

**UNIVERSIDAD MAYOR DE SAN ANDRÉS**  
**FACULTAD DE CIENCIAS PURAS Y NATURALES**  
**CARRERA DE INFORMATICA**



**PROYECTO DE GRADO**  
**“SISTEMA DE GESTIÓN DE INVENTARIOS”**  
**CASO: KAUTSCH S.R.L.**

**PARA OPTAR AL TÍTULO DE LICENCIATURA EN INFORMÁTICA**  
**MENCION: INGENIERÍA DE SISTEMAS**

**POSTULANTE: René Antonio Espinoza Coharite**

**TUTORA: Lic. Nancy Orihuela Sequeiros**

**REVISOR: Lic. Hugo Javier Reyes Pacheco**

**La Paz    Bolivia**

**2007**

## ÍNDICE

### Página

#### CAPÍTULO 1 MARCO REFERENCIAL

1.1. Introducción. . . . .	11
1. 2. Antecedentes. . . . .	11
1.3. Planteamiento de problema. . . . .	13
1.3.1. Problema principal. . . . .	14
1.3.2. Problema específico. . . . .	15
1.4. Objetivos. . . . .	16
1.4.1. Objetivo principal. . . . .	16
1.4.2. Objetivos específicos. . . . .	16
1.5. Justificación. . . . .	17
1.5.1. Justificación técnica. . . . .	17
1.5.2. Justificación económica. . . . .	17
1.5.3. Justificación social. . . . .	17
1.6. Alcances y aportes. . . . .	18

#### CAPÍTULO 2 MARCO INSTITUCIONAL

2.1. Situación Actual. . . . .	19
2.1.1. Análisis de Involucrados. . . . .	20
2.2. Sistematización de información. . . . .	21
2.2.1. Procesos. . . . .	22

#### CAPÍTULO 3 MARCO TEÓRICO

3.1. Ingeniería de software Orientado a Objetos. . . . .	28
3. 1.1 Planificación y análisis de riegos. . . . .	30
3.1.2 Análisis Orientado a Objetos. . . . .	31
3.1.3 Diseño Orientado a Objetos. . . . .	33
3.1.4 Pruebas Orientadas a Objetos. . . . .	34
3.1.5 Métricas Orientada a objetos. . . . .	36
3.2 Técnicas de modelado a objetos. . . . .	45
3.2.1. Análisis. . . . .	46
3.2.2 Diseño de sistemas. . . . .	48
3.2.3 Diseño de objetos. . . . .	50
3.2.4 Lenguaje de Modelado Unificado. . . . .	51

3.3	Modelo de inventarios. ....	55
3.4	Técnicas de migración de datos. ....	57
3.4.1	Metodología. ....	58
3.5	Calidad del Software. ....	55

## CAPÍTULO 4 MARCO PRÁCTICO

4.1	Análisis. ....	61
4.1.1	Análisis y especificación de requerimientos . ....	62
4.1.2	Modelo Conceptual. ....	64
4.1.3	Modelo Dinámico. ....	67
4.1.4	Diagrama de Casos de Uso. ....	72
4.2	Diseño de sistemas. ....	74
4.2.1	Arquitectura del sistema. ....	75
4.3	Diseño de objetos . ....	76
4.3.1	Diagrama de clases . ....	76
4.3.2	Modelo de Base de Datos Objeto Relacional. ....	77
4.3.3	Diseño Procedimental. ....	79
4.4	Implementación. ....	82
4.5	Migración de datos. ....	88
4.6	Pruebas . ....	89
4.7	Métricas de calidad. ....	90
4.8	Calidad del software. ....	94

## CAPÍTULO 5 MARCO FINAL

5.1	Conclusiones. ....	98
5.2	Recomendaciones. ....	98

Bibliografía.

Anexos.

Documentación.

## ÍNDICE DE FIGURAS

Figura 1. Estructura y organización de la empresa. ....	11
Figura 2. Simbología de la norma ISO 9000. ....	13
Figura 3. Importación de la mercadería. ....	13
Figura 4. Salida o actualización de ventas. ....	13
Figura 5. Elaboración de proformas. ....	14
Figura 6. Facturación. ....	14
Figura 7. Cobro de la mercadería. ....	15
Figura 8. Actualización de ventas en almacenes. ....	15
Figura 9. Modelo de proceso Orientado a Objetos. ....	20
Figura 10. Diagrama de casos de uso. ....	40
Figura 11. Diagrama de clases. ....	41
Figura 12. Diagrama de secuencia. ....	41
Figura 13. Diagrama de estados. ....	42
Figura 14. Notación de los diagramas de flujo de datos. ....	42
Figura 15. Sistema Q. ....	44
Figura 16. Modelo de McCall. ....	48
Figura 17. Planificación de actividades. ....	52
Figura 18. Usuarios registrados para el sistema. ....	54
Figura 19. Modelo conceptual. ....	56
Figura 20. Modelo de clases. ....	58
Figura 21. Validación del usuario. ....	59
Figura 22. Menú principal. ....	59
Figura 23. Ventana Proveedor. ....	60
Figura 24. Ventana de búsqueda. ....	60
Figura 25. Herencia de la Interfaz gráfica de usuario. ....	61
Figura 26. Diagrama de secuencia ingresar mercadería. ....	62
Figura 27. Diagrama de transición de estados de la clase mercadería. ....	63
Figura 28. Diagrama de casos de uso del sistema. ....	63
Figura 29. División del sistema en subsistemas. ....	67
Figura 30. Diagrama de clases detallado. ....	68
Figura 31. Diagrama Entidad Relación. ....	69
Figura 32. Asociación de las clases mercadería y categoría. ....	74
Figura 33. Ventana de inicio del sistema e inicio del nuevo día. ....	75

Figura 34. Ventana de control de mercadería. . . . .	76
Figura 35. Ventana de cambio de precio por archivo. . . . .	76
Figura 36. Ventana de búsqueda en un rango de códigos. . . . .	77
Figura 37. Ventana de reporte de ventas según factura. . . . .	77
Figura 38. Esquema de navegación de la interfaz de usuario. . . . .	78
Figura 39. Archivos Excel y ASCII. . . . .	79
Figura 40 Llave extranjera en un archivo ASCII. . . . .	80
Figura 41. Relaciones de la clase mercadería. . . . .	82
Figura 42. Grafo de flujo del formulario mercadería. . . . .	84



## ÍNDICE DE TABLAS

Tabla 1. Pago de impuestos. ....	12
Tabla 2. Acoplamiento entre objetos. ....	20
Tabla 3. Complejidad ciclomática. ....	28
Tabla 4. Número de hijos. ....	29
Tabla 5. Falta de cohesión. ....	29
Tabla 6. Profundidad de un árbol de herencia. ....	30
Tabla 7. Selección de clases. ....	52
Tabla 8. Actor dirección y/o gerencia. ....	61
Tabla 9. Caso de Uso: Importación de mercadería. ....	61
Tabla 10. Computo de métricas de puntos de función. ....	79
Tabla 11. Factores de evaluación. ....	80





### **DEDICATORIA**

A mis padres, Andrés y Juana, mis hermanos Freddy y Juan Carlos y todas las personas que confiaron en mí.



### **AGRADECIMIENTOS**

A Dios por iluminar mi camino y permitirme llegar hasta donde estoy.

A los docentes Lic. Nancy Orihuela por las observaciones y sugerencias que tuvo en el presente proyecto, Lic. Javier Reyes por su paciencia y colaboración y Lic. Allisón Zuazo por su cooperación.

A todos los docentes que me formaron en mi carrera profesional, en especial a los licenciados Guillermo Choque, Elizabeth García y Lucio Tórrico.

Un agradecimiento especial al Sr. Werner Kautsch, por permitirme desarrollar el presente proyecto.

A mis padres por su apoyo incondicional, mi esposa Silvia e hijas Aylín y Nadine por darme su tiempo mientras duró este proyecto.

A los bibliotecarios Fernando y Daniel por poner siempre su buena voluntad para colaborar y todos mis compañeros que me brindaron su apoyo y amistad a lo largo de toda la carrera universitaria.



## RESÚMEN

Un elemento importante en las empresas que manejan grandes cantidades de datos, son los sistemas de información que ayudan a controlar y automatizar procesos de toda la información que disponen.

En este sentido se propone un sistema de información seguro y confiable para gestionar la información de toda la mercadería de Kautsch S.R.L. empresa dedicada al comercio de artículos de ferretería.

La información de la empresa almacenada en hojas electrónicas, es estructurada y clasificada a un esquema de base de datos y se realiza una migración de datos siguiendo las directrices de La Guía de Migración de Fuentes Abiertas de la Unión Europea, de acuerdo a los procesos de la empresa.

El desarrollo del software esta enmarcado en el paradigma orientado a objetos. La metodología que se utiliza es la Técnica de Modelado a Objetos OMT propuesta por Rumbaugh, en las partes que tiene falencias esta metodología, como ser el modelo funcional y la transición del análisis al diseño se cubre con los principios de la Ingeniería de Software Orientado a Objetos.

Para tener un proceso de análisis y diseño general, se emplea el Lenguaje Unificado de Modelado por ser aceptado por toda la comunidad de desarrollo de software orientado a objetos, dejando de lado la notación de la metodología OMT.

La implementación del sistema se realiza con software libre lenguaje de programación Java, reportes JasperReport , gestor de base de datos MySQL y sistema operativo Windows XP.

## ABSTRACT

An important element in the companies that manage big quantities of data, are the information systems that help to automate and to control processes of everything the information they have.

In this sense this it proposes a sure and reliable system of information to manager the information of everything merchandise of Kautsch S.R.L. dedicated company to the trade of hardware store articles.

The information of the company stored in electronic format, it is structured and classified to a database and it makes a migration of data following the guidelines of The Guide of Migration of Source Open of the European Union, according to the processes of the company.

The development of the software this framed in the oriented objects paradigm. The methodology that is used is the Technique of Modeling to Objects OMT proposed by Rumbaugh, in the parts that have weaknesses this methodology, as being the functional model and the transition from the analysis to the design covers with the principles from the Oriented Objects Software Engineering

To have an analysis process and design general, used the Unified modeling Language to be accepted by everything community of development of software to oriented object, leaving outside the notation of the methodology OMT.

The implementation of the system is realized with software free, programming language Java, reports JasperReport, database manage MySQL and operating system Windows XP.

## **CAPÍTULO 1**

## **MARCO REFERENCIAL**

### **1.1. INTRODUCCIÓN.-**

La tecnología avanza rápidamente hoy en día las empresas no pueden dejar de lado los sistemas de información, en especial aquellas que manejan gran cantidad de datos, porque crece la complejidad al controlar y hacer un seguimiento de toda esa información.

El presente el proyecto de grado, consiste en desarrollar un sistema de información, que permitirá llevar un control y hacer un seguimiento de la información de la mercadería de Kautsch S.R.L. empresa dedicada a la importación y venta de artículos de ferretería.

Toda la información de la empresa esta almacenada en hojas electrónicas, esta información se migrará a una base de datos que estará estructurada y clasificada de acuerdo al tipo de mercadería y el proveedor, se tendrá la posibilidad de actualizar las ventas y los precios por proveedor.

La metodología a utilizar es la Técnica de Modelado a Objetos (Object Modeling Tenique, OMT) propuesta por James Rumbaugh, que estará apoyada sobre los principios de la ingeniería de software orientado a objetos. OMT tiene su propia notación para representar el modelo de análisis y diseño, sin embargo el modelado del sistema, hace uso del Lenguaje Unificado de Modelado (Unified Modelling Language, UML), ya que UML es aceptado por toda la comunidad de desarrollo de software orientado a objetos.

En la implementación del producto software se ha determinado trabajar con software libre, tanto como en las herramientas de desarrollo, el motor de base de datos y el lenguaje de programación, porque presenta todo un conjunto de desarrollo completo, actualizaciones constantes, abundante documentación y licencias libres.

### **1.2. ANTECEDENTES.-**

En la actualidad el desarrollo de sistemas de software, utilizando la metodología Técnica de Modelado a Objetos OMT, esta siendo remplazado por el Proceso Unificado del cual también es autor James Rumbaugh junto Ivar Jacobson y Graddy

Booch. Aun así la metodología OMT sigue siendo, una de las más difundidas en los desarrollos de sistemas de información actualmente [PIA-2000] y se apoya en el análisis y diseño estructurado que constituyen la base para el desarrollo orientado a objeto, esta la razón porque OMT esta dirigida por los datos.

La metodología se hace cargo de todo el ciclo de vida del software, se divide en cuatro fases consecutivas, se centra mucho en un buen análisis y la parte de diseño no es muy compleja, la parte de la implementación y las pruebas las deja al programador.

OMT es una metodología bastante sólida (si exceptuamos su modelo Funcional, bastante criticado y la parte de diseño que es incompleta), completado con otras técnicas de representación como tarjetas de colaboración responsabilidades clase CRC, casos de uso y el lenguaje de modelado unificado UML que es un estándar de facto para modelar el desarrollo de software orientado a objetos [PIA-2000].

Kautsch S.R.L. es una empresa dedicada a la importación y venta de artículos de ferretería de primera calidad, destinada a todo lo que es industrias, empresas mineras, construcción, mueblería, fabricas con diferentes rubros y clientela en general.

La mercadería que posee la empresa esta alrededor de los ocho mil artículos, organizados en diferentes categorías.

Los procesos que se llevan a cabo en la empresa son los siguientes:

- a. Ingreso o entrada de la mercadería, consiste en ingresar la mercadería importada.
- b. Salida o actualización de las ventas, es el registro de las ventas y la actualización de las existencias.
- c. Realización de proformas o cotizaciones, se realiza a clientes mayoristas o del interior.
- d. Actualización de las ventas en kardex de almacenes, toda la mercadería tiene un kardex (tarjetas de control y seguimiento).
- e. Facturación de la mercadería, consiste en registrar la mercadería en la factura.
- f. Cobranza de la factura, según el monto establecido en la factura, se la realiza en bolivianos.

Los procesos de los incisos a, b, y c están automatizados utilizando macros<sup>1</sup> (en el siguiente capítulo se verá el funcionamiento de las macros en Lotus).

En el área internacional existen una gran variedad de sitios Web que ofrecen artículos de ferretería, incluyendo a los proveedores. En este contexto se encontró una tesis sobre sistemas de control de ferretería de la UBP Tesis Sistema de ferretería de Daniel Biolatto, Gabriela Bruschini, Leonardo Tonelli el año 1999 en la Argentina [<http://www.ubp.edu.ar>].

En el área nacional la tesis de la Universidad Católica de Bolivia UCB de Cochabamba bajo el título de Plan operativo de marketing para ferretería Gálvez de Gálvez Padilla Jhony Gustavo el año 2002 [<https://siaa.ucbcba.edu.bo/siaa>]

En la carrera de informática se ha encontrado dos proyectos de manejo de inventarios empleando OMT y UML como el Sistema de Control de Almacén y Facturación (2002 T700) y el sistema de Seguimiento y control de Personal e Inventarios (2002 T806). También se ha encontrado otros proyectos que maneja Inventarios como el Sistema de control de inventarios para Multicon, Sistema de control de inventarios de Osmar importaciones, el Sistema de Almacenes y Producción caso Socovial, el sistema de información para el control y seguimiento de almacén y facturación caso Zystem Solution o el Sistema de información para el control de pedido e inventarios de almacén y farmacias de la Caja Petrolera de salud, en todos estos proyectos consideran la demanda constante, esto generalmente no es verdad, el presente proyecto se diferencia de las anteriores porque realiza una migración de los datos en archivos planos de Lotus a una Base de datos con información organizada y estructurada y de esta se obtiene la demanda promedio para realizar el pedido óptimo y calcular el costo del mismo por periodo planeado.

### **1.3. PLANTEAMIENTO DEL PROBLEMA.-**

Actualmente la ferretería Kautsch S.R.L. utiliza la hoja electrónica Lotus versión 5.0 en alemán para manejar la información de los procesos automatizados (ingreso o importación de la mercadería, salida o actualización de las ventas y elaboración de proformas o cotizaciones) mediante macros elaborados por la dirección (dueño de la

---

<sup>1</sup>Consiste en una serie de comandos y funciones que se ejecuta siempre que sea necesario realizar una tarea repetitiva



empresa) e implementados en todos los archivos (categorías de la mercadería, aproximadamente 30) donde se registra información acerca de la importación y venta de la mercadería. La versión de Lotus en alemán dificulta el manejo de todas las utilidades de esta herramienta de cálculo a los actuales usuarios (gerencia y personal de apoyo) exceptuando a la dirección y aun mas la explotación de las macros, la interfaz de trabajo es similar a Excel de Microsoft que ha tenido una gran acogida en las tareas de calculo lo que hace predecir que el mantenimiento de Lotus será muy difícil.

Para la gerencia y el personal de administración es difícil tomar decisiones con la información que generan los procesos de ingresos y ventas ya que es insuficiente para la toma de decisiones, como cuándo importar y en qué cantidad.

La importación de mercadería se realiza de forma empírica de parte de la dirección y gerencia, ya que no se cuenta con un historial de ventas que apoye a la toma de decisiones en la importación de mercadería, que desde luego no es mala, sin embargo no es la mejor opción, porque se dan casos en que el cliente desea adquirir un artículo determinado y no se tiene a disposición, lo que origina perdida en las ventas, o lo contrario que se tiene mercadería que se ha vendido muy poco o nada por periodos largos(años) según el kardex de algunos artículos. La revisión de la mercadería no se realiza de forma periódica solo cuando director o gerencia planean importar.

Los datos registrados en kardex a menudo<sup>2</sup> son diferentes de la cantidad física del artículo, en especial si son pequeños.

El proceso de salida o actualización de las ventas es realizado por el personal de apoyo registrando las facturas, existiendo la posibilidad de cometer errores de transcripción y verificación, también se halla la posibilidad de encontrar fallas en el registro de datos como el código o cálculo de precio en la facturación de algún artículo, de parte del vendedor.

La actualización de las ventas que lleva almacenes es realizado de forma manual, existiendo la posibilidad de fallas humanas para registrar la salida de la mercadería, lo que produce que kardex de almacenes presenten fallas en las cantidades registradas, incluso presenta casos que kardex no indica la cantidad física disponible, esto por

---

<sup>2</sup> Revisión de una de las marcas mas variadas de artículos de ferretería se tuvo que el 15% de diferencias en las cantidades que indica kardex y la cantidades físicas.

omisión o descuido en la actualización en particular cuando sale una gran cantidad de mercadería.

La consulta de existencia de mercadería, llega a ser lento cuando el cliente solicita gran cantidad de artículos o cuando se realiza consultas simultaneas de vendedores a los almacenes, de esta manera se pierden clientes por no tener un acceso inmediato y confiable a los artículos en existencia.

En resumen se tiene que en los procesos de entrada o ingreso de la mercadería, y actualización de las ventas en kardex, presentan incoherencias en las cantidades que figuran en kardex y los que se tienen registrados en las hojas electrónicas.

**1.3.1 Problema principal.-** El control y seguimiento de la mercadería existente en los procesos de entrada y salida no es fiable<sup>3</sup> porque a menudo la cantidad que indica kardex es diferente al registro que indican los archivos, lo que origina un mal desempeño en las tareas administrativas como la importación de la mercadería o consulta de saldos.

**1.3.2 problemas específicos.-** Los problemas identificados son los siguientes:

- P1. Para la gerencia es difícil tomar decisiones con la poca información que generan los procesos de entrada y salida.
- P2. Inexistencia de reportes para diferentes inquietudes de la administración con interfaces amigables.
- P3. Inexistencia de historial de ventas para realizar los pedidos.
- P5. La cantidad que indica kardex es diferente a la cantidad física existente.
- P6. Fallas que se arrastran en las facturas por el registro incorrecto del código, cantidad y/o precio del personal de ventas.
- P7. Errores de transcripción de las facturas, en la actualización de la cantidad del (de los) artículo(s) en los diferentes archivos de parte del personal de apoyo.
- P8. La consulta de los saldos de la mercadería se realiza en forma lenta cuando existe consultas múltiples de los vendedores a los almacenes.
- P10. Falta de control en el ingreso de mercadería en kardex.

---

<sup>3</sup> Confiable, seguro, digno de confianza.

P13. Descuido u omisión involuntaria para dar baja a la mercadería por parte de los encargados de almacenes.

#### **1.4. OBJETIVOS.-**

##### **1.4.1 Objetivo general.-**

Desarrollar un producto software para el control y gestión de inventarios empleando la metodología Técnicas de Modelado a Objetos OMT y la notación del Lenguaje de Modelado Unificado UML para una administración eficiente y efectiva y así mejorar el rubro al que se dedica la empresa.

##### **1.4.2 Objetivos Específicos.-**

- a) Realizar un relevamiento de la información referente a toda la mercadería existente.
- b) Sistematizar, organizar y clasificar la mercadería de acuerdo a las necesidades de la empresa.
- c) Definir técnicas de migración de archivos planos (Lotus 123 o Excel) a una base de datos.
- d) Aplicar la metodología técnica de modelado a objetos para el análisis y diseño, y representarlo gráficamente con el lenguaje de modelado unificado para tener una documentación completa y universal en el desarrollo del producto software.
- e) Evaluar la calidad del producto software a través de los estándares y atributos de calidad.

#### **1.4. JUSTIFICACIONES.-**

Las justificaciones que se determinaron son: la justificación técnica, justificación económica y la justificación social.

**1.4.1 Técnica.-** El proyecto se justifica técnicamente por que la empresa Kautsch S.R.L. se actualizará con el avance tecnológico en el campo de la informática. Tomando en cuenta de los equipos que dispone. La implementación del sistema se lo realizará en la red local bajo la plataforma de trabajo de Windows XP.



**1.4.2 Económica.-** La empresa Kautsch S.R.L. será beneficiada con el desarrollo de un sistema de información elaborado por un estudiante de la U.M.S.A. para lo cual no se tendrá que recurrir al servicio de consultoras externas que tiene un elevado costo.

La implementación del software se realizará con software libre, lo que reducirá el costo de desarrollo notablemente, al no comprar licencias.

La inversión en cuanto el hardware es nulo ya que se utilizarán los mismos equipos con que cuenta la empresa.

Se tendrá un mejor control de la mercadería que tiene la empresa, de esta manera se realizará la toma de decisiones de forma acertada en cuanto a la administración y gestión de inventarios de la ferretería.

**1.4.2 Justificación social.-** El sistema de información ayudara a realizar un mejor control y seguimiento de la mercadería lo que implica tener un control automatizado de la mercadería. Y así en cierta forma tener una mejor atención al cliente que en algún momento requiere de un artículo de ferretería, que no pudo obtenerlo por ser un artículo agotado, pues el sistema apoyará en las decisiones administrativas de la importación de mercadería.

**1.5. ALCANCES Y APORTES.-** Con este producto software se desea abarcar los procesos de entrada, salida y actualización de la mercadería, actualización de kardex en los almacenes y la elaboración de proformas. Los procesos que van a ser encarados tienen operaciones y consultas sobre los diferentes tipos de artículos.

Por decisión de la empresa no se tomará en cuenta el proceso de facturación.

El proyecto de sistema de gestión de inventarios manejará toda la información que concierne a la importación de la mercadería, actualización de ventas, registro de mercadería nueva, elaboración de precios indicando la ganancia para un determinado artículo y sugiriendo un posible precio tomando en cuenta el costo de importación del artículo, menos los impuestos y el descuento que se da en determinadas ocasiones. También se implementara un historial de importación y un historial de ventas, que apoyará la toma de decisiones de cuanto y cuando importar.

Los aportes realizados en este proyecto son el desarrollo de un producto software, haciendo uso de software libre, en el que se pretende mostrar los beneficios del paradigma orientada a objetos, en el área científico se pretende mostrar que se puede desarrollar un software de calidad utilizando la metodología OMT, articulando la notación de UML para el análisis y diseño de un producto software, apoyado en los principios de ingeniería de software orientado a objetos y la propuesta de formular una serie de pasos o métodos para realizar la migración de datos, de archivos planos almacenados en hojas electrónicas (Lotus o Excel) a un esquema de base de datos.



## **CAPÍTULO 2**

## **MARCO INSTITUCIONAL**

### **2.1. SITUACIÓN ACTUAL.-**

Kautsch S.R.L. es una empresa dedicada a la importación y venta de artículos de ferretería de primera calidad que oferta al mercado de la ciudad de La Paz. La mercadería es importada principalmente de Estados Unidos y Europa.

La empresa viene trabajando desde el año 1976 ubicada en la calle Sagarnaga N° 115 – 119 y cuenta con un sucursal desde hace 4 años en la zona sur de la ciudad.

La mercadería de la Central esta destinada a todo lo que es industria, empresas mineras, construcción, mueblería, fabricas de todo tipo y clientela en general.

La sucursal tiene mercadería netamente para el uso diario, mueblería y construcción, destinada a gente con poder adquisitivo más elevado. Todos los precios de la Central y sucursal están en dólares.

Entre las especialidades de la ferretería se encuentra: maquinaria y material para minería e industria, herramientas, chapas, manijas, jaladores, accesorios de jardinería y baño.

La cantidad de artículos que posee la empresa esta alrededor de los 8000, en sus diferentes variedades, clasificada alfabéticamente y agrupados según el tipo en 30 categorías, con excepciones por fabricante como son las líneas Stanley, Proto y Yale. Para identificar la misma, lleva una codificación numérica secuencial, que va desde 100000 hasta el 999999, donde se tiene registrado el historial de importación, costo, proveedor, cantidad de entrada, cambio monetario, fecha de llegada y número de importación.

La empresa no hace distinción del tipo de cliente, porque el “cliente es el rey al cual se lo debe atender según sus requerimientos”<sup>4</sup> aunque lleve algún artículo de muy poco valor o una compra significativa.

Los clientes tienen descuentos según el monto de compra o si el cliente es mayorista y/o revendedor, los descuentos se dan según el tipo de artículo que compran, porque no en todos se obtiene el mismo porcentaje de utilidad.

---

<sup>4</sup> Lema de la empresa

La empresa en los inicios de los años noventa se ha visto en la necesidad de incorporar tecnología informática en poca dimensión (una maquina modelo Sony), con el propósito de actualizarse con la tecnología de la computación que se veía avanzar día a día.

Actualmente esta pequeña tecnología ha crecido ahora comprende de seis máquinas Pentium IV, que están distribuidas de la siguiente manera: tres en la oficina de la dirección y gerencia, dos en el despacho de la oficina y una en caja.

La plataforma de trabajo es Windows XP, la herramienta principal que se utiliza para guardar y administrar la información de entrada, salida e inventario de la mercadería es la hoja de calculo Lotus 123 versión 5.0 (en alemán). Las hojas electrónicas emplean macros para las tareas repetitivas como la búsqueda de artículos, la actualización de las existencias o el ingreso de mercadería. También se utiliza Word y Excel de Microsoft Office para la elaboración de cartas, proformas y otros usos de texto, se recurre a Excel para leer los catálogos de los proveedores y la elaboración de planillas.

**2.1.1 Análisis de Involucrados.-** El personal de la empresa esta compuesto por dirección, gerencia, personal de apoyo, encargados de almacenes y vendedores. La estructura del personal se aprecia en la figura 1.

La contabilidad es realizada por personal externo, la gerencia y el personal de apoyo hacen seguimiento de la mercadería.

La Dirección esta encargada de manejar toda la información concerniente a la importación y control de la mercadería y el seguimiento de las ventas.

La gerencia, plantea las necesidades que se tiene para administrar la entidad, maneja información que corresponde a las importaciones, la contabilidad y el pago de sueldos.

Así mismo es el encargado de atender la sucursal.

El personal de apoyo verifica la facturación emitida por los vendedores y actualiza la mercadería existente registrando las ventas. También realiza la tarea de cobranza de facturas (caja).

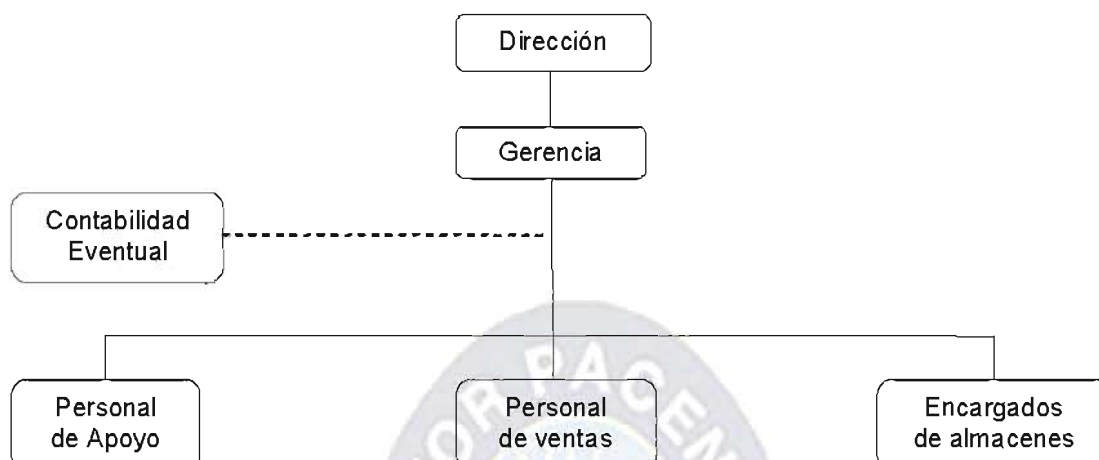


Figura 1: Estructura y organización de la empresa. Fuente: Elaboración propia

Los vendedores se encargan de la facturación, despacho de mercadería y la elaboración de proformas o cotizaciones.

Los encargados de almacenes registran el kardex de los artículos vendidos y organizan toda la mercadería controlando que la misma esta ordenada.

## 2.2. SISTEMATIZACIÓN DE INFORMACIÓN.

De acuerdo con la Ley 843 reformada y disposiciones legales actualizadas, los individuos, sociedades o empresas que tengan cualquier actividad relacionada con el comercio de todo tipo de mercaderías, o de prestación de servicios de cualquier naturaleza, deben tener el número de identificación tributaria (NIT), para pagar tributos al estado boliviano. Es así que Kautsch S.R.L. se encuentra registrado en el régimen general y cumple con las siguientes obligaciones:

- Impuesto al Valor Agregado (IVA), impuesto que grava la compra y venta de bienes en la comercialización, el pago de este impuesto es realizado mensualmente que corresponde a la diferencia entre el crédito fiscal<sup>5</sup> y el debito fiscal<sup>6</sup>. Para realizar esta liquidación Kautsch SRL lleva un libro de compras y ventas.
- Régimen Complementario al Impuesto al Valor Agregado (RC-IVA), impuesto que es pagado mensualmente por Dirección y Gerencia del 13% del sueldo total.
- Impuesto a las Utilidades o Renta de Empresas (IUE), este impuesto corresponde al 25% sobre la utilidad neta que es extraída de los estados financieros, para ello existe un

<sup>5</sup> Debito fiscal es el monto adeudado al estado, resulta de la venta de mercadería (artículos de ferretería).

<sup>6</sup> Crédito fiscal es el IVA que resulta de las compras con factura que sirven para disminuir el crédito fiscal.

personal eventual<sup>7</sup> que lleva los registros contables. El impuesto es pagado anualmente.

- Impuesto a las Transacciones (IT), impuesto que grava el comercio. El IT corresponde al 3% sobre el valor o monto total por el concepto de venta de mercadería, este impuesto se paga mensualmente.

A continuación se presenta un resumen en la tabla 1, incluyendo el número de formulario otorgado por Impuestos Nacionales para realizar el pago de gravámenes.

Impuesto	% de retención	Forma de pago	Nº de formulario
IVA	13 %	Mensual	143
RC IVA	13 %	Mensual	98
IUE	25 %	Anual	80
IT	3 %	Mensual	15

Tabla 1: Pago de impuesto, Fuente Kautsch S.R.L., elaboración propia

En la importación de mercadería también se realiza una tributación al estado boliviano que vienen a ser el gravamen arancelario y los impuestos internos. El gravamen arancelario imponible corresponde al 10% del costo de origen de la mercadería más el costo de transporte y seguro hasta la aduana de ingreso al país. Y los impuestos internos a la importación de mercadería son el Impuesto al Valor Agregado que corresponde al 14.6%.

Kautsch S.R.L. también cumple con las normas de la ley general del trabajo. Todo el personal realiza aportes a las AFP's y están registrados en el seguro social, de la caja nacional de salud.

**2.2.1 Procesos.-** Los procesos más relevantes que se dan en la ferretería Kautsch S.R.L. son los de entrada o importación de la mercadería y salida o actualización de las ventas. También existen otros procesos o subprocesos que se generan alrededor de los mencionados como: La elaboración de proformas o cotizaciones, venta de mercadería registrada en factura, cobro de la mercadería vendida (caja), actualización física de ventas (kardex de almacenes).

Para la descripción de los procesos se utiliza los símbolos de la norma ISO 9000 (ver

---

<sup>7</sup> Personal externo de la empresa con el título de contador general para llevar los registros contables.



Anexo 2), la notación es:

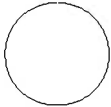


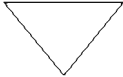
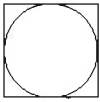
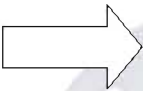

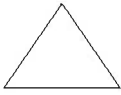
			
Operaciones	Inspección y Medición	Demora	Entrada de ítems
			
Operación e inspección	Transporte	Decisión	Almacenamiento

Figura 2: Simbología de la norma ISO 9000.

- La importación o entrada de mercadería es realizada por la dirección y gerencia, el proceso de cómo se realiza se aprecia en la figura 3:

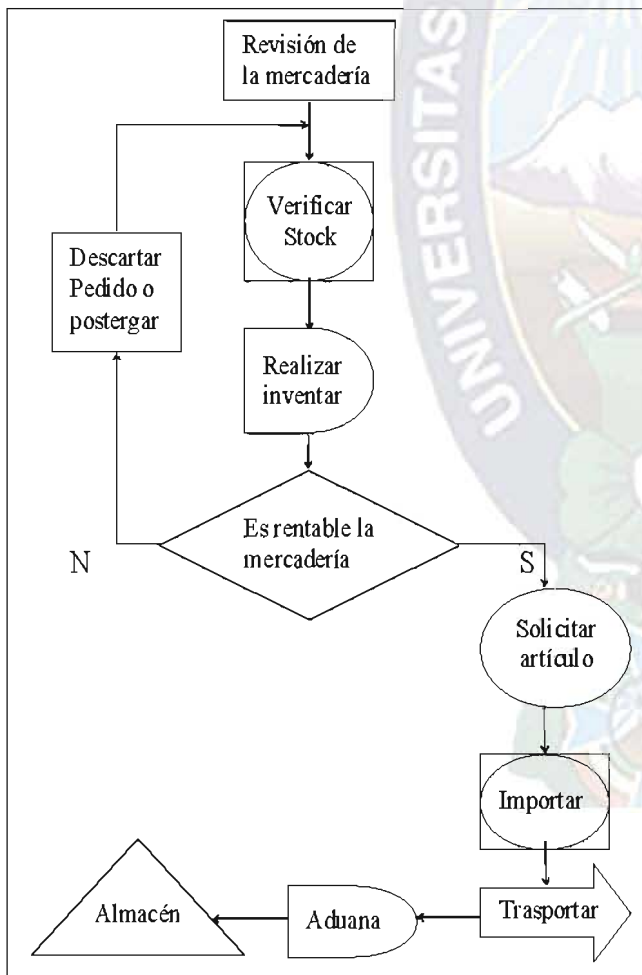


Figura 3: Importación de la mercadería.  
Fuente: Elaboración Propia.

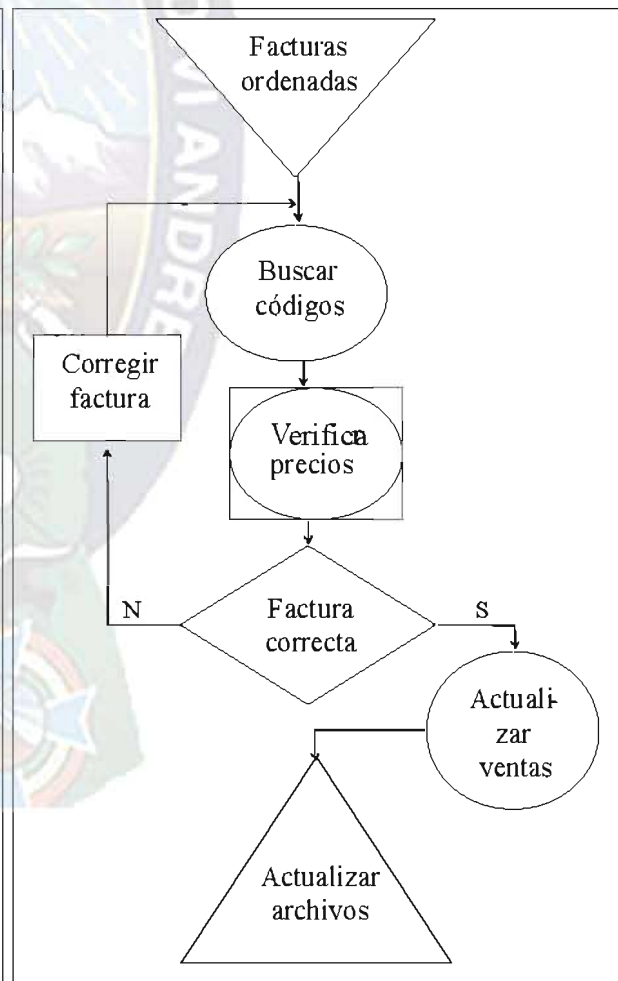


Figura 4: Salida o actualización de las ventas,  
Fuente: Elaboración Propia.

- El proceso de salida o actualización de las ventas es realizado por el personal de apoyo se puede apreciar en la figura 4.
- El proceso de elaboración de proformas se realiza cuando el cliente es mayorista o lleva una considerable cantidad de mercadería y el monto de compra es alto. Este proceso funciona como se muestra en la figura 5.
- El proceso de registrar la venta de mercadería en la factura (facturación) funciona como se muestra en la figura 6.

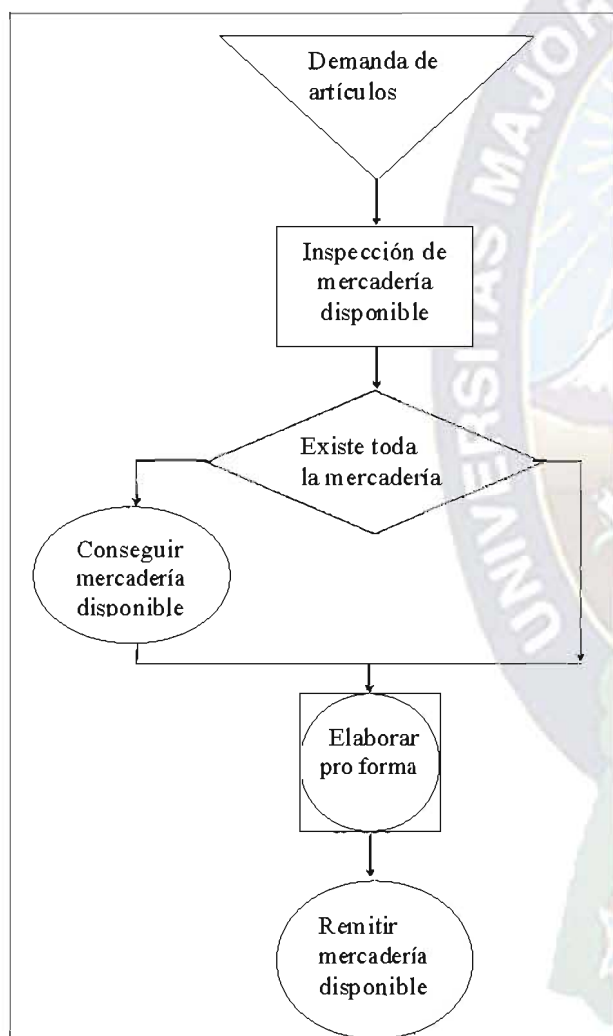


Figura 5: Elaboración de proformas  
Fuente: Elaboración Propia.

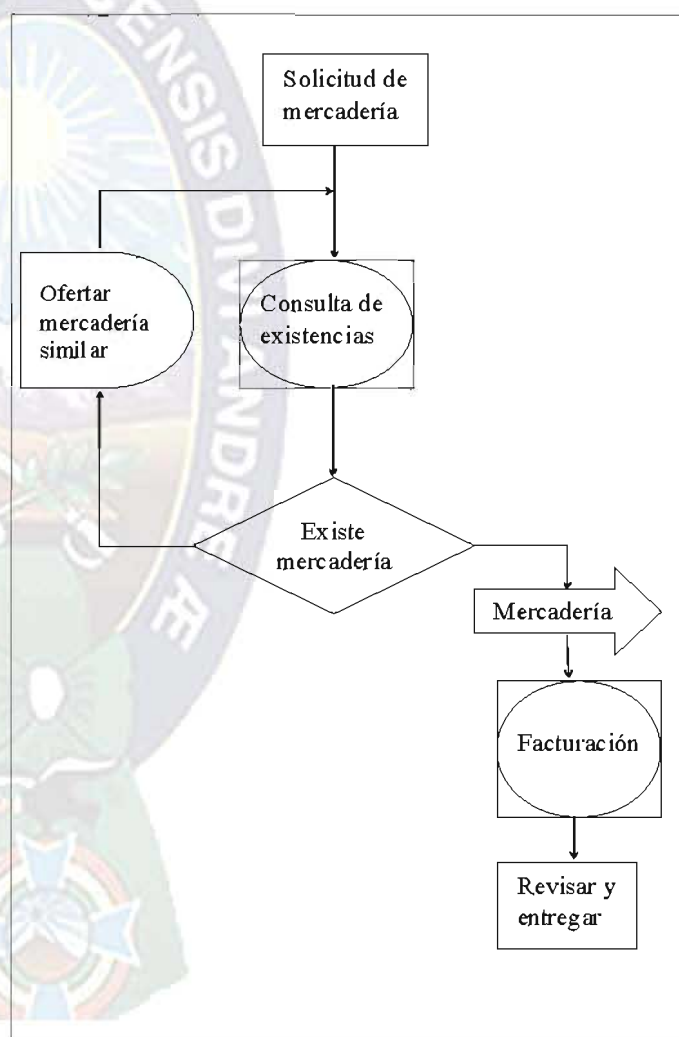


Figura 6: Facturación  
Fuente: Elaboración Propia

- El proceso de cobro de la mercadería vendida es como se muestra en la figura 7.



- El proceso de actualización de las ventas de la mercadería en kardex de almacenes es como se muestra en la figura 8.

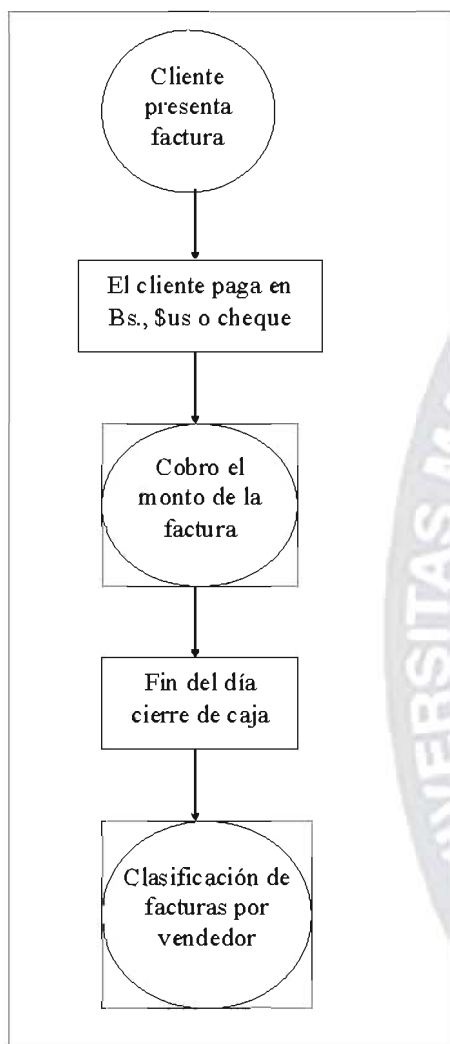


Figura 7: Cobro de la mercadería,  
Fuente: Elaboración Propia.

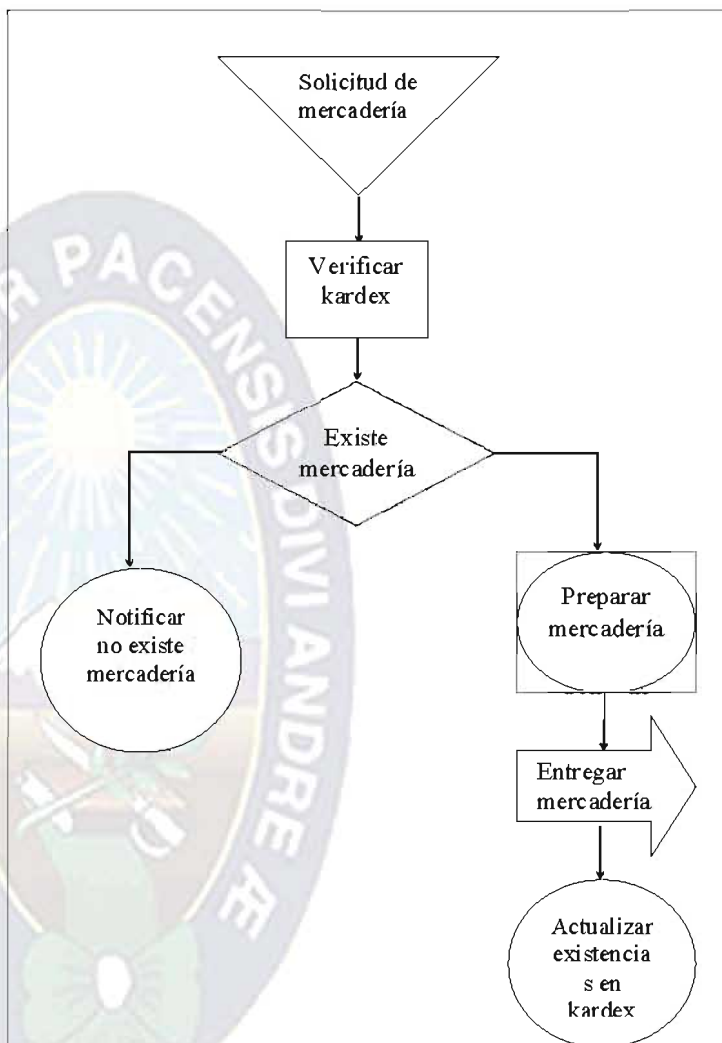


Figura 8: Actualización de las ventas en almacenes,  
Fuente: Elaboración Propia.

Los procesos de ingreso de mercadería, salida o actualización de las ventas y la elaboración de proformas y cotizaciones están automatizados con macros de la hoja electrónica Lotus 123 versión 5.0. A continuación se indica el funcionamiento de las macros.

La importación o entrada de mercadería utiliza la macro Ztar, es la macro principal, opera cada artículo en función del código, indicando a que archivo corresponde. La macro se activa con las teclas [ALT + L] y presenta las siguientes opciones:

- Buscar: Esta opción subdivide en tres alternativas: Automático, uno solo e interrupción.

Al seleccionar la opción automático se ingresa un numero de código y con [ALT + L] busca el artículo o los artículos relacionados, indicando la primera letra del archivo (cada archivo tiene un nombre diferente ordenado alfabéticamente) donde se encuentra. La opción uno solo trabaja de la misma forma que la anterior, con la diferencia que muestra solo el código del artículo digitado. La opción interrupción da la posibilidad de ingresar el nuevo código de forma ordenada después de ejecutar una búsqueda de los códigos.

- Reactualizar. Una vez en pantalla todos los datos de un código de artículo este se puede modificar en relación al costo del proveedor o precio de venta con los datos nuevos.

- Extraer. Muestra el historial de ventas, el historial de importación y la variación de precios de un artículo.

- Desreactualizar. Esta opción permite corregir los cambios realizados en la opción Reactualizar.

El proceso de salida o actualización de las ventas utiliza la macro Zlupe se activa con las teclas [ALT + L], tiene las siguientes opciones

- Buscar. Trabaja de la misma manera que la opción buscar de la macro Ztar del proceso anterior.

- Seguir. Se activa con [ALT + L] después de buscar los códigos de las facturas esta macro los ordena con los precios originales para verificar que los datos registrados en la factura son correctos y con [CTRL + L] los actualiza en los archivos de la unidad principal (unidad C:\)

- Nuevo Código. Se introduce un nuevo código y lo ordena con los datos correctos funciona también con [ALT + L].

- Reactualizar. Registra el historial de ventas de los códigos de los artículos vendidos en los archivos de Lotus en la unidad secundaria (unidad D:\).

El proceso de elaboración de proformas incluyendo descuentos es realizado por la macro Zcliente. Esta macro tiene tres opciones: Facturas, listas y especial.

- Facturas. Esta tiene dos alternativas para trabajar, la primera preparación del formato de factura como su nombre lo indica prepara la factura para luego introducir los códigos, busca los mismos y termina con la presentación con el formato de una factura. La segunda alternativa es buscar similar a la macro buscar de Ztar del proceso de importación
- Listas. Esta macro tiene dos opciones cliente especial y normal. La primera opción cliente especial lista los clientes que tienen un descuento especial. La segunda opción normal muestra los datos necesarios (código, descripción y precio de venta) para elaborar un proforma:
- Especial. Realiza una proforma (cotización) donde se aplica un porcentaje de ganancia, es decir se calcula el mayor descuento posible cuidando el margen de utilidad.



## CAPÍTULO 3

## MARCO TEÓRICO

### 3.1. INGENIERÍA DE SOFTWARE ORIENTADO A OBJETOS.-

Para definirlo es necesario recurrir primero a la Ingeniería del Software. Empezamos indicando que la ingeniería es el análisis, diseño, construcción, verificación y gestión de entidades técnicas (o sociales), en el caso de estudio la entidad es el software de computadora [PRE-2002].

Existen varias definiciones del término “Ingeniería del Software”, dentro de las cuales se encuentran las siguientes:

- Ingeniería de Software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada requerida para desarrollar, operar (funcionar) y mantenerlos. Se conoce también como desarrollo de software o producción de software [BOE-1981].
- Ingeniería de software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema hasta el mantenimiento de este después que se utiliza [SOM-2005].
- ingeniería de software es la aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software”, según IEEE<sup>8</sup>.

La última definición es la que se tomara como referencia principal, para al presente proyecto, pues el objetivo es la aplicación de los enfoques mencionados para obtener un producto software de calidad.

Para construir ingeniería del software adecuado, se define un proceso de desarrollo de software, definido en tres, fases genéricas [PRE-2002]:

Primero, la fase definición se centra sobre el que se va a desarrollar, en esta fase se identifican los requisitos clave del sistema y del software. Los métodos aplicados varían según la metodología<sup>9</sup>, no obstante se tendrán tres tareas principales: ingeniería de sistemas, planificación del proyecto y análisis de requisitos.

---

<sup>8</sup> Del ingles Institute of Electrical and Electronics Engineers, IEEE (Instituto de ingenieros eléctricos y electrónicos)

<sup>9</sup> Conjunto de pasos que deben seguirse para el desarrollo de software.

Segundo, la fase de desarrollo se centra en el como, se han de diseñar las estructuras de datos, la arquitectura del software, la implementación y las interfaces. Los métodos varían, aunque se definen tres tareas específicas: diseño del software, generación de código, y pruebas del software.

Y tercero la fase de mantenimiento se centra en el cambio, se encuentra cuatro tipos de mantenimiento.

Correctivo, modificación del software para corregir defectos.

Adaptativo, transformación del software a los cambios del entorno externo.

Perfectivo, mejorar el software descubriendo funciones adicionales.

Preventivo, tiene que ver con la reingeniería del software, debe permitir que el software sirva a los usuarios finales.

Por otro lado vivimos en un mundo de objetos. Estos objetos existen en la naturaleza, entidades hechos por el hombre, en los negocios y productos que usamos. Por esto no es sorprendente que se proponga una visión orientada a objetos OO, para la creación de software de computadora, una abstracción que modela el mundo de forma tal que nos ayuda a entenderlo y gobernarlo mejor [PRE-2002].

Las tecnologías de objetos lleva un numero de beneficios inherentes que proporcionan ventajas a los niveles de dirección y técnico, como la reutilización (de componentes de software), lleva un desarrollo de software mas rápido, programas de mejor calidad y es mas fácil de mantener debido a que su estructura es relativamente descompuesta.

Los sistemas OO tienden a evolucionar con el tiempo, por esto un modelo de proceso evolutivo acoplado con un enfoque que fomenta el ensamblaje (reutilización) de componentes es el mejor modelo para la ingeniería de software orientado a objetos. El proceso orientado a objetos se mueve a través de una espiral evolutiva como se aprecia en la figura 9. Comienza con comunicación con el usuario, es aquí donde se define el dominio del problema y se identifican las clases básicas del problema. La planificación y el análisis de riesgos establecen una base para el plan del proyecto. El trabajo asociado con la ingeniería del software orientado a objetos sigue el camino mostrado con las flechas.



Establecen una base para el proyecto Orientado a Objetos. El trabajo técnico asociado a la ingeniería del software sigue el camino iterativo que indican las flechas. La ingeniería del software.

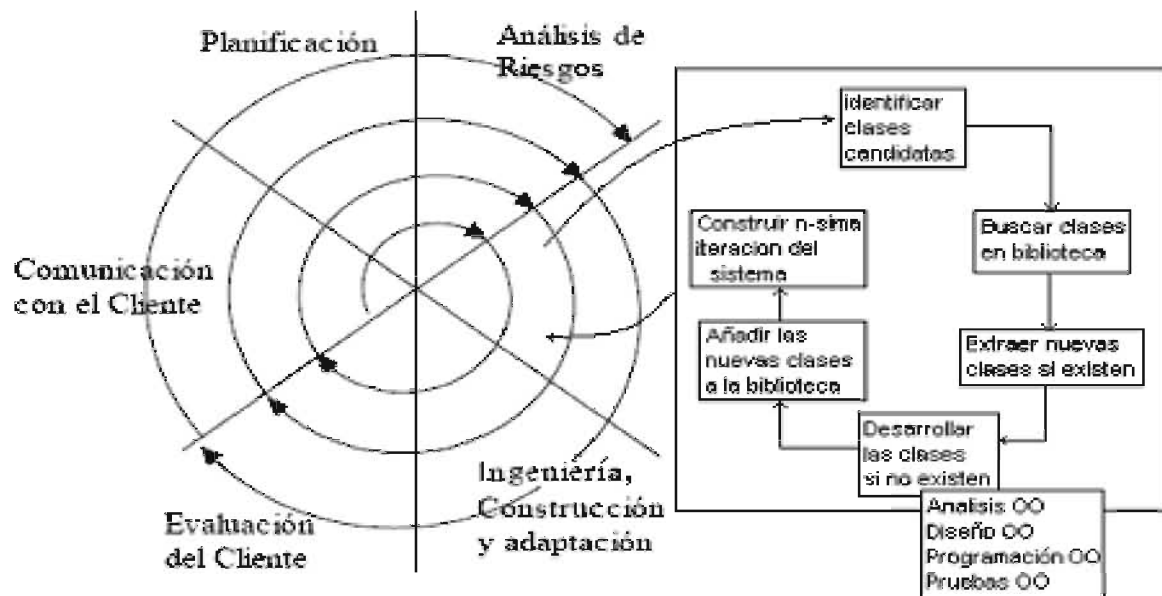


Figura 9: Modelo de proceso Orientado a Objetos. Fuente: Pressman [PRE-2002].

**3.1.1. Planificación y Análisis de Riesgos.-** El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita hacer estimaciones razonables de recursos, costos y planificación temporal. Estas estimaciones se hacen dentro de un marco de tiempo limitado al comienzo de un proyecto y debería actualizarse regularmente a medida que progresa el proyecto, es decir es un proceso iterativo que se aplica a lo largo de todo el proyecto [PRE-2002].

Una tarea importante en la gestión de proyectos es anticipar los riesgos que podrían afectar a la programación del proyecto o a la calidad del software a desarrollar y emprender acciones para evitar riesgos. De forma simple, se puede concebir un riesgo como una probabilidad de que una circunstancia ocurra. Los riesgos son una amenaza para el proyecto, para el software que se esta desarrollando y para la organización. La identificación de riesgos se lleva a cabo de un proceso en grupo utilizando un enfoque de tormenta de ideas o simplemente basarse en la experiencia. Durante el análisis de riesgos, se considera por separado cada riesgo identificado y se decide acerca de la

probabilidad y la seriedad del mismo. Los resultados de este análisis de riesgos se deben documentar a lo largo del proyecto junto con el análisis de consecuencias cuando el riesgo ocurra. En cuanto a los riesgos más probables y potencialmente serios, se deben hacer planes para anularlos, gestionarlos o tratarlos.

**3.1.2. Análisis Orientado a Objetos.-** El objetivo del análisis orientado a objetos AOO es desarrollar una serie de modelos que describan el software de computadora, para satisfacer todo un conjunto de requisitos definidos por el cliente.

El AOO se basa en un conjunto de conceptos y principios básicos del análisis convencional, en Pressman [PRE-2002] menciona los siguientes principios:

- 1) Modelar el dominio de la información.
- 2) Describir la función del módulo.
- 3) Representar el comportamiento del modelo.
- 4) Dividir el modelo para mostrar mas detalles.
- 5) Observar que los modelos iniciales representan la esencia del problema mientras que los últimos aportan detalles de la implementación.

El AOO permite modelar un problema a través de la representación de objetos, atributos y operaciones que son relevantes al problema a resolver, para cumplirlo se deben ejecutar las siguientes tareas [PRE-2002]:

- 1) Los requisitos básicos del usuario se comunican entre el cliente y el desarrollador.
- 2) Identificar las clases.
- 3) Especificar una jerarquía de clases.
- 4) Representar las relaciones objeto a objeto.
- 5) Modelar el comportamiento del objeto.
- 6) Repetir las tareas del 1 al 5 hasta completar el modelo

Una amplia variedad de métodos de AOO han sido propuestos cada uno de ellos introduce un proceso para el análisis de un producto o sistema, un conjunto de modelos que evolucionan fuera el proceso y una notación que posibilita crear cada modelo de una manera consistente. Entre los métodos más utilizados se encuentran:

- a) El método de Booch, mantiene un enfoque evolutivo basado en macro y micro procesos.
- b) El método de Rumbaugh, más conocido como Técnica de Modelado de Objetos (OMT) para análisis y diseño de sistemas. Dentro del análisis se crean tres modelos: el modelo de objetos, el modelo dinámico y el modelo funcional.
- c) El método de Jacobson, también llamado OOSE<sup>10</sup>, es una versión simplificada del proceso de desarrollo de software Objectory<sup>11</sup>.  
Se diferencia de otros por la importancia que da a los escenarios también denominados casos de uso<sup>12</sup>.
- d) El método de Coad y Yourdon, consiste en identificar objetos, definir una estructura de generalización–especialización, definir una estructura de todo – parte, identificar temas, definir atributos y operaciones.
- e) El método Diseño del Manejo de Responsabilidades RDD<sup>13</sup>, también denominado método de Wirfs-Brock, no hace una distinción clara entre las tareas de análisis y diseño. En su lugar, propone un proceso continuo que comienza con la valoración de una especificación del cliente y termina con el diseño.
- f) La notación del Lenguaje de Modelado Unificado UML, desarrollado por Grady Booch, James Rumbaugh e Ivar Jacobson, permite visualizar, especificar, construir y documentar los modelos de un sistema que involucra una gran cantidad de software, desde una perspectiva OO. Posteriormente surge la metodología del Proceso Unificado de Rational Rose RUP (Rational Unified Process) a la cabeza de de los autores de UML, que es adoptado como lenguaje de modelamiento. A la fase de análisis se la denomina inicio, donde se busca establecer los objetivos del ciclo de vida del producto, se establece el modelo del negocio con el fin de delimitar el alcance del sistema y tener una visión general del proyecto [6].
- g) La Metodología programación extrema, XP (Extreme Programming<sup>14</sup>) es una de las metodologías considerada ligeras porque buscan métodos sencillos para

---

<sup>10</sup> En español Ingeniería de Software Orientada a Objetos.

<sup>11</sup> Abreviatura de Object Factory – Fábrica de Objetos.

<sup>12</sup> Una descripción o escenario que hace referencia cómo el usuario interactúa con el sistema.

<sup>13</sup> Por sus siglas en ingles de la palabra Responsibility Driven Design

<sup>14</sup> [www.extremeprogramming.org].



obtener software de calidad y simplifica al máximo las tareas de ingeniería del software, posee pocas reglas y técnicas. A la etapa de análisis se denomina Planificación, donde se lleva a cabo dos eventos, el levantamiento de requerimientos (en XP historias de usuarios) y el acuerdo de partes para la estimación de tiempo y costos del proyecto así como un cronograma de actividades los cuales van evolucionando por iteraciones

**3.1.3. Diseño Orientado a Objetos.**- El Diseño Orientado a Objetos DOO transforma el modelo de análisis orientado a objetos, en un modelo de diseño que sirve como anteproyecto para la construcción del software. Las componentes principales del sistema están organizadas en subsistemas o módulos. Los datos y operaciones que manejan datos están encapsulados en objetos. El DOO debe describir la organización específica de datos de los atributos y los detalles procedimentales de las operaciones. Esta representación datos y algoritmos de un sistema orientado a objetos contribuyen a una modularidad general (exhibida con una cohesión alta y con un acoplamiento bajo) [PRE-2002].

El DOO se apoya en cuatro conceptos importantes de diseño de software abstracción, ocultación de la información, independencia funcional y modularidad.

Bertrand Meyer considerado el padre del paradigma orientado a objetos [MEY-1998] sugiere cinco principios de diseño básicos que pueden derivarse para arquitecturas modulares:

- 1) Unidades modulares lingüísticas, que se corresponden con unidades sintácticas en el lenguaje usado. El lenguaje de programación, debe soportar la modularidad definida.
- 2) Pocas interfaces, minimizar en lo posible el numero de interfaces.
- 3) Pequeñas interfaces, minimizar la cantidad de información que se mueven a través de la interfaz.
- 4) Interfaces explicitas, cada vez que se comunican los módulos, deben realizarse de manera obvia y directa.
- 5) Ocultación de la información, se logra cuando toda la información sobre un modulo esta oculta al acceso exterior, al menos que se defina como publica.

Los principios de diseño descritos, pueden aplicarse a cualquier método de diseño. En la sección anterior se ha presentado algunos métodos del AOO, cada uno de estos métodos posee su correspondiente proceso de diseño orientado a objetos.

- a) El método de Booch, dentro del cual, el diseño se encuentra plasmado tanto dentro del macro desarrollo como del micro desarrollo. En el primero se engloba una actividad de planificación arquitectónica y se crea un prototipo de diseño. En el segundo se define un conjunto de “reglas” que regulan el uso de operaciones y atributos.
- b) El método de Rumbaugh u OMT, engloba una actividad de diseño que alienta al mismo a ser conducido a dos diferentes niveles de abstracción: el diseño del sistema y el diseño de objetos.
- c) El método de Jacobson, dentro del cual, el diseño enfatiza la planificación para el modelo de análisis, en principio, el modelo idealizado de análisis se adapta para acoplarse al ambiente del mundo real.
- d) El método de Coad y Yourdon, este método se enfoca en la representación de cuatro componentes mayores de sistemas: el dominio del problema, la interacción humana, la administración de tareas y la administración de datos.
- e) El método de Wirfs-Brock, define un conjunto de tareas técnicas, en la cual el análisis conduce sin duda al diseño, dentro del cual cada operación (responsabilidad) y protocolo (diseño de interfaz) se diseña hasta un nivel de detalle que guiará la implementación.
- f) La notación de UML, en el contexto del diseño, se organiza en dos actividades mayores: diseño del sistema y diseño de objetos. El objetivo de diseño de sistemas, es representar la arquitectura del software y la definición de subsistemas. Por otro lado, el diseño de objetos se centra en la descripción de objetos hasta un nivel en el cual puedan ser implementados en un lenguaje de programación. En el Proceso Unificado el diseño es considerado Elaboración cuyo objetivo principal es plantear la arquitectura del ciclo de vida del producto
- g) El método de Programación Extrema, considera en esta fase, las historias que se van a utilizar en la iteración actual. Se hace uso de herramientas, Metáforas del

Sistema útil (estandarización de los nombres, que busca la forma más simple de estructurar el sistema) y las tarjetas CRC (Clase, responsabilidad y colaboración).

**3.1.4. Pruebas Orientadas a Objetos.-** El objetivo general de las pruebas orientadas a objetos es encontrar el número máximo de errores con el mínimo esfuerzo y tiempo. La prueba no puede asegurar la ausencia de defectos, solo muestra que existen defectos en el software los cuales deben ser corregidos.

Pressman [PRE-2002] sugiere un conjunto de principios de prueba:

1. A todas las pruebas se les deberá poder hacer un seguimiento hasta los requisitos del cliente
2. Las pruebas debería planificarse mucho antes que empiecen.
3. El principio de Pareto<sup>15</sup> es aplicable a la prueba del software.
4. Las pruebas deberían empezar por los módulos individuales y progresar hacia el sistema entero.
5. No son posibles las pruebas exhaustivas.
6. Las pruebas deberían ser ejecutadas por un grupo independiente, para ser más efectivas.

La prueba de software a menudo se conoce como verificación y validación (V & V). La verificación se refiere al conjunto de actividades que aseguran que el software implementa correctamente una función específica. La validación se refiere al conjunto de actividades que aseguran que el software construido se ajusta a los requisitos del cliente.

Bohem lo define de la siguiente manera [BOE- 1981]:

Verificación: ¿Estamos construyendo el producto correctamente?

Validación: ¿Estamos construyendo el producto correcto?

Es necesario probar un sistema orientado a objetos, en una variedad de niveles diferentes, en un esfuerzo para descubrir errores, que deben ocurrir cuando las clases colaboran con otras entre sí, y los subsistemas se comunican por medio de las capas arquitectónicas [PRE-2002].

Como los modelos de análisis y diseño y el código fuente están semánticamente enlazados, las pruebas en forma de revisiones técnicas comienzan durante estas

---

<sup>15</sup> El principio de Pareto indica que el 80% de todos los errores descubiertos durante las pruebas surgen al hacer un seguimiento de solo el 20% de todos los módulos del programa.

actividades. Por esta razón, la revisión de los modelos CRC, objeto-relación y objeto-comportamiento puede verse como una etapa de las pruebas.

La prueba es un conjunto de actividades que se pueden planificar y llevar a cabo sistemáticamente, para ello se elabora un estrategia de prueba de software que debe incluir pruebas de bajo nivel que verifiquen que todos los pequeños segmentos de código fuente se han implementado correctamente, así como pruebas de alto nivel que validen las principales funciones del sistema frente a los requisitos del cliente [PRE-2002].

La estrategia general de prueba del software cuenta con los siguientes pasos

- a. Prueba de unidad: Al considerar el software orientado a objetos, la menor unidad a probar lo constituyen las clases. Una clase puede contener un cierto número de operaciones y una operación particular puede existir como parte de un número de clases diferentes, por esta razón no se puede probar con detalle una operación aisladamente sino como parte de una clase. La prueba de clases esta dirigido por las operaciones encapsuladas y el estado de comportamiento de la clase.
- b. Prueba de integración: Dado que el software orientado a objetos no tiene una estructura de control jerárquica, existen dos estrategias diferentes para pruebas de integración en sistemas orientada a objetos.

La primera, integra el conjunto de clases necesario para responder a una entrada o evento del sistema. Cada hilo de control se integra y prueba individualmente. Se aplica la prueba de regresión para asegurar que no ocurren efectos colaterales.

El segundo enfoque para la integración, comienza la construcción del sistema probando aquellas clases (independientes) que usan muy pocas de las clases servidor. Después de probar las clases independientes, se comprueba la próxima capa de clases, llamadas clases dependientes, que usan las clases independientes.

- c. Prueba de validación: Para asistir en la determinación de pruebas de validación, el ejecutor de la prueba debe basarse en los casos de uso que forman parte del modelo de análisis. Los casos de uso brindan datos de entrada muy útiles en el diseño de pruebas de caja negra, basados en estados.
- d. Los métodos tradicionales de prueba de caja negra pueden usarse para dirigir las pruebas de validación.

**3.1.5. Métricas Orientadas a objetos.-** La medición permite que gestores y profesionales mejoren el proceso del software (ayudan en la planificación, seguimiento y control de un proyecto de software) y evalúan la calidad del producto software que se produce. Las medidas de los atributos específicos del proceso, del proyecto y del producto se utilizan métricas del software, estas proporcionan indicadores que guían acciones de gestión y técnicas [PRE-2002].

Uno de los objetivos fundamentales del uso de métricas en sistemas software es evaluar los mismos con el fin de controlar y mejorar su calidad.

Haciendo un repaso histórico los inicios de la medición pueden datarse en los años 60, en que sólo se recogían medidas muy primitivas tales como las líneas de código (LDC) de los módulos y el tiempo dedicado en cada etapa del ciclo de vida.

Modelos posteriores (COCOMO<sup>16</sup>) en los años 70 intentaron establecer una relación entre el tamaño de los programas y el esfuerzo necesario para realizarlos y mantenerlos [BOE-1981]. McCall y Boehm, entre otros, han definido modelos de medidas orientados hacia la evaluación de la calidad de los productos software. En los años 80 se profundizó en métricas de complejidad y de diseño modular y estructurado, modelos de estimación del costo y del tiempo de desarrollo de los proyectos software. Aparecieron las normas ISO 9000 y los modelos CMM (capacity maturity model) del Instituto de Ingeniería del Software, centrados en los procesos software.

El enfoque de métricas orientadas a objetos está en la clase, y esta se constituye en la piedra fundamental de la arquitectura orientada a objetos [PRE-2002].

La tendencia de la Ingeniería del Software es introducir la medición en todas las etapas del ciclo de vida de un proyecto software independientemente del modelo utilizado. Así las métricas se las puede ver desde tres puntos de vista:

- Característica o atributo del software a medir (por ejemplo, cohesión, complejidad).
- Etapa del ciclo de vida en la que se mide (análisis, diseño o Implementación).
- Nivel de granularidad en el que se mide (por ejemplo, nivel de sistema, de programa, de clase, de método o de variable).

---

<sup>16</sup> Del inglés CONstructive Cost Model (MOdelo CONstructivo de COsto).



Las métricas no pueden ser aplicadas indiscriminadamente a cualquiera de los atributos del software que se quieren medir. El caso más típico de una asignación errónea de métricas a atributos es el de líneas de código, que siendo válida como una medida del tamaño, se usaba equivocadamente para medir la complejidad de los programas. Por otro lado, no todas las métricas han de ser recogidas durante la fase de implementación del código fuente de los programas, aunque un gran número de ellas se obtenga de ahí. Sería deseable obtenerlas más tempranamente en la fase de diseño.

En [PIA-2001] Chidamber y Kemerer establecen 6 métricas para medir cinco atributos básicos en el diseño orientado a objetos:

- acoplamiento,
- complejidad de una clase (no se toma en cuenta porque No se dan guías para la interpretación de esta métrica),
- reutilización, cohesión,
- y herencia.

Acoplamiento. Es una medida de la interconexión entre objetos. Indica en qué medida una clase utiliza atributos y/o métodos de otro objeto. En la tabla 2 se muestra el acoplamiento entre objetos

Acoplamiento entre objetos (Coupling between Objects-CBO-) [Chidamber y Kemerer, 1994]			
Definición:	CBO de una clase es el número de clases a las cuales una clase está ligada, sin tener con ella relaciones de herencia. Hay dependencia entre dos clases cuando una de ellas usa métodos o variables de la otra clase. Es consistente con las tradicionales definiciones de acoplamiento: "medida del grado de interdependencia entre módulos".		
Valoración:	Los autores sugieren que sea un indicador del esfuerzo necesario para el mantenimiento y en las pruebas. Cuanto más independiente es un objeto, más fácil es reutilizarlo en otra aplicación. Al reducir el acoplamiento se reduce la complejidad, se mejora la modularidad y se promueve el encapsulamiento. Una medida del acoplamiento es útil para determinar la complejidad de las pruebas necesarias de distintas partes de un diseño. Cuanto mayor sea el acoplamiento entre objetos más rigurosos han de ser las pruebas.		
Ciclo de vida:	diseño	Nivel de granularidad:	Clase, programa, sistema.

Tabla 2: Acoplamiento entre objetos, Fuente: Piatinni 2001

## Complejidad.

Complejidad ciclomática media (Average v(G) [McCabe, 1976])			
Definición:	Es la complejidad ciclomática media de los métodos de una clase. La complejidad ciclomática v(G) de un método es el número de caminos independientes a lo largo del método. En métodos estructurados, v(G)=número de nodos de decisión más uno.		
Valoración:	Se ofrece como una medida de la complejidad estructural. Una medida que se concentra en los métodos más complejos es la complejidad ciclomática máxima (Maximum v(G)). Es un indicador útil de la dificultad de prueba y mantenimiento de un programa. Se sugiere un valor máximo de 10.		
Ciclo de vida:	Implementación.	Nivel de granularidad:	Clase, programa, sistema.

Tabla 3: Complejidad ciclomática, Fuente: Piatinni 2001

Reusabilidad. Propiedad de los componentes software por la que pueden aplicarse a tareas distintas de aquellas para las que se construyeron inicialmente.

Número de hijos (Number of Children-NOC-) [Chidamber y Kemerer, 1994]			
Definición:	Es el número de subclases subordinadas a una clase en la jerarquía, es decir, la cantidad de subclases que pertenecen a una clase.		
Valoración:	Es un indicador del nivel de reutilización, la posibilidad de haber creado abstracciones erróneas, y es un indicador del nivel de pruebas requerido. Un mayor número de hijos requiere más pruebas de los métodos de esa clase y mayor dificultad para modificar una clase pues afecta a todos los hijos de dicha clase. Clases en un nivel más alto en la jerarquía deberían tener más subclases que las clases en un nivel más bajo en la jerarquía. NOC puede ser también un indicador del uso inadecuado de la herencia. Es un potencial indicador de la influencia que una clase puede tener sobre el diseño del sistema. Si el diseño depende mucho de la reutilización a través de la herencia, quizás sea mejor dividir la funcionalidad en varias clases		
Ciclo de vida:	diseño	Nivel de granularidad:	Clase, programa, sistema.

Tabla 4: Número de hijos, Fuente: Piatinni 2001

Cohesión. Un objeto tiene un alto nivel de cohesión si ejecuta una tarea sencilla y requiere poca interacción con métodos que se ejecutan en otros objetos. Es una

extensión del concepto de ocultamiento de la información. Un objeto coherente debe hacer (idealmente) una sola cosa.

Falta de cohesión en los métodos (Lack of Cohesion in Methods-LCOM-) [Chidamber y Kemerer, 1994]			
Definición:	Número de grupos de métodos locales que no acceden a atributos comunes.		
Valoración:	Indica la calidad de la abstracción hecha en la clase. Usa el concepto de grado de similitud de métodos. Si no hay atributos comunes, el grado de similitud es cero. Una baja cohesión incrementa la complejidad y por tanto la facilidad de cometer errores durante el proceso de desarrollo. Estas clases podrían probablemente ser divididas en dos o más subclases aumentando la cohesión de las clases resultantes. Es deseable una alta cohesión en los métodos dentro de una clase, ya que ésta no se puede dividir fomentando el encapsulamiento. [Henderson-Sellers, 1996] destaca dos problemas con esta métrica: Dos clases pueden tener LCOM=0, mientras una tiene más variables comunes que la otra y No se dan guías para la interpretación de esta métrica.		
Ciclo de vida:	Implementación	Nivel de granularidad:	Clase.

Tabla 5: Falta de cohesión, Fuente: Piatinni 2001

Herencia. El uso de la herencia debe entenderse como un compromiso entre la facilidad de reutilización que proporciona y la facilidad de comprensión y de mantenimiento del sistema, que en general, están en relación inversa. Altos niveles de herencia indican objetos complejos, los cuales pueden ser difíciles de probar y reutilizar, y bajos niveles en la herencia pueden señalar que el código está escrito en un estilo funcional, sin aprovechar el mecanismo de herencia proporcionado por la orientación a objetos.

Profundidad del árbol de herencia (Depth of Inheritance Tree-DIT-) [Chidamber y Kemerer, 1994]	
Definición:	DIT mide el máximo nivel en la jerarquía de herencia. Se trata de la cuenta directa de los niveles en la jerarquía de herencia. En el nivel cero de la jerarquía de encuentra la raíz.
Valoración:	Medida de la complejidad de una clase, la complejidad del diseño y la reutilización potencial, lo que se debe a que cuanto más profunda se encuentra una clase en la jerarquía, mayor es la probabilidad de heredar más métodos. Es una medida de cuántos ancestros pueden afectar a esta clase. En general, la herencia se maneja poco. [Cartwright y Shepperd, 1996] han encontrado una correlación positiva entre DIT y el número de problemas emitidos por el usuario. Lorenz y Kidd, sugieren un umbral de 6 niveles como indicador de un abuso en la herencia tanto en Smalltalk como en C++. Sistemas construidos a partir de frameworks suelen presentar unos niveles de herencia



	altos, ya que las clases se construyen a partir de una jerarquía existente. En lenguajes como Java o Smalltalk, las clases siempre heredan de la clase Object, lo que añade uno a DIT. Los problemas que surgen con esta métrica se deben a las diferentes características de la herencia, ya que DIT no queda claramente definida y no se puede ver como una medida de reutilización. Es fácil imaginar clases con gran profundidad en la jerarquía reutilizando menos métodos que una clase poco profunda, pero que es muy ancha.		
Ciclo de vida:	Diseño.	Nivel de granularidad:	Clase.

Tabla 6: Profundidad de un árbol de herencia, Fuente: Piatinni 2001

### 3.2. TÉCNICAS DE MODELADO A OBJETOS.-

La metodología técnica de modelado a objetos OMT (Object Modeling Technique) fue creada por James Rumbaugh en 1991, mientras dirigía un equipo de investigación de los laboratorios General Electric.

La metodología OMT es una de las metodologías de análisis y diseño orientadas a objetos, más maduros y eficientes que existen en la actualidad. La metodología es de carácter abierta, lo que permite ser de dominio público. Esto facilita su evolución para acoplarse a todas las necesidades actuales y futuras de la ingeniería de software [RUM-1996]

La metodología se divide en tres clases de modelos para describir el sistema

- **Modelo de Objetos.** Esta es la parte principal de la Técnica para modelado ya que se fundamenta en la teoría del paradigma orientado a objetos. La definición clara de las entidades que intervienen en el sistema es un paso inicial necesario para definir qué transformaciones ocurren en ellas y cuándo se producen estas transformaciones. Esta forma de pensar es inherente al paradigma Orientado a Objetos donde las clases y su jerarquía determinan el sistema. Los diagramas de objetos permiten representar gráficamente los objetos, las clases y sus relaciones mediante dos tipos de diagramas: los diagramas de clases y los diagramas de casos concretos u objetos (instancias).

Los diagramas de clases describen las clases que componen el sistema y que permitirán la creación de casos concretos, los diagramas de casos concretos describen la manera en que los objetos del sistema se relacionan y los casos concretos que existen en

el sistema de cada clase. En los diagramas que componen este modelo se representan los siguientes elementos del sistema: objetos y clases, atributos, operaciones, y relaciones o asociaciones.

- **Modelo Dinámico.** Los aspectos del sistema que están relacionados con el tiempo y con los cambios constituyen el modelo dinámico.

Los conceptos más importantes del modelado dinámico son los sucesos, que representan estímulos externos, y los estados, que representan los valores de los objetos.

Los diagramas de estados muestran los valores de los atributos y los enlaces mantenidos por un objeto lo que se denomina estado. A lo largo del tiempo, los objetos se estimulan unos a otros, dando lugar a una serie de cambios en sus estados. Un estímulo individual proveniente de un objeto y que llega a otro es un suceso. La respuesta a un suceso depende del estado del objeto que lo recibe, y puede incluir un cambio de estado o el envío de otro suceso al remitente o a un tercer objeto. La trama de sucesos, estados y transiciones de estados para una clase dada se puede abstraer y representar en forma de un diagrama de estados. El modelo dinámico consta de múltiples diagramas de estados, con un diagrama de estados para cada clase que posea un comportamiento dinámico importante, y muestra la trama de actividad para todo el sistema.

- **Modelo Funcional.-** El modelo funcional describe los cálculos existentes dentro del sistema siendo la tercera parte del modelado. El modelo funcional muestra la forma en que se derivan los valores producidos en un cálculo a partir de los valores introducidos, sin tener en cuenta el orden en el cual se calculan los valores. Consta de múltiples diagramas de flujo de datos, que muestran el flujo de valores desde las entradas externas, a través de las operaciones y almacenes internos de datos hasta las salidas externas. También incluyen restricciones entre valores dentro del modelo de objetos. Los diagramas de flujo de datos no muestran el control ni tampoco información acerca de la estructura de los objetos; todo esto pertenece a los modelos dinámicos y de objetos. Dentro del modelado del sistema, el modelo funcional especifica lo que sucede, el modelo dinámico cuándo sucede, y el modelo de objetos especifica a qué le sucede.

La metodología se divide en cuatro fases consecutivas, tiene una parte de diseño no muy compleja, se centra mucho en un buen análisis y esta dirigida a los datos

**3.2.1. Análisis.-** Es el primer pasó de la metodología que concierne a la obtención de un modelo preciso, conciso, comprensible y correcto del mundo real. El modelo de análisis es una abstracción resumida y precisa de lo que hará el sistema y no en como lo hace. Los elementos del modelo deben ser conceptos del dominio de aplicación y no conceptos informáticos tales como estructuras de datos. Para este propósito es necesario examinar los requisitos, analizar sus implicaciones y volver a plantearlos rigurosamente. El modelo del mundo real consta de los modelos de objetos, dinámico y funcional el análisis no siempre se efectúa en una secuencia rígida, los modelos se construyen iterativamente [RUM-1996]. El análisis no es un proceso mecánico

Las actividades del modelo de análisis son:

1. Definición del problema, debería indicar lo que hay que hacer y no como hacerlo, debe ser una exposición de nuestras necesidades y no una propuesta de solución.
2. Construcción del modelo de objetos. El modelo de objetos muestra la estructura estática de los datos correspondientes al sistema del mundo real y sus relaciones entre si. La información del modelo de objetos proviene del modelo de la definición del problema del conocimiento de expertos acerca del dominio de la aplicación y del conocimiento general del mundo real [RUM-1996].

La construcción del modelo de objetos implica los siguientes pasos:

- Identificar las clases de objetos relevantes en el dominio de la aplicación entre estos se cuentan las entidades físicas y conceptos. Todas las clases tienen que tener sentido en el dominio de la aplicación.
- Comenzar a crear un diccionario de datos que contengan descripción de las clases, atributos y sus acciones porque las palabras aisladas tienen demasiadas interpretaciones, también se describen los alcances y descripciones de las clases.
- Añadir las asociaciones entre las clases. Toda dependencia entre dos o mas clases es una asociación y una referencia de una clase a otra también lo es.
- Añadir los atributos de los objetos y sus uniones. Los atributos son propiedades de objetos individuales, los atributos no deben ser objetos, se consideran solo aquellos que estén relacionados directamente con la aplicación.

- Organizar y simplificar las clases usando herencias para compartir una estructura común. La herencia se realiza en dos direcciones, generalizando (refinamiento ascendente) y especializando (refinamiento descendente).
  - Probar que los accesos son correctos siguiendo el diagrama del modelo de objetos para ver si se tienen resultados sensatos, usando escenarios e iterando los pasos siguientes cuando sea necesario por que es raro que un modelo de objetos sea correcto a la primera pasada. Todo el proceso de desarrollo de software es una continua iteración.
  - Agrupar las clases en módulos, un modulo es un conjunto de clases que captura algún subconjunto lógico del modelo completo. Las clases que estén fuertemente acopladas<sup>17</sup> deberían ir agrupadas pero buscando que un modulo hace (de manera ideal) una sola cosa.
3. Desarrollo del modelo dinámico. El modelo dinámico muestra la forma en que el comportamiento del sistema y de los objetos de que consta va variando con el tiempo. Para construir el modelo de objetos se lleva a cabo los siguientes pasos:
- Preparar los escenarios para las secuencias de interacción típicas entre los usuarios y el sistema. Estos escenarios muestran las interacciones principales, los formatos de visualización externa y los intercambios de información. La mayoría de las interacciones se pueden descomponer en dos partes: lógica de aplicación e interfaz de usuario.
  - Identificar los sucesos que se dan entre objetos y preparar los diagramas de secuencia<sup>18</sup> para modelar el paso de mensajes.
  - Preparar un diagrama de transición de estados DTE para el sistema y una traza de eventos para cada escenario.
  - Verificar la consistencia de los que aparecen en los diagramas.
  - Modelo dinámico esta compuesto por diagramas de estado y diagrama global de flujo de eventos.
4. Construcción del modelo funcional.
- Se identifican los valores de entrada y salida.
  - Se utilizan diagramas de flujo de datos para mostrar las dependencias funcionales.

<sup>17</sup> Medida de interconexión entre los módulos [PRE 2002]

<sup>18</sup> Los diagramas de secuencia son parte de los diagramas de interacción del lenguaje de modelado unificado

- Se describe lo que hace cada función en lenguaje natural o pseudocódigo o algún otro formato adecuado.
- Realizar una verificación, iteración y refinación de los modelos de objetos, dinámico y funcional para producir un diseño limpio y coherente.
- El objetivo del análisis es especificar por completo el problema y el dominio de la aplicación sin introducir ningún sesgo hacia la implementación.

**3.2.2. Diseño de Sistemas.-** Es la estrategia de alto nivel para resolver el problema y construir una solución. El diseño de sistemas es la primera fase de diseño en la cual se selecciona la aproximación básica para resolver el problema, se decide la estructura y el estilo global. La arquitectura del sistema es la organización global del mismo en componentes llamados subsistemas [RUM-1996].

Se consideran los siguientes puntos:

1. Descomposición de un sistema en subsistemas. Cada subsistema abarca aspectos del sistema que comportan alguna propiedad común. Un subsistema es paquete de clases asociaciones, operaciones, sucesos y restricciones interrelacionados que tienen un interfaz definida. Normalmente un subsistema se identifica por los servicios que proporciona. Un servicio es un grupo de funciones relacionadas que comparten algún propósito común, como el procesamiento de entrada y salida. La descomposición de sistemas en subsistemas se puede organizar como una secuencia de capas horizontales (conjunto ordenado de mundos virtuales cada uno de los cuales esta construido en términos de los que tiene por debajo, y proporciona la base para la implementación para aquellos que estén por encima) o en particiones verticales (dividen verticalmente un sistema en varios subsistemas independientes o débilmente acopladas, cada una de las cuales proporciona una clase de servicio).
2. se Identifica la concurrencia inherente en el problema. Dos objetos son concurrentes si pueden recibir sucesos al mismo tiempo sin interactuar
3. Selección de una estrategia para implementar los almacenes de datos en términos de estructuras de datos, archivos y/o bases de datos. Los almacenes de datos internos y externos dentro de un sistema proporciona puntos limpios de separación entre subsistemas con interfaces bien definidas



4. Selección de una aproximación para implementar el control del software. Existen dos clases de flujos de control: el control interno y el control externo, el primero es el flujo de control dentro de un proceso, existe en la implementación, el control externo es el flujo de sucesos externamente visibles entre los objetos del sistema. Existen tres tipos de control para sucesos externos: controlado por procedimientos, controlada por sucesos y concurrente.
5. Considerar las condiciones de entorno. La iniciación, terminación y fallos (o terminaciones no planeadas).
6. Establecer las prioridades de compensación necesarias durante el diseño del sistema.

**3.2.3. Diseño de Objetos.** Sigue al análisis y al diseño de sistemas y no parte de cero, en esta fase se añade detalles de implementación, tales como una reestructuración de las clases para mayor eficiencia, unas estructuras internas y unos algoritmos para implementar las operaciones, la implementación de control y asociaciones y el empaquetamiento en módulos. El diseño de objetos extiende el modelo de análisis mediante decisiones de implementación específicas y con clases, atributos, asociaciones y operaciones internas [RUM-1996].

Durante el diseño de objetos se llevan los siguientes pasos:

1. Obtener las operaciones para el modelo de objetos a partir de los demás modelos. Al efectuar esta transformación, comienza el proceso consistente en hacer corresponder la estructura lógica del modelo de análisis en una organización física de un programa.
  - Encontrar una operación para cada modelo del proceso funcional.
  - Definir una operación por cada evento del modelo dinámico, dependiendo de la implementación del control.
2. Diseñar algoritmos para implementar las operaciones. Cada operación especificada en el modelo funcional debe ser formulada como un algoritmo. El análisis de especificaciones dice lo que hace la operación desde el punto de vista de sus clientes y los algoritmos muestran como se hace.

Se eligen los algoritmos que minimicen el costo de implementar las operaciones y las estructuras de datos apropiadas. También se definen nuevas clases internas y



operaciones como sea necesario y se asignan responsabilidades a operaciones que no han sido claramente asociadas a una clase.

3. Optimizar las vías de acceso a los datos entre eficiencia y claridad. Se consideran los siguientes puntos:
  - Ver la posibilidad de añadir asociaciones redundantes para minimizar el costo del acceso.
  - Se reajustan los procesos computacionales para lograr una mayor eficiencia.
  - Y se guardan los atributos derivados para evitar los cálculos repetidos.
4. Implementar el control del software complementando la aproximación seleccionada (sistema controlado por procedimientos y sucesos) durante el diseño del sistema.
5. Ajustar las definiciones de las clases y de las operaciones para incrementar la herencia. Para esto se reajustan las clases y operaciones para aumentar la herencia. Abstractar el comportamiento común de los grupos de clases.
6. Diseñar la implementación de las asociaciones que proporcionan vías de acceso entre objetos para se realiza lo siguiente:
  - Se analizan las asociaciones si es posible implementar mediante un puntero si la multiplicidad es de uno y se trata de un conjunto de punteros cuando la multiplicidad es de muchos. También es posible implementar cada asociación como si fuera un objeto distinto o añadiendo el valor de objetos como atributos de una o ambas clases de la asociación.
7. Determinar la interpretación exacta de los atributos que son objetos.
8. Empaquetar las clases y las asociaciones en módulos. Los lenguajes orientados a objetos tienen distintos grados de empaquetamiento.  
Para implementar un diseño orientado de objetos en un lenguaje orientado a objetos rigen las siguientes consideraciones
  - Definición de las clases. Se declaran las clases de objetos con todos sus atributos y operaciones del diagrama de objetos.
  - Creación de objetos. El lenguaje de programación tiene que reservar espacio en memoria para los objetos. se recomienda que el lenguaje de programación tengan recolectores de basura automáticos

- Llamadas a operaciones. Las operaciones pueden o no tener argumentos adicionales.
- Uso de la herencia. Los lenguajes orientado a objetos varían en los mecanismos que proporcionan para implementar la herencia la declaración de cada de clase incluye una lista de superclases de las cuales hereda sus atributos y métodos.
- Implementación de asociaciones. Hay dos aproximaciones generales punteros enterrados y objetos de asociación explícitos

**3.2.4. Lenguaje Unificado de Modelado.**- El lenguaje de modelado unificado o UML<sup>19</sup> (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido concebido por los autores de los tres métodos más usados de orientación a objetos (Graddy Booch, Ivar Jacobson y James Rumbaugh). UML incluye los siguientes diagramas

- Diagrama de casos de uso para modelar los procesos
- Diagrama de clases para modelar la estructura estática de las clases en el sistema
- Diagrama de objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de interacción se presentan dos tipos de diagramas
  - Diagrama de secuencia para modelar el paso de mensajes entre objetos.
  - Diagrama de colaboración para modelar interacciones entre objetos.
- Diagrama de estados para modelar el comportamiento de los objetos en el sistema.
- Diagrama de actividades para modelar el comportamiento de los objetos en el sistema.
- Diagrama de componentes para modelar componentes.
- Diagrama de componentes para modelar la distribución del sistema.

Los diagramas que serán parte en el presente proyecto son:

Los Diagramas de Casos de uso, muestran la relación entre los actores y los casos de uso del sistema y representa la funcionalidad que ofrece el sistema en lo que se refiere a su interacción externa.

---

<sup>19</sup> Se mantendrá las siglas en ingles UML, por ser mas conocida en el mundo del desarrollo de software

Los elementos que pueden aparecer en un Diagrama de Casos de Uso son: actores, casos de uso y relaciones entre casos de uso.

Un actor es una entidad externa al sistema que realiza algún tipo de interacción con el mismo. Se representa mediante una figura humana dibujada con palotes. Esta representación sirve tanto para actores que son personas como sistemas, sensores u otras entidades externas.

Un caso de uso es una descripción de la secuencia de interacciones que se producen entre un actor y el sistema, cuando el actor usa el sistema para llevar a cabo una tarea específica. Expresa una unidad coherente de funcionalidad, y se representa en el Diagrama de Casos de Uso mediante una elipse con el nombre del caso de uso en su interior. El nombre del caso de uso debe reflejar la tarea específica que el actor desea llevar a cabo usando el sistema. Entre dos casos de uso puede haber las siguientes relaciones:

Extiende: Cuando un caso de uso especializa a otro extendiendo su funcionalidad.

Usa: Cuando un caso de uso utiliza a otro.

Se representan como una línea que une a los dos casos de uso relacionados, con una flecha en forma de triángulo y con una etiqueta <<extiende>> o <<usa>> según sea el tipo de relación.

En el diagrama de casos de uso se representa también el sistema como una caja rectangular con el nombre en su interior. Los casos de uso están en el interior de la caja del sistema, y los actores fuera, y cada actor está unido a los casos de uso en los que participa mediante una línea. En la figura 10 se muestra un ejemplo de la interacción del personal de apoyo con el sistema

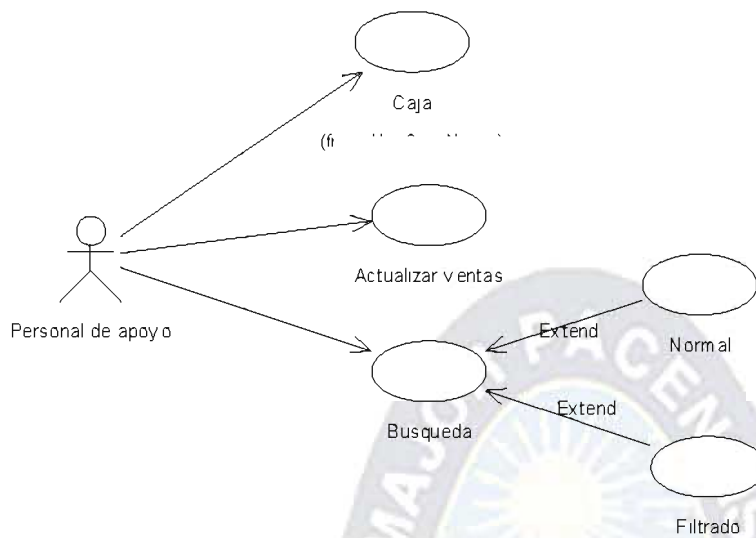


Figura 10: Diagrama de casos de uso, Fuente:UML en 24 horas[SCH-2001]

Los Diagrama de clases, presentan las clases, junto con sus atributos, operaciones, interfaces y relaciones. Los diagramas de clase corresponden al modelo de objetos, gráficamente son representados como se indica en la figura 11.

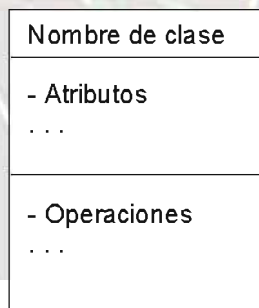


Figura 11: Diagrama de clases, Fuente: UML en 24 horas [SCH-2001]

Los diagramas de interacción, muestran la interacción entre objetos. Hay dos tipos de diagrama de interacción, ambos basados en la misma información, en nuestro caso se usara los diagramas de secuencia.

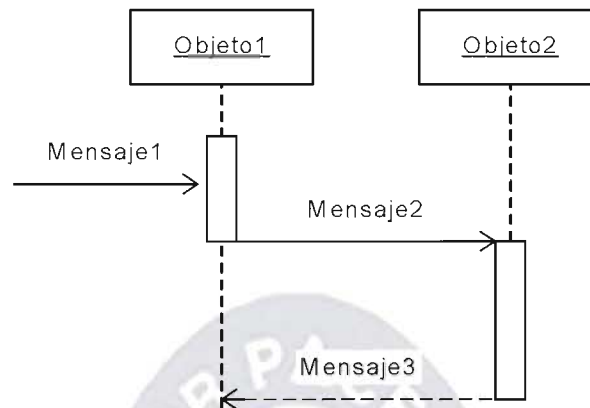


Figura 12: Diagramas de interacción, Fuente: UML en 24 horas [SCH-2001]

Un diagrama de Secuencia muestra una interacción ordenada según la secuencia temporal de eventos como se indica en la figura 12. En particular, muestra los objetos participantes en la interacción y los mensajes que intercambian ordenados según su secuencia en el tiempo. El eje vertical representa el tiempo, y en el eje horizontal se colocan los objetos y actores participantes en la interacción, sin un orden prefijado. Cada objeto o actor tiene una línea vertical, y los mensajes se representan mediante flechas entre los distintos objetos. El tiempo fluye de arriba abajo.

Un Diagrama de Estados muestra la secuencia de estados por los que pasa un objeto a lo largo de su vida, indicando qué eventos hacen que se pase de un estado a otro y cuáles son las respuestas y acciones que genera.

Un diagrama de estados se representa mediante un grafo cuyos nodos son estados y cuyos arcos dirigidos son transiciones etiquetadas con los nombres de los eventos.

Como se aprecia en la figura 13.

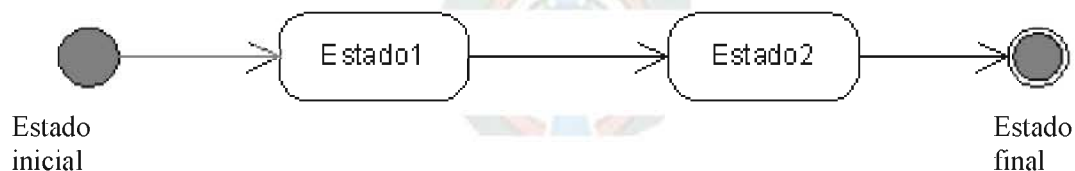


Figura 13: Diagrama de estados, Fuente: UML en 24 horas [SCH-2001]

Los diagramas de estados y los diagramas de secuencia formaran parte del modelo dinámico.

Los diagramas de flujo de datos no son parte UML, sin embargo el modelo funcional de OMT hace uso de los diagramas de flujo de datos para representar gráficamente el flujo de datos a través de los diferentes procesos y almacenes de datos hasta una salida a partir de los valores introducidos por las entidades externas. La notación se presenta en la figura 14.



Figura 14: Notación de los diagramas de flujo de datos, fuente: ingeniería del software [PRE-2002]

### 3.3. MODELO DE INVENTARIOS.

Se le llama inventario a las existencias de cualquier artículo, producto o recurso utilizado en una organización. Las empresas mantienen inventarios de materia prima o mercadería para la venta, los cuales sirven para satisfacer las demandas de los clientes.

El inventario se debe controlar de la mejor manera, es decir, se debe comparar lo que actualmente ocurre contra lo planeado. El control de los inventarios es una de las actividades más complejas, ya que hay que enfrentarse a intereses y consideraciones en conflicto por las múltiples incertidumbres que encierran. Puesto que estos inventarios representan frecuentemente una considerable inversión, las decisiones con respecto a las cantidades de inventario son muy importantes.

Dos sistemas de inventario muy utilizados son el sistema de pedido de tamaño fijo y el sistema de pedido de intervalo fijo. Se designa como sistema Q al sistema de pedido de tamaño fijo, mientras que el sistema de pedido de intervalo fijo se designa como sistema P. La diferencia básica entre los dos consiste en que en el sistema Q se pide



una cantidad fija a intervalos variables de tiempo y en el sistema P se ordena cantidad variable a intervalos fijos de tiempo.

Formulas para el sistema de inventarios.

Para determinar la cantidad pedida es:  $Q = \sqrt{\frac{2 C_2 \bar{D}}{C_3}}$

El tiempo entre pedido es (IP intervalo entre pedidos):  $IP = t = \frac{Q}{\bar{D}}$

Las existencias de seguridad (ES):  $ES = D_m - \bar{D}(L)$

Cantidad pedida = Q óptimo + existencias de seguridad - inventario disponible – unidades pedidas + demanda promedio en el tiempo de anticipación

El costo total por periodo planeado (habitualmente es anual) se calcula con la siguiente

ecuación:  $C = C_1 \bar{D} + C_2 \frac{\bar{D}}{Q} + C_3 \frac{Q}{2}$

Donde C1: es el costo de una unidad. C2: es el costo de hacer una compra.

C3: es el costo de almacenar.

Dm: es la demanda máxima.

D: es la demanda promedio.

t: tiempo entre promedios.

L: tiempo de anticipación.

ES: son las existencias de

seguridad.

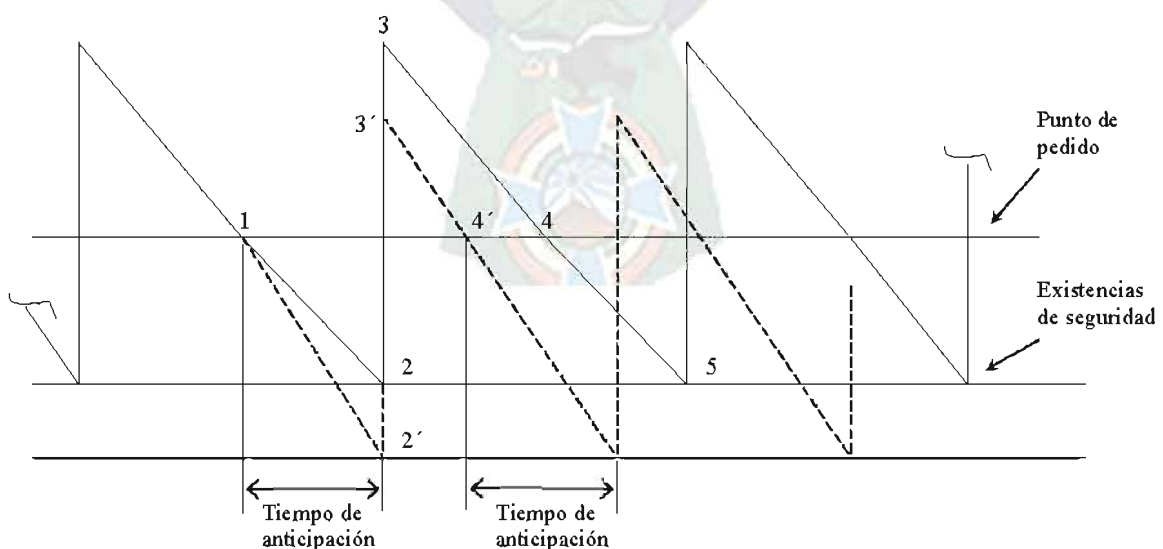


Figura 16: Sistema Q; Fuente: Taha [TAH 1996]

En la figura 15 se muestra un esquema del sistema Q. las líneas continuas (1-2-3-4-5) indican una tasa normal de demanda, las líneas punteadas (1-2'-3'-4'-5') indican una tasa máxima de demanda en el sistema el tiempo entre pedidos es menor con una tasa máxima de demanda que con una tasa normal de demanda, es decir aumenta el número de pedidos.

El costo total esperado por periodo planeado (generalmente un año) es el costo total por periodo planeado más el costo de almacenamiento de las existencias de seguridad:

$$C = C_1 \bar{D} + C_2 \frac{\bar{D}}{Q} + C_3 \frac{Q}{2} + C_4(ES)$$

Frecuentemente se presentan problemas con respecto a las variables que deben incluirse en la determinación de estos costos para ello se deben considerar los siguientes puntos de vista:

Componentes del costo  $C_1$ . Es el costo de compra de una unidad.

Costo de hacer una compra  $C_2$ . Constan de componentes tales como costos administrativos y de oficinas involucradas en el procesamiento de una compra, despacho y tramite del pedido y costo de transporte.

Costo de almacenamiento  $C_3$  algunos costos incluidos son; dinero inmovilizado, impuesto a los inventarios, costo de mantener registros de inventarios o deterioro de calidad.

Costos de déficit  $C_4$  Es el costo debido al retraso en satisfacer la demanda (no se considera el costo de ventas perdidas). La demanda se satisface después de un periodo de tiempo. Los costos de déficit incluye; requerimientos de tiempo extra ocasionados por el tiempo extra, costos por apresuramiento o los costos especiales de manipulación y costos de empaque.

### 3.4 TÉCNICAS DE MIGRACIÓN DE DATOS.-

En la actualidad Kautsch S.R.L. se enfrentan a la necesidad de adaptar sus aplicaciones informáticas del negocio, al funcionamiento que imponen las nuevas tecnologías y requerimientos del mercado a través de la migración de las aplicaciones a sistemas de administración de bases de datos y lenguajes de programación recientes para manejar eficaz y eficientemente los grandes volúmenes de datos que se almacenan en archivos planos.

Cuando se pretende realizar una migración se debe tener claro bajo que parámetros se va abordar la migración de datos, es así que en octubre de 2003, la comunidad económica europea publicó una serie de directrices aplicables para la realización de la migración a software de fuentes abiertas, plasmado en un documento titulado “The IDA Open source Migration Guides”, traducido al español en el 2004 por el Ministerio de las Administraciones Públicas, España [3]. Las directrices recomendadas son:

- Tener una idea clara de las razones de la migración antes de comenzar,
- Asegurarse de contar con apoyo activo para el cambio por parte del personal de Tecnologías de Información y los usuarios
- Asegurarse de que hay un defensor del cambio, mejor si está en el puesto más alto de la entidad;
- Adquirir experiencia y establecer relaciones con el movimiento software de fuente abierta
- Comenzar con sistemas no críticos
- Asegurarse de que cada etapa de la migración es factible.

Es probable que tanto los viejos como los nuevos sistemas tengan que funcionar “codo con codo” durante cierto tiempo. Es importante contar con una estrategia de transición que permita que ambos sistemas funcionen juntos, de manera que las actividades de producción se puedan continuar correctamente durante el período de transición. La sustitución de la máquina vieja puede llevar bastante tiempo (o no tener lugar), por lo que la coexistencia puede ser muy importante para ver mas detalles se puede ver el sitio de Directrices IDA de migración a software de fuentes abiertas [4].

**2.3.1. Metodología.-** Cualquier migración debería contener una fase de definición del proyecto y recopilación de datos, una justificación de la migración, incluido el costo asociado a la misma, una o más fases piloto. Durante la fase de definición del proyecto de recopilación de datos, deberían contemplarse al menos la descripción del conjunto de condiciones iniciales relevantes consistentes, un conjunto de condiciones finales con el mismo detalle, y una descripción de cómo llegar de las condiciones iniciales a las condiciones finales, que hará las veces de función de transformación entre condiciones iniciales y finales [4].

Se asume que no todos los caminos de migración estarán cubiertos por las Directrices, que deben ser consideradas como indicativas y no como normas, válidas en el punto de partida de la migración.

Para la migración de datos de Lotus 123 versión 5.0 a la base de datos, tomando en cuenta las directrices de la sección anterior, se plantean los siguientes pasos:

- 1.- Definir el entorno de migración del proyecto y la justificación del mismo.
  - 2.- Planificar la migración de los datos de acuerdo a la complejidad y cantidad de datos.
  - 3.- Realizar un análisis de los datos, observando la compatibilidad y validación con el esquema de la base de datos.
  - 4.- Definir una herramienta de apoyo (pívote) para la migración de los datos, soportado por el gestor de base de datos, generalmente se utiliza Excel o archivos ASCII.
  - 5.- Definir una herramienta de migración de datos, soportado por el administrador del servidor de base de datos.
  - 6.- Utilizar la herramienta de apoyo o pívote para estructurar y organizar la información de los archivos planos, de acuerdo al esquema de la base de dato.
  - 7.- Verificar y asegurarse de que los datos migrados son correctos de acuerdo con los archivos planos.
  - 8.- Realizar una o mas pruebas piloto para validar los datos migrados a la base de datos.
- De acuerdo a la validación de los datos y la duración de las pruebas piloto se establecerá un tiempo para la transición del sistema anterior al nuevo sistema.

### **3.5. CALIDAD DEL SOFTWARE.-**

No existe una definición estándar o universal de qué es calidad. En realidad algunos organismos e instituciones como ISO, IEEE o SEI brindan definiciones aceptables pero no son homogéneas, dando como resultado que cada profesional utilice su propia versión de calidad. Según Pressman [PRE-2002] pone énfasis en tres puntos importantes.

- \_ Los requisitos son la base de las medidas de calidad. La falta de concordancia con los requisitos es una falta de calidad.
- \_ Definición de estándares específicos en un conjunto de criterios de desarrollo que guían la manera en que se realiza la ingeniería del software.

Existe un conjunto de requisitos implícitos que a menudo no se nombran. Si el software cumple con los requisitos explícitos y no con los implícitos la calidad estará en duda.

La calidad del software es una compleja mezcla de factores que varían a través de diferentes aplicaciones y según los clientes que las pidan.

Los factores que afectan la calidad del software se pueden categorizar en dos amplios grupos: factores que se miden directamente y factores que se miden indirectamente.

En todos los casos debe aparecer la medición, comparar el software con un dato y llegar a una conclusión.

McCall propone una útil categorización de factores que afectan la calidad del software mostrado en la figura 16. Hace énfasis en tres aspectos muy importantes de un producto software: Características operativas, capacidad de cambios y adaptabilidad a nuevos entornos.

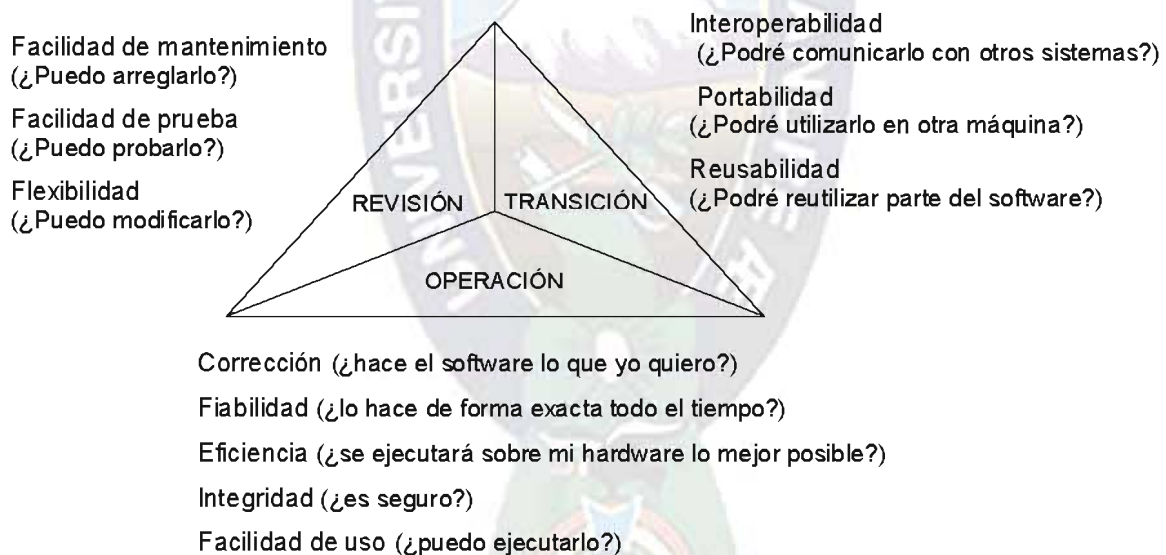


Figura 16: Modelo de McCall, Fuente: Pressman [PRE-2002]

Hewlett- Packard ha desarrollado un conjunto de factores de calidad del software al que se ha denominado FURPS<sup>20</sup>: Funcionalidad, facilidad de empleo, fiabilidad, rendimiento y capacidad de soporte, pueden usarse para establecer métricas de calidad para todas las actividades del proceso del software.

<sup>20</sup> Del ingles Functionality, Usability, Reliability, Performance, Supportability



Según la IEEE La calidad debe ser medible y predecible. Los factores deben ser: ausencia de defectos, satisfacción del usuario y conformidad con los requerimientos.

Sin lugar a dudas el factor inherente sobre calidad de software es la ausencia de defectos, este factor usualmente se expresa de dos maneras: tasa de defecto (número de defectos por KLOC o Puntos de función) y confiabilidad (número de fallas por n horas de operación, tiempo medio entre fallas, otra probabilidad libre de fallas por unidad de tiempo).

La satisfacción del usuario normalmente es medida en porcentaje de satisfacción o insatisfacción. Para evitar prejuicios se utiliza las técnicas de estudio o encuesta ciega (el entrevistador no sabe quién es el cliente, y el cliente no sabe a qué empresa representa el entrevistador).

Según la norma ISO 9126 las características son: portabilidad, eficiencia, confiabilidad, usabilidad, funcionalidad y mantenibilidad.

Algunas empresas definen sus propios factores/atributos de calidad de software, por ejemplo en el caso de IBM se enfoca hacia: Capacidad (funcionalidad), usabilidad, performance, confiabilidad, instalación, mantenibilidad, documentación/información, servicio y totalidad.

El presente proyecto utilizará los siguientes factores de calidad teniendo en cuenta los anteriores conceptos: La funcionalidad, confiabilidad, mantenibilidad y portabilidad.

**Funcionalidad.** La funcionalidad no se puede medir directamente, se debe derivar indirectamente mediante otras medidas directas, como los puntos de función, que fueron propuestos por primera vez por Albretch como métricas orientadas a la función. Los puntos de función se derivan con una relación empírica según las medidas directas del dominio de la información del software y las evaluaciones de la complejidad del software [PRE-2002].

Para calcular puntos de función PF se utiliza la siguiente relación:

$$PF = \text{cuenta total} * [0.65 + (0.01 * \sum F_i)]$$

Donde cuenta total es la suma total de las entradas obtenidas y  $F_i$  son valores de ajuste de la complejidad con  $i = 1, \dots, 14$ .



**Confiabilidad.** La confiabilidad de un sistema de cómputo es una propiedad que implica el grado de confianza esperado por parte del usuario en la operación adecuada del sistema al utilizarlo.

Para evaluar la confiabilidad se tomara en cuenta, que los sistemas de software no son entidades estáticas. Es posible caracterizar el comportamiento de la fiabilidad, estudiando el comportamiento de las fallas como tiempo de falla o periodo de vida de un sistema, medido a partir de un tiempo determinado hasta que se presenta una falla

El tiempo de falla se representa por la variable aleatoria continua  $T$  con función de probabilidad  $f(t)$ , donde  $f(t)$  es la distribución exponencial.

Para aplicar la distribución Exponencial a la teoría de la confiabilidad, se define la confiabilidad del sistema como la probabilidad de que funcionara correctamente durante un tiempo especificado bajo condiciones experimentales determinadas [PRO-1992]. Por lo tanto si  $R(t)$  se define como la confiabilidad del software dado en el tiempo  $t$ , se tiene:

$$\begin{aligned} R(t) &= P(T > t) \\ &= \int_t^{\infty} f(t) dt \\ &= 1 - F(t) \end{aligned}$$

Donde  $F(t)$  es la distribución acumulada de  $T$ , al resolver esta ecuación se tiene:

$$R(t) = e^{-\alpha t}$$

Donde  $\alpha$  es el margen de error y  $t$  el tiempo que se estima de confiabilidad.

La resolución de esta ecuación se puede ver en [PRO-1992].

**Mantenibilidad.** Todas las métricas del software presentadas, pueden usarse para el desarrollo de nuevo software y para el mantenimiento del existente, sin embargo se han propuestos métricas diseñadas explícitamente para actividades de mantenimiento.

El estándar IEEE982.1 sugiere un índice de madurez del software (IMS) que proporciona una indicación de la estabilidad de un producto software (basada en los cambios que ocurren en cada versión del producto). Se determina la siguiente información [PRE-2002]:

$M_t$  = número de módulos de la versión actual.

$F_c$  = número de módulos en la versión actual que se han cambiado.

$F_a$  = número de módulos en la función actual que se han añadido.

$F_d$  = número de módulos de la versión anterior que se han borrado en la versión actual.

El índice de madurez del software se calcula de la siguiente manera:

$$IMS = [ M_t - ( F_a + F_c + F_d ) ] / M_t$$

A medida que el IMS se aproxima a 1 el producto se empieza a estabilizar. El IMS puede emplearse también como métrica para la planificación de las actividades de mantenimiento del software

**Portabilidad.** La Portabilidad se define como la capacidad del software para ser trasladado de un entorno a otro o la capacidad de ejecutarse en diferentes plataformas sin cambios (abstracción). El entorno puede incluir entornos organizacionales, de hardware o de software

La portabilidad parece a todas luces completamente fuera del alcance de la etapa de diseño, como atributo mensurable de calidad, dado que el diseño debe mantenerse conceptualmente separado del entorno de implementación final; aún así no siendo esto más que una generalización muy discutible tampoco hemos encontrado suficiente evidencia como para apoyar que este atributo o característica es importante, desde el punto de vista de la calidad, durante etapas tempranas de diseño [4].

Según lo mencionado en el párrafo anterior la portabilidad es como un requisito más que puede o no ser cumplido. Actualmente existe una gran cantidad de sistemas operativos, es así que la sociedad exige que el software se ejecute desde cualquier entorno lo que implica que debe ser portable.

## MARCO PRÁCTICO

Desarrollo del software

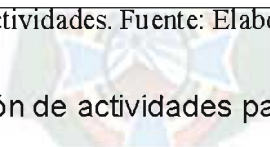


Figura 17. Planificación de actividades. Fuente: Elaboración propia

La figura 17 muestra la planificación de actividades para el presente proyecto.

La fase de análisis es el primer paso de la metodología que se refiere a la obtención de un modelo preciso, conciso, comprensible y correcto del mundo real.

**4.1.1. Análisis y especificación de requerimientos.-** El análisis de la institución se realizó en detalle en el capítulo 2 Marco Institucional, donde se especifica las actividades de la empresa y los procesos que se presentan dentro la institución en detalle.

Una vez identificados los procesos y el funcionamiento de la empresa, se elabora la especificación de requerimientos o la definición del problema de la institución, definiendo lo que se va hacer y no en como se lo hará, como se explica a continuación:

Desarrollar un sistema de gestión de inventarios para Kautsch S.R.L. que permitirá llevar el control y hacer un seguimiento de la información de la mercadería que posee. La mercadería con la que cuenta esta alrededor de los ocho mil ítems, utiliza una codificación numérica, que va desde el 100000 hasta el 999999, ordenado alfabéticamente según el tipo de mercadería o la marca. La mercadería esta clasificado en 31 archivos de Lotus (similar a una hoja Excel), la dirección ha decidido asignar la letra Y por delante al nombre del archivo que corresponde a la marca o al tipo de mercadería por ejemplo la marca Stanley tiene la siguiente codificación del 680000 al 680999 y corresponde al archivo ystanley.

Toda la información es estructurada, y clasificada de acuerdo al tipo de mercadería y la marca del proveedor.

Los datos de interés para la institución acerca de las importaciones son: fecha, cantidad, país de origen, línea o marca, lo que permite tener un historial de las importaciones.

Una vez realizado la importación de artículos, la información debe ser actualizada y registrada, si la mercadería es nueva, los datos del nuevo proveedor deben ser actualizados. La importación de artículos, implica determinar el nuevo precio de venta (en relación al dólar), tomando en cuenta el costo de importación, impuestos de ley, sueldos y otros gastos.

El cambio de precio estará disponible por marca o sección (corresponde a un archivo o categoría), teniendo en cuenta que las importaciones se realizan por marcas o líneas.

La búsqueda de artículos es muy importante y será realizado no solo en base al código, si no también por el nombre del artículo, marca, rango de códigos o en una combinación de las anteriores, que permitirá obtener información acertada de lo que se busca.

El registro de facturas y la actualización de ventas tendrán una interfaz de usuario similar al de una factura, en el cual se ingresan datos del cliente, vendedor, código de artículo y cantidad de compra, el sistema mostrará descripción, precio del o los artículos lo que facilitará la verificación de la factura, en cuanto al costo parcial y total de la mercadería vendida.

El sistema permitirá la elaboración de proformas, según el tipo de cliente y si la cantidad y monto de la mercadería es considerable, el cliente obtendrá un descuento. La elaboración de proformas registrará los siguientes datos: nombre del cliente, fecha y número de proforma y si el cliente accede al descuento, este será registrado según la marca del artículo y se tendrá la opción de guardar la Proforma en una hoja electrónica de Excel.

Para apoyar la toma de decisiones se plantean reportes acerca de ventas e importaciones de mercadería. Los reportes de ventas mostraran información de ventas según factura, fecha, artículo. Los reportes de importación se realizaran según la marca y el producto.

El acceso al sistema será de acuerdo a los siguientes roles como indica en la figura 18 que sigue a continuación.

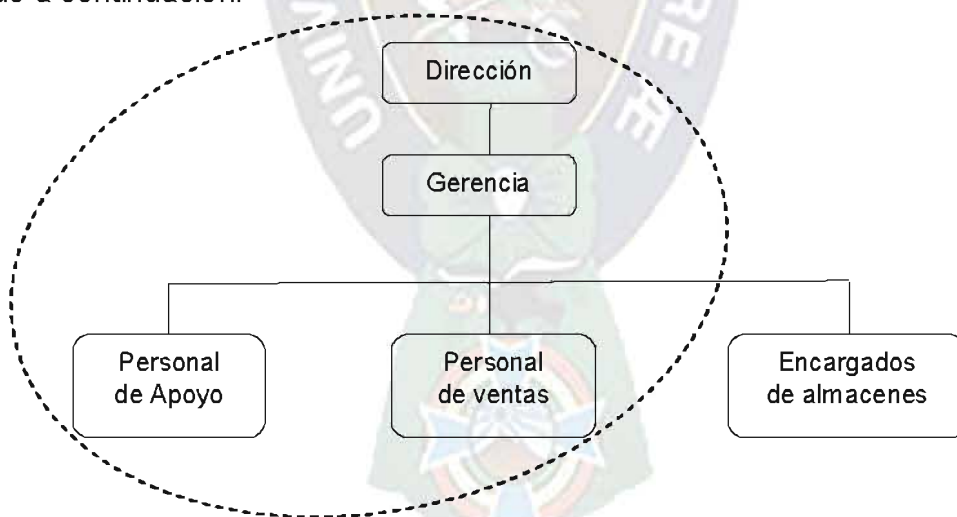


Figura18: Usuarios del sistema, Fuente: elaboración propia

Dirección y gerencia: se encarga de realizar altas, bajas, modificaciones a la mercadería, proveedor y los pedidos, también tiene acceso completo a las búsquedas y reportes.

Personal de apoyo: se encarga de actualizar las ventas, verificar el registro de la facturación y accede a las búsquedas y reportes.

Personal de ventas: el usuario más restringido, solo puede realizar la elaboración de proformas y visualizar los reportes.

El modelo de análisis se encarga de modelar las abstracciones del mundo real, en el dominio del problema. La metodología OMT recurre a los diagramas de clases, los diagramas de transición de estados y los diagramas de casos de uso para realizar la abstracción del entorno del negocio este último no es parte de OMT.

**4.1.2. Modelo conceptual.** Los diagramas de clases sirven para representar el modelo de objetos, donde se considera las entidades u objetos que se presentan en el dominio de la definición del problema. Los objetos se obtienen a partir de un análisis gramatical de la definición del problema para identificar los posibles candidatos a clase, a continuación se tiene los siguientes candidatos:

Sistema, control, seguimiento, mercadería, marca, archivo, fecha, cantidad, país de origen, importación, precio de venta, impuestos, cliente, vendedor, factura, proforma, descuento y reportes. La tabla 7 presenta un resumen de lo mencionado.

Sistema	Referencia al sistema	Precio	Atributo del sistema
Control	Función de sistema	Moneda	Candidato a clase
Seguimiento	Función del sistema	Impuestos	No corresponde al sistema
Mercadería	Candidato a clase	Cliente	Candidato a clase
Marca	Candidato a clase	Vendedor	Candidato a clase
Archivo	Nombre genérico	Categoría	Candidato a clase
fecha	Atributo del sistema	Búsqueda	Operación del sistema
Cantidad	Atributo del sistema	factura	Candidato a clase
País	Atributo del sistema	ventas	Candidato a clase
Importación	Candidato a clase	Proforma	Candidato a clase

Tabla 7. Selección de clases, Fuente Elaboración propia



Los candidatos a clase son:

Mercadería: hace referencia a los artículos de la empresa.

Marca: se denominará proveedor por estar más acorde, al sistema.

Archivo: se considera mejor categoría para evitar confusiones de términos, esta clase almacenara información acerca de la clasificación o tipo de mercadería.

La figura 18 muestra el modelo conceptual de las clases candidatas y las relaciones que existen entre ellas.

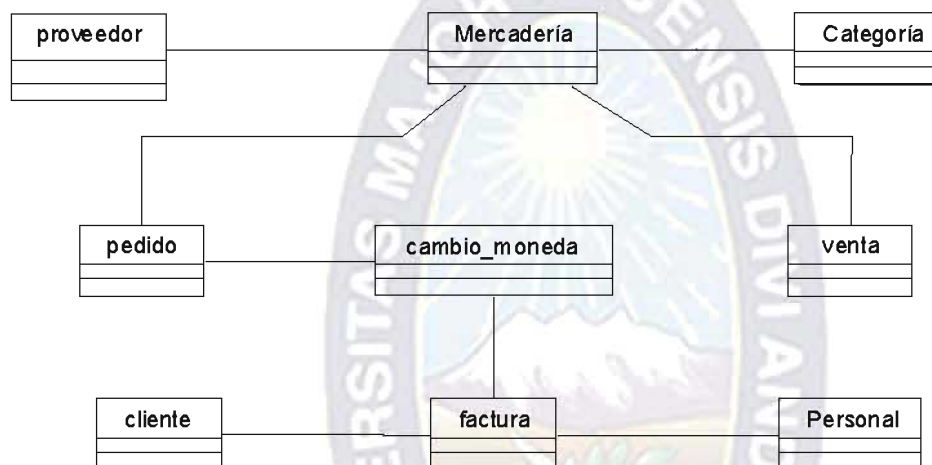


Figura 19. Modelo conceptual, Fuente: Elaboración propia

Una vez determinado el modelo conceptual, se establece la multiplicidad de las asociaciones entre las clases y se identifican los atributos que corresponden a las mismas.

Mercadería: características de los diferentes artículos.

Código de mercadería, identificador del artículo asignado por la empresa.

Artículo, nombre del producto.

Descripción, descripción del artículo.

Cantidad, existencia del artículo en almacenes.

Precio venta, precio del artículo, se encuentra en dólares.

Unidad, indica la unidad del artículo por ejemplo pieza, metro, caja y otros.

Proveedor: clase que almacena datos acerca del proveedor.

Marca, nombre de la línea o marca del artículo,

Sitio web, indica el sitio web del proveedor si existe.

Email, correo electrónico del proveedor.

Descripción, datos complementarios del proveedor

Categoría: Clase que almacena la subclasificación de la mercadería

Archivo, comprende el nombre de la subclasificación de la mercadería.

Categoría, nombre de la categoría o tipo de mercadería.

Descripción, datos complementarios de categoría.

Pedido: Clase que almacena datos de la importación de la mercadería

Nro de pedido, indica el número secuencial de pedido o importación de un artículo.

Fecha, indica la fecha de importación o pedido.

Cantidad, indica la cantidad de importación de un artículo determinado.

Costo, indica el costo de importación de la mercadería hasta la llegada al país.

País, indica el país de procedencia de la mercadería

Ciudad, indica la ciudad del país de procedencia.

Moneda, registro la moneda en la que se realizó la importación (dólar o euro).

Venta: Clase que almacena las ventas de los artículos en detalle

Cantidad de venta, indica la cantidad vendida de un artículo.

Precio, indica el precio total del o los artículo(s) vendido(s) en moneda nacional.

Código del artículo: indica el código del artículo vendido.

Nro de factura, Corresponde al número de factura que corresponde esta venta.

Factura, Clase que almacena las ventas totales de una factura

Nro De factura, indica la numeración de la factura.

Fecha, indica la fecha de emisión de factura.

Precio total, indica el monto total de la factura en moneda nacional.

Descuento, indica si hubo descuento en la factura.

Tipo de pago, indica si se pago en bolivianos, dólares o con tarjeta

Personal: Clase que almacena datos del personal de la empresa

Nombre, indica el o los nombres del personal de la empresa.

Ap\_paterno, indica el apellido paterno del personal de la empresa.

Ap\_materno, indica el apellido materno del personal de la empresa.

Cargo, indica el cargo que ocupa en la empresa

Cliente: Clase que almacena datos del cliente.

Nit\_ci, indica el nit o ci del cliente.

Razón social, indica el nombre del cliente que puede ser una persona o empresa.

Descripción, indica una breve descripción del cliente.

Teléfono, indica el teléfono del cliente si es que tiene.

En la figura 19 se presenta el diagrama de clases donde se identifica la multiplicidad y las asociaciones que existe entre las clases. También se puede apreciar la relación que existe entre mercadería y proveedor que será representado por la clase provee, que cuenta con el atributo de estado.

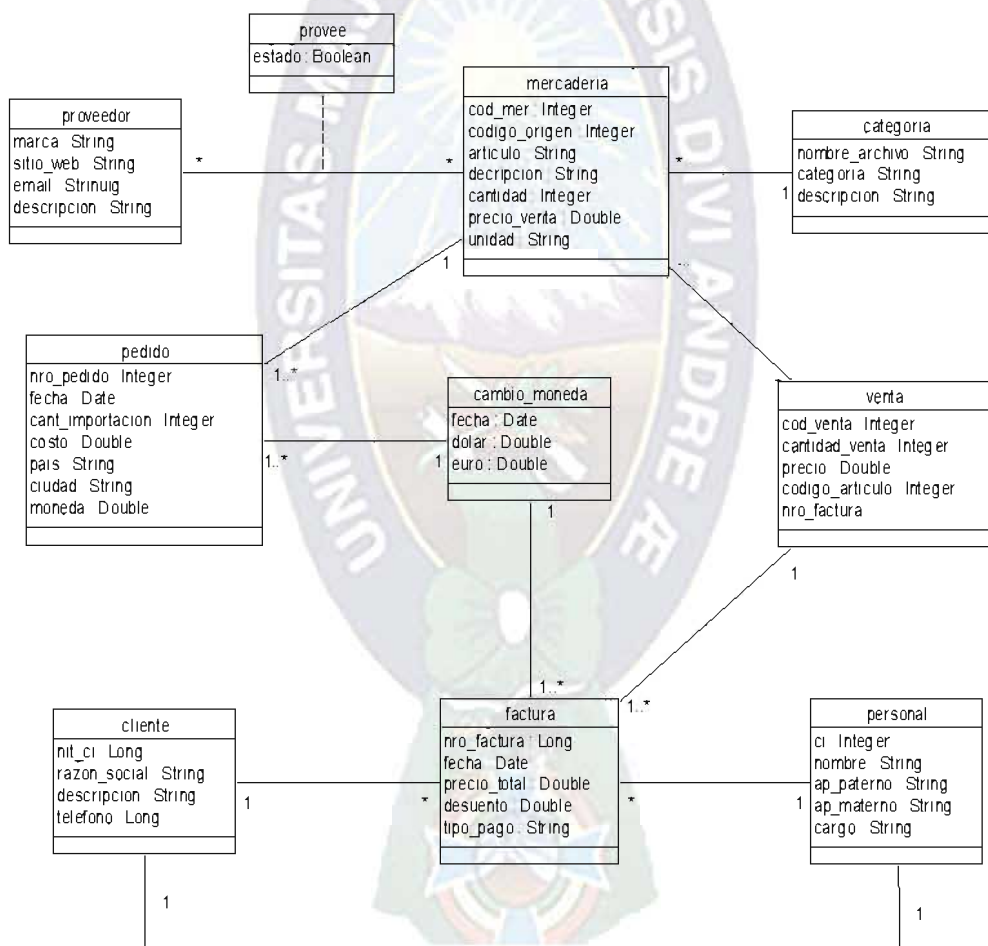


Figura 20. Modelo de clases Fuente: Elaboración propia

**4.1.3 Modelo dinámico.** El modelo dinámico tiene que ver con el cambio de estado de un objeto en el tiempo y el intercambio de mensajes entre objetos.

Para la construcción del modelo dinámico se inicia identificando los escenarios para ver la interacción entre el usuario y el sistema. Los escenarios muestran las interacciones

principales, como los formatos de visualización y el intercambio de información. La visualización se refiere a las interfaces de usuario, el sistema de gestión de inventarios se tiene las siguientes interfaces:



Figura 21. Validación del usuario, Fuente: Elaboración propia

La primera presenta la validación del usuario para acceder al sistema, como se muestra en la figura 21. Una vez dado de alta al usuario, accede al menú principal del sistema, donde se encuentran las operaciones disponibles, la búsqueda de mercadería y la generación de reportes como se aprecia en la figura 22.



Figura 22. Menú principal, Fuente: Elaboración propia

Las operaciones del sistema son: altas, bajas y modificaciones de la mercadería, proveedor, pedido, personal y cliente. Las consultas son: la Búsqueda de información y la generación de reportes. En la pestaña inventario se presenta la opción pedido por demanda, que genera la cantidad óptima de pedido de un artículo según las ventas del mismo.

La figura 23 presenta el formato de las diferentes ventanas, que tiene las operaciones adicionar, modificar, eliminar, mostrar, limpiar y cancelar, la ventana mostrada presenta la interfaz de usuario de proveedor.

La figura 24 muestra el formato de las ventanas de búsqueda, tiene la opción de buscar palabras completas o parte de ellas.

En todas las ventanas se deshabilitan las opciones que no están disponibles, esto para facilitar la operación del sistema, como se aprecia en las figuras 23 y 24.

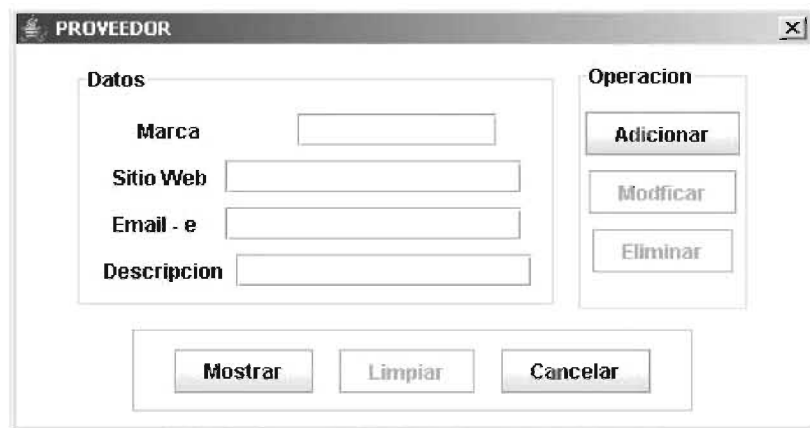


Figura 23. Ventana Proveedor, Fuente: Elaboración propia

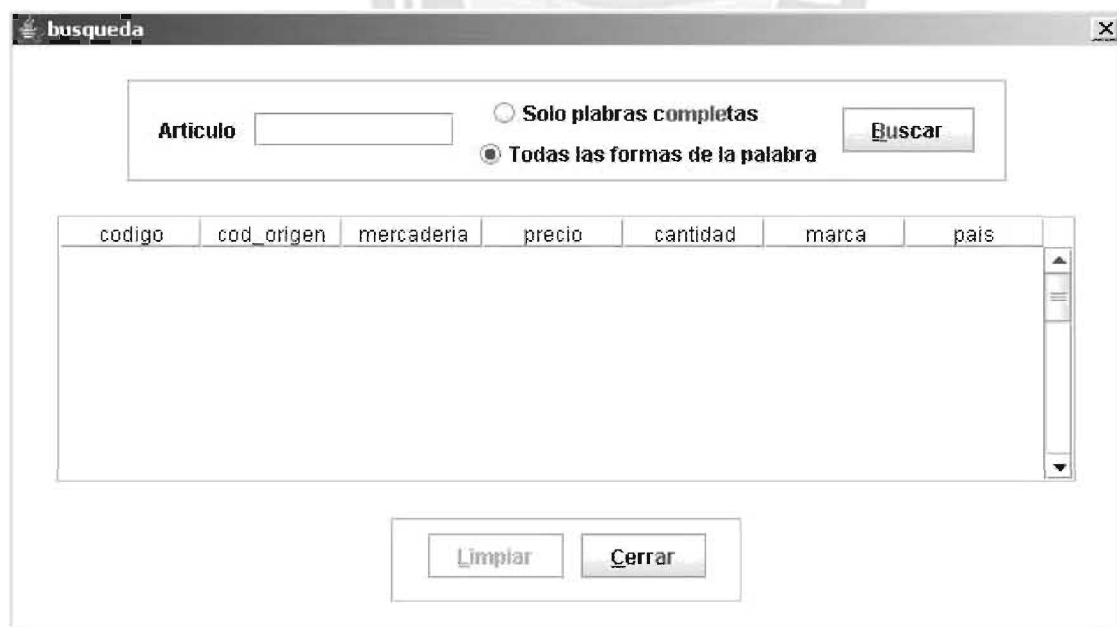


Figura 24. Ventana de búsqueda, Fuente: Elaboración propia.

Estas ventanas tendrán una clase generalizada llamada ventana que hereda las características de la clase JFrame de java, como se muestra en la figura 25, de esta



clase se hereda la funcionalidad y las operaciones de las diferentes ventanas como proveedor y búsqueda. Es decir la clase ventana es la generalización de los formularios de la interfaz de usuario.

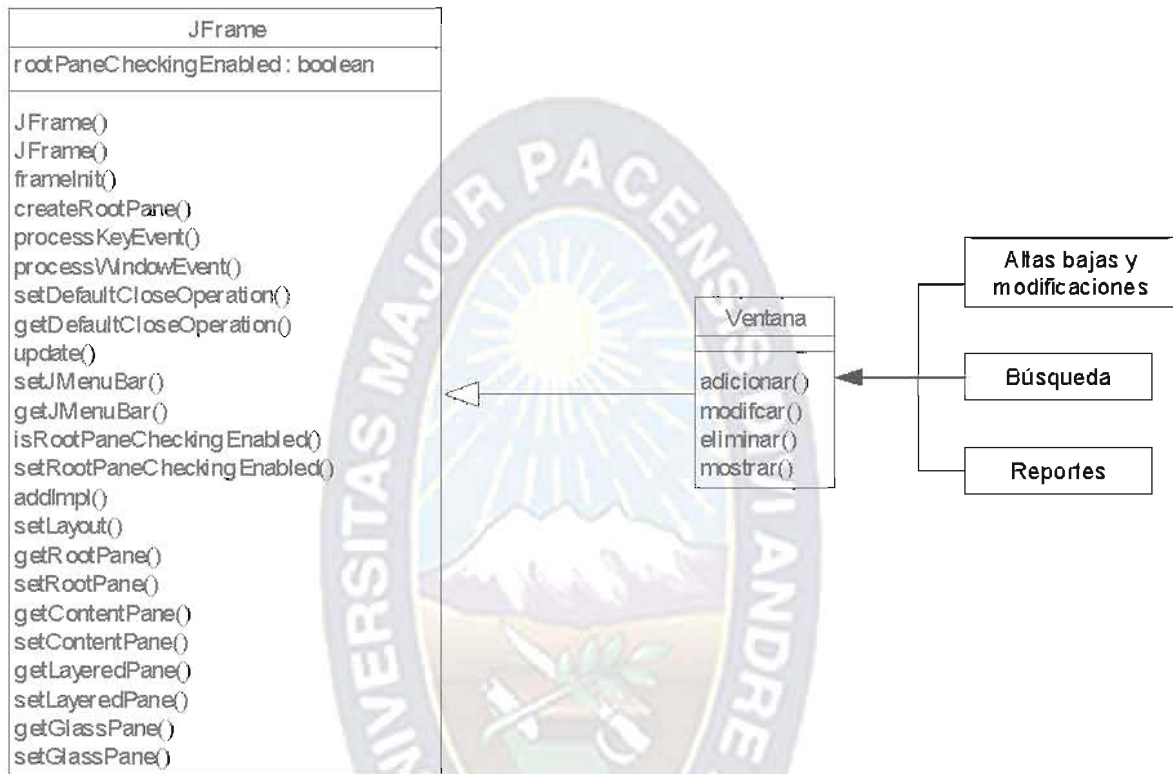


Figura 25. Herencia de la Interfaz gráfica de usuario, Fuente: Elaboración propia.

Una vez identificadas las clases y los escenarios donde se desenvuelven los objetos se describe la interacción entre ellas.

Para representar el intercambio de información de las diferentes clases se recurre a los diagramas de secuencia, que muestran el paso de mensajes entre los diferentes objetos (que son interfaces de usuario, estructuras de datos u operaciones del sistema). Este es un paso muy importante ya que en base a esta funcionalidad se define posteriormente la arquitectura del sistema.

En la siguiente página se describe el diagrama de secuencia, para la importación de mercadería.

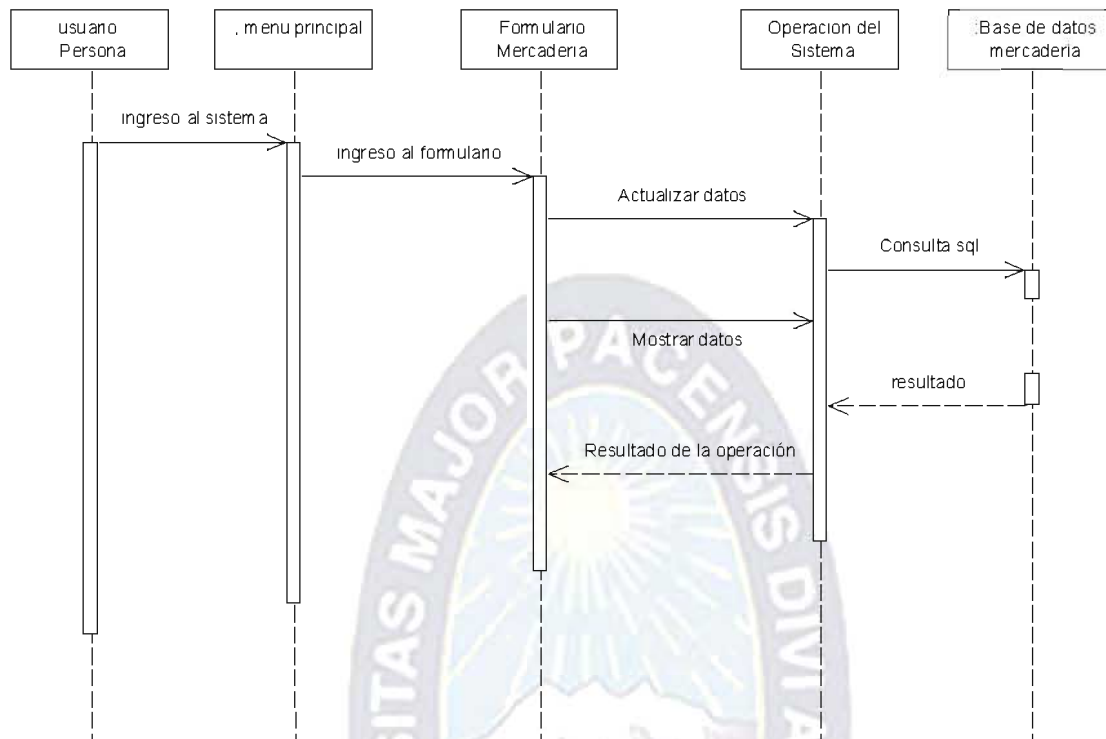


Figura 26. Diagrama de secuencia ingresar mercadería, Fuente: Elaboración propia.

La figura 26 ilustra el diagrama de secuencia para realizar la importación de mercadería. Se inicia la secuencia con un usuario ya registrado que ingresa al sistema, el usuario selecciona el formulario de mercadería del menú principal y registra o actualiza la mercadería dependiendo si el producto es nuevo, mediante la ejecución de una consulta SQL a la base de datos.

Para modelar el comportamiento de los objetos, se utiliza los Diagramas de Transición de Estados, estos modelan el comportamiento de las clases que tienen un comportamiento dinámico importante, ya que no es necesario elaborar diagramas de transición de estados para todas las clases.

La figura 27 muestra el diagrama de transición de estados para la clase mercadería.



Figura 27 Diagrama de transición de estados de la clase mercadería, Fuente: Elaboración propia.

**4.1.4 Diagrama de Casos de Uso.** La metodología de Rumbaugh utiliza los diagramas de flujo de datos para representar el modelo funcional que es parte del análisis estructurado.

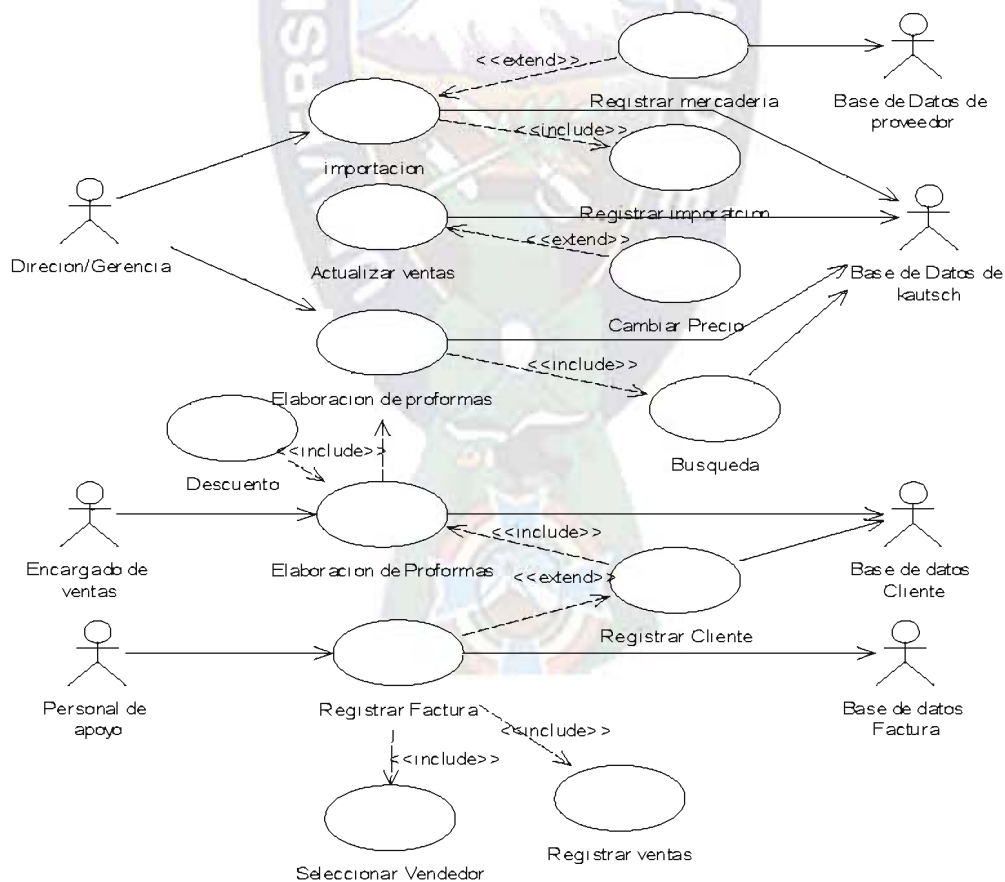


Figura 28. Diagrama de casos de uso del sistema, Fuente: Elaboración propia.

El modelo funcional de Rumbaugh no sigue los principios del Análisis Orientado a Objetos, esta la razón por la que no se toma en cuenta este modelo. Una alternativa para modelar los procesos son los diagramas de casos de uso, que representan gráficamente la relación entre el usuario y los casos de uso del sistema. La figura 28 muestra los diagramas de casos del sistema, una parte fundamental es la descripción textual detallada los actores y casos de uso identificados.

Actor	Dirección o gerencia
Casos de uso	Importación y elaboración de proformas
Tipo	Primario
Descripción	Persona que se encarga de realizar la importación y actualización de las ventas goza de altos privilegios en el uso del sistema.

Tabla 8. Actor: dirección y/o gerencia, Fuente: Elaboración propia.

Caso de uso	Importación
Actores	Dirección y/o gerencia y base de datos mercadería
Tipo	Inclusión
Propósito	Registrar la importación y actualización de la mercadería
Resumen	El usuario inicia este casos de uso tiene diversas opciones para realizar el registro de la importación.
Precondiciones	Se requiere haber validado correctamente al usuario.
Flujo principal	El usuario selecciona la opción pedido del menú principal En el formulario de pedido se registra la importación de la mercadería (E1, E2) Después de registrar el pedido, se vuelve al menú principal y se selecciona la opción mercadería y se actualiza los nuevos datos (E1, E2). Si el artículo es nuevo se crea un nuevo registro de mercadería y se registra también al proveedor (E1, E2).
Subflujos	Ninguno.
Excepciones	E1. Información incompleta: falta llenar información en el formulario, se vuelve a solicitar al usuario que complete el formulario E2. Registro ya existente: Se pide que ingrese un nuevo código o identificador del registro o que lo modifique

Tabla 8. Caso de Uso: Importación de mercadería, Fuente: Elaboración propia.

La tabla 8 muestra la descripción textual del actor Dirección y/o gerencia y la tabla 9 la descripción del caso de uso importación de mercadería.

#### 4.2. DISEÑO DE SISTEMAS.

En esta fase se realiza la descomposición del sistema en subsistemas o módulos, la representación de los módulos se hace mediante paquetes de UML. La figura 29 muestra la división del sistema en subsistemas o módulos.

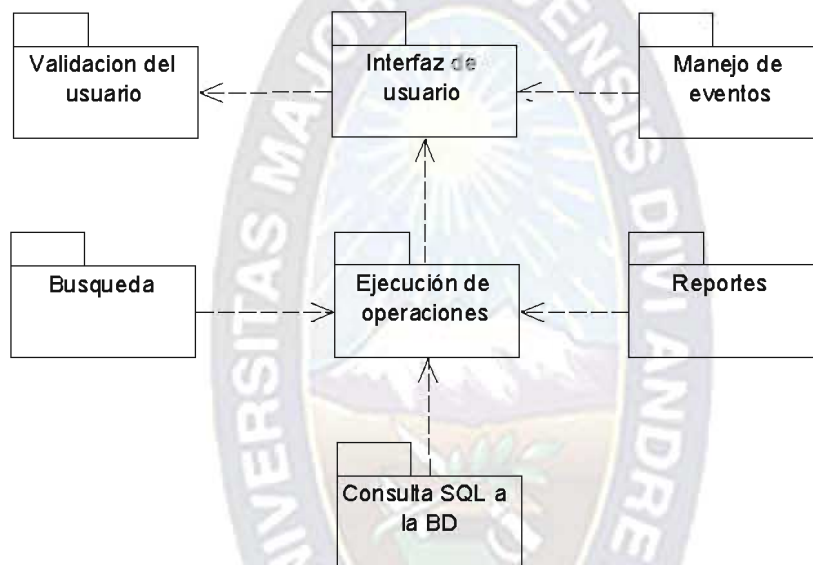


Figura 29. División del sistema en subsistemas, Fuente: Elaboración propia.

A continuación se da una breve descripción de los subsistemas identificados:

*Validación del usuario.* Este módulo permite dar de alta a un usuario (dirección o gerencia, personal de apoyo y encargado de ventas) y así realizar la conexión a la base de datos. El usuario habilitado inicia el nuevo día y registra el tipo de cambio de la moneda extranjera (dólar y euro) frente a la moneda nacional.

*Interfaz de usuario.* Una vez registrado el usuario, accede al menú principal (ver figura 22), donde se tiene a disposición diferentes operaciones o consultas al sistema, cada opción del menú principal presenta una ventana que sirve de interfaz entre el usuario y las operaciones del sistema sobre diferentes objetos.

*Manejo de eventos.* Este módulo brinda apoyo a la interfaz de usuario, para llevar el control de los diferentes eventos que se invocan en las operaciones del sistema.



*Ejecución de operaciones.* Módulo que se encarga de llevar el control de todas las operaciones solicitadas a través de las ventanas de la interfaz de usuario, como la inserción, modificación, eliminación de algún registro o mostrar datos.

*Búsqueda.* Módulo que permite recuperar información de acuerdo a diferentes parámetros, como la búsqueda de un artículo por nombre, en un rango de códigos o una opción de las anteriores.

*Reportes.* Módulo encargado de representar reportes, los reportes pueden ser de dos tipos de ventas e importaciones. Los reportes de las ventas pueden ser por artículo, fecha, o la combinación de las anteriores y las importaciones pueden ser por artículos o por fecha.

*Consulta SQL a la base de datos.* Este módulo se encarga de realizar las operaciones solicitadas a través de las interfaces de usuario, sobre la base de datos, para ello utiliza la correspondencia de objetos a tablas de la base de datos, se explica este en mayor detalle en la siguiente sección.

Cuando se realiza la división del sistema en módulos, se busca un acoplamiento bajo y se promueve el encapsulamiento, para reducir el esfuerzo en el mantenimiento y en las pruebas.

**4.2.1 Arquitectura del sistema.** La arquitectura presentada en el presente proyecto es una arquitectura basada en capas, como se muestra en la figura 29, se tiene la capa interfaz gráfica de usuario, apoyada sobre la capa de control del sistema, cuenta con los módulos de validación del usuario, interfaz de usuario y el manejo de eventos.

El control del sistema presenta toda la funcionalidad y el control entre la interfaz gráfica del usuario y la base de datos, cuenta con los módulos de ejecución de operaciones, búsqueda y reportes.

La capa de modelo del sistema. Presenta la correspondencia del modelo de objetos al modelo objeto-relacional de la base de datos y ejecuta diferentes operaciones del lenguaje de consulta a la base de datos. Cuenta con el modulo de consulta SQL a la base de datos.

La base de datos esta representada por tablas, que son la correspondencia directa de los objetos o clases del diagrama de clases.



Figura 29. Arquitectura del sistema basada en capas, Fuente: Elaboración propia.

### 4.3. DISEÑO DE OBJETOS.

En esta fase diseño se establece los detalles de la implementación, del modelo de objetos, las asociaciones y los algoritmos que llevan adelante las operaciones o métodos de acuerdo al lenguaje de programación orientado a objetos.

**4.3.1. Diagrama de clases.** Al modelo conceptual definido en el modelo de análisis, se agrega los detalles de la implementación de los atributos y operaciones. Las operaciones se identifican, de los diagramas de secuencia, los diagramas de transición de estados y los diagramas de casos de uso, efectuando la correspondencia a una clase del modelo conceptual.

La figura 29 muestra el diagrama de clases detallado, especificando los atributos con el tipo de dato que le corresponde y se define las operaciones correspondientes a cada una de las clases.

En el diagrama de clases de la figura 29 se muestra el primer atributo como identificador de la clase. Por ejemplo en el caso de mercadería se establece cod\_mer como atributo identificador, en el caso de la clase venta se asigna un identificador denominado cod\_venta.

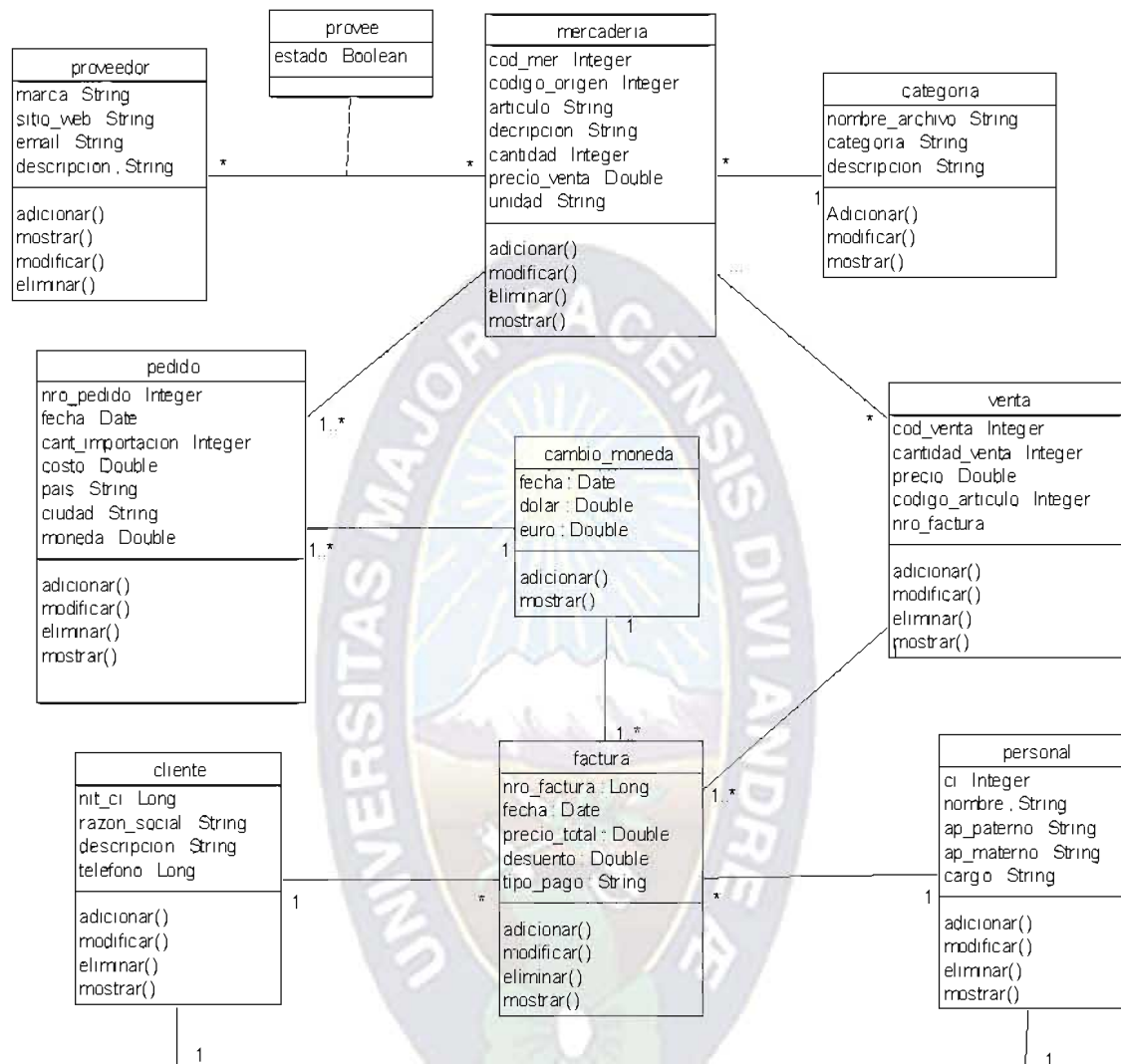


Figura 30. Diagrama de clases detallado, Fuente: Elaboración propia.

**4.3.2. Modelo de Base de Datos Objeto Relacional.** La correspondencia del diagrama de clases se realiza de la siguiente manera: Cada objeto del diagrama de clases se convierte en una tabla de la base de datos, como se muestra en la figura 31.

Los atributos subrayados, son las llaves primarias, de cada tabla.

La navegación de la base de datos se realiza a través de las llaves externas, en la figura 31 se puede apreciar la navegación de acuerdo al sentido de las flechas, la llave extranjera es el último o los últimos atributos como en la tabla factura.

La llave extranjera viene a ser la o las llave(s) primaria(s) de la tabla de donde parte la flecha.

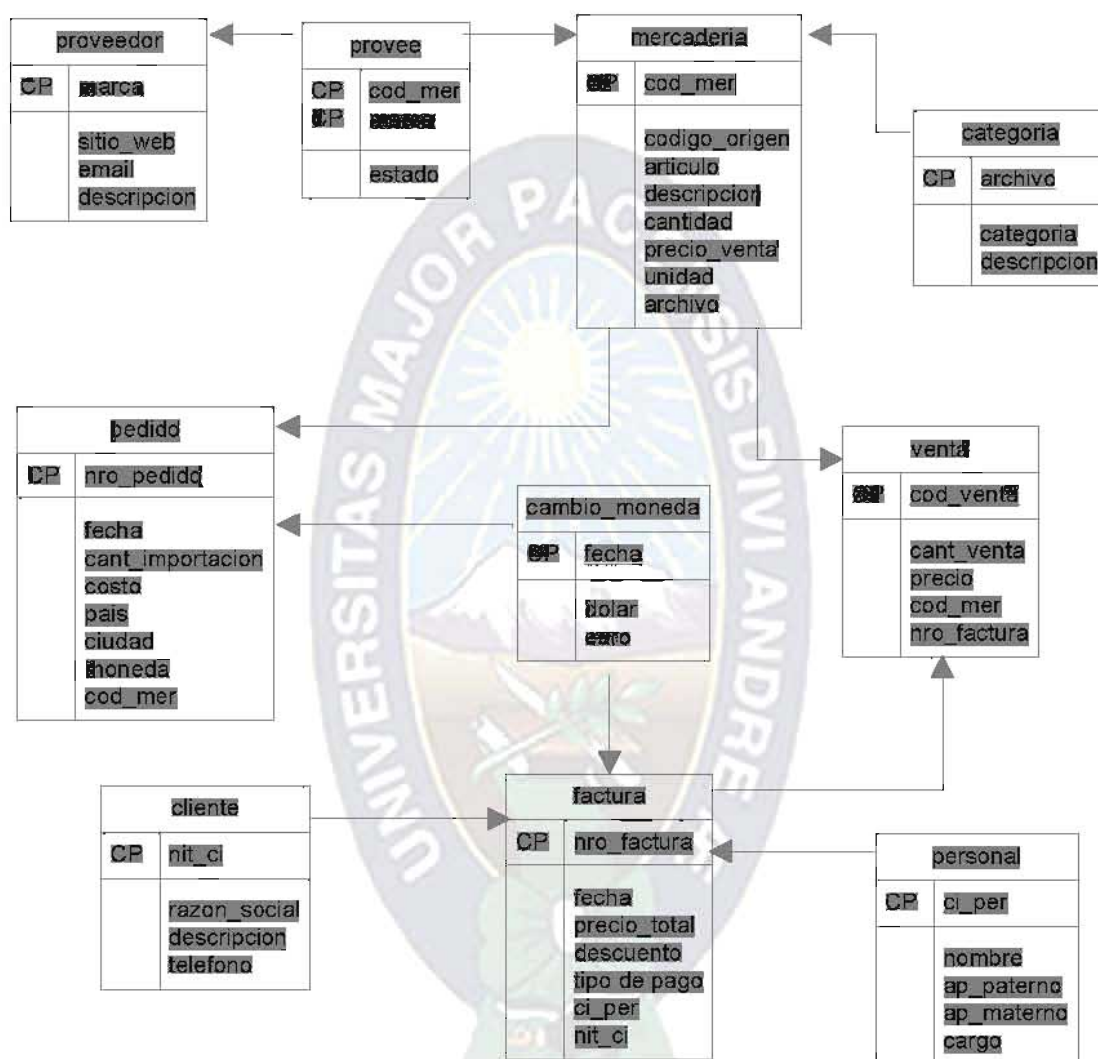


Figura 31. Diagrama Entidad Relación, Fuente: Elaboración propia.

Ya teniendo el modelo conceptual se define el esquema de la base de datos en SQL, como se muestra a continuación el esquema de la tabla mercadería:

```
CREATE TABLE mercaderia (
  cod_mer      int(11) NOT NULL,
  codigo_origen varchar(20) default NULL,
```

```

articulo    varchar(20) NOT NULL,
descripcion varchar(50) default NULL,
cantidad    int(7) NOT NULL,
precio_venta double(8,5) NOT NULL,
unidad      varchar(5) NOT NULL,
archivo     varchar(20) NOT NULL,
PRIMARY KEY (cod_mer),
CONSTRAINT mer_fk FOREIGN KEY (archivo) REFERENCES categoria (archivo)
);

```

**4.3.3. Diseño Procedimental.** A continuación se presenta el diseño Procedimental de uno de los módulos más importantes del sistema, la actualización de Mercadería.

```

Definir paquete kautsch.Interfaz;
Importar biblioteca de clases a utilizar
class MercaderiaFrm extends Ventana{
    definir etiquetas
    definir campos de texto
    definir botones //adicionar, modificar, eliminar, mostrar, limpiar y cerrar
    definir objetos MercaderiaBD, Mercaderia
    constructor MercaderiaFrm(ventanaActual, usuario, contraseña) {
        super(ventanaActual, true);
        asignar texto a las etiquetas
        definir tamaño de los campos de texto
        asignar texto a los botones
        definir la distribución del contenedor de la ventana
        asignar etiquetas al contenedor de la ventana
        asignar campos de texto al contenedor de la ventana
        asignar botones al contenedor de la ventana
        invocar visualizar ventana_contenedor
        invocar leer eventos_de_los_botones
    }
    método leer_evento_adicionar(){

```



```

    leer campos de texto en el objeto mercadería
    mercaderia.getDatos
    invocar mercaderiaBD.Adicionar(mercaderia, usuario, contraseña)
}
método leer_evento_modificar(){
    leer campos de texto en el objeto mercaderia
    mercaderia.getDatos
    invocar mercaderiaBD.modificar(mercaderia, usuario, contraseña)
}
método leer_evento_eliminar(){
    leer campo de texto del identificador del objeto en el objeto mercaderia // cod_mer
    mercaderia.getDatos
    invocar mercaderiaBD.eliminar(identificador, usuario, contraseña)
}
método leer_evento_Mostrar(){
    leer campo de texto del identificador del objeto en el objeto mercaderia // cod_mer
    mercaderia = mercaderiaBD.mostrar(identificador, usuario, contraseña)
    campos de texto = mercaderia.getDatos()
}
} //fin de la clase mercaderiaFrm

```

### Diseño Procedimental de la clase MercaderiaBD

Definir paquete kautsch.Modelo;

Importar biblioteca de clases a utilizar

class MercaderiaBD extends ConexionBD

```

{
    definir objeto Mercaderia
    definir variable usuario
    definir variable contraseña
    constructor MercaderiaBD(usuario, contraseña){
        this.usuario = usuario
        this.contraseña = contraseña
    }
    método adicionar(mercaderia) {

```

```

    invocar this.abrirConexion()
    prepararSentencia("INSERT INTO Mercaderia VALUES (parm1, parm2, ..., parmN)")
    prepararSentencia .setParametros(mercaderia.getDatos)
    prepararSentencia.ejecutarConsultaSQL()
    invocar this.cerrarConexion()
}

metodo modificar(mercaderia) {
    invocar this.abrirConexion()
    prepararSentencia("UPDATE mercaderia set parm1, parm2, ..., parmN")
    prepararSentencia .setParametros(mercaderia.getDatos)
    prepararSentencia.ejecutarConsultaSQL()
    invocar this.cerrarConexion()
}

metodo eliminar (codigo) {
    invocar this.abrirConexion()
    prepararSentencia("DELETE FROM mercaderia WHERE cod_mer = codigo")
    prepararSentencia.ejecutarConsultaSQL()
    invocar this.cerrarConexion()
}

metodo mostrar (codigo) { //metodo de tipo mercaderia
    invocar this.abrirConexion()
    prepararSentencia("SELECT * FROM mercaderia WHERE cod_mer = codigo")
    prepararSentencia.ejecutarConsultaSQL()
    mercaderia.setDatos( prepararSentencia.getParametros() )
    invocar this.cerrarConexion()
    retorna mercaderia
}

```

#### Diseño Procedimental de la clase Mercadería

Definir paquete kautsch.Modelo.Datos;

Importar biblioteca de clases a utilizar

class Mercaderia

{

    definir variables\_de\_mercaderia // corresponde a la tabla Mercaderia

```

metodo setDatos(parámetros_de_mecaderia) { //asigna valores a los atributos de la mercadería
    this.variables_de_mercaderia= parámetros_de_mecaderia
}
metodo tipoDeDato getDatos() { // obtiene los atributos del sistema según el tipo de variable
    retorna this.variables_de_mercaderia
}

```

Diseño Procedimental para la conexión a la base de datos

Definir paquete modelo.conexionBD

Importar biblioteca de clases a utilizar

Class ConexionBD{

Definir variable usuario

Definir variable contraseña

Constructor conexionad(usuario, contraseña){

    This.usuario = usuario

    This.contraseña= contraseña

    Definir url\_base\_de\_datos

    Definir driver\_base\_de\_datos

}

metodo AbrirConexion() {

    invocar clase.nombreDriver(driver\_base\_de\_datos)

    invocar coneccion\_base\_de\_datos(url\_base\_de\_datos, usuario, contraseña)

}

metodo CerrarConexion() {

    invocar coneccion.close();

}

}

#### 4.4 IMPLEMENTACIÓN.

La implementación se lleva representando los modelos del diseño de objetos al lenguaje de programación Java.

A continuación se presenta la implementación de clase, asociación y multiplicidad en el lenguaje de programación java, de las clases mercadería y categoría. La relación es de muchos a uno de mercadería a categoría y de uno a muchos de categoría a mercadería

como se muestra la figura 30, en el caso de existir asociaciones de muchos a muchos la relación se representa con una clase y un vector de punteros.

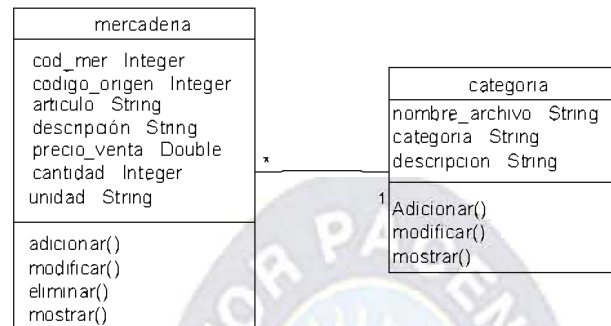


Figura 32. Asociación de las clases mercadería y categoría, Fuente: Elaboración propia.

```
public class Mercaderia {
    int        codMercaderia;
    String      codigoOrigen;
    String      articulo;
    String      descripcion;
    int         cantidad;
    double      precioVenta;
    String      unidad;
    Categoria categoria;

    public void setCodigo(int codigo){
        this.codMercaderia=codigo;
    }
    public int getCodigo(){
        return this.codMercaderia;
    }
    . . .
    public void setCategoria(Categoria categoria){
        this.categoria=categoria;    }
    public Categoria getCategoria () {
        return this.categoria;
    }
}
```

```

}

public class Categoria {
    String  archivo;
    String  categoria;
    String  descripcion;
    Mercaderia Mercaderia[];
    public void setArchivo(String nombre){
        this.archivo = nombre;
    }
    public String getArchivo(){
        return this.archivo;
    }
    . . .
    public void setMercaderia(Mercaderia Mercaderia[]){
        int i=0;
        while(Mercaderia.next()){
            this.mercaderia.addElement(Mercaderia.getElement());
        }
    }
    public Iterador getMercaderia(){
        return Mercaderia.Iterator();
    }
}

```

La interfaz gráfica de usuario del pseudocódigo es el que se muestra a continuación.

La figura 33 muestra la ventana de inicio del sistema para un usuario válido y el inicio del nuevo día registrando fecha y el tipo de cambio.

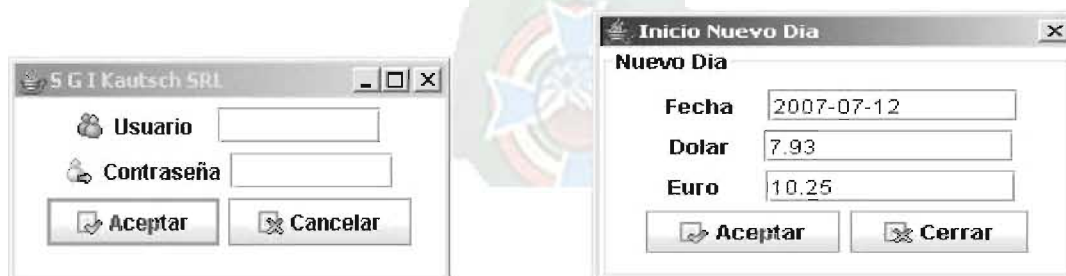


Figura 33. Ventana de inicio del sistema e inicio del nuevo día, Fuente: Elaboración propia.

La figura 34 muestra la ventana de control de mercadería donde se realizan las operaciones adicionar, modificar y eliminar.



**Control de Mercadería**

Código: 680325    Código de origen: 95-IB-69172

Artículo: SARMADOR JUEGO    Cantidad: 15

Descripción: DESARMAD. EN JUEGOS, 10 pzs.plano/estr.

Unidad: jgo    Precio de venta: 36.78

Archivo: YSTANLEY    Nuevo archivo

Marca: stanley    Proveedor

Operacion:

Adicionar

Modificar

Eliminar

Mostrar    Limpiar    Cancelar

Figura 34. Ventana de control de mercadería, Fuente: Elaboración propia.

La figura 35 muestra la ventana de cambio de precio por archivo, con un incremento del 5% en el precio de venta.

**Cambio de precio por archivo**

datos

archivo >> YTIRADORES1    Incr. en precio % 5

codigo	articulo	descripcion	precio	cantidad
902001	BOTON	BOTON p. muebles	2.31	83
902002	BOTON	BOTON p. muebles	1.91	10
902006	BOTON	BOTON p. muebles	2.11	40
902008	BOTON	BOTON p. muebles	2.01	16
902014	BOTON	BOTON p. muebles	3.01	0
902016	BOTON	BOTON p. muebles	1.91	93
902018	BOTON	BOTON p. muebles	2.61	19
902022	BOTON	BOTON p. muebles	3.12	104
902024	BOTON	BOTON p. muebles	2.81	24
902026	BOTON	BOTON p. muebles	3.92	100
902028	BOTON	BOTON p. muebles	2.61	49
902030	BOTON	BOTON p. muebles	2.71	24
902034	BOTON	BOTON p. muebles	2.31	30
902038	BOTON	BOTON p. muebles	1.91	16
902042	BOTON	BOTON p. muebles	4.74	22

Aceptar    Limpiar    Cancelar

Figura 35. Ventana de cambio de precio por archivo, Fuente: Elaboración propia.

La figura 36 muestra la ventana de búsqueda de mercadería en un rango de códigos

**busqueda**

Codigo Inicial >> 900000 y Final >> 999999 **Buscar**

codigo	articulo	descripcion	precio
920087	BOTON	BOTON, bronce antiguo de 30mm.	1.71
920089	BOTON	BOTON, bronce pulido de 30mm.	2.31
920091	BOTON	BOTON, black nickel de 30mm.	1.91
920093	BOTON	BOTON, blanco de 30mm.	1.41
920095	BOTON	BOTON, cromado de 32mm.	1.71
920096	BOTON	BOTON, bronce antiguo de 32mm.	1.81
920097	BOTON	BOTON, bronce pulido de 32mm.	2.51
920098	BOTON	BOTON, black nickel de 32mm.	1.81
920300	TIRADOR	TIRADOR, bronce cromado de 96mm.	1.71
920310	TIRADOR	TIRADOR, bronce pulido de 64mm.	3.22
920311	TIRADOR	TIRADOR, black nickel de 64mm.	3.42
920312	TIRADOR	TIRADOR, bronce antiguo de 64mm.	3.42
920315	TIRADOR	TIRADOR, bronce pulido de 96mm.	3.92
920316	TIRADOR	TIRADOR, black nickel de 96mm.	4.22
920212	TIRADOR	TIRADOR, bronce antiguo de 96mm.	4.02

**Limpiar** **Cerrar**

Figura 36. Ventana de búsqueda en un rango de códigos, Fuente: Elaboración propia.

**JasperViewer**

Reporte de ventas por Factura - fechas

Fecha	Nro de factura	Precio Total	Descuento	Razón Social
19-10-06 12:00 AM	1002	530.0	58.0	omsa
20-10-06 12:00 AM	1004	2000.0	112.0	omsa
24-10-06 12:00 AM	1007	355.0	0.0	solucion system
02-04-07 12:00 AM	1008	128.0	0.8	jalapeños
11-06-07 12:00 AM	1010	328.0	0.0	jalapeños
23-10-06 12:00 AM	1006	1218.0	0.0	sarmiento
15-10-06 12:00 AM	1001	300.0	54.0	gomez
05-05-07 12:00 AM	1009	252.8	0.0	gomez
21-10-06 12:00 AM	1005	640.0	0.0	calle
09-06-07 12:00 AM	1012	1420.8	0.0	comisal
20-10-06 12:00 AM	1003	800.0	0.0	Rene Espinoza

Página 1 de 1

Figura 37. Ventana de reporte de ventas según factura, Fuente: Elaboración propia.

La figura 37 muestra un reporte por facturas de las ventas, en un rango de fechas

En todas las ventanas las operaciones que no son posibles de ejecutar quedan deshabilitadas.

La figura 38 muestra el esquema de la interfaz gráfica de usuario del sistema, a partir del menú principal.

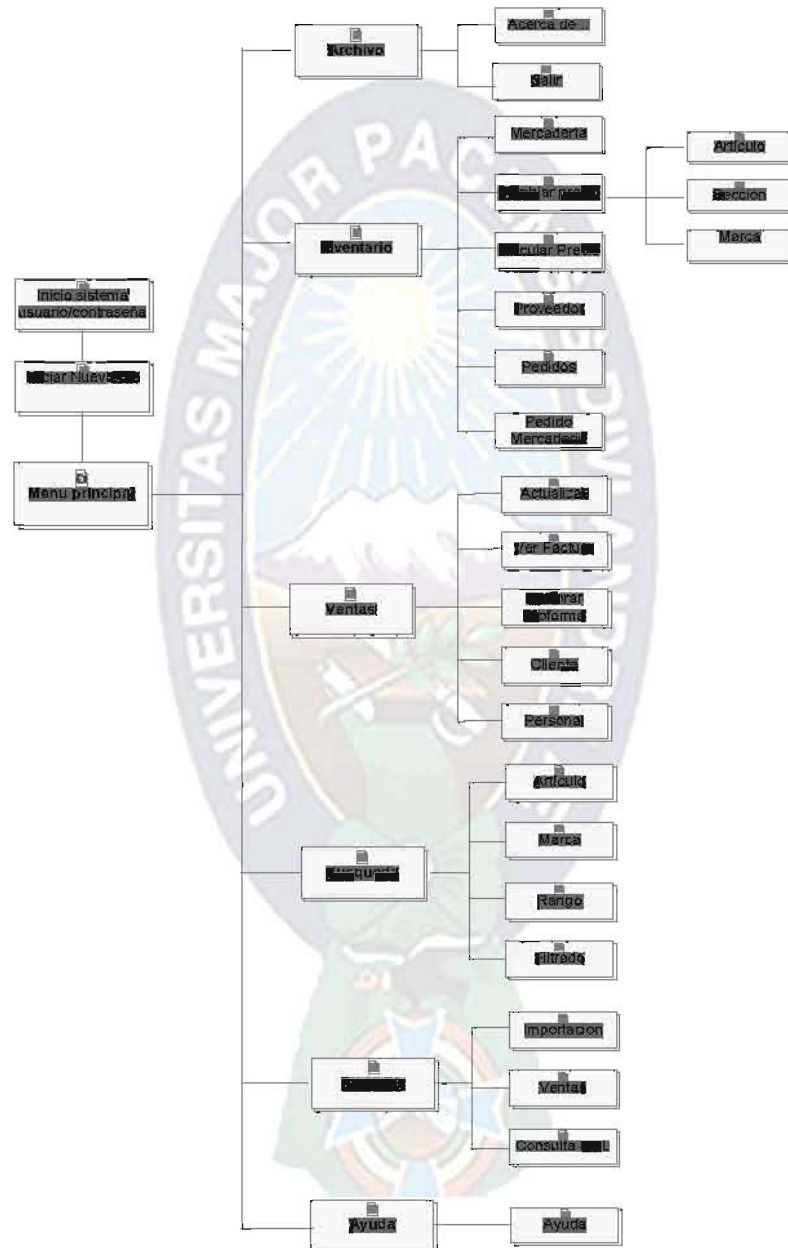
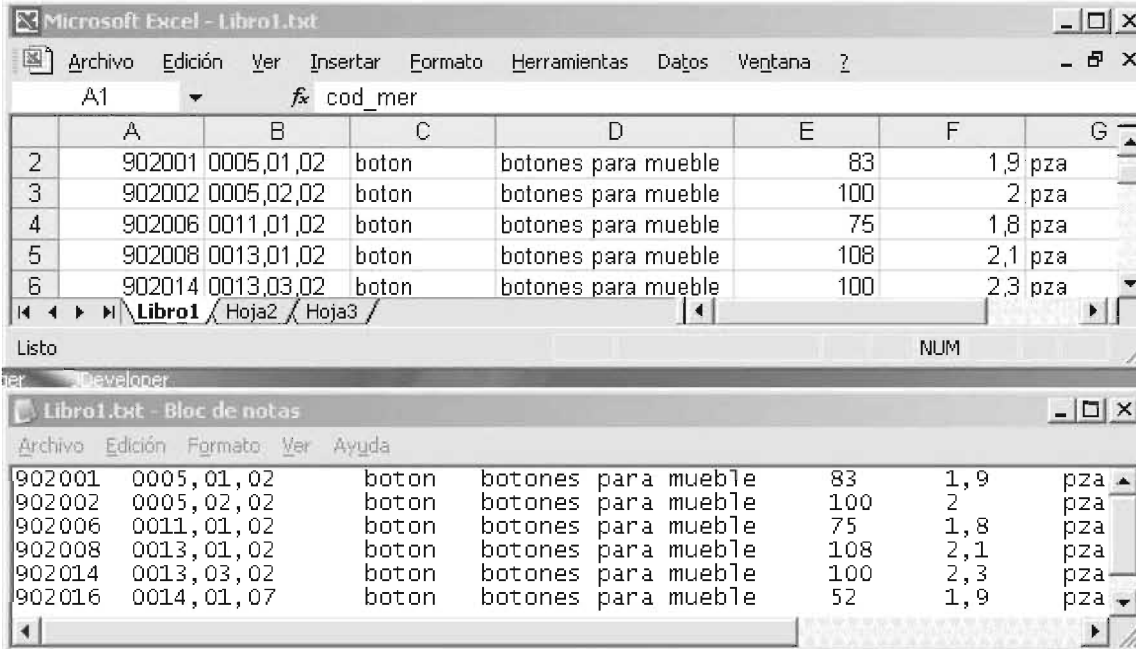


Figura 38. Esquema de navegación de la interfaz de usuario del sistema, Fuente: Elaboración propia.

#### 4.5. MIGRACIÓN DE DATOS.

Para realizar la migración de datos de archivos planos a la base de datos. Se establece los siguientes pasos:

- a.- Se define el entorno de migración de los archivos planos a la base de datos, tomando en cuenta las llaves primarias y extranjerías de la base de datos.
- b.- Se define el gestor de base de datos, en este caso Navicat MySQL GUI para MySQL y Excel la herramienta pivote para realizar la migración de datos, también se podría utilizar archivos ASCII. Los Archivos ASCII y Excel son herramientas utilizados por muchos gestores de bases de datos, para realizar la migración de archivos planos.
- c.- Estructurar la información en Excel o en un archivo ASCII, de acuerdo al esquema de la bases de datos como en la figura 39.



Microsoft Excel - Libro1.txt

	A	B	C	D	E	F	G
2	902001	0005,01,02	boton	botones para mueble	83	1,9	pza
3	902002	0005,02,02	boton	botones para mueble	100	2	pza
4	902006	0011,01,02	boton	botones para mueble	75	1,8	pza
5	902008	0013,01,02	boton	botones para mueble	108	2,1	pza
6	902014	0013,03,02	boton	botones para mueble	100	2,3	pza

Libro1.txt - Bloc de notas

902001	0005,01,02	boton	botones para mueble	83	1,9	pza
902002	0005,02,02	boton	botones para mueble	100	2	pza
902006	0011,01,02	boton	botones para mueble	75	1,8	pza
902008	0013,01,02	boton	botones para mueble	108	2,1	pza
902014	0013,03,02	boton	botones para mueble	100	2,3	pza
902016	0014,01,07	boton	botones para mueble	52	1,9	pza

Figura 39. Archivos Excel y ASCII, Fuente: Elaboración propia.

d.- Para llevar los datos del archivo Excel o ASCII a la base de datos se debe tener mucho cuidado, con los campos que son llaves extranjerías, porque estos campos no aceptan valores nulos, por lo tanto se empieza a estructurar los datos para las tablas que no tienen llaves extranjerías. La figura 40 muestra la estructura de la tabla mercadería con llave extranjera archivo encerrado en un círculo, por lo tanto se migra inicialmente la tabla categoría que cuenta con 31 ítems, que tiene como llave primaria al campo archivo, que es llave extranjera de la tabla mercadería.

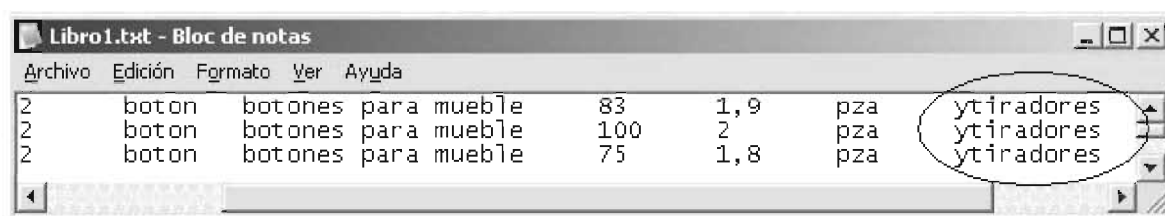


Figura 40. Archivos Excel y ASCII, Fuente: Elaboración propia.

e.- Se realiza la migración de datos, utilizando la herramienta pivote, soportada por el gestor de base de datos. Se inicia la migración, con las tablas que no tienen llaves extranjeras, continuando con aquellas tablas dependientes de las llaves extranjeras como se indica a continuación.

Categoría (archivo, categoria, descripcion),

Cliente (nit\_ci, razon\_social, descripcion, teléfono)

Personal (cod\_per, nombre, ap\_paterno, ap\_materno, cargo)

Factura (nro\_fac, fecha, precio\_total, descuento, tipo\_pago, cod\_per, cod\_cli)

Mercaderia (cod\_mer, codigo\_origen, artículo, descripcion, cantidad, precio\_venta, unidad, archivo)

Moneda\_cambio (fecha, dolar, euro)

Pedido (nro\_pedido, fecha, cant\_importacion, costo, país, ciudad, moneda, cod\_mer)

Proveedor (marca, sitio\_web, email, descripcion)

Provee (cod\_mer, marca, estado)

Venta (cod\_ven, cant\_venta, precio, cod\_mer, cod\_fac)

f.- Una vez migrado la información de los archivos planos, se verifica y corrige que los datos migrados son correctos de acuerdo a lo establecido en los archivos planos.

#### 4.6. PRUEBAS.

Dado que las pruebas son el proceso de examinar el software con la intención de encontrar errores, durante el desarrollo del producto software, se fueron probando las funciones del sistema y la forma en que estas interactúan con el usuario final, considerando en todos los casos, las siguientes características: ingreso de datos erróneos, ingreso de identificadores repetidos y campos vacíos, despliegue de mensajes de acuerdo al tipo de error generado.



Se realizaron pruebas en las variables que manejan precios (números decimales), el calculo de operaciones con variables que manejan decimales, se redondea al dígito inmediato superior, tomando en cuenta cuatro decimales.

Las pruebas de unidad son realizadas sobre el encapsulamiento de atributos y operaciones de las clases, se da inicio con las clases especializadas hacia las clases generalizadas.

Consecutivamente se realiza la prueba de integración, donde se prueba la integración de los módulos que encapsulan las clases necesarias para responder a una entrada o evento del sistema o modulo. Aquí se ha mejorado el intercambio de información entre módulos, buscando un acoplamiento bajo y una alta cohesión.

Para realizar la prueba de validación se apoya en los casos de uso y los diagramas de transición de estados, verificando que las acciones del usuario y las salidas del sistema sean los esperados. De las pruebas realizadas, se identificaron algunas deficiencias en cuanto al ingreso de datos como el tipo de dato o campos vacíos, se corrigieron validando las entradas del usuario y para las entradas campos vacíos, se superaran indicando al usuario que tiene algún campo con un dato vacío. También se ha corregido el manejo de datos numéricos cambiando la precisión de dos a cuatro dígitos.

La interfaz gráfica de usuario, cuenta con el contenido suficiente y la facilidad para la interacción entre el sistema y el usuario, durante esta prueba se identifico que debieron estar deshabilitadas, las operaciones que no son posibles de realizar: Por ejemplo no se debe habilitar la opción adicionar en una ventana, si los campos de texto están vacíos.

En las operaciones de búsqueda se agregó la opción de buscar con palabras incompletas para tener efectividad en la búsqueda de información.

#### **4.6. MÉTRICAS.-**

La tendencia de la Ingeniería del Software es introducir la medición en todas las etapas del ciclo de vida de un proyecto software independientemente del modelo utilizado: cascada, espiral o técnicas de cuarta generación.

Las métricas en [PIA-2001] Chidamber y Kemerer establecen métricas para medir cinco atributos básicos en el diseño orientado a objetos: acoplamiento, complejidad de una clase, reutilización, cohesión, y herencia.

Para el caso de estudio se utiliza la clase mercadería para ver con que objetos esta ligado, como se muestra en la figura 41.

El acoplamiento entre clases, es el número de clases ligadas, sin tener relaciones de herencia.

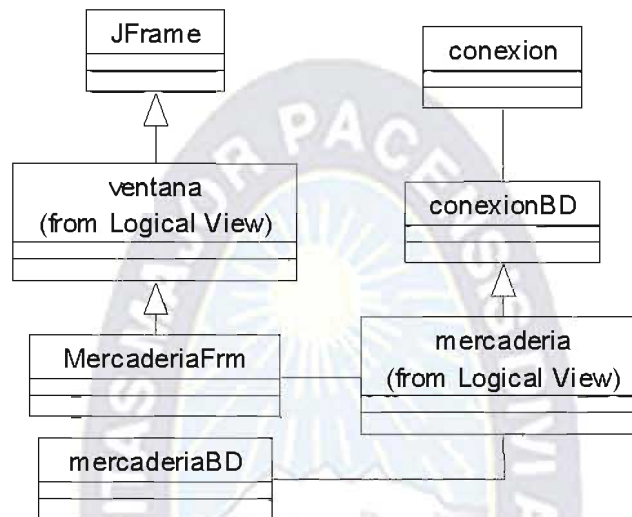


Figura 41. Relaciones de la clase mercadería, Fuente: Elaboración propia.

La figura 32 muestra los objetos que tienen un bajo acoplamiento lo que permite a un objeto ser independiente y fácil de reutilizarlo en otra aplicación. Al reducir el acoplamiento se reduce la complejidad, se mejora la modularidad y se promueve el encapsulamiento.

La complejidad ciclomática  $V(G)$  de una operación o método es el número de caminos independientes a lo largo del programa. A continuación se presenta el pseudocódigo de las operaciones de la mercadería. El pseudocódigo es el siguiente:

Formulario de Mercaderia

```

Definir variables
Iniciar variables
Mostrar ventana
Leer evento de la operación
Mientras evento != cerrar
    Si evento = adicionar entonces
        Adicionar Mercaderia
    Si no ignorar
    Fin si
    Si evento = eliminar entonces
        Eliminar Mercaderia
    Si no ignorar
  
```

```

    Fin si
    Si evento = Buscar entonces
        Mientras existe registro
            Guardar registro en Mercaderia
        Fin Mientras
    Si no ignorar
    Fin si
    Mostrar Mercaderia
    Fin Mientras
Fin de fomulario Mercaderia

```

Del siguiente pseudocódigo se obtiene el grafo de flujo

La figura 36 muestra el grafo de flujo, donde se aprecia el número de nodos, aristas y regiones, de donde se obtiene los siguientes datos

Complejidad ciclomática:

$$V(G) = \# \text{ aristas} - \# \text{ nodos} + 2$$

$$V(G) = \# \text{ nodos predocado} + 1$$

Entones:

$$\begin{aligned}
 V(G) &= 19 - 16 + 2 \\
 &= 4 + 1 \\
 &= 5
 \end{aligned}$$

La complejidad ciclomática del grafo de flujo es 5 e indica el número de caminos independientes y consecuentemente un valor limite superior del número de pruebas que se deben realizar para que cubran las sentencias del programa.

En la siguiente página la figura 36 ilustra el grafo de flujo para hallar la complejidad ciclomática.

En la tabla 3 del capitulo anterior se menciona que la complejidad ciclomática es un indicador útil de la dificultad de prueba y mantenimiento de un programa y sugiere un valor máximo de 10, lo que implica que la dificultad de prueba y mantenimiento es relativamente compleja.

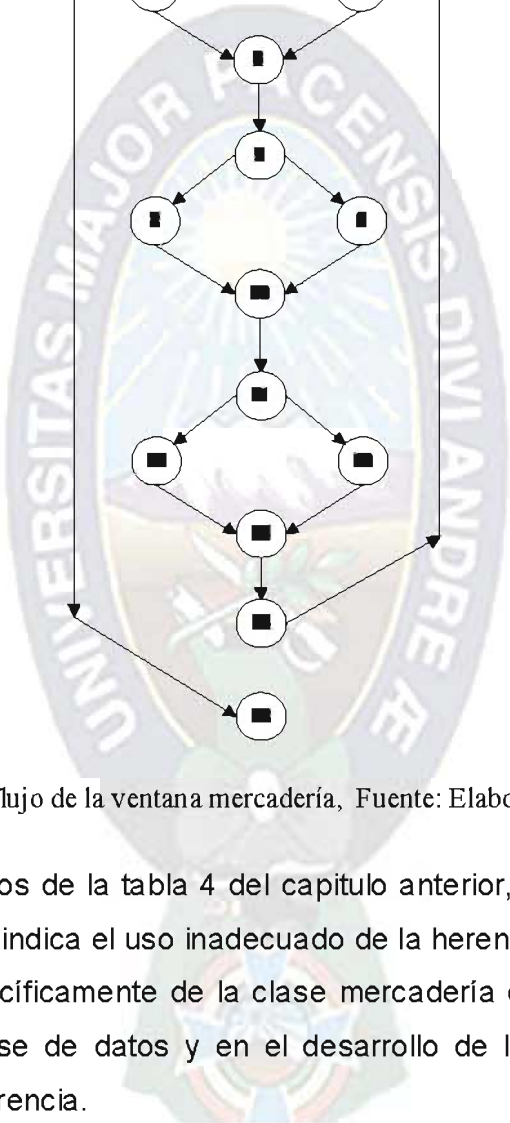


Figura 42. Grafo de flujo de la ventana mercadería, Fuente: Elaboración propia.

La métrica número de hijos de la tabla 4 del capítulo anterior, es un indicador del nivel de reutilización. También indica el uso inadecuado de la herencia la figura 32 muestra el uso de la herencia, específicamente de la clase mercadería donde se tiene diferentes operaciones sobre la base de datos y en el desarrollo de la interfaz de usuario, se busca no abusar de la herencia.

La métrica de profundidad de un árbol de herencia de la tabla 5 del capítulo anterior, mide el máximo nivel en la jerarquía de la herencia. Lorenz y Kidd, sugieren un umbral de 6 niveles como indicador de un abuso de la herencia, el lenguaje de programación java añade por defecto, la herencia de la clase Object, la figura 32 muestra la profundidad de la herencia iniciando en la clase Object, JFrame, Ventana y ventana de

mercadería, se tiene una profundidad de 4 niveles, lo que refleja un uso equilibrado de la herencia.

#### 4.7. CALIDAD DEL SOFTWARE.-

Existen varias métricas de calidad de las cuales se escogieron la funcionalidad, la portabilidad y la Mantenibilidad.

Fiabilidad. Es un factor de calidad muy importante, que indica en que medida satisface el sistema de información frente a las necesidades del cliente.

Portabilidad. Es importante que un sistema sea portable a otro entorno de hardware y/o software, para que su ejecución en el futuro no traiga problemas.

Mantenimiento. El mantenimiento del software es necesario si se quiere que el mismo sea útil a través del tiempo y se acomode a los cambios constantes de ambiente en el que funciona.

**Funcionalidad.-** funcionalidad del sistema esta dado por el dominio de la información al cual esta asociado un valor de complejidad. Los dominios de información son:

- Número de entradas de usuario.
- Número de salidas de usuario.
- Número de peticiones de usuario.
- Número de archivos o tablas
- Número de interfaces

Los puntos de función se obtienen computando la tabla 9.

PARAMETRO DE MEDICION	Factor de ponderación				
	cuenta	simple	medio	complejo	TOTAL
Nro. de entradas de usuario	13	3	4	6	39
Nro. de salidas de usuario	20	4	5	7	80
Nro. de peticiones de usuario	31	3	4	6	93
Nro. de archivos o tabla	10	7	10	15	70
Nro. de interfaces	16	5	7	10	80
Cuenta total					318

Tabla 10. Computo de métricas de punto de función, Fuente Presman, [PRE-2002]



Para ajustar la complejidad de los puntos de función según el factor de ajuste se asignan pesos de acuerdo a los siguientes casos como se ve en la tabla 10.

Factor	Valor
Sin importancia	0
Incidental	1
Moderado	2
Medio	3
Significativo	4
Esencial	5

Tabla 10. Pesos de los puntos de función, Fuente. Pressman, [PRE-2002]

Los ajustes de complejidad se ven en la tabla 11.

Factor	peso
1. Requiere el sistema copias de seguridad y de recuperación fiable	5
2. Se requiere comunicación de datos	4
3. Existen funciones de procesos distribuidos	1
4. Es crítico el rendimiento	4
5. Será ejecutado el sistema en un S.O. existente	5
6. Requiere el sistema entrada interactiva	3
7. Requiere entrada de datos interactiva sobre múltiples ventanas	3
8. Se actualiza los archivos de forma interactiva	4
9. Son complejas las salidas de los archivos a las peticiones	3
10. Es complejo el procesamiento interno	3
11. se ha diseñado el código para ser reutilizable	3
12. Están incluidas en el diseño la conversión la conversión e instalación	3
13. Se ha diseñado el sistema para soportar múltiples instalaciones	4
14. se ha diseñado la aplicación para facilitar los cambios y ser fácilmente utilizada por el usuario	4
<b><math>\Sigma F_i</math></b>	<b>54</b>

Tabla 11. Factores de evaluación, Fuente Elaboración propia

Entonces el punto de función es:

$$PF = \text{cuenta total} * [0.65 + (0.01 * \sum F_i)]$$

$$PF = 318 * [0.65 + (0.01 * 54)]$$

$$PF = 378.42$$

**Confiabilidad.** Un sistema de información es confiable si no produce fallas, si se usa de manera razonable. El sistema debe cumplir con los requerimientos y debe producir resultados correctos.

Para evaluar la confiabilidad se tomara en cuenta, las fallas que podrían presentarse en un lapso de 5 años, con un margen de error del 10 por ciento.

Datos  $\alpha = 0.10$  y  $t = 5$  años. Remplazando estos datos en la siguiente ecuación se tiene:

$$R(t) = e^{-\alpha t}$$

$$R(t) = e^{-0.10 * 5}$$

$$R(t) = 0.8187$$

Entonces la confiabilidad del sistema con un margen de error del 10 por ciento en un lapso de dos años es: 81.87 % de confiabilidad.

**Mantenibilidad.-** Las siguientes variables permitirán hallar el índice de madurez del software para ver cuan robusto es:

$M_t$  = número de módulos de la versión actual.

$F_a$  = número de módulos en la función actual que se han añadido.

$F_c$  = número de módulos en la versión actual que se han cambiado.

$F_d$  = número de módulos de la versión anterior que se han borrado en la versión actual.

$$IMS = [ M_t - ( F_a + F_c + F_d ) ] / M_t$$

Donde se tiene:  $M_t = 14$  ;  $F_a = 5$  ;  $F_c = 7$  ;  $F_d = 1$

$$IMS = [ 14 - ( 5 + 7 + 1 ) ] / 14$$

$$IMS = 0.07$$

Por lo tanto el índice de madurez del software es del 7 %.

Lo que implica que el nuevo producto software es completamente distinto al sistema anterior realizado con macros y hojas electrónicas de Lotus.

El nuevo sistema cumple con los requerimientos planteados incluyendo las tareas que se ejecutaban con el sistema anterior.

**Portabilidad.-** La portabilidad de sistema esta garantizada tomando en cuenta que el lenguaje de programación Java es multiplataforma, solo necesita ser instalado la maquina virtual según la plataforma de trabajo.

El gestor de base de datos de MySQL es también portable ya que existen diferentes distribuciones en código binario o código ejecutable según la plataforma de trabajo, además no se utilizan funciones o procedimientos almacenados, para mantener la portabilidad.

Si en el futuro se quisiera cambiar de gestor de base de datos se debe agregar el driver (archivo .jar) del nuevo gestor de base de datos en la aplicación y se realiza una leve modificación en la clase que ejecuta la conexión con la base de datos.

**Seguridad.-** La seguridad en cuanto a la información es muy importante, más aun si esta en una red, los datos deben estar seguros de cualquier tipo de ataque que surja.

El Sistema de Gestión de Inventarios será instalado en la red local que cuenta con generadores de energía en caso de que ocurran cortes de energía eléctrica, el servidor de bases se encuentra en la oficina de la dirección y gerencia por lo que sólo estos usuarios tendrán acceso completo al sistema.

Los otros usuarios como el personal de apoyo y el encargado de realizar proformas tienen autorizaciones de acceso limitados.

El gestor de base de datos se encarga de denegar o conceder el acceso a los usuarios, ya que cada usuario tiene diferentes roles en la empresa como se indicó en la figura 18.

También se tiene como política de seguridad generar backups del sistema cada día, las cuales será salvadas en un medio magnético un vez al mes.

## CAPITULO 5

## MARCO FINAL

**5.1 CONCLUSIONES.-** La importancia de llevar un control del flujo de información de la mercadería que posee Kautsch S.R.L. a través de un producto software permitirá gestionar la información de mejor manera.

En este sentido el desarrollo del sistema de información denominado sistema de gestión de inventarios ha cumplido con los objetivos establecidos:

Se sistematizaron los procesos de importación de mercadería, actualización de ventas y elaboración de proformas.

La utilización de la metodología técnica de modelado a objetos y la ingeniería de software orientado a objetos combinado con el lenguaje de modelado unificado, fue satisfactoria para modelar el entorno de trabajo e identificar los procesos y actores del sistema para su posterior implementación.

Se tiene un mejor control del stock de los artículos, acceso inmediato a la información mediante búsquedas en diferentes formatos y la generación de reportes de las importaciones y las ventas.

La migración de datos fue un proceso delicado, en especial con las tablas que son dependientes de otras tablas con llaves extranjeras, ya que estos campos no admiten datos nulos.

**5.2. RECOMENDACIONES.-** A continuación se tiene las siguientes recomendaciones, que se deben considerar para que la empresa logre ampliarse para atender mucho mejor a los clientes no solo de la ciudad de La Paz, sino también del interior del país y así competir de mejor manera en el rubro al que se dedica la empresa:

Desarrollar un sistema de control de planillas de sueldos para el personal de la empresa.

Desarrollar un sistema de contabilidad, integrado al sistema de gestión de inventarios.

Extender el esquema de la base de datos para trabajar con los sistemas de contabilidad y control de planillas.

Ampliar la cobertura del sistema de gestión de inventarios a la sucursal de la zona sur a través de la red de Internet.

Desarrollar un portal Web, tomando en cuenta la exclusividad de la mercadería con la que cuenta, para extender el mercado y así brindar un mejor servicio a las empresas del interior.



## BIBLIOGRAFÍA.-

### Libros.

- [BOE-1981] Boehm, B. (1981). "Software Engineering Economics", editorial Prentice Hall, ciudad Madrid España.
- [PRO-1992] Walpole, R. y Myers R. (1992). "Probabilidad y estadística", editorial McGraw Hill, cuarta edición ciudad México México España. Capítulos 1,2 y 5
- [RUM-1996] Rumbaugh, J. y et al. (1996). "modelado y diseño orientado a objetos Metodología OMT", editorial Prentice Hall, ciudad Madrid España, capítulos 2, 7, 8, 9,10, 11 y 13.
- [MEY-1998] Meyer, B. (1998). "*Construcción de software orientado a objetos*", editorial Prentice Hall, Segunda edición Ciudad Madrid España, capítulos 2, 9,10,11, 23 y 24.
- [PIA-2000] Piatinni, M. G. y et al (2000). "*Análisis y Diseño detallado de Aplicaciones Informáticas de Gestión*", Editorial Alfaomega, Primera Edición, Ciudad Madrid España, capítulos 2 y 3.
- [PIA-2001] Piatinni, M. G. y et al (2001)" *Métricas en el desarrollo y Mantenimiento del software*". Editorial Alfaomega, Primera Edición, Ciudad Madrid España, capítulo 5.
- [SCH-2001] Schumeller, J. (2001)" *Aprendiendo UML en 24 horas*". Editorial Prentice Hall, Primera Edición, Ciudad México México, parte I y II.
- [PRE-2002] Pressman, R. S. (2002). "*Ingeniería de software un enfoque practico*", Editorial McGraw Hill, Quinta Edición, Ciudad Madrid España, capítulos 10 y 19.
- [BRA-2003] Braude E. J. (2003). "*Ingeniería de software una perspectiva orientada a objetos*", Editorial Alfaomega, Primera Edición, Ciudad México, capítulos 5 y 6.



- [SOM-2005] Somerville, I. (2005). "*Ingeniería de software*", Editorial Pearson Addison Wesley, Séptima Edición, Ciudad Madrid, España, capítulos 4, 6, 7,8, 11, 14, 16, 23 y 27.
- [WEI-2006] Weitzenfeld A, I. (2006). "*Ingeniería de software orientada a objetos con UML, Java e Internet*", Editorial Pearson Addison Wesley, Ciudad Madrid, España, capítulos 1, 2, 3, 4, 5, 6, 7,8, 9 y 10

#### **Sitios visitados.**

- [1.]Medinilla, M. M. "Análisis y selección de estrategias de desarrollo de software" de la universidad politécnica de Madrid, paginas usadas 2-20. Disponible en:  
<http://is.ls.fi.upm.es/udis/docencia/proyecto/docs/estrategias.pdf> fecha visitada 2006-03-20.
- [2.]Meyer B. "Introducción a la PDO", tecnologías orientadas a objetos, páginas usadas2-14. Disponible en:  
<http://is.ls.fi.upm.es/udis/docencia/proyecto/docs/temas1sf.pdf> fecha visitada 2006-05-01.
- [3.] [<http://www.math.utsa.edu/sphere/salingar/Chris.text.html>]
- [4.]Mendoza E. Luis y et al. "Algoritmo para la evaluación de la calidad sistémica del software", Laboratorio en sistemas de Información, Universidad Simón Bolívar, Caracas Venezuela, páginas usadas 4-7. Disponible en:  
[http://www.lisi.usb.ve/publicaciones/02%20calidad%20sistemica/calidad\\_21.pdf](http://www.lisi.usb.ve/publicaciones/02%20calidad%20sistemica/calidad_21.pdf).  
Visitada 2006-10-30.
- [5.]Alejandro Bedini G. "Calidad tradicional de software ", universidad Técnica Federico santa María Industrias, Campus Chile, páginas usadas 1-4, Disponible en:  
[www.willydev.net/descargas/articulos/general/CalidadSoftware.pdf](http://www.willydev.net/descargas/articulos/general/CalidadSoftware.pdf)  
Visitada: 2006-10-10
- [6.][www.rationalrose/rup](http://www.rationalrose/rup)

[7.] <http://java.sun.com/>

[8.] Vázquez C. "Introducción a JasperReports e iReport ", Comunidad de programación MYGNET, páginas usadas todo. disponible en: <http://www.mygnet.com/articulos/java/301/index.php> Visitada 2007-03-03.

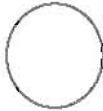

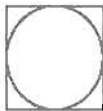



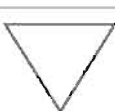

[9.] Teodor Danciu, " The JasperReports Ultimate guide", páginas usadas todo, Disponible en:

<http://jasperreports.sourceforge.net/> Visitada 2007-03-15



**Anexo 1.****SIMBOLOGÍA DE LA NORMA ISO 9000**

Símbolos de la norma ISO 9000 para elaborar diagramas de procesos

SÍMBOLO	REPRESENTA
	Operaciones. Fases del proceso, método o procedimiento.
	Inspección y medición. Representa el hecho de verificar la naturaleza, calidad y cantidad de los insumos y producto.
	Operación e inspección. Indica la verificación o supervisión durante las fases del proceso, método o procedimiento de sus componentes.
	Transportación. Indica el movimiento de personas, material o equipo.
	Demora. Indica retraso en el desarrollo del proceso, método o procedimiento.
	Decisión. Representa el hecho de efectuar una selección o decidir una alternativa específica de acción.
	Entrada de bienes. Productos o material que ingresan al proceso.
	Almacenamiento. Depósito y/o resguardo de información o productos.