Universidad de las Ciencias Informáticas
Facultad 3

Título: Análisis y Diseño del módulo de Ejercicios de Planificación de Disco para el Laboratorio Virtual de Sistemas Operativos.

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

Autores: Aniuska Vidaurreta Gómez

Tutor: Ing. Yordanis Milanés Zamora.

La gente piensa que el diseño es estilo.

El diseño no es el estilo. . .

El buen diseño es una actitud renacentista que combina la tecnología, la ciencia cognitiva, la necesidad humana y la belleza de producir algo que el mundo no sabía que estaba desaparecido.

Paola Antonelli. . . . . .

DECLARACION DE AUTORÍA

Declaro ser autora de la presente tesis y reconocer 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 _____.

_____ _____

Aniuska Vidaurreta Gómez Yordanis Milanés Zamora

Autor Tutor

AGRADECIMIENTOS

Quiero agradecerle ante todo a mi familia por confiar tanto en mí y sentirse tan orgullosos "Ya tienen una Ingeniera". A mi mamá por quererme tanto y darme siempre todo el apoyo que necesitaba para continuar esforzándome y lograr sobre todo lo que soy hoy en día tanto profesional como personal. A mi papá por la educación y los valores que me inculcó después de todo valió la pena. A mis tíos y tías muchas gracias por ayudarme tanto, ustedes agregaron un granito de arena en mi formación y a mi hermanito que quiero mucho. Quiero agradecerle a todas mis amistades que más que amigos son los hermanos que he tenido durante estos 5 años inolvidables en mi vida sin mencionar nombres para no olvidar alguno. A mi novio le doy mil gracias por haberme ayudado tanto todos estos años y darme el cariño que necesitaba creo que sin ti no hubiese llegado hasta aquí, a mis amigos y segundos tutores Aldrin, Manuel, Rodolfo y los Robertos, muchas gracias por contribuir en mi tesis y soportarme mientras trabajaban en la suya. A la profesora Lien y a Víctor muchas gracias, sin usted no hubiese sido posible graduarme. A la profesora Yiset y Ana Maris gracias por atenderme cuando más lo necesitaba, a Lisandra muchas gracias por ayudarme con la tesis la verdad me fue de mucha ayuda. A mis primas y primos y en especial a mi primita Yaselis que se preocupó más que yo por la tesis y se molestaba cuando no le decía de ella. A mis abuelas muchas gracias por su amor incondicional y a mis abuelitos que aunque no están físicamente y no pudieron ver en quien se convirtió su nieta siempre los llevo en mi corazón. A la oponente gracias por todas las recomendaciones que me ayudaron mucho al igual que los restantes del tribunal gracias por darme una oportunidad. También le

doy gracias al profe Yordanis por asumir el cargo de tutor en esta tesis cuando no tenía a nadie y defenderme cuando lo necesita, disculpe por las molestias ocasionadas. No quisiera dejar de mencionar y no por ser lo último lo menos importante un eterno agradecimiento a la Revolución por darme la oportunidad de forjarme como ingeniera en ciencias informáticas y en especial al Comandante en Jefe Fidel Castro por crear esta universidad de excelencia. A todos eternamente agradecida, y en caso de que se me quede alguien por favor que acepten mis disculpas.

DEDICATORIA

Dedico esta tesis a mi mamá, espero que estés orgullosa de tu niñita, esta tesis es para ti, gracias por tu cariño incondicional, por apoyarme y darme siempre lo que ha estado a tu alcance. Te quiero mucho.

A mis abuelos que tanto quiero en especial a mis dos abuelitos que aunque no estén presentes se que se sentirían orgullosos de su nieta, en especial a mi abuelo Manolo que se que murió sin despedirse de mí y que era lo más tenía presente cuando me encontraba a miles de kilómetros en la hermana República de Venezuela y me fue difícil su pérdida, lo siento mucho y espero que donde quiera que te encuentres sepa que siempre estás presente.

RESUMEN

La Universidad de las Ciencias Informáticas (UCI) se encuentra inmersa en profundos cambios para ayudar a elevar la calidad del proceso de enseñanza-aprendizaje, implementando el nuevo modelo de formación docencia-producción-investigación, donde la semipresencialidad y la amplia y adecuada interacción con las Tecnologías de la Información y las Comunicaciones (TICs) representan los componentes principales para esta integración. Es por ello que se ha comenzado a desarrollar un laboratorio virtual que abarcará todos los temas de la misma, desde el punto de vista didáctico. Como objeto de estudio se toma la asignatura de Sistemas Operativos impartida en el tercer año de la carrera. Su enseñanza se torna compleja, debido al alto nivel de abstracción que requiere por parte de los estudiantes para adquirir un buen entendimiento del funcionamiento interno sobre cada administrador involucrado como el Planificador de Disco.

Por este motivo el presente trabajo se enmarca en desarrollar el análisis y el diseño de una herramienta para evaluar a los usuarios a través de los ejercicios correspondientes al contenido de Planificación de Disco comprobando los conocimientos adquiridos en los módulos de auto aprendizaje y permitiéndole al profesor una medida del estado del aprendizaje del estudiante con vistas a la toma decisiones respecto a los pasos a seguir para lograr un alto nivel de conocimientos de los temas.

Con este fin se realizó un estudio que abarcará definiciones y el desarrollo de algunos laboratorio virtuales para una mejor comprensión. Se define la metodología, herramientas de modelado y herramienta CASE entre otros aspectos relacionados con la ingeniería y se le aplican como conclusión las métricas para evaluar la calidad de los requisitos del sistema y del diseño donde se obtuvieron resultados satisfactorios.

PALABRAS CLAVES:

Laboratorio Virtual, Planificación de Disco, Sistemas Operativos, Tecnologías.

TABLA DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS.

INTRODUCCIÓN

La informática se ha convertido en uno de los principales aportes que ha dado el hombre y es por ello que con el aumento de su conocimiento ha ido evolucionando y al mismo tiempo surgiendo medios que le aportaron creatividad como las Tecnologías de la Información y las Comunicaciones (TIC), dándole un nuevo giro al mundo, proporcionando oportunidades en el desarrollo científico. Uno de sus aportes facilitó una revolución al proceso enseñanza-aprendizaje generando nuevas vías para la educación a distancia.
Durante un largo período fueron apareciendo disímiles sistemas informáticos y definiciones que fueron dando a luz a una nueva tendencia "los laboratorios virtuales", que con el advenimiento de las comunicaciones digitales de alta velocidad conjuntamente con los costos decrecientes de los servicios fijos, han permitido su desarrollo con mayor eficiencia y menor movimiento de personal, sin embargo, no se considera que vaya a suplantar a los verdaderos laboratorios ni competir con ellos, pero si abren nuevas perspectivas que eran poco posibles de explorar dentro de estos a un precio asequible.

Cuba no se encuentra ajena a las nuevas tendencias provocadas por las TICs y en función de insertarse como productores de software ha ido vinculando las ramas: económicas, políticas y sociales al mundo informático, consideradas de interés primordial para el Estado Cubano, no solo por los beneficios que trae desde el punto de vista del desarrollo de sistemas para el uso interno, sino también con el objetivo de introducirse en el mercado a escala mundial aprovechando sus perspectivas económicas. A raíz de lo anterior se creó la Universidad de las Ciencias Informáticas (UCI), la cual se encuentra inmersa en profundos cambios para ayudar a elevar la calidad del proceso de enseñanza-aprendizaje, implementando el nuevo modelo de formación docencia-producción-investigación, permitiendo a los estudiantes responder a las exigencias actuales demandadas por la dirección del país.
Los Sistemas Operativos (SO) surgen para facilitar la interacción entre persona y computadora y es por ello que se manifiesta la necesidad de estudiarse como parte de una asignatura. La misma es impartida en la UCI dentro del Programa Analítico del (P1) formando parte del conjunto de disciplinas brindadas en el tercer año de la carrera. Luego de un análisis detallado se observó que carece de una adecuada integración con las TICs provocando que las actividades sean mayormente presenciales, entrando en contradicción con el nuevo modelo de formación, donde la semipresencialidad y la amplia y adecuada

integración con las TIC son los componentes principales para lograr la unificación (docencia – producción - investigación).

La asignatura se divide en tres temas fundamentales: proceso, memoria, entrada/salida de la información, concerniente a la administración de recurso como función fundamental de un SO real. La enseñanza de estos temas se torna compleja debido al alto nivel de abstracción que requiere por parte de los estudiantes para adquirir un buen entendimiento del funcionamiento interno sobre cada administrador involucrado como el Planificador de Disco.

En virtud de satisfacer las necesidades existentes de materiales didácticos y sistemas que apoyen la enseñanza de la asignatura de SO, se ha comenzó a desarrollar un laboratorio virtual que abarcará todos los temas de la misma permitiendo acceso a las conferencias y prácticas simuladas de la materia, sin embargo dentro de su concepción, ninguna de las aplicaciones proporcionan al alumno elementos pedagógicos que les permita evaluarse y poner en práctica los conocimientos impartidos en clases específicamente en la planificación de disco, resultando engorroso el proceso de aprendizaje.

De esta forma surge el **problema a resolver**: ¿Cómo transformar las necesidades de modelación de una herramienta que permita evaluar a los estudiantes de la asignatura de Sistema Operativo en el módulo de Ejercicios de Planificación de Disco, a un lenguaje entendible para el programador, permitiendo una posterior su implementación?

Por tanto el **objeto de estudio** corresponde a: Proceso de desarrollo de software en laboratorios virtuales.

Como **objetivo general:** Desarrollar los modelos de análisis y diseño del módulo de Ejercicios de Planificación de Disco para el Laboratorio Virtual de Sistema Operativos en un lenguaje entendible para el programador, de manera que permita una posterior implementación.

A partir de lo planteado se puede delimitar como **campo de acción** de la investigación a: Las fases de análisis y diseño dentro del desarrollo del software.

Como **idea a defender** se plantea que con la realización del análisis y diseño del módulo de Ejercicios de Planificación de Disco para el Laboratorio Virtual de Sistemas Operativos para evaluar a los estudiantes, se podrá lograr una posterior implementación.

Para dar cumplimiento al objetivo general se tienen como **objetivos específicos:**

- Elaborar el marco teórico de la investigación referente a los Laboratorios Virtuales.
- Realizar los modelos de análisis y diseño.
- Validación los artefactos obtenidos.

Para dar cumplimiento del objetivo propuesto se plantean las siguientes **tareas de la investigación:**

- Elaborar el marco teórico de la investigación.
- Realizar un diagnóstico de la situación actual para identificar las principales necesidades del sistema en cuestión.
- Obtener la especificación de los requisitos Funcionales y No Funcionales.
- Obtener los artefactos generados en la fase de análisis.
- Obtener los artefactos generados en la fase de diseño.
- Validar los artefactos obtenidos en la fase de análisis y diseño.

Durante la elaboración del presente trabajo de diploma se visualizan tres capítulos y se encuentra estructurado de la siguiente manera:

Capítulo 1: Enmarca la Fundamentación Teórica de la investigación, donde se abordar lo referente a los Laboratorios Virtuales, dígase definiciones, clasificaciones y algunos de los usados tanto a nivel nacional como internacional. Se hace un estudio breve referente a los tipos de simulación además de un análisis de las herramientas y tecnologías que serán usadas para dar solución al problema de la investigación, además de un estudio de los patrones de casos de uno para obtener una mayor comprensión de los mismos a la hora de dar paso a la solución de propuestas.

Capítulo 2: Se Describe la solución, da comienzo con un mapa conceptual encargado de ilustrar las relaciones de los contenidos a tener en cuenta para la realización del evaluador de ejercicios de planificación de disco. Se especifican los requisitos funcionales y no funcionales por los que se regirá el sistema propuesto dando paso a algunos artefactos presentados por la metodología definida y logrando a través del uso de patrones un correcto diseño.

Capítulo 3: Se Valida la solución, se basa en la comprobación detallada de los artefactos generados durante la descripción de la solución, considerada como la etapa final de la actual investigación aplicando a través de las métricas la validez del sistema propuesto.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA.

1.1 Introducción

Enmarca la Fundamentación Teórica de la investigación, donde se abordar lo referente a los Laboratorios Virtuales, dígase definiciones, clasificaciones y algunos de los usados tanto a nivel nacional como internacional. Se hace un estudio breve referente a los tipos de simulación además de un análisis de las herramientas y tecnologías que serán usadas para dar solución al problema de la investigación, además de un estudio de los patrones de casos de uno para obtener una mayor comprensión de los mismos a la hora de dar paso a la solución de propuestas.

1.2 Laboratorios Virtuales.

Los laboratorios virtuales (LV) han tenido varios conceptos a lo largo de todo su desarrollo por diversos autores que han tratado de aportar un nuevo enfoque a esta definición. Un grupo de expertos sobre este tema con el apoyo de la UNESCO preparado por Jame P. Vary en el 2000 plantearon que un LV no es más que:

"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".(Ames, Lowa, 2011)

Luego del estudio dado de algunos de los autores se puede definir que un laboratorio virtual es una manera de solucionar problemas a aquellas personas que carecen de un medio para poner en práctica sus conocimientos, simulando un entorno controlado que en ocasiones puede ser experimentado a distancia ahorrando tiempo y costo.

1.2.1 Tipos de Laboratorios Virtuales

- Laboratorios Virtuales de Software.

Son laboratorios virtuales desarrollados como un programa de software independiente, destinado a ejecutarse en la máquina del usuario, y cuyo servicio no requiere de un servidor Web. Los programas son con instalación propia que pueden estar destinados a plataformas Unix, Linux, M.S. Windows e incluso

necesitar que otros componente de software estén instalados previamente, pero que no necesitan 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 (HERÍAS, 2003).
A pesar de ser uno de los más usados en el mundo se considera que no es factible para la concepción de la presente investigación puesto que al trabajar como software independiente se hace un poco difícil incluirlo dentro de la plataforma Moodle a la par de los demás módulos.

- Laboratorios Virtuales Web.

Se basan en un software que depende de los recursos de un servidor determinado. Esos recursos pueden ser determinadas bases de datos, software que requiere ejecutarse en su servidor o la exigencia de determinado hardware para realizarse. Estos no son programas que un usuario pueda descargar en su computadora para ejecutar localmente de forma independiente (HERÍAS, 2003).

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

- Laboratorios Remotos.

Se trata de laboratorios 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 laboratorios requieren de 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 hace dependientes estos laboratorios de sus servidores es la habitual gestión de usuarios en el servidor (HERÍAS, 2003).

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

1.2.2 Laboratorios Virtuales en el Mundo

El desarrollo de estos laboratorios ha sido desplegado alrededor de todo el mundo, unos más sofisticados que otros, debido a las necesidades que presente cada país pero de una forma u otra tratan de llegar a su objetivo final. Dentro de los encontrados se tiene el ¨ Laboratorio Virtual de Computación para Cursos en Línea¨, en la Universidad de Baja California para cursos de licenciatura y posgrado en el área de Ingeniería de Software, trabaja con el sistema de Moodle y demás recursos Web para el desarrollo de aplicaciones para internet. En él se explica el uso de herramientas de software basado en código abierto y de libre distribución de tipo GNU\Linux, de forma tal que alimente los esquemas educativos a distancia en modalidades en línea como la semipresencialidad o las videoconferencias.

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

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

- Entorno virtual de Física**,** de la Universidad de las Ciencias Informáticas, la misma consta de 8 prácticas de LV para el estudio de algunos temas de la asignatura.

- SIDEF (Sistema Interactivo Didáctico de Enseñanza de la Física), utilizado en el Departamento de Física de la Facultad de Matemática, Física y Computación de la Universidad Central "Marta Abreu" de Las Villas.

- Laboratorio Virtual de Química General, de la Universidad de La Habana con el cual se puede aprender Química desde su investigación hasta realizando prácticas virtuales.

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

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

- Debido al nuevo modelo de formación donde ya en el tercer año de la carrera la mayoría de los estudiantes están vinculados a proyectos productivos y las asignaturas son semipresenciales, se crea la necesidad de ejecutar las horas no lectivas y no presenciales, de forma tal que se aproveche el tiempo que con que cuentan, por tal motivo se hace necesario realizar una herramienta que les permita los alumnos poner en práctica sus conocimientos, específicamente en la asignatura de Sistemas Operativos generando un aumento en los resultados académicos.
- Desde el punto de vista económico el país no cuenta con mucho desarrollo referente a las últimas actualizaciones que brindan los laboratorios presenciales por lo que se han propuesto realizar práctica de LV de forma tal que mejoren los resultados de dicha disciplina.
- De los LV visitados se detectaron que aunque incluyen varias características en común ninguno reúne los requisitos que pretende la universidad.

- Algunos presentan como características que no necesitan de un servidor para su funcionamiento por lo que no se basa en los criterios que debe cumplir un LV web.
- Aunque algunos de los LV visitados están adaptados a la enseñanza ninguno está dedicado a los SO por lo que se hace necesario crear uno con las características independientes de la asignatura.
- La mayoría necesitan de internet y el país actualmente carece de medios que le permitan una adecuada conexión para la elaboración de estas prácticas virtuales, además que una vez realizada la simulación todo el contenido tratado será cerrado por lo que no favorece el aprendizaje, no cumpliendo con los objetivos que pretende la investigación.

1.3 Simulación.

Como tal la simulación es la manera de diseñar un modelo de un sistema real, guiado por términos de la experiencia, con la finalidad de comprender el comportamiento del sistema o de evaluar nuevas estrategias dentro de los límites impuestos por uno o varios criterios para su funcionamiento. Conducido por experimentos en una computadora digital, la misma comprende relaciones matemáticas y lógicas necesarias para describir la manifestación del mundo real.

1.3.1 Simulación por Computadoras

La Simulación es un intento de modelar situaciones de la vida real por medio de un programa por computadora de manera que logre comprender el comportamiento del sistema, ya sea por cambio de variables o quizás por predicciones hechas acerca de su funcionamiento.

Según Pressman: son muchos sistemas basados en computadora que interactúan con el mundo real en forma activa imponiendo control sobre las máquinas, procesos e incluso sobre la gente que genera eventos. Son basados en sistemas que trabajan en tiempo real y en otros que a menudo pertenecen a la categoría de sistemas reactivos.

Hoy en día se hacen maniobras quirúrgicas por un cirujano o un anestesiólogo sin que ocurran consecuencias fatales que se pueden presentar en la vida, o como aterrizar un avión helicóptero bajo

condiciones adversas, manipular el núcleo de una célula o simplemente aprender a jugar golf mediante la creación de los simuladores que desde un punto de vista ayudan a salvar vidas humanas ante peligros asociados a muchas actividades que nos ponen en riesgo cada día.
En el ámbito de la educación estos presentan un elevado potencial en el desarrollo de destrezas y habilidades encaminadas al aprendizaje de los campos del conocimiento humano como: la física, química, biología, ciencias sociales, administración, matemáticas, etc., a través de multimedia y escenarios educativos.

1. 4 Proceso de Desarrollo de Software

El proceso de software se define como un marco de trabajo para las tareas que se requieren en la construcción de un software de alta calidad. Según Ivar Jacobson, Grady Booch y James
Rumbaugh: ¨ Un proceso define quien está haciendo que, cuando y como se va a lograr cierta meta, con la misión de transformar los requerimientos del usuario en un producto de software; de manera que los integrantes del equipo y todo aquel que pueda estar interesado en el producto final, tenga la misma visión y no ocurra cuando no se aplica un proceso de desarrollo. Es la definición de un conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto. (Pressman, 2005).

1.4.1 Metodologías que se utilizan en el desarrollo de software.

La informática ha sido una más de los recientes aportes que el hombre ha llevado a sus manos ya sea para bien o para mal en dependencia de su uso. A raíz del acelerado desarrollo de las tecnologías, la elaboración del software se ha vuelto cada vez más compleja creando como resultado las Metodologías, con el fin de imponer un proceso disciplinado sobre el desarrollo del software, haciendo de este cada vez más predecibles y eficiente. Forman parte de un conjunto de procedimientos, técnicas, herramientas y un soporte documental, que ayuda a los desarrolladores a realizar un nuevo software. Puede seguir uno o varios modelos de ciclo de vida, sin embargo este indica qué es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo.
Una metodología está compuesta por:

- Cómo dividir un proyecto en etapas.
- Qué tareas se llevan a cabo en cada etapa.
- Qué restricciones deben aplicarse.
- Qué técnicas y herramientas se emplean.
- Cómo se controla y gestiona un proyecto. [Hernando 2004.]

1.4.1.1 Programación Extrema:

La Programación Extrema conocida también por sus siglas en ingles como Extreme Programming (XP) se ha convertido en una de las más conocidas y exitosas dentro de las metodologías ágiles. Utiliza un enfoque orientado a objeto como su paradigma de desarrollo preferido. Abarca un conjunto de reglas y prácticas que ocurren en el contexto de cuatro actividades del marco de trabajo: planificación, diseño, codificación y pruebas. (Pressman, 2005).

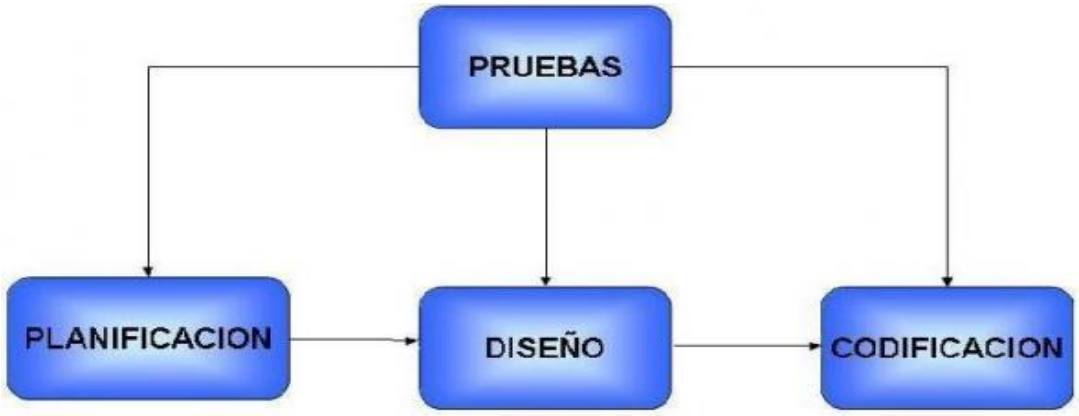


Figura: Marco de trabajo de XP

Se basa en la comunicación (entre el usuario y los desarrolladores), en la simplicidad (al desarrollar y codificar los módulos del sistema) y en la retroalimentación concreta y frecuente del equipo de desarrollo (el cliente y los usuarios finales).

1.4.1.2 Scrum

Se basa en el proceso iterativo e incremental y es utilizada comúnmente en entornos basados en el desarrollo ágil de software. Representa más que una metodología de desarrollo de software y se centra en entregar las funcionalidades de más valor para el cliente en el tiempo más corto posible, especialmente indicada para proyectos en entornos complejos donde es necesario obtener resultados lo más pronto posible y donde los requisitos son cambiantes o poco definidos. Utilizado principalmente cuando no se le está entregando al cliente lo que necesita, cuando las entregas son alargadas demasiado o los costos se disparan y la calidad no es aceptable.

En Scrum un proyecto que se ejecuta en bloques temporales cortos y fijos, donde cada iteración tiene que proporcionar un resultado completo, un incremento del producto final y que sea entregado al cliente cuando sea solicitado.

1.4.1.3 Proceso Unificado de Desarrollo de Software (RUP)

Un Proceso Unificado es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos.

RUP, es el resultado de varios años de desarrollo, define cuatro elementos, Trabajadores (roles), respondiendo a la pregunta ¿Quién?, Actividades, respondiendo a la pregunta ¿Cómo?, Artefactos (productos), que responden a la pregunta ¿Qué y los flujos de trabajo de las disciplinas que responde a la pregunta ¿Cuándo?

Figura 1: Relación entre roles, actividades, artefactos.

Está basado en componentes, lo que quiere decir que el sistema de software está formado por componentes de software, interconectados a través de interfaces bien definidas. Todo su proceso se resume en tres aspectos esenciales: dirigido por casos de uso, centrado en la arquitectura e iterativo e incremental. (I Jacobson, G Booch, J Rumbaugh, 2000)

Divide en 4 fases el desarrollo del software:

- Inicio, encargada de dar una descripción del negocio, determinando la visión del proyecto.
- Elaboración, encargada de determinar la arquitectura óptima.
- Construcción, se lleva a cabo la construcción del producto.
- Transmisión, su principal objetivo es llegar a obtener el release del proyecto, es decir, cubre el período donde el proyecto se convierte en versión beta, corrige los errores tras la entrega.

Cada una de estas etapas son desarrolladas mediante un ciclo de iteraciones y estas consisten en reproducir su ciclo de vida en cascada a menor escala. Estas iteraciones son llevadas a cabo bajo dos disciplinas (Flujo de trabajo).

Disciplina de Desarrollo:

- Modelamiento de Negocio: Encargado de describir todos los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requerimientos: Lleva las necesidades del negocio a un sistema automatizado.
- Análisis y Diseño: Encargado de transportar los requerimientos dentro de la arquitectura de software.
- Implementación: Es la que crea el software y hace que se ajuste a la arquitectura y que tenga el comportamiento deseado.
- Pruebas: Asegura que el comportamiento requerido sea el correcto y que cumpla con todo lo solicitado por el cliente.

Disciplina de Soporte:

- Administración de configuración y cambio: Encargada de guardar todas las versiones del proyecto.
- Administración del proyecto: Es quien administra los horarios y recursos.
- Ambiente: Se encarga de administrar el ambiente de desarrollo.

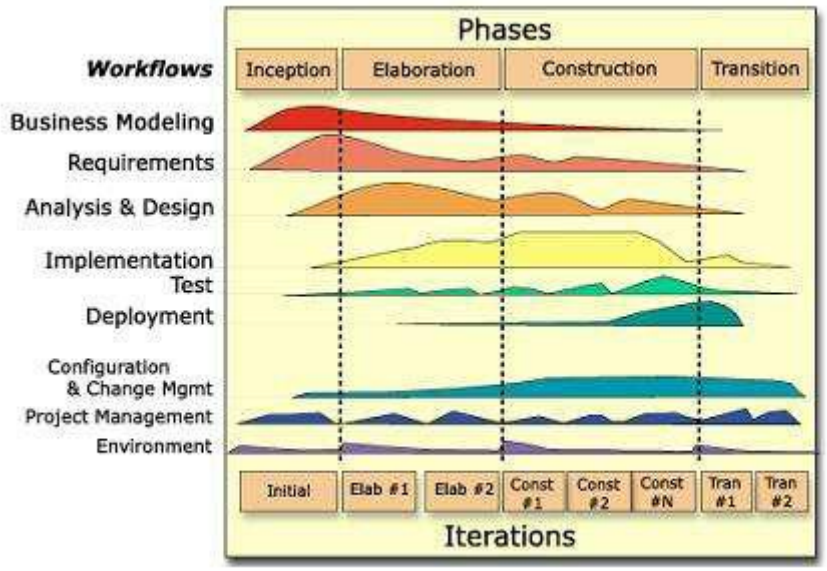


Figura 2: Fases e Iteraciones de la Metodología RUP

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.

Se basa en casos de uso para describir lo que se espera del software y está muy orientado a la arquitectura del sistema, documentándose lo mejor posible basándose en UML como herramienta principal. Es un proceso muy general y grande, por lo que antes de usarlo habrá que adaptarlo a las características de la empresa.

Luego de un estudio detallado de las metodologías presentadas se puede concluir que XP a pesar de sus ventajas como metodología ágil no es favorable su uso, ya que no se ajusta al proceso actual puesto que es centrada en la programación como su nombre lo indica y aplicada para proyectos a corto plazo, además de presentar como peculiaridad que incluye al cliente como parte del equipo de desarrollo del producto provocando cambios constantes en los requisitos creando un atraso durante el desarrollo del trabajo además de que produce una baja documentación dificultando el proceso de mantenimiento.

Referente a Scrum se puede decir que es una metodología que no está concebida como método independiente, es decir que, promueve como complemento otras metodologías incluidas como XP y RUP. Como método se enfatiza en valores y prácticas de gestión, sin pronunciarse sobre requerimientos, implementación y demás cuestiones técnicas deliberando a partir de ello insuficiencia y complementariedad, además de que al permitir la modificación de la lista de requisitos en cualquier momento del ciclo de vida del proyecto posibilita que el producto final posea inconsistencias provocando inconformidad a los usuarios finales. Otros de sus inconvenientes se deben a la pobre documentación comparada con otras metodologías constituyendo a una menor calidad del software y a una pobre mitigación de riesgos.

Por último se decidió luego de un estudio previo de dichas metodologías seleccionar a RUP ya que es considerada como una metodología muy bien organizada en fases y flujos, su pilar fundamental de desarrollo es generar los artefactos muy bien documentados. Es considerada una de las más importantes para alcanzar un grado de certificación en el desarrollo del software reduciendo el número de incidencias y por ende su costo total, además contar con un enfoque basado en modelos permitiendo un buen entendimiento entre clientes y desarrolladores facilitando un producto con altos niveles de calidad. Se tuvo en cuenta también que fue la seleccionada por el proyecto al cual tributa la presente investigación debido a sus dimensiones y necesidades de documentar cada fase de trabajo detalladamente. Además que el desarrollo del Laboratorio Virtual será ejecutado por fases donde no tienen que estar involucradas las demás personas y cada uno de los desarrolladores tendrán un rol específico constituyendo la principal ventaja que ofrece RUP.

1.5 Lenguajes de Modelado.

Los lenguajes de modelado representan un conjunto estandarizado de símbolos y sus relaciones, permitiendo expresar un Sistema Informático mediante un esquema teórico que representará posteriormente su diseño. En la actualidad el más conocido es el Lenguaje de Modelado (UML), especializado en especificar o describir métodos o procesos.

1.5.1 Lenguaje de Modelado Unificado (UML).

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. [I Jacobson, G Booch, J Rumbaugh, 2000.]
Compuesto por varios elementos gráficos que se combinan para conformar diagramas que tienen como finalidad mostrar las perspectivas de un sistema pero no como implementarlo, usable en cualquier metodología ya que es independiente del ciclo de desarrollo que se vaya a seguir.

1.6 Herramientas Case

Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador), no son más que herramientas de desarrollo de software con el objetivo de aumentar la productividad, reduciendo el costos en tiempo y dinero, brindando ayuda en todos los aspectos del ciclo de vida de desarrollo del software como a realizar un diseño en el proyecto, calcular los costos, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

1.6.1 Visual Paradigm

Visual Paradigm para UML**,** es una herramienta case, que en su totalidad soporta todo el ciclo de vida del desarrollo de un software y para más especificación, durante el análisis y diseño orientado a objetos, durante la construcción, pruebas y despliegue. Permite la obtención de todos los tipos de diagramas de clases, código inverso, genera código desde diagramas y genera documentación. Utiliza UML como lenguaje de modelaje con soporte multiplataforma que proporciona excelentes facilidades de interoperabilidad[1] con otras aplicaciones. Facilidad a la hora de ser instalado, actualizado y es compatible entre ediciones.
Ofrece una serie de funcionalidades como:

- Ingeniería inversa, código a modelo, código a diagrama.

1 [2] Interoperabilidad: Es la condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos.

- 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 base de datos, transformación de diagramas de Entidad-Relación en tablas de base de datos.
- Generador de informes para generación de documentación.
- Ingeniería inversa de bases de datos, desde Sistemas Gestores de Bases de Datos existentes a diagramas de Entidad-Relación.
- Distribución automática de las figuras y conectores de los diagramas UML.
- Modelo para realizar prototipos de interfaz. (Giraldo, 2005).

Además de señalar que es una herramienta con licencia comercial siendo de utilidad para la UCI ya que puede hacer uso de ella.

1.6.2 Rational Rose Enterprise Edition

Rational Rose Enterprise Edition, es una herramienta case diseñada para el desarrollo de la arquitectura de base del software. Elaborada por los creadores de UML y se encarga de cubrir 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.

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)

Resulta muy útil para los analistas que usan RUP con herramienta UML, sin embargo, no son compatibles con las propias tecnologías de implementación. Es un entorno de modelado que permite generar código a

partir de modelos Ada, ANSI C++, C++, CORBA, Java/J2EE, Visual C++ y Visual Basic ofreciendo un lenguaje de modelación común que agiliza la creación del software además de que presenta una necesidad de alta capacidad de procesamiento y de se necesita de tener habilidad y conocimiento de esta herramienta para su uso.

Para el diseño objeto de esta tesis, se seleccionó Visual Paradigm, ya que se encarga de generar toda la documentación cumpliendo con los estándares establecidos, soporta análisis textual y práctica para la captura de los requisitos del sistema además de ser disponible en múltiples plataformas. Soporta la metodología y el lenguaje de modelado seleccionado, es centrado en el diseño de casos de uso presentando también un enfoque al negocio garantizando la obtención de artefactos y un software final con mayor calidad. En la UCI se encuentra generalizado y cuenta con una comunidad de desarrollo que respalda su mantenimiento y actualizaciones, así como la solución de dudas en su uso.

1.7 Modelo del Sistema

El modelo del sistema realiza una descripción detallada de todas las funciones que debe realizar el sistema, incluyendo un diagrama de casos de usos con sus respectivas descripciones, de ahí que un caso de uso representa todo el intercambio entre un usuario determinado (persona o máquina) y el sistema a diseñar.

Cada caso de uso debe poseer una descripción asociada, la cual describe detalladamente el conjunto de funcionalidades que brinda. Pueden tener relaciones de inclusión o extensión con otros a su vez, con el objetivo de lograr un complemento en las funcionalidades. Por lo general su descripción incluye nombre, requisitos, restricciones, actores y la interacción con los diferentes escenarios para facilitar la comprensión. No son exactamente los requerimientos ni las especificaciones funcionales, sino que ejemplifican los requerimientos en la historia que narran.

El actor es considerado una entidad aparte del sistema que interactúa de una forma u otra con los casos de uso, participando en la historia descrita por este. De forma general realiza pedidos que funcionan como eventos de entrada o recibe algo del propio sistema. Suelen ser representado por seres humanos pero también pueden ser cualquier tipo de sistema.

1.7.1 Patrones de Modelado

El patrón es una pareja de problema/solución con un nombre y que es aplicable a otros contextos, con una sugerencia sobre la manera de usarlo en situaciones nuevas. Muchos ofrecen orientación sobre como asignar responsabilidades a los objetos ante determinado problema. (Larman, 2004)

1.7.2 Patrones de Casos de Uso

Estos patrones de una forma u otra ayudan a describir que es lo que el sistema debe hacer, describiendo el uso de este y cómo interactúan con los usuarios. Generalmente son utilizados como plantillas que describen la forma en la que deberían ser estructurados y organizados los casos de uso.

Regla del negocio: Se especializan en la extracción de información relacionada con los clientes como las (políticas, reglas y regulaciones del negocio).

- Definición estática: Es aplicado a todos los casos de uso modelando los servicios que son afectados por las reglas del negocio definidas en la organización. Sin embargo, este patrón no influye en la estructura del modelo de casos de uso. Las reglas son descritas en un documento separado, referenciadas por las descripciones de los casos de usos relevantes. Este patrón es apropiado utilizarlo cuando no hay necesidad de cambiar dinámicamente las reglas del negocio mientras el sistema se esté utilizando (Overgaard, 2004)

- Modificación dinámica: Contiene un caso de uso llamado Gestionar regla, que se encarga de crear, actualizar y eliminar las reglas del negocio. Es útil cuando la colección de reglas sea modificada dinámicamente, es decir, que puedan ser modificadas mientras el sistema se ejecute (Overgaard, 2004).

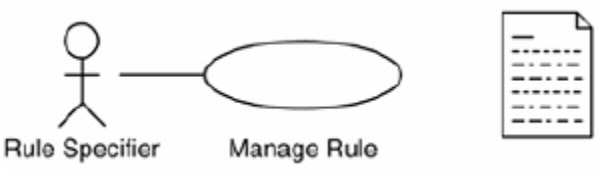


Figura 3: Patrón de CU. Reglas del negocio

Concordancia

Se encarga expresar por separado la subsecuencia de acciones que aparecen en diferentes lugares del flujo de casos de uso.

Presenta dos variantes:

- Reusabilidad: Se encarga de modelar una secuencia de acciones donde aparecerán en varios casos de uso, estos modelan el uso del sistema donde se separan los casos de uso de la sub-secuencia de acciones de manera que debe de existir al menos dos de ellos.

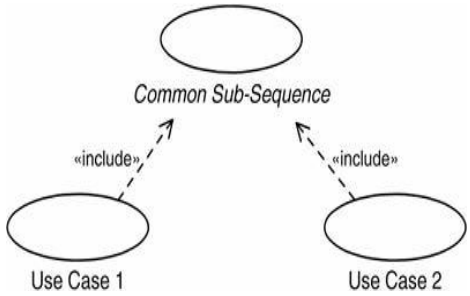


Figura 4: Patrón de CU. Concordancia - Reusabilidad.

- Adición: Extiende los casos de uso compartiendo la sub-secuencia de acciones que son a alternativas a las funcionalidades de otros casos de uso. Usarlo preferiblemente cuando otros casos de uso no requieran de una sub-secuencia común de acciones para modelar los usos completos del sistema.

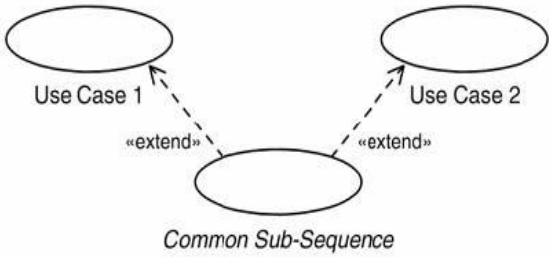


Figura 5: Patrón de CU. Concordancia - Adición

CRUD

Su nombre es acrónico a las palabras (Create, Read, Update, Delete) o (Crear, Leer, Modificar, Eliminar) más conocido como el padre de todos los patrones de capa de acceso.

En el refiere que cada objeto ha de ser creado en la base de datos de forma tal que sea persistente. Trabaja de forma:

- Completo: Es una manera de gestionar información, manejando operaciones como: crear, visualizar, modificar y eliminar información. Permite reducir el número de casos de uso y el tamaño del modelo, de forma tal que se vea más entendible.

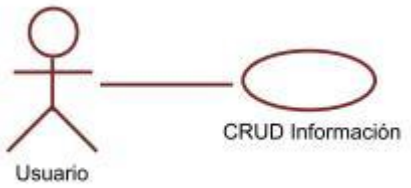


Figura 6: Patrón de CU. CRUD – Completo

- Parcial: Es preferible que sea utilizado cuando algunas de las alternativas de los casos de uso sean más significativa, larga o más compleja que las otras. Separan una funcionalidad por determinadas características que la hacen compleja o tediosa.

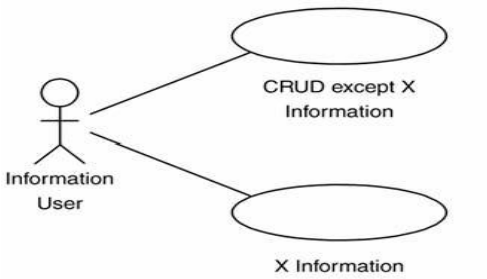


Figura 7: Patrón de CU. CRUD - Parcial.

Patrón Extensión Concreta o Exclusión

- Extensión: No es más que casos de uso y una relación extendida entre ellos. Este patrón es aplicado cuando un flujo puede extender el flujo de otro caso de uso, así como ser realizado en sí mismo

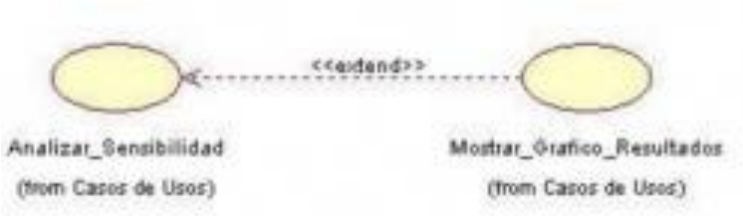


Figura 8: Patrón de CU. Extensión Concreta o Exclusión - Extensión.

- Inclusión: Incluye una relación entre el caso de uso base al caso de uso de inclusión.

Figura 9: Patrón de CU. Extensión Concreta o Exclusión - Inclusión.

Patrón Múltiples Actores:

- Roles diferentes: No es más que cuando un caso de uso es necesario para dos actores y estos interactúan de forma diferente ante el caso de uso.

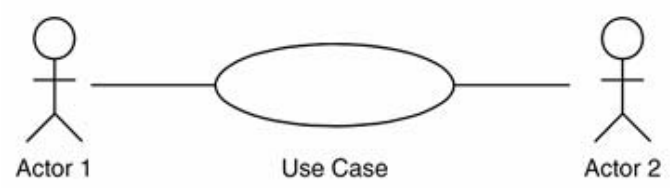


Figura 10: Patrón de CU. Múltiples Actores - Roles diferentes.

- Roles comunes: Representa una herencia entre actores y es aplicado cuando dos actores juegan un mismo rol sobre un caso de uso.

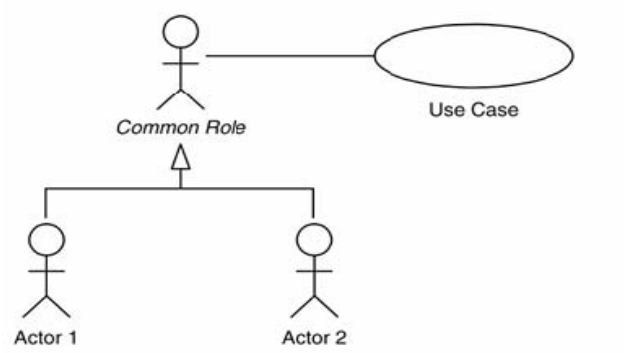


Figura 11: Patrón de CU. Múltiples Actores - Roles comunes.

1.8 Conclusiones

Teniendo en cuenta la necesidad que presentan los estudiantes de la asignatura de Sistema Operativo y enfocándose en el tema de planificación de disco, se hace un profundo estudio referente a los Laboratorios Virtuales, proponiendo la elaboración de uno en el centro de tipo web por sus características antes mencionadas, con las particulares independientes de la disciplina respondiendo a las necesidades actuales de la universidad, favoreciendo el aprendizaje de los alumnos. Se selecciona como metodología de desarrollo de software Rational Unified Process (RUP) siendo la propuesta y usada por el grupo de

desarrollo encargado de realizar el laboratorio virtual, pues a pesar de ser un proyecto pequeño es la que más se ajusta a su modelación. Se define como lenguaje de modelado UML por todas sus características mencionadas y además por ser conocido, estándar, fácil de usar y ser el propuesto por la metodología. Como herramienta de modelado se empleará Visual Paradigm, pues es una herramienta multiplataforma y facilita un lenguaje estándar común a todo el equipo de desarrollo facilitando mayor comunicación. Por último se realiza un estudio de los diferentes patrones de casos de uso, donde se obtiene una mayor comprensión y aplicación de los mismos.

CAPÍTULO 2. DESCRIPCIÓN DE LA SOLUCIÓN.

2.1 Introducción

En el presente capítulo se muestra un mapa conceptual encargado de ilustrar las relaciones de los contenidos a tener en cuenta para la realización del evaluador de ejercicios de planificación de disco. Se especifican los requisitos funcionales y no funcionales por los que se regirá el sistema propuesto dando paso a algunos artefactos presentados por la metodología definida y logrando a través del uso de patrones un correcto diseño.

2.2 Modelo Conceptual.

Un Modelo conceptual es un diagrama que ilustra una serie de relaciones entre ciertos factores que se impactan o conducen a una condición de interés.

Puede verse como la representación de conceptos del mundo real y las relaciones entre ellos. Requiere de cierto nivel de abstracción, creado para aumentar la comprensión del problema, contribuyendo a esclarecer la terminología o nomenclatura del dominio.

El modelo propuesto consta de 9 conceptos expresados a continuación:

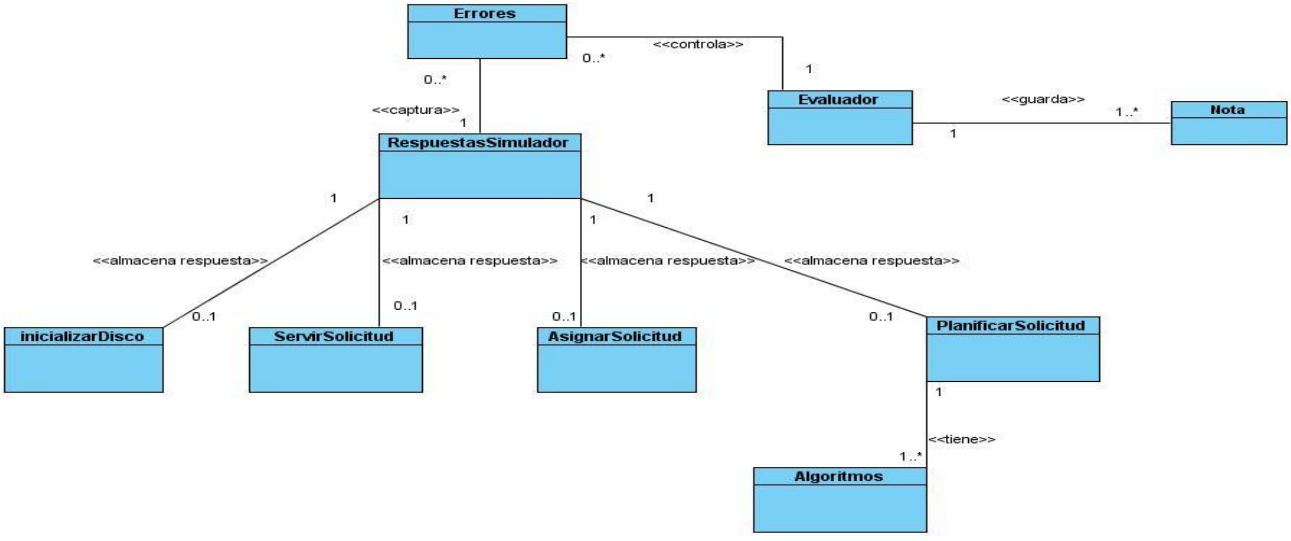


Figura 12: Modelo conceptual.

Como muestra la figura un evaluador controla todos los errores que a su vez son capturados de las respuestas del simulador quien se encarga de almacenar todos los atributos introducidos en memoria como son mostrados en el modelo presente para la verificación, captura y rectificaciones de los errores detectados durante la simulación del ejercicio, luego de ello el evaluador se encargará de guardar las notas alcanzadas para un control y seguimiento de las evaluaciones.

2.3 Ingeniería de Requisitos (IR).

La Ingeniería de Requisitos conocida también como "Análisis de requisitos" o "Especificación de requisitos", juega un papel primordial en el desarrollo del software pues se centra en definir qué es lo que realmente quiere el cliente, especificando con claridad las necesidades de los usuarios o clientes.
Según Pressman: Ingeniería de Requerimientos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajarán. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software (Pressman, 2005)
Como actividades se centra en:

1. Extracción: No es más que el comienzo de cada ciclo con el objetivo de extraer todos los problemas que debe resolver el sistema, identificándose los requisitos funcionales y no funcionales.
2. Análisis: Su principal objetivo es descubrir los inconvenientes con los requerimientos del sistema identificados hasta ese entonces, agrupándolos y clasificándolos de acuerdo a las funcionalidades que responden.
3. Especificación: Se encarga de documentar los requerimientos antes vistos con el cliente, llevándolo a un nivel más entendible para los desarrolladores.
4. Validación: Su misión es verificar que los requerimientos especificados en el documento representen una descripción aceptable al sistema que se implementará con posterioridad.
5. Gestión: Encargada de darle seguimiento a los requisitos a lo largo de las etapas anteriores.

Su propósito es tratar de alcanzar un estado óptimo antes de la fase del diseño del proyecto, de forma tal que se logren todas las especificaciones necesarias por el usuario o cliente final, logrando la obtención de requisitos medibles y comprobables, sin ambigüedades o contradicciones, precisos, que puedan ser leídos y modificables.

2.3.1 Técnicas de captura de Requisitos

La captura de requisitos es el acto de averiguar desde cualquier fuente de información disponible las necesidades que debe cubrir el sistema, de forma tal que favorezca a todo el equipo de desarrollo.
Dentro de sus principales técnicas se encuentra la *¨Entrevista¨, la ¨Tormenta de ideas ¨* siendo las utilizadas en el trabajo.
Entrevista: Se realiza a la hora de realizar entrevistas con el cliente con el objetivo de extraer toda la información posible sobre la visión que el entrevistador tiene de los requisitos. Su éxito depende de la habilidad del entrevistador y su preparación para la misma. Fue utilizada durante la investigación directamente con el cliente (Ing. Carlos Yasmany Hidalgo) abordando una serie de preguntas indispensables para la documentación y análisis del funcionamiento del sistema. Entre ellas se tuvieron en cuenta: ¿Qué necesidades existen?, ¿Qué es lo que el sistema debería hacer? ¿Con que objetivos y funcionalidades debe cumplir el sistema?, ¿Quiénes utilizarán el sistema?, ¿Qué podrá realizar cada actor en el sistema?, ¿Dónde utilizará el usuario el sistema?
A partir de estas preguntas y del negocio modelado se lograron documentar una serie de requisitos indispensables para la realización del sistema, donde se obtuvieron 9 requisitos funcionales, de ellos 5 con categoría crítica, además de 12 requisitos no funcionales importantes ante las condiciones que debe cumplir el sistema luego de ser desarrollado.

La Tormenta de ideas es parte de las técnicas y consiste en la realización de reuniones en grupo, cuyo objetivo es la intervención de sus participantes de forma tal que muestren sus ideas libremente, acumulando información sin evaluarse. Suele ofrecer una visión general de las necesidades del sistema aunque no sirve para obtener detalles concretos del mismo. Se realizó una reunión directamente con el cliente (Ing. Carlos Yasmany Hidalgo) donde estuvo presente el tutor (Ing. Yordanis Milanés Zamora).

Existieron varias intervenciones donde se plantearon las necesidades existentes aportando opiniones e ideas de las funcionalidades necesarias para el sistema, priorizando los requisitos más significativos aprobando los obtenidos en la reunión antes elaborada.
A partir de las anteriores técnicas se obtuvieron los siguientes requerimientos:

Requisitos Funcionales.

Los Requisitos Funcionales de un sistema son los que describen las funcionalidades o servicios que se espera de estos, describiendo con detalle las funciones, entradas y salidas, excepciones, etc.

Tabla de Requisitos Funcionales respecto a este trabajo.

Referencia	Requisitos Funcionales
RF 1	Evaluar
RF 2	Inicializar Disco
RF 3	Asignar Solicitudes
RF 4	Calcular Capacidad del Disco
RF 5	Calcular Dirección Lógica
RF 6	Calcular Dirección Física
RF 7	Calcular Parámetros del Disco
RF 8	Planificar Solicitudes
RF 9	Servir Solicitudes

Requisitos No Funcionales.

Estos no se refieren a las funcionalidades específicas que debe entregar el sistema, sino a las propiedades como la fiabilidad, respuestas en el tiempo y capacidades de almacenamiento, no describen información a guardar ni funciones a realizar.

Apariencia o interfaz externa:

RNF 1: 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.

RNF 2: La combinación de colores debe ser agradable a la vista del usuario.

RNF 3: Debe contar con un vínculo a la ayuda en cada ventana de trabajo.

RNF 4: El sistema tendrá las funcionalidades principales en una primera (principal) interfaz. El objetivo es no tener que acceder a varias secciones para ver los elementos que más interés le causen. Es un sistema con las funcionalidades disponibles desde una bandeja de entrada con los elementos para desempeñar su tarea.

Usabilidad:

RNF 4: El sistema puede ser usado por cualquier estudiante o profesor que posea conocimientos básicos sobre la asignatura.

Rendimiento:

RNF 5: La respuesta a solicitudes más complejas de los usuarios del sistema no debe exceder 9 segundos.

Disponibilidad

RNF 6: La aplicación deberá estar disponible las 24 horas del día.

RNF 7: El usuario solo podrá ver la nota alcanzada una vez realizado el ejercicio y en caso de que cierre la aplicación se le dará un dos automáticamente, además de solo tener la posibilidad de evaluarse dos veces y se la guardará la nota como repetición en caso de que cuente con una segunda evaluación. Todas las notas serán guardadas como un historial de evaluación.

Seguridad

RNF 8: El sistema debe poseer mecanismos de seguridad que garanticen la confidencialidad de la información basada en el modelo (Autentificación, Autorización y Auditoría).

RNF 9: Garantizar que la información solo sea modificada por las personas que tengan permiso para realizar esta actividad.

RNF 10: Incluir mecanismos de recuperación de datos (salvas automáticas o copias de respaldo) que permitan recuperar la información ante un fallo del sistema.

Requisitos de diseño e Implementación

RNF 11: El sistema será implementado en Java, usando el IDE NetBeans 6.0, se utilizará como herramienta de modelado Visual Paradigm, y un paradigma de Programación Orientado a Objetos.

Requisitos de Software

RNF 12: Navegador web con interprete JavaScript.

RNF 13: Máquina Virtual Java

Hardware:

RNF 14: Tarjeta de memoria RAM de 256 MB como mínimo.

RNF 15: Procesador Pentium IV o superior a 2.0 MHz como mínimo.

RNF 16: Tarjeta de red

2.4 Actores del Sistema.

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

Los actores del sistema pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado, son parte del sistema, y pueden intercambiar información con él o ser recipientes pasivos de información. (I Jacobson, G Booch, J Rumbaugh, 2000.)

En este caso los actores que interactúan en el sistema se presentan a continuación:

Actor del sistema	Descripción
Usuario	Es la persona que va a interactuar con el sistema, este puede ser el estudiante, el profesor o cualquier usuario que tenga conocimientos del tema y requiera de su uso.

2.4.1 Diagrama de casos de uso del sistema.

Una de las técnicas más efectivas para modelar los requisitos del sistema son los modelos de casos de uso, utilizados para modelar el funcionamiento que desea el cliente que tenga el sistema.

A continuación se muestra el diagrama de casos de uso del sistema:

Diagrama del Sistema

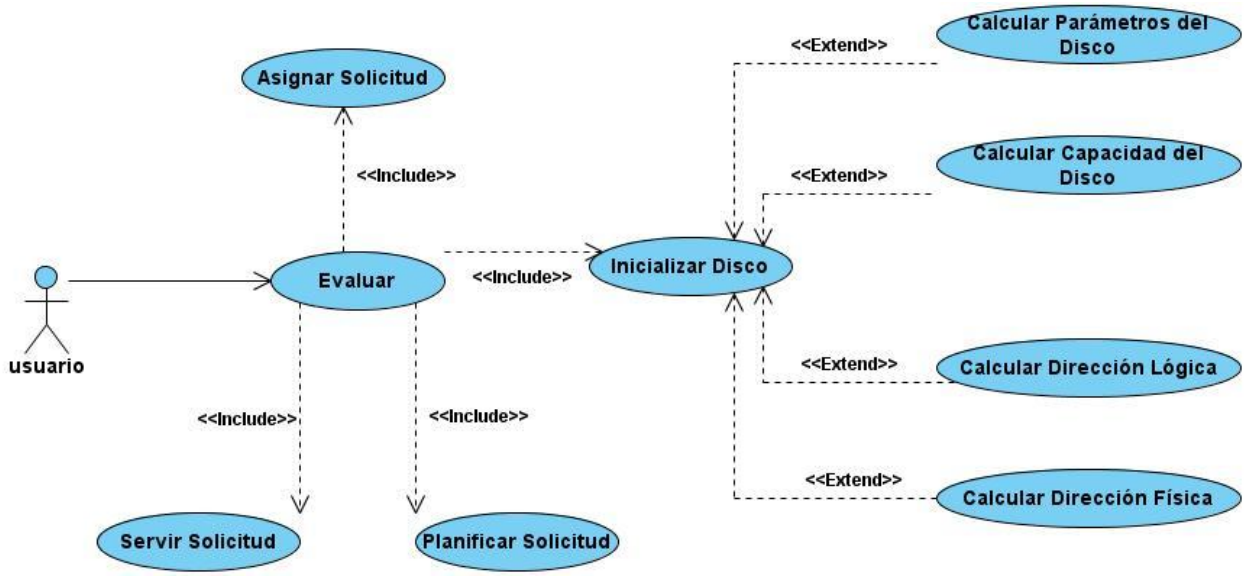


Figura 13: Diagrama de CU del Sistema

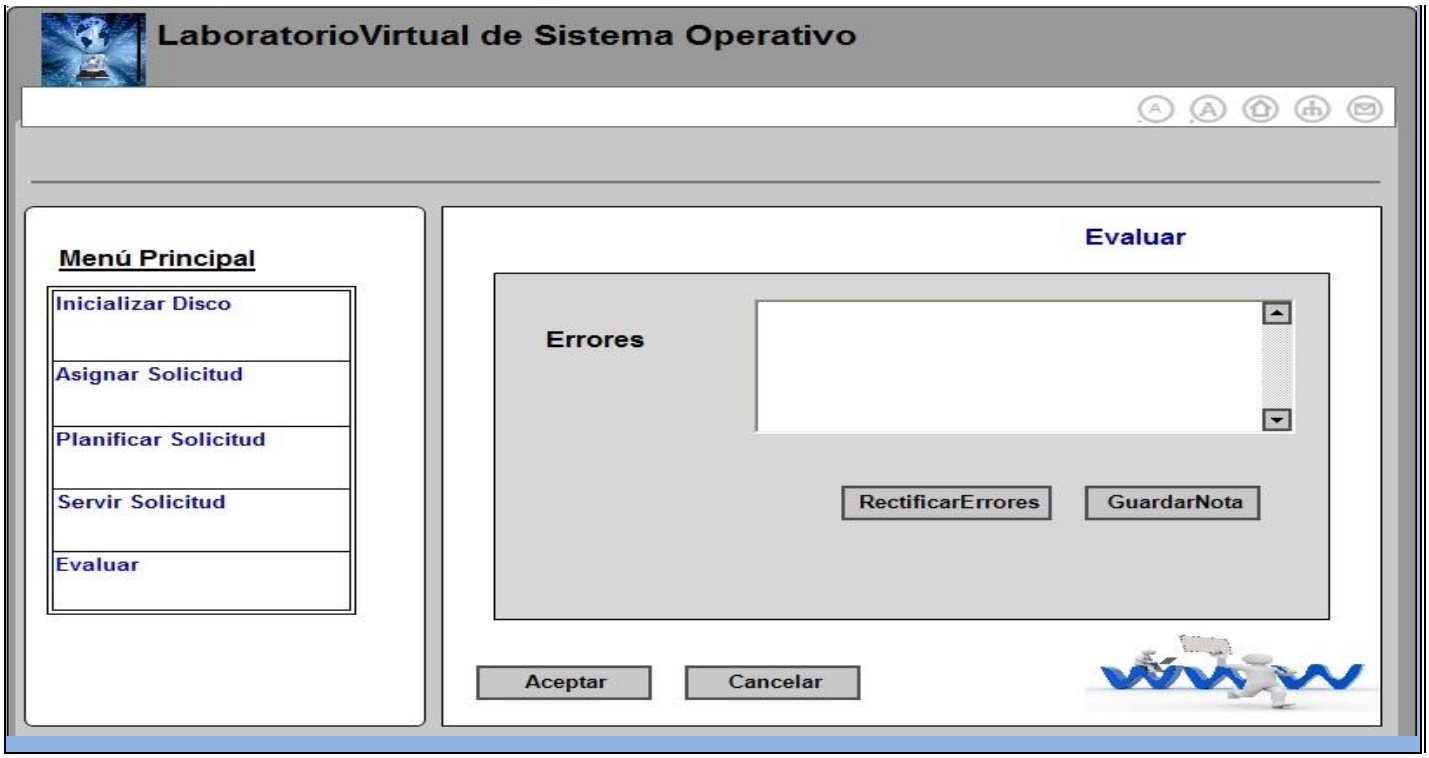
2.4.2 Descripción de los casos de uso del sistema.

Descripción del Caso de Uso: "Evaluar".

Caso de uso	Evaluar
Actores	usuario

Resumen	El caso de uso inicia una vez que el usuario desea conocer la evaluación de la comprobación realizada, el sistema se encargaría de evaluarlas verificando los resultados del sistema contra los resultados de la simulación, capturando los errores cometidos de forma tal que se logran cuantificar y como cierre guarde la nota de la comprobación en la base de datos encargada de llevar el seguimiento de las evaluaciones.
Pre condiciones	Se tienen que haber introducido todo los datos pedidos por el ejercicio y el sistema debe verificar que el usuario no se haya evaluado más de dos veces.
Pos condiciones	Los datos introducidos por el usuario son guardados en memoria que a su vez será comparada con las respuestas del sistema capturando los errores y mostrárselos al usuario, además de registrar el intento realizado.
Casos de uso asociados	**No procede**
Prioridad	**Crítico**.
Flujo Normal de Eventos	
Sección: "General"	
Acción del actor	**Respuesta del sistema**
1. El usuario desea conocer la evaluación obtenida durante el ejercicio realizado presionando en el Menú Principal opción "Evaluar"	1.1. El sistema guarda las respuestas de la simulación en memoria
	1.2 Carga las respuestas del sistema

	1.3 Verifica los resultados contra los simulados capturando los errores cometidos por el usuario
	1.4 Por último muestra en la interfaz “Evaluación” los errores señalados en color rojo.
2. El usuario desea consultar las posibles respuestas de sus errores.	2.1 El sistema luego de cargar todas las respuestas se encargará de buscar en la base de datos sus rectificaciones mostrándole al usuario las posibles respuestas de las deficiencias cometidas.
3. El usuario selecciona la opción “Cerrar Evaluación”	3.1 El sistema se encargará de obtener la evaluación y le mostrará la nota al usuario en la interfaz principal y a su vez la enviará a la base de datos encargada de llevar el control de todas las evaluaciones y cerrará la interfaz “Evaluación”.
	3.2 El sistema brindará la posibilidad de que el usuario solo una vez más se vuelva a evaluar, inicializando el nuevo ejercicio donde obtendrá otra nota que será mostrada y guardada en la base de datos a continuación de la anterior como repetición de la evaluación.
Flujo Alterno	
Acción del Actor	**Respuesta del Sistema**
	1.5 Si el usuario presiona más de dos veces la opción “Evaluar” el sistema mostrará un mensaje de error y cerrara la ventana del ejercicio.
	1.6 Si el usuario cancela o abandona la elaboración del ejercicio, el sistema enviará automáticamente a la base de datos una evaluación de 2 puntos.
Prototipo de interfaz de Usuario	



Las restantes descripciones se encuentran en el Anexo 1.

2.5 Patrones de CU.

A la hora de realizar un diagrama de casos de uso es necesario la utilización de un conjunto de patrones para su representación, de ellos se utilizarán en el desarrollo del presente trabajo:

Patrón Extensión Concreta o Exclusión:

- Extensión: No es más que casos de uso y una relación extendida entre ellos. Este patrón es aplicado cuando un flujo puede extender el flujo de otro caso de uso, así como ser realizado en sí mismo. Especifica cómo el comportamiento definido por el caso de uso de extensión puede insertarse dentro del comportamiento definido por el caso de uso base. Representa que parte del

procedimiento del caso de uso base es opcional o condicional, el cual si es importante es posible que se desee describirlo por separado como una relación de extensión.

- Inclusión: Incluye una relación entre el caso de uso base al caso de uso de inclusión. Especifica cómo el comportamiento definido para el caso de uso de inclusión se inserta explícitamente dentro del comportamiento definido para el caso de uso base. Usado para dividir un flujo de trabajo cuando del resultado de estas partes depende el valor del caso de uso base. Se puede utilizar además si este tipo de división simplifica el entendimiento del caso de uso base o si uno de estos casos de uso se pueda reutilizar en otra parte del sistema.

2.6 Análisis del sistema

"El término análisis, aplicado a sistemas, significa descomponerlos en sus componentes, para estudiar cada uno de ellos, tanto como un ente aislado, como en interacción con el resto. Para ser útil, al análisis le debe seguir la síntesis, que consiste en unir los componentes del sistema para ver cómo funcionan en conjunto." [PIA96]

2.6.1 Modelo de Análisis

Representa la estructura global del sistema, describe la realización de casos de uso y sirve como una abstracción del Modelo de Diseño. Debe lograr tres objetivos primarios:

- Describir lo que requiere el cliente.
- Establecer una base para la creación de un diseño de software.
- Definir un conjunto de requisitos que se pueda validar una vez que se construye el software.

Su propósito es definir todas las clases que son relevantes al problema que se va a resolver, las operaciones y atributos asociados, las relaciones y comportamientos asociadas con ellas. (Pressman, 2005)

Diagramas de Clases de Análisis.

Representa los conceptos fundamentales que se abarcan en el dominio del problema, las definiciones y relaciones entre las clases. Las clases se clasifican en Interfaz, de Control o Entidad.

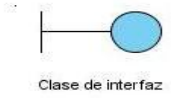


Figura 14: Estereotipo de interfaz

- Clase Interfaz: Representa gráficamente la interacción entre el sistema y sus actores.

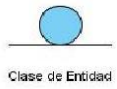


Figura 15: Estereotipo de entidad

- Clase Entidad: Modelan información que posee larga vida y que es a menudo persistente.

Figura 16: Estereotipo de control

- Clase Control: Representa de manera gráfica la coordinación, secuencia, transacciones, y control de otros objetos.

Diagramas de clases de Análisis

Diagrama de Clases de Análisis

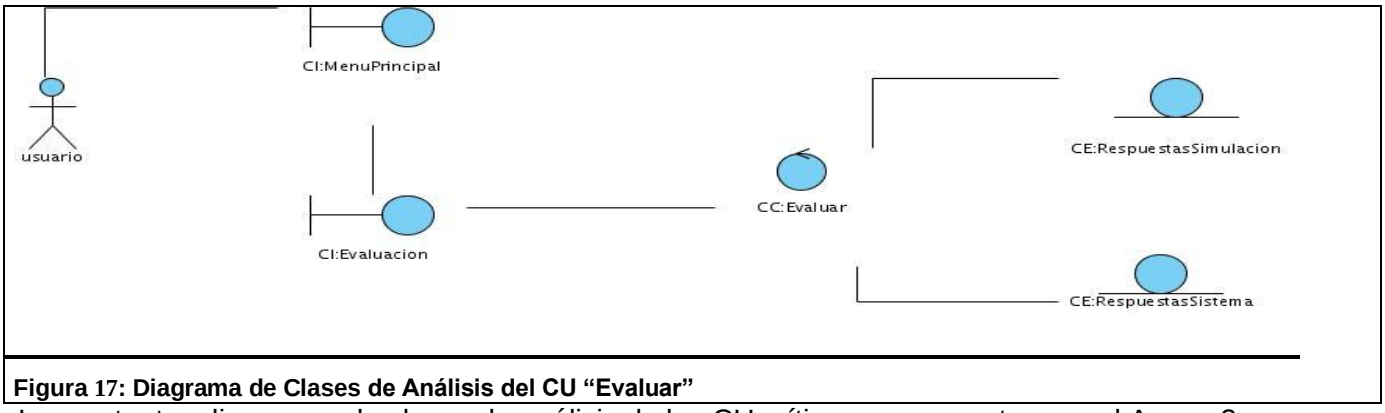


Figura 17: Diagrama de Clases de Análisis del CU "Evaluar"

Los restantes diagramas de clases de análisis de los CU críticos se encuentran en el Anexo 2.

2.7 Diagramas de Secuencia

Muestra gráficamente las interacciones del actor y las operaciones a que dan origen. Su creación depende de la formulación previa de los casos de uso conteniendo detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementarlo y los mensajes de intercambio entre los objetos.

Diagrama de Secuencia

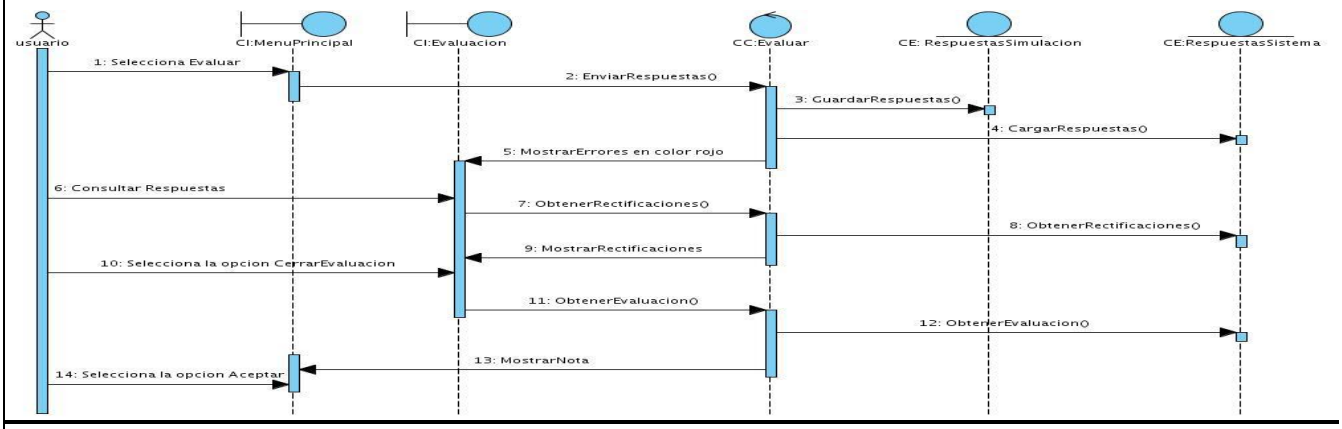


Figura 18: Diagrama de secuencia del CU"Evaluar"

Los restantes diagramas de secuencia de los CU críticos se encuentran en el Anexo 3.

2.8 Descripción de la Arquitectura.

Una Arquitectura Software, también denominada arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de la referencia necesario para guiar la construcción del software de un sistema de información. La Arquitectura de Software establece los fundamentos para trabajar una línea común que permite alcanzar los objetivos y necesidades del sistema de información. (I Jacobson, G Booch, J Rumbaugh, 2000).

2.8.1 Modelo Vista Controlador

El patrón de arquitectura de Modelos Vista Controlador (MVC) divide una aplicación interactiva en tres componentes distintos, separando los datos de una aplicación, la interfaz del usuario y la lógica de control.

Modelo: Representa las estructuras de datos y funcionalidades. Su responsabilidad es acceder a la capa de almacenamiento de datos, definir las reglas del negocio y llevar un registro de las vistas y controladores del sistema.

Vista: Muestra la información al usuario. Su responsabilidad es recibir los datos del controlador y mostrárselos al usuario.

Controlador: Actúa como intermedio entre el Modelo, la Vista y cualquier y cualquier otro recurso necesario para generar una página, encargada de manejar todos los eventos provenientes de la interfaz.

2.9 Diseño

El Diseño de Sistemas representa un proceso iterativo mediante el cual los requisitos se traducen en un plano para construir el software, el cual se encuentra en el núcleo técnico de la ingeniería del software. Su importancia se traduce en una sola palabra, calidad, ya que sin este se corre el riesgo de construir un sistema inestable, que fallará cuando se lleven a cabo cambios que pueden resultar difícil de comprobar y cuya calidad no puede evaluarse hasta muy avanzado el proceso (Pressman, 2005).

Diagrama de Clases del Diseño

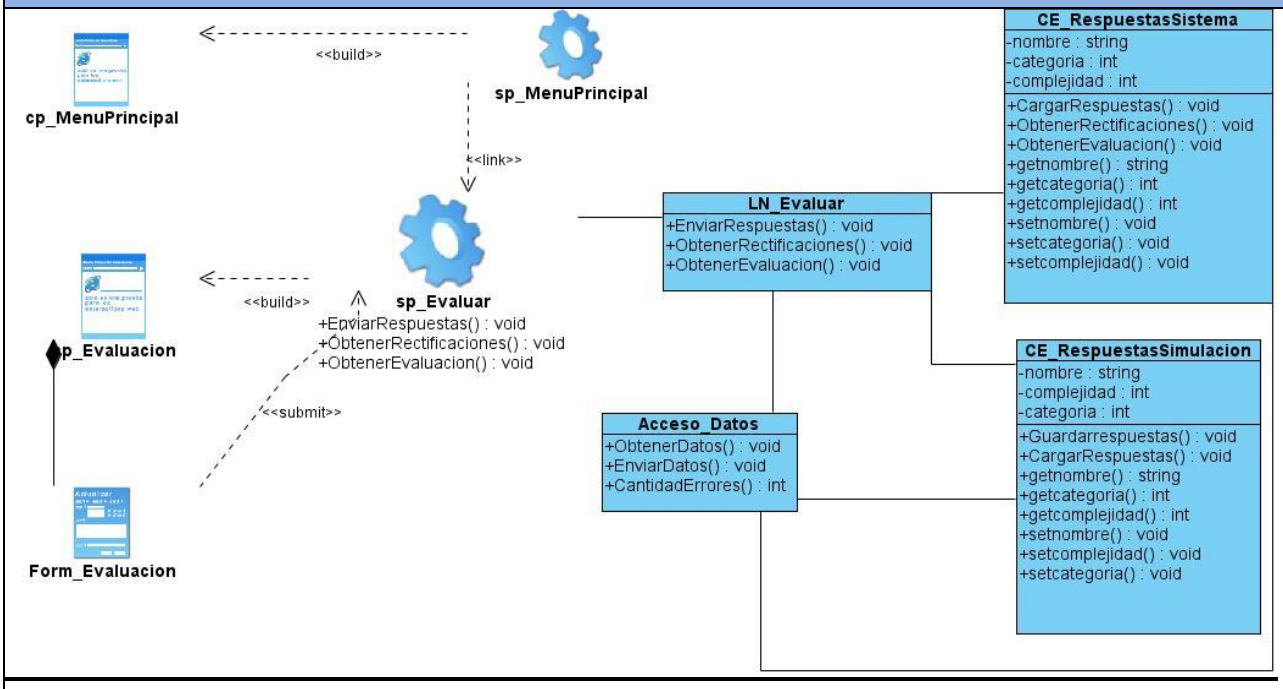


Figura 19: Diagrama de diseño del CU "Evaluar"

Los restantes diagramas de diseño de los CU críticos se encuentran en el Anexo 4.

Ya una vez realizado el diseño de un sistema, es necesario tener en cuenta una serie de elementos capaces de describir su comportamiento, como los patrones de diseño descritos a continuación los usados durante la realización del trabajo.

2.10 Patrones de Diseño

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno y describe también el núcleo de su solución, de forma que puede utilizarse un millón de veces sin hacer dos veces lo mismo (Christopher Alexander, Arquitecto y urbanista). (Félix Prieto, 2005)

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptada para resolver un problema general de diseño en un contexto particular. (Félix Prieto, 2005)

Es una descripción de un problema y su solución que recibe un nombre y que puede emplearse en otros contextos, en teoría indica cómo utilizarlo en diversas circunstancias, en donde nos dan la orientación sobre como asignar las responsabilidades a los objetos ante diversas categorías de problemas. (Larman, 2004).

Los patrones de diseño son el esqueleto de las soluciones a problemas presentes durante el desarrollo del software. En él debe tener presente: su nombre, problema (cuando aplicar un patrón), la solución (descripción abstracta del problema) y las consecuencias (costos y beneficios).

Hoy en día existen disímiles patrones de diseño de software y los usados durante la realización de la problemática planteada se encuentran:

Patrón Experto

Asignar una responsabilidad al experto en información: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase, contiene toda la información necesaria para realizar la labor que tiene encomendada. Este patrón se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos; expresa simplemente la "intuición" de que los objetos hacen cosas relacionadas con la información que poseen. (Larman, 2004)

Su solución es asignar responsabilidades a quien tiene la información necesaria para ejecutar sus responsabilidades. Es evidenciado en los diagramas sobre todo en las clases controladoras quienes poseen la mayor información para resolver las funcionalidades requeridas.

Patrón Creador

Asignar a la clase B la responsabilidad de crear una instancia de clase A en alguno de los siguientes casos (B agrega los objetos de A; B contiene a los objetos de A; B registra las instancias de los objetos de A; B utiliza específicamente los objetos de A; B tiene los datos de inicialización que serán transmitidos a A cuando este objeto sea creado). El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento. (Larman, 2004)

Su propósito es encontrar un creador que se conecte al objeto producido en cualquier evento. Encargado de asignar responsabilidades relacionadas con la creación de objetos. Brinda como solución asignarle a una clase B la responsabilidad de crear una instancia de la clase A.

Patrón Controlador

Asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Esto facilita la centralización de actividades (validaciones, seguridad y otros). La mayor parte de los sistemas reciben eventos de entrada externa, por lo cual, si se recurre a un diseño orientado a objetos, hay que elegir los controladores que manejen esos eventos de entrada. (Larman, 2004)

Ofrece una guía para tomar decisiones apropiadas que generalmente se aceptan. La clase controladora debe utilizarse con todos los eventos sistémicos de un caso de uso de modo que se conserve la información referente al estado del caso. Como solución asignan responsabilidades a una clase para responder a eventos

Patrón Alta cohesión

El patrón de alta cohesión indica que los datos y responsabilidades de una entidad están fuertemente ligados a la misma en sentido lógico. Como solución plantea que cada elemento debe realizar una única tarea y bien identificada. (Félix Prieto, 2005)

Patrón Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases. Una clase con baja o débil no depende de muchas otras clases y cuando es viceversa se considera que requiere de muchas otras. Su problema es tratar de alcanzar baja dependencia y alta reutilización y como solución asigna responsabilidades con poco acoplamiento, es decir, poca dependencia de otras clases o influencias sobre ellas. (Félix Prieto, 2005)

2.11 Conclusiones

A través de este capítulo fueron expuestos los artefactos generados durante el análisis y el diseño de la solución propuesta para el evaluador de ejercicios de planificación de disco del laboratorios virtual de SO, quedando bien claras todas las funcionalidades del sistema para una posterior implementación. A través de las técnicas de captura de requisitos fueron descritas todas las necesidades tanto funcionales como no funcionales definiendo la estructura del sistema que se pretende desarrollar como propuesta solución y para lograr un correcto diagrama tanto de caso de uso del sistema como de clases del diseño fueron empleados una serie de patrones dando paso a la validación de la solución con el uso de métricas para comprobar la calidad del diseño y las no ambigüedades de los requisitos obtenidos.

CAPÍTULO 3 VALIDACIÓN DE LA SOLUCIÓN

3.1 Introducción

El presente capítulo se basa en la comprobación detallada de los artefactos generados durante la descripción de la solución; considerada como la etapa final de la actual investigación aplicando a través de las métricas; la validez del sistema propuesto.

3.2 Métricas

Las métricas se definen como la aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo del software, sus productos y el suministro de información relevante a tiempo.

3.2.1 Métricas para la calidad de las especificaciones

Todos los requisitos capturados se validaron a través de la métrica Especificación no ambigua. Examina que todos los requisitos del sistema hayan sido establecidos sin ambigüedad, demostrando que todas las personas involucradas en el proceso de revisar los requisitos coincidieron con la interpretación de los mismos.

Para desarrollar esta métrica es preciso conocer el valor total de requisitos (Rt), el mismo se halla mediante la siguiente fórmula:

Rt= Rf + Rnf

25= 9 + 16

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:

Para ello se calcula **Q1** para determinar la especificidad de los requisitos:

Q1 = Rui / Rt

Q1= 25/25

Q1= 1.

Dónde:

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

Q1: Ausencia de ambigüedad

Cuanto más cerca de 1 esté el valor de **Q1**, menor será la ambigüedad de la especificación arrojando como resultado del presente trabajo que no existen ambigüedades en los requisitos.

Grupo de Revisores que aplicaron esta métrica:

- ✓ Ing. Carlos Y. Hidalgo García.

3.2.2 Métricas para la validación del diseño

Las métricas la mayoría de las veces nos ayudan a entender tanto el proceso técnico que se utiliza para desarrollar un producto, como el propio producto, y para intentar mejorarlo se mide para pretender aumentar su calidad.

Las métricas para aplicaciones informáticas no son perfectas, pues se necesita de mucha experimentación para su uso, sin embargo es considerada una alternativa aceptable.

Las métricas empleadas están diseñadas para evaluar los siguientes atributos de calidad:

Responsabilidad: No es más que la responsabilidad que se le es asignada a una clase en un marco de modelado de un dominio o concepto de la problemática propuesta.

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

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

Acoplamiento. Consiste en el grado de dependencia o interconexión de una clase o estructura de clase, con otras, está muy ligada a la característica de Reutilización.

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

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

3.2.3 Resultados obtenidos de la aplicación de la métrica TOC

Tamaño operacional de clases (TOC): Evalúa la calidad del diseño propuesto a nivel de componentes. Se refiere al número de métodos pertenecientes a una clase. Determinada por los atributos: Responsabilidad, Complejidad de implementación y la Reutilización, sin embargo establece una relación inversa entre estos últimos y el atributo Reutilización.

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

Tabla 1: Atributos de calidad evaluados por la métrica TOC.

Para aplicar esta métrica se tuvo en cuenta la cantidad de procedimientos que tenía cada clase y a partir de ellos y mediante un criterio se obtuvieron las categorías de (alta, media, baja) para los atributos antes mencionados.

Luego de efectuar los cálculos para un:

Total de clases	**17**
Promedio de asociaciones de uso	**1,588235294**

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
Evaluar	3	Media	Media	Media
RespuestasSimulacion	2	Media	Media	Media
RespuestasSistema	3	Media	Media	Media
Inicializar	2	Media	Media	Media
Disco	1	Baja	Baja	Alta
Asignar	1	Baja	Baja	Alta
Solicitud	1	Baja	Baja	Alta
Planificar	1	Baja	Baja	Alta
Solicitud	1	Baja	Baja	Alta
Algoritmo	1	Baja	Baja	Alta
Servir	1	Baja	Baja	Alta
Solicitud	1	Baja	Baja	Alta
Acceso_Datos_Solicitud	1	Media	Media	Media
Acceso_A_Solicitud_Algoritmo	3	Media	Media	Media
Acceso_Solicitud	1	Baja	Media	Alta
Acceso_Datos	3	Media	Media	Media
AccesoDato	1	Baja	Media	Alta

Tabla 2: Cantidad de procedimientos por clase.

Para evaluar estas métricas se tuvieron presentes los siguientes umbrales:

	Categoría	Criterio
Responsabilidad	Baja	< =Promedio de operaciones (PO)
	Media	> PO y < = 2* PO
	Alta	> 2* PO
Complejidad implementación	Baja	< =Promedio de operaciones (PO)
	Media	> PO y < = 2* PO

	Alta	> 2* PO
Reutilización	Baja	> 2*PO
	Media	> PO y < = 2* PO
	Alta	<= PO

Tabla 3: Criterio para calcular la Responsabilidad, la Complejidad implementación y la Reutilización.

A través de estas tablas para determinar la categoría se obtuvo las siguientes gráficas de porcentaje para la Responsabilidad, Complejidad y Reutilización:

Responsabilidad	Cantidad de clases
Baja	10
Media	7
Alta	0

Tabla 4: Responsabilidad de las clases.

Representación en % de la incidencia de los resultados obtenidos en el atributo Responsabilidad:

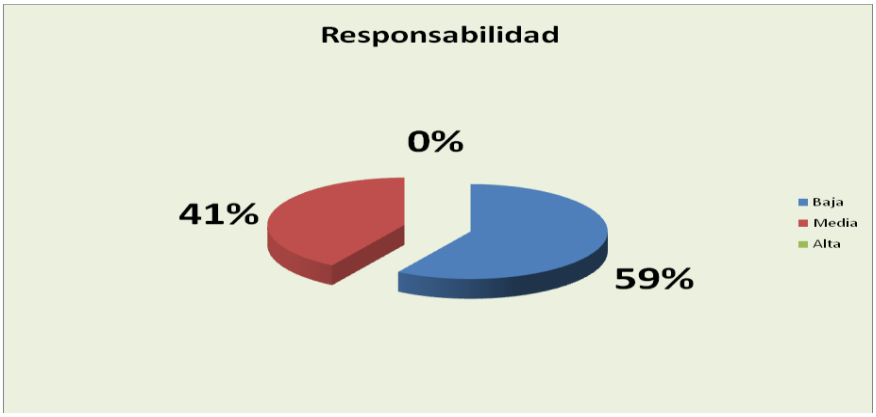


Figura 20: Representación de la Responsabilidad.

Complejidad	Cantidad de clases

Baja	10
Media	
Alta	0

Tabla 5: Complejidad de las clases.

Representación en % de la incidencia de los resultados obtenidos en el atributo Complejidad de Implementación.

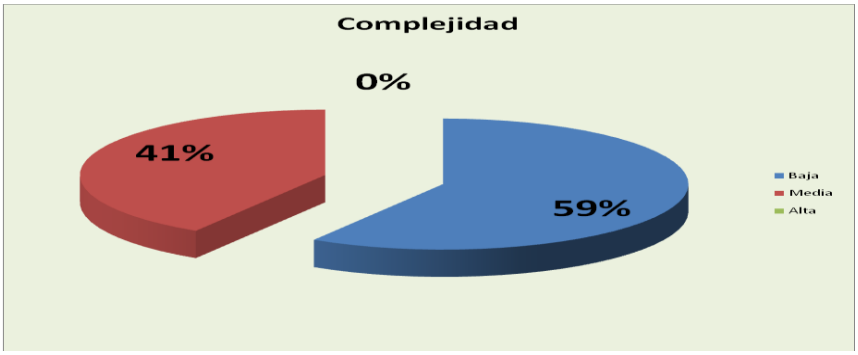


Figura 21: Representación de la Complejidad.

Como se pudo apreciar en las representaciones anteriores la Complejidad y la Responsabilidad son realmente bajas, lo que indica que el diseño es satisfactorio y la implementación no será de mucha complejidad.

Reutilización	Cantidad de clases
Alta	10
Media	7
Baja	0

Tabla 6: Complejidad de las clases.

Representación en % de la incidencia de los resultados obtenidos en el atributo Reutilización.

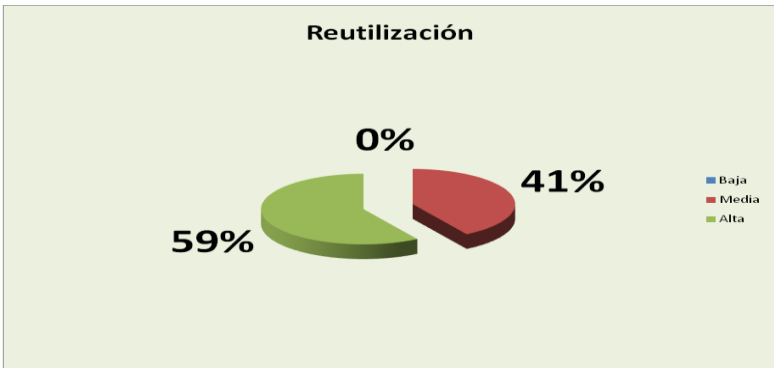


Figura 22: Representación de la Reutilización.

El 59% de las clases son de tamaño bajo según los umbrales aplicados, representando un resultado positivo por lo que implica una alta reutilización de las clases y una baja responsabilidad y complejidad de las mismas. Considerándose que el diseño elaborado se encuentra dentro de los límites de calidad.

3.2.4 Resultados obtenidos de la aplicación de la métrica RC

Se refiere al número de relaciones de uso de una clase. Determinada por los atributos: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas, existiendo una relación directa con los tres primeros e inversa con el último antes mencionado.

Atributo de calidad	Modo en que lo afecta
Acoplamiento	Un aumento del RC implica un aumento del Acoplamiento de la clase.
Complejidad de mantenimiento	Un aumento del RC implica un aumento de la complejidad del mantenimiento de la clase.
Reutilización	Un aumento del RC implica una disminución en el grado de reutilización de la clase.
Cantidad de pruebas	Un aumento del RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

Tabla 7: Atributos de calidad evaluados por la métrica RC.

Para los cuales están definidos los siguientes criterios y categorías de evaluación para un promedio de 0,882352941.

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

Tabla 8: Criterios de evaluación para la métrica RC.

A través de estas tablas para determinar la categoría se obtuvo las siguientes gráficas de porcentaje para la Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas:

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
LN_Evaluar	3	Alto	Alto	Baja	Alto
RespuestasSimulacion	0	Ninguno	Baja	Alta	Baja
RespuestasSistema	0	Ninguno	Baja	Alta	Baja
LN_Inicializar	1	Bajo	Media	Media	Media
Disco	0	Ninguno	Baja	Alta	Baja
LN_Asignar	1	Bajo	Media	Media	Media

Solicitud	0	Ninguno	Baja	Alta	Baja
LN_Planificar	1	Bajo	Media	Media	Media
Solicitud	0	Ninguno	Baja	Alta	Baja
Algoritmo	0	Ninguno	Baja	Alta	Baja
LN_Servir	1	Bajo	Media	Media	Media
Solicitud	0	Ninguno	Baja	Alta	Baja
Acceso_Datos_Solicitud	1	Bajo	Media	Media	Media
Acceso_A_Solicitud_Algoritmo	3	Alto	Alto	Baja	Alto
Acceso_Solicitud	1	Bajo	Media	Media	Media
Acceso_Datos	2	Medio	Alto	Baja	Alto
AccesoDato	1	Bajo	Media	Media	Media

Tabla 9: Instrumento de evaluación de la métrica RC

Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos establecidos:

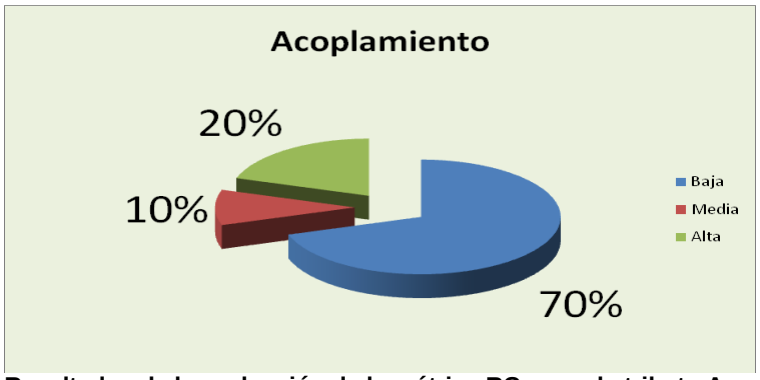


Tabla 10: Resultados de la evaluación de la métrica RC para el atributo Acoplamiento

Representación en % de la incidencia de los resultados obtenidos en el atributo Complejidad de mantenimiento:

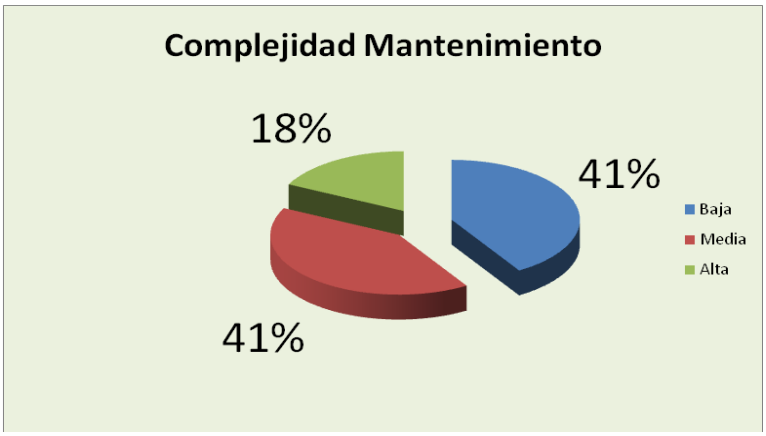


Tabla 11: Resultados de la evaluación de la métrica RC para el atributo Complejidad de Mantenimiento.

Representación en % de la incidencia de los resultados obtenidos en el atributo Reutilización:

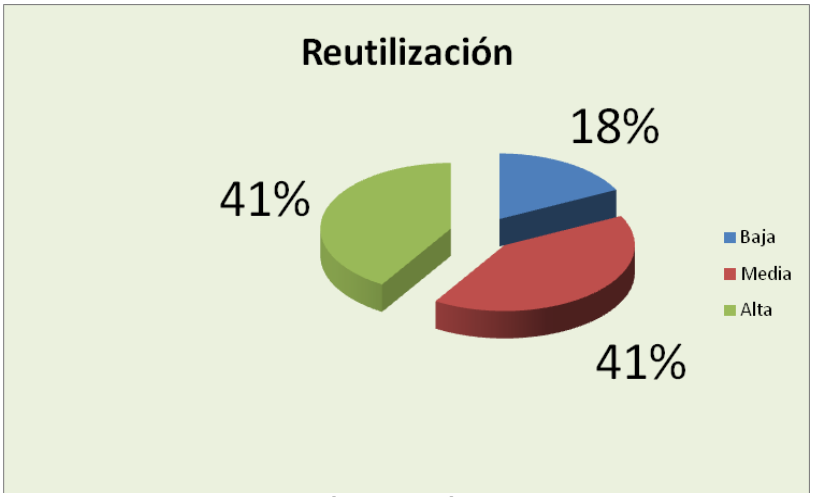


Tabla 12: Resultados de la evaluación de la métrica RC para el atributo Reutilización.

Representación en % de la incidencia de los resultados obtenidos en el atributo Cantidad de pruebas.

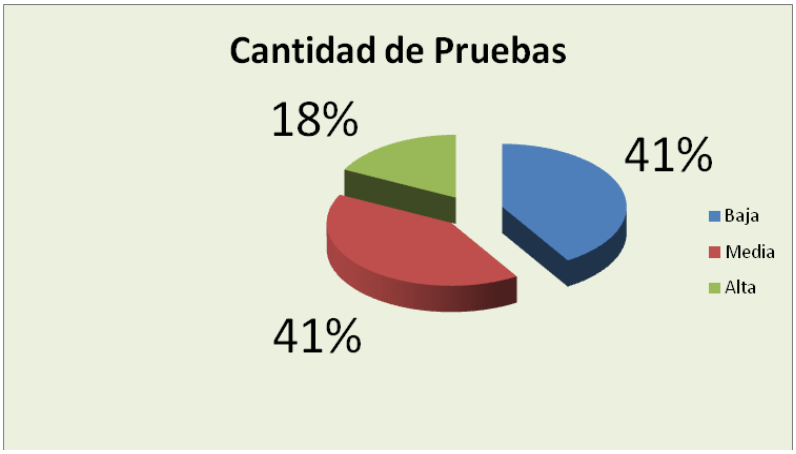


Tabla 13: Resultados de la evaluación de la métrica RC para el atributo Cantidad de Pruebas.

Una vez obtenidos los resultados de la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño propuesto tiene una calidad aceptable teniendo en cuenta que el 88,2 % de las clases empleadas posee menos 3 de dependencias de otras clases lo que conlleva a evaluaciones positivas de los atributos de calidad involucrados (acoplamiento, complejidad de mantenimiento, cantidad de pruebas y reutilización).

3.3 Conclusiones:

En el presente capítulo se validaron los requisitos obtenidos durante la descripción de la solución a través de la métrica Especificación no ambigua, demostrando que todas las personas involucradas en el proceso de revisar los requisitos coincidieron con la interpretación de los mismos. También se emplearon las métricas Tamaño Operacional de las Clases y Relaciones entre Clases para evaluar la calidad del diseño propuesto arrojando como resultado alta reutilización de las clases provocando un bajo acoplamiento y una alta cohesión considerando que el diseño se encuentra dentro de los límites de calidad.

CONCLUSIONES GENERALES:

Una vez finalizado el presente trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos:

- Se realizó un estudio detallado referente a los laboratorios virtuales con el objetivo de obtener una comprensión más explícita del tema.
- Se especificaron los casos de uso del sistema y se utilizaron patrones para lograr un mejor entendimiento del mismo.
- Se obtuvieron los requerimientos que debe cumplir el sistema identificándose los requisitos funcionales y no funcionales.
- Se especificaron los casos de uso del sistema y se utilizaron patrones para lograr un mejor entendimiento del mismo.
- Se realizó el análisis y el diseño para el módulo de Ejercicios de Planificación de Disco y para un correcto diseño de clases se emplearon los patrones de diseño.
- Se realizó la validación de los requisitos y del diseño evidenciándose que ambos cumplen con las necesidades del cliente.

RECOMENDACIONES

- ✓ A partir de los artefactos generados se recomienda realizar la implementación de las funcionalidades propuestas en las próximas fases del ciclo de vida.

- ✓ Integrar el módulo junto con los restantes que componen el laboratorio virtual.

REFERENCIAS BIBLIOGRÁFICAS

- 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 2003. [Citado el: 9 de abril de 2011.] Disponible en: http://www.disc.ua.es/docenweb/
- I Jacobson, G Booch, J Rumbaugh. 2000. El Proceso Unificado de Desarrollo de Software. La Habana, Editorial Addison Wesley Longman, 2000, Volumen 1.
- EVA, 2011, Actividad 23. Software de Entrada/Salida. Manejadores de dispositivo. [Citado el 20 de abril 2011.]
- [PIA96], Modulo III Análisis del Sistema. [Citado 21 de abril 2011.] Disponible en: www.eduardoleyton.com/apuntes/Resumen_DFD.pdf
- Larman, Craig Larman, UML y Patrones. Introducción al análisis y diseño orientado a objetos, 2004.
- C1, Conferencia 1: Continuación de la Disciplina Análisis de Diseño, Ingeniería de Software 2.[Citado 2 de mayo 2011] http://eva.uci.cu/course/view.php?id=259
- Félix Prieto, 2005, Programación III I.T.I Sistemas, Patrones de diseño, Félix Prieto Curso 2004/2005. [Citado 25 de mayo]
- Evolución de los Sistemas Operativos, sitio Aeromental 2011, artículo del 16 de agostos 2006. Disponible en: http://www.aeromental.com/2006/08/16/evolucion-de-los-sistemas-operativos/ [Citado 5 de abril 2011.]
- Ames, Lowa. Informe de la reunión de expertos sobre laboratorios virtuales, organizada por el Instituto Internacional de Física Teórica y Aplicada (IITAP), Ames, Iowa, con el apoyo de la UNESCO/ preparado por James P. Vary, compilador. – París: UNESCO. 2000. [Citado 9 de abril 2011.] Disponible en: http://unesdoc.unesco.org/images/0011/001191/119102s.pdf

- Erly Delgado Exposito, Metodologías de desarrollo de software. [Citado el 10 de abril 2011.] Disponible en: http://www.monografias.com/trabajos60/metodologias-desarrollo-software/metodologias-desarrollo-software.shtml
- Hernando, Roberto Hernando, 9/3/2004, Metodologías de desarrollo de Software. [Citado el 10 de abril de 2011.] http://www.rhernando.net/modules/tutorials/viewexttutorial.php?tid=15
- Fierro, Oliver Fierro, mayo 2010, Metodologías ágiles o robustas. [Citado 11 de abril de 2011.]. Disponible en: http://www.nexon.cl/nx/index.php/lacolumnade/8-columnaofierro/10-columnaadmproy01.html
- Herramientas CASE para el proceso de desarrollo de Software, Evelyn Menéndez Alonso 2009, [En línea 10 de agosto de 2009], [Citado 13 de abril de 2011.]. Disponible en: http://www.monografias.com/trabajos73/herramientas-case-proceso-desarrollo-software/herramientas-case-proceso-desarrollo-software2.shtml
- Rodolfo Quispe-Otazu. ¿Qué es la Ingeniería de Requerimientos? Blog de Rodolfo Quispe-Otazu [Internet]. Agosto 2007. [Citado 13 de abril de 2011.]Disponible en: http://www.rodolfoquispe.org/blog/que-es-la-ingenieria-de-requerimientos.php
- Nicolás Tedeschi, ¿Qué es un Patrón de Diseño?, [Citado 2 de mayo de 2011]. http://msdn.microsoft.com/es-es/library/bb972240.aspx#mainSectin
- Patrones de casos de uso. [Citado el 2 de mayo de 2011. Disponible en:] http://www.ecured.cu/index.php/Patrones_de_Casos_de_Uso
- Juan Pablo Torres Herrera, Odin Isaac Meling Lopez, Juan Iván Nieto Hipólito, Laboratorios Virtuales de computación para cursos en línea, [En línea 2006], [Citado 18 de mayo de 2011]. Disponible en: http://www.cibersociedad.net/congres2006/gts/comunicacio.php?id=1047

- Artículo: Aprende, Practica y Evalúa Mediante los Laboratorios Virtuales de TechNet y MSDN, [En línea 15 de diciembre de 2010], [Citado 18 de mayo de 2011]. Disponible en: http://optimustechpr.com/index.php/category/articulos-de-interes/
- Laboratorio Virtual de Física, Universidad Central "Marta Abreu" de las Villas. [Citado el 18 de mayo del 2011]. Disponible en: http://www.quadernsdigitals.net/datos_web/hemeroteca/r_1/nr_502/a_6861/6861.htm
- Laboratorio Virtual de Química General en la Universidad de La Habana, [Citado el 18 de mayo del 2011] http://rosaura08.blogspot.com/2008/06/laboratorio-virtual-de-qumica-general.html, se puede encontrar en la URL "URL: http://www.fq.uh.cu/departamentos/qg/laboratorio.html,"
- Laboratorio Virtual de Anestesiología, [Citado el 18 de mayo del 2011], Disponible en: http://scielo.sld.cu/scielo.php?pid=S0034-75072001000100002&script=sci_arttext
- Tesis de Análisis del módulo de Gestión de Entrada y Salida para el Laboratorio Virtual de Sistemas Operativos, diciembre 2010.
- José Manuel Ruiz Gutiérrez, 2000, La Simulación como instrumento de aprendizaje. [Citado 18 de mayo del 2011]. Disponible en: http: //mami.uclm.es/jmruiz/materiales/Documentos/simulación.PDF.Simulación,
- Doc. Marcela Printista, Simulación. Disponible en: http://sites.google.com/site/simulacionunsl.
- Visual Paradigm for UML. [En línea 2007]. [Citado 18 de junio de 2011]. Disponible en: http://www.freedownloadmanager.ogr/es/downloads/Paradigma_Visual_para_UML_(MÍ)_14720_p/
- Overgaard, Gunnar, Palmkvist, Karin, ¨Use Cases: Patterns and Blueprints¨. 2004. Addison Wesley.
- Pressman, Roger Pressman, 2005, Ingeniería de Sostfware. Un enfoque práctico. Parte ll. Ciudad de la Habana: Felix Varela.
- Conferencia 10: Fase de elaboración. Disciplina de Análisis y Diseño. Modelo de análisis, Modelo conceptual. Diagramas Clases y diagramas de interacción.,Introducción a la Disciplina de Análisis y Diseño, Ingeniería 1, EVA cursos 2010/2011.

- Silvia Aramúndiz, Prof. De Laboratorio de Ing. de SW, Diagrama de Secuencia, EVA curso 2010/2011.Clase práctica 9: Consolidación de diagramas de interacción del análisis.
- Roles, M.C. Esperanza Aguillón. Métricas para la Gestión de Proyectos de Software
- ¿ Qué es SCRUM ? [En línea 2010], proyectosagiles.org. Disponible en: http://www.proyectosagiles.org/que-es-scrum
- Rational Rose Enterprise Edition. Disponible en: http://mundolibre10.blogspot.com/2010/03/rational-rose-enterprise-edition-suite.html
- 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.

GLOSARIO DE TÉRMINOS

Planificador: El planificador (o Scheduler en inglés) es un componente funcional muy importante de los sistemas operativos multitarea y multiproceso, y es esencial en los sistemas operativos de tiempo real. Su función consiste en repartir el tiempo disponible de un microprocesador entre todos los procesos que están disponibles para su ejecución.

UML: Lenguaje Unificado de Modelado (LUM o UML, por sus siglas en inglés, *Unified Modeling Language*) es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

GNU GPL: La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente sus siglas del inglés GNU GPL, es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios

Software: equipamiento lógico o soporte lógico de una computadora digital; comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos, que son llamados hardware.

Linux: Es un sistema operativo creado por Linus Torvald y el cual es totalmente libre. Su código fuente puede ser accedido, modificado y redistribuido por cualquiera bajo los términos de la GPL.

sistema operativo: Un sistema operativo (SO) es el 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.

simulación: 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.

Unix: Unix (registrado oficialmente como UNIX®) es un sistema operativo portable, multitarea y multiusuario; desarrollado, en principio, en 1969 por un grupo de empleados de los laboratorios Bell de AT&T, entre los que figuran Ken Thompson, Dennis Ritchie y Douglas McIlroy.

Windows: Microsoft Windows es el nombre de una serie de sistemas operativos desarrollados por Microsoft desde 1981, año en que el proyecto se denominaba *«Interface Manager»*.

base de datos: Una base de datos o banco de datos (en ocasiones abreviada con la sigla *BD* o con la abreviatura *b. d.*) es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

Web: Un sitio web es un gran espacio documental organizado que la mayoría de las veces está típicamente dedicado a algún tema particular o propósito específico. Cualquier sitio web puede contener hiperenlaces a cualquier otro sitio web, de manera que la distinción entre sitios individuales, percibido por el usuario, puede ser a veces borrosa.

Moodle: Moodle es un Ambiente Educativo Virtual, sistema de gestión de cursos, de distribución libre, que ayuda a los educadores a crear comunidades de aprendizaje en línea. Este tipo de plataformas tecnológicas también se conoce como LMS (Learning Management Systems).

Microsoft: Microsoft Corporation (NASDAQ: MSFT) es una empresa multinacional de origen estadounidense, fundada el 4 de abril de 1975 por Bill Gates y Paul Allen. Dedicada al sector de la informática, con sede en Redmond, Washington, Estados Unidos. Microsoft desarrolla, fabrica, licencia y produce software y equipos electrónicos. Siendo sus productos más usados el sistema operativo Microsoft Windows y la suite Microsoft Office, los cuales tienen una importante posición entre los ordenadores personales. Con una cuota de mercado cercana al 90% para Office en 2003 y para Windows en el 2006. Siguiendo la estrategia de Bill Gates de *"tener una estación de trabajo que funcione con nuestro software en cada escritorio y en cada hogar"*

TechNet: Microsoft TechNet es un programa de Microsoft y los recursos de información técnica, noticias y eventos para profesionales de IT. Junto con un sitio web, también producen una suscripción mensual de la revista titulada "Revista de TechNet".

MSDN: MSDN (del Inglés: *Microsoft Developer Network*) puede referirse tanto a los servicios web orientados a desarrolladores de software basado en plataformas Microsoft como al conjunto de software que se adjunta con sus compiladores (Visual Studio) y ciertos SDK. Tiene como objeto la resolución de dudas y problemas que le puedan surgir al desarrollador; según la propia corporación "*Contiene una gran cantidad de información técnica de programación, incluidos código de ejemplo, documentación, artículos técnicos y guías de referencia*", en este último caso la denominación correcta sería "*API o biblioteca MSDN*", de la cual también se encuentra una versión en línea.

Plantilla: Con relación con los sistemas computacionales, por ejemplo paquetes de programas basados en la web, utilizan en la actualidad un sistema de plantillas para separar la lógica del programa del

formato visualizado. Típicamente, estas plantillas incluirán variable (frecuentemente denotadas como {VARIABLE}), y posiblemente unos pocos operadores lógicos para permitir una mejor adaptabilidad de la plantilla.

RUP: El Proceso Racional Unificado (*Rational Unified Process* en inglés, habitualmente resumido como RUP) es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

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.

IDE: Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Python, Java, C#, Delphi, Visual Basic, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto, como es el caso de Smalltalk u Objective-C.

POO: La programación orientada a objetos o POO (OOP según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas informáticos. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de los años 1990. En la actualidad, existe variedad de lenguajes de programación que soportan la orientación a objetos.

Hardware: Hardware (pronunciación AFI: /ˈhɑːdˌwɛə/ ó /ˈhɑɹdˌwɛə˞/) corresponde a todas las partes tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y

mecánicos; 1 sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado; contrariamente, el soporte lógico es intangible y es llamado software. El término es propio del idioma inglés (literalmente traducido: partes duras), su traducción al español no tiene un significado acorde, por tal motivo se la ha adoptado tal cual es y suena; la Real Academia Española lo define como «Conjunto de los componentes que integran la parte material de una computadora».2 El término, aunque es lo más común, no solamente se aplica a una computadora tal como se la conoce, ya que, por ejemplo, un robot, un teléfono móvil, una cámara fotográfica o un reproductor multimedia también poseen hardware (y software).

Procesador: El microprocesador, o simplemente procesador, es el circuito integrado central y más complejo de una computadora u ordenador; a modo de ilustración, se le suele asociar por analogía como el "cerebro" de una computadora.

Pentium: Intel Pentium es una gama de microprocesadores de quinta generación con arquitectura x86 producidos por Intel Corporation.

Netbeans: NetBeans es un entorno de desarrollo, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

Interfaz: La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.

TIC: Tecnología de la Información y las Comunicaciones.