Universidad de las Ciencias Informáticas

Facultad 15

"Simulador para la asignatura Sistemas Operativos"

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

Autores: Lianni Yadira Figueredo Jiménez

Misael Rodríguez Urrutia

Tutor: Ing.Yelena Hernández Estrada

Tutor: Ing. Carlos Yasmany Hidalgo García

Ciudad de la Habana, Cuba

Junio, 2010

Todos y cada uno de nosotros paga puntualmente su cuota de sacrificio consciente de recibir el premio en la satisfacción del deber cumplido, conscientes de avanzar con todos hacia el Hombre Nuevo que se vislumbra en el horizonte.

Ernesto Guevara.

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 _____ del año_____

_____

Firma del autor

(Lianni Y. Figueredo Jiménez)

_____

Firma del autor

(Misael Rodríguez Urrutia)

_____

Firma del tutor

(Yelena Hernández Estrada)

_____

Firma del tutor

(Carlos Y. Hidalgo García)

Agradecimientos

Lianni

A mis amados padres por su apoyo, por ser lo más preciado y más lindo que tengo en la vida, por sus preocupaciones, por vivir junto a mi cada momento de tensión y desvelo, por saber guiarme en situaciones difíciles y enseñarme que todo es posible, por ser los mejores del mundo y por haber confiado siempre en mí.

A mí querido hermano por todo su amor y cariño.

A mis abuelos Abelardo, Minda y María que siempre han estado tan orgullosos de mí y por todo su amor, ese amor que solo ellos saben dar.

A mí amor Yuriesky por dedicarme cada uno de sus días, por su amor y cariño por siempre estar a mi lado cuando más lo necesité, por disfrutar de mis alegrías y sufrir con mis tristezas, por tantos momentos felices, Chuchi no sé qué hubieran sido mis días en la universidad sin ti.

A todos mis tíos, primos, quienes han dado muestra en todo momento de preocupación y entrega a mi porvenir.

A tres personas que han sido mi otra familia estos 5 años, Ana, Rogelito, y Rogelio y con los que he compartido momentos inolvidables.

A la familia de mi novio, en especial a mis suegros (Milagro, Betico y Yoelito) por su cariño, por quererme como a una hija y por haber confiado en mí.

A todas mis amistades, Ricardo, Michel, Ángel, Alexis David, Carmen, Dayana, Yunet, Daniry, Rey, Dunia, Anisley, Bianka por compartir juntos momentos de preocupación y alegría.

A todos los demás amigos que he conocido durante estos años y que recordaré siempre.

A la profe Arismaida, por haber contribuido de forma eficiente en mi formación profesional.

A mi compañero de tesis, Misael por haber realizado juntos este trabajo.

A mis tutores Carlos y Yelena por guiarme y ayudarme por tener respuesta para cada pregunta, sin ustedes todo hubiera sido más difícil muchas gracias.

Agradecimientos

Misael

A mis padres por su apoyo, por ser lo más preciado que tengo en la vida, por sus preocupaciones y por haber confiado siempre en mí, especialmente a mi madre.

A mis abuelos que siempre han estado tan orgullosos de mí.

A mí querido hermano por todo su amor.

A todos mis tíos, primos, quienes han dado muestra en todo momento de preocupación y entrega a mi porvenir.

A todas mis amistades, Daniel, Edel, Yoandris, José Manuel, y Yosdany, por compartir juntos momentos de preocupación y alegría.

A mi queridísima novia Anaili Sánchez Campos.

A todos los demás amigos que he conocido durante estos años y que recordaré siempre.

A mi compañera de tesis, Lianni.

A mis tutores Carlos y Yelena por guiarme y ayudarme por tener respuesta para cada pregunta, sin ustedes todo hubiera sido más difícil muchas gracias.

Dedicatoria

Dedicatoria

Lianni

A mis amados padres por ser el faro que me guía, por ser lo más bello que tengo en la vida.

A mi adorado hermano por su amor puro, sincero e incondicional y para que tenga presente que todo lo que uno se proponga en la vida puede lograrlo.

A mí adorado amor Yuriesky por su amor y cariño por ser tan especial en mi vida.

Dedicatoria

Misael

A mi madre por ser todo en mi vida.

A mis padres por brindarme su apoyo.

A mi hermano por confiar siempre en mí.

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 desarrollar un simulador 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, lenguajes de programación, 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, Administración de Memoria, Sistema Operativo, Tecnología"

Índice de Contenido

Introducción **1**

Capítulo 1 : Fundamentación Teórica **4**

1.1 Introducción **4**

1.2 Laboratorios virtuales **4**

1.2.1 Laboratorios virtuales software. 5

1.2.2 Laboratorios virtuales Web. 5

1.2.3 Laboratorios remotos 5

1.3 Ventajas de los laboratorios virtuales **6**

1.4 Simulación por Computadora **6**

1.5 Metodologías que se utilizan para el desarrollo de Software. **7**

1.5.1 Proceso Unificado de Desarrollo de Software (RUP). 7

1.5.2 Microsoft Solution Framework (MSF) 9

1.5.3 Programación Extrema (XP) 10

1.6 Lenguajes de Modelado **11**

1.6.1 IDEF0 12

1.6.2 BPMN 12

1.6.3 Lenguaje de Modelado Unificado. 12

1.7 Herramientas CASE **14**

1.7.1 Rational Rose Enterprise Edition 14

1.7.2 Visual Paradigm. 14

1.8 Lenguaje de desarrollo **16**

1.8.1 C# 16

1.8.2 C++ 16

1.8.3 Java 17

Índice de Contenido

1.9 Herramientas de desarrollo 18

1.9.1 Visual Studio 2005 18

1.9.2 Eclipse 19

1.9.3 NetBeans. 20

Capítulo 2 : Descripción de la solución 21

2.1 Introducción 21

2.2 Diagramas de flujo de procesos 21

2.3 Especificación de los requerimientos del software 23

2.3.1 Requisitos funcionales 24

2.3.2 Requisitos no funcionales 24

2.4 Actores del sistema 25

2.4.1 Diagrama de casos de uso del sistema 26

2.4.2 Descripción de los casos de uso del sistema 27

2.5 Análisis del sistema 35

2.5.1 Diagrama de Clases del Análisis 36

2.5.2 Diagramas de Interacción 37

2.6 Vista lógica de la arquitectura 38

2.7 Diseño 39

2.7.1 Diagrama de clases del diseño 40

2.7.2 Patrones de diseño 40

Capítulo 3 : Implementación y Prueba 43

3.1 Introducción 43

3.2 Implementación 43

3.2.1 Diagrama de Componentes 43

Índice de Contenido

3.3 Validación de la Solución por métricas 44

3.3.1 Métrica de la calidad de la especificación 44

3.3.2 Métricas para validar los casos de usos del sistema 45

3.3.3 Método de prueba por caja negra 53

Conclusiones 58

Bibliografía 60

Glosario de Términos 64

Anexos 65

Índice de Figuras

Figura 1: Ciclo de vida de la Metodología RUP.---------- 7
Figura 2: Fases e Iteraciones de la Metodología RUP.---------- 8
Figura 3: Metodología MSF. ---------- 9
Figura 4: Metodología Extreme Programing. ---------- 10
Figura 5: Diagrama de Proceso del CU "Inicializar Memoria Física" ---------- 22
Figura 6: Diagrama de Proceso del CU "Simular Técnica"---------- 22
Figura 7: Diagrama de Proceso del CU "Conformar Memoria Virtual" ---------- 23
Figura 8: Diagrama de Proceso del CU "Simular Algoritmos de Reemplazo"---------- 23
Figura 9: Diagrama de clases del análisis del CU "Inicializar Memoria Física" ---------- 36
Figura 10: Diagrama de clases del análisis del CU "Simular Técnicas"---------- 36
Figura 11: Diagrama de clases del análisis del CU "Conformar Memoria Virtual" ---------- 37
Figura 12: Diagrama de clases del análisis del CU "Simular Algoritmos de Reemplazo" ---------- 37
Figura 13: Diagrama de secuencia del CU "Simular Técnicas" ---------- 38
Figura 14: Diagrama de Clases---------- 40

Índice de Tablas

Tabla 1: Requisitos funcionales..... 24
Tabla 2: Actores del sistema..... 26

.

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. Algunas de ellas son la creación de los Joven Club de Computación y el surgimiento de una universidad de nuevo tipo llamada Universidad de las Ciencias Informáticas (UCI).

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

El uso del EVA en la UCI para la realización 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 es fundamental dentro del plan de estudio del 3er 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: proceso, memoria, entrada y salida de información.

La enseñanza del tema Administración de Memoria se torna un poco compleja tanto para impartirlo 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 debido a la carencia de herramientas que ayuden a la visualización de los procesos del tema.

Introducción

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.

Bajo estas condiciones es identificado el siguiente **problema científico**: Los medios usados para desarrollar el tema Administración de Memoria en la asignatura Sistema Operativo no son suficientes para apoyar la visualización de los contenidos. Este problema se enmarca en el **objeto de estudio**: Proceso de desarrollo de software cuyo **campo de acción** es: Proceso de desarrollo de software para la creación de simuladores. En este sentido la investigación que se presenta tiene como **objetivo general**, desarrollar un simulador que apoye la visualización de los contenidos del tema Administración de Memoria en la asignatura Sistema Operativo.

Se define como **hipótesis** que si se desarrolla un simulador para apoyar la visualización de los contenidos del tema Administración de Memoria en la asignatura Sistema Operativo, entonces se incrementarán los medios que se usan de apoyo a la misma.

Para el desarrollo exitoso del objetivo general se han definido una serie de **tareas investigativas** que lograrán el desarrollo eficiente de esta investigación:

- Elaborar el marco teórico de la investigación.
- Realizar un diagnóstico de la situación actual para identificar las principales necesidades del sistema en cuestión.
- Realizar el análisis y diseño de la aplicación.
- Desarrollar la aplicación.
- Validar la aplicación.

Introducción

A lo largo del presente trabajo se muestran 3 capítulos fundamentales lo cuales se enuncian 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 y se fundamenta su selección para el proceso de desarrollo. También se conocen aspectos como las herramientas de desarrollo, así como los lenguajes de programación a utilizar.

Capítulo2: Descripción de la solución propuesta. En este capítulo se describe la solución propuesta para el problema planteado mediante un diagrama de flujo de procesos. Se describen además los requisitos establecidos y se definen los actores, casos de usos con sus descripciones así como sus diagramas y todo lo referente al análisis y diseño de la aplicación.

Capítulo 3: Implementación y Prueba. En este capítulo se aborda el tema referente a la construcción y validación de la solución propuesta, mostrando un conjunto de artefactos generados en los flujos de trabajo Implementación y Prueba, como son los diagramas de componentes, la aplicación de métricas para validar los requisitos del sistema, los casos de uso y las pruebas de caja negra aplicadas al sistema.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En el presente capítulo se abordarán, diferentes temas entre los que se encuentran, qué es 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, conjuntamente 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), lenguajes de modelado y lenguajes de desarrollo que se van a 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, 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”. (Monge-Nájera, 1999).

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

Existen diversos tipos de laboratorios virtuales, según la bibliografía consultada, se proponen 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). También determinados laboratorios virtuales pensados inicialmente como aplicaciones Java accesibles a través de un servidor Web se pueden considerar de este tipo si funcionan localmente y no necesitan recursos de un servidor en concreto.

1.2.2 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.

1.2.3 Laboratorios remotos.

Los laboratorios remotos son aquellos que permiten operar remotamente cierto equipamiento, bien sea didáctico como maquetas específicas, o industrial, además de poder ofrecer capacidades de laboratorio virtual. En general, estos requieren de equipos servidores específicos que les den acceso a las máquinas a operar de forma remota, y no pueden ofrecer su funcionalidad ejecutándose de forma local. Otro motivo que los hace dependientes de sus servidores es la habitual gestión de usuarios en el servidor. (HERÍAS, 2003)

En este sentido se va a trabajar con los laboratorios virtuales web a causa de que en la asignatura Sistema Operativo se están dando los primeros pasos en la realización de un laboratorio virtual, el mismo se va a incorporar en el EVA y va a tener integrado el trabajo de diploma realizado que forma parte de otro de sus módulos, con una aplicación de escritorio que es un simulador, facilitando que cualquier profesional o alumno interesado en el tema desarrollado pueda hacer uso del mismo y beneficiarse de sus prestaciones.

Capítulo 1: Fundamentación Teórica

1.3 Ventajas de los laboratorios virtuales

El uso de laboratorios virtuales tiene sin duda muchos beneficios, permite al estudiante buscar información, en él se pueden relacionar fenómenos con sus consecuencias, se pueden repetir los eventos o fenómenos cuantas veces se requiera. Se incorporan las tecnologías de la información y 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 más significativas como son:

- Simula situaciones que en la realidad tendrían escasas posibilidades de realizarlas.
- Se convierten en una ayuda interactiva para el aprendizaje de contenidos difíciles de demostrar en la realidad.
- 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, 2005)
- Es una herramienta de auto aprendizaje. (Herreros, 2005)

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

1.4 Simulación por Computadora

La simulación es el proceso de diseñar un modelo de un sistema real y llevar a término experiencias con él, 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.

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 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.

1.5.1 Proceso Unificado de Desarrollo de Software (RUP).

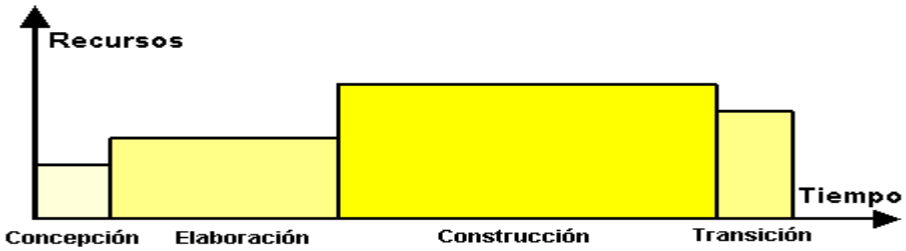
La metodología RUP (Rational Unified Process) es una metodología para la ingeniería de software que va más allá del simple análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software, se caracteriza por ser:

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

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

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

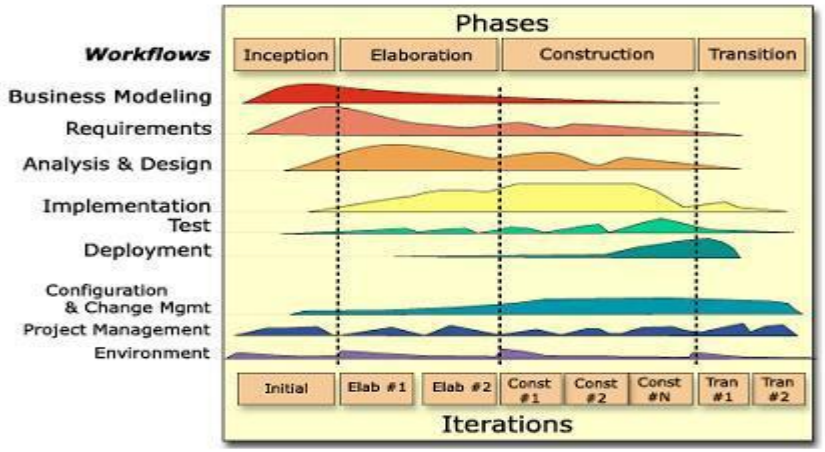
Figura 1: Ciclo de vida de la Metodología RUP.



Tomado de:(Sánchez, 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. (Sánchez, 2004)

Figura 2: Fases e Iteraciones de la Metodología RUP.



Tomado de: (Sánchez, 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. (Sánchez, 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. (Sánchez, 2004)

1.5.2 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: (Sánchez, 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. (Sánchez, 2004)

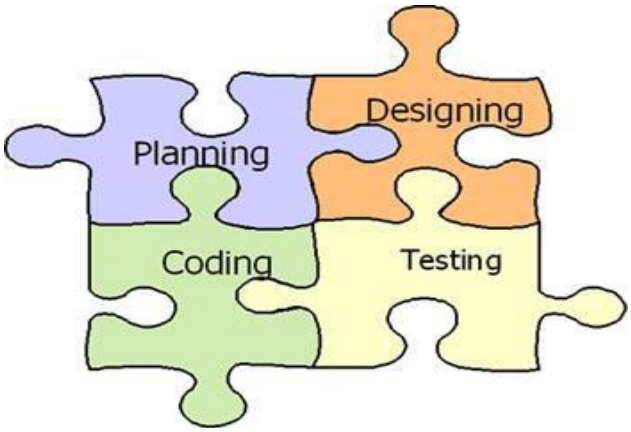
Concretamente MSF se compone de principios, disciplinas y modelos, los principios no son más que promover la comunicación abierta, trabajar para una visión compartida, fortalecer los miembros del equipo,

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.

1.5.3 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: (Sánchez, 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. (Sánchez, 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.

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.

1.6 Lenguajes de Modelado

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

1.6.1 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. (Estrada, 2009)

1.6.2 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.

El 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 *end-to-end*, con diferentes niveles de fidelidad. (Barriento, 2008)

1.6.3 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:

- 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.

Luego de analizadas estas notaciones de modelado se decide utilizar dos de estos lenguajes, UML que brinda la posibilidad de construir los modelos que define la metodología escogida para desarrollar el software, lo que permite expresar requisitos y representar todos sus detalles, además hace que se obtenga una documentación que es válida durante todo el ciclo de vida de un proyecto. Al mismo tiempo es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad para modelar los artefactos creados durante el proceso de desarrollo de software. 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.

1.7 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.

1.7.1 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. (Estrada, 2009)

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.

1.7.2 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, apoya la importación y exportación de XML de versiones 1.0, 1.2 y 2.1. 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 tiene 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. (Estrada, 2009)

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.

Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como obtener diagramas a partir de código. (Giraldo, 2005).

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.

1.8 Lenguaje de desarrollo

Para desarrollar un software es necesario una serie de requisitos, entre ellos se encuentra el conjunto de símbolos y de reglas tanto sintácticas como semánticas que indican al ordenador qué acciones desarrollar, a esta aglomeración de instrucciones se le llama lenguaje de programación. (Almaguer, 2009)

1.8.1 C#

C# es un lenguaje de programación sencillo y moderno, orientado a objetos y con seguridad de tipos. C# combina la gran productividad de los lenguajes de desarrollo rápido de aplicaciones (RAD, Rapid Application Development) con la eficacia de C++. El código de C# es compilado como código administrado, lo cual le permite beneficiarse de los servicios del Common Languaje Runtime (CLR). Estos servicios incluyen interoperabilidad, recolección de basura, seguridad y gestión de versiones.

La sintaxis y estructuración de C# es muy similar a la C++, ya que la intención de Microsoft con C# es facilitar la migración de códigos escritos en este lenguaje a C# y su aprendizaje a los desarrolladores habituados a este. (Borges., 2009)

Es importante aclarar que a pesar de esto para poder instalar C# requiere de Windows NT4 o superior y por lo menos 4 GB de espacio en el disco duro para su instalación.

1.8.2 C++

C++ es un lenguaje diseñado a mediados de los años 1980, por Bjarne Stroustrup. Es un súper conjunto de C que tiene como característica principal, el soporte para programación orientada a objetos y de plantillas o programación genérica. Se creó para añadirle cualidades y características de las que carecía C. Entre los paradigmas que abarca C++ se pueden encontrar la programación estructurada, la programación genérica y la programación orientada a objetos.

Este lenguaje mantiene la potencia para programar a bajo nivel y se la han añadido elementos que le permiten también un estilo de programación con alto nivel de abstracción. Una de las razones de

programar en C++ es su increíble versatilidad. Con él pueden programarse desde las aplicaciones más simples, a las más complejas como sistemas operativos. Su portabilidad permite que un programa escrito en C++ pueda compilarse en cualquier sistema sin necesidad de cambiar apenas el código.

A pesar de esto C++ es un lenguaje de programación difícil de aprender, debido principalmente al trabajo con punteros. (Pelegrino, 2009)

1.8.3 Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Utilizando Java, se pueden eliminar los inconvenientes de la interfaz Common Gateway Interface (CGI) y también se pueden añadir aplicaciones que vayan desde experimentos científicos interactivos de propósito educativo a juegos o aplicaciones especializadas para la tele venta. Es posible implementar publicidad interactiva y periódicos personalizados.

Java posee una curva de aprendizaje muy rápida. Resulta relativamente sencillo desarrollar applets interesantes desde el principio. A pesar de ser muy parecido a C y C++, es más fácil ya que se han eliminado algunas características como es el caso de los punteros lo que lo hace más cómodo, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel.

Se puede decir que Java es un lenguaje de programación que facilita la creación de aplicaciones distribuidas ya que proporciona una colección de clases para aplicaciones en red.

Una de las características más importantes que tiene Java es la indiferencia a la arquitectura ya que es compatible con los más variados entornos de red cualesquiera sean estos desde Windows 95, Unix a Windows Nt y Mac, para poder trabajar con diferentes sistemas operativos. Java es muy versátil ya que utiliza bytecodes que es un formato intermedio que sirve para transportar el código eficientemente o de diferentes plataformas.

Por ser indiferente a la arquitectura sobre la cual está trabajando, hace que su portabilidad sea muy eficiente, sus programas sean iguales en cualquiera de las plataformas porque especifica tamaños básicos, esto se conoce como la máquina virtual de Java. (Gonzalez, 2009)

Luego de realizar un estudio de los lenguajes de programación más utilizados, se puede llegar a la conclusión de que Java es el más indicado para desarrollar el sistema que se desea implementar. El hecho de que es multiplataforma es muy importante, porque existen compiladores del propio lenguaje para la mayoría de las plataformas e intérpretes de Java para todas, además el funcionamiento del programa es el mismo en todas las plataformas y sólo cambia la apariencia que se adapta a la del sistema operativo que lo ejecuta (Windows, Solaris, Linux, etc.). Más importante aún es la característica de su indiferencia a la arquitectura sobre la cual se está trabajando, lo que hace que su portabilidad sea muy eficiente y además que sus programas sean iguales en cualquier plataforma debido a la máquina virtual de Java que corre sobre cualquier sistema operativo.

1.9 Herramientas de desarrollo

Las herramientas de desarrollo son todos aquellos programas o aplicaciones que tienen cierta importancia en el desarrollo de un programa (programación), ya que con la ayuda y posibilidades que brindan se desarrollará mejor el programa o software.

1.9.1 Visual Studio 2005

Visual Studio .NET es un IDE desarrollado por Microsoft a partir del 2002. Es para el Sistema Operativo Microsoft Windows y está pensado, principal pero no exclusivamente, para desarrollar plataformas Win32.

Visual Studio Standard Edition es el punto de entrada en las herramientas de desarrollo profesionales, manteniendo la simplicidad de las versiones Express pero ofreciendo acceso a un poderoso conjunto de herramientas de desarrollo necesarias para la creación de aplicaciones cliente orientadas al manejo de datos, aplicaciones en n capas (Sistemas Conectados) utilizando servicios Web, y ricas aplicaciones Web.

Con Visual Studio 2005 los desarrolladores pueden crear aplicaciones de líneas de negocio utilizando Visual Basic, C#, C++ y J#, realizar aplicaciones para Windows, la Web y dispositivos móviles desde un mismo entorno unificado de desarrollo, construir aplicaciones cliente/servidor usando servicios Web e integrando herramientas de diseño para acceder a datos remotos.

1.9.2 Eclipse

La herramienta de desarrollo Eclipse IDE comenzó como un proyecto de International Business Machines (IBM) Canadá.

Esta herramienta presenta una característica principal y es que emplea módulos (en inglés *plugin*) para proporcionar toda su funcionalidad al frente de la plataforma de cliente rico, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Este mecanismo de módulos es una plataforma ligera para componentes de software. Adicionalmente, al permitirle a Eclipse extenderse usando otros lenguajes de programación como son C/C++, Python, permite trabajar con lenguajes para procesado de texto como LaTeX. La arquitectura plugin permite escribir cualquier extensión deseada en el ambiente.

El Software Development Kit (SDK) de Eclipse incluye las herramientas de desarrollo de Java, ofreciendo un IDE con un compilador de Java interno y un modelo completo de los archivos fuente de Java. Esto permite técnicas avanzadas de refactorización y análisis de código. El IDE también hace uso de un espacio de trabajo, en este caso un grupo de metadata en un espacio para archivos planos, permitiendo modificaciones externas a los archivos en tanto se refresque el espacio de trabajo correspondiente.

Esta herramienta presenta las características fundamentales como son:

- Editor de Texto
- Resaltado de Sintaxis
- Compilación en tiempo real
- Pruebas unitarias con Junit
- Control de Versiones
- Asistentes (Wizards), para la creación de proyectos, clases, test, etc.
- Refactorización

1.9.3 NetBeans.

NetBeans es un proyecto de código abierto fundado y patrocinado por Sun MicroSystems, es un IDE que permite escribir, compilar, corregir errores y ejecutar programas. Permite crear aplicaciones de escritorio, web y para dispositivos móviles. Soporta además lenguajes dinámicos como PHP, Java Script, Groovy y Ruby. Admite desarrollar aplicaciones usando la plataforma J2EE. Puede ser usado en varias plataformas entre las que se encuentran Windows, Linux, Mac OS X y Solaris.

Después de hacer un minucioso estudio de las herramientas anteriores, para el desarrollo de la aplicación se hará uso de NetBeans debido a que es libre y de código abierto, tiene una interfaz muy amigable e intuitiva, característica fundamental en un IDE, tiene todas las herramientas para crear aplicaciones profesionales ya sean de escritorio, empresariales, web, móviles y aplicaciones SOA, no solo en Java sino también en C/C++ y Ruby, puedes hacer diagramas UML y es de múltiples lenguajes.

Conclusiones Parciales

Con el estudio de los laboratorios virtuales, la simulación por computadora y los beneficios que propician para un mejor aprendizaje de los estudiantes, se demostró que sería de mucha utilidad su utilización en la asignatura Sistema Operativo y más aún en el tema Administración de Memoria, para poder alcanzar una mayor comprensión y visualización de los procesos en este contenido por parte de los estudiantes. Siendo así se definieron las herramientas y los lenguajes a ser usados para la elaboración del producto; como herramienta de desarrollo se utilizará NetBeans, determinado por ésta el lenguaje de programación Java. Para guiar el proceso de desarrollo de la aplicación se determinó el uso de la metodología RUP, utilizando como lenguajes de modelado UML, BPMN y como herramienta CASE a Visual Paradigm.

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 Diagramas de flujo de procesos

En el presente trabajo se realiza la simulación de los procesos del tema Administración de Memoria descritos desde el punto de vista teórico en el contenido # 4 de la asignatura Sistema Operativo. Como autor fundamental de referencia se utilizó Andrew S. Tanenbaum. Como flujos críticos se tienen Inicializar la Memoria Física, el proceso de inicialización de la memoria se realiza en dos momentos distintos cuando se trabaja con técnicas de asignación y con algoritmos de reemplazo, en el primer caso tiene como entrada ubicar el tamaño que tiene la memoria física y como salida se muestran los procesos que están en memoria, el segundo caso igualmente tiene como entrada ubicar el tamaño de la memoria, luego se escoge la unidad de medida en que está y en dependencia de esa información tiene como salida entrar distintos datos, como son desplazamiento, cantidad de marcos, cantidad de entradas, número de páginas y tamaño de la página, para mayor comprensión a continuación se presenta un diagrama de proceso que explica el funcionamiento.

Ver los demás diagramas en los **Anexos del (1 al 8)**

Capítulo 2: Descripción de la solución

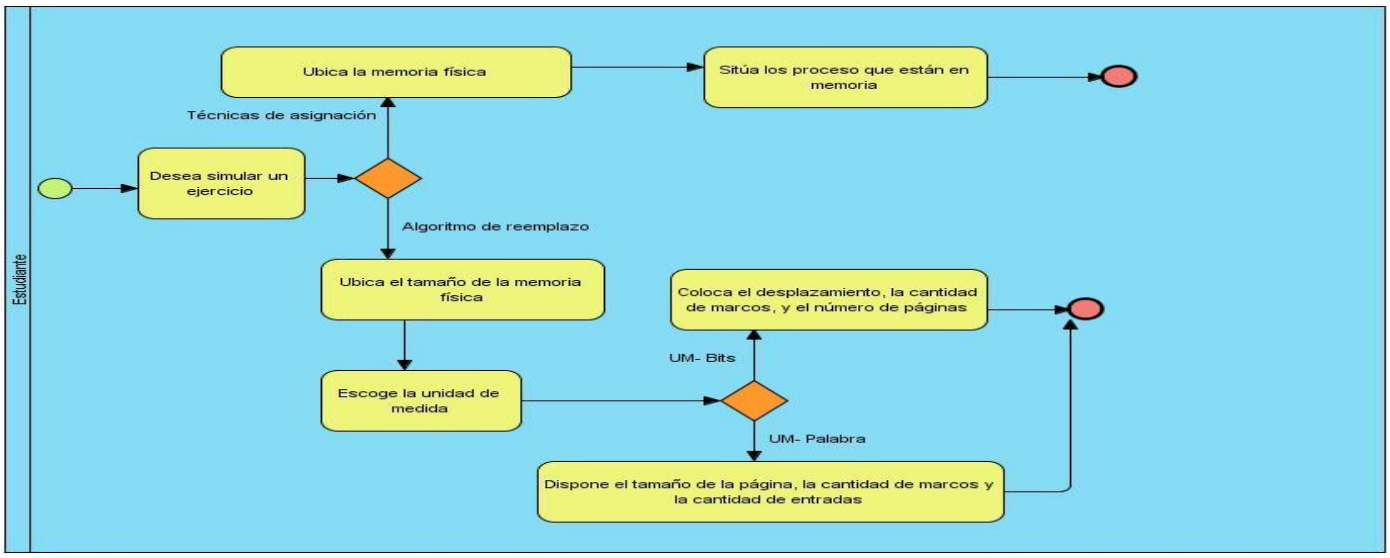


Figura 5: Diagrama de Proceso del CU "Inicializar Memoria Física"

Otro de los flujos críticos es Simular Técnicas, este proceso tiene como entrada la inicialización de la memoria física, después se decide el algoritmo para simular y la salida es los nuevos procesos ubicados según el funcionamiento del algoritmo. En el diagrama siguiente se muestra el funcionamiento.

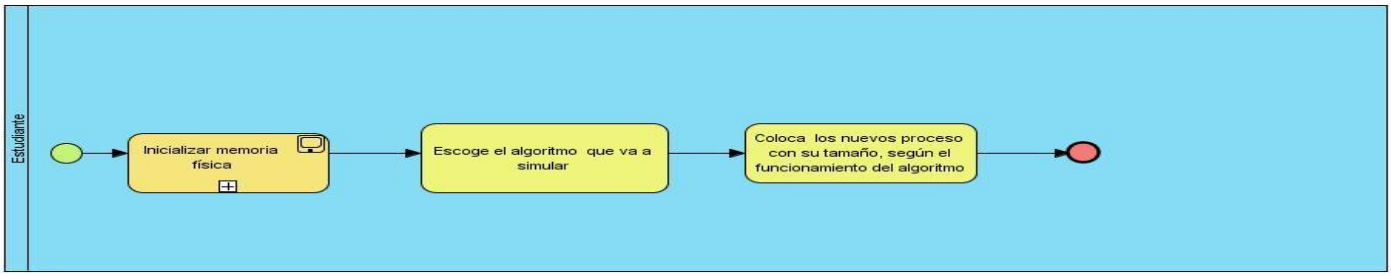


Figura 6: Diagrama de Proceso del CU "Simular Técnica"

Además de los procesos anteriores también se encuentra Conformar la Memoria Virtual, en este se tiene como entrada preliminar el flujo de inicialización de memoria, luego se tiene en cuenta el estado de la memoria y en dependencia de esto se ubican los marcos, el tiempo, los procesos con su página en cada marco y tiempo, teniendo como salida la memoria conformada. En el siguiente diagrama se presenta todo su funcionamiento.

Capítulo 2: Descripción de la solución

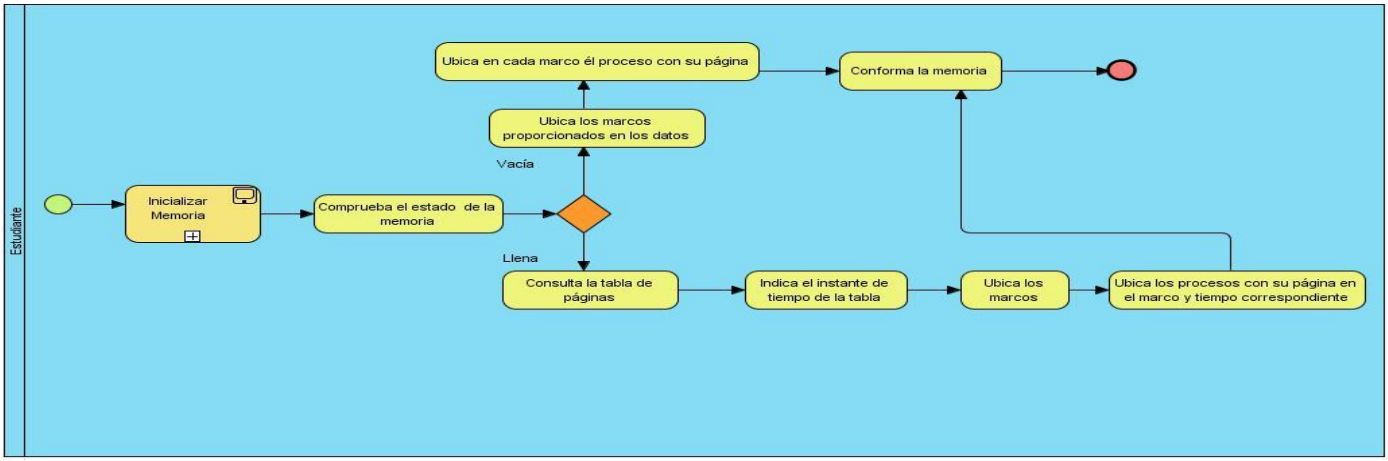


Figura 7: Diagrama de Proceso del CU "Conformar Memoria Virtual"

Aparte de los anteriores procesos se encuentra también el de Simular Algoritmos de Reemplazo, este tiene como entrada inicial conformar la memoria virtual, después de realizar esto se escoge el algoritmo de reemplazo y la política, en dependencia del funcionamiento de la misma se realiza el reemplazo de página, teniendo como salida el diagrama de situación de páginas, con todos los fallos y reemplazos realizados. A continuación se muestra el diagrama donde se especifica con detalles.

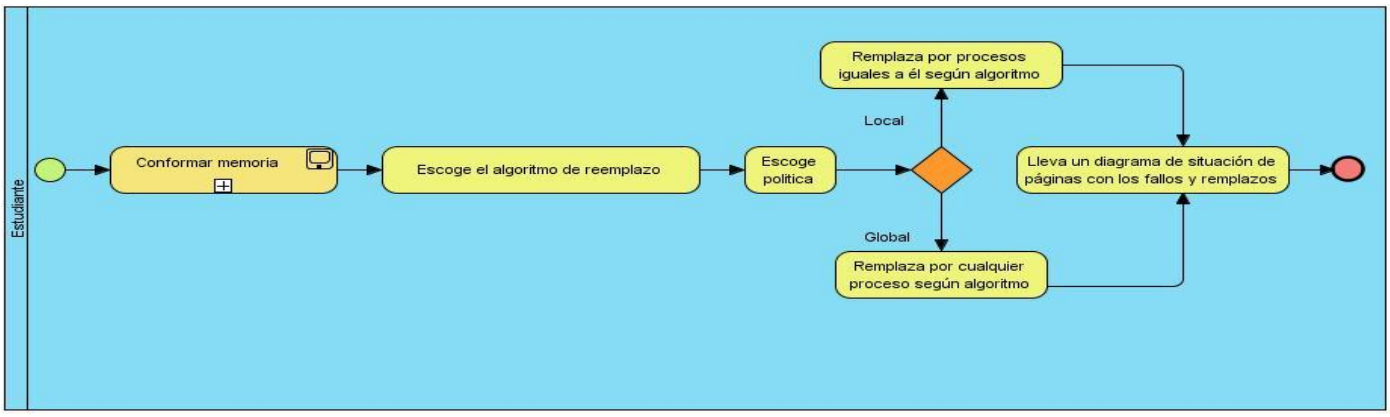


Figura 8: Diagrama de Proceso del CU "Simular Algoritmos de Reemplazo"

2.3 Especificación de los requerimientos 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, 2009)

Los Requisitos Funcionales para este sistema se presentan a continuación

Tabla 1: Requisitos funcionales

Referencia	Requisitos Funcionales
RF1	Inicializar memoria física
RF2	Visualizar estado de la memoria física
RF3	Visualizar 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

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, 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.

2.4 Actores del sistema

Los actores del sistema representan un rol que juega una o varias personas, un equipo o un sistema automatizado, pueden intercambiar información con él, pero no son parte de él. (Estrada, 2009)

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

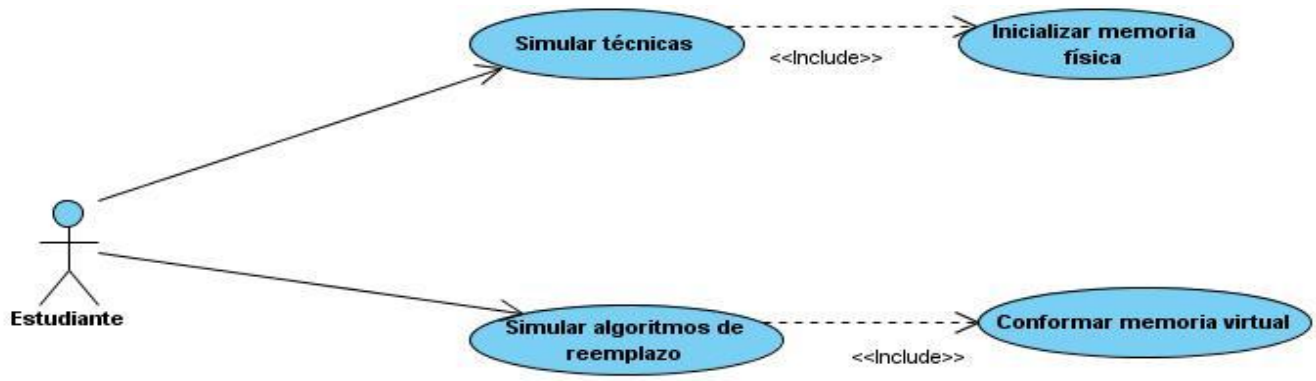
Actor del sistema	Descripción
Estudiante	Es el usuario encargado de realizar cualquier operación dentro del sistema, como inicializar la memoria física, simular cualquier técnica de asignación o algoritmos de reemplazos.

Tabla 2: Actores del sistema

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, 2009)

El presente diagrama se explica a continuación.



El estudiante puede realizar la simulación de todas las técnicas de asignación, 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 Simular técnicas, la demás descripciones se encuentran el documento modelo de caso de uso del sistema.

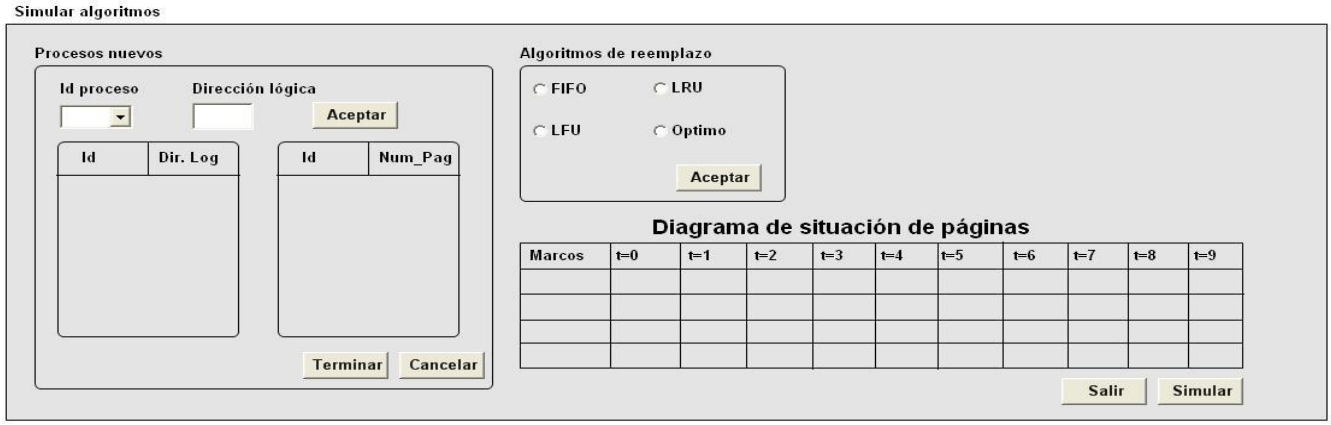
Descripción del Caso de uso: "Simular Algoritmos de reemplazo"

Caso de uso:	Simular Algoritmos de reemplazo
Actores:	Estudiante
Resumen:	El caso de uso inicia cuando el estudiante selecciona uno de los algoritmos de reemplazo para ejecutar alguna de sus páginas en memoria física, y termina cuando la página es reemplazada.
Precondiciones:	Conformar la memoria virtual
Pos condición:	Visualizar el estado final de la memoria, tiempo transcurrido y la cantidad de fallos y reemplazos.
Casos de usos asociados:	**CU Conformar memoria virtual (incluido)**
Referencia:	RF12
Prioridad:	Crítico
Flujo Normal de Eventos	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona la opción *Algoritmos de reemplazo.*	2. Se invoca el **CU incluido Conformar Memoria Virtual.**

	3. El sistema muestra la interfaz Procesos Nuevos con las siguientes opciones activadas: • Id del proceso • Dirección lógica
4. El usuario introduce los datos y presiona la opción aceptar.	5. El sistema verifica que no hayan campos vacíos.
	6. El sistema comprueba que los datos entrados sean correctos.
7. El usuario presiona la opción terminar.	8. El sistema muestra los algoritmos de reemplazo. • OPT (Algoritmo óptimo). • FIFO (First Come First Out). • LRU (Least Recently Used). • LFU (Least Frequently Used).
9. El usuario selecciona el algoritmo deseado y selecciona la opción aceptar. OPTIMO (Algoritmo óptimo). **Ir a la sección 1** FIFO (First In First Out). **Ir a la sección 2** LRU (Least Recently Used). **Ir a la sección 3.** LFU (Least Frequently Used). **Ir a la sección 4**	10. El sistema chequea el algoritmo seleccionado por el usuario.
	11. El sistema activa la opción simular.
12. El usuario selecciona la opción simular.	13. El sistema realiza la simulación en dependencia del algoritmo seleccionado y la política de reemplazo escogida.
	14. El sistema muestra el estado final de la

	memoria y la cantidad de fallos y reemplazos.

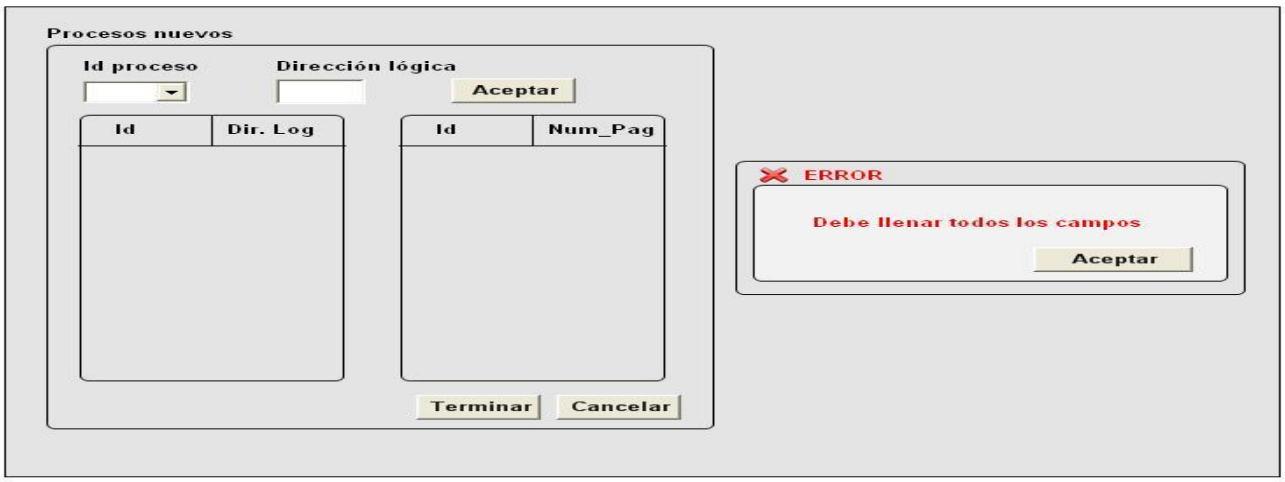
Prototipo de interfaz de usuario



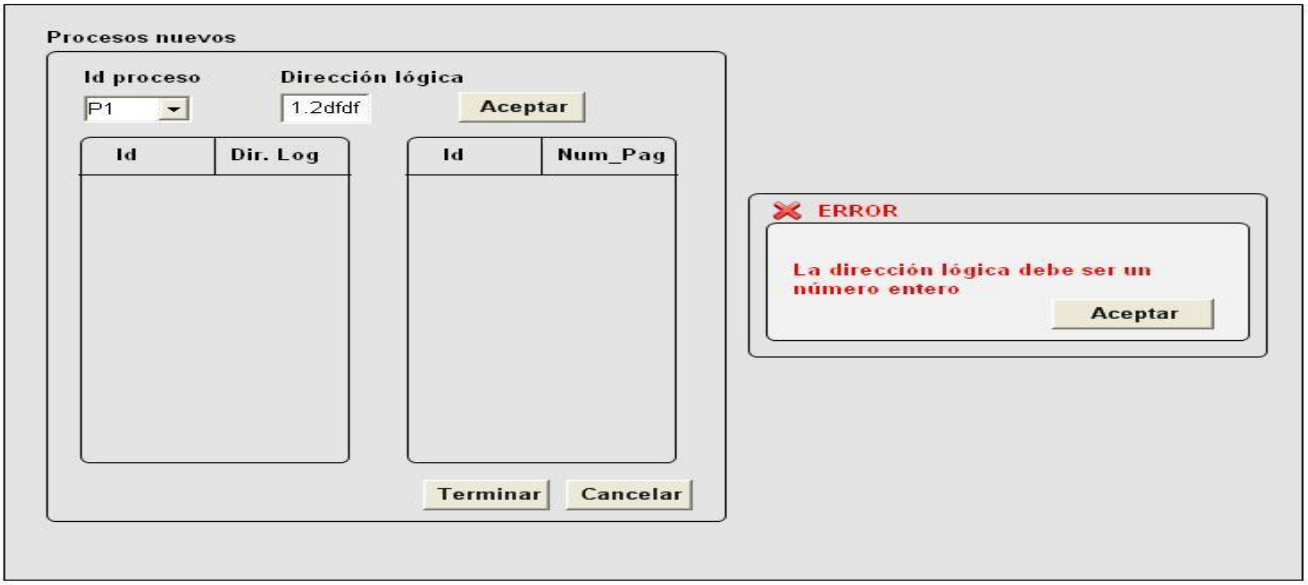
Flujo Alterno del paso 5

Acción de actor	Acción del sistema
	5a.1 En caso de haber algún campo vacío el sistema muestra el siguiente mensaje de error:"Debe llenar todos los campos".

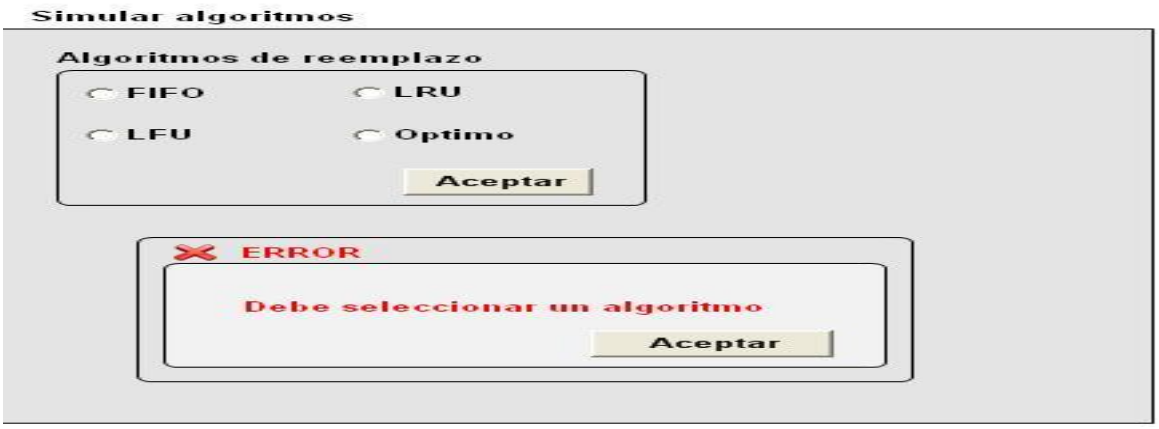
Prototipo de interfaz de usuario



Flujo Alterno del paso 6	
Acción de actor	**Acción del sistema**
	6a.1 En caso de que los datos entrados sean incorrectos el sistema muestra el siguiente mensaje de error: “La dirección lógica debe ser un número entero”.
Prototipo de interfaz de usuario	

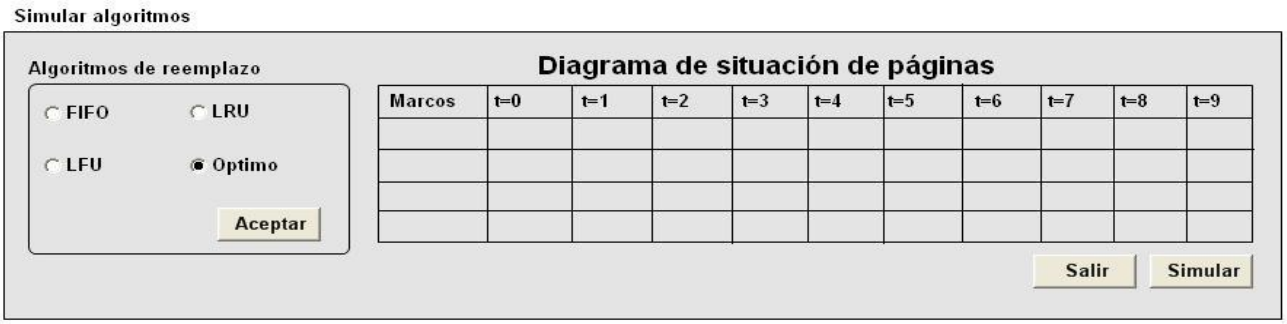


Flujo Alterno del paso 10	
Acción de actor	**Acción del sistema**
	10a.1 En caso de no haber seleccionado ningún algoritmo el sistema muestra el siguiente mensaje de error:" Debe seleccionar un algoritmo".
Prototipo de interfaz de usuario	



Sección 1 "Óptimo"	
Acción del actor	**Respuesta del sistema**
	1. El sistema reemplaza la página de memoria que más tardará en ser referenciada o que no volverá a entrar a memoria.
	2. El sistema incrementa el número de reemplazos.
	3. El sistema muestra la memoria con la nueva página.

Prototipo de interfaz de usuario



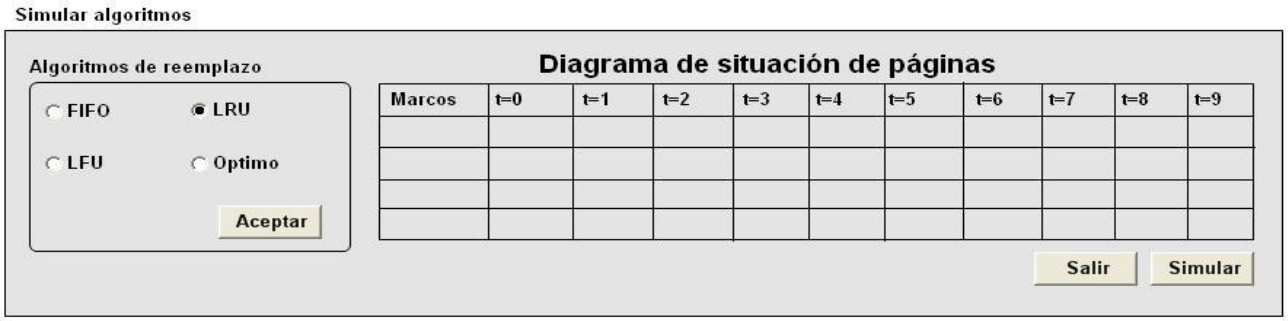
Sección 2 “FIFO”	
Acción del actor	**Respuesta del sistema**
	1. El sistema reemplaza la página que lleva más tiempo en la memoria.
	2. El sistema incrementa el número de reemplazos.
	3. El sistema muestra la memoria con la nueva página.

Prototipo de interfaz de usuario

Sección 3 “LRU”	
Acción del actor	**Respuesta del sistema**
	1. El sistema chequea si la página correspondiente al proceso seleccionado se encuentra en memoria.
	2. El sistema reemplaza a la página menos recientemente usada, es decir la que hace más tiempo no se usa y actualiza el contador.
	3. El sistema incrementa el número de reemplazos.
	4. El sistema muestra la memoria con la nueva

	página y el contador actualizado.

Prototipo de interfaz de usuario

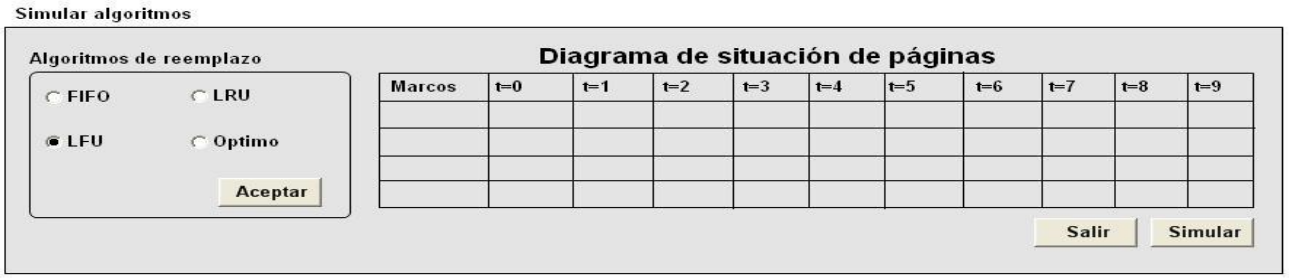


Flujo Alterno

Acción del actor	Respuesta del sistema
	1a.1 Si la página correspondiente al proceso seleccionado se encuentra en memoria el sistema actualiza el contador.

Sección 4 "LFU"

Acción del actor	Respuesta del sistema
	1. El sistema chequea si la página correspondiente al proceso seleccionado se encuentra en memoria.
	2. El sistema reemplaza a la página menos frecuentemente usada, es decir la que hace más tiempo no se referencia y actualiza el contador.
	3. El sistema incrementa el número de reemplazos.
	4. El sistema muestra la memoria con la nueva página y el contador actualizado.

Prototipo de interfaz de usuario


Flujo Alterno	
Acción del actor	**Respuesta del sistema**
	1a.1 Si la página correspondiente al proceso seleccionado se encuentra en memoria el sistema actualiza el contador.

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. Se muestran los diagramas de clases del análisis de los casos de usos más significativos. (Almaguer, 2009)

A continuación se muestran los diagramas de clases del análisis correspondiente a los casos de uso.

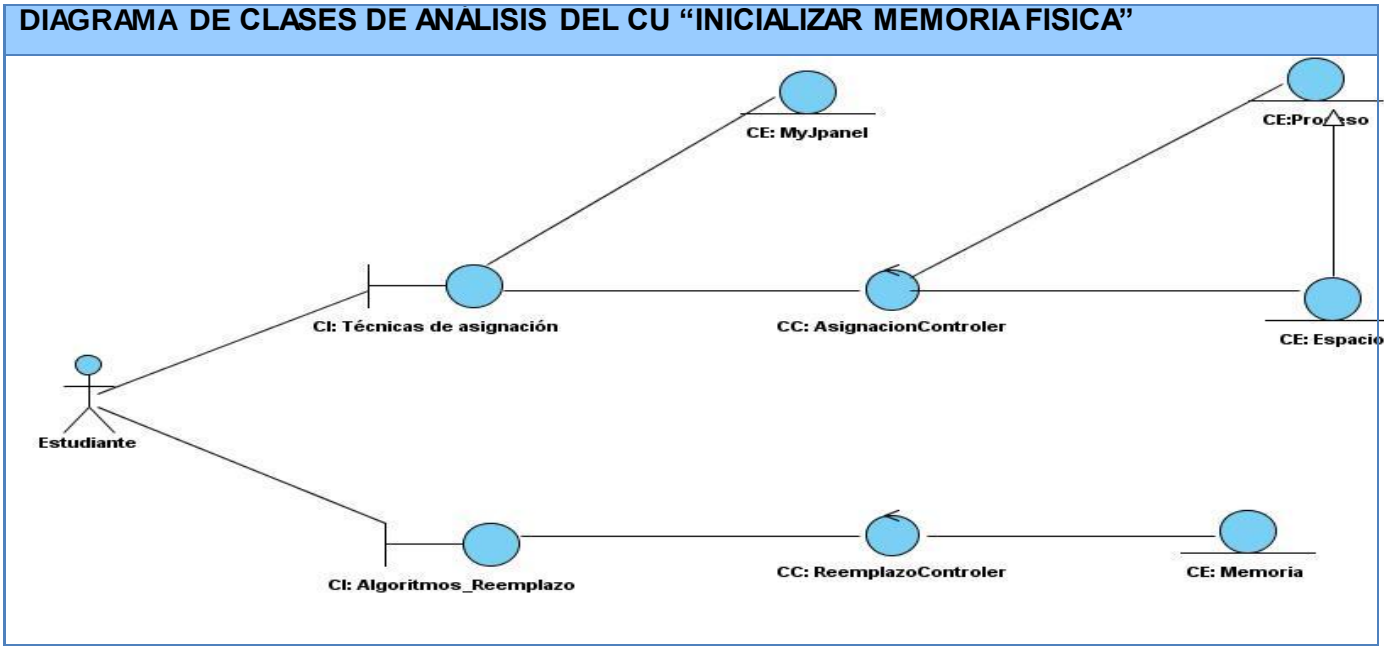


Figura 9: Diagrama de clases del análisis del CU "Inicializar Memoria Física"

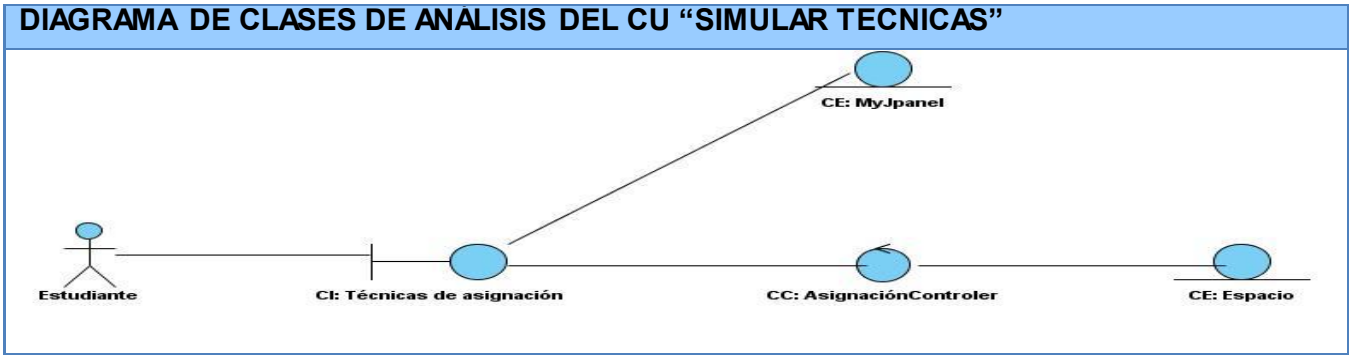


Figura 10: Diagrama de clases del análisis del CU "Simular Técnicas"

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

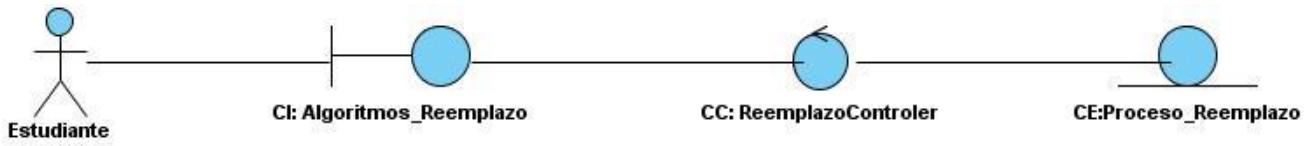


Figura 11: Diagrama de clases del análisis del CU "Conformar Memoria Virtual"

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

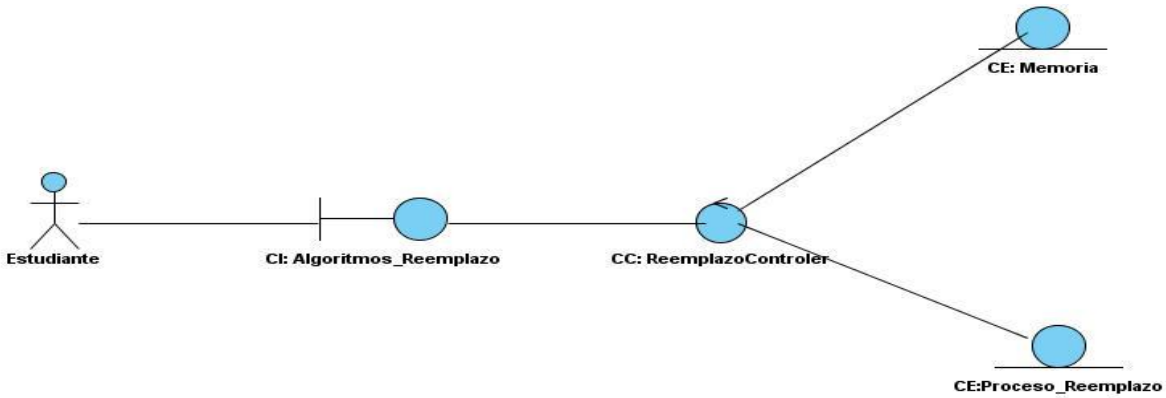


Figura 12: Diagrama de clases del análisis del CU "Simular Algoritmos de Reemplazo"

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, Simular técnicas,

Conformar memoria virtual y Simular 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.

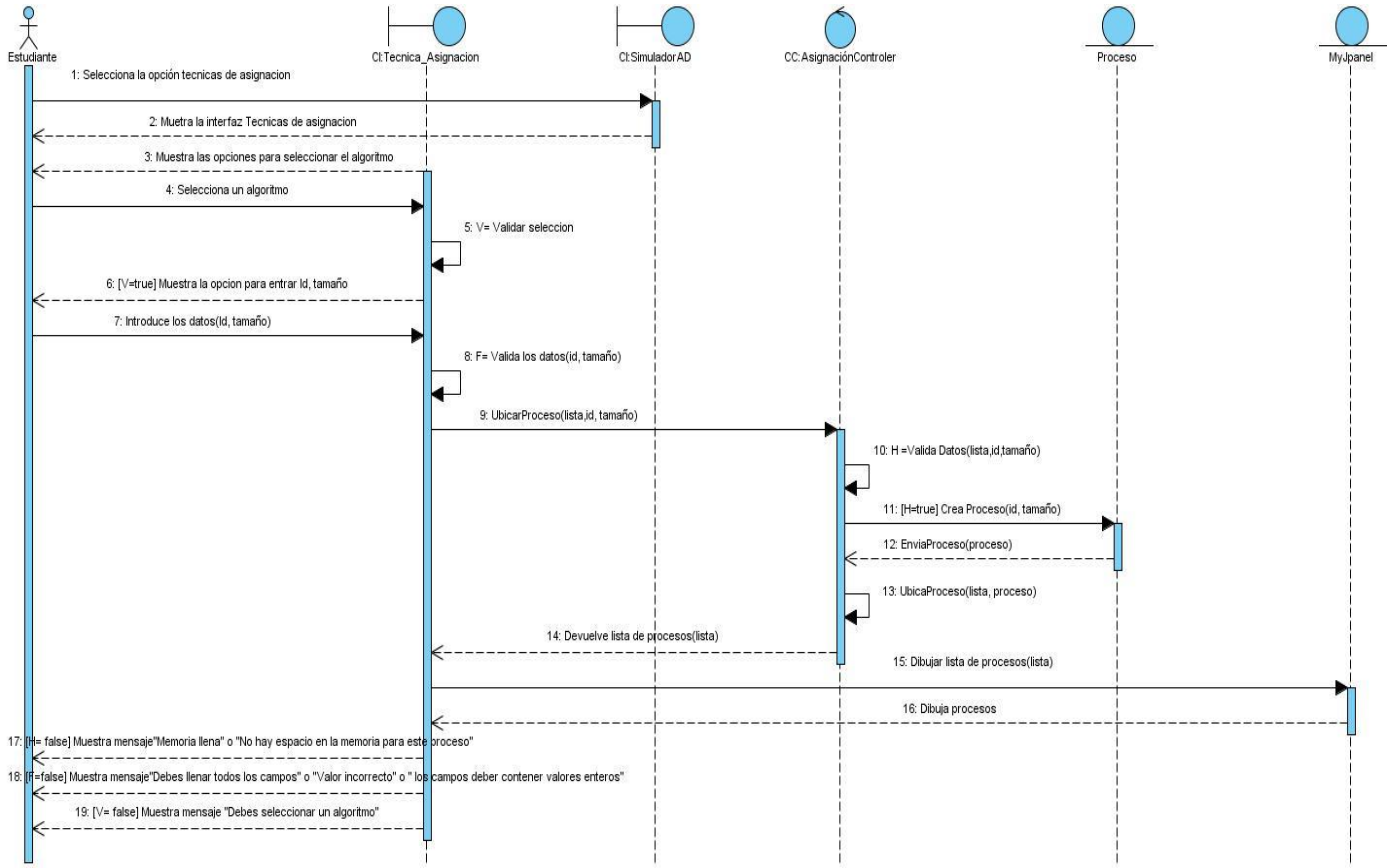


Figura 13: Diagrama de secuencia del CU "Simular Técnicas"

2.6 Vista lógica de la arquitectura

Esta vista es un conjunto del artefacto Modelo de diseño, la cual representa los elementos de diseño más importante para la arquitectura del sistema.

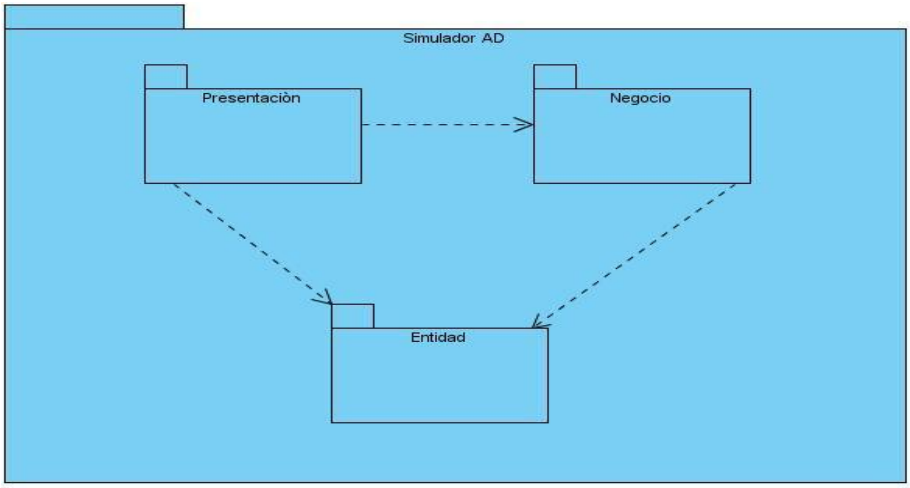


Figura 14: Vista lógica de la arquitectura

En la vista lógica se puede apreciar un paquete general llamado SimuladorAD compuesto principalmente por tres capas, (Presentación, Negocio y Entidad). Lográndose un menor acoplamiento entre las clases y respondiendo así al patrón Bajo Acoplamiento. Además a la hora de darle mantenimiento al código fuente es más fácil porque un cambio en una clase no implica cambio en todas las demás clases de la aplicación.

En la capa Presentación se encuentran todas las clases necesarias para que el usuario pueda ver la aplicación. Es aquí donde se validan los datos para que no se encuentren errores, se muestra y captura la información del usuario. Esta capa solo se comunica con la capa Negocio.

En la capa Negocio se pueden encontrar las clases controladoras que organizan todo el trabajo de los datos de la aplicación y las de estructuras de datos. Es en este nivel donde se reciben todas las peticiones del usuario y se le dan respuestas. También esta capa recibe las solicitudes de la capa Presentación y se comunica con la capa Entidad para solicitar datos.

En la capa Entidad se encuentran todas las clases que manejan los datos durante el funcionamiento de la aplicación. Además este nivel brinda las funcionalidades comunes a los demás niveles. Esta capa no se comunica con ninguna otra y recibe peticiones de los demás niveles.

2.7 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.

2.7.1 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.

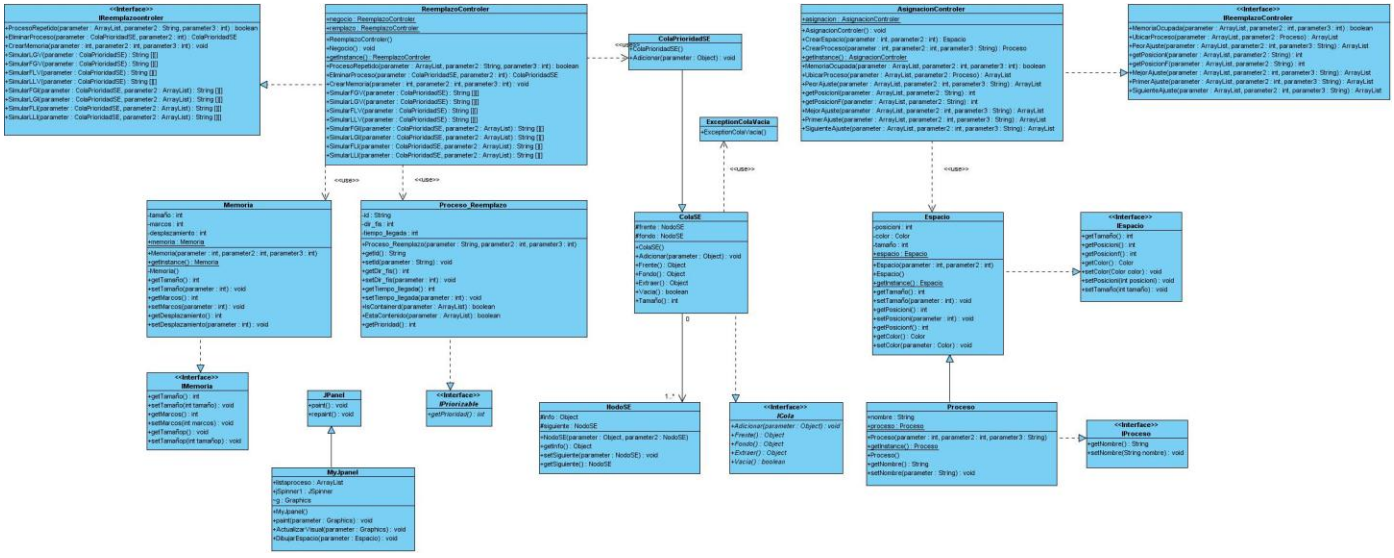


Figura 15: Diagrama de Clases

2.7.2 Patrones de diseño

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.

- **Bajo Acoplamiento:** El acoplamiento mide qué tan fuerte está una clase conectada con otras (es decir, cuántas clases conoce y necesita). Una clase con bajo acoplamiento no depende de "muchas otras" clases. Una clase con alto acoplamiento recurre a muchas otras clases. Este tipo de clase no es conveniente, pues: cambios en las clases relacionadas ocasionan cambios en la clase local; son más difíciles de entender; son más difíciles de reutilizar.

- **Alta cohesión:** Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Cada elemento de nuestro diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y auto-identificable, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo.

- **Singleton:** El patrón de diseño Singleton (instancia única) está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Este patrón se implementa creando en nuestra clase un método que crea una instancia del objeto sólo si todavía no existe alguna.

Conclusiones Parciales

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 y de mayor calidad.

Capítulo 2: Descripción de la solución

Capítulo 3: Implementación y Prueba

3.1 Introducción

El flujo de trabajo de Implementación tiene como objetivo definir la organización del código teniendo en cuenta los subsistemas de implementación, la implementación de los elementos de diseño en términos de ficheros fuentes, binarios y ejecutables, para poder integrar los diferentes componentes de desarrolladores o equipos y generar un ejecutable entregable o producto final. En este capítulo se presenta el artefacto diagrama de componentes, el cual muestra las dependencias entre estos. Además de la etapa de prueba, que se le realiza al software con el objetivo de asegurar completitud, correctitud, calidad, entre otros factores de gran importancia.

3.2 Implementación

La implementación se inicia a partir de los resultados obtenidos en el diseño y se implementa el sistema en términos de componentes. Estos conforman lo que se conoce como un modelo de implementación al describir los que se van a construir, su organización y dependencia entre nodos físicos en los que funcionará la aplicación.

3.2.1 Diagrama de Componentes

El diagrama de componente ilustra los elementos que serán usados para construir el sistema. UML define cinco estereotipos estándar que se le aplican a los mismos, como son: ejecutable, bibliotecas, tabla, archivo y documento. Los distintos componentes se pueden agrupar en paquetes según un criterio lógico y con vista a simplificar su implementación.

En el siguiente diagrama se muestra la relación existente entre el componente SimuladorAD que es un ejecutable y depende del paquete presentación, este a su vez depende del paquete entidad y del paquete negocio y finalmente este último del paquete entidad.

Los demás diagramas de componentes se encuentran en el documento modelo de implementación.

Capítulo 3: Implementación y Validación

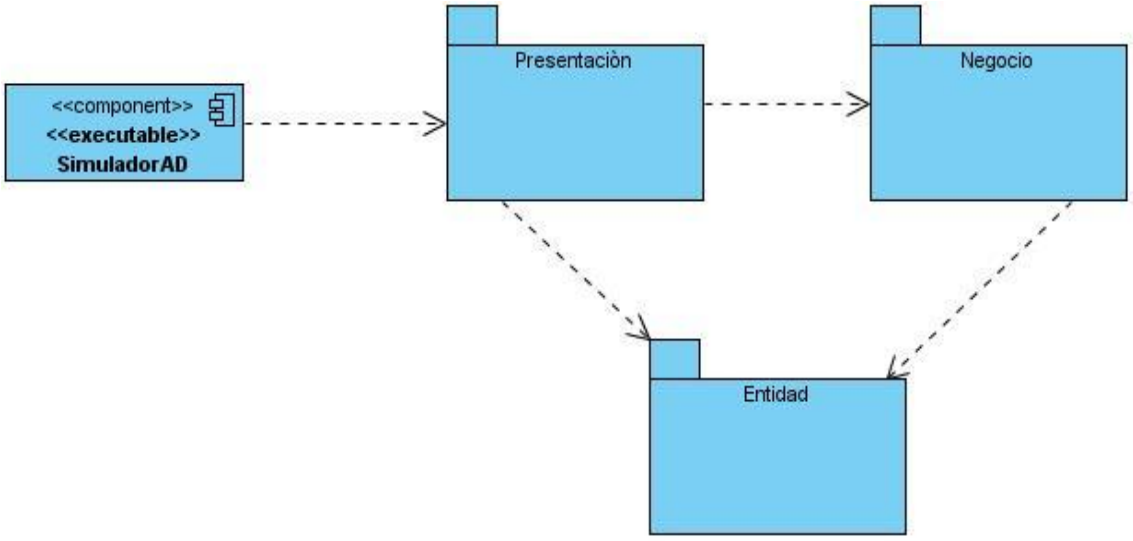


Figura 16: Diagrama de componente general

3.3 Validación de la Solución 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, 2008)

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

3.3.1 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

22 = 12 + 10

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

0.95 = 21/22

Donde:

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** = 0.95, esto demuestra que los requisitos se encuentran con un alto nivel de especificidad.

Para la revisión de la métrica aplicada a los requisitos se escogió a la ingeniera Yelena Hernández Estrada, analista principal del proyecto Sistema de Gestión Fiscal y a la ingeniera Ginisleisy Morales Gómez: Jefa del grupo de calidad en el mismo proyecto, puesto que por el rol que desempeñan poseen conocimiento en el tema y experiencia. Además no se hizo necesario escoger una población de revisores más grande debido a que la cantidad de requisitos es pequeña.

3.3.2 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.

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

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

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

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

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

Primera Revisión:

Atributo	Factor	Métrica asociada	Valor
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes 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: 1 Se presenta un 25%. Número de casos de uso que tienen requisitos omitidos: 1 Se presenta un 25%.
	Factor 4. ¿Todos los casos de uso han sido clasificados de acuerdo a su relevancia en (crítico, secundario, auxiliar, opcional)?	Métrica 5. Número casos de que no han sido clasificados.	Número casos de que no han sido clasificados: 0 Se presenta un 100%.
			Se presenta un: 87.5%
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.	Número de casos de uso que tienen un nombre incorrecto:1 Se presenta un 25%.
	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: 3. Se presenta un 75%.

	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: 1 Se presenta un 25%.
	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: 56.25%
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: 2 Se presenta un 50%.
	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: 0 Se presenta un 100%.

		sistema.	
	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: 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.	Número de casos de uso que deben ser modificados para mejorar el proceso actual: 1 Se presenta un 25%.
			Se presenta un: 56.25%
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: 1 Se presenta un 25%.
			Se presenta un: 25%

Métricas para validar diagrama de caso de uso del sistema

Segunda Revisión:

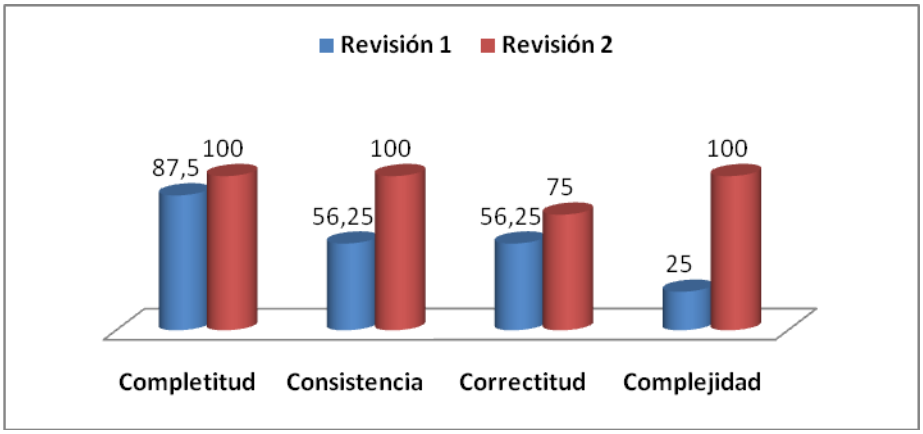
Atributo	Factor	Métrica asociada	Valor
Completitud	Factor 1. ¿Han sido definidos todos los roles relevantes de usuario encargados de	Métrica 1: Número de roles relevantes omitidos.	Número de roles relevantes omitidos: 0

	generar/ modificar o consultar información?		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 casos de que no han sido clasificados.	Número casos de que no han sido clasificados: 0 Se presenta un 100%.
			Se presenta un: 100%
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.	Número de casos de uso 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: 2 Se presenta un 50%.

	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.	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	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.	Número de casos de uso que deben ser modificados para mejorar el proceso actual: 0 Se presenta un 100%.
			Se presenta un: 75%
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%

Gráfica de Factores de Métrica



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.

Revisor: Ing. Yelena Hernández Estrada

3.3.3 Método de prueba por caja negra

Las pruebas de **caja negra** (Visual Paradimg for UML, 2010) son pruebas unitarias, consiste en ir probando uno a uno los diferentes módulos que constituyen una aplicación. Se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. Esta prueba examina algunos aspectos del modelo, fundamentalmente del sistema, sin tener mucho en cuenta la estructura interna del software. Se centran principalmente en los requisitos funcionales del software. Estas

pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

Diseño de caso de prueba para el caso de uso "Inicializar Memoria Física": Sección: "Técnicas de Asignación".

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad
S1: Inicializar la memoria física	EC 1.1: Inicializar la memoria física	1.1 El sistema muestra la interfaz con las opciones siguientes para escoger con cuál va a simular y así inicializar la memoria. • Técnicas de asignación • Algoritmos de reemplazo • Salir 1.2 El usuario selecciona la opción deseada.

S 1.1: Técnicas de asignación	EC 1.2 Técnicas de asignación	1.2.1 El sistema muestra la interfaz técnicas de asignación de memoria y activa la opción: 1.2.1.1 Tamaño de la memoria. 1.2.2 El usuario escoge el tamaño que va a tener la memoria. Se presiona la opción aceptar. 1.2.3 El sistema muestra el tamaño de memoria. 1.2.4 El sistema activa la opción procesos en memoria con los datos siguientes : 1.2.4.1 Procesos en memoria. 1.2.4. 2 Posición inicial 1.2.4. 3 Tamaño Proceso 1.2.5 El usuario escoge el id del proceso. 1.2.6 El usuario introduce los datos correspondientes. Se presiona la opción añadir. 1.2.7 El sistema muestra la memoria con cada proceso que se añadió a ella. 1.2.8 El usuario presiona la opción inicializar memoria. Termina el caso de uso.

	EC 1.3 Especificar tamaño de la memoria	1.3.1 El sistema en caso de que el campo donde se especifica el tamaño de la memoria esté vacío, muestra el siguiente mensaje de error: “Debe especificar el tamaño de la memoria”.
	EC 1.4 Escoger proceso	1.4.1 El sistema en caso de que no se haya escogido ningún proceso muestra el siguiente mensaje de error: “Debe seleccionar un proceso”.
	EC 1.5 Llenar todos los campos	1.5.1 El sistema en caso de que existan campos vacíos muestra el siguiente mensaje de error: “Debe llenar todos los campos”.
	EC 1.6 Entrar los datos correctos	1.6.1 El sistema en caso de que los datos entrados sean incorrectos muestra el siguiente mensaje de error: “Lo campos deben contener valores enteros”.

SC 1: Técnicas de asignación

Id del escenario	**Escenario**	**Tamaño de memoria**	**Var 2** *()*	**Var 3** *()*	**Respuesta del Sistema**	**Resultado de la Prueba**
EC 1.2	Técnicas de asignación	*I*			El sistema emite un mensaje para que llene los campos obligatorios.	Que los datos introducidos sean correctos.
		V				

		Id Proceso	**Posición inicial**	**Tamaño Proces**	El sistema emite un mensaje para que llene los campos obligatorios.	Que los datos introducidos sean correctos.
		I	*V*	*V*		
		V	*I*	*V*		
		V	*V*	*I*		
		V	*V*	*V*		

Los demás casos de pruebas realizados se encuentran en el artefacto diseño de casos de pruebas.

Conclusiones parciales

En este capítulo, con la realización de los diagramas de componentes quedó conformado el modelo de implementación de la aplicación, además se realizan pruebas de caja negra donde se verifican que los resultados esperados ocurran cuando se usen datos válidos y que sean desplegados los mensajes apropiados de error cuando se usa datos inválidos, también con la aplicación de métricas a los requisitos y casos de uso se pudo comprobar la validez de estos y su calidad.

Conclusiones

- Mediante el estudio de las funcionalidades de los diferentes laboratorios virtuales y simuladores en otros sistemas en el mundo, se contribuyó a elevar el nivel de comprensión de los desarrolladores y a determinar los objetivos y funcionalidades necesarias para el sistema a construir.
- Se investigó sobre las posibles herramientas y metodologías para el desarrollo de la aplicación seleccionándose el lenguaje de desarrollo Java con el IDE de desarrollo NetBeans, y como metodología de desarrollo se seleccionó RUP unido al lenguaje de modelado UML y BPMN.
- Se realizó el análisis, diseño y la implementación del producto Simulador para la Asignatura Sistemas Operativos, obteniendo la documentación necesaria para garantizar el futuro mantenimiento del sistema.
- Se obtuvo un producto interactivo que cumple con los requisitos establecidos.
- La aplicación de métricas propuestas por varios autores y pruebas realizadas al sistema posibilitó validar los resultados obtenidos.
- Con la realización de esta aplicación, en la Universidad de las Ciencias Informáticas aumentan los materiales interactivos que sirven de apoyo para impartir los temas de Administración de Memoria en la asignatura Sistemas Operativos.
- Se cumplió el objetivo general trazado para este Trabajo de Diploma: desarrollar un simulador que apoye la visualización de los contenidos del tema Administración de Memoria en la asignatura Sistema Operativo.

Recomendaciones

Tomando en cuenta los resultados obtenidos:

- Se recomienda incluir funcionalidades que permitan guardar en ficheros los ejercicios que se simulan y su posterior recuperación.
- Se recomienda terminar la implementación de los restantes algoritmos de reemplazo.

Bibliografía

Almaguer, Anileidy Basso Mesa y Dayaima Gómez. 2009. Sistema automatizado para el proceso de caracterización de los estudiantes. Ciudad de la Habana : s.n., 2009.

Barriento, Manuel Sánchez. 2008. BPMN desventajas . *BPMN desventajas .* [En línea] 2 de noviembre de 2008. [Citado el: 9 de febrero de 2010.]

http://www.aprendergratis.com/stag/bpmn-desventajas.html.

Borges., Rasiel Aponcio. 2009. Migración de la Capa de Acceso a Datos del. Ciudad de la Habana : s.n., 2009.

Cardenas, Marina Elizabeth. 2009. Software de Simulación aplicado a entornos de e-learning. 2009.

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

Ediciones Visual Studio 2005 Standard y Visual Studio 2005 Express. *Ediciones Visual Studio 2005 Standard y Visual Studio 2005 Express.* [En línea] [Citado el: 11 de febrero de 2010.]

http://www.microsoft.com/spanish/msdn/vs2005/editions/stdexp/default.mspx.

Estrada, Julián Monge Nájera y Víctor Hugo Méndez. 2007. Ventajas y desventajas de usar laboratorios virtuales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración. *Ventajas y desventajas de usar laboratorios virtuales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración.* [En línea] 2007. [Citado el: 16 de febrero de 2010.] http://www.latindex.ucr.ac.cr/edu31-1/edu-31-1-05.pdf.

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

Giraldo, L. &. (2005). Herramientas de Desarrollo de Ingeniería de SW para Linux.

Bibliografía

Gonzalez, Jesus Alberto Silva. 2009. Tarea de Java. *Tarea de Java.* [En línea] 20 de 4 de 2009. [Citado el: 10 de febrero de 2010.]

Gutiérrez, José Manuel Ruiz. 2000. La Simulación como Instrumento de Aprendizaje. *La Simulación como Instrumento de Aprendizaje.* [En línea] 2000. [Citado el: 5 de febrero de 2010.]

http://mami.uclm.es/jmruiz/materiales/Documentos/simulacion.PDF.

Herías, Francisco Andrés Candelas. 2003. Propuesta de Portal de la Red de. *Propuesta de Portal de la Red de Laboratorios Virtuales y Remotos de CEA.* [En línea] 27 de noviembre de 2003. [Citado el: 16 de febrero de 2010.]

http://www.disc.ua.es/docenweb/Docs/PropuestaDePortal.pdf.

Herreros, L. Rosado y J. R. 2005. Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física. *Nuevas aportaciones didácticas de los laboratorios virtuales y remotos en la enseñanza de la Física.* [En línea] 2005. [Citado el: 16 de febrero de 2010.] http://www.formatex.org/micte2005/286.pdf.

James P. Vary. [En línea] [Citado el: 10 de febrero de 2010.]

http://unesdoc.unesco.org/images/0011/001191/119102s.pdf.

Larman, C. (1999).*UML y Patrones. Introducción al análisis y diseño orientado a objetos.* México: Primera Edición.

Orallo, Enrique Hernández. 2007. El lenguaje Unificado de Modelado (UML). *El lenguaje Unificado de Modelado (UML).* [En línea] 2007. [Citado el: 12 de febrero de 2010.]

Pelegrino, Rodolfo González. 2009. Aplicación para el monitoreo de ventanas y notificación de tareas. Ciudad de la Habana : s.n., 2009.

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

Bibliografía

Pressman, R. S. 2005. *Ingeniería de software. Un enfoque práctico. Parte ll.* Ciudad de la Habana: Félix Varela.

Pressman, R. *Ingeniería del Software. Un enfoque práctico.* La Habana: Félix Varela, 2005

Rivas, Lornel A. Herramientas de Desarrollo de Software: Hacia la Contruccion de una Ontoloía. *Herramientas de Desarrollo de Software: Hacia la Contruccion de una Ontología.* [En línea] [Citado el: 10 de febrero de 2010.]

http://www.lisi.usb.ve/publicaciones/05%20herramientas/herramientas_25.pdf.

Sanchez, María A. Mendoza. 2004. Metodologías De Desarrollo De Software. *Metodologías De Desarrollo De Software.* [En línea] 7 de junio de 2004. [Citado el: 11 de febrero de 2010.]

http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.htm.

Vary, James P. 2000. Informe de la reunión de expertos. *Informe de la reunión de expertos.* [En línea] 2000. [Citado el: 10 de febrero de 2010.]

http://unesdoc.unesco.org/images/0011/001191/119102s.pdf.

Ventajas y desventajas de usar laboratorios Vituales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración. . *Ventajas y desventajas de usar laboratorios Vituales en educación a distancia: la opinión del estudiantado en un proyecto de seis años de duración.* [En línea] 2007. [Citado el: 16 de febrero de 2010.]

http://www.latindex.ucr.ac.cr/edu31-1/edu-31-1-05.pdf.

Visual Paradigm for UML (ME) .[En línea] 5 de marzo de 2007. [Citado el: 11 de febrero de 2010.]

http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

Bibliografía

2008. Foro de Java. *Foro de Java.* [En línea] 13 de junio de 2008. [Citado el: 18 de marzo de 2010.] http://www.forodejava.com/showthread.

Caceres, Janet Carreño. 2007.ANÁLISIS DEL SISTEMA "COLISEO VIRTUAL". Cuidad de la Habana : s.n., 2007.

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

Welicki, L. (s.f.). El Patrón Singleton. Recuperado el marzo de 25 de 2010, de El Patrón Singleton .

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

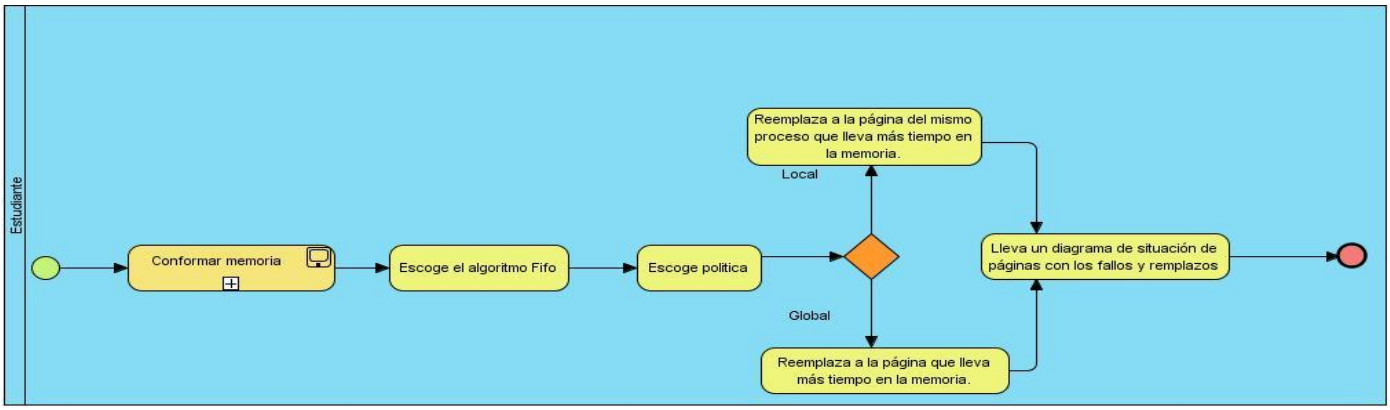
Es un aparato que permite la simulación de un sistema, reproduciendo su comportamiento. Los simuladores reproducen sensaciones que en realidad no están sucediendo.

Sistema Operativo

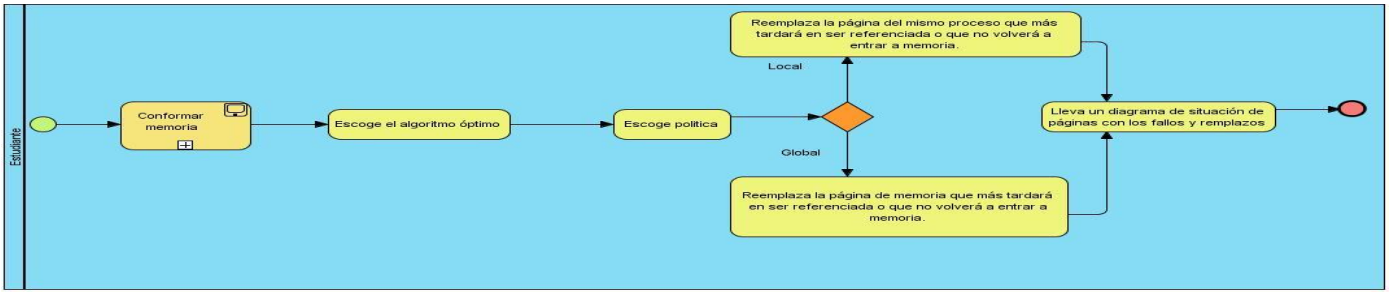
Programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permite la normal ejecución del resto de las operaciones.

Anexos

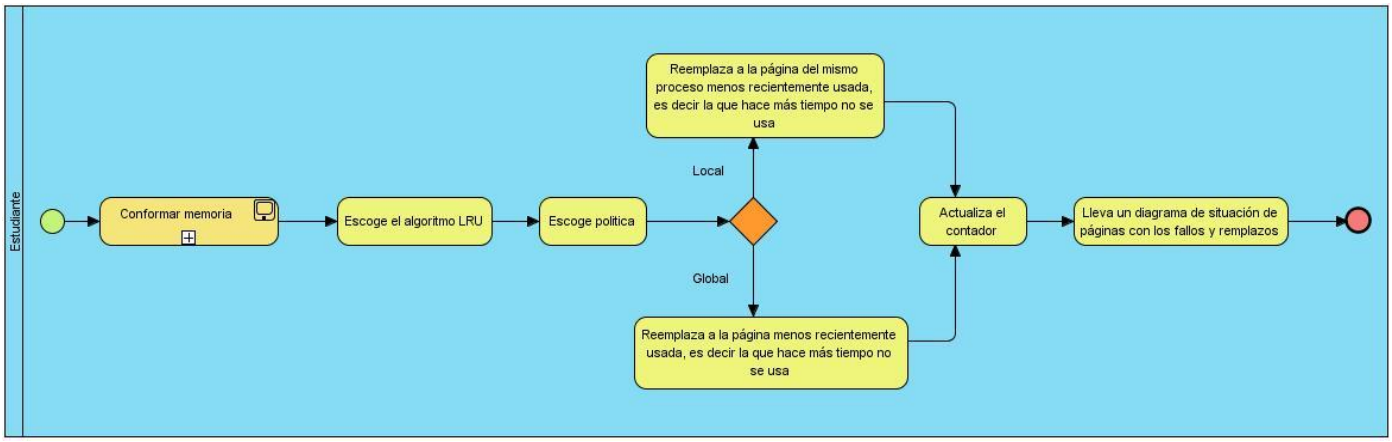
Anexo 1: Diagrama de proceso del algoritmo FIFO.



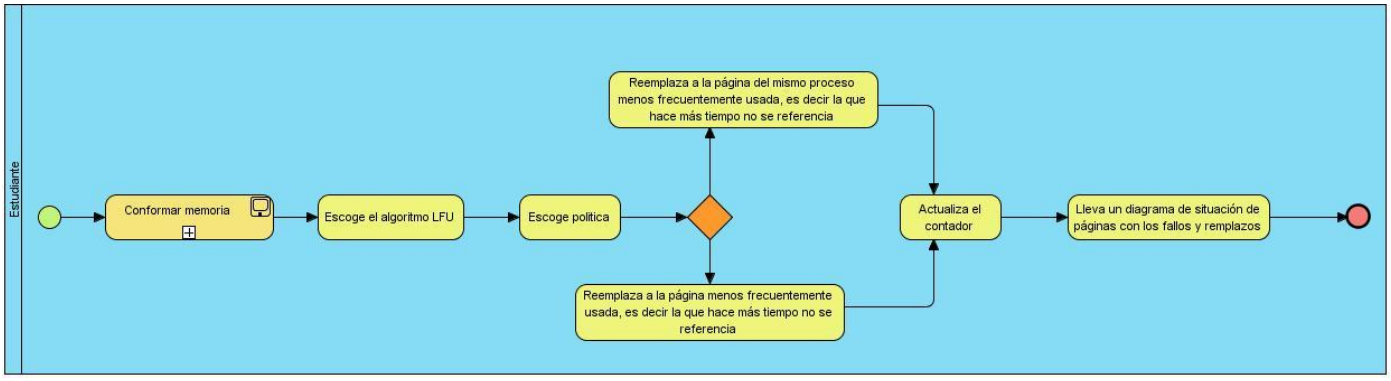
Anexo 2: Diagrama de proceso del algoritmo Óptimo.



Anexo 3: Diagrama de proceso del algoritmo LRU.

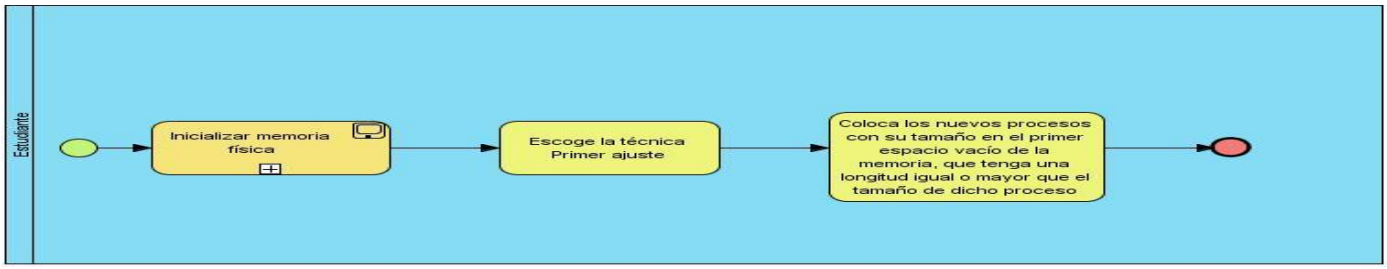


Anexo 4: Diagrama de proceso del algoritmo LFU.

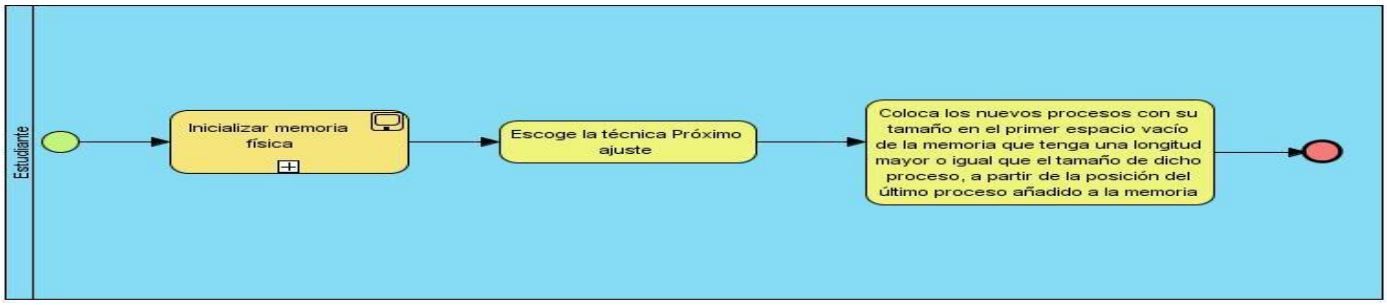


Anexo 5: Diagrama de proceso del algoritmo Mejor ajuste.

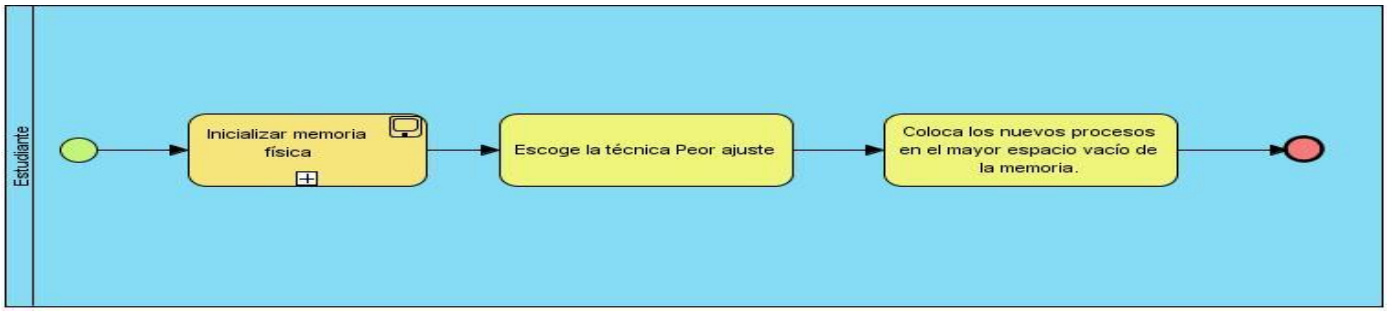
Anexo 6: Diagrama de proceso del algoritmo Primer ajuste.



Anexo 6: Diagrama de proceso del algoritmo Próximo ajuste.



Anexo 7: Diagrama de proceso del algoritmo Peor ajuste.



UCi
Universidad de las Ciencias Informáticas

Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Martes, 8 de junio de 2010.

Luego de haber efectuado 3 iteraciones de revisiones a los artefactos: Especificación de Requisitos, Modelo de Sistema, Modelo de diseño, Modelo de procesos con BPMN del Simulador para la asignatura Sistemas Operativos y haberse detectado un promedio de 31 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que queda liberada la aplicación.

Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarez

UCi Universidad de las Ciencias Informáticas
Decanato Facultad