Universidad de las Ciencias Informáticas

Facultad 3

"*Análisis y Diseño de un Evaluador de los contenidos del tema Administración de Memoria para el Laboratorio Virtual de la Asignatura Sistemas Operativos*"

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

Autor: Osmel Vento García

Tutor: Ing. Carlos Yasmany Hidalgo García

Ciudad de la Habana, Cuba

Julio, 2011

"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."

Albert Einstein

Declaración de Autoría

DECLARACIÓN DE AUTORÍA

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

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

_____

Firma del autor

(Osmel Vento García)

_____

Firma del tutor

(Carlos Y. Hidalgo García)

Agradecimientos

Agradecimientos

A mis amados padres por todo el apoyo que me han dado, por estar siempre ahí cuando los necesite, por los consejos, las buenas conversaciones y las buenas costumbres que han dejado en mí. Por preocuparse por mí, por mostrarme siempre el camino correcto y demostrarme que todo es posible. Por ser los mejores, por su confianza y por su amor.

A mis queridas abuelas y abuelos que en el transcurso de mi vida siempre han estado orgullosos de mí.

A la persona que amo: Elizabeth por las cosas que hemos compartido, por su preocupación, por la gran paciencia siempre ha tenido conmigo, por estar en todos los momentos alegres y tristes, por toda la felicidad, el cariño y amor.

A mi segunda madre (Bertica) porque siempre me ayudó en los años de Universidad y me cuidó como uno más de sus hijos.

A mi padrastro por su apoyo en los momentos que lo necesité.

A toda mi familia: A mis primos y tíos por su preocupación hacia mi porvenir.

A la familia de mi novia, a mis suegros y sus padres, y al hermano de mi novia, a todos por confiar siempre en mí.

A todas mis amistades: Yosnel, Los Pedro, Juan, Carlos, Manuel, Piry, Raydel, Ruben, Rohandy, Yunieski, a los amigos que he conocido todos estos años y que siempre recordaré.

A todos los representantes del tribunal porque sin ellos no hubiera sido posible alcanzar esta meta.

A todos los profesores que me sirvieron de guía estos años.

Dedicatoria

Dedicatoria

A mi madre por ser lo más grande en mi vida, por ser mi universo.

A mi padre por todo su apoyo y comprensión, porque gran parte de lo que soy se lo debo a él.

A mis amados padres por todo lo que han hecho por mí, por lo que me demuestran día tras día.

A mi novia por ser tan especial en mi vida.

Resumen

En la actualidad las Tecnologías de la Información y las Comunicaciones (TICs) forman parte de la cultura tecnológica que rodea el mundo y con la que se debe convivir. La gama de servicios que brindan posibilita la creación de modelos, procesos y programas; lo que permite controlar y simular actividades reales o virtuales. Es por ello que el presente trabajo está enmarcado en el análisis y diseño de un evaluador que apoye la visualización de los contenidos del tema Administración de Memoria en la asignatura Sistema Operativo puesto que la misma es un poco compleja, debido al alto nivel de abstracción que se necesita para impartir el contenido de algunos temas. Con este fin se realizó un estudio sobre: conceptos asociados a laboratorios virtuales determinando las ventajas y desventajas que poseen, además de la simulación por computadora, un aspecto fundamental para el desarrollo del trabajo, metodologías de desarrollo de software, herramientas de modelado, entre otros aspectos asociados a la Ingeniería de Software. Finalmente, se aplicaron métricas y pruebas dirigidas a evaluar la calidad de los requisitos y el sistema, obteniéndose resultados satisfactorios.

PALABRAS CLAVE

"Laboratorio Virtual, Simulación, Evaluación, Administración de Memoria, Sistema Operativo, Tecnología"

Índice de Contenido

Contenido

Índice de Contenido

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

Introducción

La informática es una de las ciencias que se encuentra en 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. Entre ellas se encuentra 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 la docencia será siguiendo una enseñanza semipresencial con uso protagónico de los Entornos Virtuales de Aprendizaje (EVA).

El uso de los EVA en la UCI para contribuir al desarrollo del 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 (SO) es fundamental dentro del plan de estudio del tercer año de la carrera de Ingeniería en Ciencias Informáticas, la cual está dividida en cuatro temas fundamentales que comprenden los principales aspectos referentes al diseño y construcción de los mismos como son: Gestión de Procesos, Gestión de Memoria, Gestión de Entrada y Salida de Información.

La enseñanza del tema Gestión de Memoria consta de dos sistemas de conocimientos: el primero es Administración de Memoria, Sistemas Monoprogramados y Multiprogramados; y la segunda es Sistemas de Memoria Virtual. El contenido de Administración de Memoria se torna un poco complejo tanto para impartirlo y evaluarlo el profesor como para recibirlo los estudiantes, puesto que cuenta con un componente teórico muy fuerte resultando difícil para los estudiantes la realización de ejercicios prácticos.

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 y la amplia y adecuada utilización de las TICs son los componentes principales para lograr la integración docencia-producción-investigación; y donde el estudiante sea capaz de realizar un autoestudio y sea el protagonista de su proceso de aprendizaje.

Introducción

En virtud de satisfacer las necesidades existentes de materiales didácticos y sistemas que apoyen el aprendizaje de la asignatura de SO se comenzó a desarrollar un Laboratorio Virtual (LV) que abarcará todos los temas de la misma, permitiendo acceso a las conferencias y prácticas simuladas de la materia. Sin embargo, dentro de su concepción ninguna de las aplicaciones proporcionan al alumno elementos pedagógicos que les permita evaluarse y poner en práctica los conocimientos impartidos en clases específicamente en el contenido de Administración de Memoria.

Bajo estas condiciones es identificado el siguiente **problema de la investigación**: ¿Cómo transformar las necesidades del cliente a un lenguaje comprensible por los desarrolladores que facilite el desarrollo de un evaluador de los contenidos del tema Administración de Memoria para el Laboratorio Virtual de la asignatura Sistemas Operativos?

Este problema se enmarca en el **objeto de estudio**: Proceso de desarrollo de software.

En este sentido la investigación que se presenta tiene como **objetivo general**: realizar el Análisis y Diseño de un evaluador de los contenidos del tema Administración de Memoria para el Laboratorio Virtual de la asignatura Sistemas Operativos.

El **campo de acción** se centra en el: Análisis y Diseño de un evaluador de los contenidos del tema Administración de Memoria para el Laboratorio Virtual de la asignatura Sistemas Operativos.

Se define como **idea a defender:** realizando el Análisis y Diseño de un evaluador de los contenidos del tema Administración de Memoria se facilitará el desarrollo de un evaluador para el Laboratorio Virtual de la asignatura Sistemas Operativos.

Para alcanzar la meta propuesta y orientada fundamentalmente a proveer las especificaciones del Análisis y Diseño se han derivado un conjunto de **objetivos específicos:**

- Elaborar el marco teórico de la investigación.
- Realizar los modelos de Análisis y Diseño.
- Validar los artefactos obtenidos.

En aras de lograr un trabajo con mejor calidad se trazaron varias **tareas**, las cuales se muestran a continuación:

- Estudio sobre los laboratorios virtuales, conceptos, tipos, características, así como sus ventajas y desventajas en su aplicación.
- Estudio sobre los simuladores, conceptos, tipos, características y sus aplicaciones.

Introducción

- Estudio de los contenidos del tema de Administración de Memoria dentro de la asignatura Sistemas Operativos.
- Estudio de las metodologías que se utilizan para el desarrollo de software, los lenguajes de modelado y las herramientas CASE.
- Estudio y selección de los Patrones de Diseño más factibles para esta propuesta de solución.
- Estudio y selección de las métricas para validar los resultados obtenidos.
- Realización del Modelo de Dominio.
- Identificación de los requisitos funcionales y no funcionales.
- Elaboración del Modelo del Sistema
- Aplicación de métricas para validar los requisitos de software 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 autogestionado 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.

Resultados esperados

- Modelo de dominio.
- Especificación de requisitos de software.

- Modelo del sistema.
- Prototipo no funcional del sistema.
- Diagrama de Casos de Uso del Sistema.
- Diagramas de Clases del Análisis.
- Diagramas de Interacción.
- Diagrama de Clases del Diseño.

A lo largo del presente trabajo se muestran 3 capítulos de los cuales se enuncia una síntesis a continuación.

Capítulo 1: Fundamentación teórica. En este capítulo se resume el estudio realizado sobre laboratorios virtuales y la simulación por computadora, se aborda el tema de las metodologías de desarrollo de software, de los lenguajes de modelado, las herramientas CASE y se fundamenta su selección para el proceso de desarrollo. Se estudia lo referente a la Ingeniería de Requisitos, sus etapas y técnicas. Además se ve lo concerniente al diseño de sistemas y las métricas para su validación.

Capítulo 2: Descripción de la solución propuesta. En este capítulo se describe la solución propuesta para el problema planteado mediante el Modelo de Dominio. Se describen además los requisitos establecidos y se definen los actores, los casos de usos con sus descripciones, así como sus diagramas. Se definen y realizan los diagramas de Clases del Análisis, Interacción, Clases del Diseño así como los Patrones de Diseño y todo lo referente al análisis y diseño de la aplicación.

Capítulo 3: Validación de los resultados. En este capítulo se aplican las métricas de la calidad de la especificación de los requisitos y para validar los casos de uso del sistema, haciendo un análisis de los resultados obtenidos en términos de ambigüedad y estabilidad de los requisitos especificados, y la completitud, complejidad, correctitud y comportamiento de los casos de uso obtenidos. Además, se aportan elementos técnicos esenciales en relación con la validación del diseño de las clases.

Capítulo 1: Fundamentación Teórica

Fundamentación Teórica

1.1 Introducción

En el presente capítulo se abordarán diferentes temas entre los que se encuentran la definición de un Laboratorio Virtual, así como los distintos tipos que existen y los que son afines a la investigación. También se exponen los beneficios e inconvenientes que estos presentan en conjunto con su función dentro del proceso de enseñanza. Simultáneamente con ellos se estudia la simulación por computadora y los beneficios que aportan a los estudiantes. Se realiza además un estudio sobre metodologías de desarrollo, herramientas CASE (por sus siglas en inglés Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) y lenguajes de modelado que se proponen utilizar en la realización de este trabajo.

1.2 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 para 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”. (Vary J. P., 2000)

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

Después de estudiar las definiciones dadas por varios autores, se define que un Laboratorio Virtual no es más que un entorno de simulación y experimentación a distancia para realizar el conjunto de prácticas de laboratorio que son necesarias para poder superar las distintas asignaturas.

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

1.2.1 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). (Velázquez, 2008)

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 (Herías, 2003)

A pesar de ser uno de los más usados en el mundo se considera que no es factible para la concepción de la presente investigación puesto que al trabajar como software independiente se torna difícil incluirlo dentro de la plataforma Moodle a la par de los demás módulos.

1.2.2 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. (Nájera Estrada & Méndez, 2007)

El Laboratorio virtual de Sistemas Operativos para el cual se modela el evaluador de los contenidos del tema Administración de Memoria clasifica entre los laboratorios virtuales de tipo web, permitiendo integrarse a la plataforma Moodle (utilizada en la UCI para la gestión de recursos) donde todos los usuarios de la institución tendrán acceso al sistema desde diferentes ubicaciones y de forma simultánea.

1.2.3 Laboratorios remotos.

Los laboratorios remotos son aquellos que permiten operar remotamente cierto equipamiento, bien sea didáctico: maquetas específicas o industriales, 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. (Herías, 2003)

Estos laboratorios como en su definición incluye, dependen de una tecnología mayor no encontrada en nuestro país para su funcionamiento, generalmente son solo usados en países desarrollados no cumpliendo con los objetivos del presente trabajo.

1.3 Laboratorios virtuales en el mundo

El desarrollo de estos laboratorios ha sido desplegado alrededor de todo el mundo, unos más sofisticados que otros, debido a las necesidades que presente cada país pero de una forma u otra tratan de llegar a su objetivo final. Dentro de los encontrados se tiene el “Laboratorio Virtual de Computación para Cursos en Línea”, en la Universidad de Baja California para cursos de licenciatura y posgrado en el área de Ingeniería de Software se trabaja con el sistema de Moodle y demás recursos Web para el desarrollo de aplicaciones para internet. En él se explica el uso de herramientas de software basado en código abierto y

de libre distribución de tipo GNU\Linux, de forma tal que alimente los esquemas educativos a distancia en modalidades en línea como la semipresencialidad o las videoconferencias.

Microsoft es una empresa multinacional de origen estadounidense dedicada al sector de la informática y entre sus aportes creó un LV de TechNet y MSDN (Microsoft Developer Network o Red de Desarrolladores de Microsoft) gratis, donde no necesita ser instalado en ninguna computadora personal y puede ser completado en 90 minutos o menos, permitiéndoles a las personas aprender, practicar y evaluar las tecnologías y productos de esta empresa.

Los laboratorios virtuales se enmarcan en lo que se conoce como Entornos Virtuales de Aprendizaje, capaces de asegurar una continua comunicación (virtual) entre el estudiante y el profesor. Cuba pone como prioridad la educación, considerada como un país libre de analfabetismo y en aras de continuar esforzándose en esta rama se ha ido desarrollando en la informática y a partir de ello ha creado en algunas de sus universidades laboratorios virtuales:

- Entorno virtual de Física de la Universidad de las Ciencias Informáticas, la misma consta de 8 prácticas de LV para el estudio de algunos temas de la asignatura.
- SIDEF (Sistema Interactivo Didáctico de Enseñanza de la Física) utilizado en el Departamento de Física de la Facultad de Matemática, Física y Computación de la Universidad Central "Marta Abreu" de Las Villas.
- Laboratorio Virtual de Química General de la Universidad de La Habana con el cual se puede aprender Química desde su investigación hasta realizando prácticas virtuales.
- Laboratorio Virtual en Anestesiología utilizado en Hospital General Docente "Aleida Fernández Chardiet" Güines, La Habana donde sirve de apoyo al aprendizaje y la investigación de técnicos y profesionales en el campo de las Ciencias Médicas del territorio.

Todos estos laboratorios han sido creados con un mismo fin, sin embargo, existen una serie de argumentos que justifican el porqué es necesario crear uno nuevo en nuestro centro:

- Debido al nuevo modelo de formación donde ya en el tercer año de la carrera la mayoría de los estudiantes están vinculados a proyectos productivos se hace necesario realizar una herramienta que les permita los alumnos poner en práctica sus conocimientos, específicamente en la asignatura de Sistemas Operativos generando un aumento en los resultados académicos.

- Desde el punto de vista económico el país no cuenta con mucho desarrollo referente a las últimas actualizaciones que brindan los laboratorios presenciales por lo que se han propuesto realizar práctica de LV de forma tal que mejoren los resultados de dicha disciplina.

- De los LV visitados se detectaron que aunque incluyen varias características en común ninguno reúne los requisitos que persigue la universidad.

- Algunos presentan como características que no necesitan de un servidor para su funcionamiento por lo que no se basa en los criterios que debe cumplir un LV web.

- Aunque algunos de los LV visitados están adaptados a la enseñanza, ninguno está dedicado a los SO por lo que se hace necesario crear uno con las características independientes de la asignatura.

- La mayoría necesitan de internet y el país actualmente carece de medios que le permitan una adecuada conexión para la elaboración de estas prácticas virtuales. Además, que una vez realizada la simulación todo el contenido tratado será cerrado por lo que no favorece el aprendizaje, no cumpliendo con los objetivos que pretende la investigación.

1.4 Ventajas y desventajas 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 comunicación en las prácticas educativas y sociales para beneficio de los estudiantes. El aprendizaje está basado en simulaciones. Pero además de esto tiene otras ventajas 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.

- La simulación en el laboratorio virtual, permite obtener una visión más intuitiva de aquellos fenómenos que en su realización manual no aportan suficiente claridad gráfica. (Herreros & R., 2005)

- Es una herramienta de auto aprendizaje. (Herreros & R., 2005)

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

1.5 Simulación por computadora

La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con el con la finalidad de comprender el comportamiento del sistema o evaluar nuevas estrategias dentro de los límites impuestos por un cierto criterio o un conjunto de ellos para el funcionamiento del sistema. (Izquierdo, 2008)

La simulación por computadora es de mucha utilidad para el aprendizaje de los estudiantes en la asignatura Sistema Operativo ya que la integración de un software educativo permitirá la incorporación de la tecnología informática, aportando herramientas que favorezcan el desempeño profesional de los alumnos, podrán experimentar en entornos que representan la realidad, a través de modelos de la misma, pero de una forma más interactiva y constructivista, además de fomentar la creatividad, el aprendizaje por descubrimiento y la enseñanza individualizada. Al mismo tiempo le es de mucha ayuda al estudiante en su proceso de auto aprendizaje porque puede observar paso a paso lo que quiere aprender de una forma más amena y apreciable. (Cárdenas, 2009)

1.5.1 Tipos de simuladores

- **Simulador de conducción:** permiten a los alumnos de autoescuela enfrentarse con mayor seguridad a las primeras clases prácticas, además de permitirles practicar de manera ilimitada situaciones específicas (aparcamientos, incorporaciones desde posiciones de escasa visibilidad, conducción en condiciones climatológicas adversas). Uno de estos simuladores es SIMESCAR.
- **Simulador de carreras:** es el tipo de simulador más popular; se puede conducir un automóvil, motocicleta, camión. Ejemplos: rFactor, GTR, GT Legends, toca racer.
- **Simulador de vuelo o de aviones:** permite dominar el mundo de la aviación y pilotar aviones, helicópteros. Ejemplo: Microsoft Flight Simulator, X-Plane
- **Simulador de trenes:** permite controlar un tren. Ejemplo: Microsoft Train Simulator, Trainz, BVE Trainsim.
- **Simulador de vida o de dinámica familiar:** permite controlar una persona y su vida. Ejemplo: Los Sims.
- **Simulador de negocio:** permite simular un entorno empresarial. Es posible jugar diferentes roles dentro de las funciones típicas de un negocio. Ejemplo: EBSims, Market Place, Flexsim.
- **Simulador político:** permite rolar como político. Ejemplo: Las Cortes de Extremapol, Política xxi, Simupol, Dolmatovia.
- **Simulador de redes:** permite simular redes. Ejemplo: Omnet++, ns2.

- **Simulador clínico médico:** permite realizar diagnósticos clínicos sobre pacientes virtuales. El objetivo es practicar con pacientes virtuales casos clínicos, bien para practicar casos muy complejos, preparando al médico para cuando se encuentre con una situación real o bien para poder observar como un colectivo se enfrenta a un caso clínico, para poder sacar conclusiones de si se está actuando correctamente, siguiendo el protocolo de actuación establecido. Ejemplo: Simulador clínico Mediteca.
- **Simulador musical:** permite reproducir sonidos con un instrumento de juguete. Ejemplo, Guitar Hero, Dj Hero, Band Hero de Activision Blizzard y Rock Band de Harmonix

1.5.2 Simuladores en la educación

Los simuladores constituyen un procedimiento tanto para la formación de conceptos y construcción en general de conocimientos, como para la aplicación de éstos a nuevos contextos a los que, por diversas razones, el estudiante no puede acceder desde el contexto metodológico donde se desarrolla su aprendizaje. (Ruiz Gutiérrez, 2000)

De hecho, "buena parte de la ciencia puntera, de frontera, se basa cada vez más en el paradigma de la simulación, más que en el experimento en sí...". (Wagensberg, 1993)

Mediante los simuladores se puede por ejemplo desarrollar experimentos de química en el laboratorio de informática con mayor seguridad, es así como si a un estudiante se le ocurre agregar más de un determinado líquido la explosión que esto cause será una simple "simulación", cuando vaya a realizarlo en la práctica él estará informado de las consecuencias de este proceso.

Características en la educación (Ruiz Gutiérrez, 2000)

- Apoyan el aprendizaje de tipo experimental y conjetural.
- Permite la ejercitación del aprendizaje.
- Suministran un entorno de aprendizaje abierto basado en modelos reales.
- Alto nivel de interactividad
- Tienen por objeto enseñar un determinado contenido.
- El usuario trata de entender las características de los fenómenos, cómo controlarlos o qué hacer ante diferentes circunstancias.
- Promueven situaciones excitantes o entretenidas que sirven de contexto al aprendizaje de un determinado tema.

- El usuario es un ser activo, convirtiéndose en el constructor de su aprendizaje a partir de su propia experiencia.

1.6 Metodologías que se utilizan para el desarrollo de software.

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. Las metodologías imponen un proceso disciplinado sobre el desarrollo de software con el fin de hacerlo más predecible y eficiente. Lo hacen desarrollando un proceso detallado con un fuerte énfasis en planificar inspirado por otras disciplinas de la ingeniería, también su propósito es establecer un contrato social entre todos los participantes en un proyecto para conseguir la solución más eficaz con los recursos disponibles. (Falgueras, 2003)

A continuación se describen las metodologías estudiadas: Proceso Unificado de Desarrollo de Software (RUP), Microsoft Solution Framework (MSF) y Programación Extrema (XP).

1.6.1 Proceso Unificado de Desarrollo de Software (RUP).

RUP es un proceso para el desarrollo de un producto de software que define quién, cómo, cuándo y qué debe hacerse en el proyecto. Se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y ser iterativo e incremental. Su evolución está medida por fases e iteraciones, que comprenden las etapas por las que transita el producto hasta ser concluido. Entre sus características más importantes están (Mendoza M. A., 2004):

- Guía a los equipos de proyecto en cómo administrar el desarrollo iterativo de un modo controlado mientras se balancean los requerimientos del negocio, el tiempo al mercado y los riesgos del proyecto.
- Describe los diversos pasos involucrados en la captura de los requerimientos y en el establecimiento de una guía arquitectónica lo más pronto posible, para diseñar y probar el sistema hecho de acuerdo a los requerimientos y a la arquitectura.
- Describe qué entregables producir, cómo desarrollarlos y también provee patrones.
- Es soportado por herramientas que automatizan, entre otras cosas, el modelado visual, la administración de cambios y las pruebas.
- Asigna una gran importancia a la captura de requisitos y la selección de un buen líder de proyecto para garantizar el trabajo del equipo de desarrollo.
- Realiza un gran número de artefactos lo que puede provocar retrasos por mala preparación de los analistas.

- Las responsabilidades están divididas y es aplicable a todo tipo de proyecto asumiendo sus extensiones.

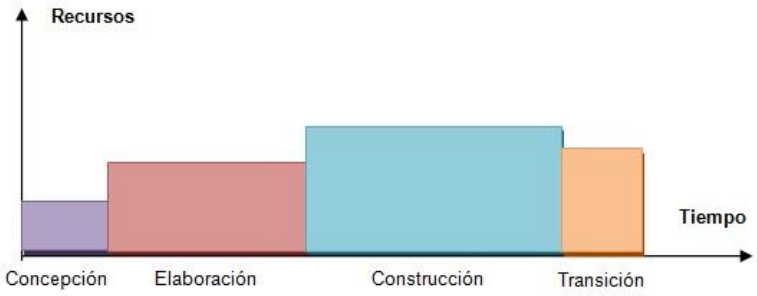
La metodología RUP (Rational Unified Process) es una metodología para la ingeniería de software que va más allá del simple análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software.

Ésta se caracteriza por ser guiado por casos de uso, los mismos son el instrumento para validar la arquitectura del software y extraer los casos de prueba; centrado en la arquitectura, es donde los modelos son proyecciones del análisis y el diseño que constituyen la arquitectura del producto a desarrollar; iterativo e incremental, durante todo el proceso de desarrollo se producen versiones incrementales del producto en desarrollo. (Mendoza M. A., 2004)

Esta metodología se divide en cuatro fases el proceso de desarrollo:

Concepción: se hace mayor énfasis en actividades de modelado del negocio y de requerimientos. La **Elaboración** con el objetivo de determinar la arquitectura óptima. Posteriormente, la de **Construcción**, en la que se lleva a cabo la construcción del producto por medio de una serie de iteraciones y por último **transición,** con el objetivo de obtener el producto preparado para su entrega a la comunidad de usuarios. (Sanchez, 2004)

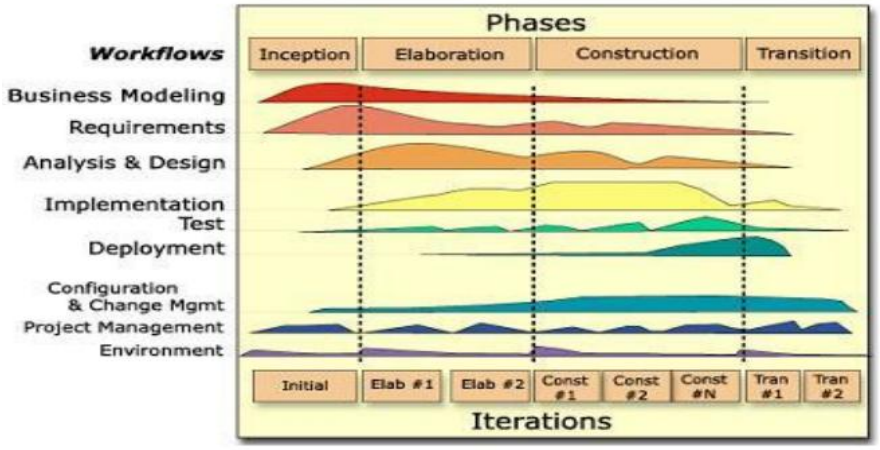
Figura 1: ciclo de vida de la Metodología RUP



Tomado de: (Mendoza M. A., 2004)

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

Figura 2: fases e iteraciones de la Metodología RUP.



Tomado de: (Mendoza M. A., 2004)

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

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

Una particularidad de esta metodología es que, en cada ciclo de iteración, se hace exigente el uso de artefactos, siendo por este motivo, una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del software. (Mendoza M. A., 2004)

1.6.2 Microsoft Solution Framework (MSF)

Esta es una metodología manejable e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.

Figura 3: Metodología MSF.

Tomado de: (Mendoza M. A., 2004)

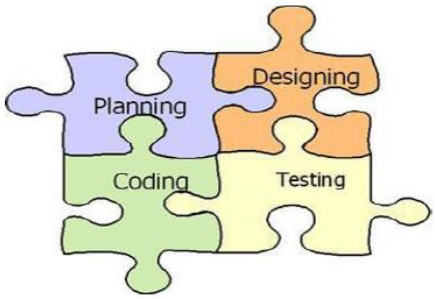
La metodología MSF tiene las características de ser **adaptable**, es parecido a un compás, usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar; **escalable**, puede organizar equipos tan pequeños entre 3 ó 4 personas, así como también, proyectos que requieren 50 personas o más; **flexible**, es utilizada en el ambiente de desarrollo de cualquier cliente y **tecnología agnóstica** porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología. (Mendoza M. A., 2004)

Concretamente MSF se compone de principios, disciplinas y modelos, los principios no son más que promover la comunicación abierta, trabajar para una visión compartida, fortalecer los miembros del equipo, establecer responsabilidades claras y compartidas, focalizarse en agregar valor al negocio e invertir en la calidad. Las disciplinas van a ser la gestión de proyecto, el control de riesgo y el control de cambios. Además se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: modelo de arquitectura del proyecto, modelo de equipo, modelo de proceso, modelo de gestión del riesgo, modelo de diseño de proceso y finalmente el modelo de aplicación. (Mendoza M. A., 2004)

1.6.3 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.

Figura 4: Metodología Programación Extrema.



Tomado de: (Mendoza M. A., 2004)

Esta metodología está basada en **pruebas unitarias**, estas son las pruebas realizadas a los principales procesos, consisten en comprobaciones (manuales o automatizadas) que se realizan para verificar que el código correspondiente a un módulo concreto de un sistema software funciona de acuerdo con los requisitos del sistema; otra característica es la **re fabricación**, que se basa en la reutilización de código, para lo cual se crean patrones o modelos estándares, siendo más flexible al cambio y la **programación en pares**, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Cada miembro lleva a cabo la acción que el otro no está haciendo en ese momento. (Mendoza M. A., 2004)

Lo fundamental en este tipo de metodología es:

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

Después del estudio realizado se puede concluir que XP a pesar de sus ventajas como metodología ágil, no se ajusta al proceso actual, en primer lugar, exige que el cliente forme parte del equipo de desarrollo, requisito que no puede ser satisfecho ya que no se cuenta con un cliente que pueda 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.

1.6.4 Selección de la metodología

A partir del estudio previo de las distintas metodologías se escogió el Proceso Unificado de Desarrollo de Software (RUP), ya que es una metodología de desarrollo de software muy bien organizada, en fases y flujos, tiene como base fundamental del desarrollo: generar los artefactos completamente documentados. Es una de las metodologías más importantes para alcanzar un grado de certificación en el desarrollo del

software; al certificar la calidad de software se pretende fundamentalmente reducir el número de incidencias, gracias a la detección temprana de defectos, reduciendo así el coste total de su desarrollo.

Paralelamente se pretende mejorar la percepción de los usuarios del producto final. Asimismo presenta ventajas por encima de XP, como es el establecimiento temprano de una arquitectura que no se vea fuertemente impactada ante cambios posteriores durante la construcción y el mantenimiento; su enfoque basado en modelos permite un buen entendimiento entre clientes y desarrolladores facilitando la obtención de un producto con altos niveles de calidad; otra ventaja es su enfoque iterativo: la metodología parte de que se trabajará en iteraciones cortas en tiempo y con metas muy claras.

1.7 Lenguajes de modelado

Un sistema tanto del mundo real como en el mundo del software es bastante complejo, por ello es necesario dividir el sistema en partes o fragmentos si se quiere entender y administrar su complejidad. Estas partes se pueden representar como modelos que describan sus aspectos esenciales. Por tanto, un paso útil en la construcción de un sistema de software es el de crear modelos que organicen y comuniquen los detalles más importante de la vida real con que se relacionan y del sistema a construir. (Falgueras, 2003)

Los lenguajes de modelado representan un conjunto estandarizado de símbolos y sus relaciones, permitiendo expresar un sistema informático mediante un esquema teórico que representará posteriormente su diseño.

1.7.1 IDEF0

IDEF0 es una técnica de modelación concebida para representar de manera estructurada y jerárquica las actividades que conforman un sistema o empresa, y los objetos o datos que soportan la interacción de esas actividades.

Un modelo IDEF0 se compone de una serie jerárquica de diagramas que permiten mediante niveles de detalle, describir las funciones especificadas en el nivel superior. En las vistas superiores del modelo la interacción entre las actividades representadas permite visualizar los procesos fundamentales que sustentan la organización. Los elementos gráficos utilizados para la construcción de los diagramas IDEF0 son cuadros y flechas.

Permite representar el proceso cronológicamente. Se describe el flujo orientado al cliente final de ese negocio, cruzando todas las actividades de la organización que dan cumplimiento a la solicitud de producto o servicio que realiza el cliente, representando así la "cadena de valor" de la empresa.

Permite incorporar en el flujo los datos que entran y salen de las actividades, así como las reglas del negocio y los actores, todo en la misma vista. (Hanrahan, 1999)

1.7.2 BPMN

BPMN define un Diagrama de Procesos de Negocio (BPD, del inglés Business Process Diagram), que se basa en una técnica de grafos de flujo para crear modelos gráficos de operaciones de procesos de negocio. Un modelo de procesos de negocio es una red de objetos gráficos que son actividades (trabajo) y controles de flujo que definen su orden de rendimiento.

BPMN se compone de varios conjuntos de elementos que abarcan la representación, tanto de los objetos del flujo y sus conexiones como los instrumentos de ayuda que son las Bandas (Swimlanes) y los artefactos.

Además está diseñado para cubrir muchos tipos de modelados y para permitir la creación de segmentos de proceso así como procesos de negocio con diferentes niveles de fidelidad. (Barriento, 2008)

El objetivo principal de BPMN es proporcionarle a toda las personas involucradas en el negocio una notación factible para entender cómo funcionan los procesos que componen una empresa, desde el analista de negocio que es el encargado de hacer el borrador inicial, pasando por los desarrolladores que son los encargados de implementar todo el sistema que ejecutará dichos procesos, hasta las personas que son los encargados de ejecutar y gestionar los procesos.

BPMN es considerado como un mecanismo simple para crear modelos de procesos de negocio, que lo maneja a través de las cuatro categorías básicas de elementos: objetos de flujo, objetos conectores, rol de proceso, artefactos.

1.7.3 Lenguaje de Modelado Unificado.

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

UML es un método formal de modelado. Esto aporta las siguientes ventajas (Orallo, 2007):

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

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.

1.7.4 Selección del lenguaje de modelado

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 permite 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. Se decidió utilizar BPMN debido a que se necesitó modelar procesos de difícil comprensión que no admiten cambios ni modificaciones y con este lenguaje se modela su flujo de información completo sin omitir ninguno de sus pasos.

Además es de fácil comprensión por los analistas y desarrolladores, también BPMN fue diseñado para permitir a los modeladores y las herramientas de modelado un poco de flexibilidad a la hora de extender la notación básica y a la hora de habilitar un contexto apropiado adicional según una situación específica.

1.8 Herramientas CASE

Las herramientas de desarrollo de software han desempeñado un importante papel en el desarrollo de aplicaciones. Como consecuencia del avance tecnológico éstas han experimentado también continuos cambios.

Estas herramientas favorecen el apoyo al desarrollo de software, proporcionando un conjunto de programas de asistencia a los analistas para la Ingeniería de Software durante todo el ciclo de vida del desarrollo del sistema.

1.8.1 Rational Rose Enterprise Edition

Rational Rose es la herramienta CASE desarrollada por los creadores de UML, que cubre todo el ciclo de vida de un proyecto, desde la fase de inicio, formalización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases. Permite establecer una trazabilidad real entre el modelo (análisis y diseño) y el código ejecutable.

Dentro de sus características principales se puede encontrar un avanzado modelado de UML para trabajar en diseños de bases de datos, con capacidad de representar la integración de los datos y los requisitos de aplicación a través de diseños lógicos y físicos, posee además una fuerte capacidad para integrarse con cualquier sistema de control de versiones. (G. Booch, 2004)

Es importante resaltar que a pesar de esto Rational Rose presenta una necesidad de alta capacidad de procesamiento y se necesita tener habilidad y conocimiento de esta herramienta para trabajar con ella.

1.8.2 Enterprise Architect

Enterprise Architect (EA) es una herramienta comprensible de diseño y análisis UML, cubriendo el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento.

EA es una herramienta multi-usuario, basada en Windows, diseñada para ayudar a construir software robusto y fácil de mantener. Ofrece salida de documentación flexible y de alta calidad.

Enterprise Architect provee trazabilidad completa desde el análisis de requerimientos hasta los artefactos de análisis y diseño, a través de la implementación y el despliegue. Combinados con la ubicación de recursos y tareas incorporados, los equipos de Administradores de Proyectos y Calidad están equipados con la información que ellos necesitan para ayudarles a entregar proyectos en tiempo.

Las bases de Enterprise Architect están construidas sobre la especificación de UML 2.0 y usa Perfiles UML para extender el dominio de modelado, mientras que la Validación del Modelo asegura integridad. Combina Procesos de Negocio, Información y Flujos de trabajo en un modelo usando las extensiones gratuitas para BPMN y el perfil Eriksson-Penker (Sparxsystems, 2008)

1.8.3 Visual Paradigm.

Visual Paradigm es una herramienta multiplataforma de modelado visual UML y una herramienta CASE muy fácil de utilizar. Tributa una excelente interoperabilidad con otras herramientas CASE ya que tiene conexión con Rational Rose en sus archivos de proyecto (.MDL / .CAT) los cuales pueden ser importados a Visual Paradigm UML a través de esta importante característica.

Visual Paradigm para UML es apoyado por un conjunto de idiomas tanto en la generación del código como en la Ingeniería Inversa por mencionar algunos ejemplos, los cuales tienen la capacidad de soporte, podríamos hablar de Java, C + +, CORBA IDL, PHP, XML Schema, Ada y Python. Permite la captura de requisitos, análisis, diseño e implementación, también proporciona características tales como generación del código, ingeniería inversa y generación de informes.

Permite dibujar todos los tipos de diagramas de clases. Apoya los estándares más recientes de las notaciones de UML. Incorpora el soporte para trabajo en equipo, permitiendo que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros. (Sierra, 2011)

Visual Paradigm presenta características esenciales como son:

- Modelado colaborativo con CVS (Concurrent Versions System) y Subversión.

- Ingeniería inversa, código a modelo, código a diagrama.
- Generación de código, modelo a código, diagrama a código.
- Editor de Detalles de Casos de Uso, entorno todo en uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- Diagramas de flujo de datos.
- Generación de bases de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Ingeniería inversa de bases de datos, desde Sistemas Gestores de Bases de Datos existentes a diagramas de Entidad-Relación.
- Generador de informes para generación de documentación.
- Distribución automática de diagramas. Reorganización de las figuras y conectores de los diagramas UML.
- Modelo para realizar prototipos de interfaz.

Destacar que es una herramienta con licencia comercial y la UCI puede hacer uso de ella.

1.8.4 Selección de la herramienta CASE

Después de investigadas estas herramientas se decide utilizar Visual Paradigm ya que genera toda la documentación de lo que se realiza cumpliendo con los estándares establecidos, es una de las pocas herramientas CASE que soporta el análisis textual, siendo esta una técnica útil y práctica para la captura de los requisitos del sistema. Otra de las características por la que se decide usar Visual Paradigm es su disponibilidad en múltiples plataformas, ya que no obliga al usuario a desarrollar solo en el sistema operativo Windows, sino que está disponible en sistemas operativos como Windows, Linux, Unix.

1.9 Ingeniería de requisitos

La actividad de análisis es una de las principales en el proceso de desarrollo de software y consiste en investigar el problema que se origina, interiorizarlo y describirlo, para luego obtener las necesidades del cliente o requisitos. En el momento de realizar el análisis es necesario una adecuada comunicación entre los clientes y los analistas, debido que partiendo del intercambio que se realiza se obtienen los requisitos a implementar en el sistema.

Capítulo 1: Fundamentación Teórica

Para lograr comprender que es la Ingeniera de Requisitos, es necesario definir que es un requisito y cuál es su función. A continuación se mencionan algunos de los conceptos existentes actualmente (Dávila, 2001):

- Es una condición o una capacidad que debe estar presente en un sistema o componentes de sistema para satisfacer un contrato, estándar, especificación u otro documento formal.

- Es una descripción de los servicios que brindará el sistema y las restricciones que este incluye. Representa la necesidad que tienen los clientes de un sistema que les resuelve determinado problema.

- Los requisitos de software se clasifican en dos grupos: los funcionales y los no funcionales. Los requisitos funcionales determinan una capacidad o condición que debe tener el sistema y los no funcionales son una propiedad o cualidad que el producto debe tener, o sea, características que hacen el producto atractivo, rápido, confiable etc.
- La Ingeniería de Requisitos es la aplicación de los procedimientos, técnicas, lenguajes y herramientas para obtener como resultado el análisis, la documentación correspondiente y el seguimiento de las necesidades del usuario. Es un proceso que incluye el descubrimiento de lo que el usuario quiere, el refinamiento de estos, su modelado y su especificación. También se realiza un análisis detallado de las soluciones que se proponen.
- La Ingeniería de Requerimientos tiene un papel muy importante en la construcción de un software, pero a su vez es uno de los procesos más complejos, puesto que según Frederick P. Brooks “La parte más difícil de construir un sistema es precisamente saber qué construir. Ninguna otra parte del trabajo conceptual es tan difícil como establecer los requerimientos técnicos detallado.

Ninguna otra parte del trabajo afecta tanto el sistema si es hecha mal. Ninguna es tan difícil de corregir más adelante. Entonces, la tarea más importante que el ingeniero de software hace para el cliente es la extracción iterativa y el refinamiento de los requerimientos del producto”. (Brooks, 1995)

Su objetivo principal es lograr que se generen las especificaciones de las necesidades de los usuarios o clientes de forma correcta, con claridad, sin ambigüedades, en forma consistente y compacta.

1.9.1 Etapas de la Ingeniería de Requisitos

La ingeniería de software tiene en cuenta un grupo de etapas que se encuentran lógicamente definidas y estructuradas. Estas etapas cuentan con un grupo de técnicas que facilitan su aplicación en el momento de saber qué es lo que desea el cliente, o qué es exactamente lo que necesita. A continuación se describen cada una de ellas (Pressman, 2005):

1.9.1.1 Elicitación de requisitos

La Elicitación o Identificación de requisitos es la primera de las etapas de la Ingeniería de requisitos. Consiste en extraer, de cualquier fuente de información disponible, los problemas que debe resolver el sistema. En esta etapa interactúan los clientes con los desarrolladores del sistema para obtener exactamente lo que debe cumplir este, identificándose los requisitos funcionales y los no funcionales.

1.9.1.2 Análisis de requisitos

Una vez que se tienen identificados los requisitos del sistema, se analizan detalladamente para ver si presentan alguna ambigüedad, si son consistentes y se verifica su completitud. En esta etapa es donde se agrupan y se clasifican de acuerdo a las funcionalidades que responden.

1.9.1.3 Especificación de requisitos

Es la etapa de la Ingeniería de requisitos donde se describen las funciones y características del sistema. Puede ser mediante un documento o cualquier modelo gráfico que describa como es el comportamiento del software.

1.9.1.4 Validación de requisitos

La validación de requisitos es la etapa que se encarga de verificar la calidad de los requisitos identificados. Vela porque los requisitos identificados hayan sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones y que los errores detectados en las anteriores etapas hayan sido corregidos.

1.9.1.5 Gestión de requisitos

La gestión de requisitos se encarga de darle seguimiento a los requisitos a lo largo de las etapas anteriores. El objetivo es realiza un conjunto de actividades que permiten identificar, seguir y controlar los cambios sufridos en cualquier momento del ciclo de vida del sistema.

1.9.2 Técnicas de Ingeniería de Requisitos

Dentro de las principales técnicas para la identificación de requisitos está la ***entrevista***, que es una técnica muy usada y consta de tres fases: preparación, realización y análisis. La preparación como su nombre lo indica es la fase donde se hace un estudio previo del dominio del problema se seleccionan los entrevistados y se debe tener claro cuál es el objetivo de la entrevista.

Se debe realizar con un lenguaje natural y se debe ir dando conclusiones parciales cada vez que así lo requiera la situación, además de dar las conclusiones finales al terminar la entrevista. Por último, el análisis es donde se estudia y se detalla todo lo hablado durante la entrevista.

Otra de las técnicas utilizadas en la elicitación de requisitos es ***Desarrollo conjunto de aplicaciones (JAD- Joint Application Development)***. Esta técnica es considerada como una alternativa a las entrevistas individuales. Son reuniones entre un grupo de personas pero de muy corta duración. Va acompañada de imágenes, gráficas o cualquier elemento visual que facilite el entendimiento del mensaje que se quiere transmitir.

La tormenta de ideas es otra importante y muy utilizada técnica de ***elicitación de requisitos***. Consiste en hacer reuniones de un grupo de participante donde se generan muchas ideas sobre un tema específico. Debe estar guiado por un jefe que sea quien de inicio al debate. El objetivo es ver los diferentes puntos de vista de los participantes.

También la técnica de ***Casos de Uso*** es una técnica muy usada en la elicitación, donde se agrupan los requisitos por funcionalidades para luego describir las acciones del usuario y la respuesta del sistema, o sea, la secuencia de acciones que sigue para lograr un objetivo.

Los ***prototipos*** es una técnica usada en la validación de requisitos que se utilizada para mostrarle al cliente o usuario una propuesta de interfaz de las funcionalidades. Se emplea para lograr una mejor comprensión de los requisitos y consiste en diseñar una propuesta de interfaz de un requisito determinado que debe ir incluido en el producto final, aunque esta puede presentar modificaciones.

1.10 Diseño de sistemas

El diseño de sistemas de software se realiza para transformar los requisitos identificados en un diseño detallado del sistema en cuestión. Tiene como finalidad evolucionar hacia una arquitectura sólida para el sistema y generar un diseño que sea entendible por los desarrolladores y que ayude y facilite la implementación. Cuando se realiza el diseño de un sistema se tienen en cuenta un grupo de elementos que amplían y describen el comportamiento del sistema, facilitando el entendimiento de cómo se deben ejecutar las funcionalidades.

A continuación se describen algunos de ellos, como los patrones de diseño, diagramas de secuencia y diagramas de clases del diseño. (Larman, 1999)

1.10.1 Patrones de diseño

Los patrones de diseño son propuestas de soluciones a problemas frecuentes surgidos en el proceso de desarrollo de software, estos brindan una solución a problemas similares que está probada y bien descrita, o sea, que están sujetos a situaciones muy parecidas. Un patrón de diseño es un ente descriptible de acuerdo a su nombre, el problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) así como sus consecuencias (costos y beneficios).

De forma general, un patrón de diseño se puede definir como (Larman, 1999):

- Una forma más práctica de describir determinados elementos de la organización de un programa.
- Una solución estándar para un problema común de programación.
- Una técnica para flexibilizar el código haciéndolo satisfacer ciertos criterios.
- Un proyecto o estructura de implementación que logra una finalidad determinada.

Actualmente existen muchos patrones de diseño de software, estos a su vez se agrupan en grandes grupos, entre los populares tenemos:

Patrones GoF (Gant of Four):

Patrones Creacionales: inicialización y configuración de objetos (Larman, 1999).

- **Abstract Factory:** proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar su clase concreta.
- **Builder:** permite a un objeto construir un objeto complejo especificando sólo su tipo y contenido.
- **Factory Method:** define una interfaz para crear un objeto dejando a las subclases decidir el tipo específico al que pertenecen.
- **Prototype:** permite a un objeto crear objetos personalizados sin conocer su clase exacta a los detalles de cómo crearlos.
- **Singleton:** garantiza que solamente se crea una instancia de la clase y provee un punto de acceso global a él.

Patrones Estructurales: separan la interfaz de la implementación. Se ocupan de cómo las clases y objetos se agrupan, para formar estructuras más grandes (Larman, 1999).

- **Adapter:** convierte la interfaz que ofrece una clase en otra esperada por los clientes.
- **Bridge:** desacopla una abstracción de su implementación y les permite variar independientemente.
- **Composite:** permite gestionar objetos complejos e individuales de forma uniforme.
- **Decorator:** extiende la funcionalidad de un objeto dinámicamente de tal modo que es transparente a sus clientes.
- **Facade:** simplifica los accesos a un conjunto de objetos relacionados proporcionando un objeto de comunicación.

- **Flyweight:** usa la compartición para dar soporte a un gran número de objetos de grano fino de forma eficiente.
- **Proxy:** proporciona un objeto con el que se controla el acceso a otro objeto.

Patrones de Comportamiento: más que describir objetos o clases, describen la comunicación entre ellos. (Larman, 1999)

- **Chain of Responsability:** evita el acoplamiento entre quien envía una petición y el receptor de la misma.
- **Command:** encapsula una petición de un comando como un objeto.
- **Interpreter:** dado un lenguaje define una representación para su gramática y permite interpretar sus sentencias.
- **Iterator:** acceso secuencial a los elementos de una colección.
- **Mediator:** define una comunicación simplificada entre clases.
- **Memento:** captura y restaura un estado interno de un objeto.
- **Observer:** una forma de notificar cambios a diferentes clases dependientes.
- **State:** modifica el comportamiento de un objeto cuando su estado interno cambia.
- **Strategy:** define una familia de algoritmos, encapsula cada uno y los hace intercambiables.
- **Template Method:** define un esqueleto de algoritmo y delega partes concretas de un algoritmo a las subclases.
- **Visitor:** representa una operación que será realizada sobre los elementos de una estructura de objetos, permitiendo definir nuevas operaciones sin cambiar las clases de los elementos sobre los que opera.

Patrones GRASP (General Responsibility Assignment Software Patterns):

En diseño orientado a objetos, GRASP son patrones generales de software para asignación de responsabilidades. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de software.

- **Experto en Información:** El GRASP de experto en información es el principio básico de asignación de responsabilidades. Nos indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la

información necesaria para crearlo. De este modo obtendremos un diseño con mayor cohesión y así la información se mantiene encapsulada (disminución del acoplamiento)

Problema: ¿Cuál es el principio general para asignar responsabilidades a los objetos?

Solución: Asignar una responsabilidad al experto en información.

Beneficios: Se mantiene el encapsulamiento, los objetos utilizan su propia información para llevar a cabo sus tareas. Se distribuye el comportamiento entre las clases que contienen la información requerida. Son más fáciles de entender y mantener.

- **Creador:** El patrón creador nos ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

La nueva instancia deberá ser creada por la clase que:

- ✓ Tiene la información necesaria para realizar la creación del objeto, o
- ✓ Usa directamente las instancias creadas del objeto, o
- ✓ Almacena o maneja varias instancias de la clase
- ✓ Contiene o agrega la clase.

Una de las consecuencias de usar este patrón es la visibilidad entre la clase creada y la clase creador. Una ventaja es el bajo acoplamiento, lo cual supone facilidad de mantenimiento y reutilización La creación de instancias es una de las actividades más comunes en un sistema orientado a objetos. En consecuencia es útil contar con un principio general para la asignación de las responsabilidades de creación. Si se asignan bien el diseño puede soportar un bajo acoplamiento, mayor claridad, encapsulación y reutilización.

- **Controlador:** El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

  Este patrón sugiere que la lógica de negocios debe estar separada de la capa de presentación, esto para aumentar la reutilización de código y a la vez tener un mayor control.

  Se recomienda dividir los eventos del sistema en el mayor número de controladores para poder aumentar la cohesión y disminuir el acoplamiento.

- **Alta cohesión y Bajo acoplamiento:** Los conceptos de cohesión y acoplamiento están íntimamente relacionados. Un mayor grado de cohesión implica uno menor de acoplamiento.

Maximizar el nivel de cohesión intramodular en todo el sistema resulta en una minimización del acoplamiento intermodular.

Alta cohesión:

Nos dice que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase.

Bajo acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

- **Polimorfismo :** Siempre que se tenga que llevar a cabo una responsabilidad que dependa del tipo, se tiene que hacer uso del polimorfismo, cuando las alternativas o comportamientos relacionados varían según el tipo (clase), asigne la responsabilidad para el comportamiento- utilizando operaciones polimórficas- a los tipos para los que varía el comportamiento. Asigna el mismo nombre a servicios en diferentes objetos.

- **Fabricación Pura:** La fabricación pura se da en las clases que no representan un ente u objeto real del dominio del problema, sino que se ha creado intencionadamente para disminuir el acoplamiento, aumentar la cohesión y/o potenciar la reutilización del código. Es la solución cuando el diseñador se encuentre con una clase poco cohesiva y no tenga otra clase en la que implementar algunos métodos. Es decir que es una clase "inventada" o que no existe en el problema como tal, pero que añadiéndola se logra mejorar estructuralmente el sistema. Como contraindicación deberemos mencionar que al abusar de este patrón suelen aparecer clases función o algoritmo (que tienen un solo método).

- **Indirección:** El patrón de indirección nos aporta mejorar el bajo acoplamiento entre dos clases asignando la responsabilidad de la mediación entre ellos a un tercer elemento (clase) intermedio.

  Problema: ¿Dónde asignar responsabilidades para evitar/reducir el acoplamiento directo entre elementos y mejorar la reutilización? Solución: Asigne la responsabilidad a un objeto que medie entre los elementos.

- **Variaciones Protegidas:** Es el principio fundamental de protegerse del cambio, de tal forma que todo lo que preveamos en un análisis previo que es susceptible de modificaciones, lo envolvamos en una interfaz, utilizando el polimorfismo para crear varias implementaciones y posibilitar implementaciones futuras, de manera que quede lo menos ligado posible a nuestro sistema, de forma que cuando se produzca la variación, nos repercuta lo mínimo.

Se recomienda de forma general usar patrones de diseño solo en aquellos casos en los que se tiene un total dominio del problema, utilizando además un algoritmo eficiente desde el comienzo y a lo largo de todo el ciclo de desarrollo. Los patrones de diseño pueden aumentar o disminuir la capacidad de entendimiento de un diseño o de una implementación, pueden hacer lo mismo con la cantidad de código, todo depende del uso adecuado que se les dé. Una vez dominados los patrones de diseño, éstos serán de gran ayuda a la hora de incorporar calidad al producto que se desarrolla, así como valor agregado al mismo.

Los patrones de diseño deben estar al alcance de todos los desarrolladores de software, difundir la experiencia adquirida por todos los miembros del equipo suele ser una ventaja, pues se generaliza el conocimiento y se agiliza de sobremanera el desarrollo.

1.10.2 Modelo de Diseño

El Modelo de Diseño es un artefacto que se genera en el flujo de trabajo Análisis y Diseño, específicamente en la etapa de Diseño. Este constituye una abstracción de la implementación del sistema y tiene como objetivo documentar el diseño sobre el que se apoya el flujo de trabajo de implementación. Está compuesto por clases de diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos.

Los paquetes de diseño son los encargados de agrupar los elementos de modelo de diseño relacionados con el objetivo de darle organización. No ofrecen una interfaz formal, aunque puede darse el caso de que expongan algunos de sus contenidos que ofrecen comportamiento. Los paquetes de diseño deben utilizarse fundamentalmente como herramienta organizativa del modelo, para agrupar elementos relacionados. Los subsistemas, a diferencia de los paquetes de diseño, encapsulan comportamiento, proporcionando interfaces formales y no incluyen el contenido interno. Las clases del diseño son una representación de un grupo de objetos que tienen la misma responsabilidades, relaciones, operaciones y atributos. (Falgueras, 2003)

Conclusiones parciales

En este capítulo se hizo un estudio detallado de los principales aspectos relacionados con el Análisis y Diseño de un sistema. Se compararon acepciones de los diferentes autores sobre los conceptos más importantes para el desarrollo de la investigación. Se hizo una comparación de las principales herramientas de gestión de proyecto usadas en el Mundo, Cuba y específicamente en la UCI. También se realizó un estudio de las tendencias mundiales sobre metodologías de desarrollo, lenguajes de modelado y las herramientas CASE que dan soporte a estos.

Como resultado del estudio y comprensión de los elementos anteriormente mencionados se llegó a las siguientes conclusiones:

- Los simuladores estudiados no presentan elementos que aporten beneficios desde el punto de vista técnico y funcional a la investigación aunque desde el punto de vista del diseño si se tomaron

algunas ideas.

- Se seleccionó la metodología RUP por ser una herramienta rica en documentación y facilitarle la comprensión a las demás personas que van a dar continuación a las otras etapas de la investigación.
- Se utilizará el lenguaje de modelado de negocio BPMN por ser un lenguaje muy descriptivo y fácil de modelar, que permite la clara comprensión del proceso en cuestión.
- Se utilizará el lenguaje de modelado de sistema UML por ser un leguaje muy utilizado en el mundo, que facilita la comprensión de lo que se desea modelar, además por ser el lenguaje que propone la metodología RUP.
- Se empleará como herramienta CASE Visual Paradigm, por ser una herramienta muy buena para el modelado, que da soporte a varios lenguajes de modelado como BPMN y UML. Además permite realizar todo tipo de diagramas como prototipos de interfaz de usuario y modelado de bases de datos además de que es la herramienta CASE utilizada en la universidad para el modelado.
- El análisis es una importante etapa dentro del proceso de desarrollo de software puesto que te permite entender, describir y determinar las características del sistema que se desea realizar.
- El diseño también es muy importante porque es la base de la implementación, que permite modelar a un alto nivel de detalles cómo será la interacción entre clases, componentes y objetos del sistema.

Capítulo 2: Descripción de la solución

2.1 Introducción

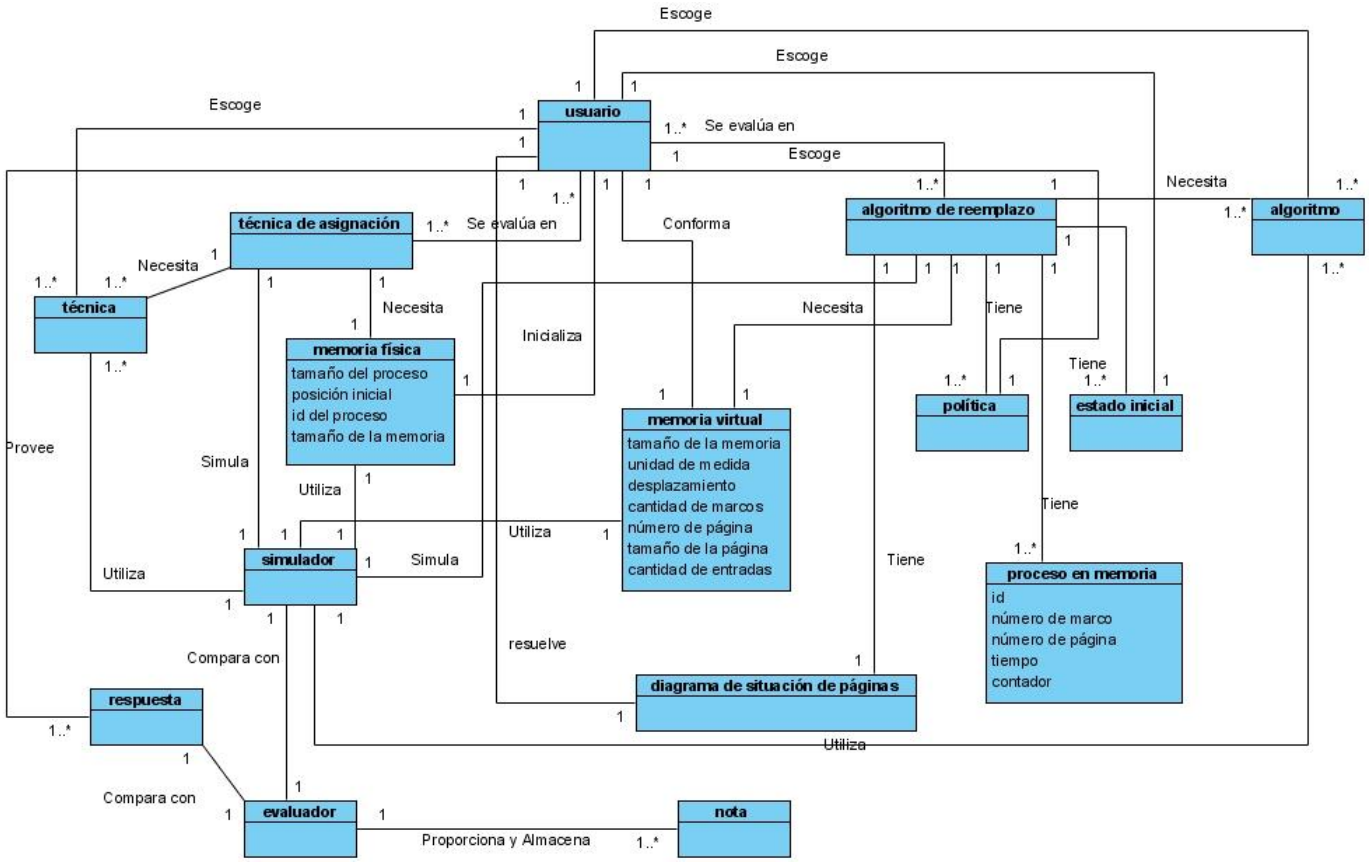
En el presente capítulo se realiza una modelación de los procesos a simular, esto se hace a través del diagrama de flujo de procesos. También se especifican los requisitos funcionales y los no funcionales por los cuales se regirá el sistema propuesto, se identificarán mediante el diagrama de casos de uso del sistema las relaciones entre sus actores y los casos de uso así como la descripción de los mismos, además en el análisis definido por la metodología de desarrollo escogida se tuvieron en cuenta los requisitos funcionales durante la confección de los diagramas de clases del análisis y los diagramas de secuencia correspondientes a cada caso de uso. El diseño consolidará el análisis propuesto con los diagramas de clases y la aplicación de patrones de diseño como una manera más práctica de describir ciertos aspectos; teniendo en cuenta las cualidades o propiedades que el sistema debe tener.

2.2 Mapa conceptual

Los mapas conceptuales son artefactos para la organización y representación del conocimiento. Su objetivo es representar relaciones entre conceptos en forma de proposiciones. Los conceptos están incluidos en cajas o círculos, mientras que las relaciones entre ellos se explicitan mediante líneas que unen sus cajas respectivas. Las líneas, a su vez, tienen palabras asociadas que describen cuál es la naturaleza de la relación que liga los conceptos. Se estructuran en forma jerárquica en la que los conceptos más generales están en la raíz del árbol y a medida que vamos descendiendo por el mismo nos vamos encontrando con conceptos más específicos. (Dürsteler, 2004)

El mapa conceptual mostrado a continuación representa la relación didáctica de todo el contenido referente al tema administración de memoria, lo que constituye la base teórica en la cual se desarrolla la investigación.

Capítulo 2: Descripción de la Solución



2.3 Especificación de los requisitos del software

Un proyecto no puede ser exitoso sin una descripción detallada, correcta y exhaustiva de los requerimientos, estos definen lo que debe hacer un sistema y la forma en que debe hacerlo.

2.3.1 Requisitos funcionales

Los Requisitos funcionales especifican acciones que el sistema debe ser capaz de realizar, sin tomar en consideración ningún tipo de restricción física, especifican el comportamiento de entrada y salida del sistema y surgen de la razón fundamental de la existencia del producto. (Estrada Y. H., 2009)

Los Requisitos Funcionales para este sistema se presentan a continuación:

Tabla 1: Requisitos funcionales

Referencia	Requisitos Funcionales

RF1	Inicializar memoria física
RF2	Registrar estado de la memoria física
RF3	Registrar proceso en memoria
RF4	Mostrar grado de multiprogramación
RF5	Asignar nuevo proceso a memoria
RF6	Administrar huecos libres
RF7	Obtener lista de huecos
RF8	Visualizar cola de entrada de procesos y páginas
RF9	Mostrar la página a partir de una dirección lógica
RF10	Configurar memoria virtual
RF11	Configurar memoria física
RF12	Reemplazar página en memoria física
RF13	Visualizar errores
RF14	Verificar resultados contra simulación
RF15	Emitir nota

2.3.2 Requisitos no funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener, forman una parte significativa de la especificación, las propiedades no funcionales; como cuán usable, seguro, conveniente y agradable, distinguen a un producto bien aceptado de uno con poca aceptación. (Estrada Y. H., 2009)

A continuación se muestran los requerimientos 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.

- **RNF3:** Debe contar con un vínculo a la ayuda en la ventana principal del trabajo.

Usabilidad:

- **RNF4:** El sistema puede ser usado por cualquier estudiante que posea conocimientos básicos sobre el funcionamiento interno de un Sistema Operativo.

Rendimiento:

- **RNF5:** La respuesta a solicitudes más complejas de los usuarios del sistema no debe exceder 9 segundos.
- **RNF6**: La aplicación deberá estar disponible las 24 horas del día.

Portabilidad:

- **RNF7:** Debe poder ejecutarse tanto en Sistemas Operativos desde Windows XP Profesional Service Pack 1 o Superior como en Sistemas Operativos Linux.

Hardware:

- **RNF8:** Tarjeta de memoria RAM de 128 MB o superior.
- **RNF9:** Procesador Pentium II o superior a 250 MHz como mínimo.
- **RNF10:** Computadora cliente de 40 Gb de disco duro o superior.

Restricciones de diseño

Uso de estándares de codificación

- **RNF11** Debe garantizarse el uso de estándares de codificación, para una mejor comprensión de los códigos ejecutados en caso de una segunda versión del sistema.

Requisitos para la documentación de usuarios en línea y ayuda del sistema

Ayuda del sistema

- **RNF12** El sistema contará con una ayuda que constituirá una guía de apoyo en el momento de hacer uso de la aplicación. Esta describe todas las funcionalidades del sistema.

2.4 Actores del sistema

EL actor del sistema es quien interactúa o hace uso del sistema.

Los actores del sistema pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado, son parte del sistema, y pueden intercambiar información con él o ser recipientes pasivos de información (G. Booch, 2004).

En este caso los actores del sistema se presentan a continuación.

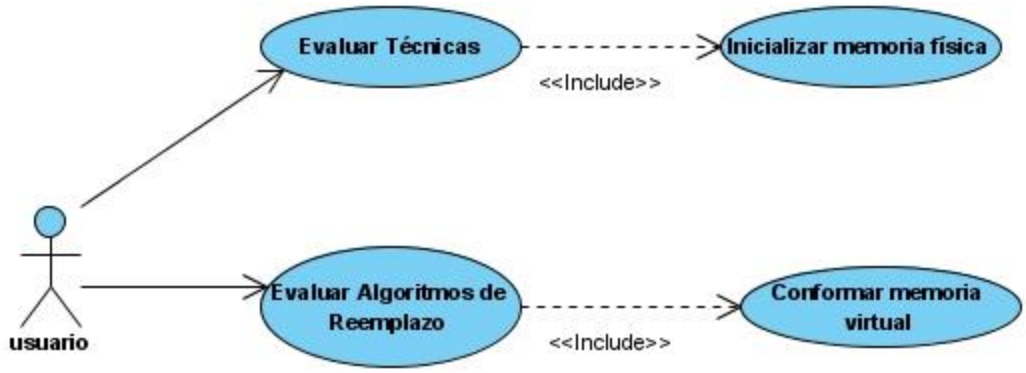
Tabla 2: Actores del sistema

Actor del Sistema	Descripción
Usuario	El usuario es el encargado de realizar cualquier operación dentro del sistema, como inicializar la memoria física, evaluarse en cualquier técnica de asignación o algoritmos de reemplazos. Este puede ser el estudiante, el profesor o cualquier usuario que tenga conocimientos del tema y requiera de su uso.

2.4.1 Diagrama de casos de uso del sistema

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 Y. H., 2009)

El presente diagrama se explica a continuación.



El estudiante puede realizar la simulación de todas las técnicas de asignación y así poder evaluarse, pero antes tiene que inicializar la memoria física para poder ubicar los procesos nuevos en cada espacio vacío según el funcionamiento de cada técnica. También puede simular los algoritmos de reemplazo, aunque a petición del cliente solo se van a implementar los dos más importantes y con los cuales los estudiantes trabajan mayormente; antes de comenzar la simulación se debe haber conformado la memoria virtual para poder llevar a cabo el reemplazo.

2.4.2 Descripción de los casos de uso del sistema

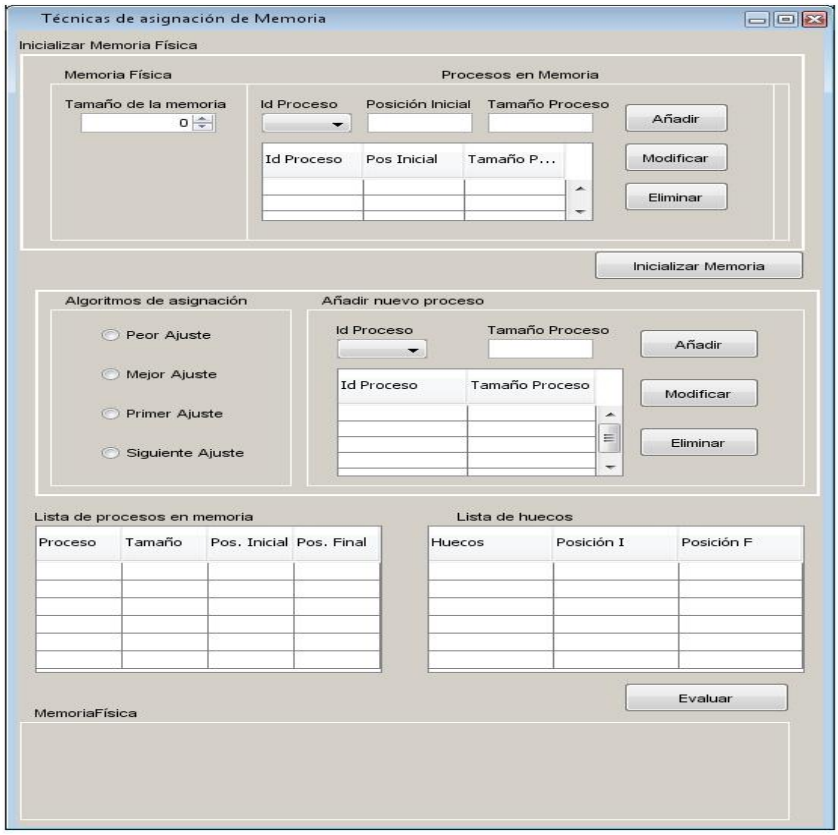
Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema. (Estrada, 2009). A continuación se muestra la descripción del caso de uso Evaluar Técnicas, la demás descripciones se encuentran el documento modelo de caso de uso del sistema.

Descripción del Caso de uso: "Evaluar Técnicas"

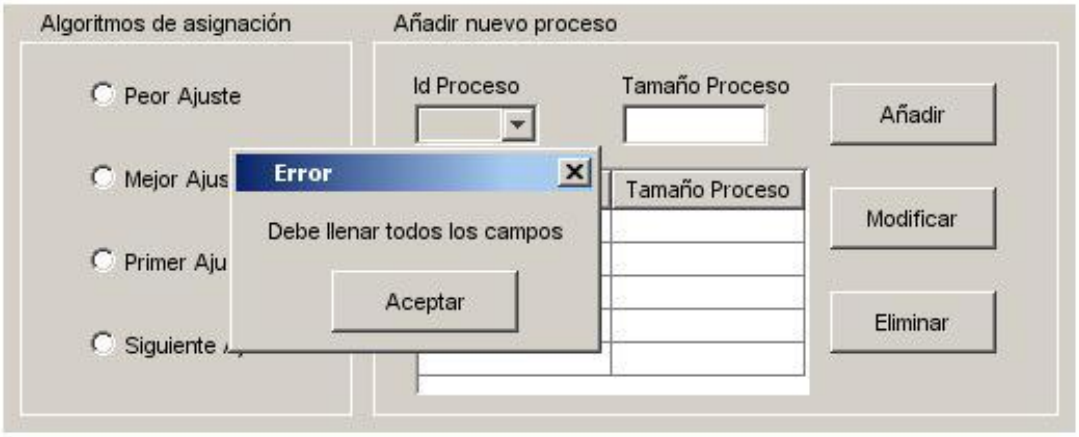
Caso de uso:	Evaluar Técnicas
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario selecciona uno de los algoritmos de asignación para ejecutar alguna de sus páginas en memoria física, y termina cuando el resultado de la simulación es almacenado y se compara con la respuesta introducida por el usuario mostrándole la nota.
Precondiciones:	
Pos condición:	Almacenar el estado final de la memoria, Id del proceso, tamaño, posición inicial, posición final, y la posición inicial y final de la lista de huecos. Además se almacena la nota.
Casos de usos asociados:	**CU Inicializar Memoria Física (incluido)**
Referencia:	RF1, RF2, RF3, RF4, RF5, RF6, RF7, RF13, RF14, RF15.
Prioridad:	Crítico

Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción Técnicas de asignación.	2. Se invoca el **CU incluido Inicializar Memoria Física.** 3. El sistema muestra la interfaz con los algoritmos de asignación: • Peor Ajuste • Mejor Ajuste • Primer Ajuste • Siguiente Ajuste
4. El usuario selecciona el algoritmo deseado.	5. El sistema muestra campos para introducir nuevos procesos: • Id Proceso • Tamaño Proceso Y las opciones Añadir, Modificar y Eliminar.
6. El usuario introduce los datos especificados y presiona la opción Añadir.	7. El sistema verifica que no hayan campos vacíos.
	8. El sistema comprueba que los datos entrados sean correctos.
	9. El sistema realiza la simulación en dependencia del algoritmo seleccionado y almacena los resultados de la simulación.
	10. El sistema muestra la opción para introducir la respuesta a evaluar: • Id del proceso • Tamaño proceso • Posición inicial

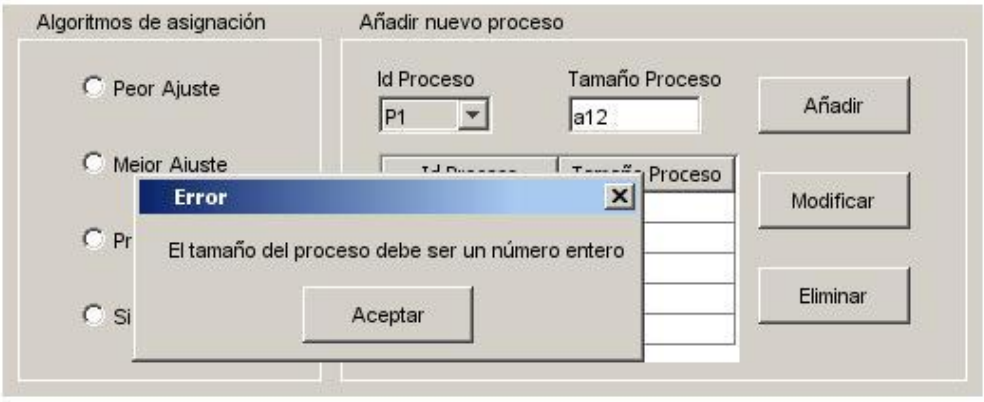
	• Posición final • Id de los huecos • Posición inicial • Posición final
11. Introduce Id del proceso, tamaño, posición inicial y final así como el Id de los huecos con su posición inicial y final, y selecciona la opción evaluar.	12. El sistema verifica que los campos no estén vacíos.
	13. El sistema muestra los errores.
	14. El sistema muestra la nota.
	15. El sistema muestra la interfaz con el dibujo de los procesos y los huecos en memoria.
Prototipo de interfaz de usuario	



Flujo Alterno del paso 7	
Acción de actor	**Acción del sistema**
	7a.1 En caso de haber algún campo vacío el sistema muestra el siguiente mensaje de error: “Debe llenar todos los campos”.
Prototipo de interfaz de usuario	



Flujo Alterno del paso 9	
Acción de actor	**Acción del sistema**
	9a.1 En caso de que los datos entrados sean incorrectos el sistema muestra el siguiente mensaje de error: “El tamaño del proceso debe ser un número entero”.



2.5 Análisis del sistema

Las actividades del análisis son desarrolladas con el objetivo de facilitar la entrada al diseño, por lo que son un paso inicial y una primera aproximación conceptual para aumentar el nivel de especificidad en aras de garantizar el cubrimiento de los requisitos funcionales y obtener una visión de qué hace el sistema.

Las clases del análisis van a representar abstracciones de conceptos, en las cuales deben incluirse atributos y operaciones a un nivel alto. Existen tres estereotipos de clases del análisis estandarizado en UML y se utilizan para ayudar a los desarrolladores a distinguir el ámbito de las diferentes clases. Estas clases son (Almaguer, 2009):

Interfaz: modelan la interacción entre el sistema y sus actores.

Control: coordinan la realización de un caso de uso, controlando las actividades de los objetos que implementan su funcionalidad.

Entidad: modelan información que posee larga vida y que por lo general es persistente. (Almaguer, 2009)

2.5.1 Diagrama de clases del análisis

El diagrama de clases del análisis (DCA) es un artefacto en el que se representan los conceptos en un dominio del problema. En este tipo de diagrama se representan las clases y sus relaciones. Ellos representan una vista estática del sistema. (Almaguer, 2009)

A continuación se muestran los diagramas de clases del análisis correspondiente a los casos de uso: Inicializar Memoria Física, Conformar Memoria Virtual, Evaluar Técnicas y Evaluar Algoritmos de Reemplazo.

DIAGRAMA DE CLASES DE ANÁLISIS DEL CU "INICIALIZAR MEMORIA FISICA"

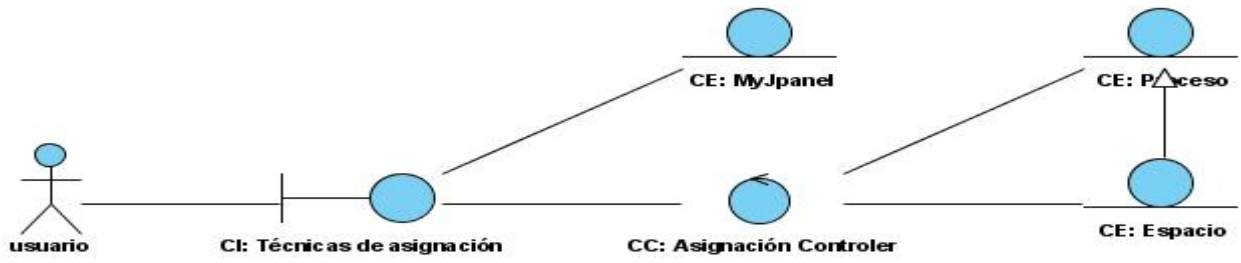


Figura 5: Diagrama de clases del análisis del CU "Inicializar Memoria Física"

DIAGRAMA DE CLASES DE ANÁLISIS DEL CU "EVALUAR TÉCNICAS"

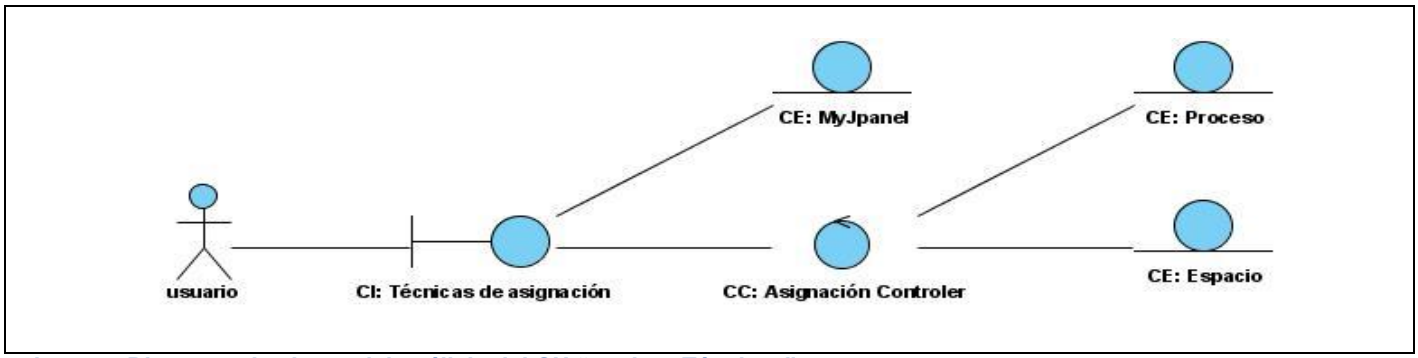


Figura 6: Diagrama de clases del análisis del CU "Evaluar Técnicas"

DIAGRAMA DE CLASES DE ANÁLISIS DEL CU "CONFORMAR MEMORIA VIRTUAL"

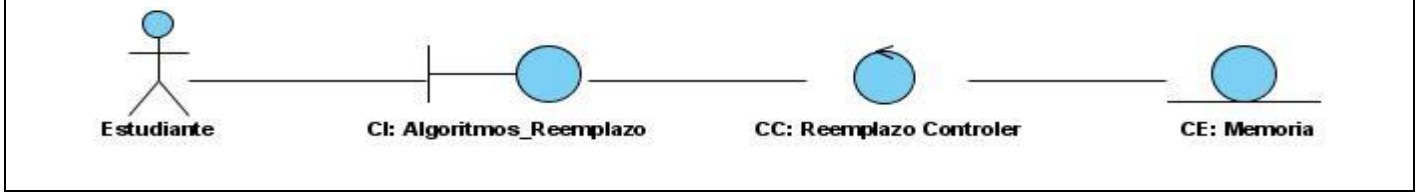


Figura 7: Diagrama de clases del análisis del CU "Conformar Memoria Virtual"

DIAGRAMA DE CLASES DE ANÁLISIS DEL CU "EVALUAR ALGORITMOS DE REEMPLAZO"

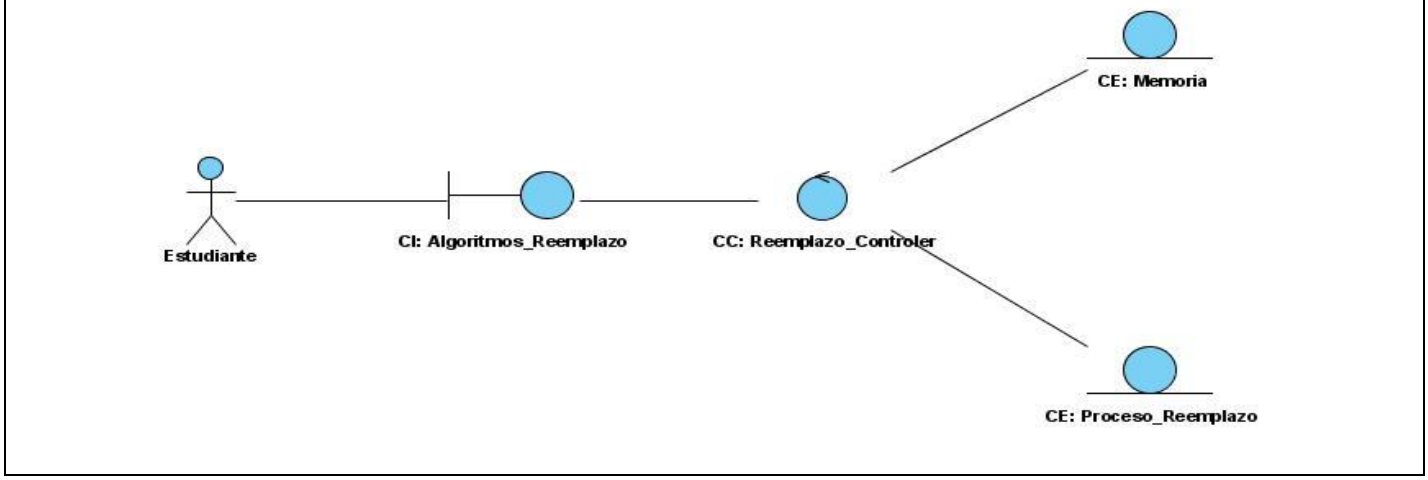


Figura 8: Diagrama de clases del análisis del CU "Evaluar Algoritmos de Reemplazo"

2.5.2 Diagramas de interacción

Un diagrama de interacción muestra gráficamente cómo los objetos interactúan a través de mensajes para realizar las tareas. No son sólo importantes para modelar los aspectos dinámicos de un sistema, sino también para construir sistemas ejecutables por medio de ingeniería directa e inversa. Existen dos tipos de diagramas de interacción: secuencia y colaboración.

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación, contiene detalles de implementación de los escenarios, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes intercambiados entre los objetos. Se realizaron los diagramas de secuencia correspondientes a los casos de uso críticos Inicializar memoria física, Conformar memoria virtual, Evaluar técnicas y Evaluar Algoritmos de reemplazo. En estos diagramas se podrá apreciar y detallar mejor el análisis (Oca, 2009).

Los demás diagramas correspondientes a cada caso de uso se encuentran en el artefacto modelo de diseño.

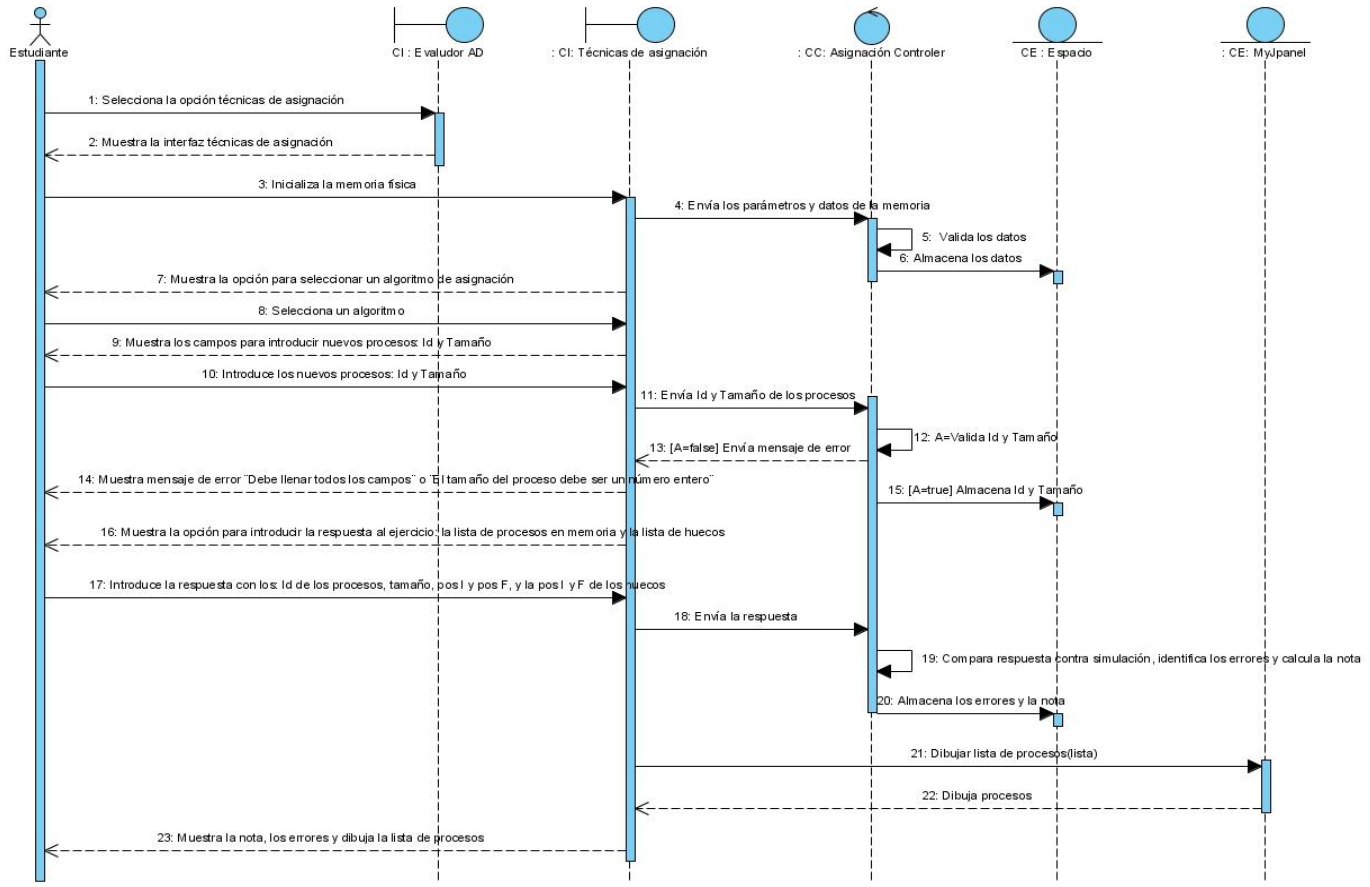


Figura 9: Diagrama de secuencia del CU "Evaluar Técnicas"

2.6 Diseño

El diseño es un refinamiento del análisis que tiene en cuenta los requisitos no funcionales, o sea cómo cumple el sistema sus objetivos. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades. En el diseño se modela el sistema y se define una arquitectura que soporte todos los requisitos. Específicamente se puede definir como propósitos del diseño:

- Adquirir una comprensión de los aspectos relacionados con los requisitos no funcionales y restricciones relacionadas con los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución y concurrencia y tecnologías de interfaz de usuario.
- Crear una entrada apropiada y un punto de partida para actividades de implementación, capturando los requisitos o subsistemas individuales, interfaces y clases.
- Descomponer los trabajos de implementación en partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo.

2.6.1 Diagrama de clases del diseño

Un diagrama de clases de diseño muestra la especificación para las clases de software de una aplicación.

El diagrama de clases referente al diseño del sistema se muestra a continuación.

DIAGRAMA DE CLASES DEL DISEÑO

Capítulo 2: Descripción de la Solución

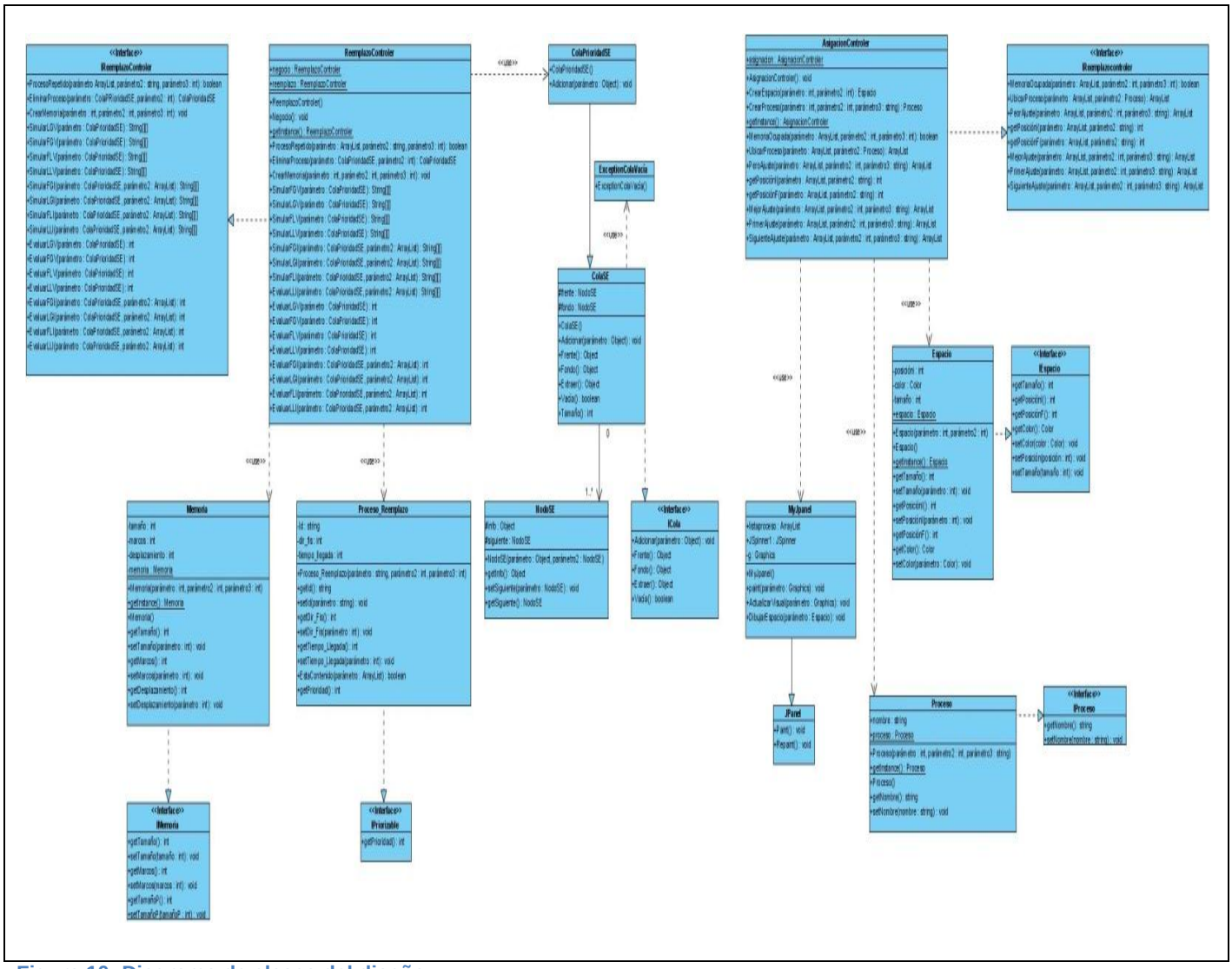


Figura 10: Diagrama de clases del diseño

2.6.2 Patrones de diseño utilizados

Durante el diseño del sistema se proponen utilizar patrones generales de software para asignar responsabilidades (GRASP), que constituyen principios básicos a tener en cuenta cuando se quiere construir eficazmente un software orientado a objetos, estos son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. (Almaguer, 2009) Entre los más evidentes en el diseño propuesto se encuentran los patrones: Bajo Acoplamiento, Alta Cohesión y Singleton.

Conclusiones parciales

Capítulo 2: Descripción de la Solución

Analizando todo lo antes expuesto en el capítulo, se puede concluir que se obtuvieron los principales artefactos referentes al análisis y diseño para la eficiente implementación de la aplicación. La realización del análisis de forma detallada facilitó la entrada al diseño. La realización de los casos de uso aportará la información necesaria para el proceso de implementación, además el uso de patrones de diseño permitió obtener un diseño flexible.

Capítulo 3: Validación de los resultados

3.1 Introducción

Un factor fundamental para efectuar la evaluación de los productos y procesos de software en los diferentes dominios de aplicación lo constituyen las métricas, las cuales abarcan un gran escenario en cuanto a medidas de software computacional se refieren. Las métricas suelen ser aplicadas a muchas organizaciones, procesos y productos que estén relacionados y afecten a la estimación de costo. En este capítulo se presenta la etapa de validación y prueba, que se realiza con el objetivo de asegurar completitud, correctitud, calidad, entre otros factores de gran importancia.

3.2 Validación de los resultados por métricas

Las métricas son el término que describen muchos y muy variados casos de medición. Siendo una medida estadística (no cuantitativa como en otras disciplinas) que se aplica a todos los aspectos de calidad de software, los cuales deben ser medidos desde diferentes puntos de vista como el análisis, construcción, funcional, documentación, métodos, proceso, usuario, entre otros.

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. También existen métricas que permiten validar el modelo de casos de uso. (Piña & Pulido, 2008)

Seguidamente se hará referencia a algunas de las métricas que fueron aplicadas a los requisitos de software y casos de uso.

3.2.1 Métrica de la calidad de la especificación

La validación de requisitos se realiza aplicando la métrica de la calidad de la especificación, examina las especificaciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad. Para esto es necesario conocer el total de los requisitos **Rt** dado por:

Rt = Rf + Rnf

26 = 15 + 11

Dónde:

Rt: Total de requisitos.

Rf: Cantidad de requisitos funcionales.

Rnf: Cantidad de requisitos no funcionales.

Para determinar la especificidad (ausencia de ambigüedad) de los requisitos. Se explica una métrica basada en la consistencia de la interpretación de los revisores para cada requisito:

Se calcula **Q1** para determinar la especificidad de los requisitos.

Q1 = Rui / Rt

0.96 = 25/26

Alta (0.90 <= **E** <= 1).
Media (0.80 <= **E** < 0.90).
Baja (0.7 <= **E** < 0.80).

Dónde:

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

Q1: Ausencia de ambigüedad.

Cuanto más cerca de 1 esté el valor de **Q1**, menor será la ambigüedad de la especificación.

El valor de **Q1** = 0.96, esto demuestra que los requisitos se encuentran con un alto nivel de especificidad.

La ***estabilidad*** fue determinada con la fórmula ——— donde R_m son los requisitos modificados, que es equivalente al número de requisitos. Se considera como valor óptimo para esta métrica el valor más cerca de 1.

Se sustituyen los valores:

E = (Rt – Rm) / Rt

E = (26 – 0) / 26

E = 1.00

El resultado obtenido muestra que los requisitos son estables, ya que se obtuvo el valor máximo de estabilidad basado en el siguiente rango:

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

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

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

3.2.2 Métricas para validar los casos de usos del sistema

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

tendrá asociada una o más métricas, que establecen una medida cuantitativa del grado en que los factores presentan una pobre calidad. (Larman, 1999)

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

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

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

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

Para clasificar los resultados obtenidos se establecieron las siguientes reglas:

Alto (90% <= E <= 100%).

Medio (80%<= E < 90%).

Bajo (70%<= E < 80%).

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

Primera Revisión:

Atributo	Factor	Métrica asociada	Valor
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar/ modificar o consultar información?	Métrica 1: Número de roles relevantes omitidos. Umbral < 90%	Número de roles relevantes omitidos: 0 Se presenta un 100%.
	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 < 90%	Número de casos de uso que no tiene descripción resumida: 0. Se presenta un 100%.
	Factor 3. ¿Están definidos todos los	Métrica 3: Número de requisitos omitidos por	Número de requisitos omitidos por caso de uso:

	requisitos que justifican la funcionalidad del caso de uso?	caso de uso. Umbral: 90% Métrica 4: Número de casos de uso que tienen requisitos omitidos. Umbral: 80%	1 Se presenta un 75%. Número de casos de uso que tienen requisitos omitidos: 1 Se presenta un 75%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en (crítico, secundario, auxiliar, opcional)?.	Métrica 5: Número de casos que no han sido clasificados. Umbral: 80%	Número de casos que no han sido clasificados: 0 Se presenta un 100%.
		Umbral: 86%	Se presenta un 90%
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: 80%	Número de casos de uso que tienen un nombre incorrecto:1 Se presenta un 75%.
	Factor 6. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos?	Métrica 7: Grado de adecuación de la descripción del flujo de eventos para un caso de uso. Umbral: 95%	La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción: 2. Se presenta un 50%.
	Factor 7. ¿La descripción del flujo de eventos se inicia con la descripción de una acción externa 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 < 90%	Número de casos de uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema: 1 Se presenta un 75%.
	Factor 8. ¿Existe una adecuada separación	Métrica 9: Número de casos de uso complejos	Número de casos de uso complejos que no tienen

	entre el flujo básico de eventos y los flujos alternos y/o flujos subordinados?	que no tienen separación del flujo básico y de flujos alternos. Umbral < 80%	separación del flujo básico y de flujos alternos: 0 Se presenta un 100%.
		Umbral: 55%	Se presenta un: 75%
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 < 95%	Número de casos de uso en que los requisitos representados no son comprensibles por el usuario: 1 Se presenta un 75%.
	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 < 90%	Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema: 0 Se presenta un 100%.
	Factor 11. ¿Se ajusta la representación del diagrama del caso de uso de acuerdo a lo normado en la metodología?	Métrica 12: Grado en que se ajusta el diagrama del caso de uso a la metodología. Umbral < 90%	Grado en que se ajusta el diagrama del caso de uso a la metodología: 2. Se presenta un 50%.
	Factor 12. ¿Las interacciones definidas introducen mejoras al proceso actual?	Métrica 13: Número de casos de uso que deben ser modificados para mejorar el proceso actual. Umbral < 95%	Número de casos de uso que deben ser modificados para mejorar el proceso actual: 1 Se presenta un 75%.
		Umbral: 70%	Se presenta un: 75%
Complejidad	Factor 13. ¿Los elementos dentro del diagrama están adecuadamente ubicados	Métrica 14: Número de elementos del diagrama que requieren	Número de elementos del diagrama que requieren reubicación: 1

	de manera que facilitan su interpretación?	reubicación. Umbral < 70%	Se presenta un 75%.
		Umbral: 70%	Se presenta un 75%.

Métricas para validar diagrama de caso de uso del sistema

Segunda Revisión:

Atributo	Factor	Métrica asociada	Valor
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes de usuario encargados de generar/ modificar o consultar información?	Métrica 1: Número de roles relevantes omitidos.	Número de roles relevantes omitidos: 0 Se presenta un 100%.
	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.	Número de casos de uso que no tiene descripción resumida: 0. Se presenta un 100%.
	Factor 3. ¿Están definidos todos los requisitos que justifican la funcionalidad del caso de uso?	Métrica 3: Número de requisitos omitidos por caso de uso. Métrica 4: Número de casos de uso que tienen requisitos omitidos.	Número de requisitos omitidos por caso de uso: 0 Se presenta un 100%. Número de casos de uso que tienen requisitos omitidos: 0 Se presenta un 100%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en (crítico, secundario, auxiliar, opcional)?.	Métrica 5: Número de casos que no han sido clasificados.	Número de casos que no han sido clasificados: 0 Se presenta un 100%.
			Se presenta un 100%
Consistencia	Factor 5. ¿El nombre	Métrica 6: Número de	Número de casos de uso

	dado a los casos de uso es una expresión verbal que describe alguna funcionalidad relevante en el contexto del usuario?	casos de uso que tienen un nombre incorrecto.	que tienen un nombre incorrecto: 0 Se presenta un 100%.
	Factor 6. ¿Está adecuadamente redactado (en el lenguaje del usuario) el flujo de eventos?	Métrica 7: Grado de adecuación de la descripción del flujo de eventos para un caso de uso	La descripción se define en el lenguaje del usuario. Se define el responsable de cada acción: 0. Se presenta un 100%.
	Factor 7. ¿La descripción del flujo de eventos se inicia con la descripción de una acción externa 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.	Número de casos de uso cuya descripción incluida no inicia con una acción externa o con una condición monitoreada por el sistema: 0 Se presenta un 100%.
	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.	Número de casos de uso complejos que no tienen separación del flujo básico y de flujos alternos: 0 Se presenta un 100%.
			Se presenta un: 100%
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.	Número de casos de uso en que los requisitos representados no son comprensibles por el usuario: 0 Se presenta un 100%.
	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	Número de casos de uso que deben ser modificados para adecuarlos a la funcionalidad del sistema:

		sistema.	0 Se presenta un 100%.
	Factor 11. ¿Se ajusta la representación del diagrama del caso de uso de acuerdo a lo normado en la metodología?	Métrica 12: Grado en que se ajusta el diagrama del caso de uso a la metodología.	Grado en que se ajusta el diagrama del caso de uso a la metodología: 0. Se presenta un 100 %.
	Factor 12. ¿Las interacciones definidas introducen mejoras al proceso actual?	Métrica 13: Número de casos de uso que deben ser modificados para mejorar el proceso actual.	Número de casos de uso que deben ser modificados para mejorar el proceso actual: 0 Se presenta un 100%.
			Se presenta un: 100%
Complejidad	Factor 13. ¿Los elementos dentro del diagrama están adecuadamente ubicados de manera que facilitan su interpretación?	Métrica 14: Número de elementos del diagrama que requieren reubicación.	Número de elementos del diagrama que requieren reubicación: 0 Se presenta un 100%.
			Se presenta un 100%.

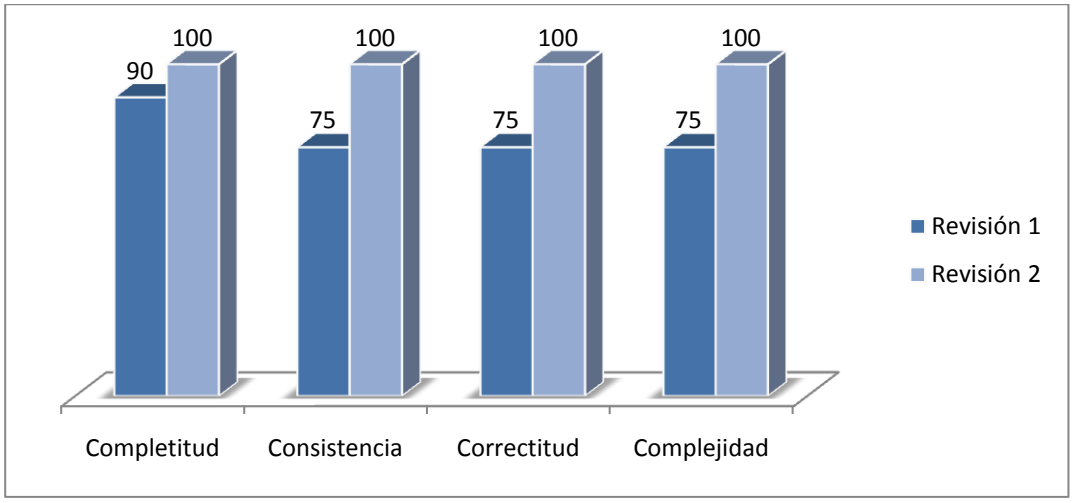


Figura 11: Gráfica de factores de métrica para validar los casos de uso

Resultados: Como se puede apreciar en el gráfico anterior los resultados obtenidos en la segunda revisión fueron relevantes y con la aplicación de estas métricas se pudieron evaluar los factores completitud, consistencia, correctitud y complejidad del diagrama de casos de uso del sistema lo que permitió demostrar que los requisitos identificados son implementados en al menos un caso de uso, abarcando de esta forma todas las necesidades expresadas por el cliente, que lo casos de uso fueron descritos detalladamente mostrando el flujo alterno a parte del flujo básico lo que da una mayor legibilidad de los mismos, evidenciándose además que estos son iniciados por la interacción del usuario con el sistema o por un evento interno dentro del mismo.

Todo esto expuesto anteriormente permitió comprobar que el artefacto caso de uso del sistema se encuentra con la calidad requerida para entrar al flujo de trabajo análisis y diseño donde se comenzaría la realización del software.

3.3 Métricas para validar el diseño

- **Tamaño operacional de clase (TOC)**
  Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad: (Colectivo de Autores, 2001)
  - **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 que 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.

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

Criterios de evaluación para la métrica TOC.

Capítulo 3: Validación de los resultados

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

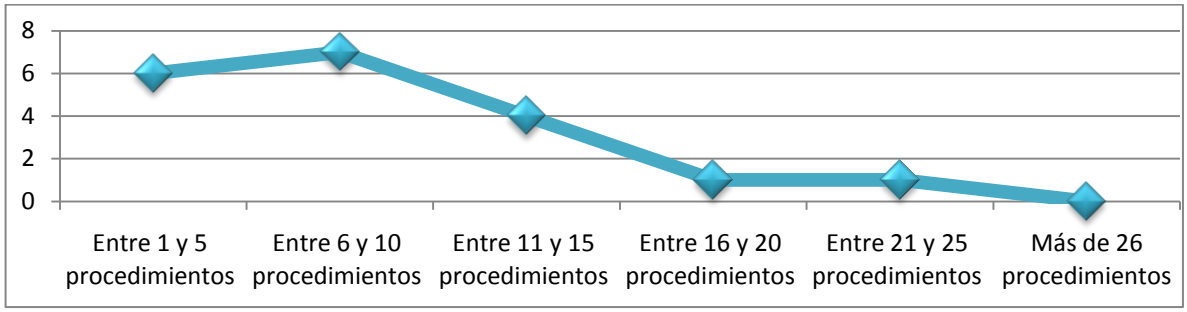


Figura 12: Representación de los resultados por intervalos definidos.

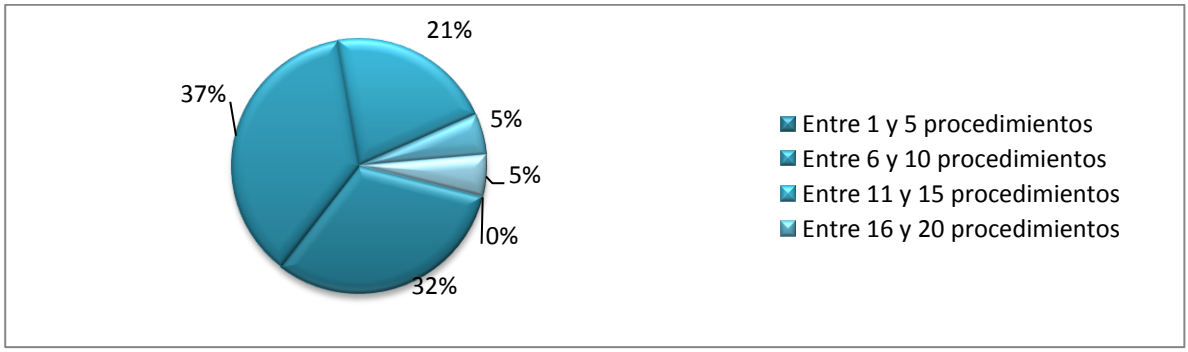


Figura 13: Representación en % de los resultados obtenidos para la métrica TOC.

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

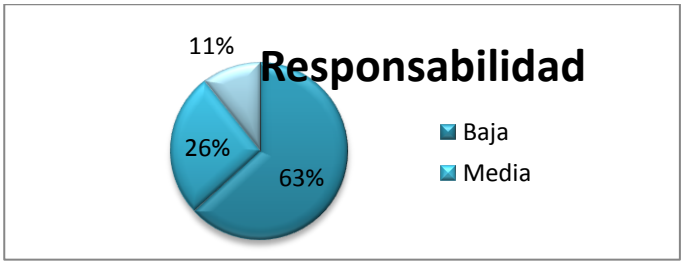


Figura 14: Resultados de la métrica TOC en el atributo Responsabilidad.

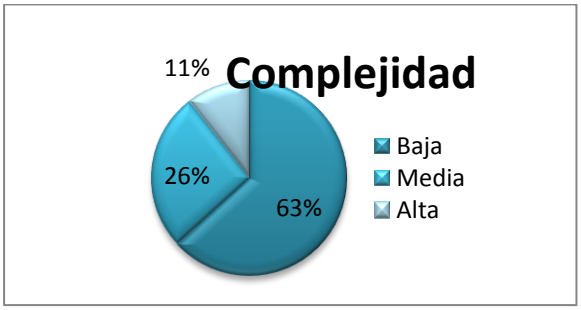


Figura 15: Resultados de la métrica TOC en el atributo Complejidad de Implementación.

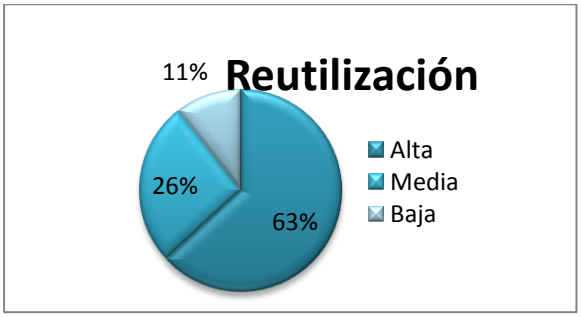


Figura 16: Resultados de la métrica TOC en el atributo Reutilización.

Al analizar los resultados obtenidos luego de aplicar el instrumento de medición de la métrica TOC, se puede concluir que el diseño propuesto está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (37%) posee menos cantidad de operaciones que la media registrada en las mediciones. Los atributos de calidad se encuentran en un nivel medio satisfactorio en el 37% de las clases; de manera que se puede observar cómo se fomenta la Reutilización (elemento clave en el proceso de desarrollo de software) y cómo están reducidas en menor grado la Responsabilidad y la Complejidad de implementación.

- **Relaciones entre clases (RC)**

Con la presente métrica se evalúan los siguientes atributos de calidad:

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

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

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

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

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

Criterios de evaluación de la métrica RC:

Atributo	Categoría	Criterio
Acoplamiento.	Ninguno.	0
	Bajo.	1
	Medio.	2
	Alto.	>2

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

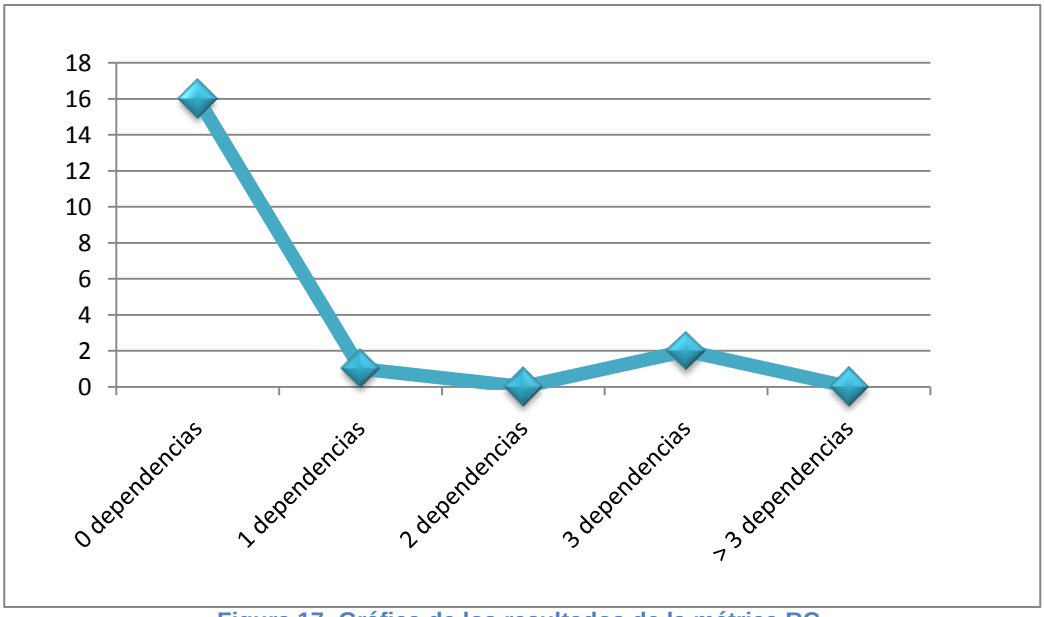


Figura 17: Gráfica de los resultados de la métrica RC.

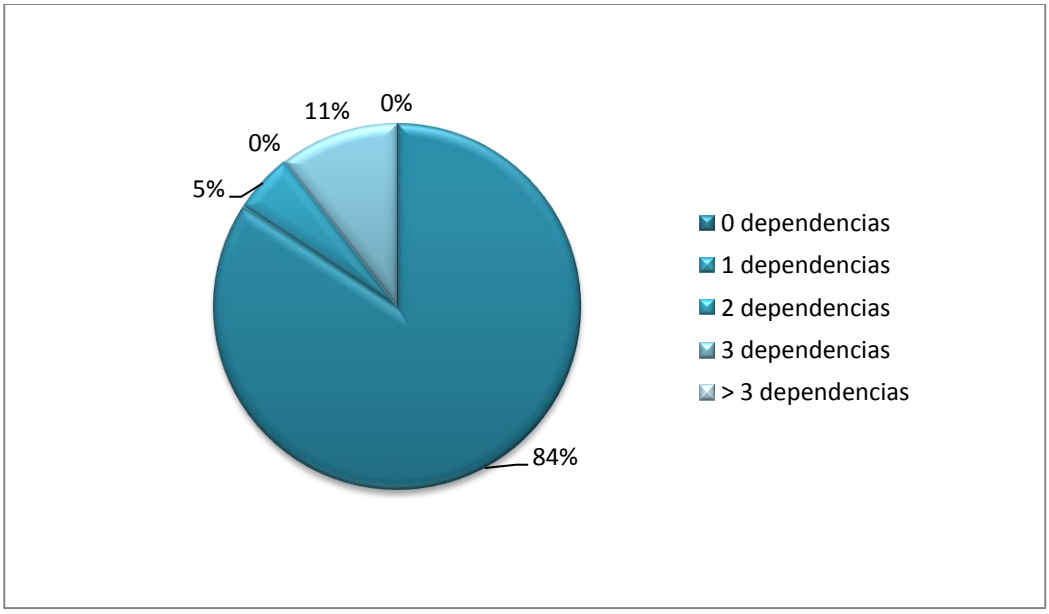


Figura 18: Representación en % de los resultados por intervalos definidos.

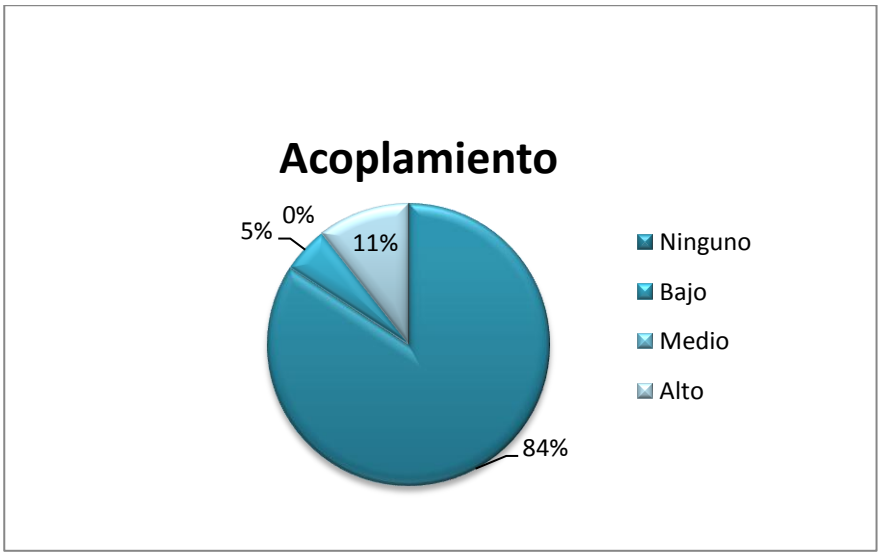


Figura 19: Resultados de la métrica RC en el atributo Acoplamiento.

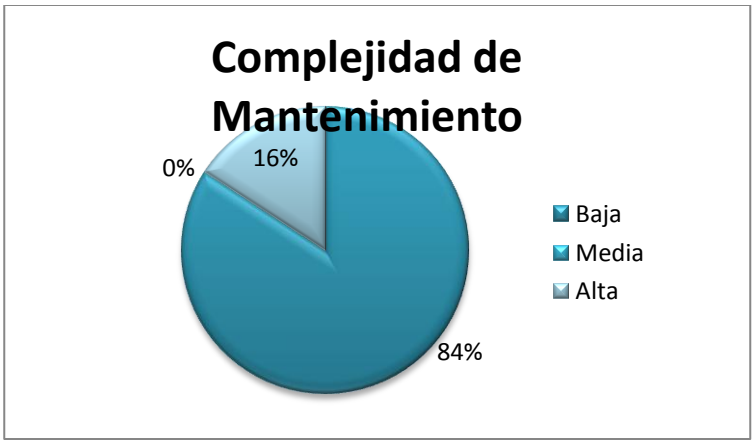


Figura 20: Resultados de la métrica RC en el atributo Complejidad de mantenimiento.

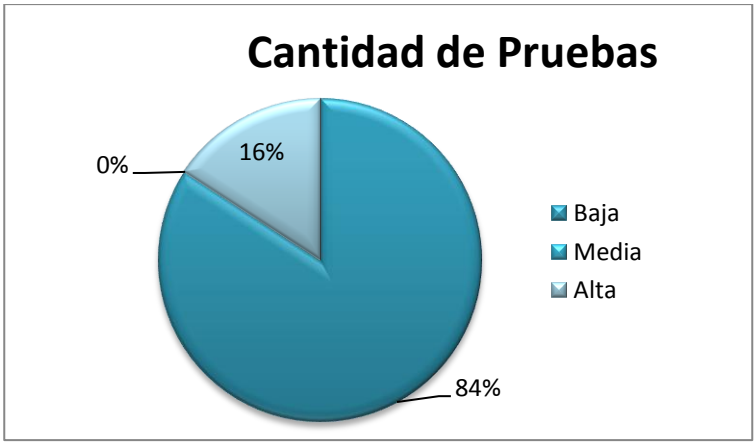


Figura 21: Resultados de la métrica RC en el atributo Cantidad de pruebas.

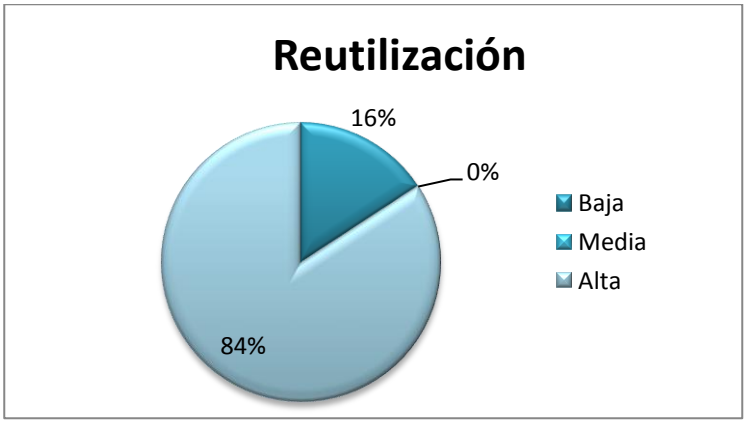


Figura 22: Resultados de la métrica RC en el atributo Reutilización.

Capítulo 3: Validación de los resultados

Al examinar los resultados obtenidos, luego de emplear el instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto está entre los límites aceptables de calidad, teniendo en cuenta que la mayoría de las clases (84%) no poseen dependencias respecto a otras. Los atributos de calidad se encuentran en un nivel satisfactorio; el grado de dependencia o acoplamiento, la complejidad de mantenimiento, la cantidad de pruebas y la reutilización se comportan favorablemente para un 84% en todos los casos.

Conclusiones parciales

En el presente capítulo se abordaron los temas que hacen alusión a la aplicación de métricas a los modelos obtenidos con el fin de validar la calidad de los mismos.

Se pudo constatar teniendo en cuenta los resultados que, tanto los requisitos obtenidos como parte del proceso de captura de requisitos como el diagrama de casos de uso del sistema, se encuentran dentro de los parámetros de calidad establecidos por las características de las métricas aplicadas.

A partir de la aplicación de las métricas para el diseño se puede afirmar que los artefactos obtenidos cuentan con un nivel óptimo de calidad lo que permitirá pasar a la fase de implementación.

Conclusiones

Mediante el estudio de las funcionalidades de los diferentes laboratorios virtuales, simuladores y evaluadores en otros sistemas en el mundo, se contribuyó a elevar el nivel de comprensión de los desarrolladores y a determinar los modelos de análisis y diseño necesarios para dar paso al futuro sistema a construir.

- Se investigó sobre las posibles herramientas y metodologías para el desarrollo de la aplicación seleccionándose como metodología de desarrollo a RUP unido al lenguaje de modelado UML y BPMN.
- Se realizó el análisis y diseño del evaluador para la Asignatura Sistemas Operativos, obteniendo la documentación necesaria para garantizar el futuro desarrollo del sistema.
- Se obtuvieron los modelos de análisis y diseño que cumplen con los requisitos establecidos.
- La aplicación de métricas propuestas por varios autores y realizadas posibilitó validar los resultados obtenidos.
- Con la realización del análisis y diseño para la posterior implementación de esta aplicación, en la Universidad de las Ciencias Informáticas aumentan los materiales interactivos que sirven de apoyo para impartir el tema de Administración de Memoria en la asignatura Sistemas Operativos.
- Se cumplió el objetivo general trazado para este Trabajo de Diploma: realizar el Análisis y Diseño de un Evaluador de los contenidos del tema Administración de Memoria para el Laboratorio Virtual de la Asignatura Sistemas Operativos que permita su posterior implementación.

Recomendaciones

Tomando en cuenta los resultados obtenidos:

- Se recomienda terminar el análisis y diseño del evaluador para los restantes algoritmos de reemplazo.
- Continuar con los flujos de trabajo propuestos por RUP para lograr el desarrollo del evaluador de los contenidos del tema Administración de Memoria.

Bibliografía

Almaguer, A. B. (2009). *Sistema automatizado para el proceso de caracterización de los estudiantes.* Cuidad de la Habana.

Barriento, M. S. (2 de noviembre de 2008). *BPMN desventajas.* Recuperado el 4 de noviembre de 2010, de http://www.aprendergratis.com/stag/bpmn-desventajas.html.

Barriento, M. S. (2 de Noviembre de 2008). *BPMN desventajas.* Recuperado el 9 de Febrero de 2010

Beck, K. (1999). *Planning Extreme Programming.* ISBN 0-201-71091-9.

Borges, R. A. (2009). *Migración de la Capa de Acceso a Datos.* Cuidad de la Habana.

Brooks, F. (1995). *The Mythical Man-Month: Essays on Software Engineering.*

Cárdenas, M. E. (2009). *Software de Simulación aplicado a entornos de e-learning.*

Caro, P. S. (2007). Recuperado el Marzo de 2011, de http://www.dcc.uchile.cl/~psalinas/uml/introduccion.html

Colectivo de Autores. (2001). *Metricas para el diseño de software*. Recuperado el 21 de Enero de 2011, de www.infor.uva.es/~manso/calidad/metricasoo-2011.pdf

Dávila, N. (2001). *Ingeniería de Requerimientos una guía para extraer, analizar, especificar y validar los requerimientos de un proyecto.*

Dürsteler, J. (2004). Mapas Conceptuales y sus aplicaciones. Inf@Vis.

Estrada, Y. H. (junio de 2009). Análisis del Módulo Proceso Confiscatorio de Bienes del proyecto Sistema Gestión Fiscal. Ciudad Habana.

Falgueras, B. C. (2003). *Ingenieria de Software.* Barcelona: Editoria UOC, edicion Aragon 182 08011.

G. Booch, J. R. (2000). *El proceso unificado de desarrollo de software.* Addison Wesley.

Bibliografía

G. Booch, J. R. (2004). *Clikear.com.* Recuperado el 12 de 11 de 2010, de http://www.clikear.com/manuales/uml/bibliografia.aspx

Hanrahan, R. P. (1999). *The IDEF0 Process Modeling Methodology.* Recuperado el 2010 de Noviembre de 5, de Software Technology Support Center.

Herías, F. A. (27 de noviembre de 2003). *Propuesta de Portal de la Red de Laboratorios Virtuales y Remotos de CEA.* Recuperado el 20 de octubre de 2010, de http://www.disc.ua.es/docenweb/Docs/PropuestaDePortal.pdf.

Herreros, L. R., & R., J. (2005). *Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física.* Recuperado el 19 de octubre de 2010, de http://www.formatex.org/micte2005/286.pdf.

ISW, C. D. (2010). *Ingeniería de Software II.* Recuperado el 27 de Febrero de 2010

Izquierdo, L. (2008). Modelado de sistemas complejos mediante simulación basada en agentes y mediante dinámica de sistemas. EMPIRIA. Revista de Metodología de Ciencias Sociales.

Jacobson, I. B. (2000). *El proceso unificado de desarrollo de software.* Addison Wesley.

Larman, C. (1999). *UML y Patrones: Introducción al análisis y diseño orientado a objetos.* Prentice Hall.

Mendoza, M. A. (7 de junio de 2004). *Metodologías de Desarrollo de Software.* Recuperado el 2 de noviembre de 2010, de http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.htm.

Mosquera, J. D. (2008). *Herramientas CASE.*

Nájera Estrada, J. M., & Méndez, V. H. (2007). *Ventajas y desventajas de usar laboratorios Vituales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración. la opinión del estudiantado en un proyecto de seis años de duración.* Recuperado el 3 de noviembre de 2010, de http://www.latindex.ucr.ac.cr/edu31-1/edu-31-1-05.pdf.

Bibliografía

Oca, E. C. (junio de 2009). Análisis y Diseño de una herramienta interactiva de simulación de procesos: Nodo Virtual de Procesos. Segunda Iteración. . Ciudad Habana, Cuba.

Orallo, E. H. (2007). *El lenguaje Unificado de Modelado (UML).* Recuperado el 25 de septiembre de 2010

Piña, L. R., & Pulido, Y. (2008). *Análisis de los módulos Planificanción de Disco y Administración de Memoria de un Laboratorio Virtual de apoyo a la asignatura de SIstemas Operativos.* Habana, Cuba: Universisdad de Las Ciencias Informáticas.

PMBOK, G. d. (2004). *Guía de los fundamentos de la dirección de proyecto.* Estados Unidos: Project Management Institute, Inc.

Pressman, R. S. (2005). *Ingeniería del software. Un enfoque práactico.* La Habana: Felix Varela.

PRUEBASDESOFTWARE. (2005). *Gestión de Calidad y Pruebas de Software*. Recuperado el 31 de 3 de 2011, de http://www.pruebasdesoftware.com/laspruebasdesoftware.htm

Ruiz Gutiérrez, J. M. (2000). *La Simulación como Instrumento de Aprendizaje.* Recuperado el 5 de Febrero de 2010, de http://mami.uclm.es/jmruiz/materiales/Documentos/simulacion.PDF

Sanchez, M. A. (2004). *Metodologia de desarrollo de software.*

Sierra, D. (2011). *www.slideshare.net*. (www.slideshare.net) Recuperado el 14 de Junio de 2011, de http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml

Sparxsystems. (2008). *www.sparxsystems.com*. Recuperado el 14 de Junio de 2011, de http://www.sparxsystems.com.ar

Stanley, A. (4 de febrero de 2010). *Exploreb Minnesota Echoes.* Recuperado el 14 de diciembre de 2010, de http://www.echoes.com/msf/

Tanenbaum, A. S. (1997). *Sistemas Operativos. Diseño e Implementación.* Prentice Hall.

Vary, J. P. (2000). *Informe de la reunión de expertos.*

Bibliografía

Velázquez, E. (2008). *Laboratorios Virtuales de TechNet: una forma diferente de probar las soluciones Microsoft.* Recuperado el 5 de diciembre de 2010, de http://www.tecnologiapyme.com/software/laboratorios-virtuales-de-technet-una-forma-diferente-de-probar-las-soluciones-microsoft

Glosario de Términos

Dirección Lógica
Direcciones existentes en los programas.

Grado de Multiprogramación
Es el número de particiones existentes en la memoria.

Marcos de Páginas
Memoria de la máquina que se divide en bloques de igual longitud donde sólo se almacena una página.

Memoria Virtual
Es una técnica que consiste en permitir la ejecución de procesos que puedan no estar completamente en memoria.

Páginas
Programas que se dividen en unidades de tamaño fijo.

Software
Es la suma total de los programas de cómputo, procedimientos, reglas, documentación y datos asociados que forman parte de las operaciones de un sistema de cómputo y que “un producto de software es un producto diseñado para un usuario”.

Simulador
Es un aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los simuladores reproducen sensaciones que en realidad no están sucediendo.

Sistema Operativo
Programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permita su posterior implementación.