Universidad de las Ciencias Informáticas

Facultad 15

"Análisis y Diseño del módulo Análisis de Riesgo de Crédito del proyecto SAGEB"

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

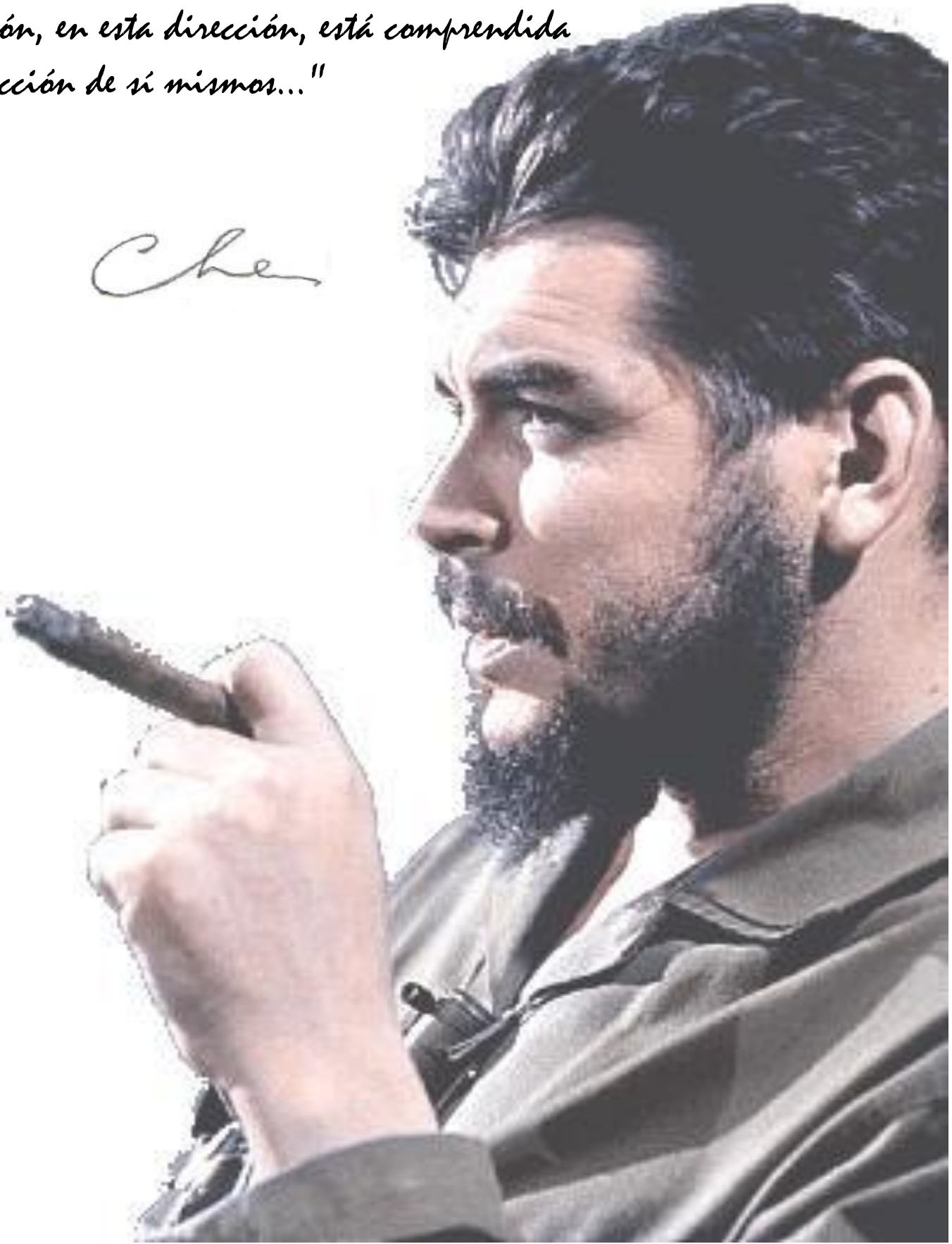
Autor(es): Yaniuska García Romero

Denise Fernández Miranda

Tutor(es): Ing. Yoan Antonio López Rodríguez

La Habana, Cuba
junio 2010

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



DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo titulado: *Análisis y Diseño del módulo Análisis de Riesgo de Crédito del proyecto SAGEB* y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Denise Fernández Miranda

_____

Firma del Autor

Yaniuska García Romero

_____

Firma del Autor

Yoan Antonio López Rodríguez

_____

Firma del Tutor

DATOS DE CONTACTO

Ing. Yoan Antonio López Rodríguez.

Ingeniero en Ciencias Informáticas, graduado en Julio del 2008, Título de Oro, adiestrado. En la producción ejerce como Jefe de Equipo en el Proyecto SAGEB perteneciente al Dpto. de Soluciones Financieras del Centro CESGE. Docentemente se desempeña como profesor de Teleinformática de la Facultad 15.

AGRADECIMIENTOS

Compartidos:

Agradecer a nuestro comandante en jefe Fidel Castro Ruz y a la Revolución Cubana por hacer realidad este bello sueño que es la UCI y por dejarnos formar parte de ella.

A nuestro tutor por su paciencia, ayuda y dedicación y por estar pendiente del correcto desempeño de esta tesis.

A todos los que colaboraron y nos dieron la mano, a Yadira Calimano, Bello, Tania, Yadian, Liset, Daniellys y Arasay, a los compañeros del banco.

A nuestros padres y amigos por ayudarnos a ser mejores personas, por comprendernos y apoyarnos gracias.

DEDICATORIA

De Yaniuska:

A mi mamá y a mi hermano por confiar en mí, por ser este momento uno de sus mayores sueños, por apoyarme y a la memoria de mi papá.

De Denise:

A mi mamá y mi papá por ser el motor que impulsa mi vida, por su ejemplo, apoyo y comprensión a la realización de este sueño que es suyo también, a todos los que creyeron en mí.

RESUMEN

En la actualidad los bancos necesitan sistemas informáticos perfeccionados para llevar a cabo sus procesos. Por su parte la Gerencia de Riesgo del Banco Nacional de Cuba carece de un sistema informático que ayude a determinar la factibilidad del otorgamiento de crédito a clientes.

El presente Trabajo de Diploma realiza el análisis y diseño de un sistema automatizado para el Análisis de Riesgo de Crédito a clientes, ajustándose a los lineamientos arquitectónicos establecidos dentro del proyecto SAGEB.

Como parte del desarrollo se llevó a cabo la Ingeniería de Requerimientos, la misma comprendió las actividades Elicitación, Especificación y Validación de los requisitos y permitió definir los requerimientos del sistema. La herramienta utilizada durante todo el desarrollo fue Visual Paradigm y los artefactos se generaron usando como lenguajes de modelado el Bussines Process Manager Notation y el Lenguaje Unificado de Modelado. Se aplicaron un grupo de patrones en el diseño definidos por el proyecto. Se utilizó la metodología Proceso Unificado de Desarrollo.

Mediante el flujo de trabajo de análisis y diseño se obtuvieron artefactos entendibles que viabilizan a los implementadores la automatización de una plataforma capaz de satisfacer las necesidades de los clientes.

PALABRAS CLAVE

Procesos bancarios, requerimientos, riesgo, crédito, análisis, diseño.

TABLA DE CONTENIDO

AGRADECIMIENTOS 3

DEDICATORIA 4

RESUMEN 5

INTRODUCCIÓN 8

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 11

1.1 Introducción 11
1.2 Ingeniería de Software 11
1.2.1 Ingeniería de Requerimientos (IR) 12
1.2.1.1 Elicitación de Requisitos (ER) 13
1.2.1.2 Especificación de Requisitos (ER) 15
1.2.1.3 Validación de Requisitos (VRE) 17
1.2.2 Metodologías de Desarrollo de Software 18
1.2.3 Herramienta CASE 21
1.2.4 Lenguaje de modelado 22
1.2.5 Framework 23
1.2.6 Lenguaje de programación 26
1.2.7 Entorno de desarrollo integrado (IDE) 26
1.3 Sistema Bancario 27
1.3.1 Procesos Bancarios 28
1.3.1.1 Análisis de Riesgo Financiero 28
1.3.2 Sistema Bancario Cubano 31
1.4 Sistema para el Análisis de Riesgo de Crédito 31
1.5 Conclusiones Parciales 32
CAPÍTULO 2: ELICITACIÓN, ESPECIFICACIÓN Y VALIDACIÓN DE REQUISITOS. 34

2.1 Introducción 34
2.2 Elicitación de requisitos (ER) 34
2.2.1 Modelación del Negocio del módulo Análisis de Riesgo de Crédito 34
2.2.1.1 Involucrados 35
2.2.1.2 Trabajadores 35
2.2.1.3 Artefactos 35
2.2.1.4 Resumen de los procesos del negocio 36

2.2.1.5 Reglas del Negocio....37
2.2.1.6 Mejoras de los procesos a automatizar....38
2.3 Requisitos Funcionales del módulo Análisis de Riesgo de Crédito....38
2.4 Diagrama de Casos de Uso del Sistema....39
2.5 Descripción de los casos de uso del módulo Análisis de Riesgo de Crédito....40
2.6 Validación de Requisitos de Software....48
2.7 Indicadores Contables....48
2.8 Conclusiones Parciales....53
CAPÍTULO 3: ANÁLISIS Y DISEÑO....54
3.1 Introducción....54
3.2 Análisis....54
3.2.1 Modelo del análisis....54
3.2.1.1 Clases del análisis....55
3.2.1.2 Diagrama de clases del análisis....55
3.2.1.3 Diagrama de interacción (colaboración)....56
3.3 Diseño....57
3.3.1 Descripción de la arquitectura propuesta....58
3.3.1.1 Patrones a utilizar....60
3.3.2 Modelo del diseño....62
3.3.2.1 Clases del diseño....62
3.3.2.2 Diagramas de clases del diseño....63
3.3.2.3 Diagramas de interacción (secuencia)....67
3.3.3 Modelo de Datos....68
3.4 Conclusiones parciales....69
CONCLUSIONES....70
RECOMENDACIONES....71
ANEXOS....76
GLOSARIO DE TÉRMINOS....79

INTRODUCCIÓN

Las ciencias informáticas constituyen en la actualidad un tema apasionante en todos los sentidos, su estudio conlleva a pensar en cómo mejorar una organización, una industria, un país. La Informática es una herramienta útil a todas las materias y de enormes posibilidades de desarrollo, es por esto que resulta de un valor inestimable en muchos campos de la industria, las ciencias y la economía.

Una consecuencia del desarrollo y la complejidad alcanzados por la actividad económica en las últimas décadas, ha sido el perfeccionamiento de los sistemas informáticos, así como su evolución y aplicación en las diferentes esferas. Un banco es una institución financiera que se encarga de administrar y prestar dinero, ya sea a otras instituciones o a la población en general. Ellos, controlan la vida económica del mundo y su actuación estratégica se ha potenciado en las últimas décadas con el uso de los sistemas informáticos, los cuales han contribuido a lograr una mayor eficiencia en la toma de decisiones y en el resto de sus operaciones. Tales actividades cuentan con un factor de riesgo que desde el punto de vista general, es la posibilidad de sufrir un daño, una situación potencial de daño, que puede producirse o no. Existen metodologías para la administración de riesgos en sus distintas fases, siendo estos: Riesgo de Crédito, Riesgo de Liquidez, Riesgo de Mercado y Riesgo Operativo.

El presente trabajo de diploma se centrará en el Riesgo de Crédito, que se define como la volatilidad de los ingresos debido a pérdidas potenciales en crédito por falta de pago de un acreditado o contraparte, en otras palabras es el riesgo de que los clientes no cumplan con sus obligaciones de pago. Controlar la capacidad financiera de los clientes y analizar la factibilidad del otorgamiento de créditos y garantías son algunas de las actividades desarrolladas por las entidades bancarias a la hora de llevar a cabo este tipo de análisis.

El Banco Nacional de Cuba como entidad bancaria no presenta ningún sistema automatizado para llevar a cabo el Análisis de Riesgo de Crédito a clientes, el mismo es realizado por los operarios que apoyados en tablas Excel, manejan la información recogida de los estados financieros solicitada a los clientes. En estos momentos se desarrolla en la Universidad de las Ciencias Informáticas, específicamente en el Departamento de Soluciones Financieras un sistema informático que se implantará una vez terminado, en el Banco Nacional de Cuba. Como parte del desarrollo se pretende realizar un módulo para el Análisis de

Riesgo de Crédito a clientes. Para su implementación se necesita contar con un diseño previo. Tomará como entradas la información de los estados financieros de los clientes y tendrá como salida la factibilidad del otorgamiento del crédito.

Dada la situación problemática anterior, podemos identificar el siguiente **problema a resolver:** No se cuenta con el diseño del módulo Análisis de Riesgo de Crédito para el proyecto SAGEB.
El **objeto de estudio** del presente trabajo se enfocará hacia los Sistemas para el Análisis de Riesgo de Crédito en entidades bancarias a nivel mundial y el **campo de acción** hacia los procesos de Análisis de Riesgo de Crédito en el Banco Nacional de Cuba.
Para responder al problema anteriormente planteado, los autores se trazaron como **objetivo general:** Modelar el diseño del módulo Análisis de Riesgo de Crédito para el proyecto SAGEB.
Para cumplir lo planteado, se llevaron a cabo las siguientes **Tareas**:

1. Analizar el estado del arte acerca de la Ingeniería de Requerimiento.
2. Caracterizar la metodología RUP.
3. Caracterizar los lenguajes BPMN y UML.
4. Caracterizar la herramienta Visual Paradigm.
5. Analizar Sistemas automatizados existentes en el mundo en Análisis de Riesgo de Crédito.
6. Analizar los procesos de Análisis de Riesgo en el Banco Nacional de Cuba.
7. Desarrollar la Ingeniería de Requerimiento.
8. Caracterizar los patrones de diseño a utilizar.
9. Modelar el diseño del módulo Análisis de Riesgo de Crédito para el proyecto SAGEB.

Como **posible resultado:** Obtener el Modelo de Diseño del módulo Análisis de Riesgo de Crédito para el proyecto SAGEB.

Para lograr la comprensión y claridad de los contenidos de la investigación realizada se ha estructurado el documento en tres capítulos.

Capítulo 1. Fundamentación Teórica

En este capítulo se abordan temas de vital importancia para la investigación, pudiendo señalar lo referente al estudio de la metodología que guiará el proceso de desarrollo de software, las herramientas a utilizar y el estado del arte del tema relacionado a los sistemas para el Análisis de Riesgo existentes en el mundo.

Capítulo 2. Elicitación, Especificación y Validación de Requisitos del Sistema.

Se aborda el entorno donde se implantará el sistema y se proponen los requisitos que el módulo deberá tener. Se realiza la especificación de requisitos y se validan. Además, se obtienen los ratios contables que ayudarán a determinar la factibilidad del otorgamiento de crédito.

Capítulo 3. Análisis y Diseño del Sistema.

En este capítulo se modelan los diagramas de clases del análisis y del diseño, así como los diagramas de interacción correspondientes, y se abordan conceptos de gran importancia para un mejor entendimiento del tema.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el presente capítulo se define el conjunto de actividades que guían los esfuerzos de los autores que tienen la misión de ya transformadas las necesidades de los usuarios en requerimientos, realizar el análisis y el diseño para la posterior implementación del módulo Análisis de Riesgo de Crédito. Incluyéndose además, un estudio relacionado con el análisis de riesgo de crédito en entidades bancarias y las herramientas a utilizar con el fin de lograr de forma productiva los objetivos propuestos.

1.2 Ingeniería de Software

Durante los primeros años de la informática, no existía una disciplina en el desarrollo del software. La programación era considerada un "arte", y no se controlaba que los ingenieros utilizaran de forma consistente los métodos. El proceso de desarrollo de un sistema se realizaba sin planificación alguna (Menéndez Barzanallana, 2008). La evolución y perspectivas de la Ingeniería del Software se caracterizaron, desde mediados de la década de los 60 hasta finales de los 70 por el establecimiento del software como un producto que se desarrollaba para una distribución general. En esta época nació lo que se conoce como el mantenimiento del software que se realiza cuando cambian los requisitos de los usuarios y se hace necesaria la modificación del mismo. El esfuerzo necesario para realizar este mantenimiento era en la mayoría de los casos tan elevado que se hacía imposible llevarlo adelante. En la actualidad en la Industria de Software hay tendencia al crecimiento del volumen y complejidad de los productos, los proyectos están excesivamente tardes, se exige mayor calidad y productividad en menos tiempo y hay insuficiente personal calificado; por lo que se puede decir que las fallas de los proyectos de software se deben a:

•Planificación irreal: Los usuarios piden un sistema para hoy que tenga costo cero y los ingenieros no son capaces de enfrentar un plan porque no están entrenados para usar métodos de planificación y, frecuentemente, las estimaciones no se basan en datos reales.

•Mala calidad del trabajo: Las prácticas pobres de ingeniería, la carencia de métricas adecuadas de calidad y las decisiones de los directivos guiadas por una planificación irreal; traen como consecuencia tiempos de pruebas impredecibles, productos con muchos defectos, demoras en la aceptación de los

usuarios y una extensa garantía de servicios y reparaciones. Una pobre calidad afecta la planificación y torna ineficiente el proceso de prueba.

•Personal inadecuado: En múltiples ocasiones el personal asignado a un proyecto se incorpora tarde, no cubre las necesidades en cuanto a cantidad y calidad y se incorpora a tiempo parcial al proyecto. Como consecuencia el trabajo se demora o descuida, es ineficiente y sufre la moral del equipo. Con independencia del plan, los proyectos deben comenzar en tiempo y con todo el personal.

•Cambios no controlados: Es importante recordar que siempre ocurren cambios en los requerimientos, que los planes del proyecto se basan en el alcance del trabajo conocido, que los cambios siempre requieren más trabajo, sin planes detallados los equipos no pueden estimar el efecto o magnitud de los cambios y si los equipos no controlan cada cambio, se pierde gradualmente el control del plan del proyecto (Fernández Pérez, 2007).

Para enfrentar esta situación las empresas requieren desarrollar o adquirir una disciplina en el desarrollo del software y controlar que los ingenieros usen de forma consistente los nuevos métodos. La Ingeniería de Software es una tecnología multicapa en la que se pueden identificar: los métodos (indican cómo construir técnicamente el software), el proceso (es el fundamento de la Ingeniería de Software, es la unión que mantiene juntas las capas de la tecnología) y las herramientas (soporte automático o semiautomático para el proceso y los métodos). (Pressman, 2005) **[Anexos, Figura1]**

La industria del software debe asentar cada vez más sus bases en la Ingeniería de Software, proporcionando la investigación continua para la práctica y estandarización de sus políticas, con el objetivo de que sus productos se realicen con calidad y que respondan a los requerimientos definidos.

1.2.1 Ingeniería de Requerimientos (IR)

Algunos de los principales problemas que persisten en el desarrollo de software, tienen que ver con la concepción de los requerimientos del sistema, entre ellos encontramos: un inadecuado entendimiento de las necesidades de los usuarios, incapacidad de absorber cambios en los requisitos e insatisfacciones de los clientes por inaceptable o bajo desempeño del software. Se considera que las principales causas de

ello son: la administración insuficiente de requisitos, la incorrecta comunicación, las inconsistencias no detectadas entre requisitos, las validaciones tardías de requisitos, el enfrentamiento reactivo de riesgos y la propagación de cambios sin control.

Un requerimiento es:
1. Condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo.
2. Condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de un sistema para satisfacer un contrato, estándar, u otro documento impuesto formalmente.
3. Una representación documentada de una condición o capacidad como en 1 ó 2.

La determinación de los requisitos de software constituye la base para las etapas futuras de desarrollo. Todos los integrantes del equipo a partir de aquí conocen cuales son las funcionalidades que el sistema deberá tener y no solo las funcionalidades, ya que en la determinación de los requisitos se tienen en cuenta además las propiedades que el cliente desea para su sistema, de ahí que se separen los requisitos en dos grandes grupos: los funcionales y los no funcionales. Los requisitos funcionales se definen como las capacidades o condiciones que el sistema deberá cumplir o los servicios que deberá brindar y los no funcionales, se definen como las propiedades o cualidades, que deberá tener el software o de qué manera se brindarán esos servicios**.**

Si el documento de requerimientos es el producto final de la etapa de requerimientos, entonces la IR puede pensarse como el proceso por el cual el documento de requerimientos es generado y publicado. Muchos autores han propuesto estrategias para llevar a cabo la IR, en esta investigación el desarrollo de la IR se basará en el desarrollo de tres de las actividades propuestas por Pressman: Elicitación, Especificación y Validación de los Requisitos

1.2.1.1 Elicitación de Requisitos (ER)

El proceso mediante el cual se identifican los ítems de información que determinan las características deseadas y las restricciones que deberá satisfacer el sistema software es también conocido como elicitación de requisitos.

La ER proporciona la interacción con los clientes y expertos del negocio con el objetivo de crear las primeras relaciones entre ellos y el equipo de desarrollo, y así obtener la información necesaria para determinar las necesidades reales de la organización, tanto explícitas como implícitas y el dominio del problema.

Antes de identificar los requisitos es necesario conocer el ambiente, incluyendo los sistemas actuales tanto manuales como informatizados, sus aspectos negativos y positivos, además de conocer los procesos que están implicados en la organización en la que se aplicará el estudio.

Técnicas de la Elicitación de Requisitos (TER)

Producto de la experiencia obtenida en el desarrollo de la actividad Elicitación de Requisitos los desarrolladores de esta rama han definido un conjunto de técnicas que ayudan a los analistas a llevarla a cabo:

1. Estudio de la documentación: depende de la información que exista en las entidades sobre los procesos y la terminología que se maneja en la misma. Esta técnica es utilizada para capturar requisitos, que luego deben ser validados por otras técnicas para su comprobación pues no se considera efectiva por sí sola.
2. Glosario de términos: Se almacenan los términos y significados relativos al problema, esta técnica intenta registrar el conocimiento obtenido del dominio del problema y compartirlo con todos los participantes. El glosario puede ser tan auto contenido como sea posible, en donde los términos estén relacionados entre sí, a esta técnica se le conoce como Principio de Circularidad.
3. Modelado de Negocio: En el modelo de negocio se describen los procesos del negocio y es fundamental para la comprensión general de los mismos.
4. Entrevistas: Es la más utilizada, ya que es la forma natural de entendimiento de los humanos. Es importante conocer el vocabulario del dominio del problema para lograr un buen entendimiento con el cliente.
5. Join Aplication Development (JAD): Se refiere al Desarrollo de Aplicaciones en conjunto y es una alternativa de la entrevista, se desarrolla durante varios días y el cliente participa junto con el equipo de desarrollo dando opiniones y posibles soluciones. Esta técnica intenta atraer más al cliente para que no se sienta fuera del equipo, se apoya en el uso de técnicas audiovisuales, diagramas, herramientas Case y organización. Tiene como ventaja que los clientes pueden revisar

la documentación que se genera, sin embargo, es muy compleja de organizar y depende de los horarios de los clientes y usuarios.

A pesar de existir muchas técnicas para desarrollar la ER, por la naturaleza del proyecto se han decidido combinar en el presente trabajo las técnicas: estudio de la documentación, entrevistas, JAD y modelo de negocio.

Con la ER se descubren y capturan los requisitos que deberán ser implementados más adelante. Esta actividad se considera más humana que técnica, donde se identifica a los interesados y se establecen las primeras relaciones entre ellos y el equipo de trabajo.

1.2.1.2 Especificación de Requisitos (ER)

También conocida como la definición de los requisitos, es la manera habitual de guardar y comunicar los requisitos encontrados. El objetivo que se persigue es obtener un documento de especificación (ERS) que defina los requisitos tanto funcionales como no funcionales, que debe cumplir el sistema, así como las restricciones aplicables al diseño (software y hardware), abordando la descripción de qué hay que desarrollar y no cómo ni cuándo.

La obtención de una ERS de alta calidad es fundamental para asegurar que el software se corresponde con las necesidades del cliente. Sin embargo, obtener una ERS de calidad es difícil.
La especificación describe las funciones y características de un sistema de computación y las restricciones que gobiernan su desarrollo (Pressman, 2005).
Para las ERS se han definido una serie de características deseables, dentro de las cuales se tienen las siguientes:

a) Correcta: Todo requisito en la ERS contribuye a satisfacer una necesidad real.
b) No ambigua: Todo requisito posee una sola interpretación.
c) Completa: Todo lo que se supone que el software debe hacer está incluido en la ERS.
d) Internamente Consistente: No existen subconjuntos de requisitos contradictorios.

e) Verificable: Para cada requisito expresado en la ERS existe un procedimiento de prueba finito y no costoso para demostrar que el futuro sistema lo satisface.
f) Modificable**:** Su estructura y estilo permiten hacer cualquier cambio a los requisitos fácilmente.
g) Precisa: La ERS hace uso de valores numéricos para precisar las características del sistema.
h) Realizable: Si dados los actuales recursos, la ERS se puede implementar.
i) Concisa: La ERS debe ser lo más breve posible sin afectar a los atributos de calidad.

Técnicas de la Especificación de Requisitos (TERS)

Para desarrollar la ERS se proponen un gran número de técnicas, dentro de ellas las más significativas son las siguientes:
a) Glosario de términos: Se recogen y definen los conceptos más relevantes y críticos para el sistema y permanece accesible a todos los implicados en el proyecto.
b) Plantillas: Se describen los requisitos mediante el lenguaje natural pero de una forma estructurada. Una plantilla es una tabla con una serie de campos y una estructura predefinida que el equipo de desarrollo va cumplimentando usando para ello el lenguaje del usuario. Las plantillas eliminan parte de la ambigüedad del lenguaje natural al estructurar la información; cuanto más estructurada sea esta, menos ambigüedad ofrece.
c) Escenarios: la técnica de los escenarios consiste en describir las características del sistema a desarrollar mediante una secuencia de pasos. La representación del escenario puede ser casi textual o ir encaminada hacia una representación gráfica en forma de diagramas de flujo. El análisis de los escenarios, hechos de una forma u otra, pueden ofrecer información importante sobre las necesidades funcionales del sistema.

d) Casos de uso: Es la técnica básica del proceso RUP. Sin embargo, varios autores defienden que pueden resultar ambiguos a la hora de definir los requisitos, por lo que hay propuestas que los acompañan de descripciones basadas en plantillas o de diccionarios de datos que eliminen su ambigüedad.

En el presente trabajo se decidieron utilizar las técnicas: glosario de términos, plantillas y casos de uso durante la ERS ya que resultan las actividades más prácticas, frecuentes y a la vez sumamente demostrativas.

Una ERS perfecta es imposible. La calidad en ella es muy difícil de cuantificar. En general, una ERS de calidad no garantiza la ausencia de problemas, pero una ERS pésima garantiza su presencia. Cuanto más esfuerzo se ponga por parte de los ingenieros en lograr una buena ERS, menos serán los defectos del sistema que aparecerán en el futuro.

1.2.1.3 Validación de Requisitos (VRE)

En esta actividad se comprueba que las ERS se ajustan a las necesidades de clientes/usuarios y otros implicados, se valida por los usuarios que sus necesidades fueron adecuadamente interpretadas, se desarrollan actividades de verificación de requisitos con el fin de comprobar que las ERS se construyeron de acuerdo con los estándares establecidos, sin ambigüedad, sin inconsistencias, sin omisiones, que hayan sido corregidos los errores detectados y que el resultado del trabajo se ajuste a los estándares establecidos para el proceso, el proyecto y el producto (Pressman, 2005).

Para el buen desempeño de la validación de los requisitos es necesaria la presencia de los clientes, usuarios y demás implicados, esto resulta una condición determinante en el éxito de la actividad.

Técnicas de la Validación de Requisitos (TVRE)

Existen un conjunto de técnicas que se utilizan para llevar a cabo la VRE:

a) Revisiones: Está técnica consiste en la lectura y corrección de la completa documentación o modelado de la definición de requisitos. Con ello solamente se puede validar la correcta interpretación de la información transmitida.

b) Auditorías: Esta técnica consiste en la revisión de la documentación, controlando los resultados contra una lista de chequeo predefinida o definida a comienzos del proceso, es decir, sólo una muestra es revisada.

c) Matrices de trazabilidad: Esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma, se podrán detectar inconsistencias u objetivos no cubiertos.

d) Prototipos: Algunas propuestas sobre esta técnica señalan que se basa en obtener de la definición de requisitos prototipos, que a pesar de no tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene

como problema que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final (Escalona; Koch, 2002).

Con la VRE se pretende descubrir los problemas que el documento ERS pueda tener antes de comprometer recursos en su implementación. En esta investigación se seleccionaron las revisiones y los prototipos como las técnicas a desarrollar en la validación de los requisitos.

1.2.2 Metodologías de Desarrollo de Software

Las metodologías se desarrollan con el objetivo de dar solución a los problemas existentes en la producción de software, que cada vez son más complejos. Estas engloban procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de software. Un proceso de desarrollo de software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto.

Este conjunto de actividades, en el proceso de desarrollo de software, es el que proponen Jacobson, Rumbaugh y Booch, tiene 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, tengan la misma visión. Las piedras angulares del proceso de desarrollo del software son: el proyecto, las personas y el producto; siendo las características del cliente, el entorno de desarrollo y las condiciones del negocio, elementos que influyen en el proceso. Existe una estrecha relación entre personas, proyecto, producto y proceso.
Estos términos son conocidos como las cuatro “P” en el desarrollo de software:

Personas: Los principales autores de un proyecto software.
Proyecto: Elemento organizativo a través del cual se gestiona el desarrollo de software. El resultado de un proyecto es una versión de un producto.
Producto: Artefactos que se crean durante la vida del proyecto, como los modelos, código fuente, ejecutables y documentación.

Proceso: Un proceso de ingeniería de software es una definición del conjunto completo de actividades necesarias que guían los esfuerzos de los desarrolladores, para transformar los requisitos de usuario en un producto.

Si tomamos como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, podemos clasificar las metodologías en dos grupos: Metodologías Estructuradas y Metodologías Orientadas a Objetos. Por otra parte, considerando su filosofía de desarrollo aparece el apelativo Metodologías Tradicionales o Ágiles.

Dentro de las metodologías más utilizadas hoy día se pueden citar: RUP, XP y MSF, de ellas Rational Unified Process (RUP) se adapta mejor a los proyectos de largo plazo, dividiendo el desarrollo de software en cuatro fases desarrolladas iterativamente y abarcando seis disciplinas ingenieriles y tres de apoyo; Extreme Programing (XP) por su parte se recomienda para proyectos cortos y tiene por particularidad contar con el usuario final como parte del equipo; Microsoft Solution Framework (MSF) se adapta por su parte a cualquier tipo de proyecto y se centra en los modelos de procesos y de equipo (Mendoza Sánchez, 2004).
Para el desarrollo de este proyecto se ha seleccionado la metodología RUP debido al alcance del proyecto y a que la misma aplica muchas de las mejores prácticas del desarrollo de software moderno, enfocadas a la producción de software con calidad.

Rational Unified Process (RUP)

El Proceso Unificado de Desarrollo (RUP-Rational Unified Process) es un proceso pensado en dos dimensiones. El mismo se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto para los clientes. Cada ciclo consta de cuatro fases. Cada fase termina con un hito (Jacobson Ivar, et al., 2000).Como RUP es un proceso, en su modelación define como sus principales elementos:

Trabajadores (“quién”)	Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son

	propietarios de elementos.
Actividades (“cómo”)	Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
Artefactos (”qué”)	Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
Flujo de actividades (“Cuándo”)	Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

El ciclo de vida de RUP se caracteriza por:

- Dirigido por casos de uso: Los casos de uso (CU) reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos.
- Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura. El modelo de arquitectura se representa a través de vistas en las que se incluyen los diagramas de UML.
- Iterativo e Incremental: Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Cuenta con 4 fases, 9 flujos de trabajos, los primeros seis son conocidos como flujos de ingeniería y los tres últimos como de apoyo**. [Anexos, Figura2]**

La fase de Elaboración tiene como principal finalidad completar el análisis de los Casos de Uso y definir la arquitectura del sistema, en esta fase se encuentran los flujos de trabajo análisis y diseño los cuales tienen entre sus objetivos:

- Transformar los requerimientos en un diseño de cómo va a ser implementado el sistema.
- Evolucionar hacia una arquitectura de software robusta.
- Adaptar el diseño para que coincida con el ambiente de implementación, diseñando el sistema con un enfoque hacia el rendimiento.

1.2.3 Herramienta CASE

El acrónimo CASE en inglés significa Computer Aided Software Engineering y según su traducción en español es Ingeniería de Software Asistida por Computadoras.

Visual Paradigm 6.1

Visual Paradigm es una herramienta que ofrece un entorno de creación de diagramas usando las notaciones UML 2.0, con un diseño centrado en casos de uso y enfocado además al negocio, lo cual genera un software de alta calidad. Esta herramienta fue creada para el ciclo completo de desarrollo del software que lo automatiza y acelera, permitiendo la captura de requisitos, análisis, diseño e implementación, también proporciona características tales como generación del código, ingeniería inversa y generación de informes. Permite escribir toda la especificación de un caso de uso sin necesidad de utilizar una herramienta externa como editor de texto, utilizando plantillas que se encuentran definidas, o que pueden ser creadas por los usuarios. Está diseñada para usuarios interesados en sistemas de software de gran escala, apoya los estándares más recientes de la notación UML. A pesar de todo es importante destacar que la licencia de Visual Paradigm es muy restringida.

Debido a las funcionalidades brindadas por el Visual Paradigm al permitir crear diagramas usando las notaciones UML y BPMN y por ser una herramienta multiplataforma que se integra fácilmente con varios Entorno de desarrollo integrado (IDEs), se decidió por parte de la dirección del proyecto usarla como herramienta durante todo el proceso de modelado.

1.2.4 Lenguaje de modelado

Un lenguaje de modelado de objetos se puede definir como el conjunto estandarizado de símbolos y de modos de disponerlos para modelar (parte de) un diseño de software orientado a objetos.

Unified Modeling Language (UML)

UML es un lenguaje, que permite modelar analizar y diseñar sistemas orientados a objetos. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad (Jacobson Ivar, et al., 2000).Ofrece un estándar para describir un panorama del sistema modelo, incluyendo aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables y aspectos conceptuales como los procesos de negocios y funciones del sistema.

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software. El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos. Es importante recalcar que UML no es una guía para realizar el análisis y diseño orientado a objetos, es decir, no es un proceso. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos.
En este trabajo se usó la notación UML en la representación de los diagramas del módulo

Business Process Management Notation (BPMN)

Como parte de BPM se ha desarrollado una notación estándar de procesos de negocio conocida como Business Process Management Notation (BPMN), con el objetivo de proveer una notación entendible para cualquiera, desde el analista del proceso, el desarrollador técnico y hasta el cliente, así como para crear un puente estandarizado entre el diseño de procesos de negocio y su implementación. Dentro de los resultados de BPMN se encuentran la notación y semántica de un BPD (Business Process Diagram) mejorando las capacidades de las notaciones de procesos de negocios tradicionales.

El BPD (Business Process Diagram) es un diagrama diseñado para ser usado por las personas que diseñan y administran procesos de negocio. Dentro de las cuatro categorías básicas de elementos que se

pueden encontrar en un BPD se tienen: objetos de flujo, objetos de conexión, swimlanes y artefactos. Como objetos de flujo aparecen las actividades, que representan el trabajo ejecutado dentro de un proceso de negocio, pueden ser atómicas o no, existen tres tipos fundamentales de actividades: los procesos, subprocesos y tareas. Los procesos son actividades ejecutadas dentro de una compañía y son dibujados como un grafo de objetos de flujo, los cuales son un conjunto de otras actividades y sus transiciones, cada proceso puede tener sus propios subprocesos. Los subprocesos son actividades compuestas por un flujo de otras actividades.

Por su parte las tareas son actividades atómicas que aparecen incluidas dentro de procesos, que no necesitan descomponerse en más detalle y son ejecutadas por el usuario final o aplicación. Los eventos también son objetos de flujo e indican la ocurrencia de algo en el transcurso de un proceso de negocio, existen tres tipos de eventos: comienzo, intermedio y final. Las swimlanes son vistas en BPMN como un concepto para dividir y organizar las actividades, vienen dadas por pools y lanes. En la representación se utilizan los pools para componentes jerárquicamente abarcadores, áreas donde se desarrollan los procesos. Dentro de un pool se representa la interacción a través de lanes para indicar los roles que intervienen.

Los artefactos se usan para mostrar información adicional de los procesos. Hay tres tipos de artefactos: objetos de datos, anotación y grupo.

En el modelado de los procesos de negocio del módulo de la presente investigación se usó la notación BPMN conjuntamente con sus diagramas BPD por las grandes ventajas con que cuenta.

1.2.5 Framework

Un framework es un término muy utilizado últimamente en el campo de la informática, se utiliza para referirse a un conjunto de bibliotecas, que se utilizan para implementar la estructura de un modelo para una aplicación. Esto se realiza con el objetivo de promover la reutilización de código, posibilitando que no sea necesario perder tiempo en reinventar la rueda. Existen diferentes tipos de framework, para diferentes propósitos, algunos orientados al desarrollo de aplicaciones web, o para un determinado sistema operativo o lenguaje. En un sentido muy amplio el framework que se utiliza determina la arquitectura del software.

Spring Framework

Los frameworks dan la posibilidad de definir la forma de realización de aplicaciones de software, dando soporte y simplificando parte de la complejidad. Spring; es un framework que tiene el objetivo de facilitar la construcción de aplicaciones Java, presenta un entorno diseñado para aumentar la productividad, liberando al desarrollador de tareas repetitivas, ayudándolo a hacer diseños más consistentes. Se puede utilizar en cualquier tipo de aplicación, y es ligero por el mínimo impacto que tiene en las aplicaciones. Spring se basa en la técnica Inversión de Control (IC), técnica que promueve el bajo acoplamiento a partir de la inyección de dependencias (DI) entre los objetos y una implementación de desarrollo según el paradigma de Orientación a Aspectos (AOP) que presenta una estructura simplificada para el desarrollo y utilización de aspectos (módulos multiple object crosscutting).

El paradigma de Orientación a Aspectos (AOP) complementa la Programación Orientada a Objetos (POO), proponiendo otra manera de pensar sobre la estructura de un programa. Mientras que la POO descompone las aplicaciones en una jerarquía de objetos, la AOP descompone los programas en aspectos o preocupaciones. Dichas preocupaciones se convierten en servicios del sistema, separados de la lógica de negocio y que se ejecutan de manera transversal a la funcionalidad base, lo que proporciona una definición de responsabilidades superior. Spring básicamente es un contenedor, que se encarga de gestionar, administrar el ciclo de vida y de la configuración de las clases de la aplicación. Es una aplicación open source, lo cual implica que no tiene ningún costo, ni se necesita licencia para utilizarlo, dando la libertad de incursionar en la utilización de esta aplicación, además de que está disponible todo el código fuente de este framework en el paquete de la instalación. En sentido general Spring no intenta reinventar la rueda, sino que integra las diferentes tecnologías existentes en un solo framework posibilitando un desarrollo más sencillo y eficaz.

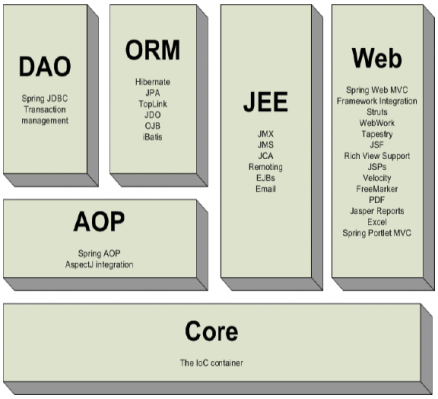


Figura3: Arquitectura del framework Spring.

- Core: Como su nombre indica, es el núcleo de Spring. Permite técnicas de Inversión del Control (IoC) como la inyección de dependencias.
- AOP: Proporciona una implementación de programación orientada a aspectos, permitiendo definir puntos de corte e interceptores.
- DAO: Proporciona el acceso a una capa de abstracción JDBC (Java Database Connectivity) con una forma de administrar transacciones desde el módulo AOP como es el caso de Hibernate y JDO.
- ORM: Provee capas de integración para APIs de mapeo objeto-relacional.
- Web: posibilita determinadas características apropiadas para el desarrollo de aplicaciones web e integración con otros frameworks (Struts, JSF, Tapestry, etc.).
- JEE: acceso e interacción con servicios Enterprise.

Spring web flow

Spring web flow forma parte del paquete de desarrollo de aplicaciones web de Spring. Permite la navegación múltiple y compleja permitiendo guiar al usuario a través de una serie de pasos para completar una transacción de aplicación y por ser una plataforma web de alto nivel, además incrementa la productividad, calidad y facilidad para realizar pruebas en el proceso de desarrollo.

Hibernate

Hibernate es una solución ORM (*Object-Relational Mapping*) para Java, es open source (código abierto) y la licencia está eximida de costo. *Hibernate* busca solucionar el problema de la diferencia entre el modelo orientado a objetos y el usado en las bases de datos de modelo relacional mediante archivos declarativos. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de un modelo objetual existente. Cabe mencionar que provee con Hibernate Query Language (HQL) una poderosa vía de comunicación entre el programador y la base de datos, al dotarlo de un lenguaje invariante con respecto a esta y además sintácticamente muy parecido al Structured Query Language (SQL).

1.2.6 Lenguaje de programación

Java

Java es un lenguaje de desarrollo orientado a objetos, multiplataforma. Proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas. Es un lenguaje interpretado y compilado, esto quiere decir que su código fuente se transforma en algo similar al código máquina, los bytecodes y a la vez estos bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time). Además, es considerado un lenguaje robusto, fiable y seguro, para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución.

Soporta la sincronización múltiple de hilos de ejecución (*multithreading*), lo cual da la posibilidad que un hilo puede ocuparse de interactuar con el cliente y otro de realizar una operación. Otra de las tantas características que lo hacen idóneo para su utilización, es que se utiliza para dos tipos de programas, aplicaciones independientes y applets.

1.2.7 Entorno de desarrollo integrado (IDE)

Eclipse

Eclipse es un entorno de desarrollo integrado (IDE) de código abierto independiente de una plataforma, es una aplicación de cliente enriquecido, emplea *plug-ins* para proporcionar toda su funcionalidad, a

diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Eclipse es, en el fondo, únicamente un armazón sobre la que se pueden montar herramientas de desarrollo. La arquitectura de *plug-ins* permite, además de integrar diversos lenguajes, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo, tales como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, entre otros.

1.3 Sistema Bancario

Se dice que toda actividad industrial, comercial y personal está controlada y determinada por la Banca. Desde el momento en que un negocio se proyecta y una fase ulterior se desarrolla, se ve en la necesidad de contar con los servicios del banco, incluso si ese negocio produjese beneficios, ellos en gran medida tendrían como destino la inversión, también relacionada con la Banca. Por tanto, y en conclusión: la Banca determina y controla la vida económica del mundo. Los bancos por su parte constituyen los entes indiscutibles del sistema bancario y son instituciones que realizan operaciones como prestatarias y prestamistas de crédito, reciben y concentran en forma de depósitos los capitales captados para ponerlos a disposición de quienes puedan hacerlos fructificar.

En la actualidad el sistema bancario enfrenta retos diariamente, entre los cuales está el que implica relacionarse con millones de clientes, basándose en la planificación de captura de datos individuales y el cálculo de un importante conjunto de variables y perfiles de propensión que conforman una base de conocimientos, destinada a personalizar tanto la oferta como el trato otorgados a cada cliente. Otro de los retos es que para todo banco, independientemente de su tamaño, es imprescindible superar la presión a la que se enfrentan a diario para lograr disparar sus beneficios y ser más efectivo. Además, es muy competitiva y compleja la red que se ha creado entre los Bancos, Cajas y demás Servicios Financieros que existen, incluso las nuevas generaciones y futuros clientes, totalmente integrados a las nuevas tecnologías, también fuerzan a las entidades financieras a innovar, integrar tecnología y ser cada día más competitivas y eficientes. En resumen, la situación que enfrenta el sistema bancario exige contar con un buen sistema informático capaz de gestionar toda la información necesaria y ayudar en el desarrollo de las operaciones**.**

Para desarrollar un sistema capaz de informatizar las actividades contables y el resto de las operaciones de un banco, es imprescindible conocer a fondo su funcionamiento. Desde un inicio el banco tiene la función de captar pasivos a clientes, los cuales de una manera bien planificada se utilizan en la

adquisición de intereses para la entidad. De la misma manera se realizan un grupo de operaciones tales como préstamos, transferencias, una serie de pagos y otras operaciones. Además de las muchas negociaciones, ya sea entre clientes cercanos o a disímiles distancias que también son apoyadas por los bancos.

En cuanto a la estructura física de un banco, se puede plantear que cada uno se estructura de la manera más conveniente, contando con una serie de departamentos, en los cuales lleva a cabo las operaciones de una manera coordinada.

1.3.1 Procesos Bancarios

Los procesos bancarios se pueden comprenden a partir del estudio de los objetivos generales y específicos que tiene un banco como entidad financiera. La atención personalizada a los clientes y la búsqueda incesante de ingresos, resultan algunos de ellos, además de promover su actividad como instrumento, para el desarrollo del Comercio Exterior y sostener las relaciones con otros bancos.

Las operaciones bancarias son todas aquellas operaciones de crédito practicadas por un banco de manera profesional. Las mismas se clasifican en activas, pasivas y neutras. Las operaciones activas son aquellas en las que el banco otorga el crédito, como ejemplos de ellas se tienen los préstamos y los descuentos. Las operaciones pasivas son aquellas en las que el banco recibe dinero de clientes y paga intereses por esas prestaciones, como ejemplos de operaciones pasivas se tienen las Cuentas Corrientes, las de Ahorro, a Plazo Fijo y Cédulas Hipotecarias. Por su parte las operaciones neutras o accesorias son aquellas donde el banco no recibe ni otorga crédito, como las operaciones de mediación, donde el banco solo sirve de intermediario (Toca Ramos, 2006). Todas esas operaciones en su conjunto hacen posible que un banco mantenga su estabilidad como entidad financiera.

1.3.1.1 Análisis de Riesgo Financiero

La misión fundamental de una entidad bancaria consiste en canalizar recursos financieros de unidades económicas superavitarias (los depositantes y otros acreedores) a otras deficitarias (prestatarios, inversionistas y demás tomadores de fondos), con el objetivo de financiar actividades productivas y rentables que sean capaces de generar empleo y crecimiento económico. Para ello los intermediarios de

crédito normalmente utilizan una serie de recursos humanos, operativos y financieros que asumen un conjunto de riesgos, algunos de ellos de compleja identificación y de difícil medición. (Buniak)

No obstante, gestionar eficazmente estos riesgos para garantizar resultados operacionales cónsonos con los grandes objetivos estratégicos de la organización, quizás sea uno de los mayores retos de administradores y gestores bancarios en los tiempos que corren. (Buniak)

Desde este punto de vista, la gestión integral de los riesgos se vuelve parte fundamental de la estrategia y factor clave de éxito en la creación de valor económico agregado para los Stakeholders (accionistas, empleados, depositantes, inversionistas, supervisores y reguladores, etc.)

Existen metodologías para la administración de riesgos en sus distintas fases, siendo estos:

1. Riesgo de Crédito.
2. Riesgo de Mercado.
3. Riesgo de Liquidez.
4. Riesgo Operacional.

El presente trabajo de diploma se centrará en el Riesgo de Crédito que se define como la volatilidad de los ingresos debido a pérdidas potenciales en crédito por falta de pago de un acreditado o contraparte, en otras palabras es el riesgo de que los clientes no cumplan con sus obligaciones de pago.

Riesgo de Crédito

Es la posibilidad de que una entidad incurra en pérdidas y se disminuya el valor de sus activos, como consecuencia de que sus deudores o contraparte fallen en el cumplimiento oportuno o cumplan imperfectamente los términos acordados en los contratos de crédito. (FOGACOOP, 2005)

Consiste en la probabilidad que ocurra un siniestro financiero por incapacidad de pago de los deudores del banco, tanto en forma individual como en forma consolidada. (Buniak)

El activo más importante y con mayor participación en una cooperativa que desarrolla actividad financiera es la cartera de créditos. Es el área principal de exposición de las cooperativas con sus asociados. El riesgo de crédito es la principal fuente de problemas en los entes financieros. (FOGACOOP, 2005)

Los beneficios del Riesgo de Crédito son:

- Mejorar la rentabilidad de la institución (económica y social)
- Mejorar la calidad crediticia (Minimizar Cartera Vencida)
- Eficiencia operativa
- Generar reservas adecuadas y anticipar requerimientos de capital
- Desarrollar plataforma para un sano crecimiento de la cartera crediticia

Entre sus objetivos está calcular y determinar:

- Cálculo de Probabilidad Default para clientes actuales y para nuevos clientes.
- Fijación de límites máximos de tolerancia según probabilidad default.
- Pérdida Esperada (LGD). Se obtiene el máximo valor que una institución podría perder por incumplimiento en el pago de la cartera crediticia en la eventualidad de default.
- Cálculo de la Pérdida Esperada *(Loss given default)*
- -Definición Período de Espera
- -Definición Nivel de Confianza
- -Cálculo de Pérdidas Esperadas
- Cálculo de Pérdidas no Esperadas.
- Cálculo de Capital en Riesgo.
- Monto de la Exposición (EAD).
- Cálculo Exposición Neta.
- Exposición Futura.
- Evaluación de Provisiones.
- Tasa de recuperación de la Probabilidad de Default (PD)

1.3.2 Sistema Bancario Cubano

El sistema bancario cubano ha sufrido con el transcurso de los años una gran modernización: durante 1995 y 1996 el esfuerzo principal se concentró en la automatización de las 500 sucursales y demás oficinas bancarias, instalando en la totalidad de ellas redes locales con computadoras personales en prácticamente todos los puestos de trabajo. Durante 1997 y 1998 el centro de atención estuvo en la interconexión de las sucursales a la Red Pública de Transmisión de Datos. A finales de 1998 los bancos cubanos que realizan transacciones internacionales quedaron conectados a SWIFT. Actualmente en Cuba se cuenta con nueve bancos: Banco Central de Cuba (BCC), Banco Nacional de Cuba (BNC), Banco de Crédito y Comercio (BANDEC), Banco Popular de Ahorro (BPA), Banco Financiero Internacional, S.A. (BFI), Banco Internacional de Comercio S.A. (BICSA), Banco Metropolitano S.A., (BM), Banco de Inversiones, S.A. y Banco Exterior de Cuba, además existen oficinas de representación de bancos extranjeros en Cuba, que operan bajo licencia de BCC.

Entre los bancos cubanos podemos citar el Banco Nacional de Cuba (BNC), el cual una vez liberado de las funciones de banca central y de rector del sistema bancario, continúa existiendo con el carácter de banco comercial, autorizado a ejercer funciones inherentes a la banca universal, teniendo además la función de registro, control, servicio y atención de la deuda externa que el Estado y el propio banco han contraído con acreedores extranjeros con la garantía del Estado.

El Banco Nacional de Cuba como entidad bancaria señala la necesidad de un sistema automatizado para llevar a cabo el Análisis de Riesgo de Crédito a clientes; que no es más que un sistema que tomará por entradas los estados financieros de una empresa o cliente determinado, ya sea natural o jurídico, a través de fórmulas dadas por los ratios financieros, que nos dirá si la transacción resulta factible o no.

1.4 Sistema para el Análisis de Riesgo de Crédito

CreditRisk plus es un modelo de riesgo de crédito estadístico lanzado por Credit Suisse First Boston (CSFB) en 1997, se puede aplicar a cualquier tipo de producto de crédito, incluidos préstamos, bonos, cartas de crédito y derivados financieros y permite sólo dos resultados, factible y no factible. La probabilidad de incumplimiento de crédito depende de la clasificación, factores de riesgo y la sensibilidad de la parte obligada a los factores de riesgo.

CreditRisk plus utiliza técnicas de análisis, en contraposición a las simulaciones, para estimar el riesgo de crédito. Las técnicas utilizadas son similares a las aplicadas en la industria de seguros, se modela el riesgo de crédito basado en el tratamiento de las tasas de morosidad como variables aleatorias continuas.
Cuando se estima el riesgo de crédito considera:

- Calidad crediticia y el riesgo sistemático de los deudores
- Tamaño y la madurez de cada exposición
- Concentraciones de las exposiciones dentro de una operación

Representa la correlación entre eventos predeterminados diferentes mediante el análisis de volatilidades por incumplimiento en los distintos sectores, como las industrias o países diferentes. Este método funciona porque los incumplimientos suelen estar relacionados con factores del mismo medio, tales como una recesión económica.
Para estimar el riesgo de crédito debido a los fenómenos extremos tales como terremotos, CreditRisk plus utiliza la prueba de stress y para eventos de baja probabilidad que no pueden ser cubiertos por el modelo estadístico, utiliza un enfoque basado en hipótesis.
También cuenta con aplicaciones para:

- Cálculo disposiciones de riesgo de crédito
- Forzar el límite de crédito
- Gestión de operaciones de crédito

Se considera CreditRisk plus como una de las herramientas que más aportó a la investigación, ya que cuenta con una estructura bien definida, sus técnicas y metodologías son bien aplicadas, muestran resultados factibles y objetivos al tipo de análisis que se va a realizar. Todos sus procesos están bien documentados y constituyen en gran medida una guía para proyectos de esta índole que se quieran llevar a cabo, tales como los que se desarrollan en el módulo bancario en nuestra universidad.

1.5 Conclusiones Parciales

En el presente capítulo se llevó a cabo un estudio de la Ingeniería de Requerimientos, una de las principales disciplinas dentro de la Ingeniería de Software. Además, se pueden consultar las metodologías de desarrollo de software, herramientas CASE, lenguajes de modelado, así como frameworks a utilizar.

El Banco Nacional de Cuba como entidad bancaria señala la necesidad de un sistema automatizado para llevar a cabo el Análisis de Riesgo de Crédito a clientes, para ello se deriva el estudio en este capítulo de los procesos bancarios, y de manera mas específica el sistema bancario cubano y su funcionamiento.

CAPÍTULO 2: ELICITACIÓN, ESPECIFICACIÓN Y VALIDACIÓN DE REQUISITOS.

2.1 Introducción

En el presente capítulo se realiza la Ingeniería de Requerimientos, fundamentalmente las tres actividades propuestas por Pressman: Elicitación, Especificación y Validación de los Requisitos, y su vinculación con una serie de procesos bancarios. Partiendo de los procesos que tienen lugar en el área de Riesgo del Banco Nacional de Cuba, se obtiene la modelación del negocio, que contiene la especificación detallada de esos procesos, una serie de posibles mejoras y reglas a tener en cuenta en próximas etapas.

2.2 Elicitación de requisitos (ER)

La elicitación de requisitos es considerada el primer paso en la Ingeniería de Requisitos, en ella se identifican los requerimientos que deberá satisfacer el sistema, como resultado de la interacción que se tiene con los clientes y expertos del negocio.

Descripción de las técnicas empleadas en la Elicitación de Requisitos (TER)

En la identificación de los requerimientos del módulo Análisis de Riesgo de Crédito se utilizaron las técnicas revisión de documentos, entrevistas y la modelación del negocio. Se estudiaron los manuales de procedimientos y se realizaron entrevistas a los operadores de esta área, que sirvieron para delimitar el alcance del módulo y los procesos que se llevan a cabo en el mismo, estos últimos se pudieron apreciar en su mayoría ejecutándose en la práctica, ya que se participó en conjunto con los operadores del banco.

2.2.1 Modelación del Negocio del módulo Análisis de Riesgo de Crédito

Una de las TER es la modelación del negocio, donde se trata de que sea el propio negocio el que fuerce los requerimientos del sistema. Con esta técnica se comprenden las actividades y procesos que ocurren en la entidad y se proponen posibles mejoras.

En el estudio de los procesos del módulo Análisis de Riesgo de Crédito se pudo percibir que los procesos que se desarrollan en esta área están bien delimitados unos de otros y que los roles que participan se pueden encontrar fácilmente.

En el Banco Nacional de Cuba la Gerencia de Riesgo es la responsable de gestionar las solicitudes de análisis de riesgos con su correspondiente expediente, el cual es presentado por la Dirección de Gestión de Negocios ambos con la aprobación y firma del director de dicho departamento. También en esta área

se analiza y dictamina la situación económico-financiera del cliente y su garante si lo hubiese. Como resultado de esta actividad se aprueba o no el crédito, para ello el analista deberá poseer con anterioridad los documentos necesarios y así comenzar los procesos de análisis de riesgo para el cliente en cuestión.

2.2.1.1 Involucrados

En el estudio realizado a los procesos del área Gerencia de Riesgo se identificó un involucrado que de alguna manera se interrelaciona con esos procesos.

Involucrado	Descripción
Dirección de Gestión de Negocios	Deberá presentar a la gerencia de riesgos la documentación necesaria, así como un resumen de la operación objeto de análisis.

2.2.1.2 Trabajadores

En los procesos de Gerencia de Riesgo se pudieron detectar un conjunto de roles que participan en la ejecución de las actividades dentro de las áreas del módulo.

Trabajadores	Descripción
Comité de Crédito del BNC	Analiza el dictamen final y da su aprobación o no a las operaciones solicitadas.
Analista de Riesgo Financiero	Analiza e investiga la información de la solicitud de análisis de riesgo. Emite una valoración final referente a las operaciones en cuestión.

2.2.1.3 Artefactos

En la ejecución de los procesos del área Gerencia de Riesgo existen una serie de objetos que son tomados, manipulados o producidos por los trabajadores en las diferentes actividades que realizan, generalmente son documentos físicos, aunque algunos son electrónicos, que se utilizan y luego son archivados en las áreas, o se entregan a los interesados. A continuación se describen esos artefactos.

Artefactos	Descripción
Informe Financiero	Balance General: Documento en forma de tablas que contiene la información de los activos y los pasivos circulantes de la empresa en un periodo de un año.

	Estado Resultado: Documento en forma de tablas que contiene la información de las utilidades en el periodo de un año.
Análisis económico-financiero	Tablas elaboradas en sistema Excel por el Analista para llevar el análisis porcentual en las variaciones así como el cálculo de las razones financieras.
Informe de Flujo de Caja	Documento consultado por el analista para realizar un estudio profundo de los flujos y las estimaciones proyectadas.
Expediente de Solicitud de Crédito	Conjunto de documentos que resume toda la operación objeto de análisis
Dictamen Final	Documento resumen de todo el análisis de la solicitud de riesgo que expresa la valoración final del analista.

2.2.1.4 Resumen de los procesos del negocio

En el área de Riesgo se hizo un estudio de aquellas operaciones que generan algún valor o ayudan a mitigar riesgos del BNC, como resultado se delimitaron por una parte, procesos que representan actividades comerciales importantes, por otro lado, se definieron procesos que aunque no tienen valor comercial, son necesarios para la entidad.

Proceso	Registrar información financiera
Involucrados	Dirección de Gestión de Negocios(DGN)
Trabajadores	Analista de Riesgo
Resumen	El proceso se inicia cuando el analista de riesgo recibe de la DGN un expediente referente a la solicitud de crédito del cliente y además la información financiera que se agrupa correspondientemente y se registra en tablas Excel.

Proceso	Determinar la factibilidad de la operación
Involucrados	_
Trabajadores	Analista de Riesgo
Resumen	El proceso inicia cuando el Analista de Riesgo evalúa el resultado de los indicadores y determina la situación financiera de la empresa. Partiendo de la misma y de su experiencia personal dictamina la factibilidad de la operación a

	realizar.

Proceso	Validar resultado de la operación
Involucrados	_
Trabajadores	Analista de Riesgo, Comité de Crédito de BNC
Resumen	El proceso inicia cuando el Analista de Riesgo elabora el dictamen final con el resumen del análisis realizado y lo presenta al Comité de Crédito del BNC, este lo analiza y da su aprobación o no a las operaciones a realizar.

Para una mejor comprensión de los procesos del negocio estudiados en este módulo, se realizó usando la notación BPMN y sus diagramas BPD en la herramienta Visual Paradigm un diagrama para cada uno de los procesos encontrados (**Anexos, Figura 4**, **Figura 5, Figura 6**).

2.2.1.5 Reglas del Negocio

Una regla del negocio es una condición que debe ser satisfecha dentro de la organización y que no puede ser violada por ninguno de los trabajadores que realizan las diferentes actividades. En el presente módulo se identificaron un conjunto de reglas, de las cuales se seleccionaron aquellas de mayor relevancia que deberán ser tenidas en cuenta durante la etapa de implementación.

Procesos	**Reglas por procesos**
Registrar información financiera	➢ La solicitud de análisis deberá estar firmada y avalado por el Director de la DGN correspondiente.
Calcular indicadores contables	➢ La información financiera del cliente deberá estar previamente almacenada y agrupada correctamente en el sistema.
Determinar la factibilidad de la operación	➢ Todos los indicadores deberán tener una prioridad asignada.
Validar resultado de la operación	➢ El dictamen final deberá estar firmado por el analista que lo realizó.

2.2.1.6 Mejoras de los procesos a automatizar

Uno de los objetivos de la técnica modelo del negocio dentro de la Elicitación de Requisitos es la identificación de problemas y mejoras sobre los procesos que se desarrollan en la entidad. En el área Gerencia de Riesgo del BNC se pudo percibir que gran parte del trabajo se realiza de forma manual y que los sistemas que funcionan actualmente, además de correr sobre plataformas diferentes no son interoperables entre sí. A continuación se muestran un grupo de problemas y mejoras por procesos.

Proceso	**Problemas y Mejoras**
Mejoras Generales	➢ El nuevo sistema integrará las operaciones que se realizan actualmente en tablas Excel por separado en un único sistema automatizado.
	➢ La determinación de la factibilidad de las operaciones de asignaciones de créditos se realizarán con mayor brevedad y menos esfuerzos por parte del analista de riesgo.

2.3 Requisitos Funcionales del módulo Análisis de Riesgo de Crédito

En la definición de los requisitos funcionales de un sistema se tienen en cuenta las funcionalidades o servicios que la aplicación debe brindar. Asimismo en esta actividad los desarrolladores responden a preguntas tales como ¿Qué le permitirá hacer el sistema de software al usuario?

A continuación se muestran los requisitos funcionales para el módulo Análisis de Riesgo de Crédito:

RF1: Registrar información financiera del cliente.

1.1 Registrar manualmente

1.2 Cargar automáticamente.

RF2: Calcular indicadores contables.

RF3: Actualizar prioridad de los indicadores contables.

RF4: Listar indicadores.

RF5: Obtener resultado de factibilidad.

RF6: Crear dictamen final.

RF7: Imprimir dictamen final.

RF8: Guardar expediente de la operación.

RF9: Consultar expediente de operación.

RF10: Listar operaciones realizadas.

RF11: Organizar operaciones.

2.4 Diagrama de Casos de Uso del Sistema

Los diagramas de casos de uso del sistema constituyen una representación de los requerimientos funcionales, los roles que interactúan con el sistema y las interfaces hacia sistemas externos. En el módulo Análisis de Riesgo de Crédito se definieron un total de 9 casos de uso y 1 actor (**Figura 7**).

Un actor del sistema es la persona que inicializa un caso de uso del sistema, es decir, que interactúa con el mismo, es por esto que se define como actor del sistema al Analista de Riesgo.

El analista de Riesgo es el encargado de introducir, organizar y almacenar toda la información que se obtiene y se genera de todo el proceso de otorgamiento de crédito.

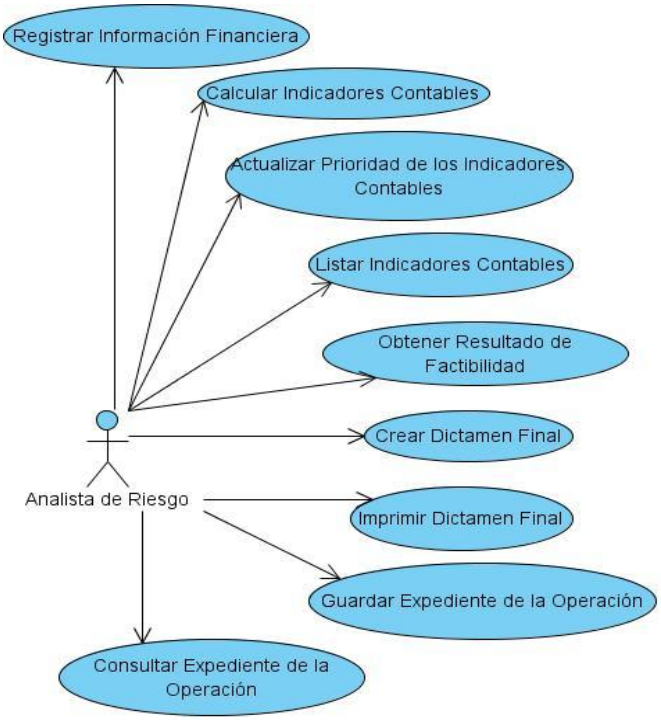


Figura 7: Diagrama de casos de uso.

2.5 Descripción de los casos de uso del módulo Análisis de Riesgo de Crédito

Para entender la funcionalidad asociada a cada caso de uso no es suficiente con la representación gráfica del Diagrama de Casos de Uso, la misma debe ir acompañada de una descripción textual, que puede ser elaborada de forma breve o extendida y debe ir acompañada del prototipo respectivo.

El prototipo del sistema que se construye en este punto, da una visión de las interfaces diseñadas para cada caso de uso. Se muestran con comportamiento estático ante el usuario para validar que los requerimientos funcionales estén correctos.

A continuación se muestran las descripciones detalladas de algunos de los casos de uso críticos del módulo Análisis de Riesgo de Crédito.

Caso de Uso:	**Registrar Información Financiera**
Actor:	Analista de Riesgo
Resumen:	El caso de uso se inicia cuando el Analista de Riesgo solicita Registrar Información Financiera. Esta es cargada a la aplicación, ya sea manual o automáticamente. El caso de uso culmina cuando el Analista de Riesgo finaliza la operación solicitada.
Precondiciones:	• El Analista de Riesgo debe estar previamente autenticado en el sistema.
Poscondiciones	• La Información financiera debe quedar correctamente registrada.
Referencia:	RF 1
Casos de uso Relacionados:	--
Prioridad:	Crítico
Flujo Normal de los Eventos	
Actor	**Respuesta del Sistema**
1. El Analista de Riesgo solicita la opción	2. El sistema muestra las opciones Registrar

Registrar Información Financiera	Manualmente, Registrar Automáticamente.
3. El Analista de Riesgo selecciona una de las opciones y solicita aceptar. a) Registrar Manualmente, ver Escenario Registrar Manualmente. b) Registrar Automáticamente, ver Escenario Registrar Automáticamente. En caso de solicitar Cancelar, ver Flujos Alternos Sección Cancelar.	

Figura 8: "Prototipo Registrar Información Financiera"

Escenario Registrar Manualmente	
	4. El sistema muestra las tablas para registrar los datos financieros del cliente
5. El Analista de Riesgo introduce la información financiera.	
6. El Analista de Riesgo solicita Aceptar. En caso de seleccionar Cancelar: ver Flujos Alternos Sección Cancelar.	7. El sistema valida la información registrada. En caso de: a) Error en los datos de entrada: ver Flujos Alternos Sección Información Incorrecta.

	8. El sistema guarda la información introducida.

Figura 9: "Prototipo Registrar Manualmente"

Escenario Cargar Automáticamente	
9. El Analista de Riesgo selecciona la ruta desde la que va a cargar la información al sistema.	10. El sistema carga la información y notifica. En caso de error en URL; ver Flujos Alternos ""Error en la Ruta de Acceso"
11. El Analista de Riesgo solicita Aceptar. En caso de seleccionar la opción Cancelar; ver Flujos Alternos Sección "Cancelar".	12. El sistema guarda la información cargada

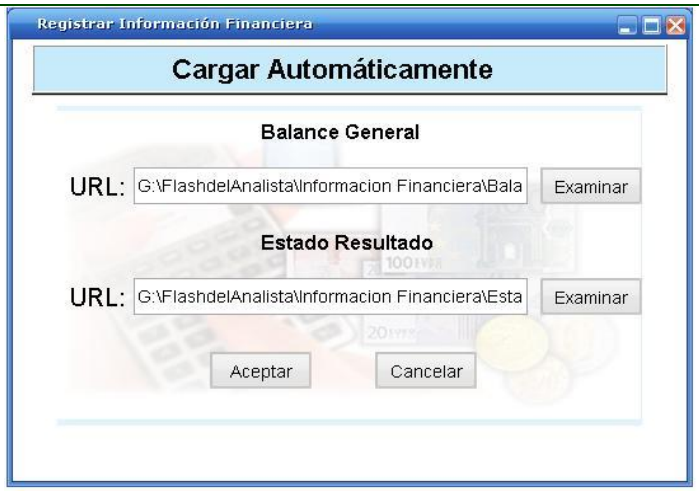


Figura 10: "Prototipo Cargar Automáticamente"

Flujos Alternos	
Sección Cancelar	
Acción del Actor	**Respuesta del Sistema**
	1.1. Finaliza el caso de uso
Sección Información Incorrecta	
Acción del Actor	**Respuesta del Sistema**
	1.1. El sistema señala los datos incorrectos.
1.2. El Analista de Riesgo corrige los datos incorrectos.	
1.3. El Analista de Riesgo solicita aceptar.	1.4. Continúa el flujo normal de eventos.
Sección Error en la ruta de acceso	
Acción del Actor	**Respuesta del Sistema**
	1.1. El sistema da la opción de volver a cargar la información.
1.2. En caso de: a) Aceptar: Vuelve al paso 9 del escenario Cargar Automáticamente. b) Cancelar: ver Flujos Alternos Sección Cancelar.	

Caso de Uso:	**Calcular Indicadores**
Actor:	Analista de Riesgo
Resumen:	El caso de uso se inicia cuando el Analista de Riesgo solicita Calcular Indicadores. El sistema calcula dichos indicadores contables a partir de la información financiera almacenada y brinda la opción listar indicadores. El caso de uso culmina cuando el Analista de Riesgo finaliza la operación solicitada.
Precondiciones:	• La Información financiera deberá estar previamente registrada y agrupada.
Poscondiciones	• Los Indicadores quedarán calculados, actualizados y registrados en el sistema.
Referencia:	RF 2, RF 4
Casos de uso Relacionados:	Registrar información financiera
Prioridad:	Crítico

Flujo Normal de los Eventos	
Actor	**Respuesta del Sistema**
1. El Analista de Riesgo solicita la opción Calcular Indicadores.	2. El sistema calcula los Indicadores Contables y muestra los resultados para los Indicadores de prioridad 1.
3. El Analista de Riesgo selecciona la opción Aceptar. En caso de: a) Seleccione la opción Cancelar, ver flujos Alternos Sección Cancelar.	4. El sistema guarda la información obtenida y muestra también las opciones Listar Indicadores (ver Caso de Uso Listar Indicadores Contables).

Figura 11: "Prototipo Calcular Indicadores Contables"

Flujos Alternos	
Sección Cancelar	
Acción del Actor	**Respuesta del Sistema**
	1.1. Finaliza el Caso de uso.

Caso de Uso:	**Obtener Resultado de Factibilidad**
Actor:	Analista de Riesgo
Resumen:	El caso de uso se inicia cuando el Analista de Riesgo solicita al sistema obtener el resultado de la factibilidad de la operación, el sistema solicita introducir el % de decisión y muestra el resultado de factibilidad. El caso de uso culmina cuando el Analista de Riesgo finaliza la operación solicitada.
Precondiciones:	• Los Indicadores Contables deberán calcularse con anterioridad.
Poscondiciones	• El resultado de factibilidad quedará registrado en el sistema.
Referencia:	RF 5
Casos de uso Relacionados:	Calcular Indicadores

Prioridad:	Crítico
Flujo Normal de los Eventos	
Actor	**Respuesta del Sistema**
1. El Analista de Riesgo solicita la opción Obtener Resultado de Factibilidad.	2. El sistema muestra un campo para introducir el % de decisión.
3. El Analista de Riesgo introduce el % de decisión En caso de: a) Solicita Cancelar, ver Flujos Alternos "Sección Cancelar"	4. El sistema muestra el resultado de factibilidad. En caso de: a) Error en el % de decisión, ver Flujos Alternos "Sección Error en el % de decisión"
	5. El sistema registra el resultado de factibilidad obtenido

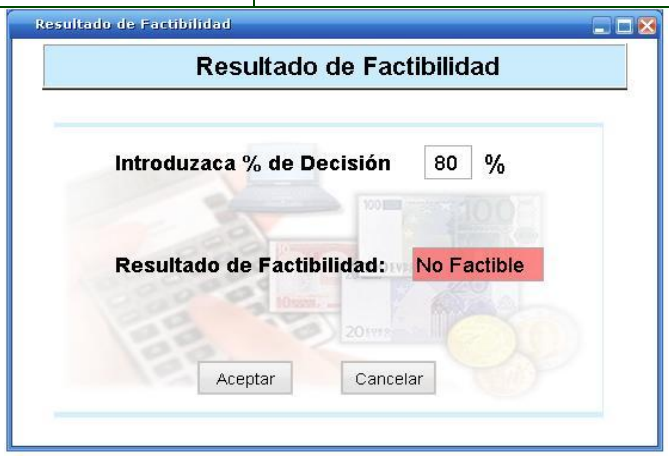


Figura 12: "Prototipo Obtener Resultado de Factibilidad"

Flujos Alternos	
Sección Cancelar	
Acción del Actor	**Respuesta del Sistema**
	1.1. Finaliza el Caso de Uso
Sección Error en el % de decisión	
Acción del Actor	**Respuesta del Sistema**
	1.1. El sistema notifica que existe un error en el %

	de decisión
1.2. El Analista de Riesgo corrige el error señalado.	
1.3. El Analista de Riesgo solicita aceptar	1.4. Continúa el flujo normal de eventos.

Caso de Uso:	**Crear Dictamen Final**
Actor:	Analista de Riesgo
Resumen:	El caso de uso se inicia cuando el Analista de Riesgo selecciona la opción crear Dictamen Final, el sistema permite elaborarlo y almacenarlo. Si el usuario solicita imprimir, el sistema manda a imprimir el dictamen El caso de uso culmina cuando el Analista de Riesgo finaliza la operación solicitada.
Precondiciones:	• Los Indicadores Contables y el resultado de factibilidad deberán estar previamente calculados.
Poscondiciones	• El Dictamen Final quedará almacenado en el sistema.
Referencia:	RF 6, RF7
Casos de uso Relacionados:	Imprimir Dictamen Final, Listar Indicadores
Prioridad:	Crítico

Flujo Normal de los Eventos	
Actor	**Respuesta del Sistema**
1. El Analista de Riesgo solicita la opción Crear Dictamen Final.	2. El sistema muestra un editor de texto donde será elaborado el Dictamen Final. El sistema permite Listar los Indicadores previamente calculados
3. El Analista de Riesgo elabora el Dictamen Final y solicita la opción Aceptar. En caso de: a) Solicite la opción Cancelar; ver Flujos Alternos “Sección Cancelar”	4. El sistema guarda el documento y muestra la opción Imprimir Dictamen Final (ver Caso de Uso Imprimir Dictamen Final)

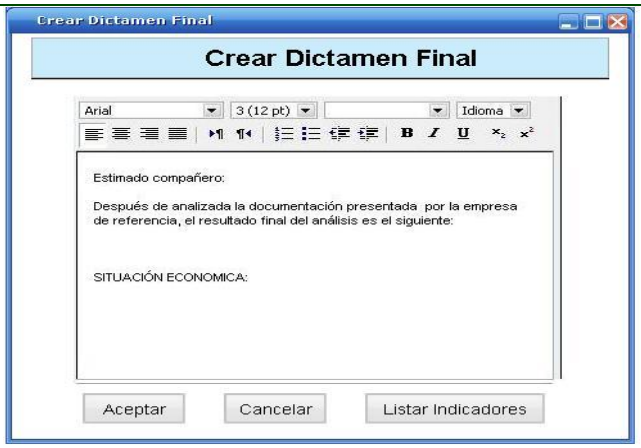


Figura 13 "Prototipo Crear Dictamen Final"

Flujos Alternos	
Sección Cancelar	
Acción del Actor	**Respuesta del Sistema**
	1.1. Finaliza el Caso de uso.

2.6 Validación de Requisitos de Software

Con el propósito de conocer errores y prevenirlos a tiempo antes de comprometer recursos en etapas futuras del desarrollo de software, se desarrolló como una tercera actividad dentro de la Ingeniería de Requisitos, la Validación de los Requisitos del módulo Análisis de Riesgo de Crédito, en la cual se revisaron por parte del equipo del proyecto todos los requisitos especificados y se validaron por los clientes del BNC, obteniéndose gran aceptación.

2.7 Indicadores Contables

Como resultado de la investigación se definieron un conjunto de indicadores que definen en su totalidad la factibilidad de las operaciones. Los mismos se muestran a continuación:

No.	**Indicadores**	**Prioridad**
Balance General		
1	Capital de trabajo=A.Circ.-P.Circ	1
2	Ventas	1
3	Costo de ventas	1
4	Costo por peso de ventas	1

5	Costo Total=Costo de Ventas+Total de Gastos	1
6	Apalancamiento Financiero	1
Estado Resultado		
I.RAZONES DE LIQUIDEZ-OPERATIVAS		
1	Efectivo en caja+banco/Pasivo Circulante	1
2	Valores Negociables/Pasivo Circulante	1
3	Activos más liquidos(caja+banco+valores n.) /Pasivos Circulantes	1
4	Efectos por cobrar/Pasivo Circulante	1
5	Cuentas por cobrar netas/Pasivo Circulante	1
6	Provision para cuentas por cobrar /Pasivo Circulante	1
7	Activos Realizables(efectos+CxC-cuentas incob.)/Pasivo Circulante	1
8	Inventarios/Pasivos circulantes	1
9	Prueba acida, Liquidez =Activo Circulante-Inventario/Pasivos Circulantes +1	1
10	Indice de capital de Trab=Activos Circulantes/Pasivo Circulante	1
11	Gastos Diarios de operaciones=Costo de venta+Gastos adm.y gen./360	1
12	Razón activo defensivo básico=EfectivoCyB+Valores Neg.+CxC/gastos diarios	1
13	Coeficiente de hacerle frente a deudas=Fondo de maniobra/Deudas a corto plazo+Q1	1
II. Razones de Rotación o Ciclo de Operaciones:		
14	Rotacion de Inventarios=Costo de Venta/Inventario + alto mejor	1
15	Costo Total/Inventario Promedio o Saldo de Inventario	1
16	Rotación de Activos= Ventas/Activo Total + alto mejor	1
17	Rotacion de activos Circ.=Ventas/Activo Circ. + alto mejor	1
18	Costo de Ventas/Capital de trabajo	1
19	Costo Total/Capital de Trabajo	1
20	Saldo de Inventario/Costos de ventas	1
III. Razones de Cobro y Pago		
21	Cuentas por Cobrarx360/Ventas Netas	1
22	Efectos y otros Documentos por cobrar x 360/ Ventas netas	1
23	Periodo de diferimiento de las cuentas por pagar=Cuentas por pagar/Costo de Venta/360 + alto + fcto. Proveedores	1
24	Relación de cobros Pagos= Días de venta sin cobrar/Días de compras sin pagar +0.5 (de1a2)	1
25	Período de Cobranza de las CXC = CxC + ExC/VENTAS/365 + bajo mejor	1
26	Período de conversión del Inventario=INV/(VENTAS/365)	1
27	Ciclo de Conversión del Efectivo o ciclo de maduracion	1
28	COEF.FINANC.ACTV.CIRCULANTE = EXIGIBLE A CP (PC) x 100/ACTV.C	1
29	ROTACIÓN DEL CAPITAL = 365/ CICLO DE MADURACIÓN	1
IV. Razones de Deuda-Solvencia:		
30	Capital contable=Activo total - Pasivo Total	1

31	Pasivo al corto plazo/Capital Contable	1
32	Pasivo a largo plazo/Capital Contable	1
33	Pasivo Total/Capital Contable 1.4 Estándar	1
34	Endeudamiento en el corto plazo=Pasivo al corto plazo/Activo total	1
35	Endeudamiento en el largo Plazo=Pasivo a largo plazo/Activo Total	1
36	Endeudamiento Total=Pasivo Total/Activo Total (De 0.62 a 0.72)	1
37	Activo Total/Pasivo Total	1
38	Activos Fijos Tangibles e Intangibles/Pasivo Total	1
39	Gastos Financieros/Activo Total	1
40	SOLVENCIA = ACTIVO CIRCULANTE/PASIVO CIRCULANTE	1
VIII. Razones de Resultado.		
41	Factor operativo(u.a.i.i o e.b.i.t)=utilidad antes de intereses e impuestos/ventas netas	1
42	Factor financiero=utilidad del periodo / utilidad antes de intereses e impuestos	1
43	Rentabilidad sobre las ventas=utilidad del periodo / ventas netas	1
44	Utilidad del periodo / activos fijos (tangibles e intangibles) netos + Saldo de inventario	1
45	Utilidad del periodo / activos fijos (tangibles e intangibles) netos	1
46	Utilidad del periodo / capital contable	1
47	Utilidad del periodo/ activo total	1
48	Utilidad del periodo / pasivo total	1
La Estructura Financiera.		
49	Activos Líquidos / Activo Total	1
50	Activos Realizables / Activo Total	1
51	Existencias o Saldos de Inventario / Activo Total	1
52	Activo Circulante/Activo Total	1
53	Activos Fijos/Activo Total	1
54	Depreciación o Amortización Acumulada / Activo Total	1
55	Diferencia de Activo Fijo/Activo Total con Depreciación/activo total	1
	Activo Total (igual 1)	
	PASIVOS	
56	Pasivos al corto plazo/Activo Total	1
57	Pasivos al Largo plazo/Activo Total	1
58	Pasivo Total/Activo Total	1
59	Capital por acciones/Activo Total	1
60	Reservas de la entidad/Activo Total	1
61	Utilidad del período/Activo Total	1
62	Capital contable/Activo Total	1
	Pasivo y Patrimonio (igual 1)	
Análisis de la Estructura Financiera		
63	Estructura del Pasivo= Pasivo al corto plazo /Pasivo al largo plazo	1
64	Estructura del activo= Activos circulantes /Activos Fijos Tangibles Intangibles	1

	Netos	
65	Relación Estructura del pasivo/ Estructura del Activo	1
66	Pasivo total / Activo total	1
67	Capital Contable/ Activos Fijos Tangibles Intangibles Netos	1
68	Pasivo Total / Capital Contable	1
69	Utilidad del período / Capital Contable	1
70	Capital Contable / Activo Total.	1
Razones para el análisis de las políticas a tomar por la Dirección.		
71	Activos circulantes / Pasivos al corto plazo	1
72	Capital Contable / Activos Fijos Tangibles Netos	1
73	Capital Contable / Pasivo Total	1
74	Ventas Netas / Saldo de cuentas por cobrar	1
75	Ventas Netas / Saldo de Inventarios.	1
76	Ventas Netas / Activos Fijos (tangibles e intangibles) Netos	1
77	Ventas Netas / Capital Contable	1
Otros Indicadores		
COMPORTAMIENTO DE LAS CUENTAS POR COBRAR.		
###	Ctas/Cob.a CP.	3
###	VENTAS	3
###	% que Representa	3
###	Efectos por Cobrar	3
###	TOTAL CTAS/COBRAR	3
###	VENTAS	3
###	% que Representa	3
###	Total de ctas por cob./Capital de T.	3
I. POSICIÓN FINANCIERA:		
###	solvencia = activo circulante/pasivo circulante 1.5 - 2	2
###	liquidez = activo circulante - inv / pasivo circulante +q1	2
###	coeficiente de hacerle frente a las deudas = fondo de maniobra / deudas a corto plazo +q 1	2
2.CICLO DE OPERACIONES		
###	rotación de inventarios = costo ventas/inventario + alto mejor	2
###	rotación de activos = ventas/activo total + alto mejor	2
###	rotación de activos circulante=ventas/activo circulante + alto mejor	2
###	período de cobranza de las cxc = cxc + exc/ventas/365 + bajo mejor	2
###	ciclo de pagos = cxp/ventas/365 + alto + fcto. proveedores	2
###	período de conversión del inventario=inv/(ventas/365)	2
###	ciclo de conversión del efectivo	2
###	coef.financ.actv.circulante = exigible a cp (pc) x 100/actv.c	2
###	rotación del capital = 365/ ciclo de maduración	2

III.RAZONES DE ENDUDAMIENTO		
###	endeudamiento total=pasivo total/activo total <0.4 + capitalpropio >0.6 descapitalizado	2
###	endeudamiento a lp = pasivo lp / patrimonio (veces)	2
###	st / garantía = activos reales / recursos ajenos +>1 mejor / <1 quiebra tec.	2
###	apalancamiento financiero =resultado antes de imp/(resultado neto x actv.total/capital total > 1	2
IV.RAZONES DE RENTABILIDAD		
###	margen bruto = resultado bruto/ventas (%)elevado	2
###	margen operativo=resultado operativo/ventas (%) elevado	2
###	margen neto=utilidad neta/ventas (%) elevado	2
###	rentabilidad s/activo = utilidad neta/activos (%) elevado	2
###	rentabilidad s/patrimonio = utilidad neta/capital . (%) elevado	2
###	rentabilidad economica=util.ai+gast. fin/activo real (%) elevado	2
###	autofinanciamiento	2
V.CAPACIDAD DE PAGO		
###	amortz. k deuda lp y cp=capital/5	2
###	fondo de maniobra = activo circulante - pasivo circulante	2
###	capacidad de devolución de la deuda	2
###	recursos generados respecto a las ventas	2
ESTRUCTURA DE LOS PASIVOS CIRCULANTES.		
###	Sobregiro y prestamos bancarios	3
###	Ingresos anticipados	3
###	Provisiones	3
###	Cuentas por Pagar	3
###	Nominas, Retenciones, otros.	3
###	Otros Pasivos Circulantes	3
ESTRUCTURA DE LOS ACTIVOS CIRCULANTES.		
###	Efectivo Caja	3
###	Efectivo Banco	3
###	Efectos por cobrar	3
###	Valores Negociables	3
###	Cuentas por Cobrar Netas	3
###	Inventarios	3
###	Otros activos Circulantes	3
###	Activos menos Liq.	3
###	% de Activos menos Liq.	3
NECESIDAD EFECTIVA DE CAPITAL DE TRABAJO		
###	Patrimonio.	3
###	(+) Pasivos a Largo Plazo.	3

###	(-) Activos Fijos.	3
###	(-) Activos a Largo Plazo.	3
###	Capital de Trabajo Efectivo.	3
###	Activos Circulantes.	3
###	(-) Pasivos Circulantes.	3
###	Capital de Trabajo Requerido.	3
###	Capital de Trabjo Efectivo.	3
###	(-) Capital de Trabajo Requerido.	3
###	Superavit (Déficit) de Capital de Trabajo.	3
Otros		
###	Días de reponer el inventario	3
###	Ratio de garantía (mientras mayor que 1 es mejor, por debajo de 1 quiebra.	3
###	Gastos financieros sobre ventas	3

2.8 Conclusiones Parciales

En este capítulo se ha tratado el tema de la Elicitación como una primera actividad dentro de la Ingeniería de Requerimientos desarrollada en el módulo Análisis de Riesgo de Crédito, por ello se han aplicado una serie de técnicas con el fin de comprender la estructura y la dinámica de la entidad, destacándose la modelación del negocio.

El estudio realizado determinó que el nuevo sistema deberá integrar muchas de las operaciones que se realizan actualmente en sistemas por separados, de igual forma gran cantidad de información que se maneja hoy en papel, con el nuevo sistema se manejará electrónicamente, lográndose con ello, mayor eficiencia en la entidad.

La Especificación de Requisitos, resultado de la modelación del negocio, constituyó el punto de partida para representar las funcionalidades que el módulo Análisis de Riesgo de Crédito debería tener para darle cumplimiento a las necesidades de los clientes del BNC. A partir de esa primera ERS se obtuvo la definitiva, modelada en el Diagrama de Casos de Uso del módulo.

Las descripciones de los casos de uso y sus prototipos sirvieron para validar con los clientes los resultados de la Ingeniería de Requisitos desarrollada, corrigiéndose así defectos que pudieran aflorar en etapas futuras del desarrollo de software. De cuya validación se obtuvo gran satisfacción por parte de los clientes, objetivo principal de todo este proceso. Además, se identificaron los indicadores contables que se utilizarán para el sistema.

CAPÍTULO 3: ANÁLISIS Y DISEÑO

3.1 Introducción

En el presente capítulo se realiza el análisis y diseño de la propuesta de solución. Se muestran los artefactos generados y se mencionan alguna de las tantas terminologías que se utilizan. Es importante destacar que aunque RUP contempla Análisis y Diseño en la misma disciplina por estar muy relacionadas, son actividades diferentes con artefactos diferentes.

3.2 Análisis

Durante el análisis, se analizan los requisitos que se describen en la captura de requerimientos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y que nos ayude a estructurar el sistema entero, -incluyendo su arquitectura" (Jacobson Ivar, et al., 2000).

De forma general el análisis consiste en transformar los requisitos funcionales en un diseño de clases, en el cual se vean las relaciones e interacciones que existe entre las clases. Teniendo presente en este proceso una arquitectura robusta, que permita adaptar el sistema al entorno de implementación que se está desarrollando. Además, en este flujo se obtiene una visión del sistema que se preocupa de ver que hace, de tal forma que se preocupa solo por los requisitos funcionales.

3.2.1 Modelo del análisis

El lenguaje que se utiliza en el análisis se basa en un modelo de objetos conceptual, que se denomina modelo del análisis. El modelo de análisis contribuye a refinar los requisitos, así como a razonar sobre los aspectos internos del sistema. A pesar de que el modelo del análisis hay un refinamiento de los requisitos, no se toman en cuenta aspectos que afectan al sistema como el lenguaje de programación, la plataforma en la que se ejecutará la aplicación, los componentes prefabricados o reusables de otras aplicaciones, el objetivo del análisis es comprender perfectamente los requisitos del software y no precisar cómo se implementará la solución.

El modelo del análisis es descrito por el lenguaje del desarrollador, presenta una vista interna del sistema, es estructurado por clases y paquetes estereotipados; proporciona la estructura a la vista interna. No

debería contener redundancias, inconsistencias, entre requisitos. Esboza cómo llevar a cabo la funcionalidad dentro del sistema incluida la funcionalidad significativa para la arquitectura; sirve como una primera aproximación del diseño. Define realizaciones de casos de uso, y cada una de ellas representa el análisis de un caso de uso del modelo de casos de uso.

3.2.1.1 Clases del análisis

Las clases del análisis se centran en los requisitos funcionales y son evidentes en el dominio del problema porque representan conceptos y relaciones del dominio. Tienen atributos y entre ellas se establecen relaciones de asociación, agregación / composición, generalización / especialización y tipos asociativos. RUP propone clasificar a las clases en:

3.2.1.2 Diagrama de clases del análisis

Un Diagrama de clases del análisis es un artefacto en el que se representan los conceptos en un dominio del problema. Representa las cosas del mundo real, no de la implementación automatizada de estas cosas.

A continuación los diagramas de clases del análisis:

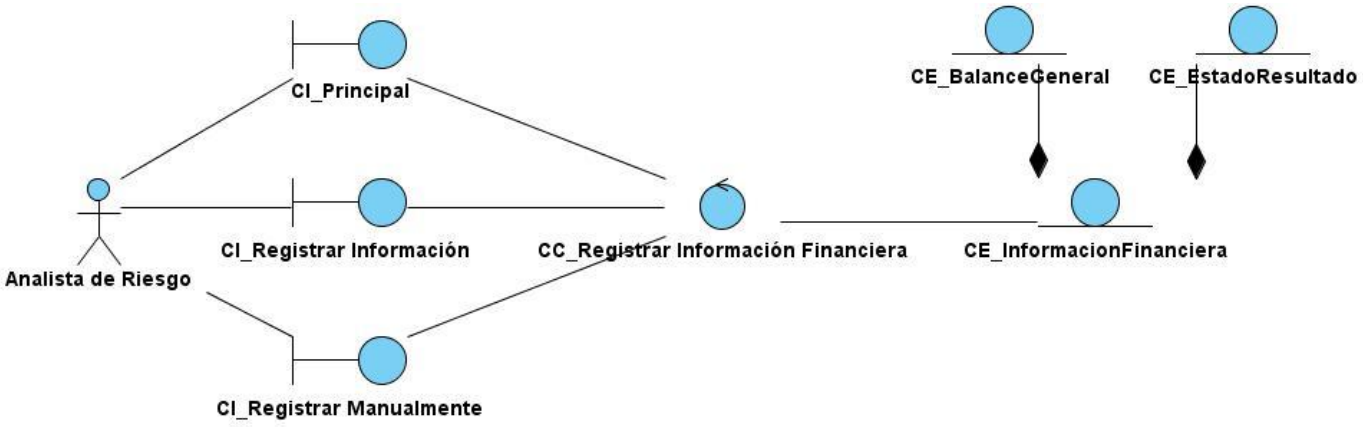


Figura 14: Diagrama de Clase de Análisis Registrar Información Manualmente

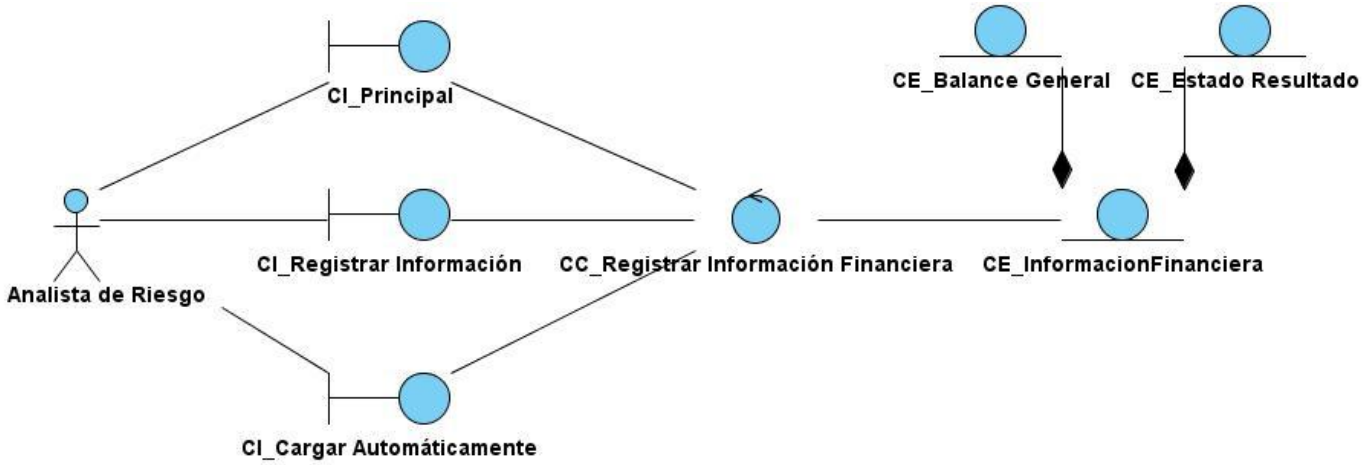


Figura 15: Diagrama de Clase de Análisis Cargar Información Automáticamente

3.2.1.3 Diagrama de interacción (colaboración)

Un diagrama de colaboración muestra una interacción organizada en torno a los objetos que efectúan operaciones. Es parecido a un diagrama de objetos que muestra los objetos y los enlaces existentes entre ellos que se necesitan para implementar una operación de nivel más elevado.

Una colaboración modela los objetos y los enlaces significativos dentro de una interacción. Los objetos y los enlaces son significativos solamente en el contexto proporcionado por la interacción. Un rol describe un objeto, y un rol en la asociación describe un enlace dentro de una colaboración.

Un diagrama de colaboración muestra los roles en la interacción en una disposición geométrica. Los mensajes se muestran como flechas, ligadas a las líneas de la relación, que conectan a los roles. La secuencia de mensajes, se indica con los números secuenciales que preceden a las descripciones del mensaje.

A continuación los diagramas de clases de colaboración:

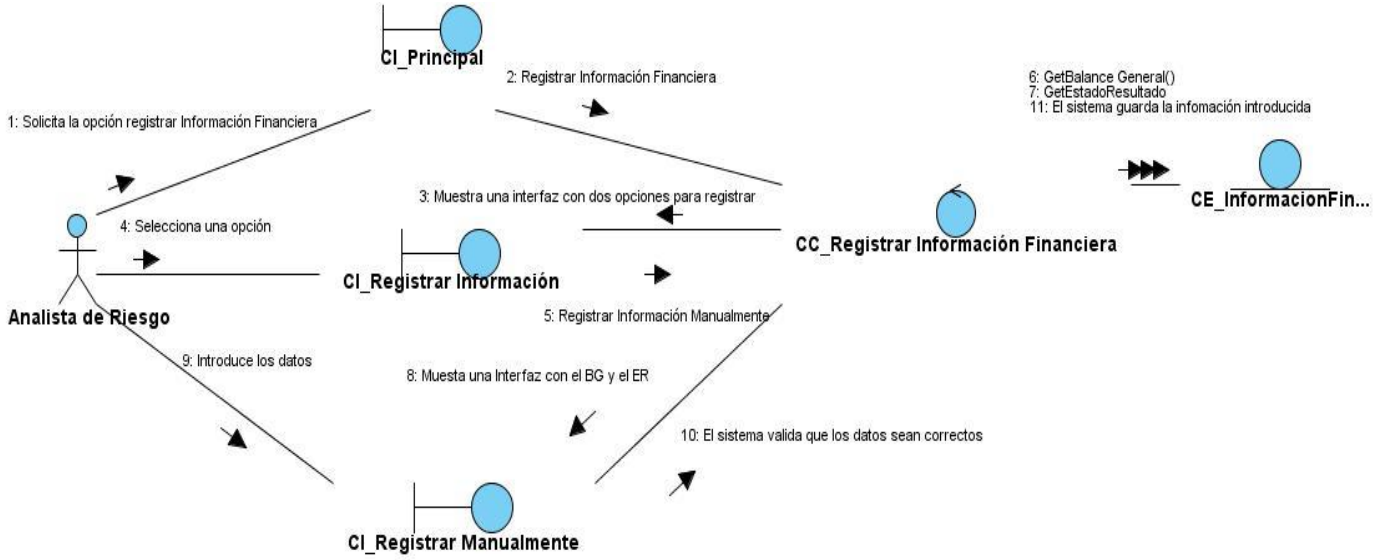


Figura 16: Diagrama de Colaboración Registrar Información Manualmente

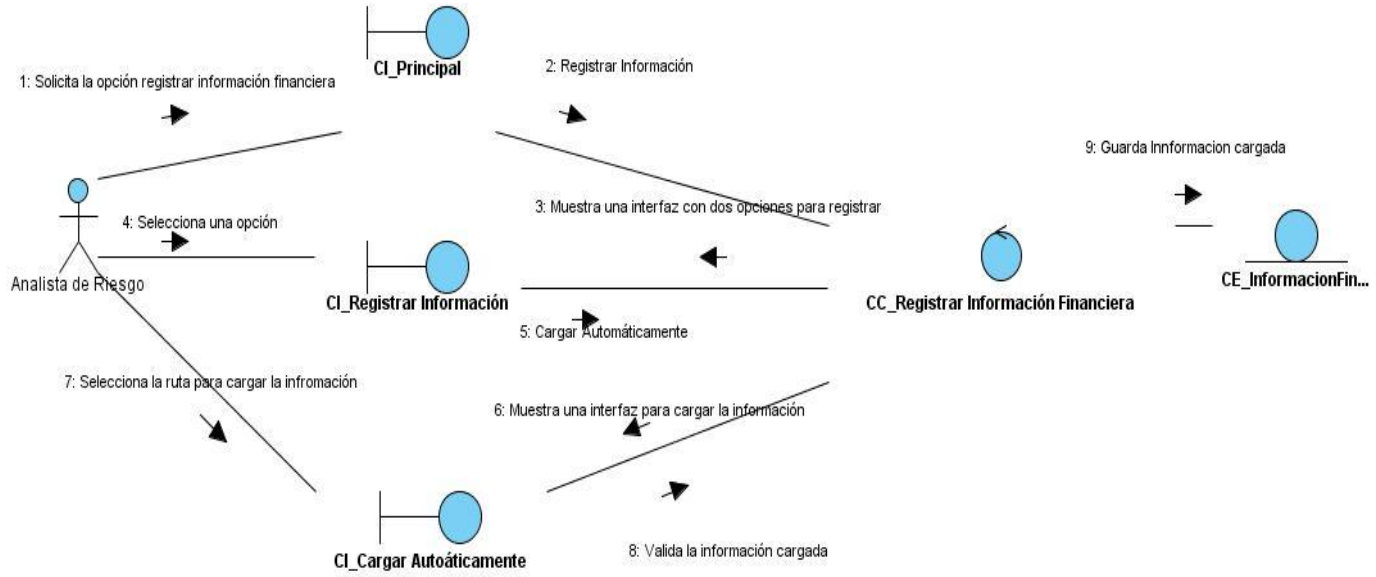


Figura 17: Diagrama de Colaboración Cargar Automáticamente

3.3 Diseño

RUMBAUGH, JACOBSON y BOOCH, en el libro “El lenguaje unificado de modelado”, señalan que el diseño es: “esa etapa de un sistema que describe cómo se implementará el sistema, en un nivel lógico

sobre código real. En el diseño, las decisiones estratégicas y tácticas se toman para resolver los requisitos funcionales y de calidad requeridos de un sistema. Los resultados de esta etapa son representados por los modelos a nivel de diseño, especialmente la vista estática, vista de la máquina de estados, y vista de interacción. Contrastar con: análisis, diseño, implementación, y despliegue (Rumbaugh, et al., 2001)

De forma general en el diseño se modela el sistema, contribuyendo a una arquitectura estable y sólida, para que soporte todos los requisitos, tanto funcionales como no funcionales. El diseño posibilita una entrada apropiada y un punto de partida para las actividades de la implementación, descompone los trabajos de implementación en partes más manejables que pueden ser llevados a cabos por diferentes equipos de desarrollo.

El diseño del sistema contribuye a una arquitectura estable y sólida en la creación del modelo de implementación. El modelo de diseño se puede utilizar para visualizar la implementación y para soportar las técnicas de programación gráfica.

3.3.1 Descripción de la arquitectura propuesta

Los Componentes son un conjunto de funcionalidades comunes que serán reutilizados por otros módulos del sistema. Estos componentes en algunas ocasiones se comportarán como módulos visuales en el sistema, y en otras ocasiones solamente recogerán funcionalidades del negocio.

Diseño de las capas lógicas

Para ganar en organización en el desarrollo y en el despliegue del sistema, se agruparán los Módulos y Componentes por Subsistema, cada Subsistema tendrá uno o más Módulos y/o Componentes estrechamente relacionados con las funcionalidades que ejecutan. Los Módulos y/o Componentes estarán separados por diferentes capas lógicas según la naturaleza de los mismos.

Las capas lógicas definidas son:

Capa de Presentación: Esta capa estará dividida en dos partes. Una subcapa del lado del servidor, encargada de recibir todos los pedidos de la interfaz de usuario, controlar el flujo de presentación del

sistema y enviar las respuestas correspondientes a la interfaz de usuario. La otra subcapa estará en el cliente, utilizándose los componentes visuales de Java Script para manejar los eventos y validaciones del lado del cliente. La subcapa colocada en el lado del servidor estará relacionada con la capa de Negocios y de Dominio.

Capa de Negocios: Esta capa está divida en dos subcapas principales sin dejar de incluir otras que se necesiten y que estén relacionadas con el negocio. En la Fachada se expondrán todas las funcionalidades que la capa de presentación necesitará. Esta capa invocará métodos de la subcapa de Desarrollo del negocio. En la capa de Desarrollo del negocio se implementará el negocio de los módulos en cuestión, y de aquí se accederá de ser necesario a la Capa de Acceso a Datos, a otras Capas de Negocios y/o a la Capa de Dominio.

Capa de Acceso a Datos: En esta capa se implementarán los métodos encargados en interactuar con el gestor de base de datos. Esta capa tendrá solamente dependencia con la Capa de Dominio.

Capa de Dominio: En esta capa se declararán todas las clases que representan entidades del negocio. Estas clases de dominio estarán presentes en todas las capas anteriormente descritas.

A continuación se muestra una figura con la estructura de las capas lógicas descritas anteriormente.

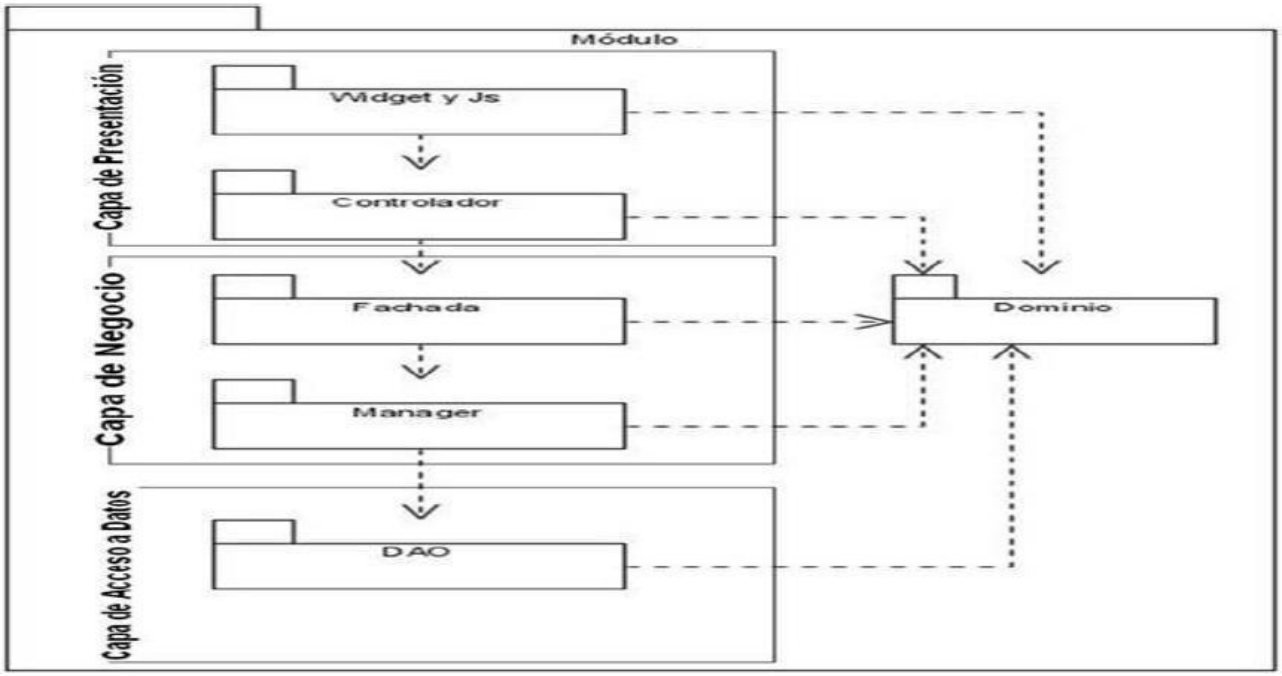


Figura 18: Estructura de las capas lógicas del sistema.

Se muestra otra figura con las capas lógicas y los frameworks que se utilizarán en cada una

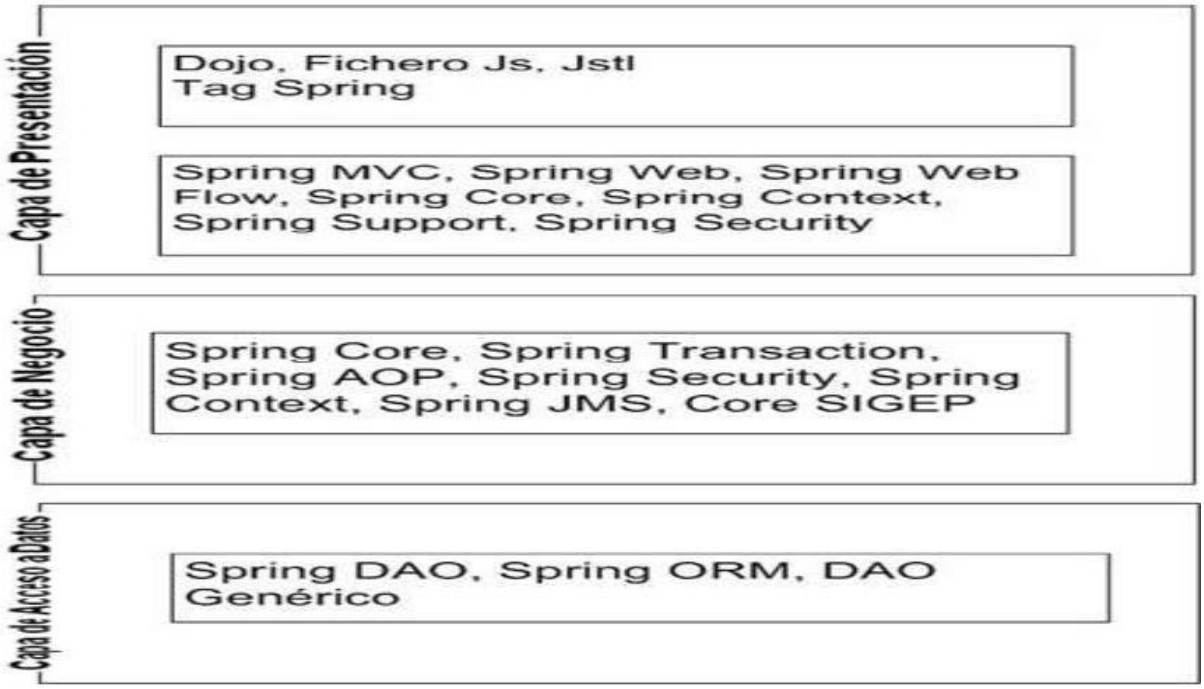


Figura 19: Relación entre capas lógicas y los frameworks.

3.3.1.1 Patrones a utilizar

El hombre durante su historia ha dominado muchas técnicas. El desarrollo de un software es una tarea complicada, la cual depende en gran medida de la experiencia del equipo de desarrollo. En ocasiones las operaciones se realizan de forma artesanal, los desarrolladores aprenden por un proceso de ensayo y errores y por transmisión de otros desarrolladores, se desarrollan técnicas generalmente aceptadas en el área de trabajo, lográndose un conocimiento común sobre cómo aplicar estas técnicas y además se crea una ciencia alrededor de la tarea; esto ha provocado la evolución de la utilización de estándares de solución a un problema de forma efectiva y reutilizable, es decir, la utilización de un patrón que facilite la solución del problema. El patrón describe un problema que ocurre infinidad de veces en nuestro entorno, así como la solución al mismo, de tal modo que se pueda utilizar esta solución un millón de veces más adelante sin tener que volver a pensarla otra vez.

Patrones de asignación de Responsabilidades GRASP

En los patrones GRASP se codifican algunos de los principios, que se aplican al preparar los diagramas de interacción.

Patrón Experto**:** Asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad (LARMAN, 1999).

Patrón Bajo Acoplamiento: Asignar las responsabilidades de forma tal que las clases se comuniquen con el menor número de clases que sea posible (LARMAN, 1999).

Patrón Alta Cohesión*:* Asignar a las clases responsabilidades que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad.

- Muy baja cohesión. Una clase es la única responsable de muchas cosas en áreas funcionales muy heterogéneas.
- Baja cohesión. Una clase tiene la responsabilidad exclusiva de una tarea compleja dentro de un área funcional.
- Alta cohesión. Una clase tiene responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas (LARMAN, 1999).

Patrón Controlador: Asignar la responsabilidad del manejo de mensajes de los eventos del sistema a una clase (LARMAN, 1999).

El diseño obtenido cumple con los patrones de Bajo acoplamiento y Alta cohesión permitiendo la colaboración entre los elementos del diseño, sin verse afectados la reutilización de los mismos y el entendimiento de estos cuando se encuentran aislados.

La creación de clases controladoras facilitó realizar las operaciones del sistema, debido a que estas operaciones reflejan los procesos de la empresa o dominio y no es factible manejarse en la capa de interfaz o presentación.

Patrón Estructural

Facade*:* Simplifica el acceso a un conjunto de clases o interfaces. Proporciona una interfaz unificada de un subsistema sin ocultar sus interfaces. Permite acceder a elementos del sistema y realizar operaciones más complejas con ellos de forma transparente. Reduce la dependencia entre clases. Ofrece un punto de acceso al resto de clases, si estas cambian o se sustituyen por otras, sólo hay que actualizar la clase Facade sin que el cambio afecte a las aplicaciones cliente

Patrón de Comportamiento

Command: El objetivo de utilizar este patrón es tener parametrizados los objetos por las acciones que realizan. Este patrón permite especificar, administrar y ejecutar solicitudes en tiempos distintos. Puede guardar un estado que permita deshacer la ejecución del comando. Soporta la capacidad de generar bitácoras que permitan la recuperación del estado en caso de que el sistema falle. Facilita la estructuración del sistema en torno a operaciones de alto nivel construidas con base en operaciones primitivas o de bajo nivel. Un comando nos desliga el objeto invocador del objeto receptor. Permite que las acciones sean objetos de primera clase y se puedan agrupar comandos de uso frecuente en comandos compuestos.

Patrón de acceso a datos (DAO)

DAO: La utilización de este patrón permitirá acceder a la fuente de datos y encapsular los objetos clientes, ocultando tanto la fuente como el modo de acceder a ella. Los DAOs deben implementar los métodos del interface (InterfaceDAO) que declaran. Pero además pueden implementar otros métodos que no están en el interfaz. DAO permite el acceso a reglas de validación, esto es posible porque tiene capacidad de especificar relaciones entre tablas.

3.3.2 Modelo del diseño

El modelo del diseño es un modelo de objetos que describe la realización física de los casos de usos centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas en el entorno de implementación, tienen un impacto en el sistema a considerar y sirve de abstracción a la implementación y al código fuente del sistema. Contiene protocolo, cápsula, realización de casos de usos, señales, eventos, subsistema de diseño, paquetes de diseño, interfaces, clases del diseño, clases de prueba, diseño de pruebas.

3.3.2.1 Clases del diseño

Server Page (página servidora): representa la página web que tiene código, que se ejecuta en el servidor.

Client Page (página cliente): una instancia de Página Cliente es una página web, con formato HTML.

Form (formulario): colección de elementos de entrada que son parte de una página cliente.

<<Build>>: representa una asociación especial que relaciona las páginas cliente con las páginas servidor.
<<Link>>: expresa las asociaciones más comunes entre las páginas, en este caso la del hipervínculo.
<<Submit>>: es la relación que se crea siempre entre una página servidor y un formulario.

3.3.2.2 Diagramas de clases del diseño

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los diagramas de clases se utilizan para modelar la vista de diseño estática de un sistema, esto incluye modelar el vocabulario del sistema, modelar las colaboraciones o modelar esquemas.

A continuación los diagramas de clases del diseño.

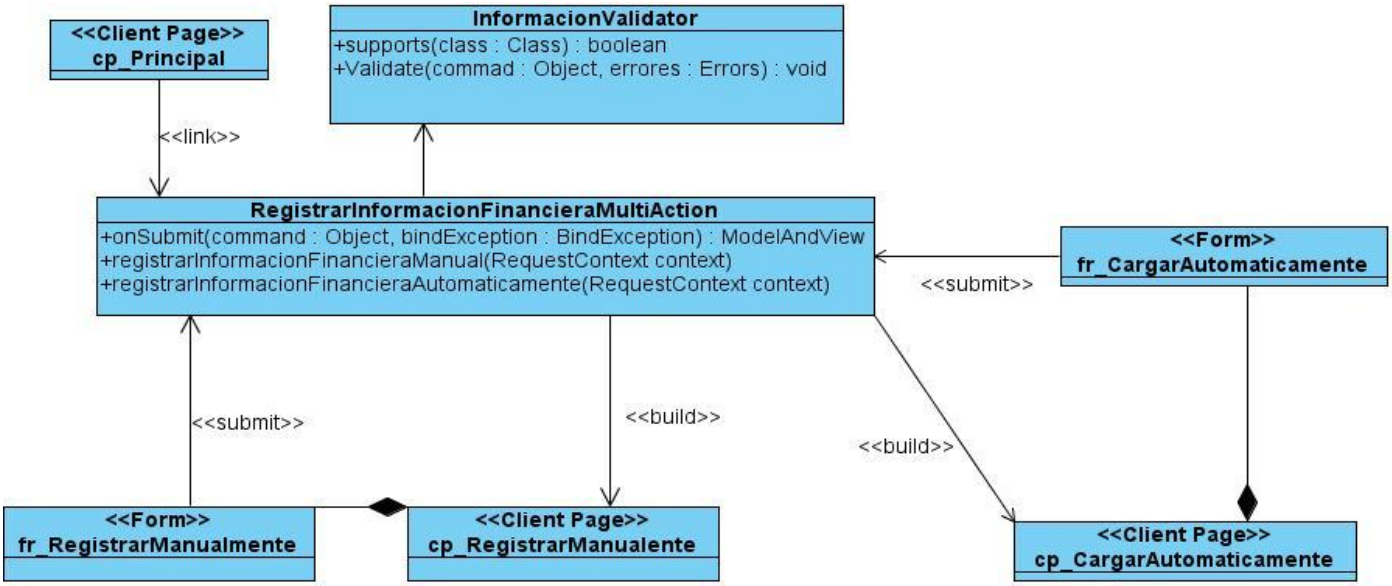


Figura 20: Diagrama de clases del Diseño. Registrar Información Financiera. Capa presentación.

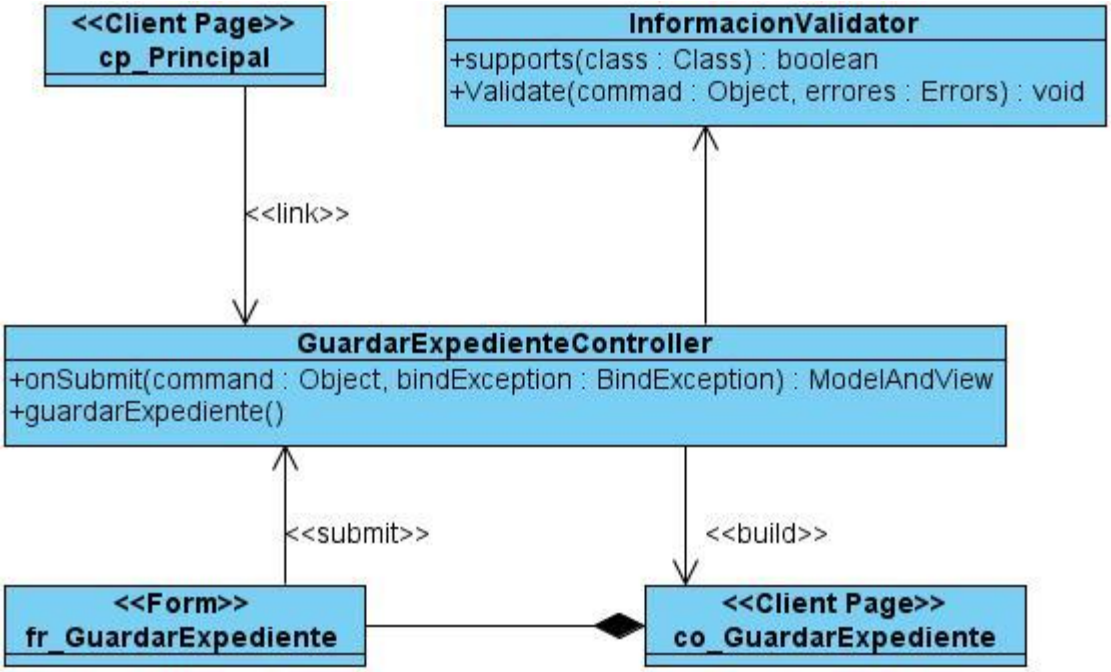


Figura 21: Diagrama de clases del Diseño. Guardar Expediente. Capa de presentación.

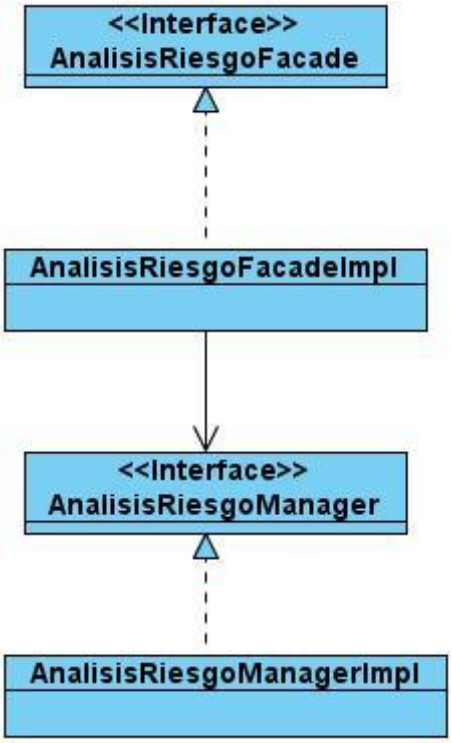


Figura 22: Diagrama de clases del diseño. Capa de negocio.

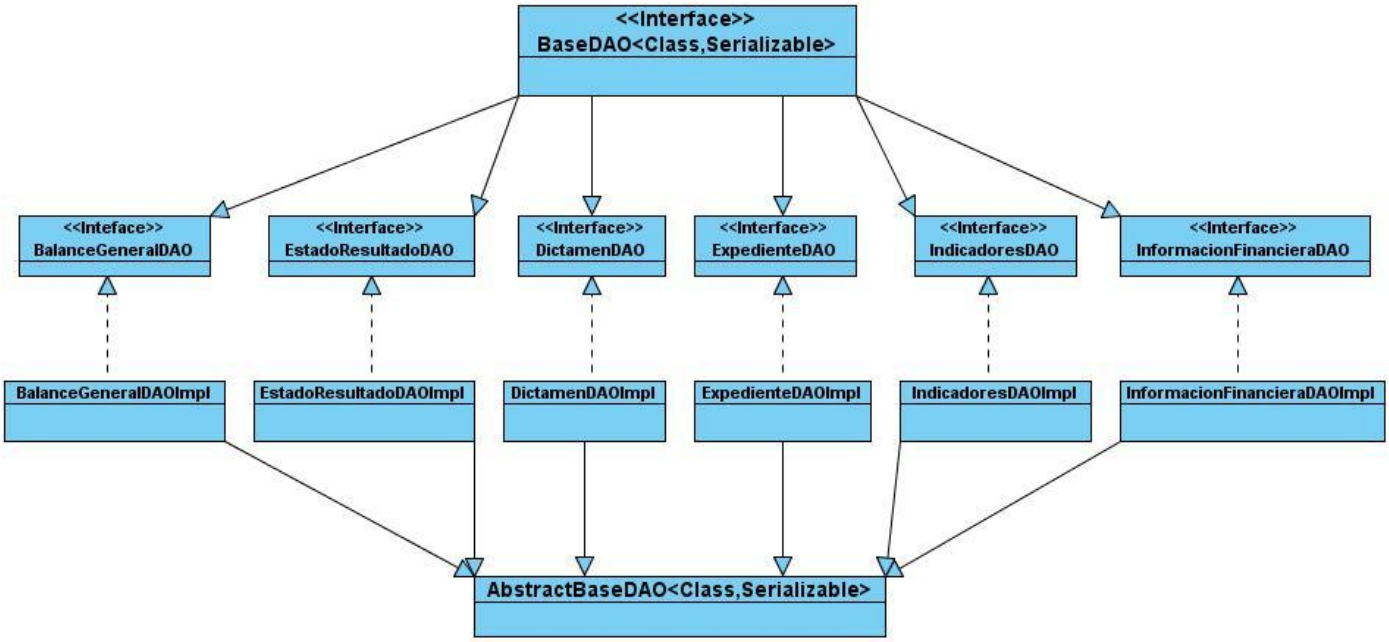


Figura 23: Diagrama de clases del diseño de las clases más representativas. Capa de acceso a datos.

Descripción de las principales clases de la lógica de negocio y de acceso a datos.

AnalisisRiesgoFacade: Interfaz de la capa de lógica de negocio que contiene las funcionalidades relacionadas con el sistema, brindadas a la capa de presentación y a la vez le sirve de fachada.

AnalisisRiesgoFacadeImpl: Clase que implementa las funcionalidades definidas en la interfaz "*AnalisisRiesgoFacade*". En esta no se implementa ninguna lógica de negocio, simplemente se limita a delegar las responsabilidades a la clase Manager correspondiente.

AnalisisRiesgoManager: Interfaz que define las funcionalidades de lógica de negocio específicas para el sistema de Análisis de Riesgo.

AnalisisRiesgoManagerImpl: Interfaz que implementa los métodos de lógica de negocio definidos en la interfaz "*AnalisisRiesgoManager*". Se encarga de implementar las funcionalidades brindadas por la fachada.

BaseDAO: Es una interfaz que agrupa funcionalidades comunes relacionadas con los métodos de acceso a datos.

AbstractBaseDAO: Esta clase abstracta, que no implementa ninguna lógica de acceso a datos, ella sólo se encarga de delegar esas responsabilidades a los DAOs**.**

EstadoResultadoDAO: Es una interfaz que brinda las funcionalidades correspondientes al manejo del acceso a datos para el estado resultado.

EstadoResultadoDAOImpl: Esta clase implementa la interfaz "*EstadoResultadoDAO"*. En ella se implementa el manejo del acceso a datos correspondientes al estado resultado.

BalanceGeneralDAO: Es una interfaz que brinda las funcionalidades correspondientes al manejo del acceso a datos para el Balance General.

BalanceGeneralDAOImpl: Esta clase implementa la interfaz "*BalanceGeneralDAO"*. En ella se implementa el manejo del acceso a datos correspondientes al Balance General.

IndicadoresDAO: Es una interfaz que brinda las funcionalidades correspondientes al manejo del acceso a datos para los Indicadores Contables.

IndicadoresDAOImpl: Esta clase implementa la interfaz "*IndicadoresDAO"*. En ella se implementa el manejo del acceso a datos correspondiente los Indicadores Contables.

ExpedienteDAO: Es una interfaz que brinda las funcionalidades correspondientes al manejo del acceso a datos para el Expediente de la Operación.

ExpedienteDAOImpl: Esta clase implementa la interfaz "*ExpedienteDAO"*. En ella se implementa el manejo del acceso a datos correspondiente Expediente de la Operación.

DictamenDAO: Es una interfaz que brinda las funcionalidades correspondientes al manejo del acceso a datos para el Dictamen Final.

DictamenDAOImpl: Esta clase implementa la interfaz "*DictamenDAO"*. En ella se implementa el manejo del acceso a datos correspondiente Dictamen Final.

InformacionFinancieraDAO: Es una interfaz que brinda las funcionalidades correspondientes al manejo del acceso a datos para la información financiera.

InformacionFinancieraDAOImpl: Esta clase implementa la interfaz "*InformacionFinancieraDAO"*. En ella se implementa el manejo del acceso a datos correspondiente información financiera.

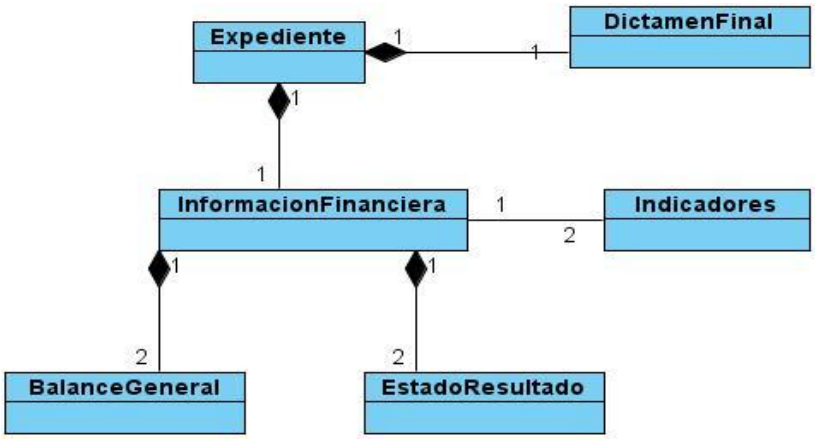


Figura 24: Diagrama de clases del diseño. Capa de Dominio.

3.3.2.3 Diagramas de interacción (secuencia)

Un diagrama de secuencia es un diagrama de interacción que destaca la ordenación temporal de los mensajes (mensaje simple, síncrono, asíncrono, y/o de retorno). Este tipo de diagrama se destaca por tener la línea de vida y el foco de control que representa el período de tiempo durante el cual un objeto ejecuta una acción.

A continuación los Diagramas de secuencia.

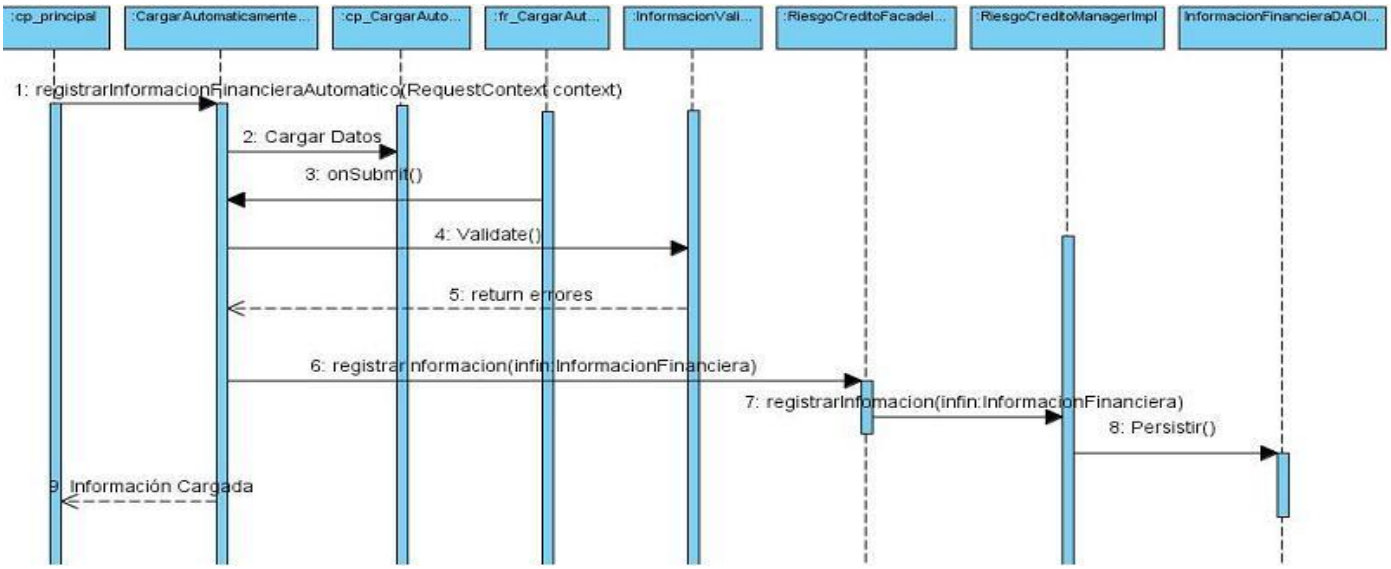


Figura 25: Diagrama de Secuencia Cargar Automáticamente. Flujo Básico

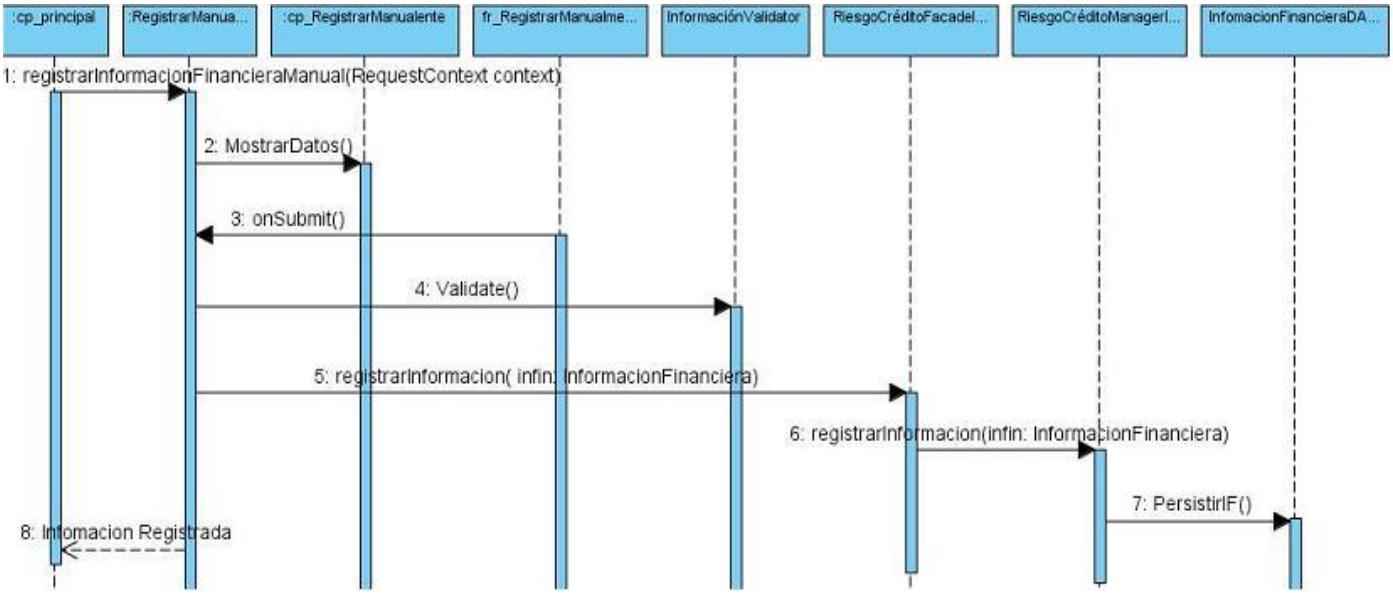


Figura 26 Diagrama de Secuencia Registrar Manualmente. Flujo Básico

3.3.3 Modelo de Datos

El modelo de datos es usado para describir la representación lógica y física de la información persistente manejada por el sistema. A continuación, el modelo de datos:

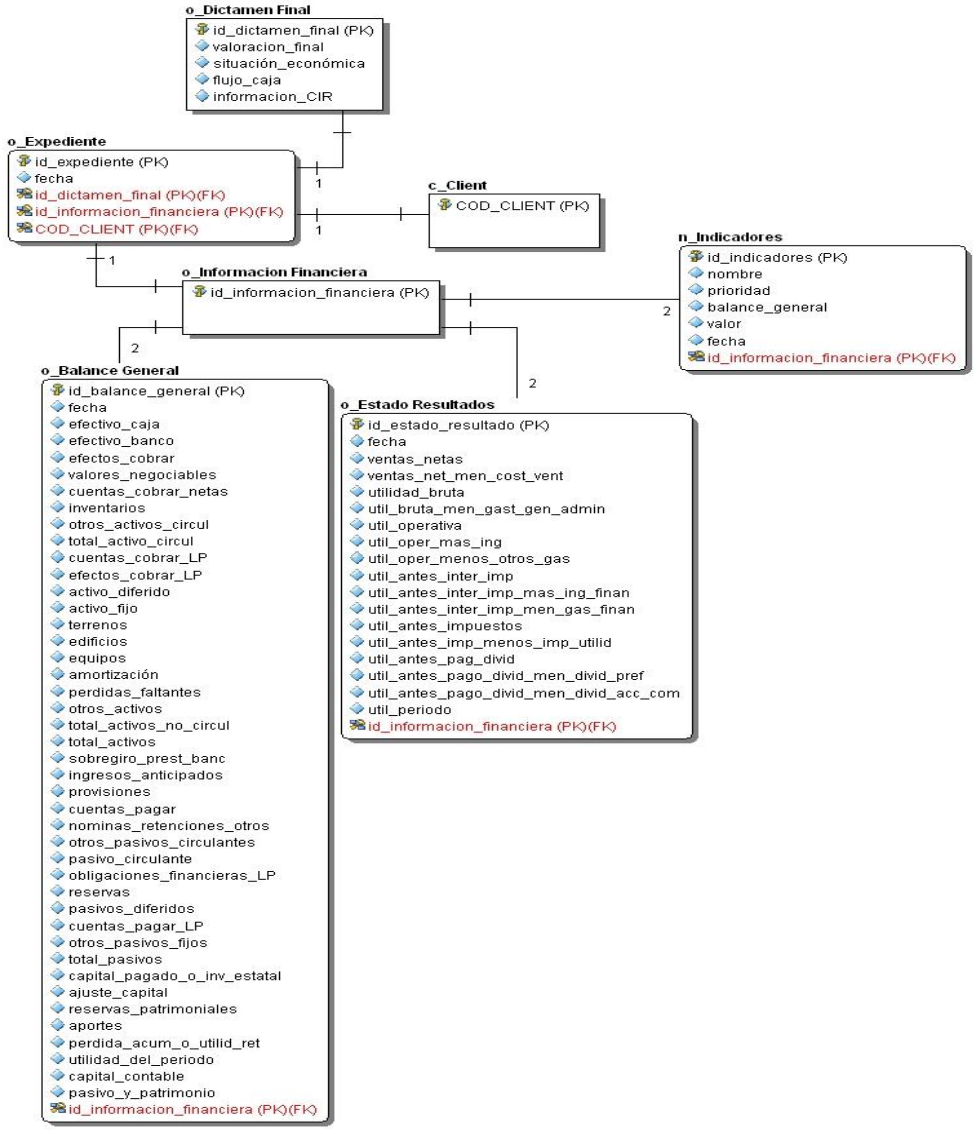


Figura 27: Modelo de Datos

3.4 Conclusiones parciales

En el capítulo se muestran los artefactos generados en el flujo de trabajo de análisis y diseño, además se incluye una breve descripción de los mismos, con el objetivo de que se comprenda perfectamente los requisitos del software. Posibilitando la correcta transformación de los mismos a un diseño que indique como debe ser implementado el software. Además, se describieron los patrones que se utilizaron durante el diseño.

CONCLUSIONES

Una vez concluido el estudio de sistemas para Análisis de Riesgo a nivel mundial, se realizó el análisis y diseño del módulo Análisis de Riesgo de Crédito, lo cual permitirá su posterior implementación.

Se decidió realizar, teniendo en cuenta las características del proyecto, las actividades Elicitación, Especificación y Validación de los requisitos.

En la Elicitación se utilizaron diferentes técnicas, destacándose la técnica modelo de negocio que permitió definir los involucrados, trabajadores, artefactos, reglas y mejoras de los procesos del negocio. Como resultado de la Elicitación se obtuvo una primera versión de los requisitos funcionales del módulo, los cuales fueron refinados y especificados formalmente durante la Especificación de Requisitos. Por último, los requisitos se validaron con los clientes.

Se definieron un total de 9 casos de uso como resultado de la Ingeniería de Requerimientos que representan la propuesta de funcionalidades.

En el del análisis se obtuvieron los diagramas de colaboración y los de las clases del análisis. Para la realización del diseño se tuvieron en cuenta algunos patrones para modelar el sistema, lo que permitió generar artefactos empleando buenas prácticas, necesarias para desarrollar un software con mayor robustez. Entre los artefactos generados en el diseño se pueden mencionar: diagramas de secuencia, diagramas de clases del diseño y el modelo de datos, imprescindibles para el buen desarrollo de una solución informática.

RECOMENDACIONES

- Se recomienda realizar los restantes flujos de trabajo que propone la metodología utilizada, RUP, llegando a implementar las funcionalidades que hemos propuesto para el módulo Análisis de Riesgo de Crédito del proyecto SAGEB.
- Se recomienda profundizar en el estudio de los procesos de Riesgo en otras entidades bancarias para una futura versión ampliada del sistema.

Referencia Bibliográfica

Menéndez Barzanallana, Arsenio Rafael. Informática Aplicada a la Gestión Pública. Facultad de Derecho. Capítulo 1. Ingeniería de Software. Introducción, 2008. [Cited: febrero 7, 2010.] http://www.um.es/docencia/barzana/IAGP/IAGP2-Ingenieria-software-introduccion.html#BM3.

Fernández Pérez, Yudid. Enfoque Administrativo de la Ingeniería de Software. [Online] 2007. [Cited: febrero 7, 2010.] http://www.monografias.com/trabajos-pdf/enfoque-administrativo-ingenieria-software/enfoque-administrativo-ingenieria-software.pdf

Mendoza Sánchez, María A. Metodologías de Desarrollo de Software. Informatizate, 2004. [Cited: febrero 7, 2010.] http://www.informatizate.net/articulos/metodologias_de_desarrollo_de_software_07062004.html

Jacobson, Ivar, Rubaugh, James and Booch, Grady. 2004. El Proceso Unificado de Desarrollo de Software. [Online] 2004. [Cited: febrero 5, 2010.] http://bibliodoc.uci.cu/pdf/reg00060.pdf

Software, I. R. Rational Software ,2007. [Cited: febrero 4, 2010.] http://www.rational.com.

LARMAN, C. UML Y Patrones. Introducción al análisis y diseño orientado a objeto. México 1999. p.

Pressman, Roger S. Ingeniería del Software: Un enfoque práctico. S.l: Félix Varela, 2005. Vol. 1.

Toca Ramos, Lázaro Alberto. 2006. Estudio de la Contabilidad General. La Habana: Félix Varela, 2006.

FOGACOOP. 2005. Riesgo de Crédito. 2005. [Cited: enero 8, 2010.] http://www.fogacoop.gov.co/.../PRESENTACI%D3N%20RIESGO%20DE%20CR%C9DITO.ppt.

Buniak, Leonardo. Gestión de Riesgo para Instituciones Financieras. [Online] [Cited: enero 10, 2010.] http://www.buniak.com/negocio.php?id_seccion=8&id_documento=166.

Rumbaugh, James, Jacobson, Ivar and Booch, Grady. 2001. El lenguaje unificado de modelado. Manual de referencia. [Online] 2001. [Cited: abril 10, 2010.] http://bibliodoc.uci.cu/pdf/reg03050.pdf.

Escalona, María José y Koch, Nora. 2002. Ingeniería de Requisitos en Aplicaciones para la WEB. Un estudio comparativo. [En línea] Diciembre de 2002. http://www.lsi.us.es/docs/informes/LSI-2002-4.pdf.

M.Jacbson. 2001. Software Requirements and Specifications. s. l. : Addison-Wesley, 2001.

Milián Sardiña, Lic. Beatriz, y otros. 1998. Principales características del Sistema Contable del Banco Central de Cuba. Octubre de 1998. http://www.cemla.org/pdf/pub-di-sis-mb.pdf.

Filcun, Juan Carlos. 2006. Curso: Conocimientos básicos de contabilidad. 1 de Septiembre de 2006. http://www.mailxmail.com/curso/empresa/basicocontabilidad/capitulo5.htm.

Martínez Vilches, Ramón. 1993. 1993. http://www.observatorio-iberoamericano.org/paises/Spain/Art%C3%ADculos%20diversos%20sobre%20Contabilidad%20de%20Gesti%C3%B3n/Nuevos%20retos%20cont%20gest%20en%20entid%20bancarias%20-%20Vilches.htm.

Cerezal Tamargo, Lourdes. La revista del empresario cubano. La contabilidad en una nueva tecnología. http://www.betsime.disaic.cu/secciones/tec_feb_02.htm.

Bibliografía

Bauer, F. 1997. *Software engineering: a practitioner's approach. s. l. : Mc Graw-Hil, 1997. BCC. Evolución del Sistema Bancario y Financiero a partir de 1995. [En línea] http://www.cubagob.cu/des_eco/banco/espanol/sistema_bancario/bcc.htm.*

Bravo Santos, Crescencio y García Rubio, Félix Oscar. *Ingeniería del Software. Metodologías de Desarrollo de Software. [En línea] http://alarcos.inf-cr.uclm.es/per/fgarcia/isoftware/doc/tema3_1xh.pdf.*

Cerezal Tamargo, Lourdes. *La revista del empresario cubano. La contabilidad en una nueva tecnología. [En línea] http://www.betsime.disaic.cu/secciones/tec_feb_02.htm.*

Diccionario de economía y finanzas. [En línea] http://www.eumed.net/cursecon/dic/P8.htm.

Durán Toro, Amador. 2007. *Lenguajes y Sistemas Informáticos. Ingeniería de Requisitos. [En línea] 2007. http://www.lsi.us.es/docencia/get.php?id=2003.*

Fernández Pérez, Yudid. 2007. *Enfoque administrativo de la Ingeniería de Software. [En línea] 2007. http://www.monografias.com/trabajos-pdf/enfoque-administrativo-ingenieria-software/enfoque-administrativo-ingenieria-software.pdf.*

Jacobson, Ivar, Rumbaugh, James y Booch, Grady. 2000. *El proceso Unificado de desarrollo de software. s.l. : Félix Varela, 2000. Vol. Tomo I.*

Pressman, Roger. 2001. *Ingeniería del Software: Un Enfoque Practico. s.l. : MacGraw-Hill, 2001.*

Pressman, Roger S. 2005. Ingeniería del Software: Un enfoque práctico, 2005. http://bibliodoc.uci.cu/pdf/reg02689.pdf

FOGACOOP. 2005. Riesgo de Crédito. 2005. [Cited: enero 8, 2010.] http://www.fogacoop.gov.co/.../PRESENTACI%D3N%20RIESGO%20DE%20CR%C9DITO.ppt.

Buniak, Leonardo. Gestión de Riesgo para Instituciones Financieras. [Online] [Cited: enero 10, 2010.] http://www.buniak.com/negocio.php?id_seccion=8&id_documento=166.

A.V.Vedpuriswar. *Credit Risk Plus and Credit Metrics.* 10 4, 2009. (accessed 4 12, 2010) http://www.vedpuriswar.org/.../Financial%20Risk%20Management/Credit%20Risk%20Plus%20and%20Credit%20Metrics.ppt

Manual de Procedimientos de la Gerencia de Riesgo del Banco Nacional de Cuba.

ANEXOS

Figura 1. La Ingeniería de Software es una tecnología multicapa.

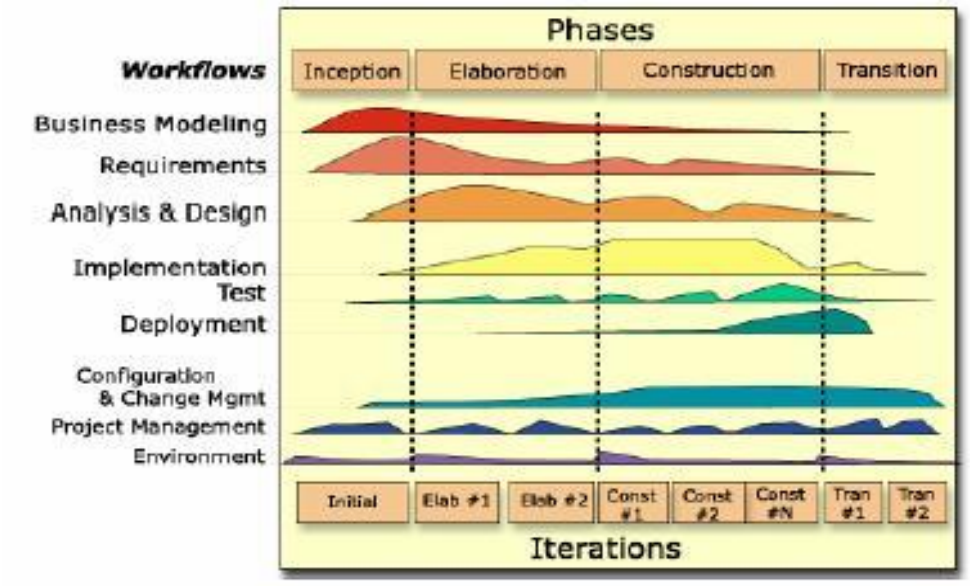


Figura 2. Metodología Rational Unified Process (RUP)

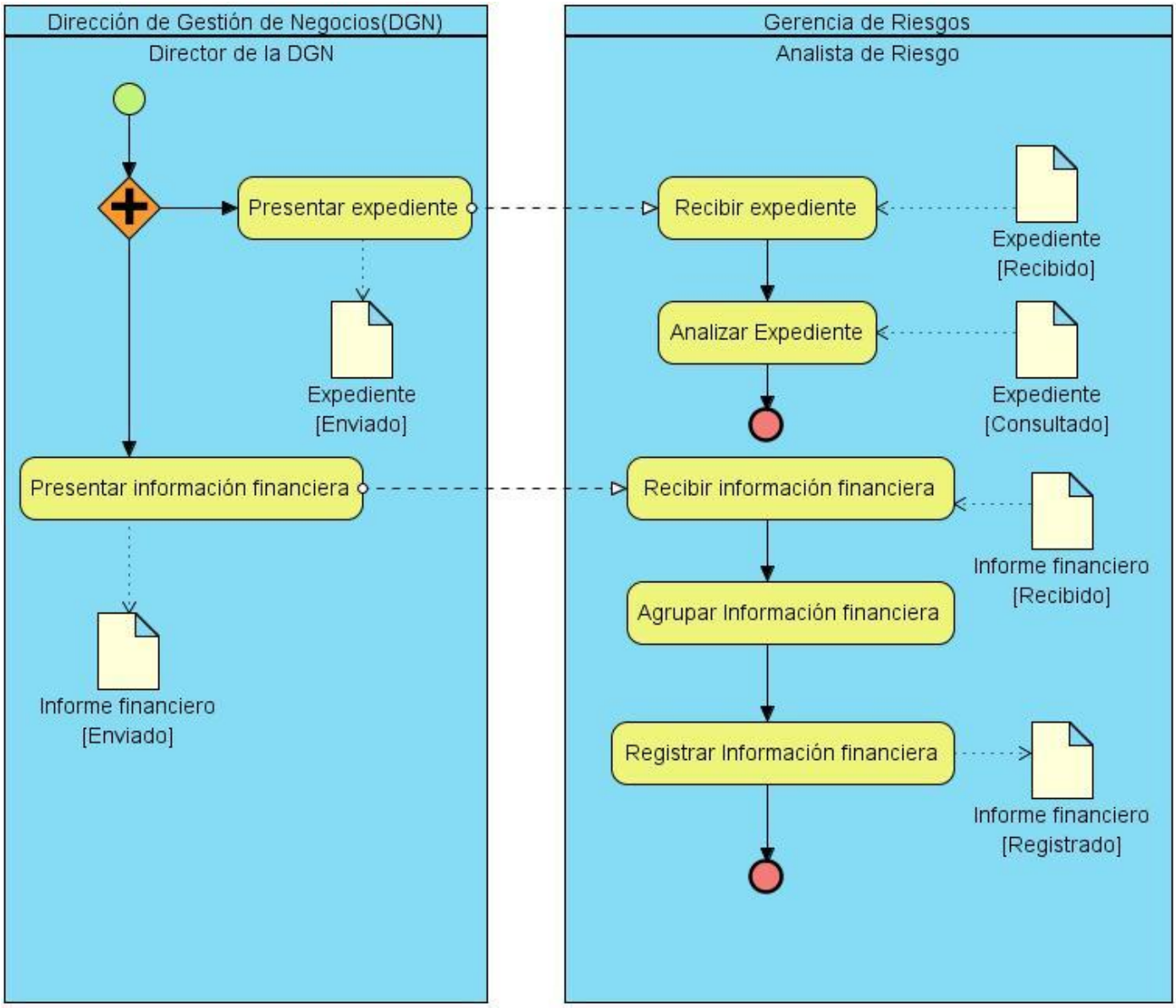


Figura 4: DPN Recibir y registrar la solicitud de análisis de riesgos.

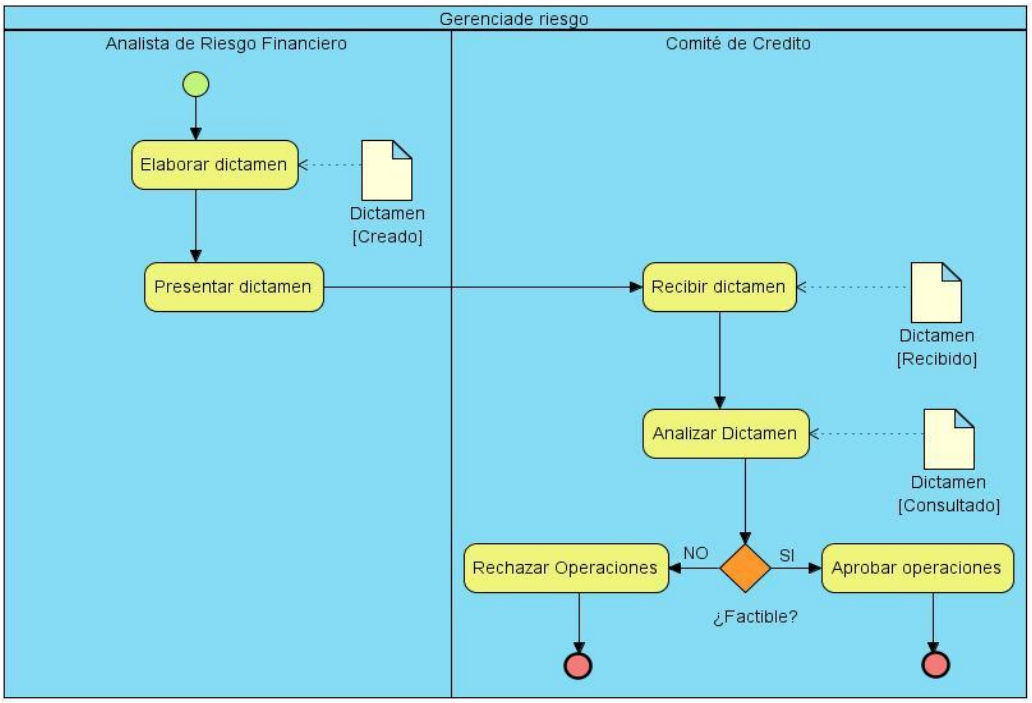


Figura 5: DPN Analizar y dictaminar la situación económico-financiera del cliente y su garante.

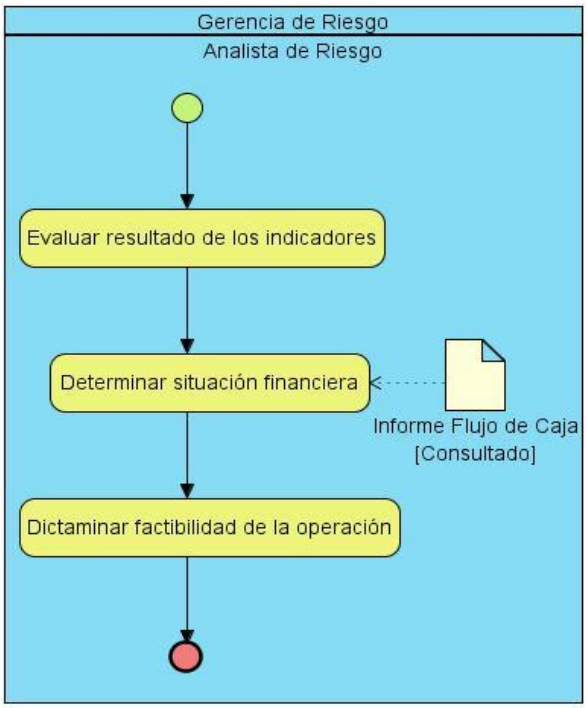


Figura 6: DPN Determinar la factibilidad de la operación.

GLOSARIO DE TÉRMINOS

SAGEB: Sistema Automatizado para la Gestión Bancaria

Factibilidad: La factibilidad se refiere a que el proyecto q tienes en mente puede llevarse a cabo. La factibilidad económica surge de analizar si los recursos económicos y financieros necesarios para desarrollar las actividades pueden ser cubiertos con el capital del que dispones.

Volatilidad: La volatilidad es una medida de la frecuencia e intensidad de los cambios del precio de un activo o de un tipo definida como la desviación estándar de dicho cambio en un horizonte temporal específico. Se usa con frecuencia para cuantificar el riesgo del instrumento.

Acreditado: Aquella persona física o jurídica legítimamente facultada para exigir el pago o cumplimiento de una obligación contraída por dos partes con anterioridad.

Contraparte: Término con el que se designa a la parte contraria en una operación de compraventa. Si somos los compradores, la contraparte es el vendedor, y si vendemos, habrá una contraparte que compra. Un inversor desarrollará sus movimientos siempre en ambas facetas, es decir, comprará y venderá, teniendo en cada momento sus contrapartes vendedoras y compradores. A su vez, para esa otra parte, nosotros representamos la contraparte en la operación.

Dictamen Final: Documento elaborado por el Analista de Riesgo, constituye la valoración final de este sobre el proceso de Análisis de Riesgo de Crédito.

Framework: Marco de trabajo.

Unified Modeling Language(UML): Lenguaje Unificado de Modelado.

Business Process Diagram(BPD): Diagrama del Proceso del Negocio.

Business Process Managment Notation (BPMN): Notación para la Gestión de Procesos del Negocio.