ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN TECNOLÓGICA

SISTEMA DE MANEJO, CONTROL DE CUENTAS CLIENTE Y TRANSACCIONES PARA LA COOPERATIVA DE AHORRO Y CRÉDITO SAN ANTONIO DE PICHINCHA LTDA.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ANÁLISIS DE SISTEMAS INFORMÁTICOS

ANGÉLICA MARIA ARTEAGA HUERA
DIANA ROCIO IZA SANIPATIN

DIRECTOR: Ing. Maritzol Tenemaza MSc.

Quito, Enero 2007

DECLARACIÓN

Nosotros, Angélica María Arteaga Huera y Diana Rocío Iza Sanipatín, declaramos que el trabajo aquí descrito es de nuestra autoría: que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de propiedad Intelectual. Por su Reglamento y por la normativa institucional vigente.

ANGELICA MARIA ARTEAGA

DIANA ROCIO IZA

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Angélica María Arteaga Huera y Diana del Rocío Iza Sanipatín, bajo mi supervisión

Ing. MARITZOL TENEMAZA MSc.

RESUMEN

La Cooperativa de Ahorro y Crédito se encuentra ubicada en San Antonio de Pichincha. La meta fundamental de está empresa es la buena atención a sus clientes, sin embargo este proceso actualmente es manual, la necesidad de la automatización del manejo de datos es imperativa.

Este proyecto presenta una solución para la Cooperativa, pues, la atención a los clientes es sistematizada.

El primer capitulo, resuelve aspectos generales del proyecto, como aspectos metodológicos y ámbitos del proyecto.

Luego se fundamenta con aspectos teóricos importantes en el proyecto, tales como, el paradigma, la metodología entre otros.

Por último se presenta conclusiones y recomendaciones.

Como anexos se presenta el Manual de Usuario, el Manual Técnico y el Manual de Instalación, Manual de Pruebas y Glosario.

Su construcción se llevo acabo con las siguientes herramientas como: Rational Rose 2000, SQL Server 2000 y Visual Basic 6.0 soportando una Arquitectura Cliente / Servidor.

CONTENIDO

DECLARACIÓN _____ *2*

CERTIFICACIÓN _____ *3*

Figura 1.- Gráfico paradigma espiral orientado a objetos…….9 _____ **6**

I. ASPECTOS GENERALES _____ *7*

1.1 ÁMBITO _____ **7**

1.2 PLANTEAMIENTO DEL PROBLEMA _____ **8**

1.3 FORMULACIÓN Y SISTEMATIZACIÓN DEL PROBLEMA _____ **8**

1.3.1 FORMULACIÓN _____ 8

1.3.2 SISTEMATIZACIÓN _____ 8

1.4 OBJETIVOS DE LA INVESTIGACIÓN _____ **9**

1.4.1 OBJETIVO GENERAL _____ 9

1.4.2 OBJETIVOS ESPECIFICOS _____ 9

1.5 ALCANCE _____ **9**

1.6 JUSTIFICACIÓN DEL PROYECTO _____ **10**

1.6.1 JUSTIFICACIÓN PRÁCTICA _____ 10

1.7 ASPECTOS METOLÓGICOS _____ **11**

II. MARCO TEÓRICO _____ *12*

2.1 PARADIGMA ESPIRAL ORIENTADO A OBJETOS _____ **12**

Figura 1.- Gráfico paradigma espiral orientado a objetos. _____ **14**

2.2 METODOLOGÍA OMT _____ **14**

2.2.1 INTRODUCIÓN _____ 14

2.2.2 FASES DE LA METODOLOGÍA OMT _____ 15

2.3 LENGUAJE UNIFICADO DE MODELADO (UML) _____ **19**

2.3.1 OBJETIVOS DEL UML _____ 20

2.3.2 ARQUITECTURA DEL UML _____ 20

Estado _____ **37**

Eventos _____ **37**

Envío de mensajes _____ **38**

2.3.6 CONSTRUCCIÓN _____ 39

III. CONCLUSIONES Y RECOMENDACIONES _____ *45*

3.1 CONCLUSIONES _____ **45**

3.2 RECOMENDACIONES _____ **46**

3.3 BIBLIOGRAFÍA _____ **47**

INDICE DE FIGURAS

Figura 1.- Gráfico paradigma espiral orientado a objetos……9
Figura 2.- Representación de actor……17
Figura 3.- Representación caso de uso……18
Figura 4.- Representa la inclusión en los Diagramas de Caso de Uso……18
Figura 5.- Representa de la extensión en los Diagramas de Caso de Uso ……18
Figura 6.- Generalización en los Diagramas de Caso de Uso……...19
Figura 7.- Representa asociación……..19
Figura 8.- Representa una carpeta en los Diagramas de Caso de Uso……....19
Figura 9.- Gráfico de los elementos que componen una clase…… 20
Figura 10.- Gráfico de los elementos que componen un objeto……..23
Figura 11.- Representa el objeto en el Diagrama de Secuencia……25
Figura 12.- Imagen que representa el mensaje de un objeto a otro……...26
Figura 13.- Mensaje en el Diagrama de Secuencia……....26
Figura 14.- Muestra la figura de mensaje de regreso……...26
Figura 15.- Modelo del gráfico de llegada de mensajes……....26
Figura 16.- Representa el objeto en el Diagrama de Colaboración……27
Figura 17.- Muestra la figura de asociación……28
Figura 18.- Representa el flujo de mensajes Diagrama de Colaboración……..28
Figura 19.- Gráfico de estado inicial del Diagrama de Actividades……....29
Figura 20.- Muestra la figura que representa actividad……....29
Figura 21.- Gráfico de transición del Diagrama de Actividades……....29
Figura 22.- Modelo del gráfico ramificación en el Diagrama de Actividades.....30
Figura 23.- Muestra el gráfico que representa unión en el Diagrama de Actividades…… 30
Figura 24.- Gráfico que representa ramificar una transición en el Diagrama de Actividades…….. 30
Figura 25.- Gráfico de estado final del Diagrama de Actividades……31
Figura 26.- Muestra los canales que se utilizan en el Diagrama de Actividades……..31
Figura 27.- Representa el gráfico de estado en el Diagrama de Estados…….32
Figura 28.- Representa la figura de envío de mensajes……33

I. ASPECTOS GENERALES

1.1 ÁMBITO

La Cooperativa de Ahorro y Crédito se encuentra ubicada en San Antonio de Pichincha por lo cual lleva su nombre.

La Cooperativa se encuentra organizada de la siguiente manera:

Un comité ejecutivo comprendido por: Presidente, Vicepresidente, Secretario.

Representante de cada Área de la Cooperativa: La Parte Legal, Abogados, Auditores.

La parte laboral esta organizada por: Gerencia que incluye Contabilidad, Atención al Cliente, Caja.

La meta fundamental de esta empresa es la buena atención a sus clientes; y ha visto la necesidad de la automatización del manejo de datos.

En la actualidad todo se lo realiza manualmente por lo tanto el tiempo requerido para atender a un cliente es alto, especialmente en el módulo de cajas. En la actualidad el trabajo se lo realiza mediante tarjetas de archivo en donde se van anotando cada uno de los movimientos que realiza el cliente.

La Cooperativa por no contar con un sistema de información adecuado, da lugar al surgimiento de algunos problemas como; la búsqueda de un cliente se vuelve demasiado lenta ya que el registro es buscado en archiveros muy grandes y muchas veces esta información puede ser extraviada; además se dificulta la realización de transacciones ya sean estos depósitos, retiros y transferencias. Sus reportes muchas veces no están bien especificados, no se puede conocer rápidamente y con exactitud la información de cuentas del cliente como: reportes del Estado de sus Cuentas, Anexos de Depósitos, si el interés ha sido acreditado y la obtención de libreta con saldos actualizados.

1.2 PLANTEAMIENTO DEL PROBLEMA

Actualmente los movimientos de transacciones se los realiza de forma manual, no cuentan con un sistema que les brinde confiabilidad y seguridad en la manipulación de datos. Dando lugar a los siguientes problemas:

1. Ineficiencia en la atención al público, ya que los clientes tienen que realizar grandes colas para ser atendidos y causa molestia por lo cual hay motivo de quejas y muchas veces el cliente se retira sin obtener el servicio.
2. Inseguridad en el manejo de los datos e información debido a que se guardan en archiveros y cartillas. Lo que ocasiona perdida de los datos.
3. La búsqueda de un cliente es demasiado lenta, porque deben ser buscados en archiveros demasiado extensos.
4. No se obtienen reportes a tiempo o exactos y corren el riesgo de ser manejados por cualquier persona. Y no se conoce con exactitud el estado de la Cooperativa y de sus clientes.

Tomando en cuenta la delicada situación se plantea desarrollar un sistema informático que apoye en todas las actividades, proporcionando seguridad en los datos con la utilización de un servidor y sus clientes que contengan el hardware y software adecuado para el funcionamiento de la Cooperativa.

1.3 FORMULACIÓN Y SISTEMATIZACIÓN DEL PROBLEMA

1.3.1 FORMULACIÓN

¿Cómo brindar confiabilidad en la manipulación de datos y ofrecer una mejor atención a los clientes de la Cooperativa de Ahorro y Crédito San Antonio de Pichincha?

1.3.2 SISTEMATIZACIÓN

- ¿Cómo minimizar el tiempo de espera para la apertura de cuentas del cliente y que sus datos se encuentren seguros en una base de datos para mayor confidencialidad?

- ¿Cómo se optimizaría la realización de las transacciones en cuanto a depósitos, retiros y transferencias para que estas actividades que son de mayor importancia debido al manejo de dinero sean manipuladas mediante un hardware y software que administre todas las acciones del cliente y que brinden seguridad y confianza dentro de la Cooperativa?
- ¿La Cooperativa necesita estar al tanto de la situación financiera como se conseguiría información a tiempo y que sea confiable?

1.4 OBJETIVOS DE LA INVESTIGACIÓN

1.4.1 OBJETIVO GENERAL

Desarrollar e implantar un Sistema de Manejo, Control de Cuentas Cliente y Transacciones para la Cooperativa de Ahorro y Crédito San Antonio de Pichincha Ltda.

1.4.2 OBJETIVOS ESPECIFICOS

- Desarrollar un módulo de Atención a Clientes para optimizar el tiempo de apertura de las cuentas.
- Desarrollar un módulo de Caja para que los clientes realicen sus depósitos, retiros y transferencias reduciendo el tiempo de espera además que ofrezca mayor seguridad en los datos.
- Presentar reportes a tiempo y tener una diversidad de los mismos.

1.5 ALCANCE

El sistema estará desarrollado a través de la utilización de Microsoft Visual Basic 6.0 y SQL Server 2000 y estará orientado a objetos.

- Crear cuentas de clientes.
- Controlar los clientes activos y sus debidos saldos.
- Tener un control de las cuentas respectivas del cliente en cuanto a las transacciones que realice el mismo.(Estado de Cuenta)
- Administrar los usuarios y empleados de la Cooperativa.
- Generar reportes de acuerdo a las necesidades de los clientes y de la parte administrativa que incluiría también al Gerente.

1.6 JUSTIFICACIÓN DEL PROYECTO

1.6.1 JUSTIFICACIÓN PRÁCTICA

El Sistema de Manejo, Control de Cuentas Cliente y Transacciones le permitirá a La Cooperativa San Antonio de Pichincha optimizar tiempo, mejorar la atención al cliente, garantizar la seguridad de los datos obteniendo beneficios tales como:

- Atención rápida al cliente.
- Brindar confiabilidad y seguridad en la información con la utilización de una base de datos.
- Controlar una constante actualización de la información.
- Otorgar mediante el sistema reportes para el área administrativa y a sus clientes.

1.7 ASPECTOS METOLÓGICOS

<table>
<tr><td>PARADIGMA ESPIRAL ORIENTADO A OBJETOS</td><td>METODOLOGIA OMT</td><td colspan="2">DIAGRAMAS UML</td></tr>
<tr><td colspan="2" rowspan="3">ESPECIFICACION DE REQUERIMIENTOS</td><td colspan="2">Descripción de Actores</td></tr>
<tr><td colspan="2">Diagrama de Casos de Uso</td></tr>
<tr><td colspan="2">Diccionario de Casos de Uso</td></tr>
<tr><td rowspan="5">ANÁLISIS Y DISEÑO</td><td rowspan="2">MODELO ESTÁTICO</td><td colspan="2">Diagrama de Clases</td></tr>
<tr><td colspan="2">Diccionario de Clases</td></tr>
<tr><td rowspan="2">MODELO DINÁMICO</td><td colspan="2">Diagrama de Objetos</td></tr>
<tr><td rowspan="2">Diagramas de Interacción</td><td>Diagrama de Secuencia</td></tr>
<tr><td>Diagrama de Colaboración</td></tr>
<tr><td rowspan="2">MODELO FUNCIONAL</td><td colspan="2">Diagrama de Actividades</td></tr>
<tr><td colspan="2">Diagrama de Estados</td></tr>
<tr><td>CONSTRUCCIÓN</td><td colspan="3">Rational Rose 2000
SQL Server 2000
Visual Basic 6.0
Cliente Servidor</td></tr>
<tr><td>PRUEBAS</td><td colspan="3">Prueba Funcional</td></tr>
<tr><td>MANTENIMIENTO</td><td colspan="3">A posterior, basados en la documentación técnica se podrá dar un correcto mantenimiento.</td></tr>
</table>

II. MARCO TEÓRICO[1]

2.1 PARADIGMA ESPIRAL ORIENTADO A OBJETOS

Este modelo, definido por Boehm en 1988, presenta un desarrollo evolutivo. También introduce como elemento distintivo la actividad de "análisis de riesgo" para guiar la evolución del proceso de desarrollo.

El modelo en espiral combina las principales ventajas del modelo de ciclo de vida en cascada y del modelo de construcción de prototipos. Proporciona un modelo evolutivo para el desarrollo de sistemas de software complejos, mucho más realista que el ciclo de vida clásico, y permite la utilización de prototipos en cualquier etapa de la evolución del proyecto.

Otra característica de este modelo es que incorpora en el ciclo de vida el análisis de riesgos. Los prototipos se utilizan como mecanismo de reducción del riesgo, permitiendo finalizar el proyecto antes de haberse embarcado en el desarrollo del producto final, si el riesgo es demasiado grande.

El modelo se divide en un número de actividades estructurales llamadas regiones de tareas que pueden ser de tres a seis. Pressman define las siguientes regiones de tareas:

- Comunicación con el usuario.- Se define la influencia del problema y se identifican las clases básicas.
- Planificación.-Consiste en determinar los objetivos del proyecto, las posibles alternativas y las restricciones. Esta fase equivale a la de recolección de requisitos del ciclo de vida clásico e incluye además la planificación de las actividades a realizar en cada iteración.
- Análisis de riesgos.- El desarrollo de cualquier proyecto complejo lleva implícito una serie de riesgos: unos relativos al propio proyecto (los riesgos que pueden hacer que el proyecto fracase) y otros relativos a las decisiones que deben tomarse durante su desarrollo (los riesgos asociados a que una de estas decisiones sea errónea).

[1] INGENIERIA DE SOFTWARE, Presuman Roger, 5ta. Edición, Pág. 344,345

El análisis de riesgos consiste en cuatro actividades principales:
Identificar los riesgos. Pueden ser:
Riesgos del proyecto (presupuestarios, de organización, del cliente. etc.)
Riesgos técnicos (problemas de diseño, codificación, mantenimiento)
Riesgos del negocio (riesgos de mercado: que se adelante la competencia o que el producto no se venda bien).

- Ingeniería, Construcción y terminación:
  Ingeniería: Consiste en el desarrollo del sistema o de un prototipo del mismo.
  Construcción y adaptación: Construye, prueba, instala y proporciona soporte al usuario.
- Evaluación del cliente.-Consiste en la valoración, por parte del cliente, de los resultados de la ingeniería.

  En la primera iteración se definen los requisitos del sistema y se realiza la planificación inicial del mismo. A continuación se analizan los riesgos del proyecto, basándonos en los requisitos iniciales y se procede a construir un prototipo del sistema. Entonces el cliente procede a evaluar el prototipo y con sus comentarios, se procede a refinar los requisitos y a reajustar la planificación inicial, volviendo a empezar el ciclo.

  En cada una de las iteraciones se realiza el análisis de riesgos, teniendo en cuenta los requisitos y la reacción del cliente ante el último prototipo. Si los riesgos son demasiado grandes se terminará el proyecto, aunque lo normal es que se siga avanzando a lo largo de la espiral.

  Con cada iteración, se construyen sucesivas versiones del software, cada vez más completas, y aumenta la duración de las operaciones del cuadrante de ingeniería, obteniéndose al final el sistema de ingeniería completo.

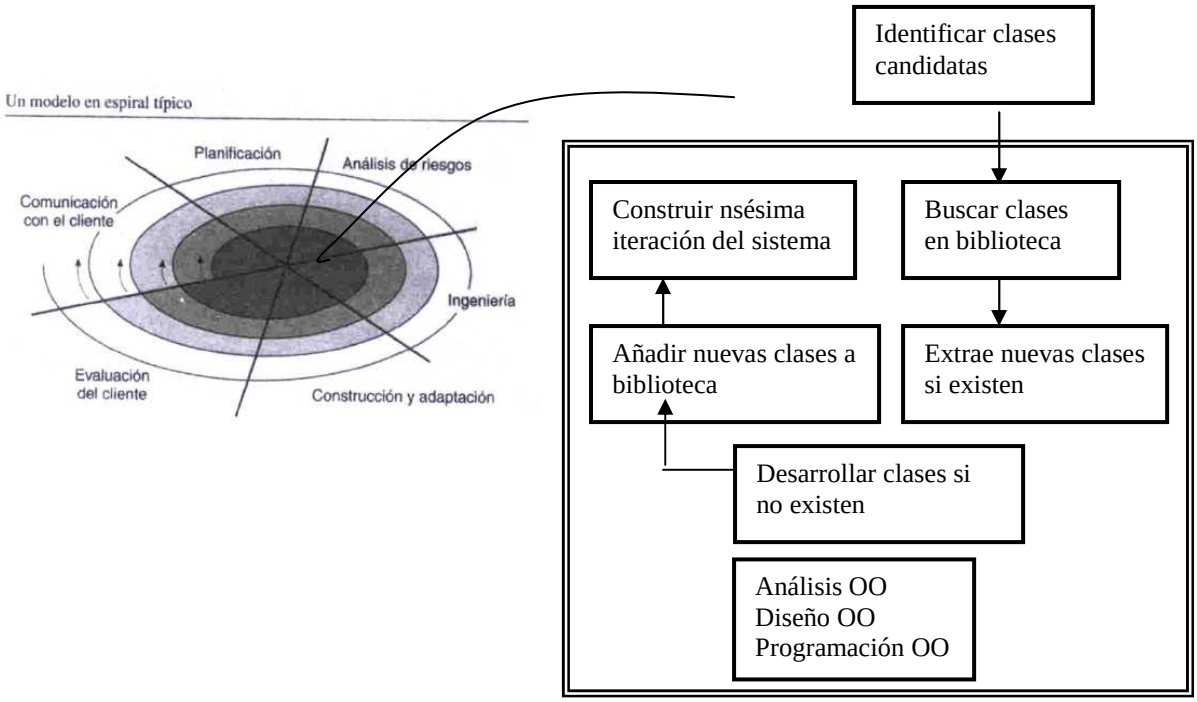

Figura 1.- Gráfico paradigma espiral orientado a objetos.

2.2 METODOLOGÍA OMT [2]

2.2.1 INTRODUCIÓN

La metodología OMT es una técnica de modelado de objetos, desarrollada por James Rumbaugh, que es uno de los precursores del Lenguaje Unificado de Modelado (UML). El significado de las siglas de esta metodología es Técnica de Modelado en Objetos (Object Modeling Technique), la definen como una de las metodologías de la Ingeniería de Software aplicable al desarrollo orientado a objetos en las fases de análisis y diseño.

La fase de análisis comienza con la declaración del problema que incluye, una lista de de objetivos (metas) y conceptos claves definitivos definidos para el dominio del problema a resolver. La declaración del problema se "expande"después en tres modelos:

- Modelo de Objetos
- Modelo Dinámico
- Modelo Funcional

[2] http://ccc.inaoep.mx/~labvision/doo/proy/T32.pdf

2.2.2 FASES DE LA METODOLOGÍA OMT [3]

Divide el ciclo de vida del Software en 4 Fases: análisis de objetos, diseño del sistema, diseño de objetos e implementación

2.2.2.1 Análisis de Objetos

2.2.2.1.1 Descripción del problema

Los requerimientos establecidos durante la conceptualización son revisados y analizados para la construcción del modelo real. La meta del análisis es especificar las necesidades que deben ser satisfechas.

Aquí es donde se determina el modelo del objeto, se hace una tentativa de clases, se eliminan las clases irrelevantes, se definen las posibles asociaciones entre las clases y se hace la refinación de ellas eliminando las redundantes o las que no tienen relevancia, posteriormente se hace una tentativa de atributos de objetos y enlaces.

Una vez determinados los objetos del sistema, se hace la depuración del modelo, posteriormente se busca un nivel de abstracción para modelar subsistemas, para buscar un sistema tangible y sólido.

Luego de tener definido el modelo, se introduce la noción de transacción, que es una forma de modelar procesos o describir cambio de datos y flujo de datos, una vez definido el flujo de datos se define el diccionario de datos de todas las entidades modeladas.

2.2.2.1.2 Modelado de Objetos

El modelo de objetos representa los objetos del sistema. En él se define la estructura de los objetos y clases así como las relaciones que les unen. Comprende tanto un Diagrama de Clases como un Diccionario de Datos que las explique. Este modelo debe ser refinado por medio de la iteración.

---

[3] http://ccc.inaoep.mx/~labvision/doo/proy/T32.pdf , http://www.monografias.com/trabajos6/meto/meto.shtml#intro

- Identificar las clases
- Preparar un diccionario de clases (datos)
- Identificar las asociaciones entre objetos.
- Identificar atributos
- Organizar y simplificar clases usando herencia.
- Verificar que existan trayectorias en el modelo de objetos para preguntas probables
- Iterar y refinar el modelo
- Agrupar las clases en módulos

2.2.2.1.3 Modelo Dinámico

El modelo dinámico representa la interacción entre esos objetos representados como eventos, estados y transiciones.

Los pasos que se siguen en el modelo dinámico son:

- Preparar escenarios de una interacción típica.
- Identificar eventos entre objetos preparando un trazado de eventos para cada escenario.
- Construir los diagramas de estados.
- Revisar los eventos entre los objetos para verificar su consistencia.

2.2.2.1.4 Modelo Funcional

El modelo funcional representa los métodos del sistema desde la perspectiva de flujo de datos.

- Especifica el significado de las operaciones o métodos en el modelo de objetos y de las acciones en el modelo dinámico.
- Muestra como se calculan los valores sin importar la secuencia, las decisiones ni la estructura de los objetos.
- Se utilizan diagramas de flujo de datos para mostrar las dependencias funcionales.

2.2.2.2 Diseño del Sistema

El diseño tiene un alto nivel estratégico y decisivo para la resolución de problemas. Los problemas grandes se deben ver desde el punto de vista de análisis y diseño, subdividir el sistema en subsistemas, poder modular los subsistemas de tal manera que cada modulo sea manejable y comprensible.
En esta etapa se deben crear estrategias, formular una arquitectura para el sistema y las políticas que deben guiarla, además del detalle de diseño. Se deben considerar los siguientes aspectos:

• Visualización de la arquitectura
• Elegir un tipo de implementación para control extremo
• Si se usa una base de datos, definir el paradigma a utilizar
• Determinar oportunidades para el re-uso
• Elegir estrategia para interacción de datos
• Elegir una forma de identificar objetos
• Detalle del diseño

Durante el diseño del sistema se debe hacer un “cuadro“de estrategias y decisiones en cuestiones de arquitectura, tener ideas precisas de clases y métodos individuales.
Adicionalmente se puede mejorar el modelo de diseño para mejorar la implementación.
Se deben considerar los siguientes puntos:
• Uso de transformaciones para simplificar y optimizar el modelo de objetos desde el análisis,
• Elaborar un modelo de objeto,
• Elaborar un modelo funcional,
• Evaluar la calidad del diseño del modelo,
• Implementación.

Luego, el diseño se traduce a un lenguaje de programación, es decir, se codifica. Este paso puede considerarse para las etapas de análisis y diseño para elevar el desempeño del sistema.

2.2.2.3 Diseño de Objetos

Su objetivo es refinar el modelo del análisis y proporcionar una base detallada para la implementación tomando en cuenta el ambiente en que se implementará.

Los pasos que se realizan en el diseño de objetos son los siguientes:

- Se refinan las operaciones para el modelo de objetos a partir de los demás modelos:
- Se diseñan algoritmos para implementar las operaciones y las estructuras de datos
- Se optimizan las vías de acceso a los datos.
- Se implementa el control del software completando la aproximación propuesta en el diseño del sistema.
- Se ajusta la estructura de clases incrementando la herencia.
- Se diseña la implementación de las asociaciones.
- Se determina la representación exacta de los atributos de los objetos.
- Se empaquetan las clases y las asociaciones en módulos.

2.2.2.4 Implementación

La implementación del modelo consiste de la notación del código. La información de espacio es la opción del lenguaje de programación que se usa. La base para la implementación es el modelo de diseño. Aquí se especifica la interfase de cada bloque.

2.2.2.5 Pruebas

El modelo de prueba es el último modelo a construir. Describe simplemente el estado de resultados de la prueba. El modelo de requerimientos de nuevo representa una herramienta potente de prueba, al probar cada caso de uso, se verifica que los objetos se comuniquen correctamente en dicho caso de uso.

Es una actividad para determinar si el sistema esta siendo construida correctamente .Tanto la implementación como las pruebas son dos etapas que están involucradas durante el análisis y diseño.

2.3 LENGUAJE UNIFICADO DE MODELADO (UML) [4]

UML es una especificación de notación orientada a objetos. Se basa en las anteriores especificaciones BOOCH, RUMBAUGH y COAD-YOURDON. Divide cada proyecto en un número de diagramas que representan las diferentes vistas del proyecto. Estos diagramas juntos son los que representa la arquitectura del proyecto.

Con UML nos debemos olvidar del protagonismo excesivo que se le da al diagrama de clases, este representa una parte importante del sistema, pero solo representa una vista estática, es decir muestra al sistema parado. Sabemos su estructura pero no sabemos que le sucede a sus diferentes partes cuando el sistema empieza a funcionar. UML introduce nuevos diagramas que representa una visión dinámica del sistema. Es decir, gracias al diseño de la parte dinámica del sistema podemos darnos cuenta en la fase de diseño de problemas de la estructura al propagar errores o de las partes que necesitan ser sincronizadas, así como del estado de cada una de las instancias en cada momento. El diagrama de clases continua siendo muy importante, pero se debe tener en cuenta que su representación es limitada, y que ayuda a diseñar un sistema robusto con partes reutilizables, pero no a solucionar problemas de propagación de mensajes ni de sincronización o recuperación ante estados de error. En resumen, un sistema debe estar bien diseñado, pero también debe funcionar bien.

UML también intenta solucionar el problema de propiedad de código que se da con los desarrolladores, al implementar un lenguaje de modelado común para todos los desarrollos se crea una documentación también común, que cualquier desarrollador con conocimientos de UML será capaz de entender, independientemente del lenguaje utilizado para el desarrollo.

UML es ahora un Standard, no existe otra especificación de diseño orientado a objetos, ya que es el resultado de las tres opciones existentes en el mercado. Su utilización es independiente del lenguaje de programación y de las características de los proyectos, ya que UML ha sido diseñado para modelar cualquier tipo de proyectos, tanto informáticos como de arquitectura, o de cualquier otro ramo.

---

[4] http://usuarios.lycos.es/oopere/uml.htm

UML permite la modificación de todos sus miembros mediante estereotipos y restricciones. Un estereotipo nos permite indicar especificaciones del lenguaje al que se refiere el diagrama de UML. Una restricción identifica un comportamiento forzado de una clase o relación, es decir mediante la restricción estamos forzando el comportamiento que debe tener el objeto al que se le aplica.

2.3.1 OBJETIVOS DEL UML

- Proporcionar a los usuarios un lenguaje de modelado visual expresivo y utilizable para el desarrollo e intercambio de modelos significativos.
- Proporcionar mecanismos de extensión y especialización.
- Ser independiente del proceso de desarrollo y de los lenguajes de programación.
- Proporcionar una base formal para entender el lenguaje de modelado.
- Fomentar el crecimiento del mercado de las herramientas OO.
- Soportar conceptos de desarrollo de alto nivel como pueden ser colaboraciones, frameworks, patterns, y componentes.
- Integrar las mejores prácticas utilizadas hasta el momento.

2.3.2 ARQUITECTURA DEL UML

Arquitectura de cuatro capas, definida a fin de cumplir con la especificación Meta Object Facility del OMG:

- **Meta-metamodelo:** define el lenguaje para especificar metamodelos.
- **Metamodelo:** define el lenguaje para especificar modelos.
- **Modelo:** define el lenguaje para describir un dominio de información.
- **Objetos de usuario:** define un dominio de información específico.

2.3.3 MODELO ESTATICO[5]

2.3.3.1 Descripción de actores

Se puede diferenciar diferentes actores al momento de la construcción de un diagrama de casos de uso los mismos que se detallan a continuación:

- Principales.- personas que usan el sistema.
- Secundarios.- personas que mantienen o administran el sistema.
- Material externo.- dispositivos materiales imprescindibles que forman parte del ámbito de la aplicación y deben ser utilizados.
- Otros sistemas.- sistemas con los que el sistema interactúa.

2.3.3.2 Diagrama de Casos de Uso[6]

Un diagrama de Casos de Uso muestra las distintas operaciones que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

Los diagramas de casos de uso describen las relaciones y las dependencias entre un grupo de casos de uso y los actores participantes en el proceso.

Es importante resaltar que los diagramas de casos de uso no están pensados para representar el diseño y no puede describir los elementos internos de un sistema. Los diagramas de casos de uso sirven para facilitar la comunicación con los futuros usuarios del sistema, y con el cliente, y resultan especialmente útiles para determinar las características necesarias que tendrá el sistema. En otras palabras, los diagramas de casos de uso describen qué es lo que debe hacer el sistema, pero no cómo.

---

[5] http://www.infor.uva.es/~mbarrio/ISO1/Teoria/uml.htm

[6] http://www.cs.ualberta.ca/~pfiguero/soo/uml/casos_uso01.html,

http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html

2.3.3.2.1 Elementos de un Diagrama de Casos de Uso

- **Actor.-** Un actor es una entidad externa (de fuera del sistema) que interacciona con el sistema participando (y normalmente iniciando) en un caso de uso. Los actores pueden ser gente real (por ejemplo, usuarios del sistema), otros ordenadores o eventos externos.

- Los actores no representan a personas físicas o a sistemas, sino su papel. Esto significa que cuando una persona interacciones con el sistema de diferentes maneras (asumiendo diferentes papeles), estará representado por varios actores. Por ejemplo, una persona que proporciona servicios de atención al cliente telefónicamente y realiza pedidos para los clientes estaría representada por un actor “equipo de soporte” y por otro actor “representante de ventas”.

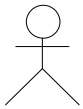

Figura 2.- Representación de actor

- **Caso de Uso.-** Un caso de uso describe, desde el punto de vista de los actores, un grupo de actividades de un sistema que produce un resultado concreto y tangible.

  Los casos de uso son descriptores de las interacciones típicas entre los usuarios de un sistema y ese mismo sistema. Representa la interfaz externa del sistema y especifican qué requisitos de funcionamiento debe tener este (recuerde, únicamente el qué, nunca el cómo).

Cuando se trabaja con casos de uso, es importante tener presentes algunas sencillas reglas:

- Cada caso de uso está relacionado como mínimo con un actor.
- Cada caso de uso es un iniciador (es decir, un actor)
- Cada caso de uso lleva a un resultado relevante (un resultado con "valor intrínseco")

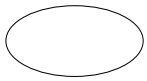

Figura 3.- Representación de caso de uso

Los casos de uso pueden tener relaciones con otros casos de uso. Los tres tipos de relaciones más comunes entre casos de uso son:

- <<include>> que especifica una situación en la que un caso de uso tiene lugar dentro de otro caso de uso.

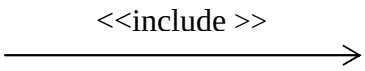

Figura 4.- Representa la inclusión en los Diagramas de Caso de Uso

- <<extends>> que especifica que en ciertas situaciones, o en algún punto (llamado punto de extensión) un caso de uso será extendido por otro.

Figura 5.- Representa la extensión en los Diagramas de Caso de Uso

- Generalización que especifica que un caso de uso hereda las características del "super" caso de uso, y puede volver a especificar

algunas o todas ellas de una forma muy similar a las herencias entre clases.

Figura 6.- Generalización en los Diagramas de Caso de Uso

- Comunicación.- Este elemento representa la relación que existe entre un Caso de Uso y un Actor, dicho elemento es representado simplemente por una línea recta que se extiende de la figura del actor hacia el ovalo del caso de uso.

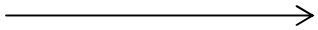

Figura 7.- Representa asociación.

- **Línea de Sistema.-** Empleado para delimitar los límites del sistema, y representado por un rectángulo con color de fondo distintivo.

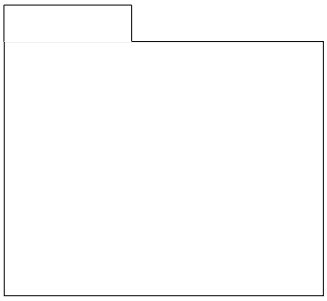

Figura 8.- Representa una carpeta en los diagramas de Caso de Uso.

2.3.3.3 Diccionario de Casos de Uso

Para describir los caso de uso primero se describe que actor lo produce, la interacción del actor –caso de uso, el objetivo del caso de uso, repeticiones de comportamiento, ejecuciones alternativas que presente el caso de uso y cuando se produce y que valor entrega.

2.3.3.4 Diagrama de clases [7]

El Diagrama de Clases es el diagrama principal para el análisis y diseño. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia. La definición de clase incluye definiciones para atributos y operaciones. El modelo de casos de uso aporta información para establecer las clases, objetos, atributos y operaciones.

2.3.3.4.1 Elementos de la clase

Clase.- Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).

En UML, una clase es representada por un rectángulo que posee tres divisiones:

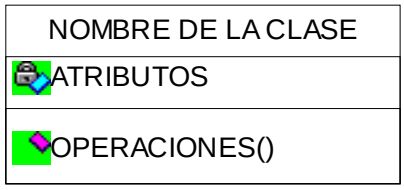

Figura 9.-Gráfico de los elementos que componen una clase

En donde:

Nombre de la clase: Contiene el nombre de la Clase

Atributos: son variables de instancia que caracterizan a la Clase pueden ser:

- **Public (+,):** Indica que el atributo será visible tanto dentro como fuera de la clase, es decir, es accesible desde todos lados.
- **Private (-,):** Indica que el atributo sólo será accesible desde dentro de la clase (sólo sus métodos lo pueden acceder).
- **Protected (#,):** Indica que el atributo no será accesible desde fuera de la clase, pero si podrá ser accesado por métodos de la clase además de las subclases que se deriven.

[7] http://www.creangel.com/uml/clases.php

Operaciones: es la forma como interactúa el objeto con su entorno (dependiendo de la visibilidad: private, protected o public).

2.3.3.4.2 Relación entre Clases

Los enlaces entre objetos pueden representarse entre las respectivas clases y sus formas de relación son:

Asociación: La relación entre clases conocida como Asociación, permite asociar objetos que colaboran entre si. Cabe destacar que no es una relación fuerte, es decir, el tiempo de vida de un objeto no depende del otro.

- Uno y sólo uno
- 0..1 Cero o uno
- M..N Desde M hasta N (enteros naturales)
- Cero o muchos
- 0..* Cero o muchos
- 1..* Uno o muchos (al menos uno)

Agregación:

Para modelar objetos complejos, n bastan los tipos de datos básicos que proveen los lenguajes: enteros, reales y secuencias de caracteres. Cuando se requiere componer objetos que son instancias de clases definidas por el desarrollador de la aplicación, tenemos dos posibilidades:

- **Por Valor:** Es un tipo de relación estática, en donde el tiempo de vida del objeto incluido esta condicionado por el tiempo de vida del que lo incluye. Este tipo de relación es comúnmente llamada Composición (el Objeto base se construye a partir del objeto incluido, es decir, es "parte/todo").
- **Por Referencia:** Es un tipo de relación dinámica, en donde el tiempo de vida del objeto incluido es independiente del que lo incluye. Este tipo de relación es comúnmente llamada Agregación (el objeto base utiliza al incluido para su funcionamiento).

Generalización:

La Generalización consiste en factorizar las propiedades comunes de un conjunto de clases en una clase más general. Los nombres usados: clase padre - clase hija. Otros nombres: superclase - subclase, clase base - clase derivada. Las subclases heredan propiedades de sus clases padre, es decir, atributos y operaciones (y asociaciones) de la clase padre están disponibles en sus clases hijas. La Generalización y Especialización son equivalentes en cuanto al resultado: la jerarquía y herencia establecidas. Generalización y Especialización no son operaciones reflexivas ni simétricas pero sí transitivas. La especialización es una técnica muy eficaz para la extensión y reutilización.

2.3.3.5 Diccionario de clases

El diagrama de clases se completa con un diccionario de datos que contiene la descripción de las clases, atributos y métodos.

El diccionario de clases describe textualmente las clases durante el modelo del dominio del problema.

Este diccionario sirve como un glosario.

El uso del diccionario de datos ayuda a determinar cuales son los elementos de un sistema de software.

Los elementos se deben definir e indicar en que lugar están siendo utilizados.

2.3.3.6 Diagrama de Objetos[8]

Los diagramas de objetos modelan las instancias de elementos contenidos en los diagramas de clases. Un diagrama de objetos muestra un conjunto de objetos y sus relaciones en un momento concreto.

[8] http://www-gris.det.uvigo.es/~avilas/UML/node39.html

En UML, los diagramas de clase se utilizan para visualizar los aspectos estáticos del sistema y los diagramas de interacción se utilizan para ver los aspectos dinámicos del sistema, y constan de instancias de los elementos del diagrama de clases y mensajes enviados entre ellos. En un punto intermedio podemos situar los diagramas de objetos, que contiene un conjunto de instancias de los elementos encontrados en el diagrama de clases, representando sólo la parte estática de un interacción, consistiendo en los objetos que colaborar pero sin ninguno de los mensajes intercambiados entre ellos.

En general los diagramas de objetos se utilizan para modelar estructuras de objetos, lo que implica tomar una instantánea de los objetos de un sistema en un cierto momento. Un diagrama de objetos representa una escena estática dentro de la historia representad a por un diagrama de interacción. Los diagramas de objetos se utilizan para visualizar, especificar, construir y documentar la existencia de ciertas instancias en el sistema, junto a las relaciones entre ellas.

La notación de un objeto es un rectángulo con tres compartimientos. En el primero va el nombre del objeto, en el segundo sus atributos y en el tercero sus operaciones.

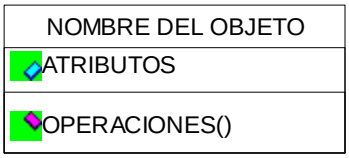

Figura10.- Gráficos de los elementos que componen un objeto

2.3.4 MODELO DINAMICO[9]

2.3.4.1 Diagramas de Interacción[10]

La vista de interacción describe secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema. Un rol clasificador, o simplemente "un rol", es la descripción de un objeto, que desempeña un determinado papel dentro de una interacción, distinto de los otros objetos de la misma clase.

Esta visión proporciona una vista integral del comportamiento del sistema, es decir, muestra el flujo de control a través de muchos objetos. La vista de interacción se exhibe en dos diagramas centrados en distintos aspectos pero complementarios: centrados en los objetos individuales y centrados en objetos cooperantes.

Los objetos interactúan para realizar colectivamente los servicios ofrecidos por las aplicaciones. Los diagramas de interacción muestran cómo se comunican los objetos en una interacción.

Existen dos tipos de diagramas de interacción:

El Diagrama de Colaboración y El Diagrama de Secuencia.

2.3.4.1.1 Diagrama de secuencia[11]

El Diagrama de Secuencia es uno de los diagramas más efectivos para modelar interacción entre objetos en un sistema. Un diagrama de secuencia se modela para cada caso de uso. Mientras que el diagrama de caso de uso permite el modelado de una vista 'business' del escenario, el diagrama de secuencia

[9] http://www.creangel.com/uml/interaccion.php

[10] http://www.creangel.com/uml/interaccion.php

[11] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html#sequence-diagram

contiene detalles de implementación del escenario, incluyendo los objetos y clases que se usan para implementar el escenario, y mensajes pasados entre los objetos.

Típicamente uno examina la descripción de un caso de uso para determinar qué objetos son necesarios para la implementación del escenario. Si tienes modelada la descripción de cada caso de uso como una secuencia de varios pasos, entonces puedes "caminar sobre" esos pasos para descubrir qué objetos son necesarios para que se puedan seguir los pasos.

2.3.4.1.1.1 Elementos de los diagramas de secuencia

- **Línea de vida de un objeto (lifeline):** La línea de vida de un objeto representa la vida del objeto durante la interacción. En un diagrama de secuencia un objeto se representa como una línea vertical punteada con un rectángulo de encabezado y con rectángulos a través de la línea principal que denotan la ejecución de métodos (activación). El rectángulo de encabezado contiene el nombre del objeto y el de su clase, en un formato nombreObjeto: nombreClase.

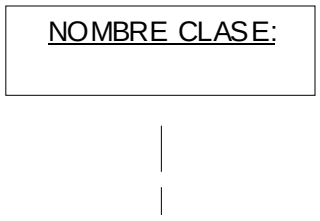

Figura 11.- Representa el objeto en el Diagrama de Secuencia

- **Activación:** Muestra el período de tiempo en el cual el objeto se encuentra desarrollando alguna operación, bien sea por sí mismo o por medio de delegación a alguno de sus atributos. Se denota como un rectángulo delgado sobre la línea de vida del objeto.

Figura 12.- Imagen que representa el mensaje de un objeto a otro

- **Mensaje:** El envío de mensajes entre objetos se denota mediante una línea sólida dirigida, desde el objeto que emite el mensaje hacia el objeto que lo ejecuta.

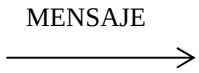

Figura 13.-Mensaje en el Diagrama de Secuencia

También se pueden enviar mensajes que se haga la acción sobre el mismo objeto y se representa por la siguiente simbología

Figura14.- Muestra la figura de mensaje de regreso

- **Tiempos de transición:** En un entorno de objetos concurrentes o de demoras en la recepción de mensajes, es útil agregar nombres a los tiempos de salida y llegada de mensajes.

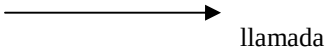

Figura 15.- Modelo del gráfico de llegada de mensajes

2.3.4.1.2 Diagramas de colaboración[12]

Los diagramas de colaboración muestran las interacciones que ocurren entre los objetos que participan en una situación determinada. Esta es más o menos la misma información que la mostrada por los diagramas de secuencia, pero destacando la forma en que las operaciones se producen en el tiempo, mientras que los diagramas de colaboración fijan el interés en las relaciones entre los objetos y su topología.

En los diagramas de colaboración los mensajes enviados de un objeto a otro se representan mediante flechas, mostrando el nombre del mensaje, los parámetros y la secuencia del mensaje. Los diagramas de colaboración están indicados para mostrar una situación o flujo programa específicos y son unos de los mejores tipos de diagramas para demostrar o explicar rápidamente un proceso dentro de la lógica del programa.

2.3.4.1.2.1 Elementos del Diagrama de Colaboración

- **Objeto:** Se representa con un rectángulo que contiene el nombre y la clase del objeto en un formato nombreObjeto: nombreClase.

Figura 16.-Representa el objeto en el Diagrama de Colaboración

- **Enlaces:** Un enlace es una instancia de una asociación en un diagrama de clases. Se representa como una línea continua que une a dos objetos, acompañada por un número que indica el orden dentro de la interacción. Pueden darse varios niveles de subíndices para indicar anidamiento de operaciones. Se pueden utilizar estereotipos para indicar si el objeto que

[12] http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html#sequence-diagram

recibe el mensaje es un atributo, un parámetro de un mensaje anterior, si es un objeto local o global.

Figura17.-Muestra la figura de asociación

- **Flujo de mensajes:** Expresa el envío de un mensaje. Se representa mediante una flecha dirigida cerca de un enlace.

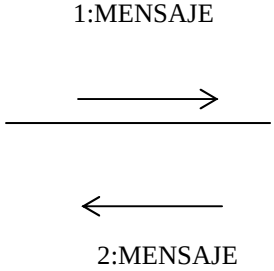

Figura18.-Representa el flujo de mensajes Diagrama de Colaboración

- **Marcadores de creación y destrucción de objetos:** Puede mostrarse en la gráfica qué objetos son creados y destruidos, agregando una restricción con la palabra new o delete respectivamente.
- **Objeto compuesto:** Es una representación alternativa de un objeto y sus atributos. En esta representación se muestran los objetos contenidos dentro del rectángulo que representa al objeto que los contiene.

2.3.5 MODELO FUNCIONAL[13]

2.3.5.1 Diagrama de Actividades[14]

Un diagrama de actividades es un caso especial de un diagrama de estados en el cual casi todos los estados son estados de acción (identifican que acción se ejecuta al estar en él) y casi todas las transiciones son enviadas al terminar la acción ejecutada en el estado anterior. Puede dar detalle a un caso de uso, un

[13] http://www.cs.ualberta.ca/~pfiguero/soo/uml/actividades01.html

[14] http://www.creangel.com/uml/actividades.php# #

objeto o un mensaje en un objeto. Sirven para representar transiciones internas, sin hacer mucho énfasis en transiciones o eventos externos. Se presenta a continuación un ejemplo de diagrama de actividades para un mensaje de un objeto. Generalmente modelan los pasos de un algoritmo.

2.3.5.1.1 Elementos de un Diagrama de Actividades

- **Inicio.-** El inicio de un diagrama de actividad es representado por un círculo de color negro sólido.

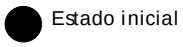

Figura 19.- Gráfico de estado inicial del Diagrama de Actividades.

- **Actividad.-** Una actividad representa la acción que será realizada por el sistema la cual es representada dentro de un ovalo.

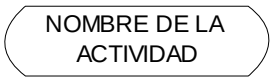

Figura 20.- Muestra la figura que representa actividad.

- **Transición.-** Una transición ocurre cuando se lleva acabo el cambio de una actividad a otra, la transición es representada simplemente por una línea con una flecha en su terminación para indicar dirección.

Figura 21.- Gráfico de transición del Diagrama de Actividades.

- **Ramificación (*Branch*).-** Una ramificación ocurre cuando existe la posibilidad que ocurra más de una transición (resultado) al terminar determinada actividad.

Este elemento es representado a través de un rombo.

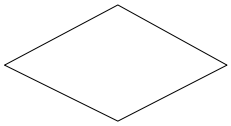

Figura 22.- Modelo del gráfico ramificación en el Diagrama de Actividades.

- **Unión (*Merge*).-** Una unión ocurre al fusionar dos o más transiciones en una sola transición o actividad. Este elemento también es representado a través de un rombo.

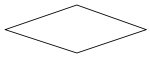

Figura 23.- Muestra el gráfico que representa unión en el Diagrama de Actividades.

- **Expresiones Resguardadas (*Guard Expressions*).-** Una expresión resguardada es utilizada para indicar una descripción explicita acerca de una transición. Este tipo de expresión es representada mediante corchetes y es colocada sobre la línea de transición **[...].**
- ***Fork*.-** Un fork representa una necesidad de ramificar una transición en más de una posibilidad. Aunque similar a una ramificación (*Branch*) la diferencia radica en que un *fork* representa más de una ramificación *obligada*, esto es, la actividad debe proceder por ambos o más caminos, mientras que una ramificación (*Branch*) representa una transición u otra para la actividad (como una condicional). Un *fork* es representado por una línea negra sólida, perpendicular a las líneas de transición.

Figura 24.- Gráfico que representa ramificar una transición en el Diagrama de Actividades.

- **Fin.-** El fin de un diagrama de actividad es representado por un círculo, con otro círculo concéntrico de color negro sólido.

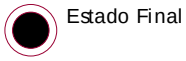

Figura 25.- Gráfico de estado final del Diagrama de Actividades.

- **Canales (*Swimlanes*).-** En determinadas ocasiones ocurre que un diagrama de actividad se expanda a lo largo de más de un entidad o actor, cuando esto ocurre el diagrama de actividad es particionada en canales o (swimlines), donde cada canal representa la entidad o actor que esta llevando acabo la actividad.

Figura 26.- Muestra los canales que se utilizan en el Diagrama de Actividades.

2.3.5.2 Diagrama de Estados[15]

Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, junto con los cambios que permiten pasar de un estado a otro. Los Diagramas de Estado representan autómatas de estados finitos, desde el p.d.v. de los estados y las transiciones. Son útiles sólo para los objetos con un comportamiento significativo. Cada objeto está en un estado en cierto instante. El estado está caracterizado parcialmente por los valores algunos de los atributos del objeto.

---

[15] http://www.creangel.com/uml/estados.php# #

El estado en el que se encuentra un objeto determina su comportamiento. Cada objeto sigue el comportamiento descrito en el Diagrama de Estados asociado a su clase.

Los Diagramas de Estados y escenarios son complementarios, son autómatas jerárquicos que permiten expresar concurrencia, sincronización y jerarquías de objetos, son grafos dirigidos y deterministas. La transición entre estados es instantánea y se debe a la ocurrencia de un evento.

Estado

Identifica un período de tiempo del objeto (no instantáneo) en el cual el objeto está esperando alguna operación, tiene cierto estado característico o puede recibir cierto tipo de estímulos. Se representa mediante un rectángulo con los bordes redondeados.

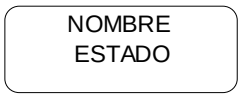

Figura 27.- Representa el gráfico de estado en el Diagrama de Estados.

Eventos

Es una ocurrencia que puede causar la transición de un estado a otro de un objeto. Esta ocurrencia puede ser una de varias cosas:

- Condición que toma el valor de verdadero o falso
- Recepción de una señal de otro objeto en el modelo
- Recepción de un mensaje
- Paso de cierto período de tiempo, después de entrar al estado o de cierta hora y fecha particular

Envío de mensajes

Además de mostrar y transición de estados por medio de eventos, puede representarse el momento en el cual se envían mensajes a otros objetos. Esto se realiza mediante una línea punteada dirigida al diagrama de estados del objeto receptor del mensaje.

Figura 28.- Representa la figura de envío de mensajes.

2.3.6 CONSTRUCCIÓN

2.3.6.1 Rational Rose 2000[16]

El desarrollo en Rational Rose tiene una poderosa combinación de notas, procesos y herramientas para resolver los retos de la actualidad. A través de la industria UML estándar y ahora la construcción y diseño, generación de código y ejecución de modelos. Rational Rose técnicamente desarrolla direcciones para completar el ciclo de vida de un proyecto, desde el análisis en los casos de uso, durante el diseño, implantación y las pruebas.

Los modelos ejecutables permiten compilar y observar simulaciones de los diseños UML. La ejecución de diseños fomenta a un mejor diseño y una continua actualización.

Rational Rose soporta una estrategia de desarrollo iterativo que reduce los riesgos. Técnicamente la generación de código permite una rápida ejecución de un diseño de una máquina enfocada a dicho sistema entonces se consigue un resultado, con lo que se necesita descubrir y corregir los diseños errados y minimizar el riesgo.

Con Rational Rose se específica el diseño en una manera de reflejo en tiempo real, manejando los sistemas que trabajan en la actualidad.

Los bloques de construcción básica son objetos activos llamados cápsulas, los que interactúan unos entre otros limitándose en objetos llamados puertos. La funcionalidad de las cápsulas simples es realizada directamente por un conjunto de máquinas asociadas con la cápsula.

Aún más complejo la combinación de cápsulas dentro de un conjunto de máquinas en red colaborando con subcápsulas.

Un completo soporte en UML en la actualidad construye la fomentación de las más avanzadas soluciones de desarrollo para la industria.

Rational Rose es una herramienta que nos ha facilitado la elaboración de diferentes modelos de UML para el desarrollo de análisis de nuestro sistema.

---

[16] http://www.lynuxworks.com/partners/shows_product.php3?ID=213

2.3.6.2 SQL Server Server 2000[17]

“Es una base de datos moderna con una arquitectura implantada completada por Microsoft, y diseñada para las direcciones más exigentes de aplicaciones de base de datos requeridas por decisiones de apoyo operacionales para sistemas implantados hoy y en el futuro. **SQL Server 7.0** y MSDE 1.0 soporta todos los 32-bit de las plataformas de Microsoft Windows (excepto Win32), códigos base compatibles con adaptaciones dinámicas del hardware capaces de soportar un sistema en el cual ya esté instalado. Esto significa que los desarrolladores pueden escribir un conjunto sencillo de aplicaciones con códigos fuente y mandarlo a las bases de datos de un Windows 98, de un Windows NT **Server** y de un Windows NT **Server** Enterprise Edition. Las filas de Base de Datos también son compatibles con todas las ediciones de **SQL Server 7.0**.
Una versión especial y variante del **SQL Server 7.0**, edición Estándar, es la llamada **SQL Server** Developer's Edition, (edición Promotora), compatible con Microsoft Visual Studio y autorizado aplicaciones desarrolladas que utilizan solamente Visual Studio.

SQL SERVER facilita la creación y la invocación de procedimientos o búsquedas, ya que trabaja con la arquitectura cliente-servidor.

La asignación dinámica de recursos permite la escalabilidad del uso del disco y memoria para acomodarse a las necesidades de la base de datos en cada momento. Esta flexibilidad permite un mejor rendimiento y simplifica la administración del software. La eliminación de dispositivos también es una ventaja complementaria.

[17] (http://www.islasoft.com/Producto/Plataforma %20SQL%20Server.pdf, 1,2).

2.3.6.3 Visual Basic 6.0[18]

Visual Basic 6.0 incluye muchas características nuevas, especialmente en las áreas de Internet y de las bases de datos. Entre ellas se pueden destacar ADO, aplicaciones DHTML y WebClasses, sólo por mencionar las más significativas.

Visual Basic es uno de los tantos lenguajes de programación que podemos encontrar hoy en día. Dicho lenguaje nace del BASIC (Beginner´s All-purpose Symbolic Instruction Code) que fue creado en su versión original en el Dartmouth College, con el propósito de servir a aquellas personas que estaban interesadas en iniciarse en algún lenguaje de programación. Luego de sufrir varias modificaciones, en el año 1978 se estableció el BASIC estándar.

Visual Basic 6.0 es un lenguaje de programación visual, también llamado lenguaje de 4ª generación. Esto quiere decir que un gran número de tareas se realizan sin escribir código, simplemente con operaciones gráficas realizadas con el ratón sobre la pantalla, es simple de usarlo, presenta diseños de diferentes tipos que se los pueden insertar; de esta manera hace que la programación sea fácil.

Proporciona al programador las herramientas suficientes para dar soluciones a niveles empresariales y satisfacer las necesidades de los usuarios más exigentes

[18] McGRAW-HILL/INTERAMERICANA DE ESPAÑA
http://perso.wanadoo.es/tutoriales/cursos/vb/index.htm

2.3.6.4 Seagate Crystal Reports

Esta última versión de Seagate Cristal Reports es una herramienta para generar reportes que trabaja con el programa Visual Basic que la utilizan millones de personas, se puede realizar informes que se los puede personalizar.

2.3.6.4.1 Ventajas:

El reporte forma parte del proyecto, de manera que no se tiene que agregar ningún archivo de reporte al empaquetado.

Al usar el diseñador se tiene la posibilidad de capturar algunos eventos que el reporte dispara, por ejemplo a la hora de leer los registros y a la hora de imprimir el reporte.

Existe una mayor flexibilidad para manejar la jerarquía de objetos y acceder a ella

Se tiene mayor control sobre los objetos que forman parte del reporte.

2.3.6.4.2 Desventajas:

El uso de diseñadores hace que el proyecto crezca en tamaño.

Un paquete de instalación con diseñadores de Crystal Reports es más propenso a fallar (en mi experiencia) que si se lee un reporte desde un archivo externo.

2.3.6.5 Arquitectura Cliente - Servidor[19]

La arquitectura Cliente/Servidor es una forma de dividir y especializar programas y equipos de cómputo de forma que la tarea que cada uno de ellos realiza se efectúa con la mayor eficiencia posible y permita simplificar las actualizaciones y mantenimiento del sistema.

2.3.6.5.1 El Sistema de Información moderno Cliente/Servidor

- Administra y despliega grandes redes
- Ofrece estándares de interoperabilidad
- Distribuye sus funcionalidades

[19] http://www.it.uc3m.es/mcfp/docencia/si/material/1_cli-ser_mcfp.pdf

- Saca partido del modelo Cliente/Servidor
- Muchas veces requiere habilidades híbridas
- Procesamiento de transacciones, bases de datos, comunicaciones o conocimientos sobre GUI.

2.3.6.5.2 El Modelo Cliente/Servidor tiene

Sistema distribuido donde el software está dividido entre

- ✓ tareas servidor
- ✓ tareas cliente

Separación clara de responsabilidades

- ✓ en base a la noción de servicio

2.3.6.5.3 Papel del cliente:

- ✓ inicia el diálogo
- ✓ envía peticiones al servidor conforme a algún protocolo asimétrico
- ✓ pide que el servidor actúe, o que le informe, o ambas cosas

2.3.6.5.4 Papel del servidor:

- ✓ espera pasivamente peticiones de los clientes
- ✓ responde a las peticiones según su política

2.3.6.5.5 Consecuencias

- ✓ Un servidor puede atender a muchos clientes.
- ✓ Puede haber uno o varios servidores en un sistema.
- ✓ Un servidor puede ser substituido por otro que ofrece (al menos) el mismo servicio sin afectar a los clientes.
- ✓ Se puede ocultar a los clientes la ubicación del servidor.
- ✓ La ubicación no afecta la manera de utilizar los servicios
- ✓ El servidor puede regular el acceso a recursos compartidos
- ✓ En el caso general, un objeto/componente/programa puede ser cliente, servidor o ambos

2.3.6.5.6 Ventajas

- ✓ Base en la noción de servicio →buena estructura
- ✓ Acoplamiento cliente-servidor débil, comunicación por mensajes
- ✓ Interfaces claras, modularidad, flexibilidad
- ✓ Escalabilidad “vertical”
- ✓ Facilita: migrar a servidor más grande / veloz o servidores múltiples Escalabilidad “horizontal”
- ✓ Facilita: añadir clientes
- ✓ Hardware y plataformas software (SO) heterogéneos
- ✓ Despliegue independiente de cliente y servidor
- ✓ clientes / servidores pueden usar el hardware y SO más adecuados para su función, ej.cliente barato, servidor rápido
- ✓ Robustez
- ✓ Servidor protegido contra fallos en el cliente

III. CONCLUSIONES Y RECOMENDACIONES

3.1 CONCLUSIONES

- ✓ Las herramientas utilizadas y la aplicación de la Ingeniería de Software permitieron obtener un producto sólido que responde adecuadamente a los requerimientos que la Cooperativa necesita.

- ✓ El desarrollo del sistema para la Cooperativa será de gran utilidad ya que por medio de este se facilitará su desempeño en cuanto a dar mejor servicio al cliente, en el menor tiempo y con eficiencia.

- ✓ La característica fundamental de la Metodología OMT es tener en cuenta todos los principios del paradigma de orientación a objetos y refuerza particularmente la necesidad de modelos consistentes de análisis y diseño del negocio como paso previo y fundamental al desarrollo de la programación y la definición de las bases de datos.

- ✓ En la actualidad, gracias al avance de la tecnología y a las herramientas que se utilizan los sistemas actuales son escalables y la confiables con características importantes para un obtener un producto de calidad.

3.2 RECOMENDACIONES

- ✓ Se recomienda en un futuro incrementar dentro del Sistema de Cuentas y Transacciones de la Cooperativa San Antonio de Pichincha módulos para Inversiones, Créditos y Contabilidad para brindar una mejor atención a las necesidades del cliente.

- ✓ Se recomienda la creación de un portal Web para el facilitar la obtención de la información para los clientes.

- ✓ Se recomienda una adecuada capacitación a las personas que se encargarán de manipular el sistema.

- ✓ Se recomienda incluir en el sistema, el cobro de valores para transacciones como son las transferencias.

- ✓ Se recomienda para la creación de un nuevo empleado realizar un mantenimiento con personas autorizadas para la creación de permisos.

3.3 BIBLIOGRAFÍA

TEXTOS

- ✓ INGENIERIA DE SOFTWARE, Presuman Roger, 5ta. Edición.
- ✓ McGRAW-HILL/INTERAMERICANA DE ESPAÑA

INTERNET

- ✓ http://mitecnologico.com/Main/PrincipiosB%E1sicosDeModeladoDeObjetos
- ✓ http://www.dcc.uchile.cl/~psalinas/uml/modelo.html
- ✓ http://ccc.inaoep.mx/~labvision/doo/proy/T32.pdf
- ✓ http://www.monografias.com/trabajos6/meto/meto.shtml#intro
- ✓ http://usuarios.lycos.es/oopere/uml.htm
- ✓ http://www-gris.det.uvigo.es/~avilas/UML/node8.html
- ✓ http://www.infor.uva.es/~mbarrio/ISO1/Teoria/uml.htm
- ✓ http://www.cs.ualberta.ca/~pfiguero/soo/uml/casos_uso01.html
- ✓ http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html
- ✓ http://www.creangel.com/uml/clases.php
- ✓ http://www-gris.det.uvigo.es/~avilas/UML/node39.html
- ✓ http://www.creangel.com/uml/interaccion.php
- ✓ http://docs.kde.org/stable/es/kdesdk/umbrello/uml-lements.html#sequence-diagram
- ✓ http://www-gris.det.uvigo.es/~avilas/UML/node42.html
- ✓ http://www.cs.ualberta.ca/~pfiguero/soo/uml/actividades01.html
- ✓ http://www.creangel.com/uml/actividades.php# #
- ✓ http://www.creangel.com/uml/estados.php# #
- ✓ http://www.lynuxworks.com/partners/shows_product.php3?ID=213
- ✓ http://www.islasoft.com/Producto/Plataforma %20SQL%20Server.pdf, 1,2

- ✓ http://perso.wanadoo.es/tutoriales/cursos/vb/index.htm
- ✓ http://www.it.uc3m.es/mcfp/docencia/si/material/1_cli-ser_mcfp.pdf

ANEXOS

GLOSARIO DE TÉRMINOS

ANEXO 1: GLOSARIO

Confiabilidad.- la confiabilidad es la medida del número de interrupciones críticas durante el tiempo que un programa o sistema está en funcionamiento.

Automatización.- Es un sistema donde se trasfieren tareas de producción, realizadas habitualmente por operadores humanos a un conjunto de elementos tecnológicos.

Sistema.- Es una parte del universo con una limitada extensión en cuanto al espacio y tiempo. Existen más correlaciones o correlaciones más fuertes entre una parte del sistema y otra que entre esta parte del sistema y partes fuera del sistema.

Programa.- Grupo de instrucciones que sirven para realizar determinadas tareas. También llamadas aplicaciones.

Información.- Es la suma de conceptos y de reglas de actuación que fueron extraídas de una comunicación. Información es un conjunto de datos significativos y pertinentes que describan sucesos o entidades

Seguridad.- Medidas de resguardos contra el acceso no autorizado a los datos. Los programas y datos se pueden asegurar entregando números de identificación y contraseñas a los usuarios autorizados de una computadora.

Cuenta.-Es la que la entidad bancaria autoriza al cliente para disponer, sobre su saldo favorable.

Transacciones.- Las transacciones económicas son transferencias de derechos de propiedad. Cualquier transacción requiere una serie de mecanismos que protejan a los agentes que intervienen de los riesgos relacionados con el intercambio.

Depósitos.- Asiento que señala la tendencia por parte del cliente. Es el movimiento que realiza una persona en una entidad bancaria para que esta guarde su dinero.

Retiros.- Es la acción de sacar algo de un lugar. En una entidad bancaria

Transferencia.- Permite el traslado de valores de una cuenta a otra dependiendo de las necesidades.

Paradigma.- Es una técnica, un modelo o un conjunto de herramientas para representar la solución de problemas específicos.

Metodología.- Conjunto de métodos empleados para el desarrollo de sistemas automatizados.
Una metodología completa es algo más que una notación, un proceso, y herramientas. Además de una "notación, de un proceso, y de herramientas"

Método.- En programación orientada a objetos, método es el procesamiento que realiza un objeto, se implementa el método.

UML.- Se le denomina así al lenguaje Unificado de Modelado, basados en los primeros métodos de Programación Orientada a Objetos (POO) y está pensado para realizar análisis completos para desarrollo de aplicaciones de unas dimensiones amplias.

Prototipo.- Un prototipo es una representación limitada del diseño de un producto que permite a las partes responsables de su creación experimentar su uso, probarlo en situaciones reales y explorar su uso.

Un prototipo puede ser cualquier cosa, desde un trozo de papel con sencillos dibujos a un complejo software.

Actor.- Lenguaje de programación orientado a objetos para computadores personales, se ejecuta sobre Windows y posee una sintaxis parecida a Pascal para facilitar la transición a los lenguajes orientados a objetos.

Clase.- .- Es la unidad básica que encapsula toda la información de un Objeto (un objeto es una instancia de una clase). A través de ella podemos modelar el entorno en estudio (una Casa, un Auto, una Cuenta Corriente, etc.).

Objeto.- Un objeto de datos es una representación de cualquier composición de información compuesta que debe comprender el software.
Un objeto es la instancia de una clase que combina datos y procedimientos. También puede expresarse como una entidad que encapsula sus propiedades, eventos y métodos.

Cliente.- Es el que inicia un requerimiento de servicio. El requerimiento inicial puede convertirse en múltiples requerimientos de trabajo a través de redes LAN o WAN. La ubicación de los datos o de las aplicaciones es totalmente transparente para el cliente.

Servidor.- Es cualquier recurso de cómputo dedicado a responder a los requerimientos del cliente. Los servidores pueden estar conectados a los clientes a través de redes LANs o WANs. Un servidor da servicio a múltiples clientes en forma concurrente.

Cliente/Servidor.- Es la tecnología que proporciona al usuario final el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro recurso del grupo de trabajo y/o, a través de la organización, en múltiples plataformas.

Entorno.-Es aquella parte del universo que está en comunicación con el sistema, pero que no es parte del sistema.