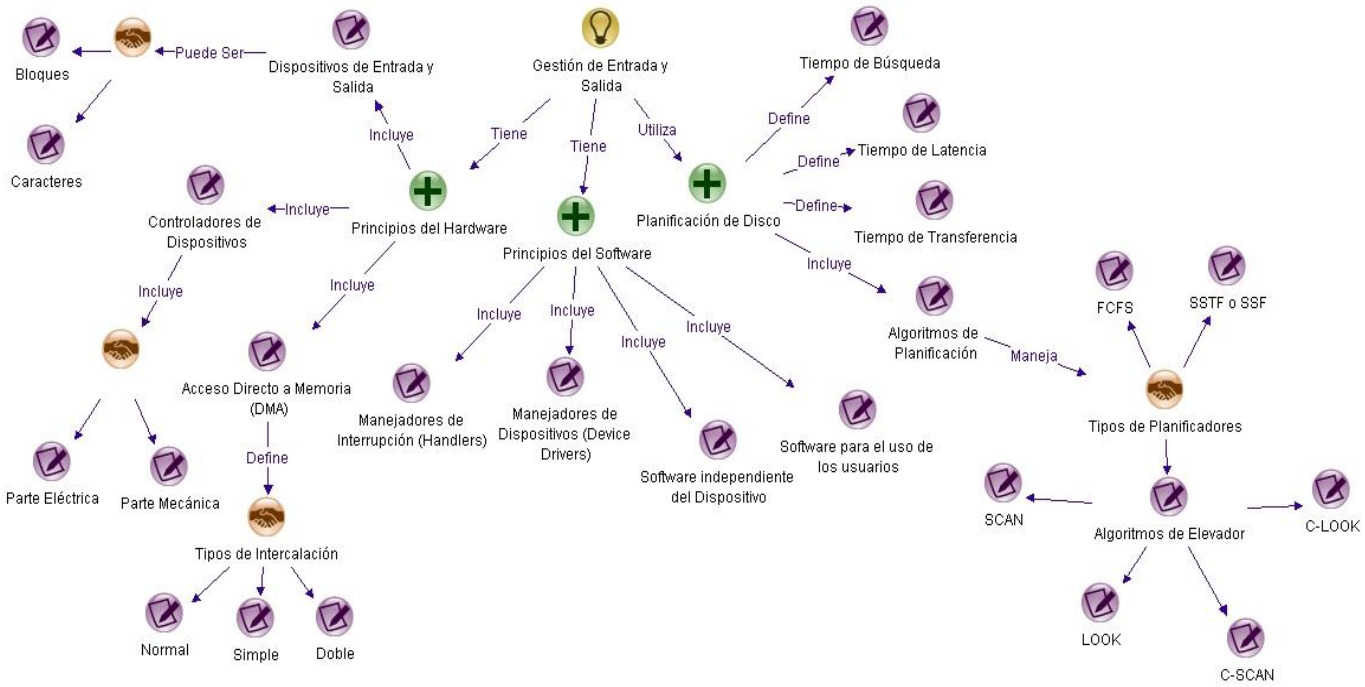


Figura 1: Mapa Conceptual

Capítulo 2: Descripción de la Solución

2.3 Técnicas aplicadas para la captura y especificación de requisitos.

Para realizar la captura de requisitos se aplicaron técnicas como la entrevista abierta ya que no se formularon preguntas y la tormenta de ideas, donde se realizó un debate en grupo para analizar una serie de ideas que dieron una visión más amplia de cuáles serían las funcionalidades y restricciones que tendría aplicación.

Para la especificación de requisitos escogió la técnica de Lenguaje Natural para un mejor entendimiento por las personas, sin la necesidad de tener amplios conocimientos técnicos.

2.4 Requisitos funcionales

Los requisitos funcionales especifican lo que el sistema debe hacer para satisfacer las necesidades del cliente. A continuación se muestran los requisitos funcionales obtenidos mediantes las técnicas utilizadas:

Requisitos Funcionales Críticos:

Los requisitos críticos son los que se refieren a las funcionalidades básicas del sistema.

RF 1: El sistema debe permitir modificar el contenido.

RF 2: El sistema debe permitir deshabilitar el contenido.

RF 3: El sistema debe permitir habilitar el contenido.

RF 4: El sistema debe permitir estudiar contenidos.

RF 4.1: El sistema debe permitir reanudar lección.

RF 4.2: El sistema debe permitir seleccionar el contenido a estudiar.

RF4.3: El sistema debe permitir guardar las trazas de navegación.

RF 4.4: El sistema debe permitir culminar su navegación.

RF 5: El sistema debe permitir consultar su evaluación.

RF 6: El sistema debe permitir rechazar ejercicio.

RF 7: El sistema debe permitir solicitar ayuda.

Requisitos Funcionales importantes:

Los requisitos funcionales importantes son los que brindan soporte a las funcionalidades básicas del sistema.

RF 8: El sistema debe permitir revalorizar evaluación.

RF 9: El sistema debe permitir Abandonar tema.

Requisitos Funcionales útiles:

RF 10: El sistema debe permitir visualizar el contenido que compone el tema de Entrada y Salida.

Los requisitos útiles son las características que le ofrecen comodidad al sistema (tecnología multimedia).

2.5 Requisitos no funcionales

Los requisitos no funcionales especifican cualidades o propiedades que el producto debe tener, o sea características que debe poseer el producto que lo haga fiable, rápido o confiable. Para este trabajo se han obtenido los siguientes 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: Debe poder ejecutarse en varios Sistemas Operativos.

Seguridad:
RNF7: Se 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. De esta forma se asegura la integridad y confidencialidad de la aplicación.

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.

Capítulo 2: Descripción de la Solución

2.6 Diagrama de Caso de Uso

Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y otros sistemas. (Estrada, 2009)

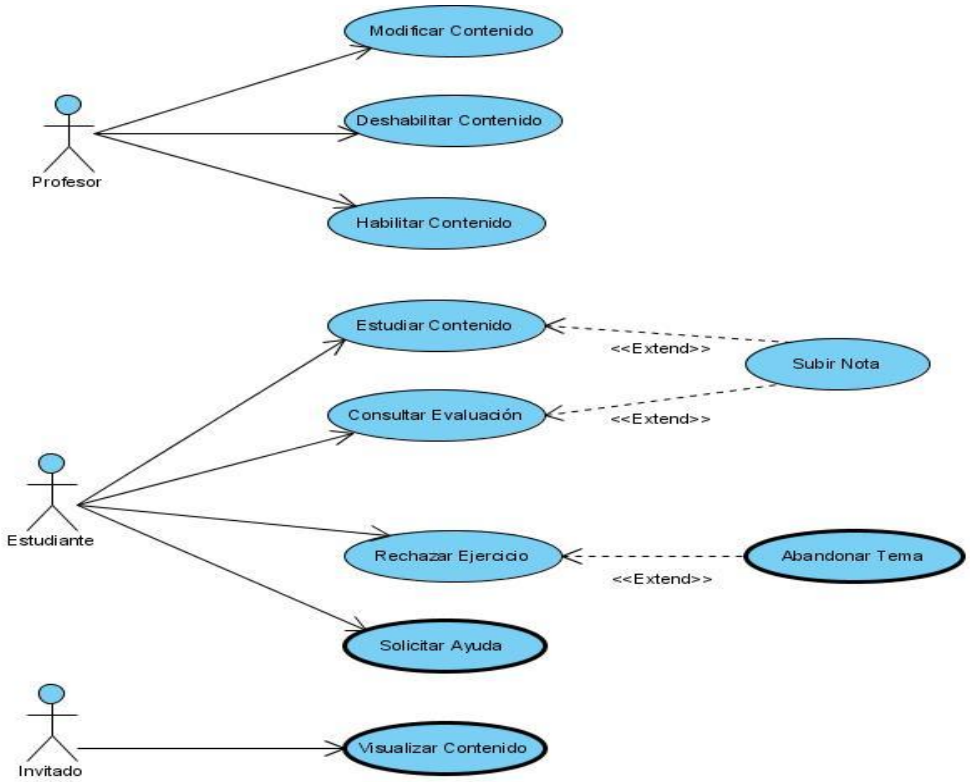


Figura 2: Diagrama de Caso de Uso

Descripción del diagrama.

Este diagrama de caso de uso está compuesto por el actor Profesor, que puede Modificar, Deshabilitar y Habilitar un contenido. Otro de los actores es el Estudiante, el mismo podrá Estudiar un Contenido, este si desea puede Subir Nota en caso de que haya aprobado el tema y tenga una nota baja. También puede, Consultar una Evaluación que le permitirá Subir Nota. Otro de los casos de uso asociados a este actor, es el de Rechazar Ejercicio, que a su vez puede Abandonar el Tema donde se encuentra y por último puede Solicitar Ayuda a algún profesor o estudiante. El tercer actor es el invitado que solamente puede visualizar el contenido.

Capítulo 2: Descripción de la Solución

2.7 Patrones aplicados en el diagrama de Caso de Uso

Patrones de Caso de Uso

Concordancia (Adición): este patrón fue utilizado en el caso de uso Subir Nota ya que este comparte una subsecuencia común de acciones con los casos de uso Estudiar Contenido y Consultar Evaluación.

Extensión Concreta: se empleó este patrón ya que existe una relación de extensión entre los casos de uso Rechazar Ejercicio y Abandonar Tema.

2.8 Descripción del Caso de Uso "Estudiar Contenido"

Descripción Textual del Caso de Uso "*Estudiar Contenido"*	
Actores del caso de uso	Estudiante (Inicia)
Propósito	El caso de uso tiene como propósito permitir al estudiante observar el contenido a estudiar que se encuentra reflejado en la aplicación.
Resumen	El caso de uso se inicia cuando el Estudiante selecciona la opción "Estudiar Contenido", una vez seleccionada la opción deseada el Estudiante debe seleccionar la opción "Guardar Estado", luego al finalizar el contenido el estudiante debe escoger la opción "Culminar" y si desea, seleccionar la opción "Revalorizar Evaluación" en el transcurso del contenido y culmina así el caso de uso.
Casos de uso asociados	• Revalorizar Evaluación
Referencias	RF 5
Precondiciones	• El estudiante tiene que haber sido matriculado en el curso con anterioridad.
Poscondiciones	• El estudiante puede seleccionar la opción "Guardar Estado". • El estudiante puede seleccionar la opción "Culminar". • El estudiante puede seleccionar la opción "Revalorizar

<table>
<tr><td></td><td colspan="3">Evaluación”.</td></tr>
<tr><td colspan="4">Curso Normal de los Eventos</td></tr>
<tr><td colspan="2">Acciones del Actor</td><td colspan="2">Respuesta del Sistema</td></tr>
<tr><td colspan="2">1. El estudiante selecciona la opción “Estudiar Contenido”.
2. El estudiante oprime botón “Reanudar Lección”.
3. El estudiante oprime el botón “Guardar Estado”
4. El estudiante oprime el botón “Culminar”.</td><td colspan="2">1.1 El sistema verifica trazas de navegación.
1.2 Si se guardó estado, el sistema habilita botón “Reanudar Lección”.
2.1 El sistema muestra el contenido</td></tr>
<tr><td colspan="4">Cursos Alternos de los Eventos</td></tr>
<tr><td>Acción</td><td colspan="3">Curso Alterno</td></tr>
<tr><td>1.2</td><td colspan="3">Si el estudiante no oprime el botón, el sistema chequea las reglas de precedencia y si se cumple las reglas el sistema muestra el contenido, sino se cumplen las reglas el sistema muestra un mensaje “Error de Precedencia”, luego muestra el Menú Principal y concluye el caso de uso</td></tr>
<tr><td>2.1</td><td colspan="3">Si el estudiante presenta una puntuación de 2 o 5 puntos el sistema no activa la opción “Revalorizar Evaluación”.</td></tr>
<tr><td>2.2</td><td colspan="3">Si la nota del estudiante es 3 o 4 el sistema habilita la opción “Revalorizar Evaluación”.</td></tr>
<tr><td>Prioridad</td><td colspan="3">Crítico</td></tr>
<tr><td>Mejoras</td><td colspan="3"></td></tr>
<tr><td rowspan="3">Medias a utilizar</td><td>Tipo de Media</td><td>Descripción</td><td>Estado</td></tr>
<tr><td rowspan="2">Imagen</td><td>Movimiento del cabezal con el algoritmo SSF.</td><td rowspan="2">Existente</td></tr>
<tr><td>Movimiento del cabezal con el</td></tr>
</table>

		algoritmo SCAN.	Existente
		Capas del sistema de entrada/salida y algunas de sus funciones.	Existente
		Intercalación de los sectores.	Existente
		Iconos representativos de las opciones estándares en la aplicación.	En construcción
	Video o Animación		
	Sonido	Sonido leve cuando se oprima un botón.	En localización
		Sonido pequeño para cuando se pase por encima de las opciones del menú.	En localización
		Sonido para cuando se seleccione opciones del menú.	En localización
	Texto	Texto del contenido del tema a estudiar.	Existente
		Textos de las opciones.	
Elementos pedagógicos			

Tabla 1: Descripción del Caso de Uso "Estudiar Contenido"

Para ver las descripciones de los restantes casos de uso **(Consultar Anexo #1).**

2.9 Diagrama de Actividades

Los diagramas de actividades representan y visualizan el flujo de actividades entre el usuario y el sistema, permitiendo un mejor entendimiento para el equipo de desarrollo.

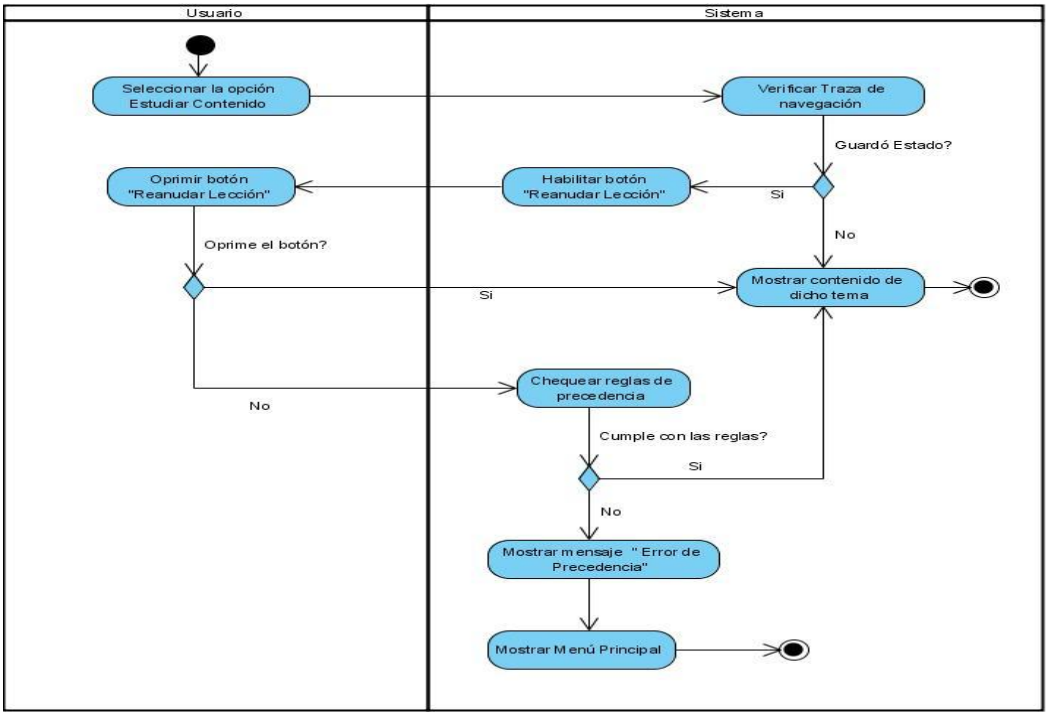


Figura 3: Diagrama de Actividades del caso de uso "Estudiar Contenido"

Para ver los Diagramas de Actividades de los restantes Casos de Uso **(Consultar Anexo #2 de documento de artefactos del módulo de Entrada y Salida).**

Capítulo 2: Descripción de la Solución

2.10 Diagrama de Clase del Diseño

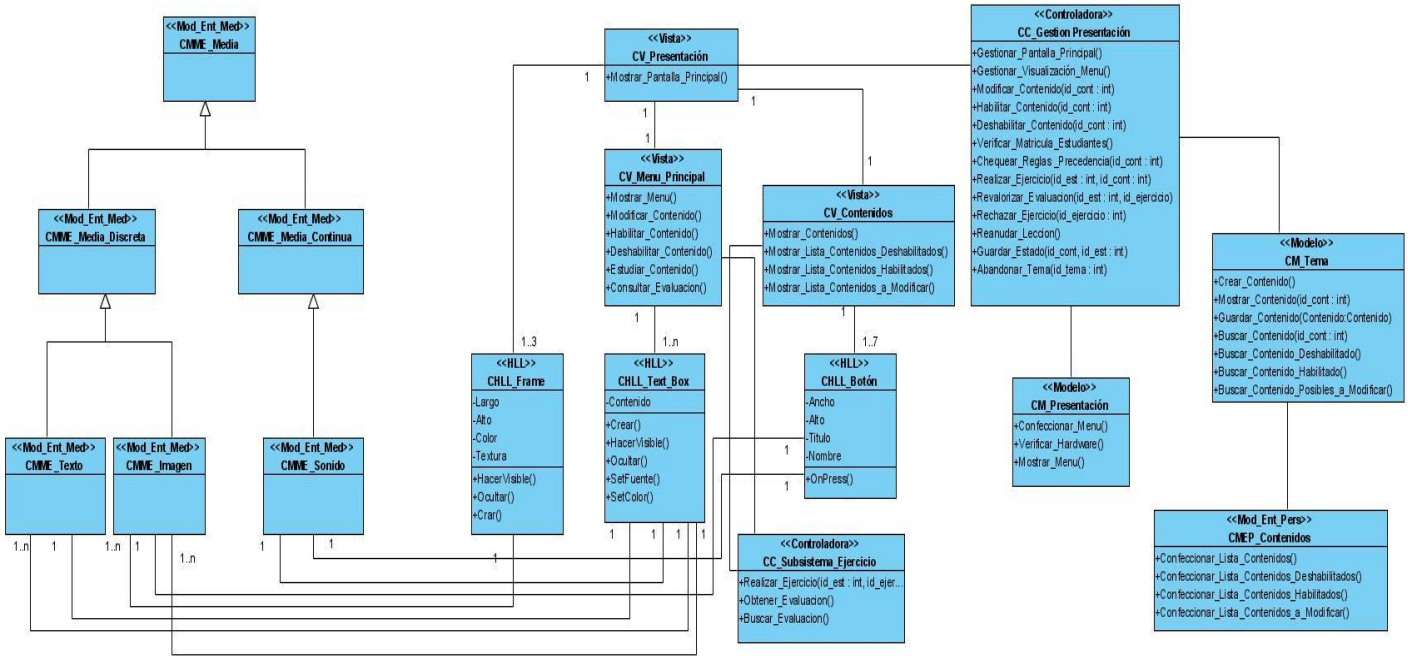


Figura 4: Diagrama de Clases del Diseño del caso de uso "Estudiar Contenido"

Este diagrama de clases está compuesto por las clases mencionada en el patrón (MVC-E) (*Ver Capítulo 1, epígrafe: Patrón Arquitectónico*) y las clases HLL que son las que corresponden al lenguaje de alto nivel seleccionado para la programación de la aplicación.

Para ver los Diagramas de Clases del Diseño de los restantes Casos de Uso **(Consultar Anexo #3 del documento de artefactos del módulo de Entrada y Salida).**

2.11 Patrones aplicados en el diagrama de Clases del Diseño.

Patrones de diseño

Fueron aplicados los patrones GRASP de responsabilidades para lograr una alta cohesión en las clases, donde los métodos que la componen posean alto nivel de correspondencia con la clase en cuestión. Igualmente está presente el patrón bajo acoplamiento, logrando con esto que cualquier cambio que pueda ocurrir en estas clases no tenga una alta repercusión en las demás.

También fue aplicado el patrón controlador, este está presente en las clases controladora pues es la que tiene la responsabilidad de controlar todas las acciones hechas por el usuario en las clases vistas y de dar las repuestas al mismo. Otro de los patrones utilizados fue el experto en información, para asignarle a una clase los métodos, en la que ella sea experta en la información referente a este.

Patrón Arquitectónico

Se empleó el patrón Modelo Vista Controlador Entidad (MVC-E), este es una extensión del patrón MVC que sufre cambios con el objetivo de representar detalladamente las clases y sus relaciones en las aplicaciones multimedia. Fue utilizado para una mejor estructuración del diagrama de clases siendo una propuesta empleada en el lenguaje de modelado ApEM-L.

2.12 Diagrama de Estructura de Navegación

El diagrama de navegación se encuentra dentro de la vista de presentación que define ApEM-L. El mismo representa las clases que posibilitan la navegación entre las distintas clases de la aplicación. ApEM-L define estas clases:

Clase menú (<<Menu>>): es el elemento de composición de una clase vista, donde se puede llegar a diferentes clases vistas con las cuales se conecta este menú, pues contiene una lista de las opciones de movimiento siguiente. Este 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).

Clase consulta (<<Consulta>>): esta clase permite el enlace con una clase vista a y 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

Clase índice (<<Indice>>): denota el valor de direccionamiento hacia una clase vista a partir de una opción determinada.

Clase botón (<<Boton>>): denota la existencia de un elemento interactivo del tipo botón para producir un camino en la navegación hacia una clase vista, y utilizando como intermediarias a clases de tipo consulta o índice.

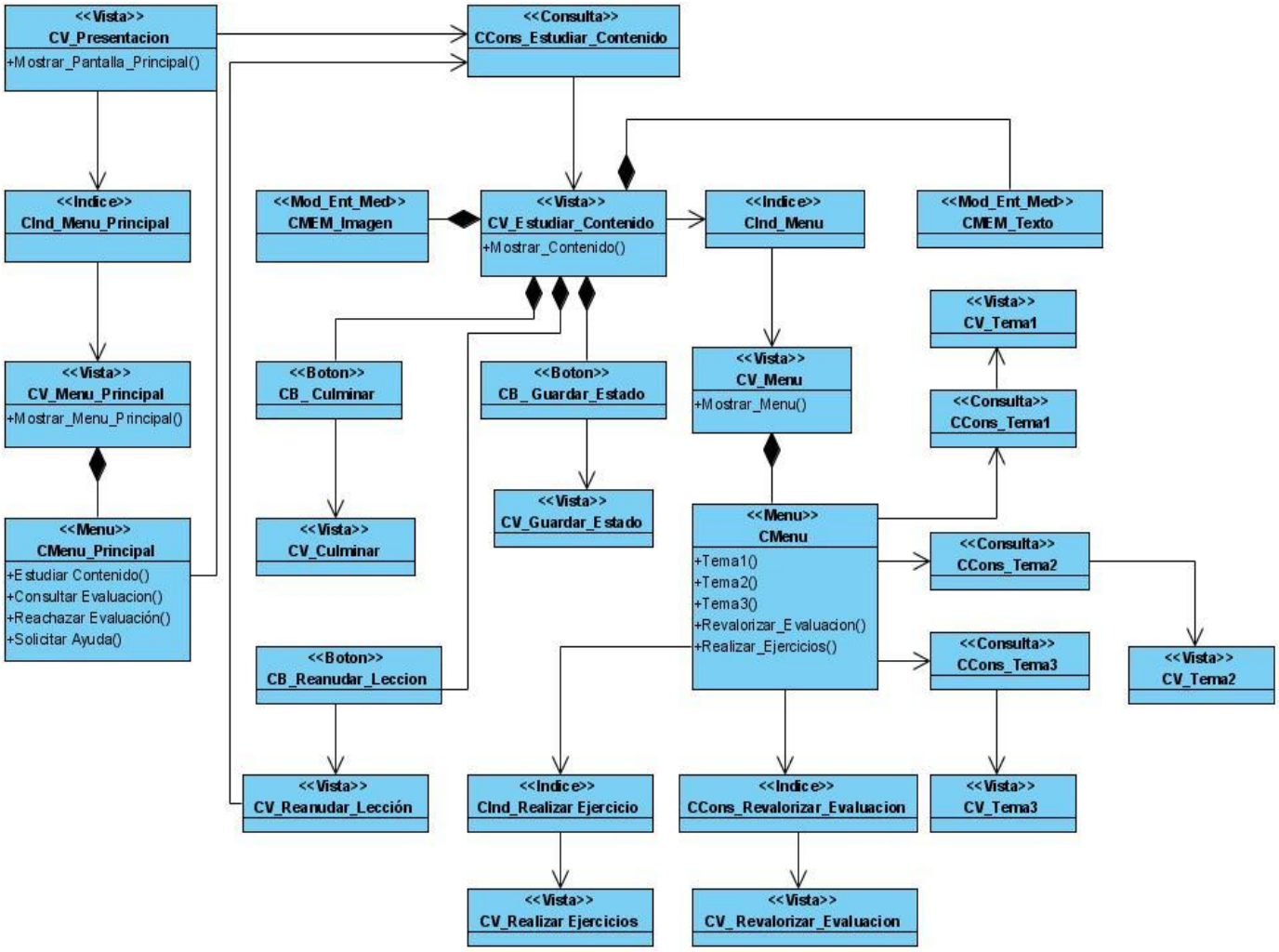


Figura 5: Diagrama de Estructura de Navegación del caso de uso "Estudiar Contenido"

Para ver los Diagramas de Estructura de Navegación de los restantes Casos de Uso **(Consultar Anexo #4 del documento de artefactos del módulo de Entrada y Salida).**

2.13 Diagrama de Estructura de Presentación

El diagrama de Estructura de Presentación es parte de la vista de Presentación en ApEM-L. El mismo, establece una organización lógica de los elementos que conforman cada interfaz de comunicación con el usuario en el futuro. Define dos nuevos tipos de clases:

Clase Estática: Agrupará los componentes que tienen como función visualizar información, pero no permiten interacción con el usuario.

Clase Interacción: Agrupará los elementos de la vista que permiten la interacción del usuario con el sistema informático creado.

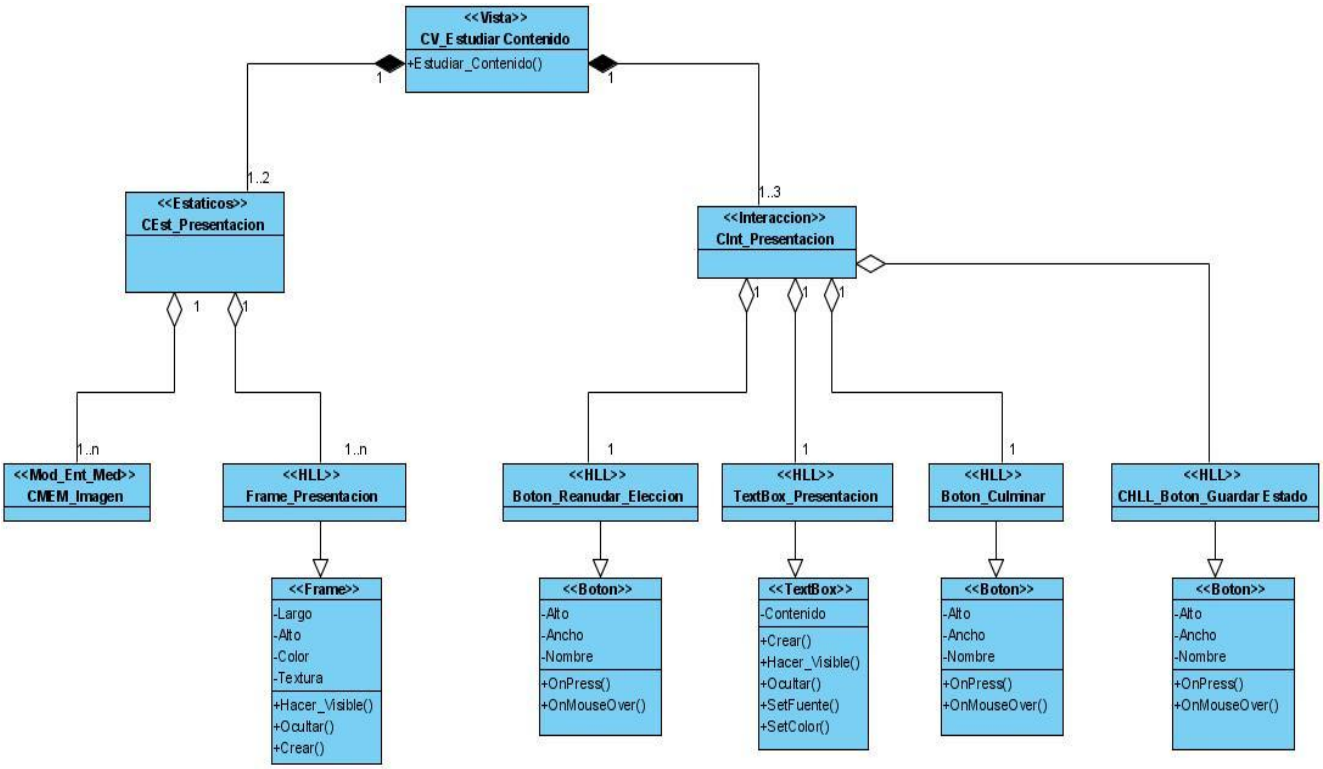


Figura 6: Diagrama de Estructura de Presentación del caso de uso "Estudiar Contenido"

Para ver los Diagramas de Estructura de Presentación de los restantes Casos de Uso **(Consultar Anexo #5 del documento de artefactos del módulo de Entrada y Salida).**

2.14 Diagrama de Secuencia

Los diagramas de secuencia representan el intercambio de mensajes entre las clases, permitiendo visualizar las relaciones entre ellas y sus responsabilidades.

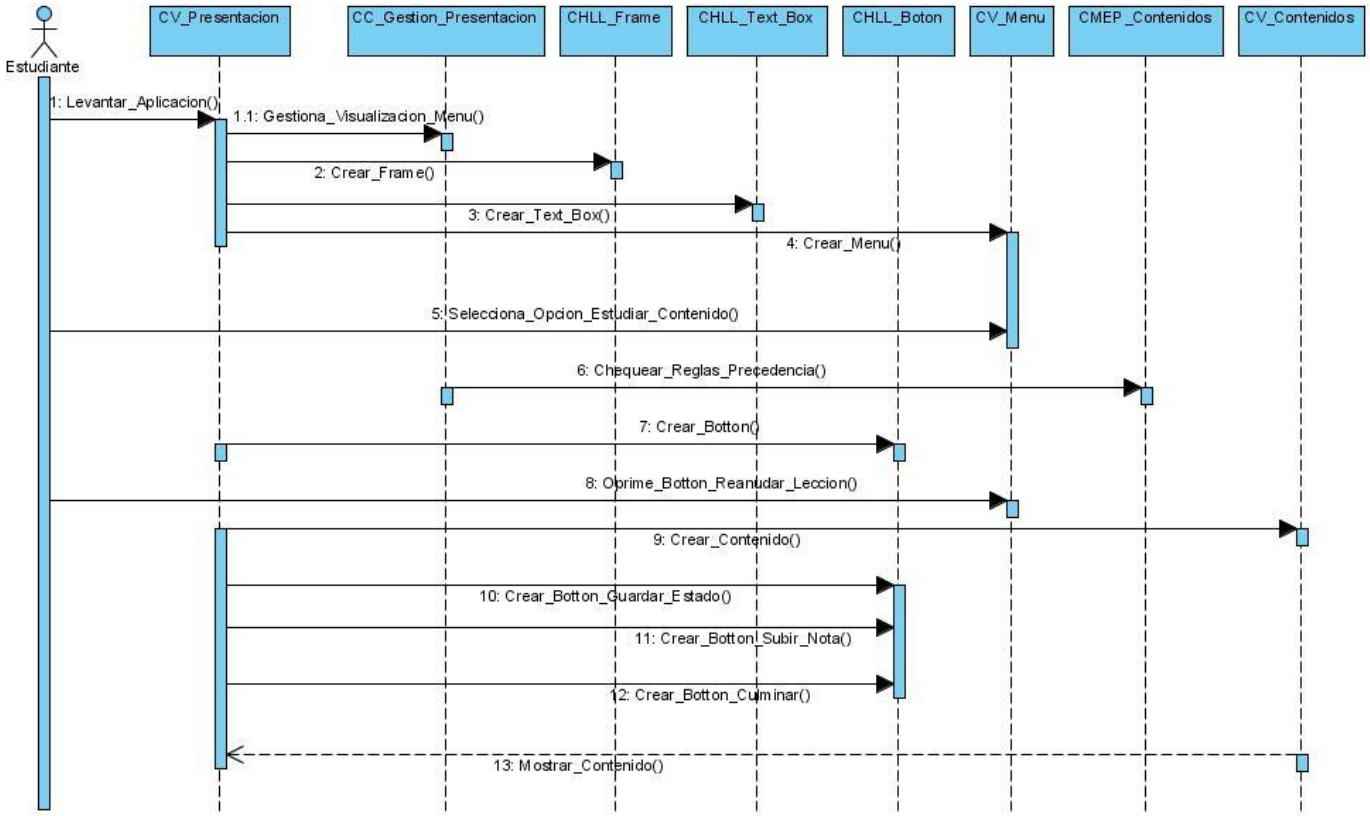


Figura 7: Diagrama de Secuencia del caso de uso "Estudiar Contenido"

Para ver los Diagramas de Secuencia de los restantes Casos de Uso **(Consultar Anexo #6 del documento de artefactos del módulo de Entrada y Salida).**

2.15 Conclusiones

En este capítulo se realizó el diseño de la aplicación que le permitirá a los desarrolladores realizar la implementación más adelante, para ello se obtuvieron los requisitos que permiten un mejor entendimiento de las funcionalidades que debe poseer la aplicación para satisfacer las necesidades del cliente. Además se elaboraron los distintos diagramas aplicando varios patrones ya antes mencionados, ayudando a mejorar la calidad de la construcción de los mismos.

Capítulo 3: Validación y Prueba

3.1 Introducción

En el presente capítulo se realizará la validación a través de métricas, estas se aplicaran con el objetivo de comprobar que el trabajo realizado se desarrolló con la calidad requerida, dándole pasos a actividades posteriores que permitirán la elaboración de la aplicación.

3.2 Métrica de la calidad de la especificación

Un elemento clave de cualquier proceso de ingeniería es la medición. Las medidas se emplean para entender mejor los atributos de los modelos que se crean. Pero, fundamentalmente, se emplean para valorar la calidad de los productos de ingeniería o de los sistemas que se construyen (Pressman, 2005).

Actualmente en el proceso de desarrollo de software están presentes un conjunto de métricas, las cuales se utilizan para la validación de los requisitos identificados en la realización de un software, estas métricas permiten validar de una manera correcta que los requisitos identificados durante el proceso de desarrollo tienen la calidad requerida y cumplen con las normas internacionales. (Piña, 2008)

Para medir la calidad de la Especificación en este trabajo se tendran en cuenta las características de especificidad (ausencia de ambigüedad), compleción, corrección y comprensión.

Antes de realizar el proceso de medición para la calidad de la Especificación de los requisitos es necesario conocer el valor total de requisitos:

$Rt = Rf + Rnf$

$Rt = 14 + 8 = 22$

Rt: Total de los requisitos.
Rf: Requisitos Funcionales
Rnf: Requisitos no funcionales

Para determinar la ***especificidad*** (ausencia de ambigüedad) de los requisitos se sugiere una métrica basada en la consistencia de la interpretación de los revisores para cada requisito:

$$Q1 = \frac{Rii}{Rt} = \frac{22}{22} = 1$$

Rii: Número de requisitos para los que todos los revisores tuvieron interpretaciones idénticas.

Rt: Total de los requisitos.

La ***compleción*** de los requisitos funcionales puede determinarse calculando la relación:

$$Q2 = \frac{na}{na + nb} = \frac{22}{22 + 0} = 1$$

na: Número de requisitos funcionales completos.

nb: Número de requisitos funcionales pobremente especificados.

Una especificación se considera correcta cuando cada requisito contenido en ella represente una característica que el sistema debe poseer. La ***corrección*** de los requisitos se define usando la siguiente ecuación:

$$Q3 = \frac{Rc}{Rc + Rnv} = \frac{22}{22 + 0} = 1$$

Rc: Número de requisitos que se han validado como correctos.

Rnv: Número de requisitos que no se han validado como correctos todavía.

La ***comprensión*** de los requisitos se determina a partir de la relación que se muestra a continuación.

$$Q4 = \frac{Rbc}{Rt} = \frac{22}{22} = 1$$

Rbc: Número de requisitos que todos los revisores entienden.

Rt: Total de los requisitos.

Revisores:

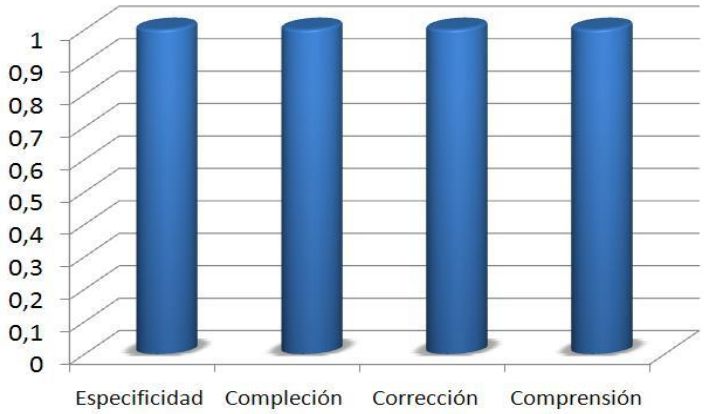
Ing. Aray Pérez Degue.

Ing. Yinett Hernández Hernández.

Ing. Carlos Y. Hidalgo García.

Ing. Susana Fadragas Rodríguez.

Capítulo 3: Validación y Pruebas



Resultados obtenidos.

En la gráfica mostrada se pueden observar los valores obtenidos después de haber aplicado las métricas correspondientes a la calidad de la especificación, los resultados muestran que esta se realizó con la calidad requerida porque entre más cercano sean los valores a 1 será mayor la calidad, logrando con esto un mejor entendimiento entre los cliente y desarrolladores.

Las métricas aplicadas permitieron demostrar que los requisitos fueron interpretados de la misma forma por los revisores, sin ambigüedad, pero además han sido incluidos y bien especificados todos los requisitos que la aplicación debe cumplir. Con la evaluación de la corrección y la comprensión se validó que fueron entendidos y que presentan todas las funcionalidades que la aplicación debe tener para su construcción.

3.3 Métricas para validar los casos de usos del sistema

Para realizar la validación de los casos de usos es necesario evaluar las siguientes propiedades de la calidad: consistencia, correctitud, completitud y complejidad, para esto, cada una de estas propiedades está compuesta por varios factores y estos se encuentran relacionado con una o varias métricas orientadas a objeto, esto permite conocer el grado de los factores que presentan una baja calidad.

Propiedades de la 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.

Se realizaron dos revisiones para obtener una mayor calidad de las métricas aplicadas.

Revisión 1

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. Numero de requisitos omitidos por casos de uso. Umbral<15	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<15	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%.
Total			**Para un incumplimiento: 2.1 %**
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:11 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 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%.

Total			**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%.
Total			**Para un incumplimiento de: 7.7 %.**

Tabla 2: Métricas de validación de Caso de Uso (Revisión 1).

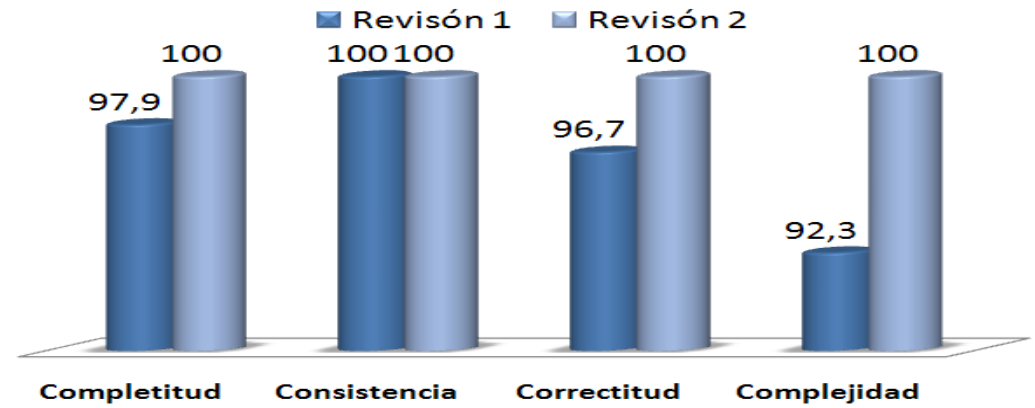
Revisión 2

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 3. Numero de requisitos omitidos por casos de uso. Umbral<15	Total de requisitos funcionales: 14 Número de requisitos omitidos por casos de uso: 0 Representa: 0 %.
	Factor 3. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	Métrica 4. Número de casos de uso que tienen requisitos omitidos. Umbral<15	Total de casos de uso: 10 Número de casos de uso que tienen requisitos omitidos: 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: 100%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en	Métrica 5. Número de casos que no han sido clasificados.	Total de casos de uso: 10 Número de casos que no han sido clasificados: 0

	(crítico, importantes y útiles)?	Umbral<10	Representa: 0%.
Total			**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:11 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%.
Total	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%.
Correctitud			**Para un incumplimiento de:** 0%

	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: 0 Representa: 0%.
Total			**Para un incumplimiento de:** 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%.
Total			**Para un incumplimiento de:** 0%.

Tabla 3: Métricas de validación de Casos de Uso (Revisión 2).



Resultados obtenidos:

En la gráfica mostrada se puede observar los valores iniciales y finales obtenidos por las propiedades de la calidad, completitud, consistencia, correctitud y complejidad. Luego de realizar la segunda revisión, los resultados expuestos, permitieron demostrar que los requisitos obtenidos se encuentran implementados en al menos un caso de uso, logrando con esto abarcar las necesidades del usuario, también se observó que los casos de uso fueron descritos con la calidad requerida para una mejor legibilidad de los mismos. Además con las métricas aplicadas, se validó que estos se encontraban adecuados a las funcionalidades del sistema y que todos estaban ubicados correctamente en el diagrama de caso de uso, dándole paso para entrar a la actividad de modelación de la aplicación.

Matriz de trazabilidad

La trazabilidad de los requisitos se realizó mediante la matriz de trazabilidad de requisito La primera columna pertenece a los requisitos y la primera fila a los casos de uso, se harán coincidir mediante una X en cada caso que exista relación. Mediante esta matriz se podrá saber si los casos de usos se encuentran relacionado con los requisitos.

	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

Resultado obtenido:

Después de haber realizado la tabla de trazabilidad y observado los resultados, se puede decir que los requisitos se encuentran relacionados con al menos un caso de uso, demostrando que el diagrama de caso de uso cumple con los requisitos descritos en la especificación.

3.4 Métricas para validar el diseño

Tamaño operacional de clase (siglas: TOC): Se refiere al número de métodos pertenecientes a una clase. Está determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización.

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.

Atributos de calidad evaluados por la métrica TOC.

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 en la complejidad de implementación de la clase.
Reutilización.	Un aumento del TOC implica una disminución del grado de reutilización de la clase.

Tabla 5: Atributos de calidad evaluados por la métrica TOC.

Para evaluar la métrica TOC se tuvo en cuenta la cantidad de procedimientos por clases a partir del promedio de las relaciones y mediante un criterio se obtuvo la categoría (baja, media, alta) para la Responsabilidad, Complejidad y Reutilización.

Criterios de evaluación para la métrica TOC.

	Categoría	Criterio
Responsabilidad	Baja	< =Promedio.

	Media	Entre Promedio y 2* Promedio.
	Alta	> 2* Promedio.
Complejidad 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: Criterios de evaluación para la métrica TOC

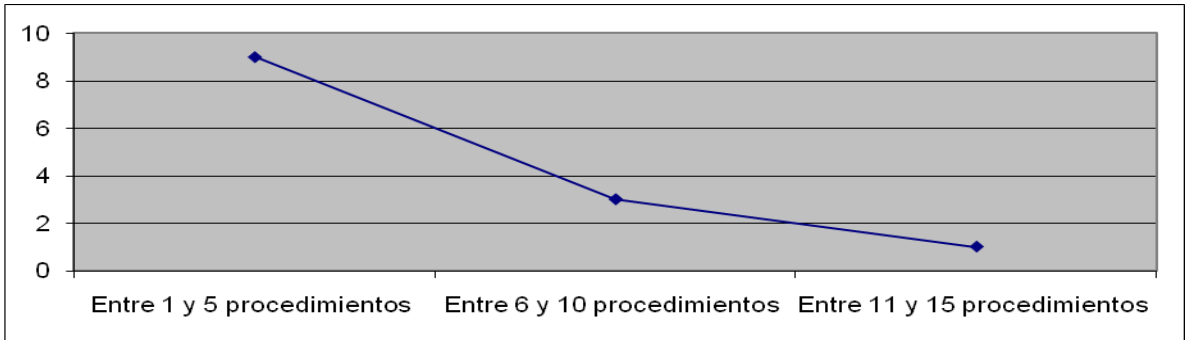


Figura 8 Representación de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

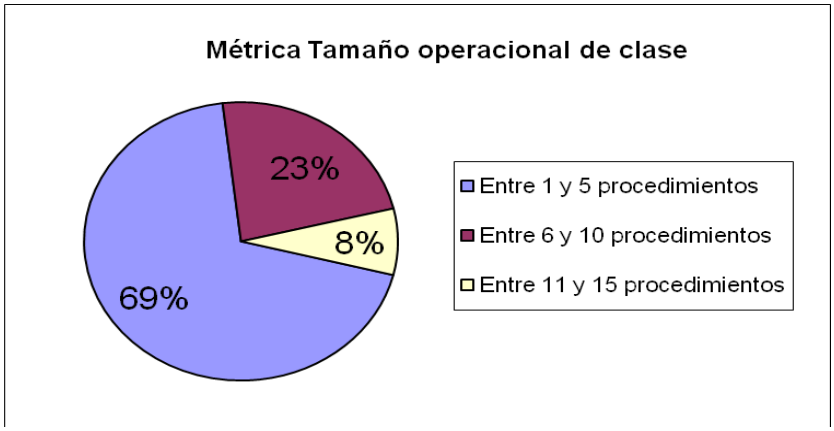


Figura 9 : Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

Representación de la incidencia de los resultados de la evaluación de la métrica TOC en cada uno de los atributos:

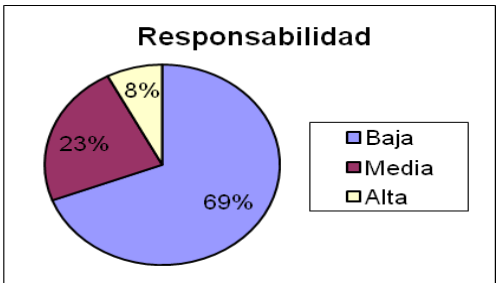


Figura 10: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

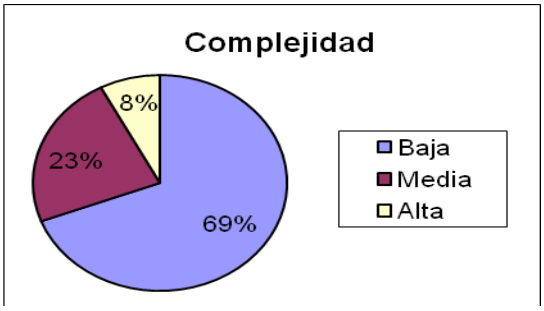


Figura 11: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

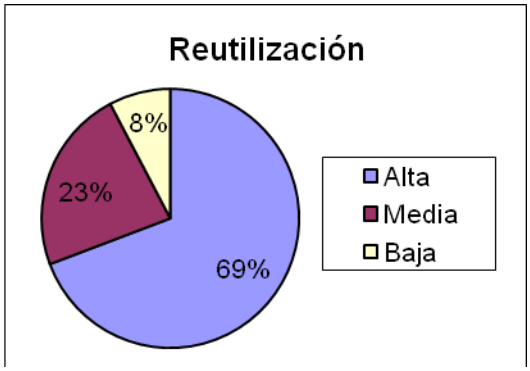


Figura 12: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Resultados obtenidos

Si se observan detenidamente las gráficas después de haber aplicado la métrica (TOC) se puede decir que el diseño propuesto se encuentra entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases posee menos cantidad de operaciones que la media registrada en las mediciones. que los atributos de calidad se encuentran en un nivel satisfactorio en el 69% de las clases; de manera que se puede observar cómo se fomenta la Reutilización siendo un elemento fundamental en el proceso de desarrollo de software y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

Otras de las métricas a aplicar para validar el diseño realizado es:

Relaciones entre clases (bajo) (RC)

Para aplicar la métrica RC se tuvo en cuenta la cantidad de relaciones que tenían las clases del componente Incidencias, a partir del promedio de las relaciones y mediante un criterio se obtuvo la categoría (baja, media, alta y ninguna en caso del Acoplamiento) y (baja, media, alta) para la Complejidad de Mantenimiento, Reutilización y Cantidad de Pruebas.

Con la presente métrica se evalúan algunos atributos de calidad mencionados en la métrica anteriormente aplicada, pero también se incorpora los siguientes atributos:

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.

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

Atributos de calidad evaluados por la métrica RC.

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 7: Atributos de calidad evaluados por la métrica RC.

Criterios de evaluación de la métrica RC.

	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad 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 8: Criterios de evaluación de la métrica RC.

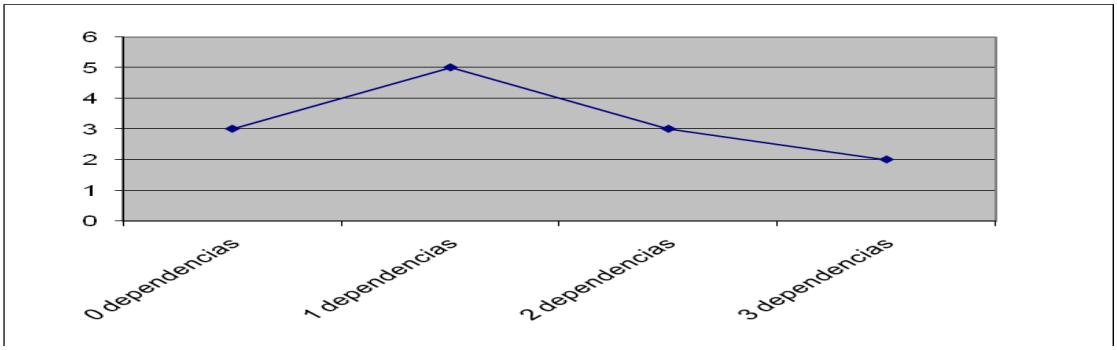


Figura 13: Gráfica de los resultados de la evaluación de la métrica RC agrupados por la tendencia de los Valores.

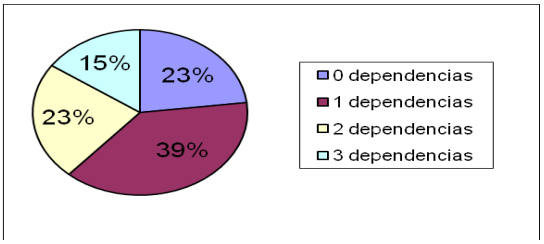


Figura 14: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

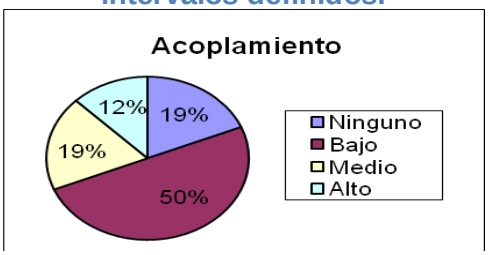


Figura 15: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Acoplamiento.

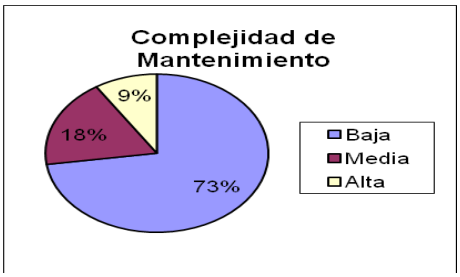


Figura 16: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Complejidad de Mantenimiento.

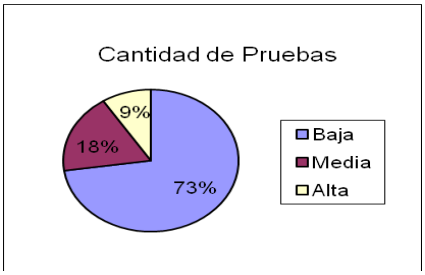


Figura 17: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Cantidad de Pruebas.

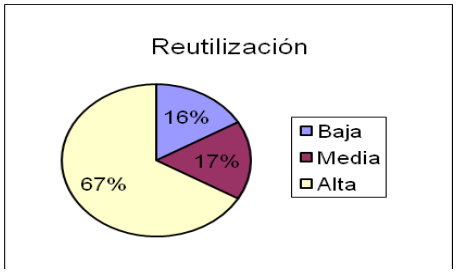


Figura 18: Representación de la incidencia de los resultados de la evaluación de la métrica RC en el atributo Reutilización.

Resultados obtenidos

Después de analizados los resultados obtenidos, luego de aplicar el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto se encuentra entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases poseen menos de 3 dependencias respecto a otras se puede concluir que los atributos de calidad se encuentran en un nivel satisfactorio; en el 50% de las clases el grado de dependencia o acoplamiento es mínimo, la Complejidad de Mantenimiento, la Cantidad de Pruebas y la Reutilización se comportan favorablemente para un 73%, 73% y 67% de las clases respectivamente.

3.5 Conclusiones

En este capítulo se aplicaron métricas para la validación de requisitos y casos de uso donde se obtuvieron valores significativos que permitieron evaluar la calidad y eficacia de los mismos. También se realizó la matriz de trazabilidad para verificar la relación entre los requisitos y casos de uso. Se validó además el diseño a través de la métrica TOC y RC arrojándose valores satisfactorios para cada uno de los atributos correspondientes.

Conclusiones

Conclusiones

Con la culminación del presente trabajo, se dio cumplimiento a los objetivos planteados, arribando a las siguientes conclusiones:

- Se realizó el estudio sobre los Laboratorios Virtuales, Objetos de Aprendizaje, SCORM y Recursos Didácticos y Sistemas de Autoaprendizaje permitiendo con esto un mejor entendimiento del sistema de aprendizaje que se está desarrollando.
- Para el diseño de la aplicación se realizó la identificación de los requisitos, y se desarrollaron todos los diagramas para lograr en un futuro la implementación de la aplicación.
- Se aplicaron métricas para la validación de la especificación de requisitos, casos de uso y el diseño permitiendo que el diseño elaborado tuviera la calidad requerida.

Recomendaciones

Recomendaciones

Se recomienda:

- Realizar la implementación de la aplicación del diseño realizado para el Módulo de Entrada y Salida del subsistema de autoaprendizaje para el Laboratorio Virtual de Sistemas Operativos.

Anexos

Bibliografía

Bibliografía Referenciada

Castañeda Hevia, Ángel Emilio. (2002) *¿Qué modelo, qué gestor y qué centro virtual de recursos debo comprar? ¿Cuándo y Como debo hacerlo? Conferencia No 2. Universidad Técnica de Ambato. Biblioteca electrónica del CREA.*

Wiley, David A. (2000) The Instructional Use of Learning Objects. Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. Universidad Estatal de Utah. Disponible en: http://www.reusability.org/read/chapters/wiley.doc

González, Yailin Molina y Martínez Peña, Evelio Rafael. 2010. *Diseño e implementación de un módulo de administración para el Sistema Unificado de Gestión de la Fuerza de Trabajo Calificada. Ciudad de la Habana : s.n., 2010.*

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

Valladares, Tania Teresa Laureiro y Aguilar Rodríguez, Lázaro Amed. 2008-2009. Análisis y Diseño de la solución Informática para el subsistema de Caja, del sistema de gestión empresarial Cedrux. La Habana: s.n., 2008-2009.

Ramil, Reinier Daniel. 2007. *"Módulo de cuestionario interactivo para prácticas de laboratorios virtuales sobre Web". Ciudad de la Habana : s.n., 2007.*

Sánchez, José Ignacio Peláez. 2005. *Lenguajes y Ciencias de la Comunicación. 2005.*

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

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *"El Proceso Unificado de Desarrollo de Software".* España: Addison Wesley, 2000. ISBN/ 8478290362.

Anexos

Morales, Dalgys La Rosa. 2010. *Ingeniería de Requisitos del Módulo Reportes del Sistema Nacional Público.* Ciudad de la Habana : s.n., 2010.

Pressman, Roger.S. 2005. *Ingeniería del Software. Un enfoque práctico.* 2005.

Estrada, Yelena Hernández. 2009. *Análisis del Módulo Proceso Confiscatorio de Bienes del proyecto Sistema Gestión Fiscal. Ciudad de la Habana : s.n., 2009.*

González, D. P., & Hernández Pérez, G. (2010). *"Guía para realizar Ingeniería de Requisitos a la línea de productos informáticos que utilizan tecnología multimedia".* La Habana.

León, Lisandra Consuegra and Frias Cañadilla, Mirtha Lianna. 2009. *Multimedia Interactiva para el aprendizaje de la Lengua Inglesa.* La Habana : s.n., 2009.

Pérez, Luis Angel Díaz. 2010. *Multimedia promocional de productos del Centro de Informática Médica (CESIM).* La Habana : s.n., 2010.

Pereira, Tamara Pérez and Cruz Álvarez, Yailén. 2007. *Propuesta de un Laboratorio Virtual de Física I sobre Fuerza de Fricción Seca. Estudio Preliminar.* La Habana : s.n., 2007.

Monge-Nájera, Julián. 1999. *La evolución de los laboratorios virtuales durante una experiencia de seis años con estudiantes a distancia.* 1999.

P.Vary, James. 1999. *Informe de la reunión de expertos sobre laboratorios virtuales, Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura.* 1999.

León, Lianyi Ramos and Pulido Piña, Yanelis. 2008. *Análisis de los módulos Planificación de Disco y Administración de Memoria de un Laboratorio Virtual de apoyo a la asignatura de Sistemas Operativos.* Habana, Cuba : s.n., 2008.

Anexos

Bibliografía Consultada

Lemus, Yudisleidys Peña and Hernández Díaz, Yuniersy. 2007. *SIMETSE – SIstema de METricas para evaluar el.* Ciudad de la Habana, : s.n., 2007.

Liber, Oleg (2000). *"Colloquia - a Conversation Manager". Campus Wide Information Systems 17(2). 56-62.*

Pressman, Roger.S. 2005. *Ingeniería del Software. Un enfoque práctico.* 2005.

Pichardo, Juan P. 1999. *Didáctica de los mapas conceptuales. México : s.n., 1999.*

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

Ramil, Reinier Daniel. 2007. *"Módulo de cuestionario interactivo para prácticas de laboratorios virtuales sobre Web". Ciudad de la Habana : s.n., 2007.*

Van Harmelen, Mark (August 2006). "Personal Learning Environments". http://octette.cs.man.ac.uk/jitt/index.php/Personal_Learning_Environments.

2010-2011. e-ABC. *e-ABC.* [Online] 2010-2011. [Cited: 2 8, 2011.] http://www.e-abclearning.com/queesscorm

Flores, Héctor Tapia. 2007. Blog Biblioteca Universidad Arturo Prat. *og Biblioteca Universidad Arturo Prat.* [Online] julio 12, 2007. [Cited: 2 8, 2011.] http://bibliopress.wordpress.com

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

González, D. P., & Hernández Pérez, G. (2010). *"Guía para realizar Ingeniería de Requisitos a la línea de productos informáticos que utilizan tecnología multimedia".* La Habana.

Cabrera Ramos, Juan Francisco; Castañeda Hevia, Ángel Emilio. (2003) *Modelo de Centro Virtual de Recursos para el desarrollo de habilidades de gestión de información y conocimientos en profesores y estudiantes universitarios.*

Anexos

Ricardo, Febe Ángel Ciudad y Herrera Martínez, Yosnel. DISEÑO DE APLICACIONES EDUCATIVAS MULTIMEDIAS UTILIZANDO UNA NUEVA VERSIÓN DEL LENGUAJE DE MODELACIÓN ApEM – L. Ciudad de La Habana. : s.n.

NUEZ, B. L. D. L. Programa Analítico de Sistemas Operativos I, 2005.

Zilberstein Toruncha, José; Collazo Delgado, Ramón y otros. (2004) Fundamentos del Modelo Universidad para la Autoeducación Cujae (UAC). Informe parcial a CITMA. Base datos AGIC-CREA

Downes, Stephen (October 2005). "E-learning 2.0".
http://www.elearnmag.org/subpage.cfm?section=articles&article=29-1

Suárez Guerrero C. Los entornos virtuales de aprendizaje como instrumento de mediación. Rev. Electrónica de la Universidad de Salamanca. [Online]. Vol.4. 2003. [Citado 20 de agosto de 2008]. Disponible en: http://bvs.sld.cu/revistas/ems/vol22_1_08/El_incremento_de_la_motivación_de_los_estudiantes.

Gisbert Cervera Mercè. *Las Tecnologías de la Información y la Comunicación como favorecedoras de los procesos de autoaprendizaje y de formación permanente. Educar 25, 1999*

Gutierrez, Jorge A. Saavedra. El Mundo Informatico y tú en que mundo vives? *El Mundo Informatico y tú en que mundo vives?* [Online] [Cited: mayo 6, 2011.]

Balda, José Manuel Medina and López López, María Gertrudis. *Metodología De Contrucción De Objetos De Aprendizaje.*

Compendium Institute. [Online] [Cited: Mayo 8, 2011.] http://compendium.open.ac.uk/institute/