121
21

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA PARA LA AUTOMATIZACION, REGISTRO Y CONTROL DE LA INFORMACION DE UN BROKER (COMPAÑIA DE AGENTES DE VENTAS DE FIANZAS)

T E S I S P R O F E S I O N A L

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A :

JUAN DANIEL VIGUERAS SANCHEZ

DIRECTOR DE TESIS:
ING. ADOLFO MILLAN NAJERA

CIUDAD UNIVERSITARIA 1996

AGRADECIMIENTOS

A mi madre:

Por su gran cariño, apoyo, sacrificio y sobre todo, porqué sin una madre como ella, jamas hubiera logrado ser lo que hoy soy.

A mi padre:

Por sus consejos, ejemplo y apoyo en momentos muy difíciles, de mi vida, y en la cual siempre conté con el para compartir cualquier problema o alegría.

A mis hermanos:

Víctor, Fernando y Cynthia, por que de alguna manera fueron los que me impulsaron a continuar siempre adelante y sobre todo por los grandes momentos de alegrías y satisfacciones que hemos vivido juntos..

A mis Tías:

Juanita y Lulú, nunca pude haber tenido unas tías mejores, que me han brindado su apoyo en todo lo que han podido y sobre todo la gran confianza y amor que han depositado en mí., me han ayudado para alcanzar todas las metas que me he fijado.

A mis Tíos:

Pascual y Estela, Víctor y Cosme, Efren y Luz María, siempre que necesite un consejo, alguien con quien contar, un estímulo o apoyo, siempre estuvieron respaldándome y haciéndome participe de sus dichas y alegrías.

A mis Primos:

Prisciliano, Ana, Susana, Omar, Hugo, Ivan e Irasema, por su confianza, su ejemplo y en algunos casos sus consejos o sugerencias, que me han servido para lograr mis objetivos y para ser mejor.

En memoria de mis Abuelos

Juan y Eva, Leopoldo y Juana, a los últimos nos los conocí, pero estoy seguro que al igual que Juan y Eva, me hubieran apoyado y dado su amor, cariño y algunos regaños, para que fuera un hombre de bien y hubiera alcanzado todas mis metas. Estén donde estén quiero decirles a ustedes, que sentaron las bases de mi familia, que este esfuerzo y trabajo también es parte suya.

Agradecimiento Especial:

En la elaboración de un trabajo como este no sólo interviene una sola persona, si no es el esfuerzo, recursos y tiempo de un gran número de personas que colaboran e intervienen, como son amigos que ayudan a corregir errores y a hacer sugerencias, debido al gran número de personas que intervinieron y debido a que mi memoria es muy mala voy a mencionar sólo algunos, espero me disculpen los que falten.

Ing. Adolfo Millán Nájera

Gracias a su experiencia, consejos y sobre todo a su valioso tiempo han hecho que éste trabajo sea lo que hoy es y sobre todo a contribuido a que yo de el último paso para lograr la titulación.

L.D.G. Adriana Vigueras y Juan Carlos

Por su apoyo y tiempo que les quite de alguna manera al prestarme los recursos necesarios para la elaboración de este trabajo, así como algunos consejos para la elaboración del mismo.

A las Corchofatas:

Por la gran época que vivimos en la escuela y que aún seguimos viviendo, espero que la amistad que tenemos se fortifique con el paso del tiempo.

ÍNDICE.

INTRODUCCIÓN

El Cuerpo Humano es el carruaje;
el yo, el hombre que lo conduce;
el pensamiento son las riendas,
y los sentimientos, los caballos.

Platón

INTRODUCCIÓN

En la época que vivimos actualmente, la computación y los sistemas de información juegan un papel demasiado importante en el campo empresarial, ya que no sólo sirven para controlar la información existente y generada por ésta, sino también para administrar, auxiliar, automatizar y tomar decisiones que son de suma importancia para una empresa, las cuales hoy en dia se enfrentan a un dilema. Las empresas se están convirtiendo cada vez más en organizaciones basadas en información, dependiendo de un flujo continúo de datos para cualquier aspecto de sus operaciones. Sin embargo su habilidad para el manejo de los datos se ve disminuido porque el volumen de información se expande más rápido que su capacidad de procesamiento. El resultado de ello es que la empresa se ahoga en sus propios datos y el problema no es de hardware, porque éste sigue incrementando su capacidad y velocidad a pasos agigantados, la falla por consiguiente es del software. Dado que desarrollar software que iguale el potencial de las computadoras resulta ser un reto mucho mayor que el construir máquinas rápidas. Por otra parte tenemos que los equipos de cómputo cada vez son más accesibles tanto en tamaño como en costo mientras que el costo del desarrollo de software se ha incrementado.

Anteriormente decíamos que el poder computacional estaba limitado por los equipos que existían, pero hoy en día las cosas han cambiado y lo más importante con la existencia de nuevas herramientas de programación y los equipos actuales es la creación de software a la medida y a los requerimientos que el cliente o empresa disponga. Así como el hardware, también el software ha evolucionado y esto hace cambiar el proceso tradicional de la ingeniería de software tanto en el método y modo de analizar, diseñar y crear sistemas.

La aparición de la programación orientada a objetos y a eventos ha traído consigo una nueva forma de pensar y de desarrollar sistemas, es un nuevo concepto. Las nuevas interfaces gráficas que existen hoy en día hacen que los sistemas sean amigables, accesibles y atrayentes al usuario. Con lo cual nuestra tarea es crear sistemas y soluciones poderosas y que se ajusten a las necesidades de las organizaciones y a un precio razonable. Nuestro objetivo principal en el presente trabajo es desarrollar un sistema de información para una organización formada por agentes de fianzas (Broker), y para el cual se cuenta con una herramienta de cuarta generación en una plataforma comercial como lo es Windows en un ambiente de base de datos.

La estructura a seguir en este texto será el del ciclo de desarrollo del sistema, así que en el capítulo uno hablaremos de algunos antecedentes y empezaremos con la historia de la computación y cómo fue evolucionando a través de la historia, después veremos los sistemas basados en computadoras

para dar paso a el ciclo de vida clásico del software y poder explicar las técnicas de Cuarta Generación y para finalizar el capítulo explicaremos rápidamente los conceptos que involucra la programación orientada a objetos y a eventos, así como sus principales elementos.

El segundo capítulo nos introducirá un poco a lo que será el sistema de información, ya que veremos qué es el broker y qué es lo que realiza, así como sus principales tareas y operaciones, además conoceremos su situación y veremos por qué es necesario y conveniente realizar el sistema.

El análisis del sistema se realizará una vez conocido el objetivo del sistema. El análisis incluirá análisis de requerimientos, análisis estructurado y modelización de datos, además surgirán algunos productos intermedios como lo son la relación de requerimientos generales, los diagramas de flujo de datos, así como su diccionario de requisitos y diagramas de entidad relación. Lo anterior lo podremos ver en el capítulo tres que lleva como nombre Análisis del Sistema.

El Diseño del Sistema es tema del capítulo cuatro y éste se realizará una vez que se ha concluido el análisis, el diseño incluye: el diseño de datos y el diseño estructural, además de que ésto también arroja algunos productos intermedios como son el diccionario de datos, la relación de estructuras de las bases de datos, las especificaciones de la interface con el usuario y el prototipo del software.

En el quinto capítulo tendremos el Desarrollo del Sistema. Esta etapa es la de creación del producto de software que se dividirá en la programación y la creación de la base de datos. Para la parte correspondiente a la programación veremos la evolución que han tenido los lenguajes computacionales recalcando las características de los que se consideran de cuarta generación y además veremos algunas de las herramientas de cuarta generación con que se cuenta para la creación de software. Mientras que para la parte de base de datos analizaremos de manera general la teoría de bases de datos, sus elementos, el sistema de manejo de bases de datos y la arquitectura de un sistema relacional.

En el último capítulo se realizarán las pruebas y la puesta en marcha del sistema, dentro de la etapa de pruebas se explicarán las pruebas llamadas de caja blanca y de caja negra, así como la estrategia que se sigue para llevar a cabo las pruebas para el sistema desarrollado. En la parte referente a la puesta en marcha se especifican los cursos que se llevaron a cabo para los usuarios del sistema y la carga de información para el uso del mismo. Y para finalizar con el trabajo daremos a conocer las conclusiones y resultados finales del producto de software creado.

CAPÍTULO 1

Ingeniería de Software

Todo aquello que deseamos con pasión,
que creemos con sinceridad,
que guiamos con inteligencia,
que perseguimos con entusiasmo,
inevitablemente tiene que suceder.

Pensar es el trabajo más difícil que existe.
Quizá sea ésta la razón por la que haya
pocas personas que lo practiquen.

Henry Ford

CAPÍTULO 1

Ingeniería de Software

Es importante conocer algunos de los detalles principales y trascendente a través de la historia de la computación, ya que nos dará un enfoque y un panorama más general de lo que es la computación, la ingeniería de software y algunos detalles importantes para el entendimiento de los capítulos siguientes.

Como veremos en los párrafos siguientes la tecnología del software y del hardware han evolucionado a pasos agigantados, al comienzo el avance se enfocaba a mejorar la tecnología del hardware y a incrementar la capacidad de procesamiento y almacenamiento, pero una vez superado y alcanzado estos objetivos en la pasada década, la potencia del cómputo se centra en los sistemas personales que tengan un costo razonable y logren hacer la diferencia en los negocios y servicios.

El software entonces presenta el reto de mejorar la calidad y de reducir los costos de las soluciones basadas en computadoras, no sin olvidar que pese al avance logrado y a las posibilidades que hoy en día existen el conocimiento humano es la parte principal y fundamental de cualquier sistema y gracias a él las computadoras son una herramienta de gran utilidad en nuestra vida.

1.1 Historia de la Computación y del Software

El hombre desde el inicio de su evolución, vivió la necesidad de cuantificar sus pertenencias, así como el tiempo principalmente. Entonces se vió en la necesidad de crear técnicas y aparatos que le ayudaran a entender su entorno, su mundo, su medio ambiente, es decir la naturaleza.

1.1.1 Historia de la computación

Lo primero que utilizó el hombre para cuantificar objetos fueron sus propias manos, utilizando sus dedos, lo cual le fue en algunas ocasiones insuficiente por lo que ideó marcar los troncos de los árboles, huesos, cavernas, etc.

Entre los primeros elementos que el hombre desarrolló, se encuentra el ábaco, que se originó en el Oriente Medio, el ábaco es una calculadora completa y manual aunque su significado es el de tabla lisa cubierta de arena. El ábaco que actualmente conocemos aparece a fines del Imperio Romano y con el se pueden realizar las operaciones aritméticas básicas. Debido a la dificultad para realizar operaciones como la división y multiplicación llevaron a John Napier a

descubrir los logaritmos en 1614 aproximadamente y fue él quien ideó los llamados huesos de Napier, que son tablas de multiplicar marcadas con una varilla. En 1630 apareció la regla de cálculo, este instrumento trabaja a base de medir longitudes entre dos reglas que guardan relación entre sí, utilizando la escala logaritmica.

Se dice que la primera máquina de calcular verdadera fue construida por Wilhelm Schickard, la cual podía sumar, restar, multiplicar y dividir, pero se perdió en la guerra de los treinta años y su propio autor murió por la peste con lo que no pudo establecer su prioridad como inventor.

Pasaron 12 años más para que en 1642 Blase Pascal inventara una máquina de calcular llamada "Pascalina", aunque sólo podía sumar y restar. Cabe mencionar que a Blase Pascal se le reconoce como el primer inventor de una máquina de calcular.

Poco tiempo después Gottfried Wilhelm Leibniz mejoró el invento de Pascal y se imaginaba el día en que todo razonamiento fuera resuelto con solo darle vuelta a una manivela.

El siguiente paso importante se dió a mediados del siglo XVIII cuando se inventó un sistema para representar en tarjetas perforadas los dibujos de los tejidos aunque estas tarjetas eran utilizadas en los telares manuales y el tejedor leía directamente el dibujo de las tarjetas perforadas. Pero en el año de 1801 Joseph Marie Jacquard inventó en Francia un telar que leía automáticamente las tarjetas.

Por esas fechas en Gran Bretaña, la idea de Jacquard motivó al cerebro de Charles Babbage, a quien se le conoce como el padre de la computadora. Trabajó por muchos años en la máquina de diferencias la cual era una enorme calculadora mecánica que nunca pudo terminar.

Babbage generó otra idea y a su vez otra máquina a la que llamo "Máquina Analítica", se asemejaba bastante a las primeras computadoras; contaba con una sección de entrada (tarjetas perforadas), un analizador (molino, así lo llamó Babbage), una sección de salida (una máquina tipográfica capaz de imprimir los resultados de los cálculos), y una unidad de memoria. Hasta este punto Babbage tenia todo en papel y comenzó a buscar alguien que financiara su nuevo proyecto, así fue como conoció a Ada Augusta quien se convertiría en la primera programadora, ella escribió las primeras secuencias de instrucciones para la máquina analítica, inventó la subrutina, advirtió la importancia de una iteración y se imaginó el salto condicional. Pero tomando en cuenta que la máquina de diferencias nunca funcionó, el gobierno le negó todo apoyo para construir la máquina analítica.

Fue hasta un siglo después que se construyó la primera computadora. Entretanto hubo progresos en dos direcciones: por una parte las calculadoras mecánicas y por otra surgieron las máquinas perforadoras de tarjetas para efectos censales diseñadas por Herman Hollerit. Hasta antes de Hollerit, las oficinas de censos procesaban manualmente la información, en 1880 en Estados Unidos tardaron 7 1/2 años en obtener resultados, con la máquina de Hollerit se redujo a 2 1/2 años para reunir los datos.

Con el descubrimiento de la electricidad se facilitaron varios inventos, entre ellos el bulbo, elemento importante en el desarrollo con las computadoras y es precisamente con este elemento con el que se inicia la primera generación de computadoras, la figura 1.1 ilustra la tecnología usada en las generaciones de las computadoras.

La Mark 1 (1937-1944) se construyó en la Universidad de Harvard por Howard Eiken con el apoyo de IBM y fue la primera computadora electromecánica automática, contaba con aproximadamente 750,000 componentes con lo que disminuía considerablemente su velocidad y fue utilizada durante quince años para realizar cálculos astronómicos.

La ENIAC fue construida en la Universidad de Pennsylvania (1943-1945), no tenía partes mecánicas, en su lugar utilizaba alrededor de 18,000 bulbos, 70,000 resistencias y 10,000 capacitores y su peso era de aproximadamente 30 toneladas, era capaz de ejecutar 5,000 sumas o restas por segundo o 3,000 multiplicaciones o divisiones por segundo. Su mayor mérito consistía en tener gran cantidad de elementos electrónicos y hacerlos funcionar simultáneamente, pero por la gran cantidad de bulbos se calentaba en extremo en poco tiempo. Las computadoras anteriores pertenecieron a la primera generación de computadoras y cuyas características son su gran tamaño, utilizan bulbos, consumo de gran cantidad de energía y generaban muchísimo calor. Las unidades periféricas con que se contaban eran lectoras y perforadoras de tarjetas y cinta de papel. Los lenguajes existentes eran el lenguaje máquina y ensambladores primitivos además de que aun no existen sistemas operativos.

El siguiente avance tecnológico fue la sustitución de bulbos por transistores con lo que se redujeron las deficiencias, así como las implementaciones de ferritas que permitieron reducir el tamaño de las computadoras, así como su costo y temperatura. Todas las máquinas de transistores pertenecen a la segunda generación de computadoras y se utilizaban principalmente para el proceso de datos y como instrumento de cálculo. Se cuentan aun con lectoras y perforadoras de tarjetas, pero aparecen las impresoras y cintas magnéticas como dispositivos periféricos. Surgen los lenguajes ensambladores y los primeros compiladores (Fortran, Algol) y se cuenta con algunos sistemas operativos muy rudimentarios para el control de unidades periféricas e inicio y término de tareas. Algunos modelos típicos son CDC-160, IBM 7090, BURROGHS 5500, CDC-3600, etc.

La tercera generación de computadoras aparece con los circuitos integrados monolíticos, o sea, agrupaciones de transistores en pequeños componentes, los circuitos integrados aumentaron considerablemente la velocidad operacional, incrementando la confiabilidad y disminuyendo costo y tamaño, sus aplicaciones principales se centran en sistemas de información. Aparecen cintas y discos magnéticos, terminales de video y teletipos como unidades periféricas. La arquitectura empleada es en base a la multiprogramación, multiprocesamiento, sistemas de interrupciones y optimización de código. Los lenguajes empleados son de alto nivel, Cobol, PI/1, Bases de Datos (DMS). Modelos de la generación son IBM-360, BURROGHS 6700, UNIVAC 1106, CYBER 170.

La cuarta generación de computadoras cuenta con chips o circuitos integrados de mayor densidad que los de la tercera generación y ésta es la era de computadoras que vivimos actualmente. Las computadoras las empleamos principalmente en sistemas de comunicaciones, sistemas de información, para negocios, uso personal, etc. Hoy en día existen más dispositivos periféricos como las terminales inteligentes, equipos de graficación, lectores ópticos, digitalizadores, scanners, CD's, etc. Los lenguajes con los que se cuentan son lenguajes interactivos, descriptivos y gráficos procedurales, bases de datos distribuidas. Las máquinas de esta generación son la IBM 4330, UNIVAC 1100, MP 3100 VAX, APPLE, IBM-PC, etc.

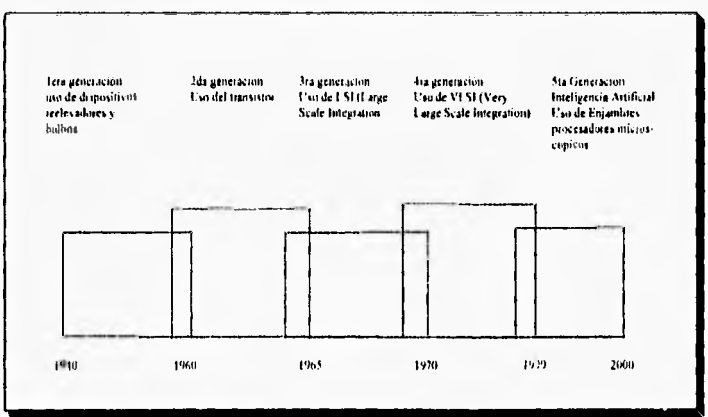


Figura 1.1 Generación de Computadoras

La quinta generación de computadoras se encuentra en pleno desarrollo, en las primeras potencias mundiales y consiste en el desarrollo de la inteligencia artificial en las computadoras. Estas máquinas se caracterizan por la utilización de enjambres que son procesadores microscópicos operando simultáneamente para recibir y clasificar información, por su capacidad básica de inferencia y generación

de conocimientos y esquemas generales a partir de información particular, así como su estrecha relación con el hombre.

1.1.2 Historia del Software

Empecemos definiendo lo que es el software. El software es un conjunto de instrucciones de programas de computadora que cuando se ejecutan proporcionan la función y el comportamiento deseado; también puede ser definido como estructuras de datos que facilitan a los programas manipular adecuadamente la información y documentos que describen la operación y el uso de los programas.

Como es de esperarse la evolución del software está ligado con la evolución de los sistemas de información, la figura 1.2 muestra la evolución que ha tenido el software. Al inicio el hardware era la parte más relevante en la computación y al software se le veía sólo como un elemento casi insignificante ya que el hardware se dedicaba a la ejecución de un solo programa que generalmente estaba orientado a ejecutarse por lotes y no existía ningún método para el desarrollo del software. Los sistemas estaban conformados con hardware de propósito general y software diseñado a la medida de la aplicación. Aun no existía el concepto de software como producto (programas desarrollados para ser vendidos a uno o más clientes).

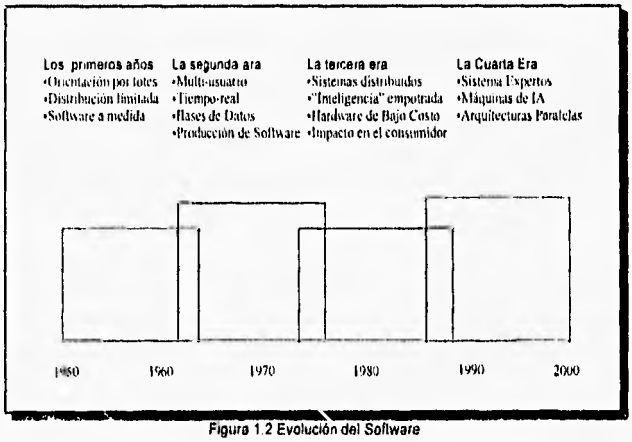


Figura 1.2 Evolución del Software

La segunda era se da a mediados de los sesenta y se extiende a finales de los setenta. Se introducen dos nuevos conceptos importantes, el de multiprogramación y el de sistemas multiusuario, además de que los avances tecnológicos de dispositivos de almacenamiento dan lugar a la primera generación

de sistemas de bases de datos. Un aspecto importante de esta era es que se ve al software como producto por primera vez y es así como nacen las primeras casas de software que lograron obtener grandes ganancias. Las empresas se vieron en la necesidad de modificar el software original debido a que tenía fallas y los cambios realizados fueron los requisitos que el usuario exigía o la implementación de nuevas tecnologías, a las actividades anteriores se les llamó mantenimiento de software y su costo alarmó a las empresas.

La tercera era fue la llegada de los microprocesadores y de las computadoras personales. El auge que han tenido las computadoras personales es debido a su gran capacidad y poder tanto de cómputo como de almacenamiento, así como su bajo costo que tienen. El Hardware de las computadoras personales se ha estandarizado, mientras que el software que hace la solución, marca la diferencia en los equipos de cómputo, los usuarios invierten más tiempo y dinero en el software que en el equipo donde se ejecuta éste.

La cuarta era es la que vivimos actualmente y el avance de la tecnología de software está revolucionando la forma de conceptualizar a los sistemas, así como sus técnicas de desarrollo. La programación orientada a objetos, las técnicas de cuarta generación, la inteligencia artificial, etc.; son ejemplos del avance que se tiene en la ingeniería de software.

1.1.3 Historia de los Métodos de Almacenamiento

Al ritmo que la tecnología ha avanzado resulta que ahora es más barato almacenar información en archivos de computadora que en papel y además no importa si la información está en forma escrita, en imágenes o sonidos. Al depósito de esta información dentro de un sistema se le llama base de datos.

La información nos interesa en demasía porque es el elemento más importante en cualquier sistema.

Una base de datos puede definirse como una colección de datos interrelacionados almacenados en conjunto sin redundancias perjudiciales o innecesarias, y su finalidad es la de servir a una aplicación o más, los datos se almacenan de modo que resulten independientes de los programas que los usan, empleándose métodos bien determinados para incluir datos nuevos y para modificar o extraer los datos almacenados.

La primera evolución en los métodos de almacenamiento se da en el inicio de la década de los setenta cuando los conceptos de archivos y conjuntos de datos se ven sustituidos por el nuevo concepto de bases de datos. Aparecen los primeros paquetes para la administración de datos y se popularizaron las

principales características de una base de datos: no redundancia, independencia de datos, interconectividad, protección de seguridad y accesibilidad.

La figura 1.3 ilustra la evolución de los métodos de almacenamiento. Así en la primera época, la estructura lógica manejada en la aplicación era igual a la organización física de los archivos, las unidades de almacenamiento eran principalmente cintas. Si era necesario hacer una modificación a la organización de los datos o se cambiaban los dispositivos de almacenamiento o se modificaba el programa totalmente. Como los archivos se diseñaban para soportar una sola aplicación existía un elevado nivel de redundancia.

La segunda época abarca el fin de los sesenta y es aquí cuando ya existe una diferencia entre la organización física y lógica de los archivos. Se crea la independencia entre las unidades de almacenamiento y los programas que las utilizan. Además se introduce el concepto de método de acceso a los registros de un archivo: secuencial, indexado y directo. Aún no existe la administración de datos.

A mediados de la década de los setenta surge la tercera etapa de la evolución de los métodos de almacenamiento. Los programas se hacen independientes no sólo del dispositivo de almacenamiento, sino también de la estructura física de los archivos, si se agregan nuevos datos a las estructuras las aplicaciones no sufren cambio alguno. La organización física es independiente de la vista lógica de los programas de aplicación. De los mismos datos físicos se derivan múltiples bases de datos lógicas por lo que las aplicaciones que tienen diferentes requerimientos pueden accesar los datos de diferentes maneras. Los métodos de acceso también evolucionan y existe la posibilidad de acceso a nivel de campo.

A fines de la década de los ochenta se inicia la cuarta era con el florecimiento de los sistemas de administración de bases de datos (DBMS). Existe una independencia total entre la estructura física y la estructura lógica de los datos. El software específico de la base de datos provee medios para la administración de los datos y se crea un lenguaje especial para la descripción de los datos usado por el administrador de datos, otro lenguaje de órdenes usado por las aplicaciones y un lenguaje de interrogación para el usuario además del ambiente gráfico que presenta lo hace bastante amigable y entendible.

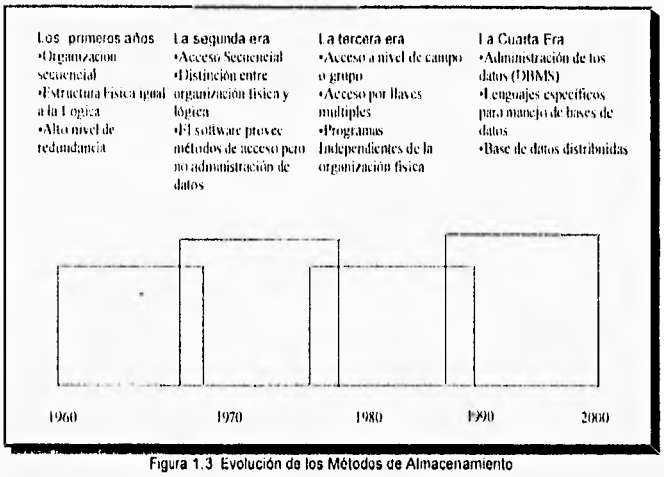


Figura 1.3 Evolución de los Métodos de Almacenamiento

1.2 Sistemas basados en computadoras

Existen varias definiciones de lo que es un sistema basado en computadoras pero una de las más completas es la siguiente: Un sistema basado en computadora puede definirse como un conjunto de elementos organizados para llevar a cabo algún método, procedimiento o control, mediante el procesamiento de información . En la figura 1.4 se muestran los elementos que intervienen en un sistema basado en computadoras.

La descripción de los elementos que intervienen en el sistema son las siguientes:
Documentación: Los manuales, los impresos, y demás información descriptiva que explica el uso y operación del sistema.

Bases de datos: Una colección grande y organizada de información a la que se accede mediante el software y que es parte integral del funcionamiento del sistema.

Procedimientos: Los pasos que definen el uso especifico de cada elemento del sistema.

Gente: Los individuos que son usuarios y operadores del hardware y software.

Hardware: Los dispositivos electrónicos y su configuración que proporcionan la capacidad de computación.

Software: Los programas de computadora, las estructuras de datos, y la documentación asociada que sirven para realizar el método lógico, procedimiento o control requerido.

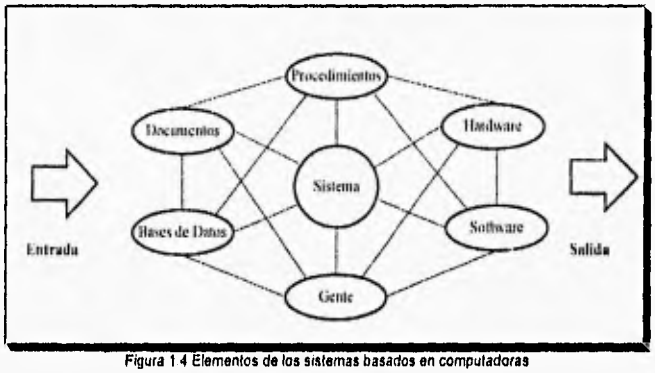


Figura 1.4 Elementos de los sistemas basados en computadoras

1.3 Modelos del Ciclo de Vida del Software

Existen diferentes procedimientos para la elaboración de sistemas y en este capítulo hablaremos de algunos de estos procedimientos.

Debido a que se debe de seguir alguna metodología para la elaboración de los sistemas para que sean entendibles y generales la ingeniería de software es la encargada específicamente del desarrollo de sistemas.

La ingeniería de software la podemos definir entonces como la disciplina preocupada por la producción sistemática y el mantenimiento de los productos de software. Para que los sistema tengan el éxito esperado la ingeniería de software debe abarcar tres elementos de suma importancia para que proporcionen el entorno ideal para construir software de buena calidad, estos elementos son: métodos, herramientas y procedimientos.

Los métodos nos indicaran como construir el software e incluyen las tareas de: planificación y costos del sistema, análisis de los requerimientos del sistema, diseño de estructuras de datos, arquitecturas de programas, procedimientos y bases de datos, codificación, prueba y mantenimiento.

Las herramientas suministran el soporte para los métodos. Se cuenta con herramientas para soportar cada uno de los métodos mencionados anteriormente.

Los procedimientos es lo que permite la unión de los métodos con las herramientas, definen la secuencia en la que se aplican los métodos, las entregas (documentos, informes, etc.) que se requieren y los controles que ayudan a asegurar la calidad del producto.

Existen varias formas de estructurar el proceso de ingeniería de softwaro y se denominan generalmente como modelos de ciclo de vida del software, dichas formas incluyen los tres elementos básicos vistos anteriormente. A continuación mencionaremos algunos de los modelos que nos servirán para la creación del sistema presentado en el correspondiente trabajo.

1.3.1 El Ciclo de Vida Clásico

También se le conoce como modelo en cascada o de fases y es un enfoque sistemático y secuencial de desarrollo de software en el que se sigue un estricto orden en las etapas. La figura 1.5 ilustra las etapas del ciclo de vida clásico. Las actividades específicas que conforman este modelo de desarrollo son:

Ingeniería y análisis del sistema: se establecen los requisitos de todos los elementos del sistema que se deberán cubrir en su forma general y más elemental.

Análisis de los requisitos del software: el proceso de recopilación se centra específicamente para el software; para comprender el ámbito de la información, la función, el rendimiento y las interfaces requeridas.

Diseño: traduce los requisitos en una representación de software y se enfoca sobre las estructuras de datos, los métodos de acceso, la arquitectura de las aplicaciones, el detalle procedimental la caracterización de la interfaz.

Codificación: es la traducción del diseño a un lenguaje de computadora.

Prueba: la prueba se centra en la lógica de los programas, en la eficiencia de los flujos de datos y en la interfaz con el usuario.

Mantenimiento: son los cambios que sufren los programas, estructuras de la base de datos e interfaces con el usuario después de que se libera el producto. Generalmente se aplica a todas las etapas anteriores para mantener un buen control sobre el sistema.

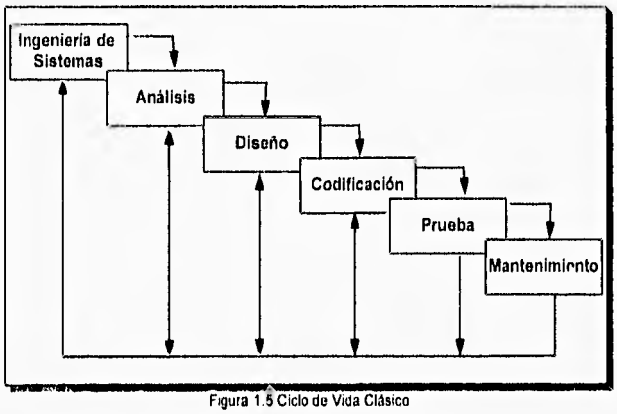


Figura 1.5 Ciclo de Vida Clásico

1.3.2 Construcción de Prototipos

Este modelo intenta cubrir algunas de las deficiencias que tiene el modelo clásico. Las etapas del modelo se presentan en la figura 1.6 y su descripción es la siguiente:

Recolección de requisitos: se identifican todos los requisitos conocidos y perfilan las áreas donde será necesaria una mayor definición.

Diseño Rápido: se enfoca principalmente a los aspectos de software visibles al usuario, como las formas de entrada y salida de los datos, los diálogos durante la ejecución, etc.

Creación y Evaluación de un Prototipo: se crea un prototipo ya que es la forma más sencilla de obtener los resultados solicitados por el cliente, quien posteriormente verifica si cumple con los objetivos establecidos y ayuda a refinar los requisitos del producto.

Refinamiento del Prototipo: es un proceso iterativo en el que el prototipo es refinado para que satisfaga las necesidades del cliente, al mismo tiempo que se facilita para el desarrollador, al tener una mejor comprensión de lo que hay que hacer.

Producto de ingeniería: es la etapa final y es nuestro sistema en funcionamiento.

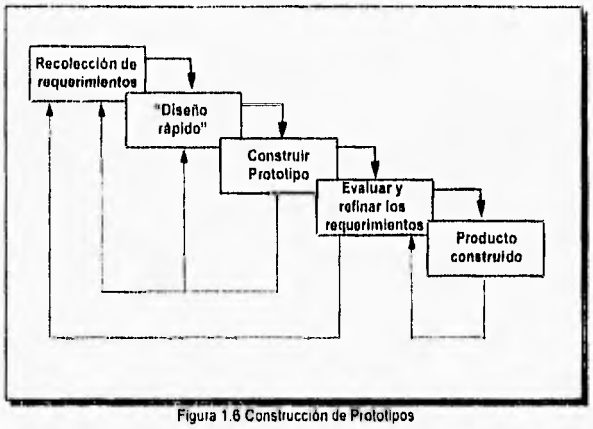


Figura 1.6 Construcción de Prototipos

1.3.3 Modelo en Espiral

El modelo en espiral es una combinación de los dos modelos anteriores, rescatando las principales ventajas de cada modelo visto anteriormente, se le llama modelo en espiral porque va de lo general al detalle, la figura 1.7 muestra el diagrama del modelo.

Las etapas del modelo son cuatro y su descripción es la siguiente:

Planificación: definimos los objetivos, metas, alternativas, y restricciones del sistema.

Análisis de Riesgos: se analizan las diferentes alternativas y la identificación y resolución de riesgos.

Ingeniería: es la etapa de desarrollo del sistema.

Evaluación del Cliente: se valoran los resultados de la ingeniería que se utilizó en la aplicación.

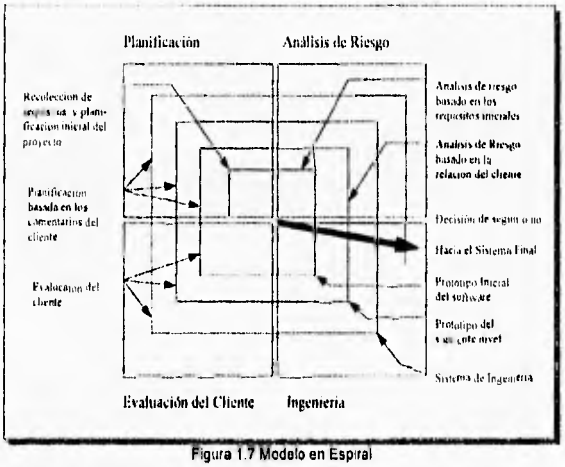


Figura 1.7 Modelo en Espiral

1.4 Técnicas de Cuarta Generación

Las técnicas de cuarta generación (T4G) pertenecen a los modelos del ciclo de vida del software aunque por su aparición reciente los contemplamos como un tema aparte.

El término técnicas de cuarta generación sugiere que las herramientas de software empleadas faciliten al desarrollador de software la especificación de algunas características del software a un alto nivel. Para que posteriormente la herramienta genere el código fuente de manera automática, así como la estructura de la base de datos a partir de las especificaciones dadas.

El modelo de cuarta generación para la ingeniería de software se orienta hacia la habilidad de especificar software a un nivel que sea más próximo al lenguaje natural o en una notación que proporcione funciones significativas.

Las técnicas de cuarta generación contemplan el uso de algunas o de todas las siguientes herramientas que se mencionan a continuación: lenguajes no procedurales tanto para la creación como para la consulta de bases de datos, interacción y definición de pantallas, generación de formas, manipulación de datos, generación de código, gráficas de alto nivel y facilidades de hoja de cálculo.

La figura 1.8 muestra las etapas de las T4G y la descripción de las etapas es la siguiente:

Recolección de Requerimientos: el cliente debe describir los requerimientos y éstos deben traducirse directamente a un prototipo operacional.

Estrategia de Diseño: se elige una estrategia que permita definir un modelo cualitativo de manera que facilite su desarrollo y codificación.

Implementación en L4G: la implementación usando un lenguaje de cuarta generación no procedimental (L4G) facilita al que desarrolló el software, la descripción de los resultados deseados, los cuales se traducen automáticamente en el código fuente para producir dichos resultados.

Producto: Para transformar una implementación T4G en un producto, él que lo desarrolla debe dirigir una prueba completa, desarrollar una documentación con sentido y ejecutar todas las otras actividades de transición requeridas en los otros modelos.

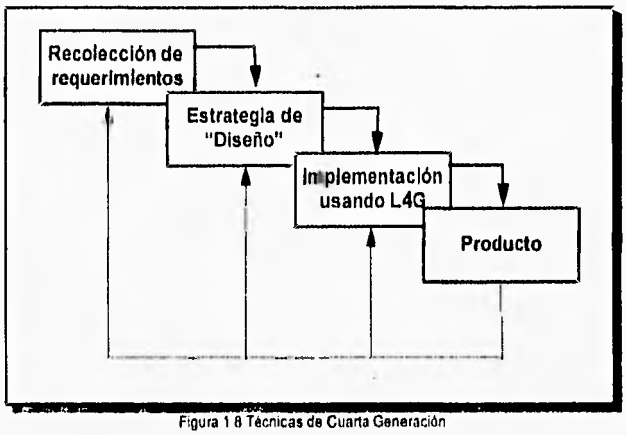


Figura 1.8 Técnicas de Cuarta Generación

Las técnicas de cuarta generación se convertirán probablemente en una parte importante del desarrollo de software durante la década presente y la demanda de software continuará creciendo durante el resto del siglo, pero los métodos convencionales, contribuirán probablemente cada vez menos al desarrollo del mismo. Las técnicas de la cuarta generación rellenarán el hueco que existe hasta que sean practicables los métodos de Inteligencia Artificial (IA) de la quinta generación.

Para fines del presente trabajo se realizará una combinación de metodologías, para intentar aprovechar los beneficios y ventajas que ofrecen las técnicas de cuarta generación y la generación de prototipos.

1.5 Programación Orientada a Objetos y a Eventos

Necesitamos un nuevo enfoque para construir software, uno que deje atrás los métodos de la programación convencional y ofrezca una mejor forma de construir sistemas pequeños y de gran escala que sean confiables, flexibles, mantenibles y capaces de evolucionar para satisfacer los requerimientos del cambio. A continuación veremos la propuesta que ofrece la tecnología de Programación Orientada a Objetos (POO) y por qué tiene el potencial de tener éxito.

En la POO se manejan varios conceptos como: clase, objeto, mensaje, abstracción, encapsulado, polimorfismo, herencia, etc. Sin embargo resulta confuso hablar de ellos separadamente, por lo que lo relacionamos en tres partes modulares de la siguiente manera.

1) Objetos
   - Abstracción
   - Encapsulado

2) Mensajes
   - Reutilización de Nombre
   - Polimorfismo

3) Clases
   - Modularidad
   - Objetos Compuestos
   - Herencia

El cuadro anterior representa la estructura para presentar los concepto de la POO.

1.5.1 Objetos

Se puede pensar en los sistemas orientados a objetos como una colección de dispositivos virtuales independientes que se comunican entre sí, cada uno de ellos con su propia estructura interna. A estos dispositivos virtuales se les llama objetos.

Un objeto es la abstracción de una entidad tangible del mundo real o de un concepto teórico. Un objeto tiene estado, comportamiento e identidad. Entonces un objeto puede ser: un avión, un automóvil, una lista ligada, una casa, etc.

Un objeto se caracteriza por tener estado (peso, posición, altura) y comportamiento (aumentar velocidad, moverse de lugar). El modificar el comportamiento provoca ya sea la ejecución de alguna acción, o el cambio en el estado del objeto. Un objeto tiene identidad porque es único y diferente a cualquier otro.

El estado de un objeto se modela a través de un conjunto de valores variables, que representan la abstracción del objeto y sus características en el tiempo y a los cuales llamamos datos. Por otra parte, el comportamiento se modela a partir de un conjunto de funciones que llevan a cabo ciertas tareas que modifican el estado del objeto y se les conoce como métodos. Los datos y los métodos se manejan como un todo inseparable.

Los datos internos de un objeto son inaccesibles a cualquier otro objeto o programa. Este es uno de los conceptos principales en la POO: el encapsulado.

En la figura 1.9 se muestra como los métodos del objeto protegen a sus datos de ser alterados por cualquier elemento externo. La única forma de modificar el estado del objeto es a través de un método, manteniendo así la integridad de los datos.

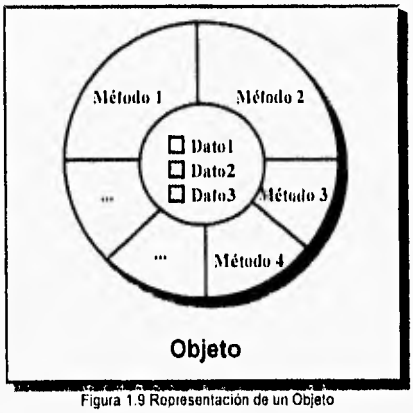


Figura 1.9 Representación de un Objeto

1.5.2 Mensajes

Los objetos se comunican entre sí a través de mensajes como se ilustra en la figura 1.10. Los mensajes modifican el comportamiento del objeto receptor, provocando que realice cierta opción o que cambie su estado. Por ejemplo, crear un nuevo objeto, preguntar por el valor de un dato, etc.

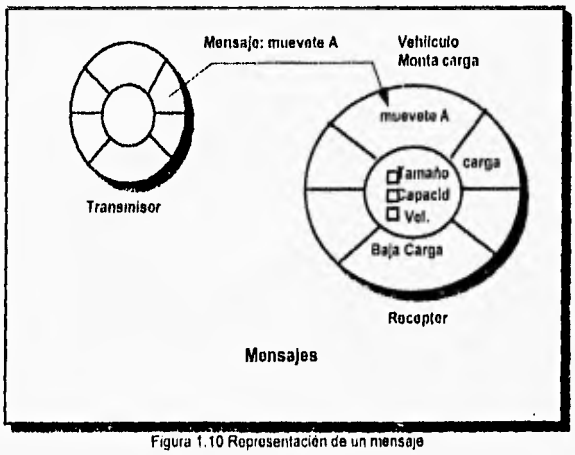


Figura 1.10 Representación de un mensaje

El concepto de polimorfismo es muy importante en la POO, ya que se refiere a que cada objeto reacciona de manera muy particular a un mensaje que se le envía.

Por ejemplo: todos los animales comen , pero todos lo hacen de manera diferente. Entonces si pongo otro animal, el sabrá como comer, enviándole solo el mismo mensaje: comer. De esta manera el flujo normal del programa no cambia, volviéndose mucho más sencillo y de fácil mantenimiento.

1.5.3 Clases

Una clase es la definición abstracta de un grupo de objetos con datos y métodos comunes.

Los objetos se crean mediante las instancias de una clase. Mientras que un objeto es la identidad concreta que existe en el tiempo y el espacio, una clase sólo representa una abstracción, la esencia del objeto.

Una clase define las propiedades y métodos que comparten todas sus instancias, pero cada objeto tiene sus propios valores únicos; como se muestra en la figura 1.11, cada montacargas tiene un tamaño, capacidad, velocidad máxima y contenido específico. Aunque cada objeto tiene sus propios datos, sí comparte los métodos de la clase con los demás objetos.

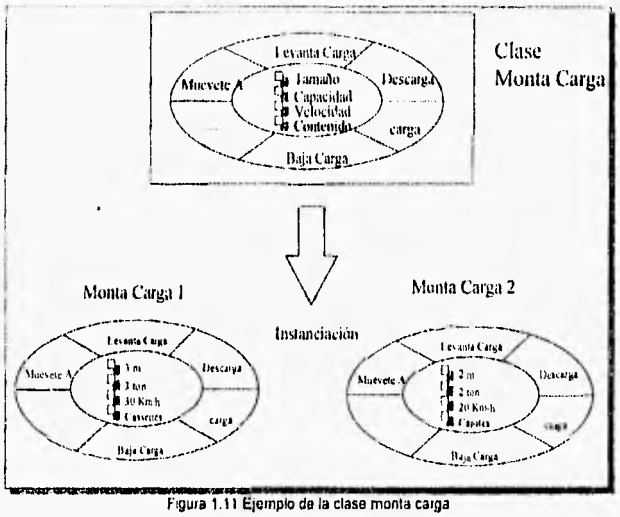


Figura 1.11 Ejemplo de la clase monta carga

Un objeto siempre es la instancia de una clase, pero una clase bien puede no producir objetos. A este tipo de clases que sólo sirven para expresar características comunes a un grupo de clases, se le llama clase abstracta y cobra sentido en una jerarquía de clases.

Las clases traen orden a los objetos organizándolos en jerarquías de clases -herencia- y en objetos compuestos -objetos que contienen otros objetos-. Estos conceptos se explican más adelante.

1.5.4 Métodos de Organización

De acuerdo con Yourdon y Coad, los métodos de organización que existen son tres:

1) Identidad. Identificar los objetos por medio de sus atributos (estado) y comportamiento. Por ejemplo un león

2) Organización Parte De (Objetos Compuestos). Descomponer al objeto en las partes que lo forman. Por ejemplo un león tiene ojos, patas y cabeza.

3) Organización Tipo De (Jerarquía de Clases). Identificar la relación que tiene con otros objetos de características semejantes. Por ejemplo el león pertenece a los felinos.

La aplicación que tienen estos métodos en la POO los presentamos a continuación.

El punto 1. Identidad, se explicó en el tema de los objetos, donde la actividad más importante es la de identificación y definición de los objetos.

Los puntos 2 y 3 están Intimamente relacionados con las clases y los utilizamos para simplificar nuestra apreciación del mundo.

Organización Parte De. Cuando le explicamos a un niño pequeño las partes que tiene un automóvil, le decimos : Un automóvil tiene puertas, ruedas, volante y motor.

La organización Parte De explica al objeto en término de sus componentes. A este tipo de objetos se les denomina objetos compuestos.

Organización Tipo De. Tomando el ejemplo anterior, cuando hacemos la descripción a un niño pequeño de lo que es un sofá, asumiendo que el niño sabe lo que es una silla, no le explicamos desde la nada, sino que le decimos: un sofá es como una silla, sólo que más cómodo y más grande.

Las clases actúan como una plantilla agrupando objetos relacionados entre sí con propiedades y comportamientos similares.

En la organización Tipo De, las clases se ordenan de una forma jerárquica en una taxonomía de superclases (padres) y subclases (hijos). Es precisamente esta taxonomía la que define la jerarquía de clases.

Con las subclases es posible definir objetos más especializados, conforme desciende la jerarquía. Las superclases expresan características comunes entre clases. entre más cerca nos encontremos de la raíz, las clases son más generales; entre más cerca a las hojas las clases son más particulares.

1.5.5 Herencia

En términos generales una subclase hereda todos los datos y métodos de la superclase a la que pertenece. La subclase puede adicionar datos y métodos a los que heredo de su padre, aunque también puede modificar los métodos heredados. La figura 1.12 muestra el concepto de jerarquía de clases y de herencia. Todos los transportes tienen un cierto tipo de combustible, una capacidad de carga y transportan bienes o personas de un lugar a otro. Todas las subclases de transporte heredan estas características.

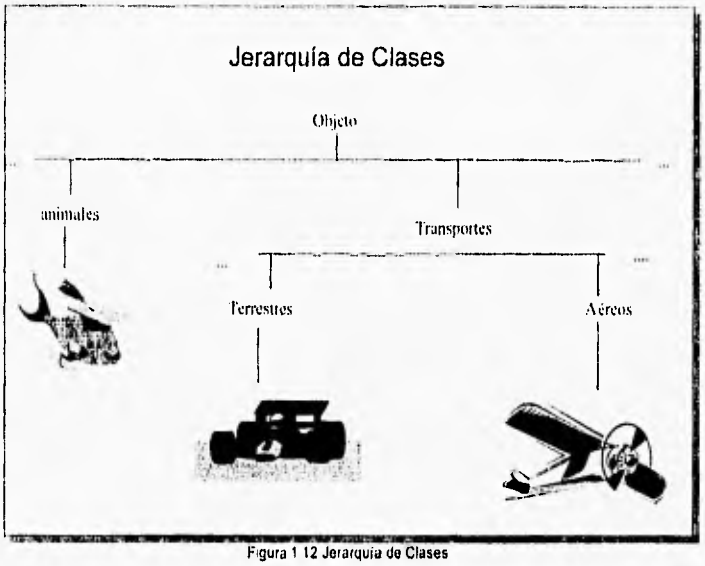


Figura 1.12 Jerarquía de Clases

Además de la rama de transportes, todos los transportes terrestres comparten algunas características comunes, de la misma manera que lo hacen los transportes aéreos. Los transporte terrestres se mueven en tierra firme, mientras que los aéreos por aire.

Un ejemplo de las ventajas de la herencia es que si se desea otra rama de transporte, por ejemplo transportes marítimos, su definición es mucho más fácil porque ya tenemos todas las características que comparten los transportes. Solo necesitamos definir la información especifica sobre esta rama, como por ejemplo, que los transportes marítimos se desplazan en el agua.

La herencia constituye la principal forma de reutilizar código en la programación orientada a objetos.

Con el resumen anterior esperamos haber dado una explicación de lo que es la POO, así como dar a conocer el nuevo concepto que involucra y la nueva forma de ver las cosas, ya que es una forma más natural.

1.3.6 Programación Orientada a Eventos

La programación orientada a eventos está basada en lo que es la programación orientada a objetos. Los objetos usados son los componentes que frecuentemente aparecen en la interfaz gráfica de Windows, estos objetos son por ejemplo las ventanas, los botones, los botones radiales, las cajas de listas, las

líneas de edición, los textos, estáticos, etc. Mientras que los métodos asociados a los objetos, son precisamente los eventos. Entre los diferentes eventos que encontramos están la creación del objeto, la destrucción del objeto, la modificación del texto, el hacer click sobre el objeto, hacer doble click, etc.

Como podemos darnos cuenta la programación orientada a eventos es fácilmonte explicada, usando la filosofía que ofrece la programación orientada a objetos para hacer una analogía entre ambas.

Es importante explicar y señalar lo que es la programación orientada a eventos, porque este tipo de programación es la que se utiliza para el desarrollo del sistema.

CAPÍTULO 2

Antecedentes del Broker

Lo importante no es lo que nos hace el destino,
sino lo que nosotros hacemos de él.

Florence Nightingale

Hay que estudiar mucho para saber poco

Barón de Montesquieu

CAPÍTULO 2

Antecedentes del Broker

En este capítulo trataremos de explicar qué es un broker, cuáles son las actividades que realiza un broker, así como la necesidad de por qué es necesario hacer un sistema de información que les ayude a simplificar su trabajo.

2.1 ¿Qué es un Broker y qué realiza?

Un Broker es una compañía o un grupo de agentes que se dedica a vender fianzas y seguros, bajo el consentimiento de las compañías afianzadoras o de seguros a las que pertenecen y están registrados. Nuestro broker en cuestión es una empresa llamada GOSAC.

GOSAC es una empresa de servicios por lo que su estructura organizacional, sus estrategias de venta, su operación y su gente deben orientarse a obtener la excelencia en el servicio a los clientes, de la misma manera cualquier sistema se debe crear para apoyar a la empresa en el logro de este objetivo primario.

2.2 Antecedentes de la empresa

2.2.1 ¿Qué es GOSAC?

GOSAC, S.C. es una empresa fundada en 1991 y cuya función principal es la de brindar asesoría y servicio en seguros y fianzas, ésto es, recibe el requerimiento de un cliente, lo asesora en relación al tipo de seguro o fianza que necesita y realiza todas las gestiones necesarias con las compañías autorizadas (Aseguradoras o Afianzadoras) hasta otorgarle el producto que satisfaga sus necesidades.

Con el objeto de tener una operación eficiente que se traduzca en un buen servicio, la empresa se ha dividido en dos áreas, el área de fianzas y el área de seguros.

En el presente trabajo se pretende hacer un análisis de la operación del área de fianzas por lo que en lo sucesivo se hará referencia a ésta.

En los últimos meses el volumen de ventas de fianzas se ha visto incrementado notablemente y si bien por un lado ésto es un claro síntoma de que el servicio que se ha venido dando ha traído negocios, por otro lado, provoca que los sistemas que han apoyado este servicio deben sufrir cambios que les permitan hacer frente a las nuevas características del medio. Vale la pena mencionar en este punto que un sistema que apoye la operación de una empresa, cualquiera que sea el giro de ésta, no necesariamente debe ser un sistema automatizado o basado en computadoras.

Antes de continuar es conveniente saber que el producto que la compañía ofrece es la fianza de empresa, la cual es un contrato accesorio en la que la afianzadora garantiza el cumplimiento de una obligación, estipulada en un contrato principal, por parte de un fiado a favor de un beneficiario.

2.2.2 Recursos Humanos

La operación del área de fianzas está soportada por cinco personas y no obstante no existe un perfil definido para cada uno de los puestos, las actividades que se realizan están bien determinadas y distribuidas entre los colaboradores.

En términos generales el organigrama del área puede definirse de la siguiente manera en la figura 2.1:

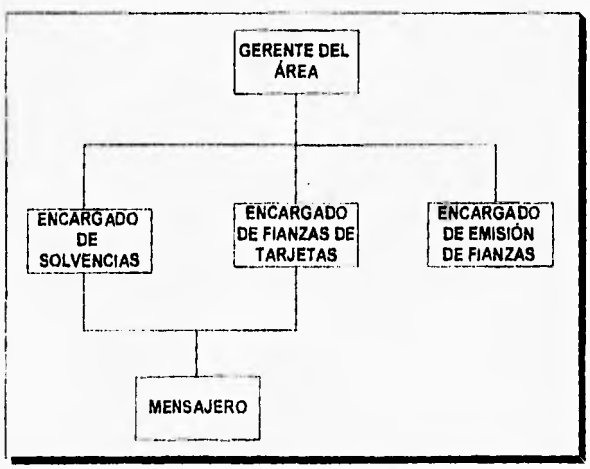


Figura 2.1 Organigrama del área de fianzas

2.2.3 Recursos de Cómputo

- Hardware

El área de fianzas cuenta con 3 computadoras personales y una impresora con las siguientes características:

Computadora 386 DX con 4 Megabytes en memoria RAM
Computadora 486 DX con 4 Megabytes en memoria RAM
Computadora 486 DX con 8 Megabytes en memoria RAM
Impresora Láser Jet II P Hewlett Packard

- Software

Sistema operativo MS-DOS ver 6.0
Windows ver 3.1
Paquete MicroSoft Office
Paquete WORKS
Sistema de emisión de fianzas de Afianzadora Insurgentes
Sistema de emisión de fianzas de Crédito Afianzador
Sistema de control de fianzas de tarjeta de crédito

2.2.4 ¿Por qué un sistema automatizado?

Como se mencionó anteriormente, en los últimos meses se ha incrementado el volumen de ventas de fianzas, esta situación puede provocar que los sistemas que han venido dando soporte a la operación se vean rebasados y en especial cuando dichos sistemas están apoyados en algunos puntos en controles manuales o en algunos dependen de la memoria de alguna persona en particular, haciéndose esta persona indispensable en todo momento para la correcta operación, situación que le impide el desarrollo de actividades de otra índole tales como la administración, promoción y en general actividades de carácter gerencial.

Por otra parte existen un gran número de empresas dedicadas a la actividad que desempeña GOSAC, con lo cual la competencia es muy grande. Entonces en empresas de servicio como es el caso de GOSAC la diferencia la dará precisamente la calidad en el servicio que ofrezcan a sus clientes, este servicio debe estar soportado en sistemas que proveen de información inmediata y confiable a las tareas de toma de decisiones.

2.2.5 Situación Actual

Como se mencionó anteriormente, GOSAC, S.A. es una empresa que proporciona servicio y asesoría tanto en seguros como en fianzas. En el ramo de fianzas, GOSAC trabaja actualmente con tres compañías:

1. Afianzadora Insurgentes, S.A.
2. Crédito Afianzador
3. Afianzadora Capital

La fuerza de venta de fianzas está representada por cuatro agentes:

1. Alejandro Gómez Oliver
2. Jorge Salas Castro
3. Manuel Zepeda Muñoz
4. Paulina Medina Mora

Dentro de las fianzas que GOSAC, S.A. ofrece a sus clientes se pueden identificar tres productos los cuales son :

1. Fianzas tradicionales
2. Fianzas de tarjetas de crédito
3. Fianzas de Proveedores

Una vez visto a lo que se dedica la empresa y sabiendo que nos enfocaremos a la área de fianzas explicaremos entonces el elemento básico del sistema en cuestión, la fianza y su ciclo de vida.

Una fianza es una garantía otorgada por una sociedad anónima, concesionada por el gobierno para otorgar todo tipo de fianzas y cobrar por este servicio, pudiendo comercializar sus ventas por agentes comisionistas, operando bajo normas técnicas y apegadas a leyes y reglamentos, y teniendo una capacidad de asumir responsabilidades por su solvencia económica y su solvencia técnica operacional.

Para explicar y entender el ciclo o proceso de vida de una fianza el ejemplo de la figura 2.2 nos ayudará a entenderlo.

Supongamos que la Secretaría de Comunicaciones y Transportes (beneficiario) necesita construir un tramo de carretera por lo que suscribe un contrato para la ejecución de la obra con una compañía constructora (fiado). Entonces para iniciar la obra la constructora solicita un anticipo, sin embargo, la Secretaría de Comunicaciones y Transporte requiere que se le garantice la inversión del anticipo en la obra, por lo que exige una fianza. La compañía constructora contacta a GOSAC y solicita la fianza. GOSAC evalúa la operación y la solvencia del fiado para finalmente emitir la póliza de la fianza en la que garantiza a la Secretaría que el fiado utilizará el anticipo en la obra, de lo contrario le pagará al beneficiario el monto de la fianza. GOSAC le cobra al fiado por la fianza una prima anual para posteriormente entregarla a la compañía afianzadora y así la afianzadora le pague la comisión debida a GOSAC por la colocación de la fianza. La fianza permanece vigente hasta que el beneficiario le comunica a

1. Afianzadora Insurgentes, S.A.
2. Crédito Afianzador
3. Afianzadora Capital

La fuerza de venta de fianzas está representada por cuatro agentes:

1. Alejandro Gómez Oliver
2. Jorge Salas Castro
3. Manuel Zepeda Muñoz
4. Paulina Medina Mora

Dentro de las fianzas que GOSAC, S.A. ofrece a sus clientes se pueden identificar tres productos los cuales son :

1. Fianzas tradicionales
2. Fianzas de tarjetas de crédito
3. Fianzas de Proveedores

Una vez visto a lo que se dedica la empresa y sabiendo que nos enfocaremos a la área de fianzas explicaremos entonces el elemento básico del sistema en cuestión, la fianza y su ciclo de vida.

Una fianza es una garantía otorgada por una sociedad anónima, concesionada por el gobierno para otorgar todo tipo de fianzas y cobrar por este servicio, pudiendo comercializar sus ventas por agentes comisionistas, operando bajo normas técnicas y apegadas a leyes y reglamentos, y teniendo una capacidad de asumir responsabilidades por su solvencia económica y su solvencia técnica operacional.

Para explicar y entender el ciclo o proceso de vida de una fianza el ejemplo de la figura 2.2 nos ayudará a entenderlo.

Supongamos que la Secretaría de Comunicaciones y Transportes (beneficiario) necesita construir un tramo de carretera por lo que suscribe un contrato para la ejecución de la obra con una compañía constructora (fiado). Entonces para iniciar la obra la constructora solicita un anticipo, sin embargo, la Secretaría de Comunicaciones y Transporte requiere que se le garantice la inversión del anticipo en la obra, por lo que exige una fianza. La compañía constructora contacta a GOSAC y solicita la fianza. GOSAC evalúa la operación y la solvencia del fiado para finalmente emitir la póliza de la fianza en la que garantiza a la Secretaría que el fiado utilizará el anticipo en la obra, de lo contrario le pagará al beneficiario el monto de la fianza. GOSAC le cobra al fiado por la fianza una prima anual para posteriormente entregarla a la compañía afianzadora y así la afianzadora le pague la comisión debida a GOSAC por la colocación de la fianza. La fianza permanece vigente hasta que el beneficiario le comunica a

GOSAC y ésta a su vez a la compañía afianzadora que la obligación objeto de la fianza se ha cumplido, para así proceder a su cancelación.

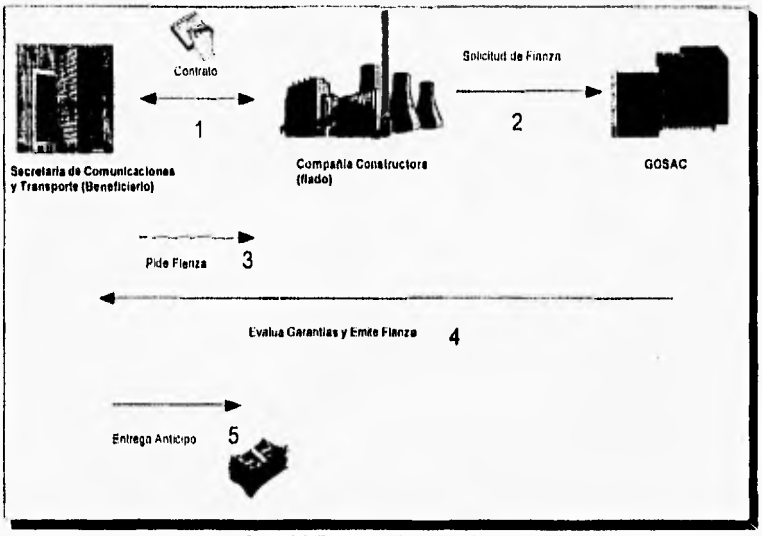


Figura 2.2 Proceso de vida de una fianza

Mientras que una fianza esté vigente es muy probable que a petición del beneficiario, el fiado tramite con GOSAC algunas modificaciones a las características originales de la fianza, a estas modificaciones se les denomina movimientos de la fianza y los que se manejan son:

1.- Expedición: toda fianza nace con este movimiento, es decir, es el inicio de la fianza; GOSAC emite una póliza de fianza garantizando el cumplimiento de una obligación por el fiado ante el beneficiario hasta un monto determinado.

2.- Renovación: si al expirar la vigencia de la fianza no se ha cumplido totalmente con la obligación, el beneficiario le exige al fiado que el beneficiario renueve la fianza por lo que se emite una póliza por renovación y se cobra una prima por este movimiento en función del monto y del periodo de renovación.

3.- Aumento del monto: dentro de la vigencia de la fianza es probable que se incremente la obligación del fiado ante el beneficiario. Entonces el fiado solicita un aumento de el monto de la fianza por el periodo de vigencia que falta transcurrir. GOSAC cobra entonces la prima por el aumento en función del incremento del monto y del periodo de tiempo de vigencia que falta por cubrir.

4.- Disminución del monto: es lo contrario al movimiento anterior, con la diferencia de que se emite un endoso de disminución pero sin cobro ni devolución de prima.

5.- Anulación de Movimiento: el fiado puede solicitar una anulación de movimiento debido a un error tipográfico o porque la fianza ya no se necesita. En este caso se devuelve la prima cobrada por el movimiento que se anula.

6.- Cancelación de la fianza: una vez que el fiado ha cumplido con la obligación ante el beneficiario, la fianza puede ser cancelada.

En todos los movimientos de fianza que se cobra prima al fiado, la afianzadora le paga una comisión a GOSAC por colocar el movimiento de la fianza, esta comisión es variable y depende del agente, el tipo de fianza y el movimiento que se está cobrando.

2.3 El Proyecto Actual

El proyecto que se tiene que realizar es un sistema que controle las operaciones de las fianzas, proporciones información en forma de reportes y en forma gráfica de los estados de las fianzas, así como de todo lo que involucra ésta.

La operación que exige el control de las fianzas pueden dividirse, para un mejor entendimiento, en siete actividades principales.

1. Control de folios
2. Control de solvencias
3. Emisión y registro de fianzas
4. Cobranza de primas de fianza
5. Pago de primas de fianza
6. Cobranza de comisiones
7. Cancelación de fianzas

2.3.1 Control de folios

Las compañías afianzadoras proveen de folios en blanco a GOSAC con el fin de agilizar el trámite y otorgamiento de la fianza al cliente que la solicite.

Eventualmente las afianzadoras realizan auditorías de los folios distribuidos con el fin de llevar un control de los folios no utilizados, así como de los que por alguna razón se inutilizaron.

Desde el momento en que GOSAC recibe los folios por parte de cualquiera de las afianzadoras se hace responsable de su custodia y de su buen uso; ésto ha dado lugar a establecer controles manuales donde se registran cada uno de los folios recibidos, su monto de protectoración, la fecha en que se recibieron, y cuando son utilizados se registra la fianza con que se usa, el fiado, el monto de la fianza y la fecha en que se usó, en caso de que un folio se inutilice se marca en el control manual como inutilizado.

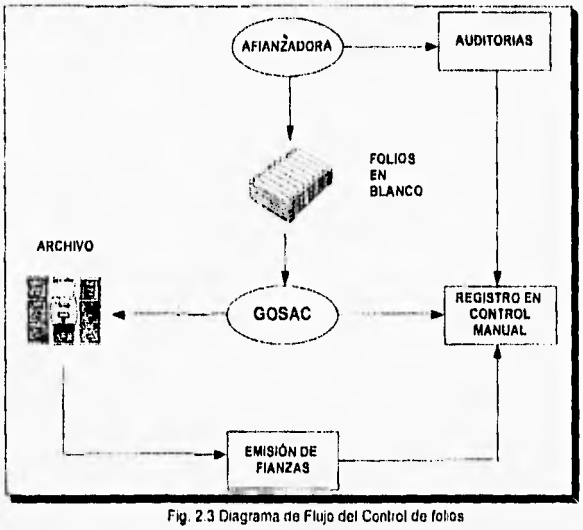


Fig. 2.3 Diagrama de Flujo del Control de folios

2.3.2 Control de solvencias

Toda fianza otorgada a un cliente debe estar soportada por garantías que cubran en un momento dado la reclamación de la fianza debido a un incumplimiento del cliente.

GOSAC es responsable de elaborar el documento de solvencia de cada cliente que le solicite una fianza y en el caso de clientes recurrentes, de mantener actualizado dicho documento con el objeto de que todas sus fianzas en vigor están cubiertas.

Todo este proceso de solvencia se lleva de forma manual y cada vez que se solicita una fianza de un cliente recurrente se revisa su documento de solvencia, se calcula el monto de su fianza en vigor y se decide si se otorga la fianza en ese momento o se le pide aumento a la solvencia.

En el caso de los clientes más recurrentes, en función de su solvencia se le asigna una linea de crédito que se irá otorgando mientras solicite fianzas y que se irá liberando mientras haya cancelaciones de sus fianzas.

En ocasiones el proceso para determinar el otorgamiento de un rechazo de una fianza resulta lento ya que hay que revisar en el archivo tanto la solvencia como las fianzas en vigor del cliente.

Cuando el cliente cae en un incumplimiento de la obligación establecida en la fianza, el beneficiario presenta la reclamación directamente en la compañía afianzadora y ésta solicita a GOSAC el documento de solvencia del cliente para dar inicio al proceso de recuperación de la reclamación.

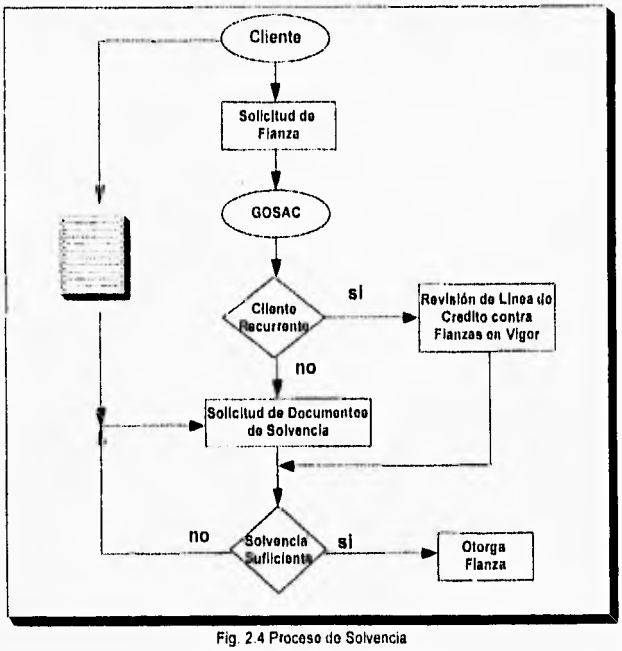


Fig. 2.4 Proceso de Solvencia

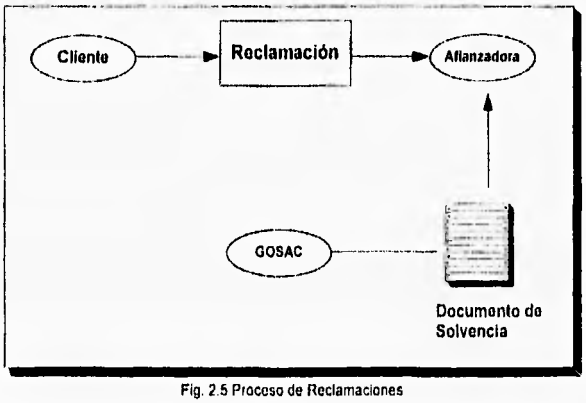


Fig. 2.5 Proceso de Reclamaciones

2.3.3 Emisión y registro de fianzas

Una vez autorizado la fianza el paso que sigue es emitir la fianza con los datos e información previamente recabada, aunque ésto resulta laborioso dado que se tienen que calcular los porcentajes de la prima en función del monto y del tipo de fianza.

Posteriormente se registra en los archivos, el número de folio de la fianza, qué agente la vendió, el monto, la prima y su comisión, así como las fechas de inicio y fin de vigencia.

2.3.4 Cobranza de primas de fianza

La compañía por medio del agente que vendió la fianza tiene que cobrarle al cliente la prima de la fianza para posteriormente notificar y registrar en la compañía que la prima ya fue cobrada.

2.3.5 Pago de primas de fianza

Una vez cobrada la prima al cliente, GOSAC tiene que llevar la suma de las primas de las fianzas vendidas a la afianzadora respectiva, y tiene que registrar en sus archivos que las primas ya fueron pagadas a las afianzadoras.

2.3.6 Cobranza de comisiones

Como GOSAC ya vendió la fianza, cobró la prima al cliente y pagó a la afianzadora la prima del movimiento, entonces GOSAC se ha hecho acreedor a cobrarle la comisión respectiva del movimiento así que tiene que dirigirse a la

afianzadora y cobrar su comisión para registrar en sus archivos que se cobró el movimiento.

2.3.7 Cancelación de fianzas

Se revisan las fechas de fin de vigencia de las fianzas vendidas por GOSAC, y si ya se cumplió con el contrato contraído el último paso es cancelar la fianza y cerrar la operación con ella y la afianzadora respectiva.

El nuevo sistema se basa en la automatización de los módulos anteriormente vistos por lo que la empresa requiere un sistema eficiente que maneje lo anterior más otro módulo que genere reportes de los agentes que interviene en las ventas, así como reportes que den información más general y detallada de los aspectos que involucran una fianza para hacer más eficiente el servicio a los clientes que recurren a GOSAC.

Para realizar el sistema se cuenta con la herramienta de cuarta generación Power Builder que contiene su propio manejador de bases de datos WATCOM.

El presente trabajo contiene la documentación de las etapas que integran el desarrollo del producto de software que contempla los módulos anteriormente mencionados, su instalación y el inicio de operaciones en GOSAC.

CAPÍTULO 3

Análisis del Sistema

Recuerda que eres tan bueno
como lo mejor que hayas
hecho en tu vida

Billy Wilder

Si uno no puede explicar lo que
está haciendo, su trabajo
carecerá de valor

Erwin Schrodinger

CAPÍTULO 3

Análisis del Sistema

Para que un sistema tenga un buen desarrollo de software es esencial realizar una especificación completa de los requerimientos de los mismos. La tarea del análisis de los requerimientos es un proceso de descubrimiento y refinamiento. El ámbito del programa, establecido inicialmente durante la ingeniería del sistema, es refinado en detalle. Se analizan y asignan a los distintos elementos de los programas las soluciones alternativas.

Tanto el que desarrolla el software como el cliente tienen un papel activo en la especificación de los requerimientos. El cliente intenta reformular su concepto, algo nebuloso, de la función y comportamiento de los programas en detalles concretos. El desarrollador de software actúa como interrogador, consultor y el que resuelve los problemas.

El análisis y especificación de requerimientos puede parecer una tarea relativamente sencilla, pero las apariencias engañan, puesto que el contenido de comunicación es muy alto, abundan los cambios por mala interpretación o falta de información.

El análisis plantea la asignación de software a nivel de sistema y el diseño de programas, el análisis facilita al ingeniero especificar la función y comportamiento de los programas, indicar la interfaz con otros elementos del sistema y establecer la ligadura de diseño que debe cumplir el programa. El análisis da al diseñador la representación de la información y las funciones que pueden ser traducidas en datos, arquitectura y diseño procedimental. Además que suministra al técnico y al cliente los medios para valorar la calidad de los programas, una vez que se haya construido.

Dentro del análisis encontramos cuatro áreas de interés: 1) reconocimiento del problema, 2) evaluación y síntesis, 3) especificación, 4) revisión.

Inicialmente se estudian la especificación del sistema para comprender la función del producto de software. El objetivo del analista es reconocer los elementos básicos del programa tal como lo percibe el usuario/cliente.

La evolución del problema y la síntesis de la solución es la siguiente área principal de trabajo del análisis. El analista debe evaluar el flujo y la estructura de la información, refinar en detalle todas las funciones del programa, establecer las características de la interface del sistema y descubrir las ligaduras del diseño.

Cada una de las tareas sirve para descubrir el problema de forma que pueda sintetizarse un enfoque o solución global. Una vez que se ha identificado los problemas, el analista determina que información va a ser producida por el sistema y que datos se le suministrarán al sistema.

Hasta la evaluación de los problemas actuales y de la información deseada (entrada y salida), el analista comienza por sintetizar una o más soluciones.

Las tareas asociadas con el análisis y especificación existen para dar una representación del programa que puede ser revisada y aprobada por el cliente.

Una vez que se hayan descrito las funciones básicas, comportamiento, interface e información, se especifican los criterios de validación para demostrar una comprensión de una correcta implementación de los programas. Estos criterios sirven como base para hacer la prueba durante el desarrollo de los programas.

Los documentos de análisis sirven como base para una revisión conducida por el cliente y el técnico. La revisión casi siempre produce modificaciones en la función comportamiento, representación de la información, ligaduras o criterios de validación. Además se realizará una nueva apreciación del plan del proyecto del software para determinar si las primeras estimaciones siguen siendo validas después del conocimiento adicional obtenido durante el análisis.

Para el modelo propuesto de desarrollo en este trabajo el análisis está integrado por los siguientes procesos:

1.- Definición de requerimientos, donde se dará la especificación de los requisitos generales de software.

2.- Análisis estructurado, lo cual da como resultado los diagramas de flujo de datos del sistema y el diccionario de requisitos.

3.- Modelización de datos, cuyo producto son los diagramas entidad-relación del sistema.

En el presente capítulo veremos los fundamentos teóricos que conforman el análisis del trabajo en cuestión.

3.1 Objetivo del Sistema

Como se mencionó en el capítulo anterior, debido a que se ha incrementado en los últimos meses el volumen de las ventas de fianzas, el control de la información se vuelve también más difícil y complicado por lo que es

necesario crear un sistema que controle y automatice la información para brindar un mejor servicio al cliente, que sea rápido, eficiente y por otro lado que la empresa además de tomar y llevar el control de la información también genere reportes de las operaciones más importantes que implican las fianzas, así como reportes de producción que indiquen cuales fueron los productos más vendidos por mes.

Resumiendo el sistema deberá ayudar a la empresa a controlar su información generada, a dar un mejor servicio a sus clientes y por último generar información que sea de gran utilidad para los directivos y personal de la empresa.

3.2 Definición de Requerimientos

La definición de requerimientos es el estudio del problema o sistema que se va tratar para encontrar una solución y/o la forma de mejorar los aspectos donde se está fallando. Ahora entrando en detalle un requerimiento es una característica que debe ser incluida en el sistema y puede ser en una forma de captar o procesar datos, producir información, controlar alguna actividad o dar apoyo para la toma de decisiones, por lo que la definición de requerimientos es la recopilación de datos en relación al sistema que se va a desarrollar para encontrar cuales son los requerimientos.

Dado que la investigación es el primer paso en la definición de requerimientos. Se utilizan una gran variedad de métodos para la recopilación de datos sobre la situación existente: observación, cuestionarios, entrevistas, etc. En el capítulo anterior dimos una visión general de la empresa y empezamos a introducirnos un poco en su funcionamiento, pero para realizar la definición de requerimientos nos tenemos que adentrar en los detalles que lleva consigo GOSAC.

3.2.1 Descripción de las operaciones

Dividimos en módulos las principales operaciones que GOSAC lleva a cabo y además está nos sirve para tener un mejor entendimiento de los requerimientos.

Los módulos siguientes describen las actividades principales que integran las operaciones de la empresa GOSAC.

1.- Mantenimiento de Catálogos

La empresa GOSAC tiene registrados en catálogos mucha de la información que requiere una fianza, es decir, que para realizar alguna operación sobre una fianza debe tener información vigente de sus compañías que maneja, de los agentes que integran su compañía, de los usuarios que operarán el

sistema, del tipo de operaciones validas que se pueden realizar en una fianza, como la fianza es un documento contable, también se debe tener información del tipo de moneda que se maneja en la fianza, se requiere además la información de los beneficiarios de las fianzas, de las gerencias o sucursales de las compañías afianzadoras, de los números (clave de identificación) que tienen asignados los agentes en las compañías correspondientes, de los tipos de fianzas que maneja cada compañía afianzadora, y de los grupos o consorcios a los que pertenece cada cliente.

La información de los catálogos está siempre cambiando, es decir, que los catálogos se deben de actualizar conforme se vaya necesitando, por ejemplo se tiene que dar de alta a un nuevo agente, o en el caso contrario, un agente que ya no preste sus servicios a la empresa se le da de baja. Por otra parte en el caso de las monedas se necesita tener la paridad de la moneda extranjera con la moneda nacional y si citamos el caso del dólar americano, la moneda cambia a diario su paridad, entonces es necesario modificar constantemente el valor de su paridad.

Como se puede apreciar es necesario dar mantenimiento a los catálogos (altas, cambios, bajas) de una manera fácil y sencilla para la debida utilidad y ayuda de la empresa.

2.- Solvencias

Las solvencias se manejan como un tipo especial de catálogos, ésto es debido a que las solvencias son un atributo, si lo podemos llamar así, del cliente.

Para poder tener acceso a la información de una solvencia es necesario primero conocer el cliente, una vez seleccionado el cliente entonces podemos ver la solvencia que tiene y además como se mencionó anteriormente, que es un catálogo entonces tiene altas, bajas, cambios y modificaciones, pero con la condición de que si un cliente no existe por ende no puede existir solvencia, además de que ésta nunca puede ser una cantidad negativa.

Como se explicó anteriormente la solvencia depende del cliente, entonces también es necesario tener un catálogo de clientes que nos permita dar de alta un nuevo cliente que requiere de los servicios de la empresa, dar de baja un cliente que ya no necesite de nuestros servicios o modificar los datos de un cliente. El catálogo de clientes tendrá entonces una opción especial que se llamará solvencias y que nos permitirá accesar a la información deseada del cliente que se especifique.

3.- Registro de Fianzas

El registro de fianzas es el control de la vida de una fianza, desde que nace con el movimiento de expedición se anotan y se registran los datos con los que

fue expedida la fianza, el cliente, el beneficiario, el agente que vende la fianza, el tipo de fianza, el tipo de movimiento de la fianza, el monto por la cual se expide la fianza, la fecha de inicio y fin de vigencia que por lo general son de un año, la prima de la fianza y la comisión que ésta representa para el agente que la vende, y demás datos que la integran.

Posteriormente también se registran los movimientos que pueden surgir en el transcurso de la vida de la fianza. Debido a que puede existir un movimiento de aumento del monto, una disminución, una renovación en el caso que se extienda el periodo de vigencia, una anulación de movimiento, y termina con la cancelación de la fianza.

Todo lo anterior es la parte que se denominará captura de producción y que estará contenida en el módulo de registro de fianzas, junto con ésta se encuentra el cambio de llave de una fianza, la consulta de fianzas y los reportes de producción.

4.- Folios

Dado que la fianza es un documento expedido por la afianzadora con un formato establecido legalmente; el documento es muy importante tanto para la compañía afianzadora que lo otorga como para GOSAC ya que se le puede dar uso fraudulento y la compañía tendría que responder por él. Es por ello que los folios o formas de operación se encuentran numerados (o foliados) con un número y una serie que los identifica totalmente.

GOSAC necesita controlar los folios desde el momento en que se reciben por la compañía afianzadora hasta que son usados por una fianza, así que GOSAC necesita conocer el estado en que se encuentra un folio (usado, inutilizado, listo para ser usado).

5.- Primas por Cobrar

Después de vender una fianza la empresa GOSAC debe asegurarse que las primas cobradas de las fianzas sean liquidadas por los fiados, por lo que se requiere controlar las primas por cobrar y su respectivo cobro además de que se debe dar la información correspondiente a los agentes que tienen que cobrarle a sus respectivos clientes.

6.- Primas por Pagar

Después de cobrar la prima respectiva al cliente, las primas tienen que ser pagadas por GOSAC a la compañía afianzadora correspondiente, por lo que se requiere controlar la primas por pagar y generar la información de las primas que se van a pagar para cada compañía.

7.- Comisiones

GOSAC se hace acreedor a una comisión por cada fianza vendida (fianza colocada) de una compañía afianzadora, pero para poder cobrar la prima, el monto de la prima ya debió de haberse pagado a la compañía respectiva, entonces se necesita tener y llevar un control de las comisiones que se generan y a que agente le corresponde dicha comisión, por lo que también se necesita generar información de las comisiones cobradas por agente-compañía.

8.- Proceso de Acumulación de Producción

Al fin de mes se necesita generar información del número y tipo de productos vendidos en dicho mes, así como el total de prima neta que se generó del producto, el total del monto del producto que se vendió y la comisión generada por las ventas del producto (expediciones, aumentos, cancelaciones, renovaciones, etc.) en cuestión.

En la siguiente figura 3.1 podemos observar el ciclo operativo de la expedición y de la vida de una fianza, y además se puede apreciar la gran relación que existe con los módulos citados anteriormente.

El impresor le entrega el folio a la afianzadora (1), el folio se distribuye a una sucursal (2), la sucursal a su vez lo distribuye a GOSAC (3), un agente de GOSAC coloca la fianza utilizando el folio (4), con la emisión de la fianza se crea una fianza en vigor, una venta y una prima por cobrar (5), al cobrarse la prima al fiado (6) (se crea un ingreso para GOSAC y la afianzadora correspondiente) se salda la prima por cobrar generando una comisión por pagar al agente (GOSAC), la que poco tiempo después es cobrada a la afianzadora (7). Si durante la vigencia de la fianza se realizan otros movimientos considerados como ventas (renovación, aumento, etc.) el ciclo se repite. Cuando la fianza se cancela (8) se actualiza el inventario de fianzas.

El movimiento de fianza llamado anulación es un caso particular y se presenta cuando el fiado avisa que no va a utilizar la fianza o el movimiento porque ya no se requiere. En este caso se procede a la anulación del movimiento, regresando la afianzadora y GOSAC la cantidad cobrada al fiado, si ya había sido liquidada ésta.

3.2.2 Requerimientos Generales

Como ya vimos en la sección anterior y en el capítulo 2, la descripción de las operaciones, nos dan como resultado los siguientes requerimientos generales que se deben cubrir.

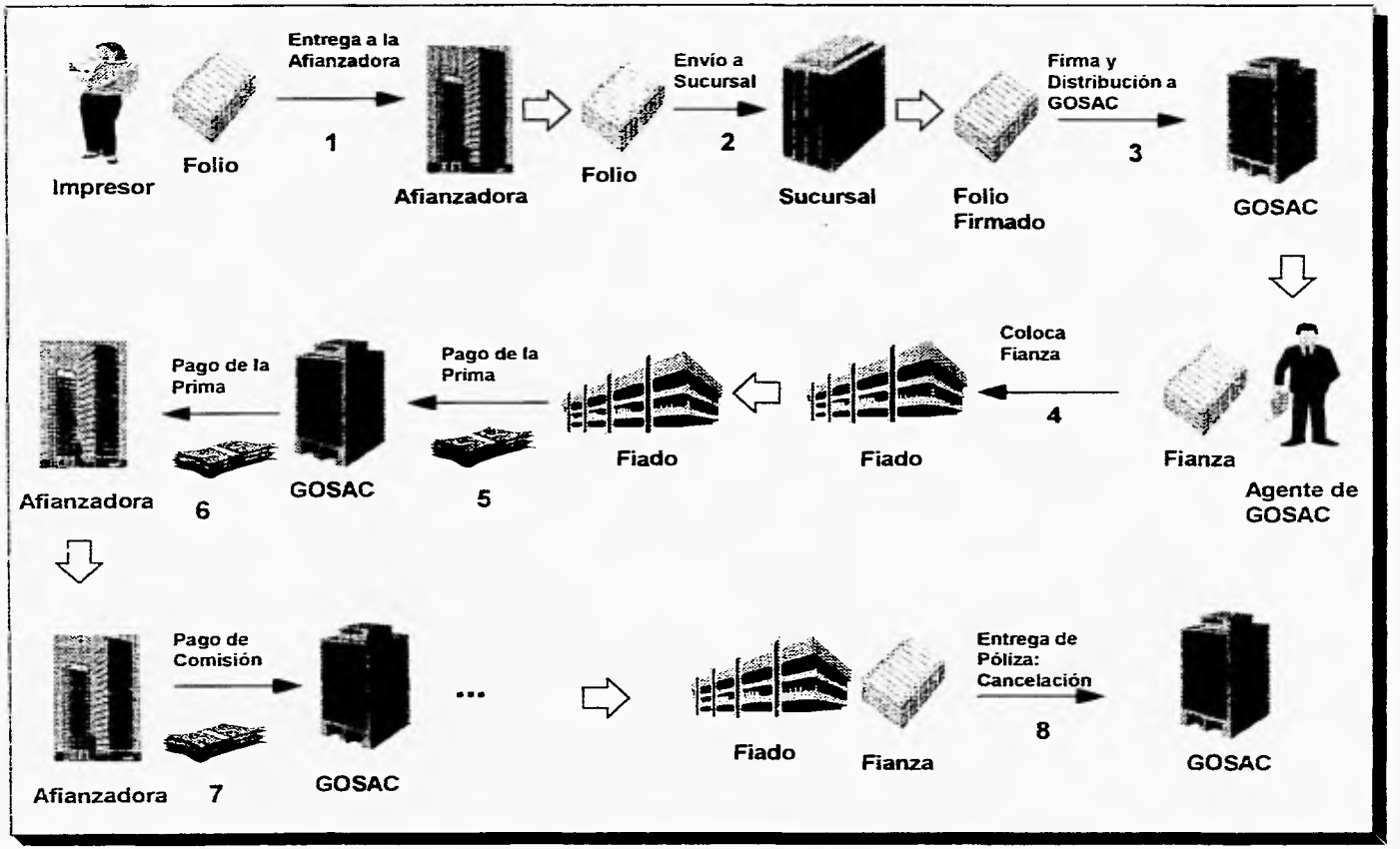


Figura 3.1 Ciclo operativo de la emisión de una fianza

1.- Mantenimiento de Catálogos

- Registro y control del catálogo de compañías afianzadoras
- Registro y control del catálogo de agentes de GOSAC y su número correspondiente en cada compañía afianzadora en la que esté registrado
- Registro y control de usuarios del sistema
- Registro y control de los tipos de fianzas que manejan las compañías afianzadoras
- Registro y control de los tipos de monedas que se manejan
- Registro y control de los beneficiarios de las fianzas
- Registro y control de las gerencias o sucursales de cada compañía afianzadora registrada en el sistema
- Registro y control de los grupos a los que pertenecen los clientes
- Registro y control del tipo de movimientos permitidos de una fianza
- Registro y control del tipo de folio existente por cada compañía

2.- Solvencias

- Registro y control de clientes
- Registro y control de la solvencia del cliente

3.- Registro de fianzas

- Registro y control de fianzas expedidas
- Registro y control de las renovaciones de las fianzas
- Registro y control de las disminuciones de las fianzas
- Registro y control del aumento en las fianzas
- Registro y control de la anulación de movimientos en las fianzas

- Registro y control de las cancelaciones de las fianzas
- Control de los cambios y modificaciones de una fianza
- Control de las fianzas en vigor (que estén vigentes) para consultas
- Generación de las comisiones por la venta de productos
- Control de las tarifas de prima a partir del tipo de la fianza
- Cálculo de la comisión, de los futuros, I.V.A., e impuestos de la fianza automáticamente
- Reportes de producción de GOSAC
- Reportes de producción por compañía afianzadora
- Reportes de producción por agente
- Inventario de fianzas en vigor de GOSAC
- Inventario de fianzas en vigor por compañía
- Inventario de fianzas en vigor por agente
- Reportes de vencimientos de GOSAC
- Reportes de vencimientos por compañía afianzadora
- Reportes de vencimientos por agente
- Reportes de responsabilidades por cliente

4..- Registro y control de inventarios de folios

- Registro y control de las series de folios existentes
- Control de los folios inutilizados
- Presentación de la información de folios por compañía y series de folio
- Generación de reportes del estado de los folios

5.- Primas por cobrar

- Registro de primas por cobrar en relación directa a las ventas
- Registro y manejo de pagos parciales
- Control de pagos a primas en moneda extranjera
- Consulta de información de primas por cobrar

6.- Primas por pagar

- Registro de primas por pagar en relación directa a las ventas
- Consulta de información de primas por pagar

7.- Comisiones

- Registro de comisiones en función de la prima, el agente y tipo de fianza
- Control de las comisiones por pagar (cobrar) en relación directa al pago de las primas
- Registro y control de los tipos de comisión que las compañías pagan a los agentes
- Generación de reportes de comisiones por cobrar por compañía
- Generación de reportes de comisiones cobradas por agente-compañía

8.- Proceso de Acumulación de Producción

Generación de un proceso que extraiga información de los movimientos que se realizaron por mes, de las ventas, de las primas, del monto total, etc.

- Generación de reportes gráficos de las ventas mensuales de productos, de la prima neta, del monto, etc.

El sistema deberá apoyar estas 8 áreas básicas de la empresa para que una vez soportada la operación, incursionar en modelos de apoyo general y directivo para la toma de decisiones y aumentar a su vez el alcance del sistema.

3.3 Análisis Estructurado

.Como todos los demás métodos existentes de análisis de requisitos, el análisis estructurado es una actividad de construcción de modelos. Los modelos creados reflejan el flujo y el contenido de la información; se parte el sistema funcionalmente y según los distintos comportamientos, se establece la esencia de lo que se debe construir. Todo lo anterior es creado mediante una notación que es única del método de análisis estructurado.

3.3.1 Diagramas de flujo de datos (D.F.D.)

La información se transforma a medida que fluye por un sistema basado en computadora. El análisis estructurado es una técnica de modelización del flujo y el contenido de información del sistema en cuestión. El diagrama de flujo de datos (D.F.D.) es una técnica gráfica que representa el flujo de datos de la información y las transformaciones que se aplican a los datos al moverse desde la entrada hasta la salida. Además el D.F.D. es una herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por "conductos" y "tanques de almacenamiento" de datos.

El diagrama de flujo de datos es una de las herramientas muy comúnmente usadas, sobre todo por sistemas operacionales en los cuales las funciones del sistema son de gran importancia y son más complejas que los datos que éste maneja. A su vez la notación aunque ha ido variando se tomo prestada de la teoría de gráficas, y continúa siendo utilizada por los ingenieros de software que trabajan en la implantación directa de los modelos de los requerimientos del usuario.

Hay que tener en mente que un D.F.D. es tan solo una herramienta de modelado disponible y que únicamente proporciona un punto de vista de un sistema, el orientado a las funciones.

Algunas de las ventajas que nos pueden ofrecer los D.F.D's son las siguientes:

- Prácticamente no requiere explicación, se puede en algunas ocasiones simplemente mirar el diagrama y entenderlo. La notación es sencilla, clara y en cierto sentido intuitivamente obvia.

- El diagrama cabe fácilmente en una página y esto significa dos cosas: 1) alguien puede mirarlo sin ofuscarse y 2) el sistema que se está modelando no es muy complejo.

Se puede usar el diagrama de flujo de datos para representar un sistema a cualquier nivel de abstracción. De hecho, los D.F.D's pueden ser refinados en niveles que representan un mayor flujo de información y un mayor detalle funcional. Un D.F.D. de nivel 0 también es denominado modelo fundamental del sistema o modelo de contexto, y representa un elemento de software completo como un solo bloque de datos de entrada y salida representadas por flechas de entrada y salida respectivamente. A partir del D.F.D. de nivel 0 para mostrar más detalles aparecen representados procesos (bloques) y caminos de flujo de información adicionales.

Para explicar la notación básica que se utiliza en la elaboración de un D.F.D. nos auxiliaremos del siguiente diagrama. La elipse se usa para representar una entidad externa, es decir, un elemento del sistema, u otro sistema que produce información a ser transformada por el sistema o que recibe información del software. Un bloque representa un proceso de transformación que se aplica a los datos y a los cambios de alguna forma. Las flechas de un diagrama de flujo de datos muestra precisamente el flujo de datos, desde su origen y hasta su destino. el rectángulo parcial representa un almacén de información, la cual es utilizada por el software. La sencillez de la notación del D.F.D. es una de las razones por las que las técnicas del análisis estructurado son ampliamente utilizadas.

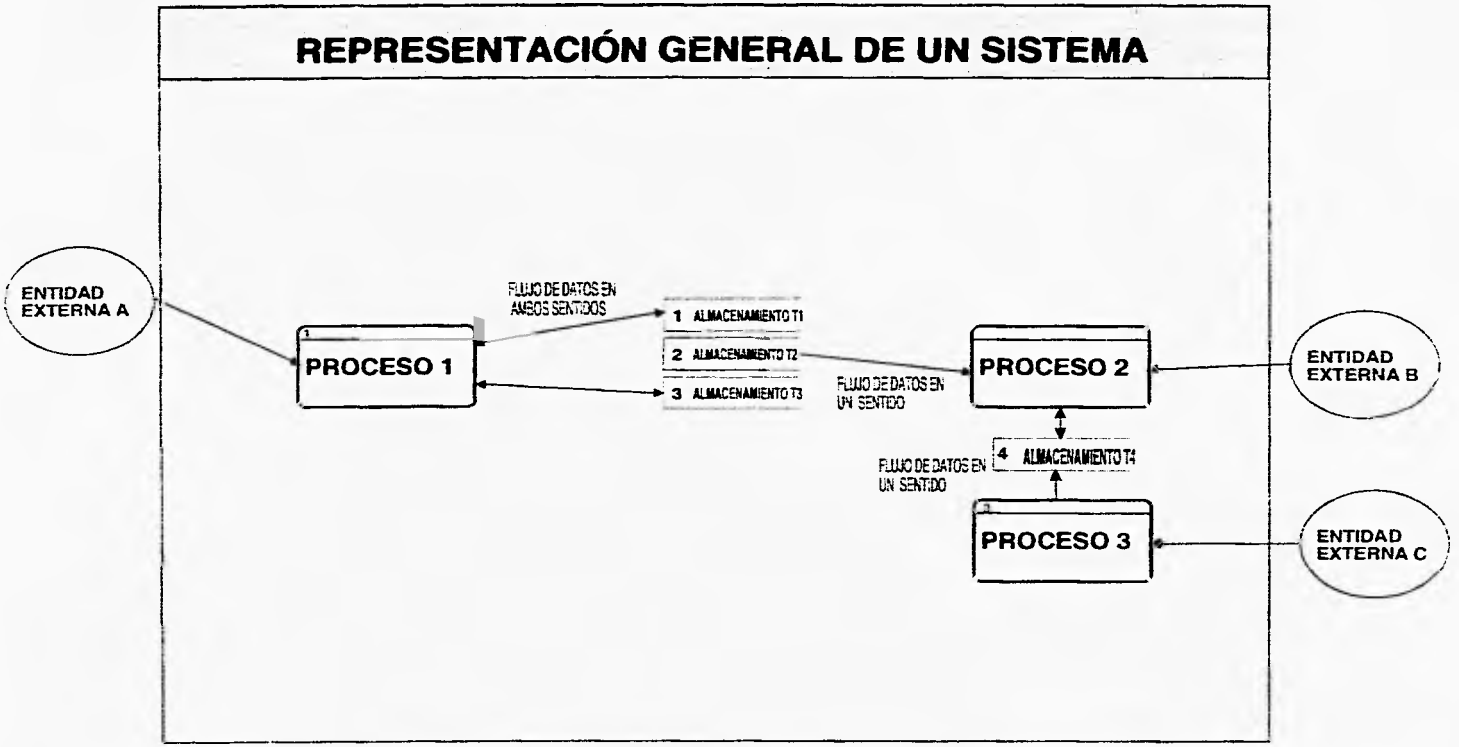
Es importante señalar que los D.F.D's no proporcionan ninguna indicación explícita de la secuencia del procesamiento. El procedimiento o la secuencia puede estar implícitamente en el diagrama, pero la representación procedimental explícita generalmente queda pospuesta hasta el diseño del software.

El diagrama de flujo de datos permite desarrollar los modelos del ámbito de información y del ámbito funcional al mismo tiempo. A medida que se refina el D.F.D. en mayores niveles de detalles, se puede llevar a cabo implícitamente una descomposición funcional del sistema.

3.3.2 El Diccionario de Requisitos

El flujo de la información no basta para tener un análisis completo del sistema. Cada elemento del diagrama de flujo de datos debe quedar totalmente definido, así como la relaciones que existan entre ellos.

El diccionario de requisitos es utilizado como gramática casi formal para describir el contenido de los objetos definidos durante el análisis estructurado.


REPRESENTACIÓN GENERAL DE UN SISTEMA
ENTIDAD EXTERNA A
PROCESO 1
FLUJO DE DATOS EN AMBOS SENTIDOS
1 ALMACENAMIENTO T1
2 ALMACENAMIENTO T2
3 ALMACENAMIENTO T3
FLUJO DE DATOS EN UN SENTIDO
PROCESO 2
ENTIDAD EXTERNA B
FLUJO DE DATOS EN UN SENTIDO
4 ALMACENAMIENTO T4
PROCESO 3
ENTIDAD EXTERNA C

El diccionario de requisitos tiene como objetivo establecer una relación detallada de todos los elementos que conforman el diagrama de flujo de datos: la definición precisa de los almacenamientos de información o estructuras de datos, la descripción detallada de los procesos y la explicación de los elementos externos integran el diccionario de requisitos.

3.3.3 Diagramas de flujos de datos del software para GOSAC y su diccionario de requisitos

Analizando tanto las entradas como, las salidas que existen en el sistema dio como resultado las entidades externas que están relacionadas con él.

La siguiente figura que es el D.F.D. inicial muestra precisamente al sistema como un solo proceso, con las entidades externas que se relacionan.

Las entidades externas son las siguientes:

- **Compañías:** La entidad de compañías representa a las compañías afianzadoras a las que se le colocan fianzas.

- **Agentes:** Esta entidad representa a los agentes de GOSAC que requieren información constante de vencimientos, clientes, cancelaciones, y primas por cobrar para facilitar su labor de venta.

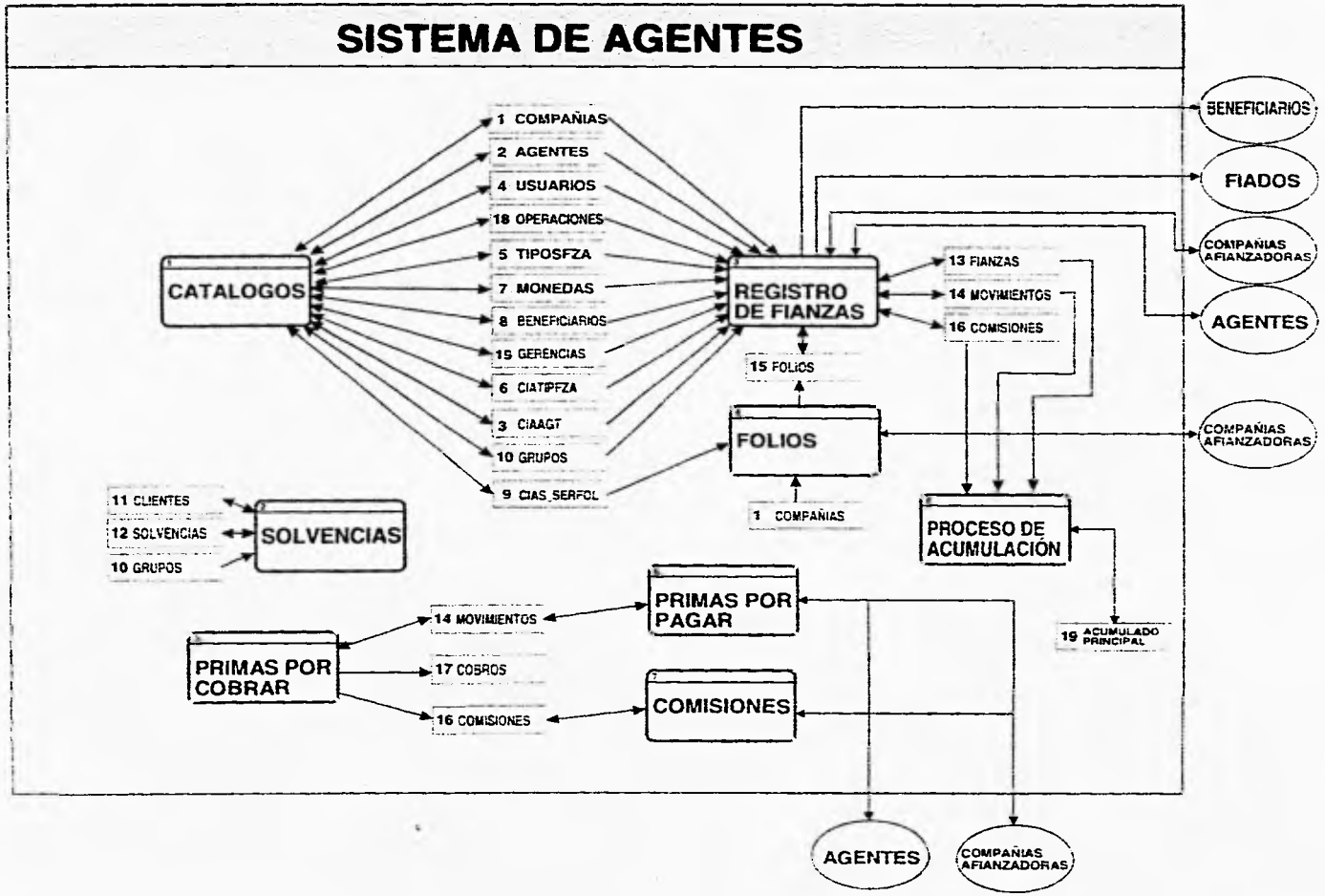
- **Clientes o Fiados:** representa a los clientes (fiados) de las fianzas otorgadas y vendidas por GOSAC a los cuales se les emiten sus estados de cuenta, sus vencimientos y otros informes de sus relaciones con la compañía.

- **Beneficiarios:** Está entidad representa a los beneficiarios de las fianzas otorgadas por la afianzadora que requieren información sobre el estado de lo que la compañía les está garantizando y le proporcionan información a la empresa sobre la procedencia de cancelaciones.

Un análisis más detallado de la operación básica de GOSAC revelo 8 procesos principales: (1) Mantenimiento de catálogos, (2) Solvencias de clientes, (3) Registro y control de fianzas y movimientos, (4) Registro y control de formas de operación o folios, (5) Registro y control de primas por cobrar, (6) Registro y control de primas por pagar, (7) Registro y control de comisiones, (8) Proceso de acumulación de producción.

Conforme uno va avanzando en el proceso de modelado por D.F.D's debe de ir progresando de lo general a lo particular, detallando separadamente cada proceso hasta el nivel necesario que muestre los alcances y limitaciones del sistema. Si el D.F.D. explota hasta el último detalle de cada proceso,

prácticamente se estará creando la estructura funcional detallada del sistema. Cuando el sistema que se está modelando es complejo por la gran cantidad de procesos, flujos de datos y almacenamientos que lo integran, el alcance del D.F.D. debe de ser de primer o segundo nivel, ya que de lo contrario la complejidad de los diagramas particulares les restará utilidad al modelado por D.F.D. del sistema.


SISTEMA DE AGENTES
CATALOGOS
1 COMPAÑIAS
2 AGENTES
4 USUARIOS
18 OPERACIONES
5 TIPOSFZA
7 MONEDAS
8 BENEFICIARIOS
15 GERENCIAS
6 CIATIPFZA
3 CIAAGT
10 GRUPOS
9 CIAS_SERFOL
REGISTRO DE FIANZAS
13 FIANZAS
14 MOVIMIENTOS
16 COMISIONES
15 FOLIOS
FOLIOS
1 COMPAÑIAS
BENEFICIARIOS
FIADOS
COMPAÑIAS AFIANZADORAS
AGENTES
COMPAÑIAS AFIANZADORAS
11 CLIENTES
12 SOLVENCIAS
10 GRUPOS
SOLVENCIAS
PROCESO DE ACUMULACION
19 ACUMULADO PRINCIPAL
PRIMAS POR PAGAR
14 MOVIMIENTOS
PRIMAS POR COBRAR
17 COBROS
16 COMISIONES
COMISIONES
AGENTES
COMPAÑIAS AFIANZADORAS

Proceso principal: 1.- Mantenimiento de catálogos

Descripción: Proceso principal de registro y control de los catálogos que utiliza la empresa GOSAC para su servicio de venta de fianzas.

Subprocesos que integran el proceso principal 1:

Proceso 1.1.- Compañías: Proceso de mantenimiento al catálogo de compañías, que se encarga de dar altas, cambios y bajas de las compañías afianzadoras con las que trabaja GOSAC.

Proceso 1.2.- Agentes: Proceso de mantenimiento al catálogo de agentes, que se encarga de dar de altas, cambios, y bajas de los agentes que trabajan para GOSAC.

Proceso 1.3.- Usuarios: Proceso de mantenimiento al catálogo de usuarios que se encarga de dar altas, cambios, y bajas de los usuarios que operan al sistema.

Proceso 1.4.- Operaciones: Proceso de mantenimiento al catálogo de operaciones que se encarga de dar altas, cambios y bajas de las operaciones validas que tienen las fianzas.

Proceso 1.5.- Tipos de Fianzas: Proceso de mantenimiento al catálogo de tipos de fianza que se encarga de dar altas, cambios y bajas de los tipos de fianza que manejan las diferentes compañías afianzadoras que tiene registrado GOSAC.

Proceso 1.6.- Monedas: Proceso de mantenimiento al catálogo de monedas, que se encarga de dar de altas, cambios y bajas de los tipos de monedas que utilizan las fianzas.

Proceso 1.7.- Beneficiarios: Proceso de mantenimiento al catálogo de beneficiarios, que se encarga de dar altas, cambios y bajas a los beneficiarios de las fianzas que maneja GOSAC.

Proceso 1.8.- Gerencias: Proceso de mantenimiento al catálogo de las gerencias o sucursales que tiene las diversas compañías afianzadoras que maneja GOSAC y se encarga de dar altas, cambios y bajas.

Proceso 1.9.- Grupos: Proceso de mantenimiento al catálogo de grupos a los que pertenece un cliente que se encarga de dar de altas, cambios y bajas de los grupos.

Proceso 1.10.- Compañías Series de Folio.- Proceso de mantenimiento al catálogo de series de folios por compañía que se encarga de dar altas, cambios y bajas de los folios asignados por una compañía.

Estructura de datos asociadas al proceso principal 1:

T1.- Compañías: Catálogo de compañías afianzadoras

T2.- Agentes: Catálogo de los agentes que trabajan para GOSAC

T3.- Compañías-Agentes: Catálogo de los agentes de GOSAC que existen en los diversas compañías afianzadoras que maneja GOSAC.

T4.- Usuarios: Catálogo de los usuarios que tienen acceso al sistema.

T5.- Tipos de Fianza: Catálogo de los tipos de fianza que maneja GOSAC.

T6.- Compañías-Tipos de Fianza: Catálogo de los tipos de fianza que maneja las compañías afianzadoras para las que vende fianzas GOSAC:

T7.- Monedas: Catálogo de las monedas que maneja GOSAC para hacer movimientos de fianzas.

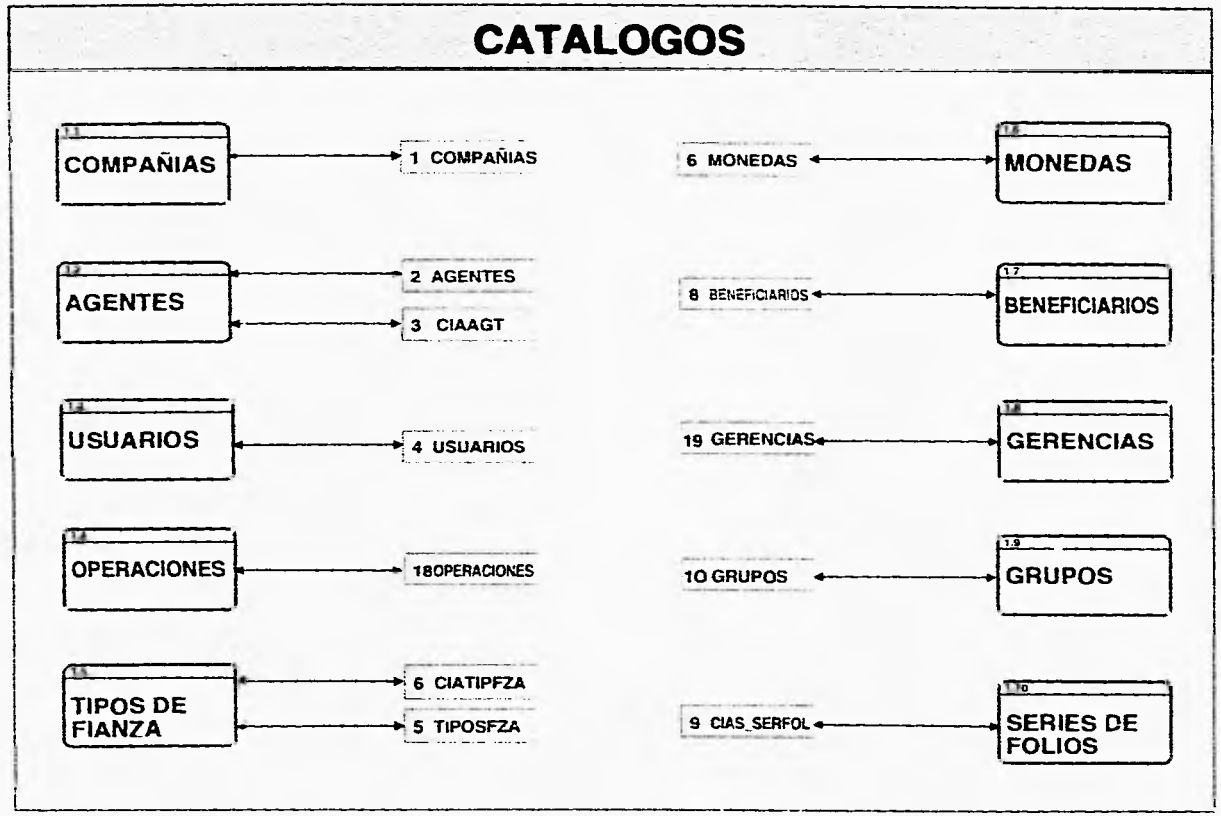
T8.- Beneficiarios: Catálogo de los beneficiarios existentes de las fianzas que vende GOSAC.

T9.- Series de Folios: Catálogo de las series de folio por compañía existentes para GOSAC.

T10.- Grupos: Catálogo de los grupos posibles a los que puede pertenecer un cliente de GOSAC.

T18.- Operaciones: Catálogo de las operaciones validas que se utilizan para las fianzas de GOSAC.

El Diagrama de Flujo de Datos de Mantenimiento de Catálogos es el siguiente:


CATALOGOS
1.1
COMPAÑIAS
1 COMPAÑIAS
1.2
AGENTES
2 AGENTES
3 CIAAGT
1.4
USUARIOS
4 USUARIOS
1.5
OPERACIONES
18OPERACIONES
1.6
TIPOS DE FIANZA
6 CIATIPFZA
5 TIPOSFZA
6 MONEDAS
1.6
MONEDAS
8 BENEFICIARIOS
1.7
BENEFICIARIOS
19 GERENCIAS
1.8
GERENCIAS
10 GRUPOS
1.9
GRUPOS
9 CIAS_SERFOL
1.10
SERIES DE FOLIOS

Proceso principal 2.- Solvencias

Descripción: Proceso principal de registro y control del catálogo de clientes y sus solvencias de la empresa GOSAC.

Subprocesos que integran el proceso principal 2:

Proceso 2.1.- Clientes: Proceso de mantenimiento al catálogo de clientes, que se encarga de dar altas, cambios y bajas a los clientes de GOSAC.

Proceso 2.2.- Solvencias: Proceso de mantenimiento al catálogo de solvencias del cliente, que se encarga de dar altas, cambios y bajas a la solvencia de un cliente.

Estructuras de datos asociadas al proceso principal 2:

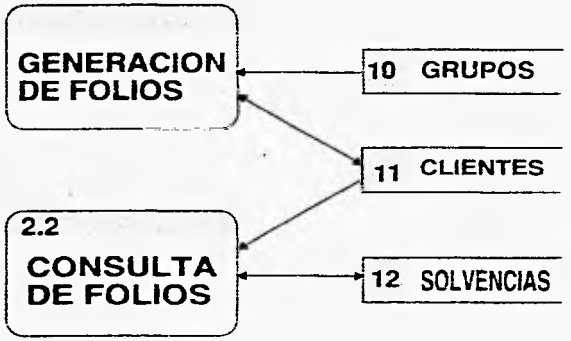
T10.- Grupos: Catálogo de los grupos posibles a los que puede pertenecer un cliente de GOSAC.

T11.- Clientes: Catálogo de los clientes de las fianzas que vende GOSAC.

T12.- Solvencias: Catálogo de las solvencias de los clientes de GOSAC.

El Diagrama de Flujo de Datos de Solvencias es el siguiente:

SOLVENCIAS


GENERACION DE FOLIOS
10 GRUPOS
11 CLIENTES
2.2
CONSULTA DE FOLIOS
12 SOLVENCIAS

Proceso principal 3: Registro de fianzas

Descripción: Proceso en el cual se realizan y se registran todas las operaciones, movimientos, consultas y reportes de las fianzas.

Subprocesos que integran el proceso principal 3:

Proceso 3.1.- Captura de Producción: Proceso de registro de los movimientos de las fianzas; estos pueden ser expedición, renovación, aumento, anulación, cancelación y disminución

Proceso 3.2.- Consulta de fianzas: Proceso de extracción de información de los movimientos de una fianza, así como información general de ésta.

Proceso 3.3.- Cambio de Llave: Proceso de cambios de atributos en el inventario de fianzas, inventario de movimientos e inventario de comisiones de una fianza.

Proceso 3.4.- Reportes de Producción: Proceso de extracción de información del inventario de fianzas y movimientos que se mostrará y se enviará a pantalla y/o impresora.

Estructura de datos asociadas al proceso principal 3:

T1.- Compañías: Catálogo de compañías afianzadoras

T2.- Agentes: Catálogo de los agentes que trabajan para GOSAC

T3.- Compañías-Agentes: Catálogo de los agentes de GOSAC que existen en los diversas compañías afianzadoras que maneja GOSAC.

T5.- Tipos de Fianza: Catálogo de los tipos de fianza que maneja GOSAC.

T6.- Compañías-Tipos de Fianza: Catálogo de los tipos de fianza que maneja las compañías afianzadoras para las que vende fianzas GOSAC:

T7.- Monedas: Catálogo de las monedas que maneja GOSAC para hacer movimientos de fianzas.

T8.- Beneficiarios: Catálogo de los beneficiarios existentes de las fianzas que vende GOSAC.

T10.- Grupos: Catálogo de los grupos posibles a los que puede pertenecer un cliente de GOSAC.

T13.- Fianzas: Inventario de las fianzas otorgadas por GOSAC.

T14.- Movimientos de Fianzas: Inventario de los movimientos realizados a las fianzas desde su expedición hasta su cancelación.

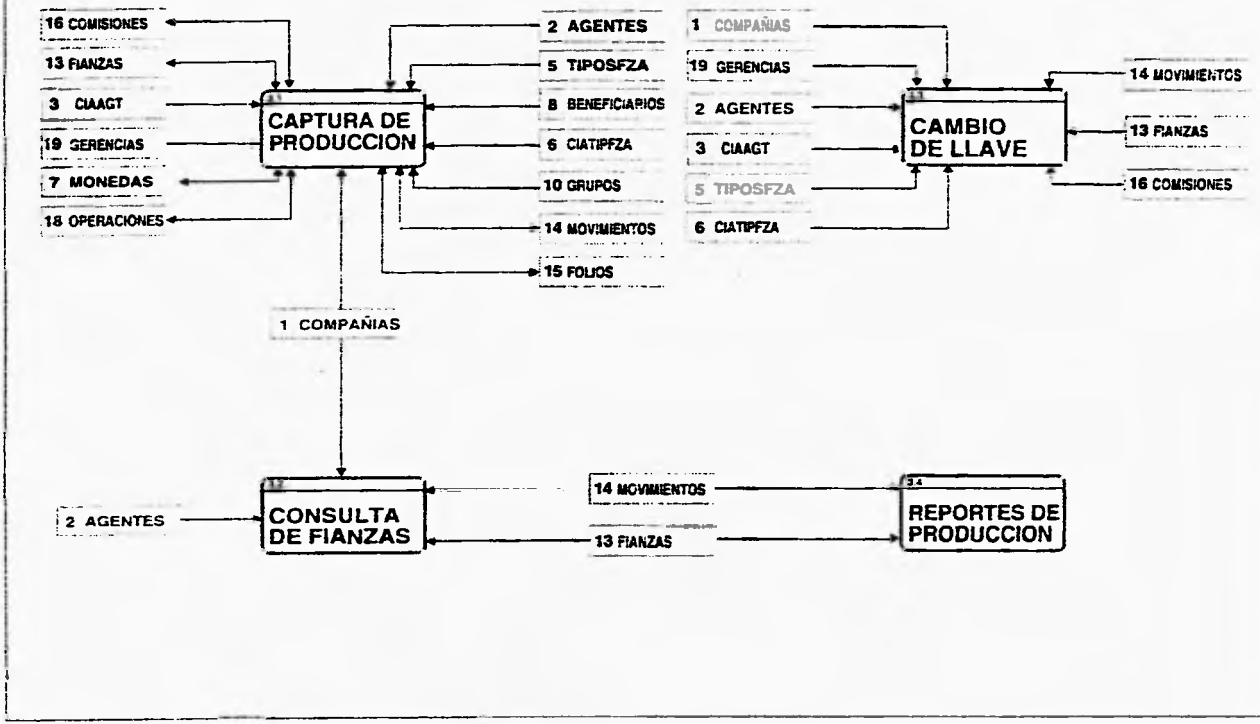
T15.- Folios: Inventario que contiene la información y características de cada forma dada por una compañía afianzadora para uso de GOSAC.

T16.- Comisiones: Inventario que contiene la información de cada comisión que genera una fianzas, así como su estado.

T18.- Operaciones: Catálogo de las operaciones validas que se utilizan para las fianzas de GOSAC.

El Diagrama de Flujo de Datos de Registro de Fianzas es el siguiente:

REGISTRO DE FIANZAS



Proceso principal 4: Folios

Descripción: Proceso principal de registro y control de folios (formas de fianzas) de las diferentes compañías afianzadoras que maneja GOSAC.

Subprocesos que integran al proceso principal 4:

Proceso 4.1.- Generación de Folios: Proceso de creación del inventario de folios o formas en base a los folios entregados por las diversas sucursales de las compañías afianzadoras a GOSAC.

Proceso 4.2.- Inutilización: Proceso de inutilización de un folio de una compañía afianzadora. El folio se marca en el inventario para que no sea usado.

Proceso 4.3.- Consulta de Folios: Proceso de extracción de información contenida en el inventario de folios que se presenta en la pantalla.

Estructura de datos asociadas al proceso principal 4:

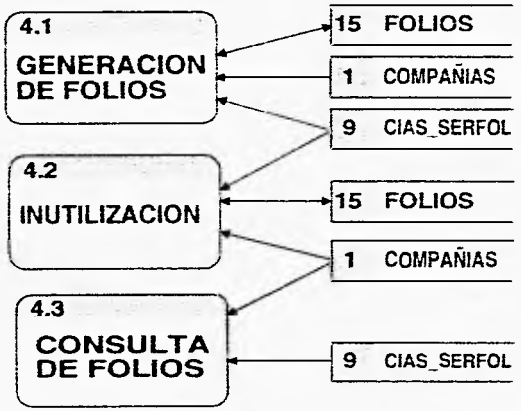
T15.- Folios: Inventario que contiene la información de cada folio o forma de una compañía afianzadora y sus características.

T1.- Compañías: Catálogo de las compañías afianzadoras.

T9.- Compañías Series de Folios: Catálogo de las series de folio por compañía existente en GOSAC.

El Diagrama de Flujo de Datos de Folios es el siguiente:

FOLIOS


4.1
GENERACION DE FOLIOS
4.2
INUTILIZACION
4.3
CONSULTA DE FOLIOS
15 FOLIOS
1 COMPAÑIAS
9 CIAS_SERFOL
15 FOLIOS
1 COMPAÑIAS
9 CIAS_SERFOL

Proceso principal 5.- Primas por cobrar

Descripción: Proceso principal del control de las primas por cobrar de la empresa GOSAC a sus clientes.

Subprocesos que integran al proceso principal 5:

Proceso 5.1.- Registro de Cobros de Primas: Control de las primas por cobrar de los agentes de GOSAC a sus respectivos clientes.

Proceso 5.2.- Reportes de primas por cobrar por agente: Proceso de extracción de información referente a las primas por cobrar.

Estructuras de datos asociadas al proceso principal 5:

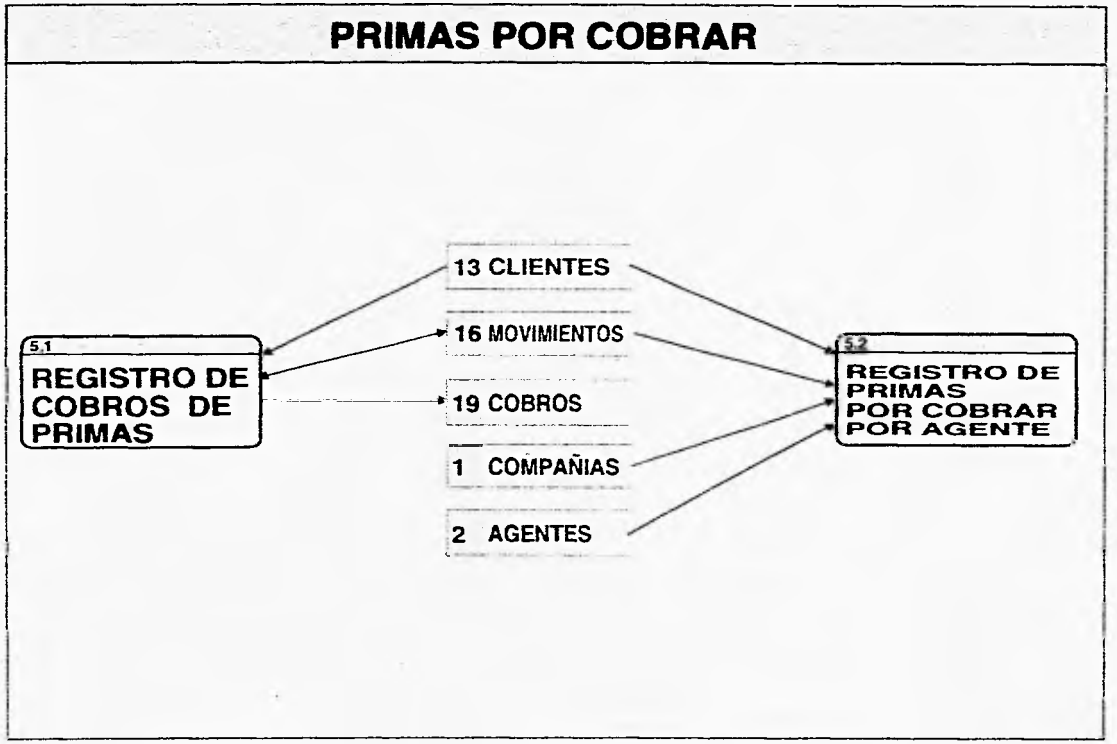
T14.- Movimientos de Fianzas: Inventario de los movimientos realizados a las fianzas desde su expedición hasta su cancelación.

T11.- Clientes: Catálogo de los clientes de las fianzas que vende GOSAC.

T2.- Agentes: Calálogo de los agentes que trabajan para GOSAC

T1.- Compañías: Catálogo de las compañías afianzadoras.

El Diagrama de Flujo de Datos de Registro de Primas por cobrar es el siguiente:


PRIMAS POR COBRAR
13 CLIENTES
16 MOVIMIENTOS
19 COBROS
1 COMPAÑIAS
2 AGENTES
5.1
REGISTRO DE COBROS DE PRIMAS
5.2
REGISTRO DE PRIMAS POR COBRAR POR AGENTE

Proceso principal 6.- Primas por pagar

Descripción: Proceso principal de las primas por pagar de GOSAC a las respectivas compañías afianzadoras.

Subprocesos que integran al proceso principal 6:

Proceso 6.1.- Registro de Pagos de Primas: Control de las primas por pagar de GOSAC a las compañías afianzadoras.

Proceso 6.2.- Reportes de Primas por Pagar: Proceso de extracción de información referente a las primas por pagar.

Estructuras de datos asociadas al proceso principal 6:

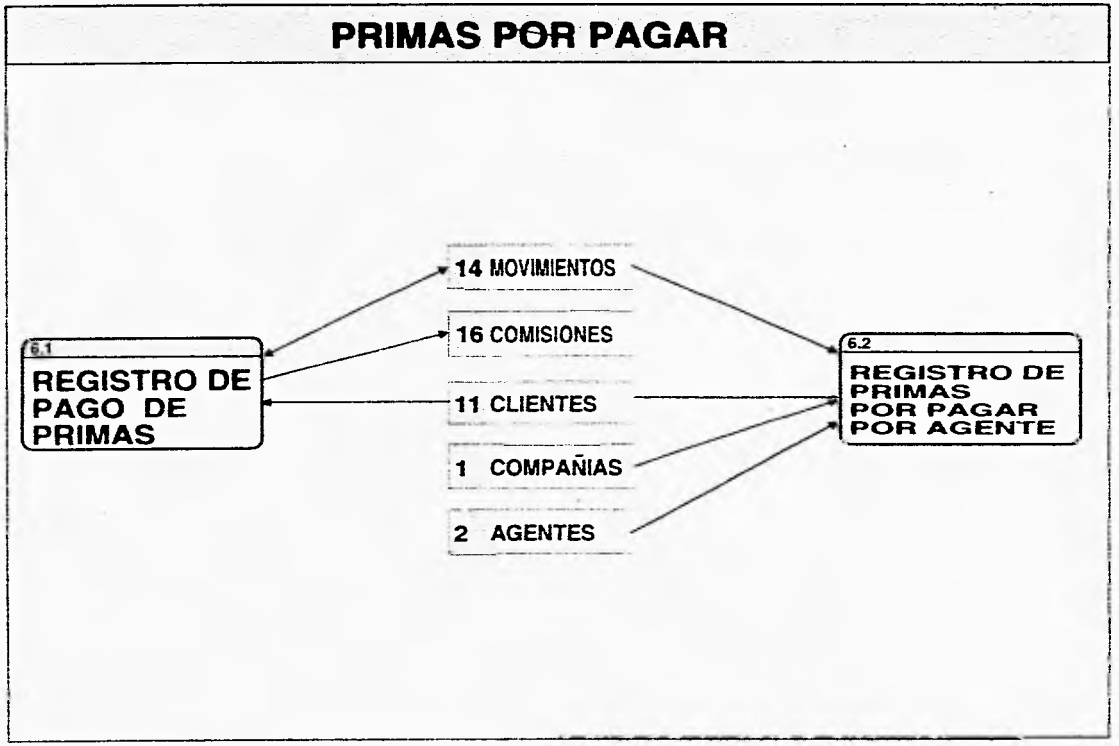
T14.- Movimientos de Fianzas: Inventario de los movimientos realizados a las fianzas desde su expedición hasta su cancelación.

T11.- Clientes: Catálogo de los clientes de las fianzas que vende GOSAC.

T2.- Agentes: Catálogo de los agentes que trabajan para GOSAC.

T1.- Compañías: Catálogo de las compañías afianzadoras.

El Diagrama de Flujo de Datos de Primas por pagar es el siguiente:


PRIMAS POR PAGAR
6.1
REGISTRO DE PAGO DE PRIMAS
14 MOVIMIENTOS
16 COMISIONES
11 CLIENTES
1 COMPAÑIAS
2 AGENTES
6.2
REGISTRO DE PRIMAS POR PAGAR POR AGENTE

Proceso principal: 7.- Comisiones

Descripción: Proceso principal de control de las comisiones a los agentes y a GOSAC.

Subprocesos que integran el proceso principal 7:

Proceso 7.1.- Registro de Cobro de Comisiones: Registro de Cobro de comisiones a las compañías afianzadoras.

Proceso 7.2.- Reportes de Comisiones por compañía: Extracción de información referente a las comisiones por pagar y pagadas de una compañía afianzadora.

Proceso 7.3.- Reporte de Comisiones por agente: Extracción de información referente a las comisiones por pagar y pagadas por agente de GOSAC.

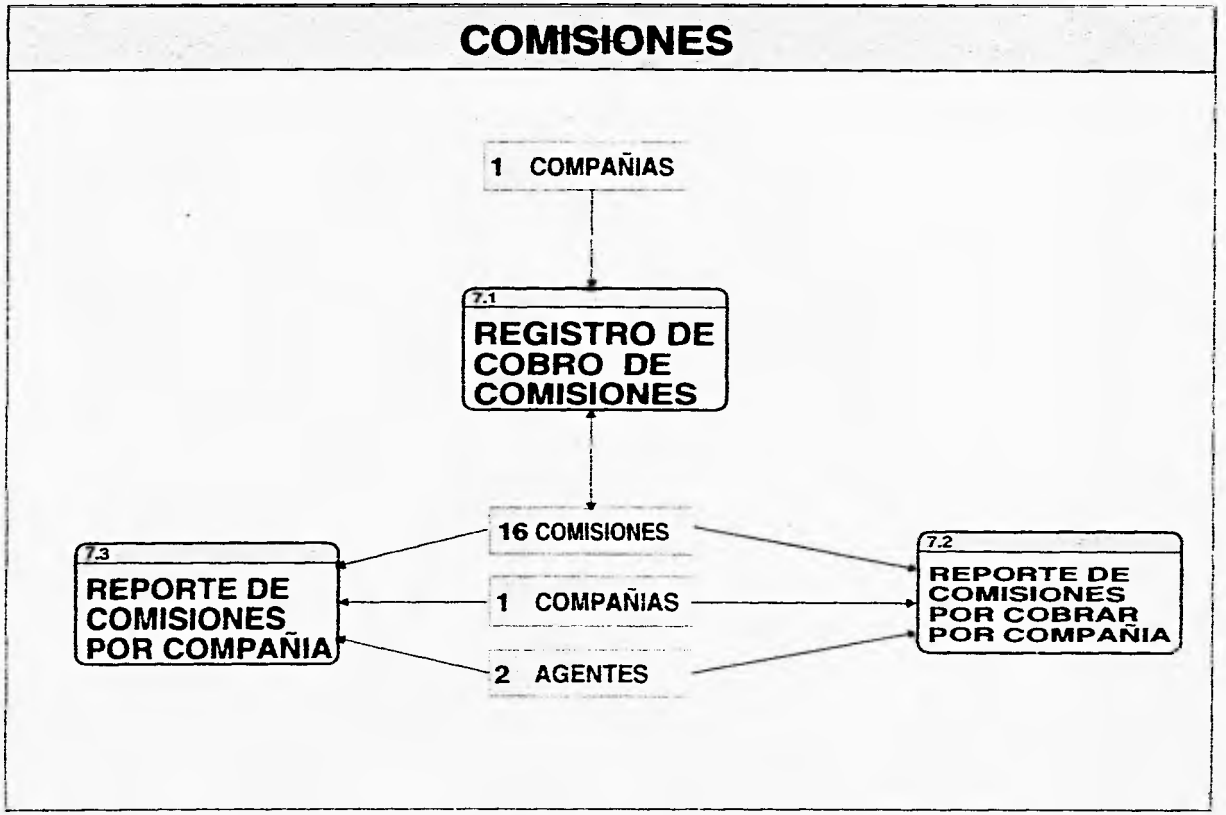
Estructura de Datos asociadas al proceso principal 7:

T1.- Compañías: Catálogo de las compañías afianzadoras.

T2.- Agentes: Catálogo de los agentes que trabajan para GOSAC.

T16.- Comisiones: Inventario que contiene la información de cada Comisión por fianza y su estado.

El Diagrama de Flujo de Datos de Comisiones es el siguiente:


COMISIONES
1 COMPAÑIAS
7.1
REGISTRO DE COBRO DE COMISIONES
16 COMISIONES
1 COMPAÑIAS
2 AGENTES
7.3
REPORTE DE COMISIONES POR COMPAÑIA
7.2
REPORTE DE COMISIONES POR COBRAR POR COMPAÑIA

Proceso principal: 8.- Proceso de acumulación de producción

Descripción: Proceso principal que extrae información de diferentes inventarios y catálogos para generar y almacenar la información en un inventario de producción mensual.

Subprocesos que integran al proceso principal 8:

Proceso 8.1.- Acumulación Mensual: Proceso que extrae información de diferentes inventarios y catálogos para generar y almacenar información en el inventario producción mensual.

Proceso 8.2.- Reportes Gráficos: Proceso que extrae información del inventario producción mensual y los presenta en pantalla en forma de reporte gráfico y con opción a ser mandados a impresora.

Estructura de Datos asociadas al proceso principal 8:

T17: Producción Mensual: Inventario que contiene información de la producción generada por GOSAC mensualmente.

T14.- Movimientos de Fianzas: Inventario de los movimientos realizados a las fianzas desde su expedición hasta su cancelación.

T13.- Fianzas: Inventario de las fianzas otorgadas por GOSAC.

T16.- Comisiones: Inventario que contiene la información de cada comisión por fianza y su estado.

El Diagrama de Flujo de Datos de Proceso de acumulación de producción es el siguiente:

PROCESO DE ACUMULACION

3.4 Modelización de Datos

Hay ocasiones que cuando los sistemas de información tienen una gran fluidez de información a través de los procesos, sin complicarse, la notación básica para el análisis estructurado es suficiente. Pero cuando las relaciones entre colecciones de datos son complejas, es necesario ampliar la notación del análisis estructurado para incluir un componente en la modelización de datos.

La técnica de la modelización de datos o modelización de información se centra únicamente en los datos, representando una red de datos existentes para un sistema dado. A diferencia del enfoque de análisis estructurado, la modelización de datos considera los datos independientemente del procesamiento que transforma los datos. Por lo anterior se considera un enfoque de análisis complementario al enfoque de análisis estructurado.

La modelización de datos se usa ampliamente en aplicaciones de bases de datos. Proporciona al diseño de la base de datos una amplia visión de los datos y las relaciones que gobiernan éstas.

3.4.1 Diagramas de Entidad-Relación (E-R).

En la siguiente figura 3.2 se muestra los elementos que el diagrama de Entidad-Relación (E-R) contiene, así como la notación principal de la modelización de datos.

El principal propósito del diagrama E-R es representar los objetos de datos y sus relaciones.

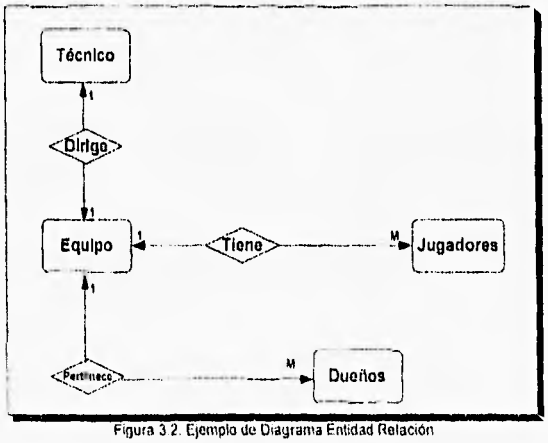


Figura 3.2. Ejemplo de Diagrama Entidad Relación

La notación del diagrama E-R es relativamente sencilla:

Los objetos de datos se representan como rectángulos etiquetados, las relaciones se indican mediante rombos, las conexiones entre los objetos de datos y las relaciones se establecen mediante variadas líneas especiales de conexión.

Para el ejemplo mostrado los objetos son: Técnico, Jugadores, Equipo y Dueño; las relaciones son: Tiene, Pertenece y Dirige; las líneas como se muestra, identifican el tipo de relación.

La relación existente entre técnico y equipo es una relación uno a uno, dado que un técnico solo dirige a un solo equipo. La relación equipo y jugadores es una relación uno a muchos, ya que un equipo está conformado por muchos jugadores. Por último la relación Equipo y Dueños es uno a muchos o uno a uno dado que un equipo pertenece a uno o muchos dueños.

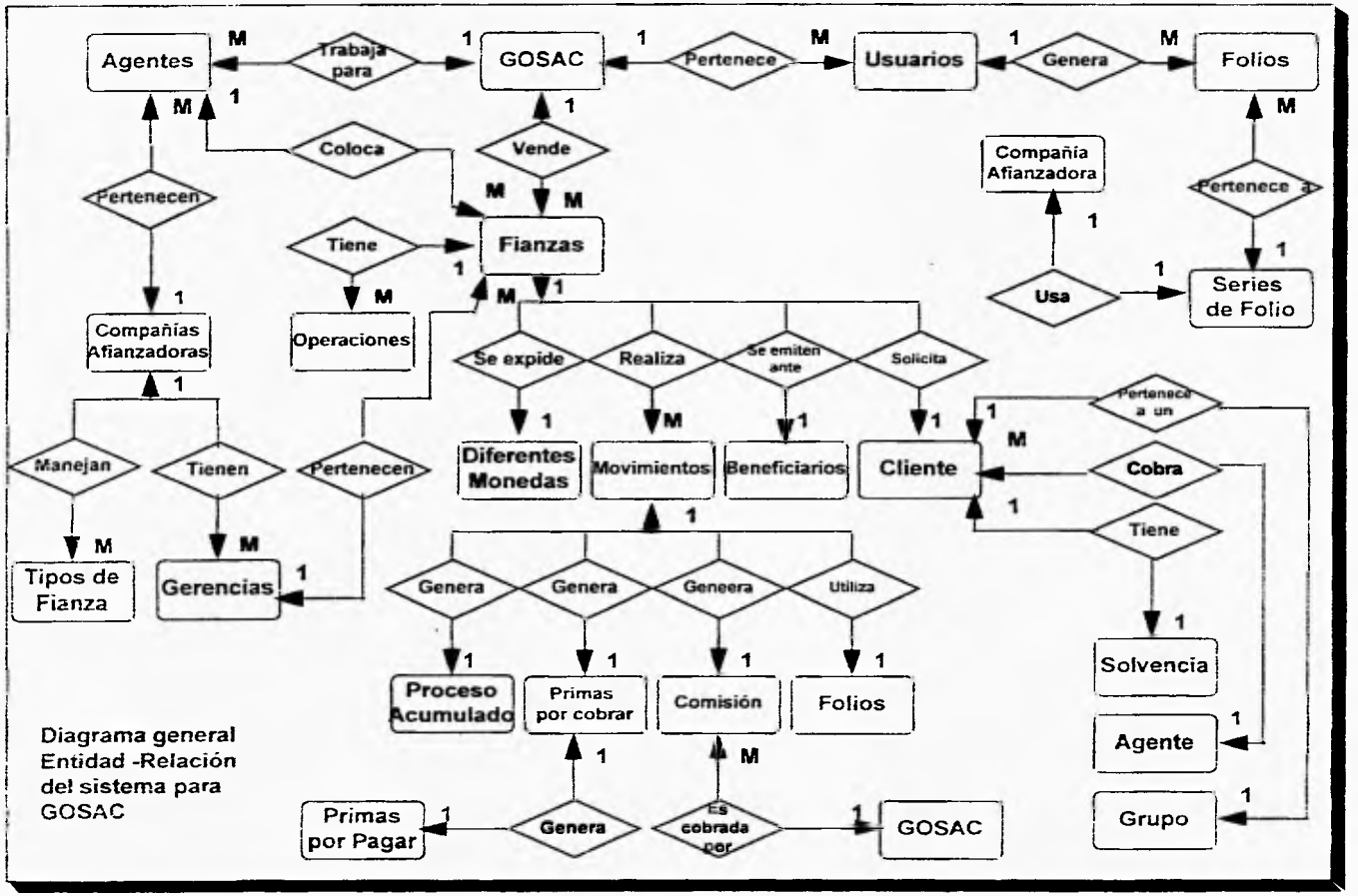
3.4.2 Diagrama de Entidad-Relación

En el diagrama de flujo de datos del sistema y en el diccionario de requisitos vistos anteriormente se reconocieron las entidades que integran al sistema en cuestión. Por la manera en que se usan y la información que contienen (el contenido se especifica en el capítulo siguiente) se definen las relaciones existentes entre ellas y el tipo de éstas.

Para la creación del diagrama general de entidad-relación para la base de datos del producto de software de la empresa GOSAC se consideraron todas las entidades (tablas) que arrojó el diccionario de requisitos.

Se analizó cuidadosamente la información y datos que representan cada una de las entidades definiendo las relaciones existentes entre ellas. Las relaciones pueden ser uno a uno, uno a muchos y muchos a muchos, se definen a partir de la forma en que se relacionan los grupos de datos contenidos en las entidades.

El resultado de todo el análisis anterior creó el siguiente diagrama general de entidad relación que a continuación se muestra en la siguiente diagrama.


Agentes
Trabaja para
GOSAC
Pertenece
Usuarios
Genera
Folios
Coloca
Vende
Pertenecen
Tiene
Fianzas
Compañía Afianzadora
Usa
Pertenece a
Series de Folio
Compañías Afianzadoras
Operaciones
Se expide
Realiza
Se emiten ante
Solicita
Manejan
Tienen
Pertenecen
Diferentes Monedas
Movimientos
Beneficiarios
Cliente
Pertenece a un
Cobra
Tiene
Tipos de Fianza
Gerencias
Genera
Genera
Geneera
Utiliza
Solvencia
Proceso Acumulado
Primas por cobrar
Comisión
Folios
Agente
Grupo
Primas por Pagar
Genera
Es cobrada por
GOSAC
Diagrama general Entidad -Relación del sistema para GOSAC

CAPÍTULO 4

Diseño del Sistema

Lo más difícil de llegar a ser sabio
son los detalles, hay que aprender a
ser distraído, olvidar el sombrero,
salir con paraguas los días de sol,
mojar la pluma en el frasco de la goma...
Ésto es lo verdaderamente difícil,
por que no se puede estar en todo.
Santiago Ramón y Cajal

Hemos modificado tan rápidamente
nuestro entorno que ahora debemos
modificarnos a nosotros mismos para
poder existir dentro de este nuevo entorno.
Norbert Wrener

Un libro es un cerebro que habla;
cerrado, un amigo que espera;
olvidado un alma que perdona;
destruido un corazón que llora...
Proverbio Hindú

CAPÍTULO 4

Diseño del Sistema

El diseño es el primer paso en la fase de desarrollo de cualquier producto o sistema de ingeniería. El diseño lo podemos definir como: "El proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física".

El objetivo del diseñador es producir un modelo o representación de una entidad que será construida más adelante. El proceso por el cual se desarrolla el modelo combina: intuición y criterios basándose en la experiencia de construir entidades similares, un conjunto de principios y/o heurísticas que guían la forma en la que se desarrolla el modelo, un conjunto de criterios que facilitan discernir sobre la calidad y un proceso de iteración que conduce finalmente a una representación del diseño final.

El diseño del software se asienta en el núcleo técnico del proceso de ingeniería de software y se aplica independientemente del modelo de desarrollo de software seleccionado. Una vez que se han establecido los requerimientos del software, el diseño de software es la primera de tres actividades técnicas: diseño, generación del código (construcción) y prueba, cada paso transforma la información de forma que finalmente se obtiene un software de computadora validado.

El flujo de información durante las tres etapas técnicas de la creación del producto de software se muestra en la figura 4.1. Los requisitos del software, establecido mediante la relación de requerimientos, el modelo de flujo de datos, y el modelo de entidad-relación ayudan y alimentan a la etapa de diseño. En la etapa de diseño se realiza el diseño de datos, el diseño arquitectónico y el diseño procedimental.

El diseño de datos transforma el modelo del campo de información, creado durante el análisis, en el diccionario y las estructuras de datos que se van a requerir para implementar el software. El diseño arquitectónico define la estructura del producto de software y la interface con el usuario. El diseño procedimental transforma los elementos estructurales en una descripción procedimental del software.

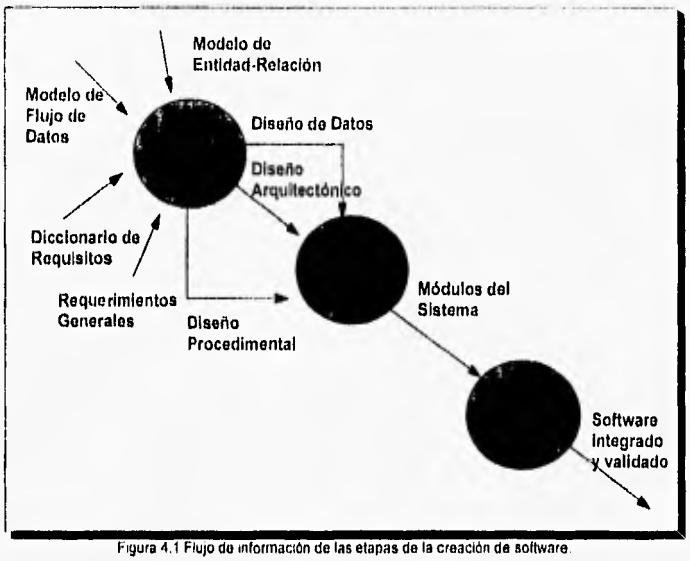


Figura 4.1 Flujo de información de las etapas de la creación de software.

La importancia del diseño de software se establece con la palabra calidad. El diseño es el lugar en donde se asienta la calidad del desarrollo del programa. El diseño nos da las representaciones del software que pueden establecerse para conseguir un producto de calidad. El diseño es la única forma mediante la que podemos traducir con precisión los requerimientos del cliente en un producto o sistema acabado.

En el modelo propuesto para el desarrollo del producto de software de la empresa GOSAC, la etapa de diseño está integrada por tres procesos:

1.- Diseño de Datos, iniciando con la creación del diccionario de datos y aplicando las reglas de normalización para la definición de las estructuras de datos.

2.- Diseño Arquitectónico, en donde se explotan los diagramas de flujo de datos para crear la estructura del producto de software orientada al flujo de datos. Y es aquí donde se incluye el diseño de la interface con el usuario.

3.- Diseño Procedimental, en el que se definen los detalles algorítmicos de los programas que integran el producto de software.

Al igual que en la etapa de análisis, cada producto de los procesos que integran esta etapa es parte de la documentación de la misma.

Para el desarrollo del presente capitulo se definen en forma muy general los fundamentos teóricos de los procesos que integran esta etapa de diseño y se aplican éstos en el desarrollo del producto de software para la empresa GOSAC.

4.1 Diseño de Datos

El diseño de datos es la primera (y la más importante) de las tres actividades de diseño realizado durante la ingeniería del software. El impacto de la estructura de datos sobre la estructura del programa y la complejidad procedimental, hace que el diseño de datos tenga una gran influencia en la calidad del software.

La actividad primaria durante el diseño de datos es seleccionar las representaciones lógicas de los objetos de datos (estructura de datos), identificados durante las fases de definición y especificación de requerimientos. El proceso de selección pueden implicar análisis algorítmicos de estructuras alternativas, en orden a determinar el diseño más eficiente o puede simplemente implicar el uso de un conjunto de módulos, que suministran la operaciones deseadas sobre alguna representación de un objeto.

Una actividad relativamente importante durante el diseño, es identificar los modelos de programa que deben operar directamente sobre las estructuras de datos lógicas. De esta forma puede restringirse el ámbito del efecto de las decisiones de diseño de datos individuales.

El diseño de datos nos da como resultado los dos siguientes productos: el diccionario de datos y las estructuras de datos.

4.1.1 El Diccionario de Datos

Un diccionario de datos es una lista de todos los elementos incluidos en el conjunto de los diagramas de flujo de datos que describen al sistema, vistos en el capítulo anterior, son el flujo de datos, el almacenamiento de datos y los procesos. El diccionario de datos almacena detalles y descripciones de estos elementos.

Si un analista u otra persona desea conocer cuantos caracteres hay en un dato, con que otros nombres se le conocen en el sistema, o donde se utilizan dentro del sistema, deben ser capaces de encontrar la respuesta en un diccionario de datos desarrollado apropiadamente.

Además el proceso de diseño de datos inicia con la creación del diccionario de datos y como vimos anteriormente se especifica la nomenclatura de cada campo, el tipo de dato que contiene y la descripción mínima necesaria para entenderlo.

El diccionario de datos es muy importante para crear sistemas utilizando modelos de cuarta generación, debido a que éste es la base de estas herramientas de programación (como se verá más adelante).

Para la creación del diccionario de datos se toman como base los diagramas de flujos de datos, el análisis de cada flujo de información agrega nuevos elementos dato que deben ser considerados en el diccionario. Para cada elemento se deben definir sus características como dato, es decir, el tipo de dato (entero, decimal, caracter, fecha, etc.), su formato y su longitud en base a la información que debe contener.

4.1.2 Diccionario de Datos del Sistema para GOSAC

La forma más sencilla de crear el diccionario de datos es seguir de lo más general a lo particular, los diagramas de flujo de datos del sistema vistos en el capítulo anterior, tomando en cuenta también el diccionario de requisitos ya que aporta la descripción de cada uno de los elementos que integran los diagramas de flujos de datos.

El diccionario de datos completo del sistema para GOSAC se encuentra en el anexo 1 al final del presente trabajo. Pero para explicar como se realizó, utilizaremos el proceso principal de solvencias, obteniendo los elementos que aporta este proceso al diccionario de datos global del sistema.

Del diagrama de flujo particular del proceso principal de Solvencias, se explotan sus dos procesos, realizando un análisis de la información que fluye y que es necesario controlar. En un principio sólo es necesario identificar los elementos dato, un proceso posterior definirá sus características y descripción para conformar el diccionario de datos.

- **Clientes:** al llegar o contactar un cliente a GOSAC y solicitar la compra de una fianza, antes de venderla GOSAC o un agente de GOSAC necesita saber algunos datos generales del cliente y registrarlos para futuras negociaciones. Cada cliente se puede identificar fácilmente por su R.F.C. (registro federal de causante) y su nombre, además se necesita conocer la clave de giro del negocio que maneja, su domicilio, el código postal, su teléfono y extensión, el nombre del contacto que lo llevó a GOSAC, y la clave del grupo a la que pertenece.

- **Solvencias:** Una vez registrado el cliente, para poder venderle una fianza hay que efectuar un análisis que demuestre que puede respaldar la transacción

que intenta realizar. Entonces se necesita tener el R.F.C. del cliente para identificar de que cliente es la solvencia, el capital contable con el que cuenta, una marca que nos diga si ya fue auditado o no, el valor de los inmuebles con los que cuenta, el valor de otras garantías que pueda tener, la línea de garantía que se le da, el total de las responsabilidades acumuladas que tiene el cliente, la fecha en que se realizó el análisis del cliente, y algunos comentarios que pueda sugerir la persona que llevó a cabo el análisis.

El análisis de los flujos de datos del módulo de solvencias de la empresa GOSAC da como resultado una relación de la información que debe manejarse en el sistema. Para cada campo debe definirse un nombre, sus características y su descripción obteniendo finalmente el diccionario de datos. La tabla 4.1 constituye el diccionario de datos para el proceso principal de solvencias dentro del sistema para el registro y control de información de la empresa GOSAC.

4.1.3 Estructuras de Datos

Una estructura de datos es un conjunto de datos que están relacionados entre sí y que describen en forma colectiva un componente del sistema. Tanto el flujo de datos como el almacenamiento de los mismos son estructuras de datos.

En el capítulo anterior se realizó una modelización entidad-relación, en la cual se obtuvieron las relaciones existentes en el entorno de base de datos del producto de software para la empresa GOSAC. Y dado que se manejara un DBMS (manejador de base de da datos relacional) la estructura fundamental de los datos será precisamente bajo el modelo relacional.

La idea de usar un modelo relacional como estructura fundamental de un sistema de manejo de base de datos fue introducida por el Dr. E. F. Codd en 1970. El proceso de cristalización de las entidades y sus relaciones en formatos de tablas usando los conceptos relacionales se llama proceso de normalización; la teoría de normalización esta basada en la observación de que cierto conjunto de relaciones presenta mejores propiedades en un medio de actualización que las que presentan otros conjuntos de relaciones que contienen los mismos datos.

Una de las principales ventajas del planteamiento relacional es su facilidad de comprensión por parte del usuario final; los usuarios finales no se tienen que preocupar por la estructura del almacenamiento físico, se les puede orientar hacia el contenido de la información de sus datos.

Existe una manera efectiva de diseñar un modelo, llamado modelo lógico y consiste en aplicar los conceptos del modelo relacional de datos. Estos conceptos

CAMPO	DEFINICIÓN	DESCRIPCIÓN
auditado	caracter(1)	Balance auditado (S/N)
cap_cont	numerico(13,2)	Capital contable del cliente
coment1	caracter(40)	Comentarios realizados por el analizador
coment2	caracter(40)	Comentarios realizados por el analizador
cos_postal_cli	caracter(5)	Código postal del cliente
cve_giro	caracter(2)	Clave del giro del cliente
cve_grupo	caracter(2)	Clave del grupo al que pertenece el cliente
domicilio_cli	caracter(50)	Domicilio del cliente
ext_tel_cli	caracter(5)	Extensión telefónica del cliente
fec_analisis	fecha	Fecha del análisis realizado al cliente
inmuebles	numerico(13,2)	Valor de los inmuebles con los que cuenta el cliente
linea_gtia	numerico(13,2)	Linea de garantia permitida por GOSAC
nom_cliente	caracter(40)	Nombre del cliente
nom_contacto	caracter(40)	Nombre del contacto
otras_gtias	numerico(13,2)	Valor de otras garantías que pueda adoptar
resp_acum	numerico(13,2)	Responsabilidades acumuladas que tiene el cliente con GOSAC
rfc_cliente	caracter(13)	Registro federal de causante que sirve como clave del cliente
tel_cli	caracter(10)	Teléfono del cliente

Tabla 4.1 Diccionario de Datos para el módulo de Solvencias

se aplican al diccionario de datos basándose en los resultados de la modelización por flujo de datos y por entidad-relación obtenidos en la etapa de análisis.

La razón de usar el proceso de normalización es asegurar que el modelo lógico de la base de datos funcionará. El proceso de normalización descompone una relación inicial universal, conteniendo todos los datos del diseño de la base (tomados del diccionario de datos) en varias relaciones más pequeñas. Cada una de las relaciones resultantes se examinan para ver si es necesaria una normalización posterior. De cualquier modo, de las relaciones derivadas finales, se pueden reconstruir cualquier información de las relaciones originales o intermedias.

El primer paso de la normalización es transformar los campos de datos a una tabla de dos dimensiones. Lo que se requiere usualmente en este paso es la eliminación de ocurrencias repetidas de campos de datos, de menera que se obtenga un archivo fijo.

El proceso de normalización identifica los datos redundantes que pueden existir en la estructura lógica, determina claves únicas necesarias para el acceso a los elementos de datos. Pueden aplicarse tres niveles de normalización llamadas formas normales.

Primera forma normal: una relación está en la primera forma normal (1FN) si todos los campos en cada registro contienen un solo valor tomado de sus dominios respectivos.

El segundo paso de la normalización es establecer las llaves y relacionarlas con los campos de datos. En la primera forma normalizada, el acceso de la tabla depende de todos los campos de llaves. En la segunda forma normalizada, se hace un intento por establecer los campos de datos que están relacionados con alguna parte de la llave completa. Si los campos de datos sólo dependen de una parte de la llave, la llave y los campos conectados a la llave parcial son susceptibles de separarse en registros independientes. La división de la primera tabla normalizada en una serie de tablas en las que cada campo sólo depende de la llave completa, se llama la segunda forma normalizada.

Segunda Forma Normal (2FN): una relación es o pertenece a la segunda forma normal si está en primera forma normal y cada atributo no llave de la relación es total y funcionalmente dependiente de su llave principal.

El tercer paso en el proceso de normalización consiste en separar los campos de las segundas relaciones normales que, aunque dependen solo de una llave, deben tener una existencia independiente en la base de datos. Esto se hace de forma tal que la información sobre estos campos pueda

introducirse separadamente a partir de las relaciones en las que se encuentra implicada.

Tercera forma normal (3FN): Una relación se encuentra en tercera forma normal si se encuentra en segunda forma normal y ningún atributo no llave en la relación es funcionalmente dependiente de algún otro atributo no llave.

A pesar de que algunos campos se pueden repetir en las relaciones normalizadas (aquellos usados como campos de conexión para enlazar relaciones), la normalización generalmente implica la reducción del almacenamiento de la base de datos. Sin embargo, la división de la relación original en varias tablas más pequeñas puede producir efectos negativos en el desempeño de la base porque se necesita el acceso a varias tablas antes de que se puedan contestar las peticiones realizadas por el usuario.

El objetivo del proceso de normalización es minimizar las anomalías encontradas en las operaciones de inserción, eliminación y actualización. Sin embargo para minimizar las anomalías de almacenamiento hay que pagar a cambio con el desempeño del sistema.

Para que el diseño de datos se pueda considerar completo es necesario realizar un proceso último a las tablas resultado del proceso de normalización; este último proceso se le conoce como desnormalización o ajustes a la normalización enfocados a obtener un mejor desempeño de las consultas realizadas a las bases de datos. Es muy importante, para este proceso, considerar los tipos de acceso a la base y la información que será solicitada por los usuarios. Este proceso de desnormalización se conjuga con la definición de métodos de acceso para obtener el desempeño de la base do datos esperado y estipulado como requisito del producto de software.

4.1.4 El proceso de normalización de datos para la empresa GOSAC.

El proceso de normalización de datos debe abarcar todo el diccionario de datos creado para la empresa GOSAC. A continuación al igual que en la sección anterior se desarrollará el proceso de normalización para uno de los módulos que integran el sistema.

El modulo a desarrollar será el del proceso principal de registro y control de fianzas. La siguiente tabla (tabla 4.2) es la relación universal de los campos que se deben considerar para el proceso de registro y control de fianzas.

La tabla anterior se obtiene del análisis de flujo de información del proceso y forma parte del diccionario de datos del sistema. La relación universal

obtenida de campos dato no es una relación normalizada de los datos utilizados en el sistema.

Tabla 4.2 Relación inicial de Fianzas No Normalizada
auditado
bis_fza
bonificación
cap_cont
cobertura
cod_ope
cod_postal_cli
coment1
coment2
contacto
cred_cnsf
cve_agt
cve_agt_afian
cve_cia
cve_edo_poliza
cve_edo_recibo
cve_gcia
cve_giro
cve_grupo
cve_grupo
cve_moneda
cve_oper
cve_tipo
cve_tipo_afian
derechos
domicilio_cli
edo_fza
ext_contacto
ext_tel_cli
fec_analisis
fec_anul_mov
fec_edo_poliza
fec_edo_recibo
fec_emis_mov
fec_emis_ult_mov
fec_emision
fec_fin_vig
Continúa

Continuación

fec_fin_vig_mov
fec_ini_ult_mov
fec_ini_vig
fec_ini_vig_mov
fec_pago_com
fec_ref_cred
fec_reg_com
fec_reg_mov
fec_registro
fec_registro_fol
gastos
gastos_cia
imp_comision
imp_maquila
imp_maquila
inmuebles
iva
iva_ger
linea_gtia
marca_cobrada
marca_cont
marca_disp
marca_pagada
monto_fza
monto_mov
nom_agt
nom_benef
nom_cia
nom_cia_ger
nom_cliente
nom_contacto
nom_grupo
nom_moneda
nom_oper
nom_tipo
num_autorizacion
num_folio
num_fza
num_liquidacion
num_poliza
num_recibo

Continúa

Continuación
origen_fol
otras_gtias
por_com
por_com_ca
por_tar
pri_fut_mov
pri_mov
prima_fza
prima_total
referencia
resp_acum
rfc_benef
rfc_cliente
rfc_cliente
serfol
serfol_p
serfol_r
suma_cobro
suma_pago
tel_cli
tel_contacto
tipo_forma
verif
verif_p
verif_r

Primera forma normal (1FN): Para obtener la primera forma normal se necesita que no existan grupos repetitivos de datos y además se necesita definir la llave primaria de la relación FIANZAS. Para identificar totalmente a una fianza necesitamos conocer la compañía, la gerencia de la compañía, el agente que la emite, el número de fianza, el bis del número, y la cobertura o inclusión.

Como se puede observar ya identificamos cual es la llave primaria de la relación FIANZAS, ahora lo que nos queda es identificar los grupos repetitivos y separarlos de la estructura. Para la llave primaria de fianzas, los valores de algunos atributos que no son llave no se pueden determinar de manera única; y precisamente este es el caso de los atributos que definen los movimientos realizados a una fianza, ya que por cada fianza se pueden presentar más de un movimiento, creando grupos repetitivos en la relación universal. Aplicando los principios que establece la 1FN se transforma la relación inicial no normalizada en dos relaciones que son MOVIMIENTOS y FIANZAS, la nueva relación llamada MOVIMIENTOS falta normalizarla, mientras que la

relación FIANZAS, se muestra en la tabla 4.3, ya se encuentra en primera forma normal.

Las letras remarcadas con negritas indican los campos que conforman la llave primaria.

Tabla 4.3 Relación de Fianzas en 1FN
cve_cia
cve_gcia
cve_agt_afian
num_fza
bis_fza
cobertura
nom_cia
gastos
imp_maquila
nom_contacto
tel_contacto
ext_contacto
nom_cia_ger
iva_gcia
cve_agt
nom_agt
cred_cnsf
fec_ref_cred
por_com_ca
cve_tipo
nom_tipo
cve_tipo_afian
por_tar
por_com
rfc_benef
nom_benef
rfc_cliente
nom_cliente
cve_giro
domicilio_cli
cod_postal_cli
tel_cli
ext_tel_cli
cve_grupo
Continúa

Continuación
cap_cont
auditado
inmuebles
otras_gtias
linea_gtia
res_acum
fec_analisis
coment1
coment2
fec_emision
fec_registro
fec_ini_vig
fec_fin_vig
cve_moneda
nom_moneda
paridad
fec_paridad
referencia
monto_fza
prima_fza
fec_emis_ult_mov
fec_ini_ult_mov
edo_fza

De la relación FIANZAS se le transfieren atributos a la relación MOVIMIENTOS y son precisamente todos aquellos que se repiten por cada ocurrencia de un movimiento. Considerando que la relación de FIANZAS se encuentra ya en primera forma normal por no tener más grupos repetitivos, aplicaremos el mismo proceso a la nueva relación MOVIMIENTOS para obtenerla en 1FN.

Aplicando los mismos principios que a la relación FIANZAS, lo primero que haremos será definir la llave primaria de la relación MOVIMIENTOS, la cual se encuentra integrada por la llave primaria de FIANZAS y la identificación de cada movimiento que es el número de recibo, un dígito que verifica el número de recibo y la operación correspondiente. Una vez realizado lo anterior lo que resta es identificar la existencia de grupos repetitivos en la relación no normalizada MOVIMIENTOS. No existen más grupos repetitivos y la tabla 4.4 nos muestra la relación MOVIMIENTOS que se encuentra en 1FN.

Tabla 4.4 Relación de Movimientos en 1FN
cve_cia
cve_gcia
cve_agt_afian
num_fza
bis_fza
cobertura
num_recibo
verif_r
cve_oper
cve_tipo
serfol
tipo_forma
num_folio
verif
cve_edo_recibo
cve_edo_poliza
fec_edo_recibo
fec_edo_poliza
origen
num_poliza
verif_p
serfol_p
cod_ope
monto_mov
pri_mov
pri_fut_mov
derechos
gastos
iva
bonificación
prima_total
fec_emis_mov
fec_ini_vig_mov
fec_fin_vig_mov
fec_reg_mov
marca_cont
fec_anul_mov
num_autorizacion
suma_cobro
suma_pago
Continúa

Continuación
marca_cobrada
marca_pagada
marca_disp
imp_comision
imp_maquila
fec_reg_com
fec_pago_com
num_liquidacion

Segunda forma normal (2FN): Una relación se encuentra en 2FN si está en 1FN y todo atributo que no sea llave es completamente dependiente de manera funcional de la llave primaria de la relación. Esta regla separa en nuevas relaciones a los atributos que dependen de elementos que conforman la llave principal de la relación.

Para aplicar la regla de normalización anterior se tienen las tablas de FIANZAS y MOVIMIENTOS en primera forma normal. Si analizamos la llave de la relación FIANZAS se observa que del atributo clave de la compañía (cve_cia) dependen otros atributos que al aplicar la segunda regla de normalización se define una estructura de una nueva relación llamada COMPAÑÍAS, ya que contiene la información de las compañías afianzadoras que maneja GOSAC. De la misma manera el atributos que define la gerencia de la fianza (cve_gcia) tiene atributos en la relación que son dependientes de él, por lo que define también una nueva estructura llamada COMPAÑÍAS-GERENCIAS. Por último el atributo cve_agt_afian también al igual que los dos atributos anteriores tiene en la relación FIANZAS otros atributos que dependen de él y forma una nueva estructura llamada AGENTES. Los atributos que dependen de estas nuevas relaciones son transferidas a éstas y dejan de pertenecer a la relación FIANZAS, y así está queda en segunda forma normal, la cual se presenta en la tabla 4.5.

Tabla 4.5 Relación de Fianzas en 2FN
cve_cia
cve_gcia
cve_agt_afian
num_fza
bis_fza
cobertura
cve_agt
nom_agt
cred_cnsf
fec_ref_cred
por_com_ca
cve_tipo
nom_tipo
cve_tipo_afian
por_tar
por_com
rfc_benef
nom_benef
rfc_cliente
nom_cliente
cve_giro
domicilio_cli
cod_postal_cli
tel_cli
ext_tel_cli
cve_grupo
cap_cont
auditado
inmuebles
otras_gtias
linea_gtia
res_acum
fec_analisis
coment1
coment2
fec_emision
fec_registro
fec_ini_vig
fec_fin_vig
cve_moneda
Continúa

Continuación
nom_moneda
paridad
fec_paridad
referencia
monto_fza
prima_fza
fec_emis_ult_mov
fec_ini_ult_mov
edo_fza

La segunda regla de normalización se debe aplicar a las relaciones que ya se encuentran en 1FN para que el proceso de normalización se encuentre en el mismo nivel en todas las relaciones.

Tercera Forma Normal (3FN): se dice que una relación está en tercera forma normal si no existe ninguna dependencia funcional transitiva entre los atributos que no son llave. Esta regla define nuevas relaciones a partir de atributos que dependen de otros que no conforman la llave principal de la relación.

Aplicando la tercera regla a la relación de FIANZAS que ya se encuentra en 2FN, se derivan las nuevas relaciones de TIPOS DE FIANZA, MONEDAS, BENEFICIARIOS, CLIENTES, FOLIOS y GRUPOS. Al igual que en las reglas anteriores, al aplicarse ésta se crean nuevas relaciones no normalizadas que deben entrar al proceso de normalización desde la primera hasta la tercera forma normal. La relación FIANZAS queda en tercera forma normal y se muestra en la tabla 4.6, con lo que se termina el proceso de normalización para la relación FIANZAS. Por otra parte las demás relaciones que surgieron en el proceso de normalización de FIANZAS también se deben normalizar hasta alcanzar la 3FN. Así entonces se llega a tener la normalización en 3FN de las siguientes relaciones, COMPAÑÍAS y se muestra en la tabla 4.7; AGENTES y se muestra en la tabla 4.8; COMPAÑÍAS-GERENCIAS y se muestra en la tabla 4.9; COMPAÑÍAS-AGENTES y se muestra en la tabla 4.10; TIPOS DE FIANZA y se muestra en la tabla 4.11; COMPAÑÍAS-TIPOS DE FIANZA y se muestra en la tabla 4.12; MONEDAS y se muestra en la tabla 4.13; BENEFICIARIOS y se muestra en la tabla 4.14; CLIENTES y se muestra en la tabla 4.15; SOLVENCIAS CLIENTES y se muestra en la tabla 4.16; GRUPOS y se muestra en la tabla 4.17.

Tabla 4.6 Relación de Fianzas en 3FN
cve_cia
cve_gcia
cve_agt_afian
num_fza
bis_fza
cobertura
cve_agt
cve_tipo
cve_tipo_afian
rfc_benef
rfc_cliente
fec_emision
fec_registro
fec_ini_vig
fec_fin_vig
cve_moneda
referencia
monto_fza
prima_fza
fec_emis_ult_mov
fec_ini_ult_mov
edo_fza

Tabla 4.7 Relación de Compañías en 3FN
cve_cia
nom_cia
gastos
imp_maquila
contacto
telefono_cli
ext_tel_cli

Tabla 4.8 Relación de Agentes en 3FN
cve_agt nom_agt cred_cnsf fec_ref

Tabla 4.9 Relación de Compañías-Gerencias en 3FN
cve_cia **cve_gcia** nom_cia_ger iva_gcia

Tabla 4.10 Relación de Compañías-Agentes en 3FN
cve_cia **cve_agt** cve_agt_afian por_com_ca

Tabla 4.11 Relación de Tipos de Fianza en 3FN
cve_tipo nom_tipo

Tabla 4.12 Relación de Compañías-Tipos de Fianza en 3FN
cve_cia
cve_tipo
cve_tipo_afian
por_tar
por_com

Tabla 4.13 Relación de Monedas en 3FN
cve_moneda
nom_moneda
paridad
fec_paridad

Tabla 4.14 Relación de Beneficiarios en 3FN
rfc_benef
nom_benef

Tabla 4.15 Relación de Clientes en 3FN
rfc_cliente
nom_cliente
cve_giro
domicilio_cli
cod_postal_cli
telefono_cli
ext_tel_cli
contacto
cve_grupo

Tabla 4.16 Relación de Solvencias en 3FN
rfc_cliente
cap_cont
auditado
inmuebles
otras_gtias
linea_gtia
resp_acum
fec_analisis
coment1
coment2

Tabla 4.17 Relación de Grupos en 3FN
cve_grupo
nom_grupo
gastos
imp_maquila
contacto
telefono_cli
ext_tel_cli

La figura 4.2 muestra gráficamente el proceso de normalización y las relaciones que se crean como resultado de aplicar las reglas de normalización a los datos. Todas las relaciones terminan en 3FN para poder aplicárseles el proceso de ajuste a la normalización.

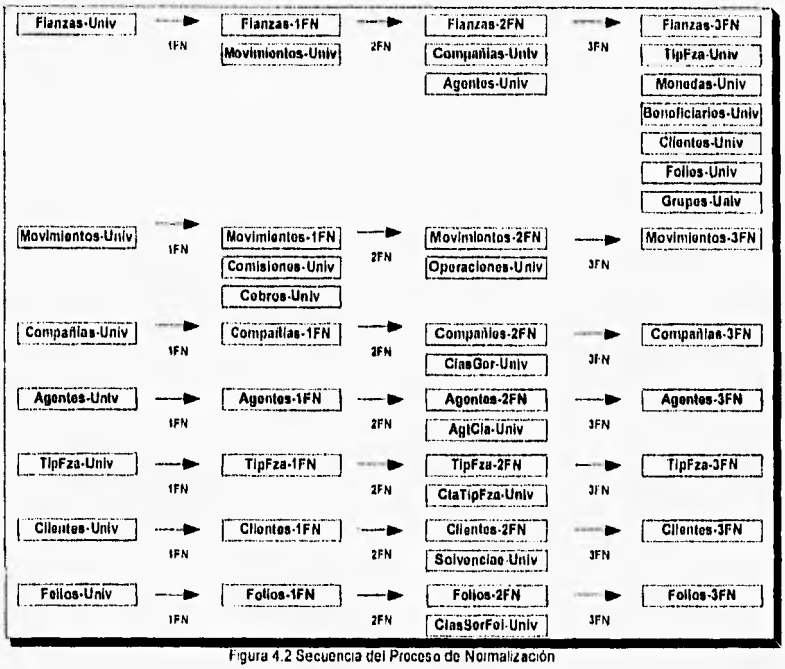


Figura 4.2 Secuencia del Proceso de Normalización

El proceso de normalización como se mencionó anteriormente, asegura que no existan fallas de inserción, actualización o borrado, pero no considera el impacto en el desempeño de las consultas realizadas a las bases de datos o la eficiencia en los procesos de datos.

El siguiente paso una vez terminado el proceso de normalización de datos al sistema, es aplicar el proceso de desnormalización o ajuste a la normalización. La desnormalización intenta modificar o ajustar las relaciones en 3FN para mejorar el desempeño general del sistema.

Para la estructura en 3FN de MOVIMIENTOS se deben de agregar algunos atributos que pertenecen a la relación FIANZAS debido a los accesos que los usuarios realizan a la información. Así los atributos de la clave del cliente (rfc_cliente), clave del agente (cve_agt), la clave del tipo de fianza (cve_tipo_afian) y la clave del grupo a la que pertenece el cliente (cve_grupo), estos atributos, se agregan a la relación para obtener la estructura de datos óptima para la entidad de MOVIMIENTOS.

Por otra parte existen movimientos que generan una comisión, aunque no todos los movimientos la generan, entonces nos vimos en la necesidad de crear una tabla auxiliar que la llamamos COMISIONES, la cual tiene la misma llave que la relación MOVIMIENTOS, incluyendo además algunos atributos que nos permitan optimar el proceso de COMISIONES, aunque aparentemente los atributos de COMISIONES podrían quedar o pertenecer a la relación de MOVIMIENTOS, pero como se mencionó anteriormente no todos los movimientos generan comisión, entonces para hacer un proceso eficiente y rápido se creó dicha estructura.

El proceso de desnormalización se debe aplicar a cada relación en 3FN resultado de la normalización para obtener las estructuras de las tablas que se deben manejar en la base de datos del sistema. La relación completa de las tablas de la base de datos se encuentra en el anexo 2.

4.2.- Diseño Arquitectónico

El objetivo principal del diseño arquitectónico es la de desarrollar una estructura modular de fácil operación al usuario y definir las interfazs que faciliten la interacción hombre-máquina.

Una forma de definir la estructura del sistema es tomar como base los diagramas de flujo de datos explotándolos hasta llegar en cada caso, a el proceso mínimo por realizar. Si los diagramas de flujo de datos llegaron al detalle de cada proceso, aquí solamente se transforman los diagramas a la estructura del sistema. La explotación genera opciones agrupadas lógicamente de acuerdo al flujo de datos al que pertenecen, por lo que el usuario (por su conocimiento de la empresa y el manejo de ésta) puede saber fácilmente en que módulo del sistema se encuentra la opción que se requiere.

La interfaz del usuario se une a la arquitectura del sistema como dos de los elementos más importantes para que el usuario entienda, confíe y utilice el sistema como parte de su trabajo diario. Si las opciones del sistema se agrupan de una manera que no comprenda el usuario, que no siga la operación lógica de la empresa o si la interfaz del sistema es muy compleja para el tipo de usuario, es predecible el rotundo fracaso del sistema.

Existen diferentes modelos de interfaces o estilos de interacción hombre-máquina, entre los que destacan: la interfaz de preguntas y órdenes, la interfaz de menú simple, la interfaz de ventanas y la interfaz de íconos. Una vez determinada la estructura o arquitectura del sistema basada en los flujos

de información, la implementación puede ser realizada independientemente del estilo de interfaz seleccionada.

4.2.1- Diseño arquitectónico y de interfaz para el software de GOSAC.

El estilo de interacción usuario-máquina definido para el software de la empresa GOSAC es el de ventanas con íconos, por ser bastante intuitivo, amigable y fácil de utilizar, además de que la herramienta de cuarta generación con la que se cuenta trabaja por medio de ventanas ya que corre bajo la plataforma windows, por lo que la creación de la estructura del sistema es rápida y simple.

La estructura de ventanas nos ofrece y nos permite la utilización de menúes e íconos que surgen de la explotación al máximo detalle de los diagramas de flujos de datos creados en el capítulo anterior. La siguiente figura 4.3 muestra la interfaz que se usará en el sistema.

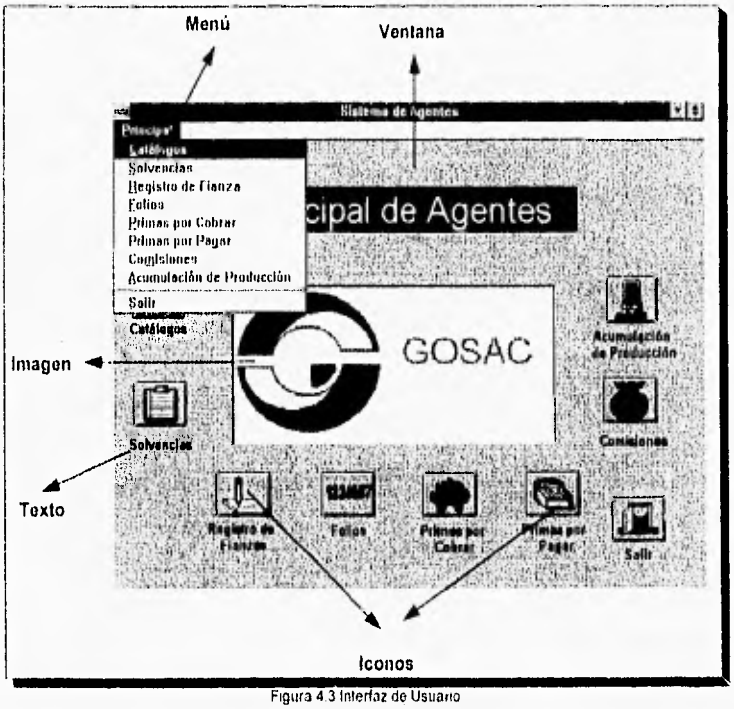


Figura 4.3 Interfaz de Usuario

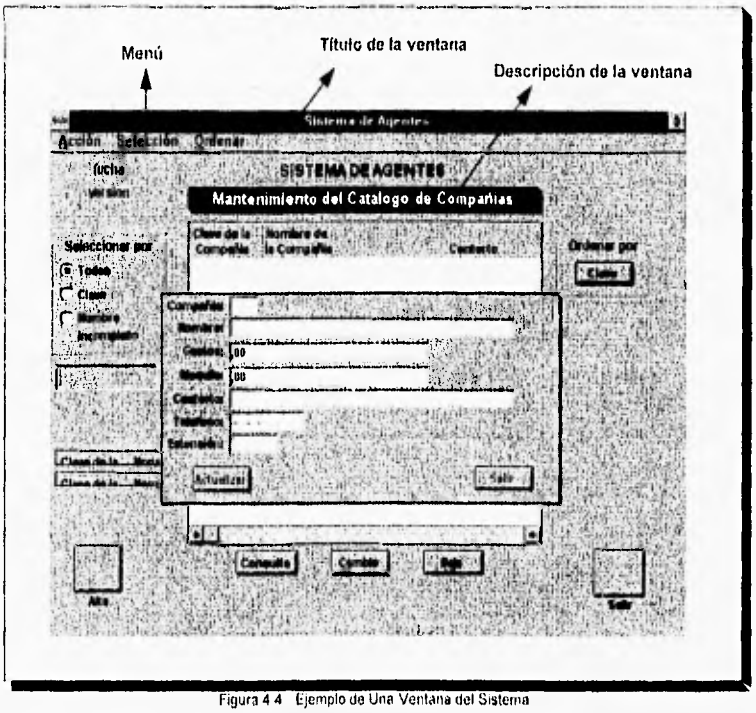


Figura 4.4 Ejemplo de Una Ventana del Sistema

A partir de la estructura funcional y de la interfaz del software puede ya construirse el prototipo del sistema. La ventaja que presenta la construcción de un prototipo como se mencionó anteriormente, es la de presentar al cliente un prototipo del producto, que sirve para la retroalimentación al grupo de desarrollo del sistema y evaluación por parte del cliente.

La herramienta de cuarta generación que se maneja es Power Builder, la cual permite y apoya la utilización de prototipos debido a la rapidez y facilidad que se tiene para crear aplicaciones simples que ejemplifican lo que será el producto final. Con el prototipo se muestra el esquema funcional del sistema, la interfaz y las aplicaciones consideradas para cubrir los requisitos del software.

4.3 Diseño Procedimental.

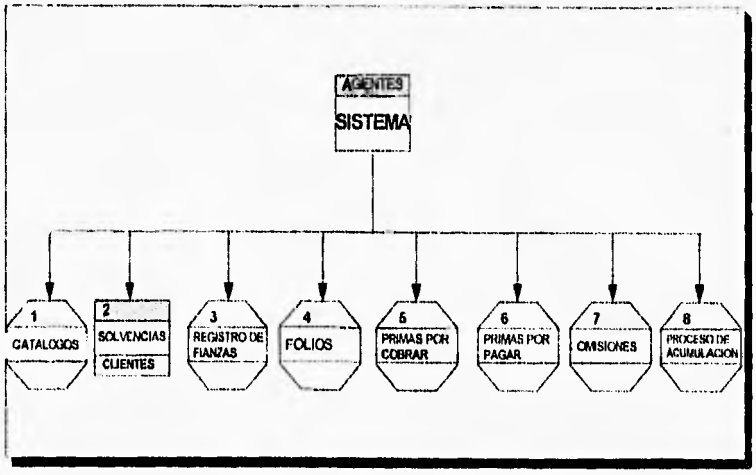
El diseño procedimental se realiza después de que se ha definido la arquitectura del sistema y las estructuras que conforman la base de datos. Lo que se realiza en este proceso de diseño, es transformar los elementos estructurales en una descripción procedimental del software.

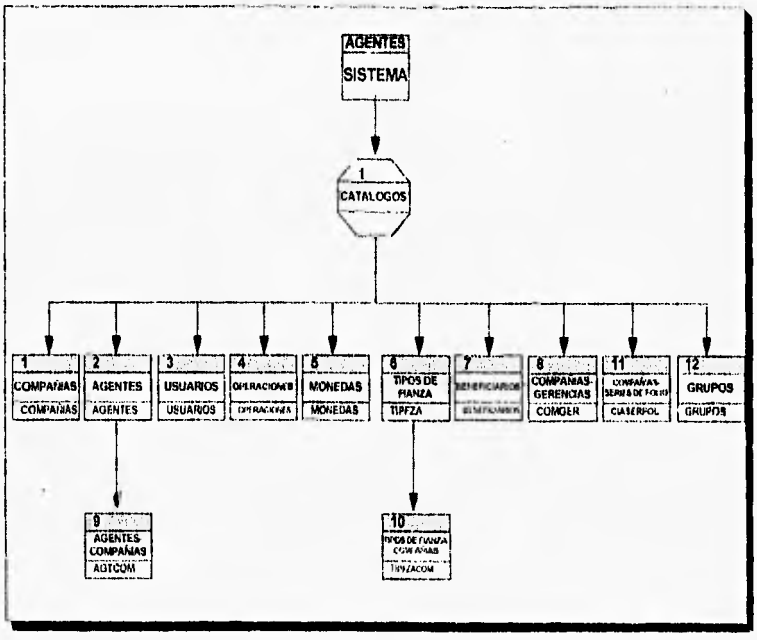
El lenguaje natural es el que utilizamos a diario para comunicarnos con otras personas y esta es su principal ventaja, aunque el propio lenguaje tiene la característica de prestarse a ambigüedades, lo cual no es deseable ya que la descripción procedimental debe ser precisa y concisa.

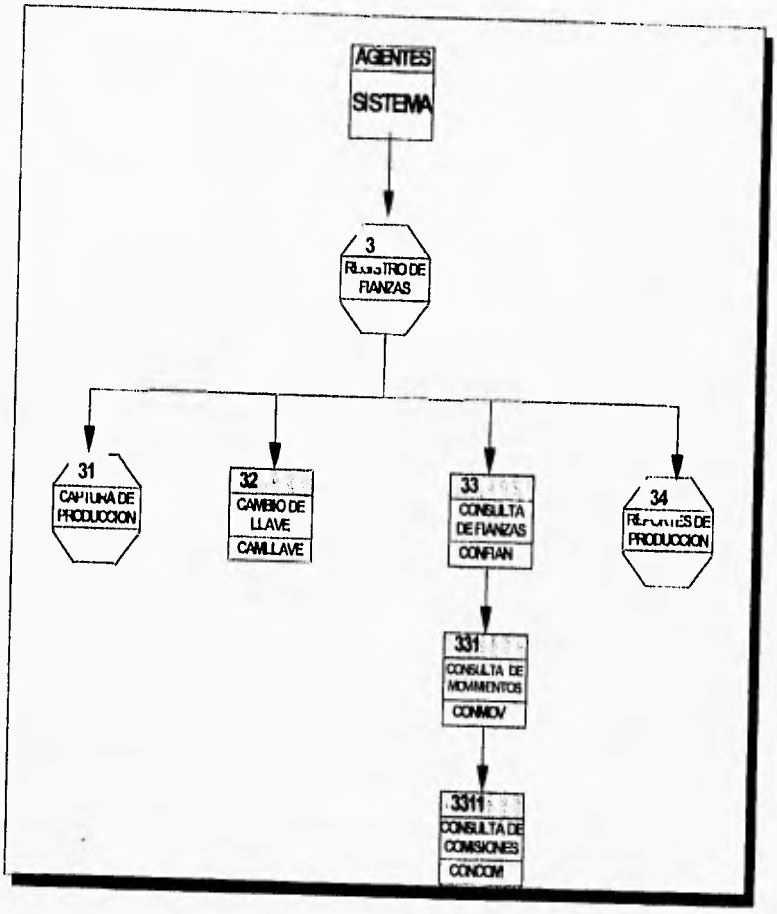
La utilización del pseudocódigo hace que la descripción procedimental sea mucho más concisa para cada parte o procedimiento, pero al revisar el pseudocódigo de un sistema para entender la estructura y funcionamiento del mismo resulta un proceso tedioso y complicado.

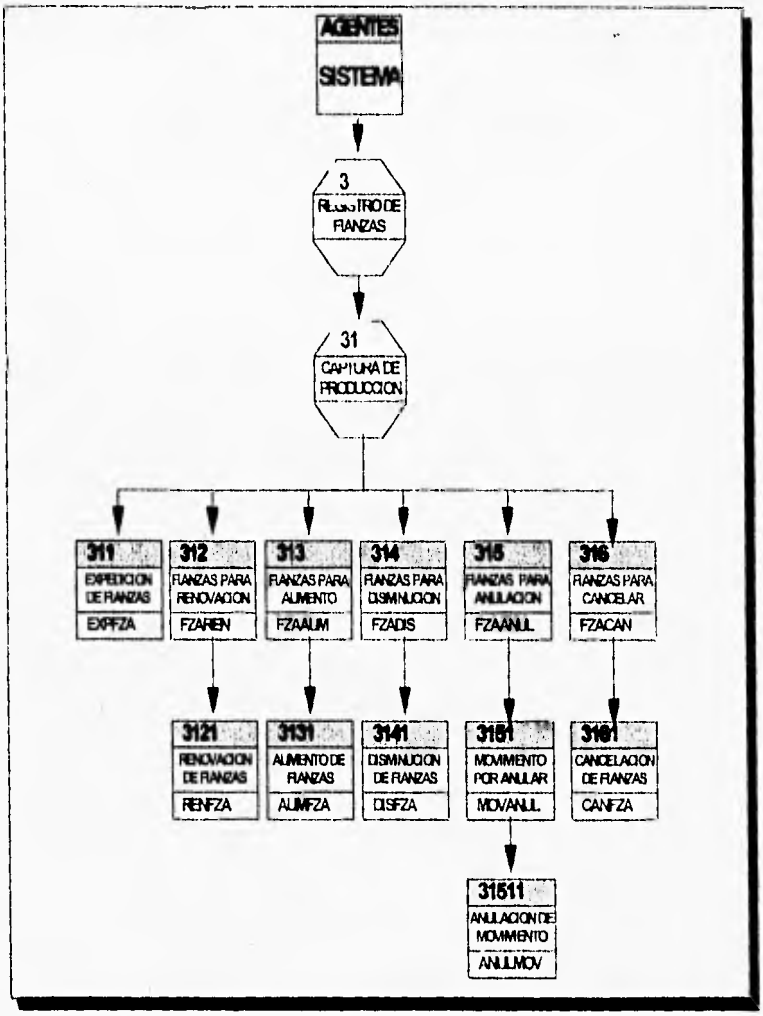
La última herramienta que revisaremos es la utilización de diagramas. Los diagramas representan un gran número de ventajas con respecto a las herramientas anteriores; primero porque permite una visualización rápida y clara de la relación entre procedimientos y su forma de operación; no se presta a ambigüedades debido a que tiene que ser muy específico y cada procedimiento no involucra más de una función; existe una gran gama de simbologías y distintos métodos de diagramación, según la necesidad de los diseñadores que las utilizan.

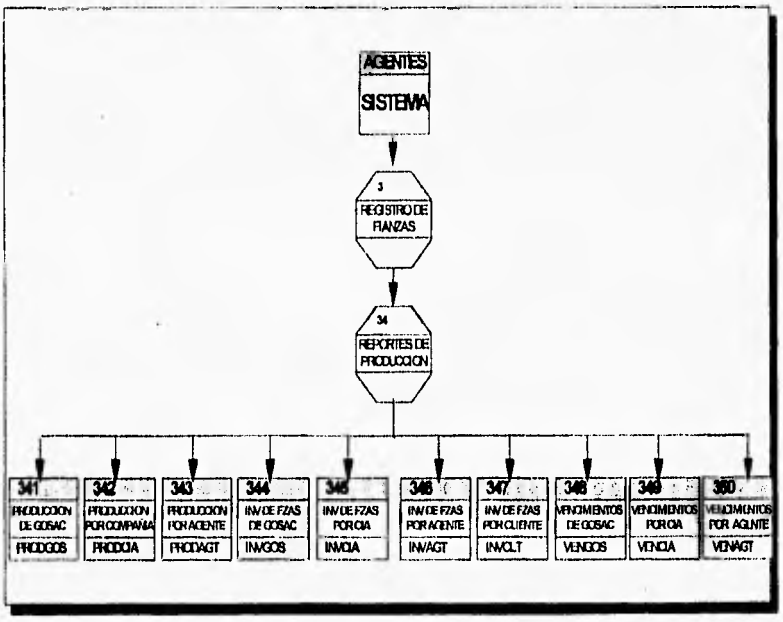
Como se vio en los párrafos anteriores, la mejor opción para realizar el diseño procedimental es utilizar la herramienta de diagramación, por lo que en las siguientes páginas encontramos los diagramas de diseño para el sistema.

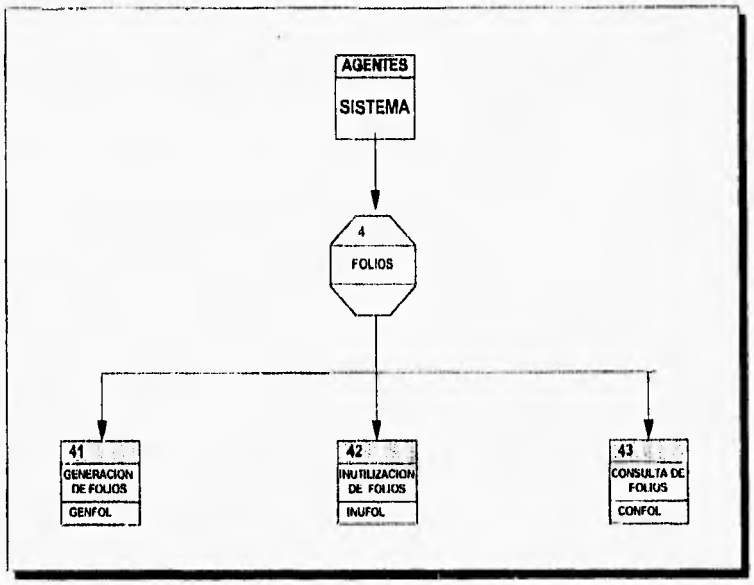

AGENTES
SISTEMA
1
CATALOGOS
2
SOLVENCIAS
CLIENTES
3
REGISTRO DE FIANZAS
4
FOLIOS
5
PRIMAS POR COBRAR
6
PRIMAS POR PAGAR
7
OMISIONES
8
PROCESO DE ACUMULACION

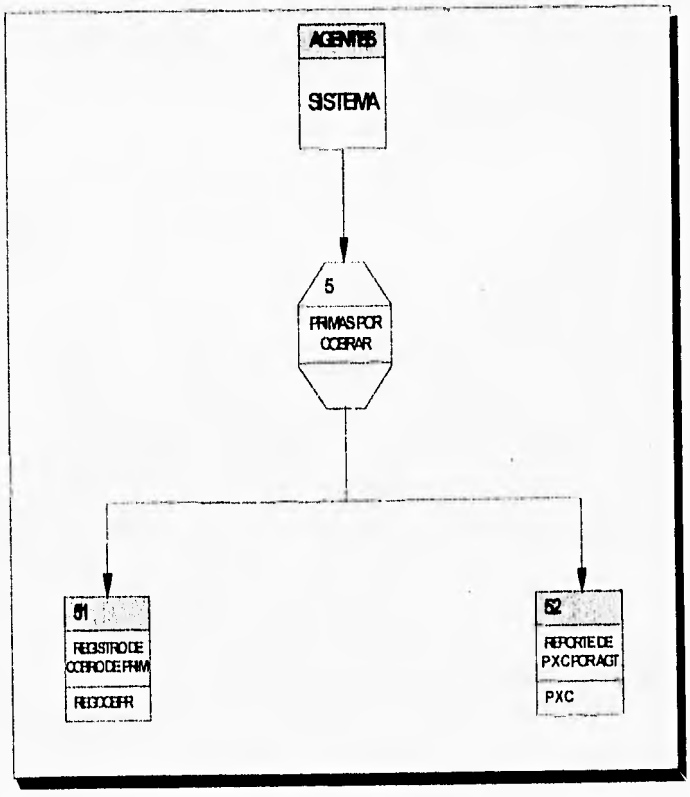

AGENTES
SISTEMA
1
CATALOGOS
1
COMPAÑIAS
COMPAÑIAS
2
AGENTES
AGENTES
3
USUARIOS
USUARIOS
4
OPERACIONES
OPERACIONES
5
MONEDAS
MONEDAS
6
TIPOS DE FIANZA
TIPFZA
7
BENEFICIARIOS
BENEFICIARIOS
8
COMPAÑIAS GERENCIAS
COMGER
11
COMPAÑIAS SERIES DE FOLIO
CIASERFOL
12
GRUPOS
GRUPOS
9
AGENTES COMPAÑIAS
AGTCOM
10
TIPOS DE FIANZA COMPAÑIAS
TIPFZACOM

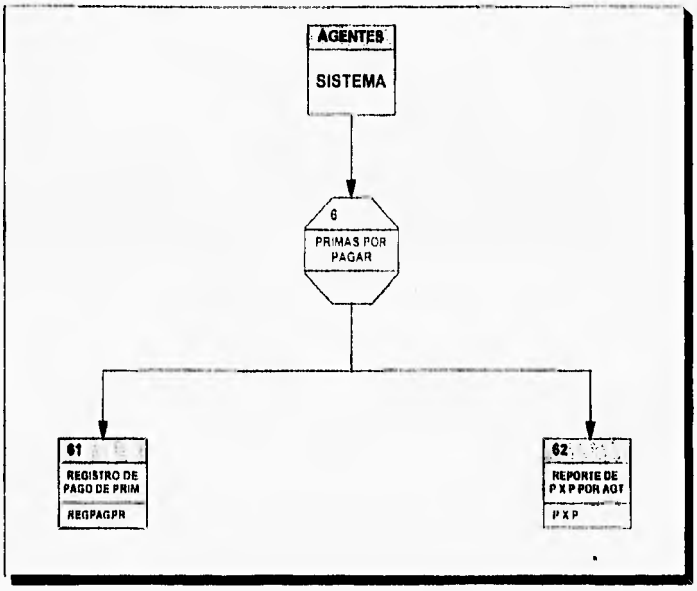

AGENTES
SISTEMA
3
REGISTRO DE FIANZAS
31
CAPTURA DE PRODUCCION
32
CAMBIO DE LLAVE
CAMLLAVE
33
CONSULTA DE FIANZAS
CONFIAN
34
331
CONSULTA DE MOVIMIENTOS
CONMOV
3311
CONSULTA DE COMISIONES
CONCOM

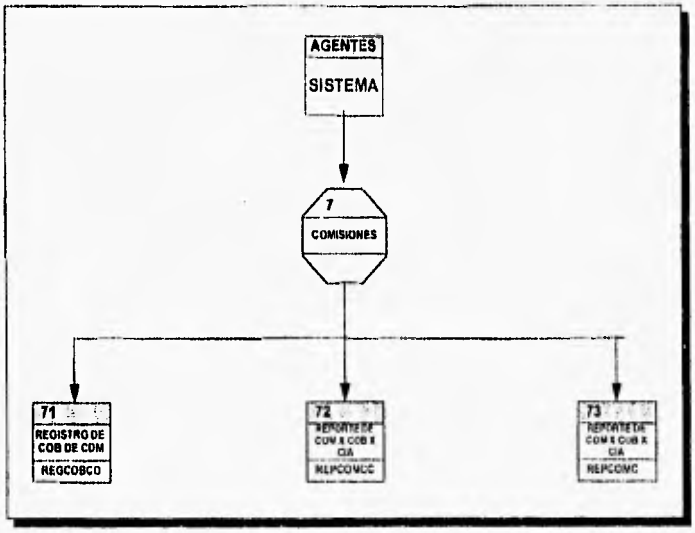

AGENTES
SISTEMA
3
REGISTRO DE FIANZAS
31
CAPTURA DE PRODUCCION
311
EXPEDICION DE FIANZAS
EXPFZA
312
FIANZAS PARA RENOVACION
FZAREN
313
FIANZAS PARA AUMENTO
FZAAUM
314
FIANZAS PARA DISMINUCION
FZADIS
315
FIANZAS PARA ANULACION
FZAANUL
316
FIANZAS PARA CANCELAR
FZACAN
3121
RENOVACION DE FIANZAS
RENFZA
3131
AUMENTO DE FIANZAS
AUMFZA
3141
DISMINUCION DE FIANZAS
DISFZA
3151
MOVIMIENTO POR ANULAR
MOVANUL
3161
CANCELACION DE FIANZAS
CANFZA
31511
ANULACION DE MOVIMIENTO
ANULMOV

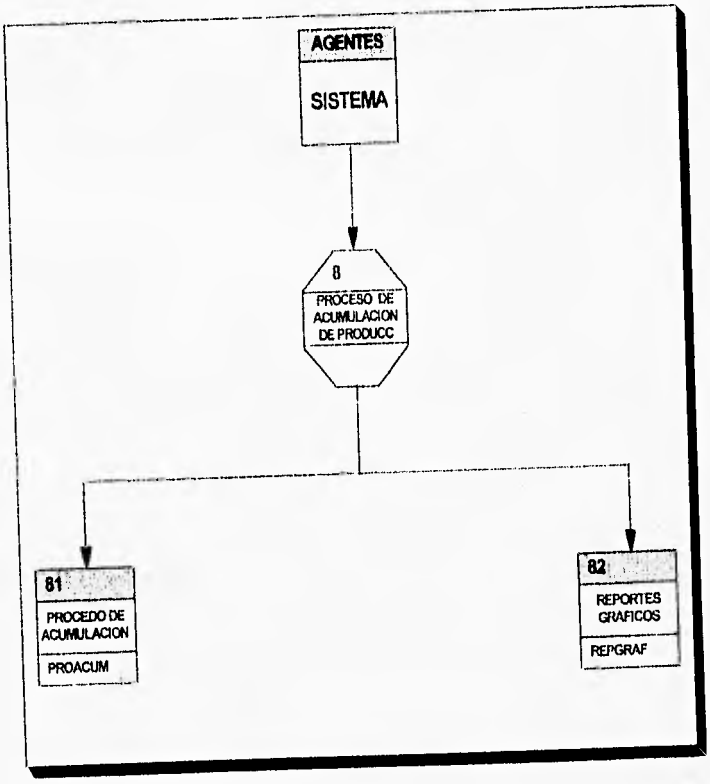

AGENTES
SISTEMA
3
REGISTRO DE FIANZAS
34
REPORTES DE PRODUCCION
341
PRODUCCION DE GOSAC
PRODGOS
342
PRODUCCION POR COMPAÑIA
PRODCIA
343
PRODUCCION POR AGENTE
PRODAGT
344
INV DE FZAS DE GOSAC
INVGOS
345
INV DE FZAS POR CIA
INVCIA
346
INV DE FZAS POR AGENTE
INVAGT
347
INV DE FZAS POR CLIENTE
INVCLT
348
VENCIMIENTOS DE GOSAC
VENGOS
349
VENCIMIENTOS POR CIA
VENCIA
350
VENCIMIENTOS POR AGENTE
VENAGT


AGENTES
SISTEMA
4
FOLIOS
41
GENERACION DE FOLIOS
GENFOL
42
INUTILIZACION DE FOLIOS
INUFOL
43
CONSULTA DE FOLIOS
CONFOL


AGENTES
SISTEMA
5
PRIMAS POR COBRAR
51
REGISTRO DE COBRO DE PRIM
52
REPORTE DE PXC POR AGT
PXC


AGENTES
SISTEMA
6
PRIMAS POR PAGAR
61
REGISTRO DE PAGO DE PRIM
REGPAGPR
62
REPORTE DE P X P POR AGT
P X P


AGENTES
SISTEMA
7
COMISIONES
71
REGISTRO DE COB DE COM
REGCOBCO
72
REPORTE DE COM X COB X CIA
REPCOMCC
73
REPORTE DE COM X COB X CIA
REPCOMC


AGENTES
SISTEMA
8
PROCESO DE ACUMULACION DE PRODUCC
81
PROCEDO DE ACUMULACION
PROACUM
82
REPORTES GRAFICOS
REPGRAF

CAPÍTULO 5

Desarrollo del Sistema

La ciencia no me interesa. Me parece
presuntuosa, analítica y superficial.
Ignora el sueño, el azar, la riza,
el sentimiento, y la contradicción,
cosas todas que me son preciosas.

Luis Buñuel (1900-1983)
Director cinematográfico español

El que no sabe, y no sabe que no sabe es un necio; evítalo.
El que no sabe y sabe que no sabe es un ignorante; enséñale.
El que sabe y no sabe que sabe esta dormido; despiértale.
El que sabe y sabe que sabe es un sabio; síguele.

San Mateo, apóstol

CAPÍTULO 5

Desarrollo del Sistema

Los pasos anteriormente realizados de la ingeniería de software van dirigidos hacia un objetivo final: traducir las representaciones del software a una forma que pueda ser comprendida por la computadora. Hemos llegado al paso de la creación de la base de datos y la codificación de los programas. Un proceso que transforma el diseño en un lenguaje de programación y que da la pauta para la división lógica que integran la construcción de la base de datos y la programación del sistema.

5.1 Implementación de la base de datos.

Antes de construir la base de datos del sistema, es necesario conocer algunos conceptos básicos para tener una mejor idea de que es lo que se realizó y cómo se llevó a cabo, a continuación se mencionan los conceptos necesarios para entender la construcción de la base de datos.

5.1.1 ¿Qué es un sistema de bases de datos?

En esencia un sistema de base de datos es un sistema de mantenimiento de registros basados en computadora, es decir, un sistema cuyo propósito general es registrar y mantener información. Tal información puede estar relacionado con cualquier cosa que sea significativa para la organización donde el sistema opera. En la siguiente figura 5.1 se muestra una representación simplificada de un sistema de base de datos, Los cuatro componentes principales que integran el sistema son: datos, hardware, software y usuarios.

Datos: Los datos almacenados en el sistema se dividen en una o más bases de datos. Aunque se puede pensar desde el punto de vista didáctico que sólo hay una base de datos, la cual contiene todos los datos almacenados en el sistema. Una base de datos es un repositorio de datos almacenados y en general es integra y compartida. Es integra por que la base de datos puede considerarse como una unificación de varios archivos independientes donde se elimina parcial o totalmente cualquier redundancia entre los mismos. Y es compartida porque, partes individuales de la base de datos pueden compartirse entre varios usuarios distintos, en el sentido de que cada uno de ellos puede tener acceso a la misma parte de la base de datos y utilizarla con propósitos diferentes.

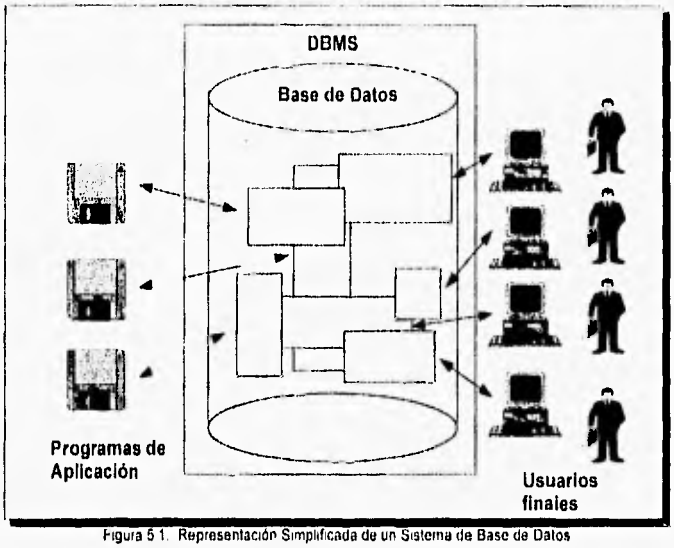


Figura 5.1. Representación Simplificada de un Sistema de Base de Datos

Hardware: El hardware se compone de los volúmenes de almacenamiento secundario (discos duros, tambores magnéticos), donde reside la base de datos, junto con dispositivos asociados como las unidades de control.

Software: Entre la base de datos física en si (es decir, el almacenamiento real de los datos) y los usuarios del sistema existe un nivel del software, que a menudo recibe el nombre de sistema de administración de base de datos o DBMS.

Usuarios: Se considera aquí tres clases de usuarios: 1) programador de aplicación, es el encargado de escribir los programas de aplicación que utilice la base de datos; 2) usuarios finales, es el que recurre a un programa de aplicación escrito por un usuario programador que acepte órdenes desde la terminal y a su vez formule solicitudes al DBMS en nombre del usuario final; 3) administrador de la base de datos, es el encargado de crear, mantener y administrar la base de datos.

Resumiendo un sistema de base de datos es un sistema computarizado de información para el manejo de datos por medio de paquetes de software llamados sistemas de manejo de bases de datos (Data Base Management System (DBMS)). Los componentes principales de un sistema de bases de datos son el software DBMS y los datos por manejar.

Las ventajas que nos da utilizar una base de datos es poder reducir la redundancia, poder evitar la inconsistencia (al menos en cierta medida), compartir datos, poder hacer cumplir las normas establecidas, aplicar restricciones de seguridad y conservar la integridad.

En el capítulo anterior de Diseño se definieron las tablas y estructuras de datos que se manejarán y usarán en nuestro sistema. La construcción de la base de datos del sistema depende entonces de las características propias del DBMS y del modelo de datos seleccionados para el sistema en cuestión.

5.1.2 Sistema de Manejo de Bases de Datos (DBMS)

Un sistema administrador de bases de datos es un paquete de programación que corre en el sistema operativo central como un largo y sofisticado programa de aplicación. Contiene rutinas relacionadas que efectúan funciones especializadas para el procesamiento de las base de datos. En la mayoría de los casos el DBMS depende del sistema operativo central para el manejo de los datos.

Los DBMS se pueden clasificar dentro de cualquiera de una de las tres siguientes categorías: red, jerárquica o relacional. La clasificación se basa en la estructura lógica a nivel externo. Estas estructuras lógicas constituyen la manera en que el usuario percibe la estructura de la base que será mapeada por el DBMS hacia su almacenamiento físico.

El sistema DBMS proporciona al usuario un lenguaje especial de descripción de datos (Data Desciption Language (DDL)) para describir esquemas conceptuales y externos, así como distintos lenguajes de manejo de datos (Data Manipulation Language (DML)) para manejar los datos de la base. El tipo de estructura lógica de la base que usa un DBMS no sólo dicta el diseño de su esquema y subesquema DDL sino también el modo de sus operaciones de entrada/salida.

Dado que el DBMS que contiene la herramienta Power Builder es del tipo relacional, a continuación se analiza a más detalle esta categoría.

En el sistema DBMS relacional, una estructura lógica se representa por medio de tablas bidimensionales llamadas relaciones. Una entidad se representa por un renglón de la tabla.

La ventaja principal que se tiene con el enfoque relacional sobre los de red y jerárquico está en la simplicidad de su representación en la estructura lógica de la base de datos y la flexibilidad para establecer relaciones de datos por medio de campos de conexión. Todas las entidades en una base de

datos relacional están representadas como tablas separadas y no están colocadas en ninguna jerarquía fija, como es el caso de los árboles o estructuras plex. La mayoría de los usuarios finales probablemente entiendan las tablas mucho mejor de lo que entenderían una estructura compleja de red.

El enfoque relacional hace posible el alcanzar mayor independencia de los datos usando campos de conexión en lugar de apuntadores para enlazar registros relacionados en diferentes archivos (o relaciones).

Una característica única del sistema relacional es su independencia de trayectorias de entrada/salida. Ya que una base de datos relacional consta de una colección de tablas separadas, cualquier tabla (o relación) se puede accesar directamente sin necesidad de accesar otras relaciones basadas en una estructura de datos fija, tales como un árbol o una red. El sistema relacional permite recuperar una tabla (o un conjunto de registros en vez de un solo registro) con una sola proposición DML.

5.1.3 Arquitectura de un sistema relacional

Para explicar la arquitectura relacional utilizamos la figura 5.2 y en la cual observamos tres niveles de abstracción y los componentes que integran la arquitectura relacional.

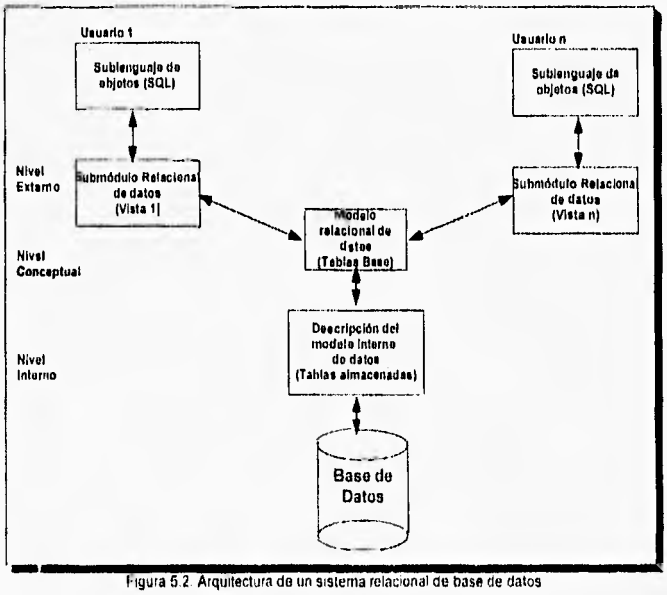


Figura 5.2. Arquitectura de un sistema relacional de base de datos

1) Esquema de almacenamiento: en el nivel interno, cada tabla base se implanta como un archivo lógico almacenado (que puede estar conformado por varios archivos físicos). Para las recuperaciones sobre las llave principal o secundaria se puede establecer uno o más índices para accesar el archivo.

2) Modelo relacional de datos: En el nivel conceptual, el modelo relacional de datos esta representado por una colección de relaciones almacenadas (también llamadas tablas base). Cada registro de tipo conceptual en un modelo relacional de datos se implanta como un archivo almacenado distinto.

3) Submodelo de datos: Los esquemas externos de un sistema relacional se llaman submodelos relacionales de datos; cada uno consta de uno o más escenarios o vistas (views) para describir los datos requeridos por una aplicación dada. Una vista puede incluir datos de una o más tablas.

Cada programa de aplicación está provisto de un buffer (también llamado área de trabajo del usuario) donde el DBMS puede depositar los datos recuperados de la base para su procesamiento, o puede guardar temporalmente sus salidas antes de que el DBMS las escriba en la base de datos.

4) Sublenguaje de datos: El sublenguaje de datos es un lenguaje de manejo de datos para el sistema relacional. Originalmente, se propusieron dos sublenguajes de datos· a) álgebra relacional, b) cálculo relacional. Estos sublenguajes fueron propuesto por Codd que además demostró que ambos lenguajes son relacionalmente completos y equivalentes, estó es, cualquier relación que puede derivarse de una o más tablas de datos también se puede derivar con un solo comando del sublenguaje. Por tanto el modelo de operación de entrada/salida en un sistema relacional se puede procesar en la forma de una tabla a la vez en lugar del procesamiento tradicional de un registro a la vez. En otras palabras, se puede recuperar una tabla en vez de un solo registro con la ejecución de un comando del sublenguaje de datos.

El álgebra relacional es un lenguaje de procedimientos de alto nivel que permite el uso de operadores para derivar la tabla deseada en diversos pasos desde las tablas base originales en el modelo relacional, o también otras tablas derivadas intermedias. Por otro lado, el cálculo relacional es un potente lenguaje de consulta que permite al usuario la recuperación de datos mediante el establecimiento de condiciones de consulta sin necesidad de codificar en forma detallada los pasos de la recuperación.

Aunque el álgebra relacional no es tan accesible para el usuario como lo es el cálculo relacional, proporciona los conceptos básicos para el desarrollo de lenguajes relacionales de manejo de datos. Los dos principales sublenguajes

relacionales, SQL (Structured Query Language) y QBE (Query By Example), están implementados con el mismo espíritu que el cálculo relacional.

Una de las principales características del SQL es que se puede usar no solo como DML para manejar los datos de la base sino también como lenguaje de definición de datos (DDL) para definir tablas base, escenarios o vistas y los archivos almacenados en los niveles conceptual, externo e interno. El SQL es un lenguaje flexible cuyos comandos se pueden emitir ya sea interactivamente desde una terminal o incluirse en el lenguaje del programa principal. Con estas características, el SQL es el lenguaje que se ha adoptado en los principales DBMS relacionales.

5) Usuarios: Éstos realizan accesos a la base de datos mediante proposiciones del sublenguaje de datos o en el caso de los usuarios programadores, algunas veces pueden hacer uso del SQL inmerso el cual representa un acoplamiento dol SQL y el lenguaje huésped. Casi cualquier proposición del lenguaje SQL puede estar inmersa en el lenguaje huésped.

Siguiendo con el tema de los DBMS relacionales, la gran mayoría de éstos constan de dos subsistemas principales:

1) Sistema de indagación de almacenamiento (Research Storage System (RSS))

2) Sistema relacional de datos (Relational Data System (RDS))

En esencia el RDS proporciona la interfaz del usuario externo, que soporta las estructuras de datos tabulares y los operadores sobre esas estructuras (SQL generalmente), y el RSS proporciona al RDS una interfaz de registros almacenados, a continuación consideramos con más detalle cada subsistema.

La siguiente figura 5.3 muestra como el sistema de indagación de almacenamiento (RSS) es una interface interna que maneja los datos almacenados, índices, buffers, distribución del almacenamiento, etc. Sin embargo, el RSS confía en los métodos de acceso subyacentes del sistema operativo para el manejo de los archivos de bajo nivel.

El RDS es una interface similar al sistema de control de la base de datos. Sus funciones principales son el análisis sintáctico y la optimización. Cuando el sistema recibe una proposición SQL, se le analiza gramaticalmente con base a la gramática del lenguaje para determinar si pertenece al lenguaje y cual es su estructura. El optimizador de RDS escogerá una trayectoria eficiente de acceso y el comando SQL se traducirá a los comandos adecuados para llamar rutinas de control de ejecución de RDS.

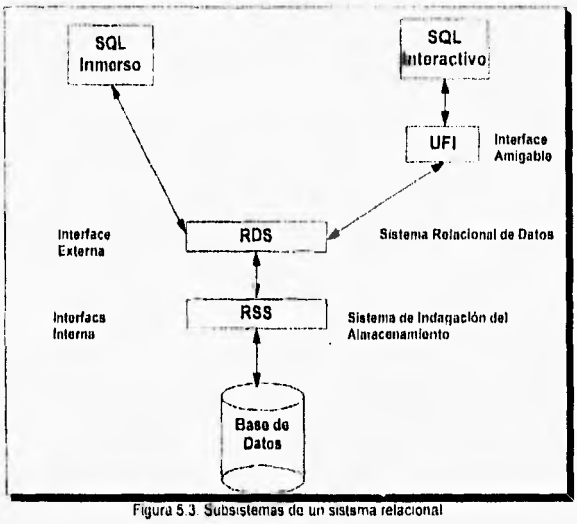


Figura 5.3. Subsistemas de un sistema relacional

5.1.4 Almacenamiento de los datos y métodos de acceso

El buen funcionamiento de los programas que interactuan con la base de datos depende en gran medida de la forma en que los datos son almacenados y accesados. Un sistema de manejo de base de datos generalizado utiliza algunos métodos de acceso del modelo interno junto con los métodos de acceso especializados disponibles por el modelo externo. El modelo interno es el modelo físico, es la forma en la cual se guarda la información y el modelo externo es el punto de vista del usuario, aunque existe el modelo conceptual, el cual tiene relación con el modelo interno la mayoría de las veces. Casi en todos los DBMS, el diseñador de la base de datos tiene la flexibilidad de escoger y combinar varios métodos de acceso que estén disponibles por el DBMS.

Al ir en aumento el número de entradas y salidas físicas que se necesitan para recuperar los datos que satisfagan la demanda del usuario, el desempeño del funcionamiento del sistema puede decrecer. El ciclo comienza con la demanda del usuario y termina con la satisfacción de ésta, pero antes tiene que pasar por varias interfaces que se muestran el la siguiente figura 5.4

Interface 1: Una vez realizada la demanda por parte del usuario, el DBMS reconoce la descripción del punto de vista del usuario, del programa de aplicación y de las condiciones de seguridad. Con base a las características anteriores sabe que base de datos accesar y de ésta toma la descripción de la base de datos para saber que métodos de acceso al modelo interno se pueden utilizar.

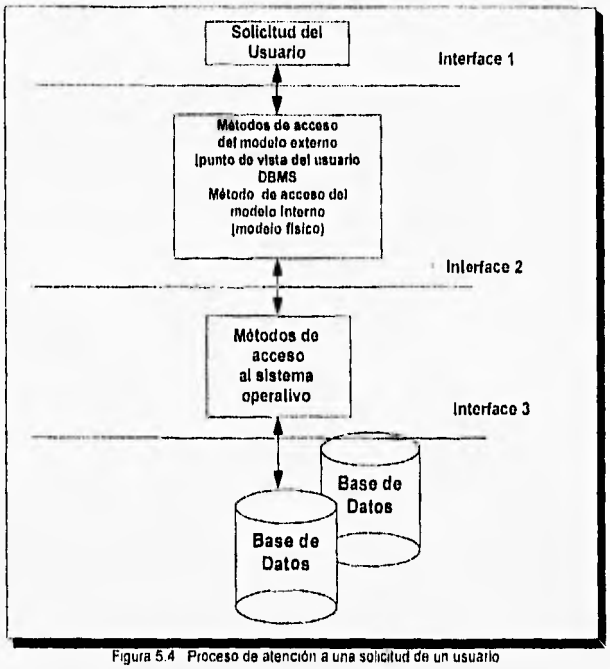


Figura 5.4 Proceso de atención a una solicitud de un usuario

Interface 2: El DBMS utiliza un método de acceso al modelo interno. Los métodos de acceso son métodos especializados proporcionados por el DBMS y otros están compuestos de métodos de acceso generalizado proporcionados por el sistema operativo.

Interface 3: Los métodos de acceso del modelo interno y los métodos de acceso del sistema operativo accesan los registros de la base de datos física; una vez que estos registros se han accesado, se encuentran los registros requeridos utilizando los métodos de acceso que describen la relación lógica entre las distintas partes del registro de la base de datos.

Los métodos de acceso del sistema operativo y los de los modelos internos y externos nos permiten recuperar datos de la base de datos física para presentarlos al DBMS. El DBMS tiene como tarea decir que parte de los datos se presentan al usuario, en que formato, etc.

Por lo que el diseñador de la base de datos debe tomar en cuenta todos los métodos de acceso que el DBMS ofrece y determinar el óptimo para cada

caso que aparezca en los programas de aplicación. En la mayoría de los DBMS el diseñador define indices como método principal de acceso a la información para que el DBMS, al precompilar las sentencias del SQL del programa, utilice los índices para accesar los datos. Otro aspecto importante a considerar son las consultas no planeadas para incluir indices y hacer más eficiente los accesos.

5.1.5 Creación de la base de datos para la empresa GOSAC.

Dado que el sistema se realizará utilizando Power Builder y además su propio DBMS es WATCOM, es conveniente mencionar algunas de sus caracterlsticas más importantes:

El WATCOM es un DBMS relacional que incluye como manejo y definición de datos al lenguaje SQL.

La información de la base de datos está almacenada en un archivo que reconoce el sistema operativo, al crearse la base de datos se define el nombre de la base, el directorio en donde se creará, y si es necesario también se especifican los subdirectorios para almacenar la base de datos. El nombre de la base de datos contendrá un máximo de 8 caracteres permisibles y la extensión por default es .db.

Las tablas que maneje la base de datos, sus especificaciones, sus Indices y sus llaves son también almacenadas en el mismo archivo creado anteriormente y el DBMS se encarga del manejo de las tablas y de las relaciones existentes entre ellas.

La tablas son definidas desde el DBMS, al momento de crear una tabla el DBMS nos solicita el nombre de la tabla para posteriormente solicitar los campos que contendrá dicha tabla, asl como el tipo de dato que se almacenará en cada uno de los campos (entero, real, fecha, etc.), aquí mismo se podrán dar las reglas de validación para cada campo, asl como también se podrá definir las máscaras con las que se desee pedir los datos de los campos. Por último nos pedirá la llave primaria de la tabla.

Una vez creado las tablas uno puede crear Indices para las tablas y asi tener diferentes métodos de acceso a la información, el DBMS presenta la particularidad de manejar dos tipos de índices para los accesos a la información contenida en las tablas : Indices Cluster e Indices Uncluster.

Una tabla puede tener un número ilimitado de indices de cualquiera de los dos tipos, pueden crearse sobre un solo campo o la unión de varios campos y éstos no necesariamente tienen que ser contiguos en la tabla. Cada índice puede definirse como único (y no permitir llaves duplicadas) o no único.

El proceso para la creación de la base de datos fue el siguiente:

1.- Creación de la base de datos
    1.1.- Seleccionar el directorio
    1.2.- Asignar el nombre físico y lógico de la base de datos

2.- Creación de las tablas
    2.1.- Asignar el nombre de la tabla
    2.2.- Asignar el nombre de los campos
    2.3.- Asignar el tipo de datos que manejarán los campos
    2.4.- Asignar la validación y la máscara para cada campo
    2.5.- Asignar la llave primaria a la tabla (puede existir o no)

3.- Creación de los índices de las tablas
    3.1.- Seleccionar la tabla
    3.2.- Seleccionar los campos índices
    3.3.- Seleccionar el tipo de índice
    3.4.- Seleccionar si es único o no

Para la base de datos de la empresa GOSAC se creó la base llamada AGENTES.DB y las tablas fueron las que se definieron en el capítulo de diseño y que se encuentran en el anexo 2 de este trabajo.

5.2 Programación del Sistema

La petición de servicios de procesamiento por computadora se puede codificar o expresar en un lenguaje natural, tal como el inglés. Hoy en día existe un conjunto de las llamadas técnicas de cuarta generación que han cambiado la idea del termino "Lenguaje de programación". Más que codificar, los que desarrollan algún tipo de sistema de gestión pueden hoy en día escribir los resultados deseados, en lugar del procedimiento deseado. Así se genera automáticamente el código fuente en un lenguaje de programación convencional.

Cuando se considera como un paso del proceso de ingeniería del software, la codificación es una consecuencia general del diseño. Sin embargo, las características del lenguaje de programación y el estilo de programación pueden afectar profundamente a la calidad y al mantenimiento del software.

5.2.1 El Proceso de traducción

El proceso de codificación traduce una representación del software dada por un diseño detallado a una realización en un lenguaje de programación. El

proceso de traducción continúa cuando un compilador acepta el código fuente como entrada y produce como salida un código objeto dependiente de la máquina. Más tarde la salida del compilador es traducida a código máquina, es decir las instrucciones reales que dirigen la lógica cableada o microprogramada de la unidad central de proceso.

El proceso inicial de traducción, del diseño detallado al lenguaje de programación, es un punto fundamental dentro del contexto de la ingeniería del software. En el proceso de traducción puede aparecer "ruido" de muchas formas distintas. La interpretación equivocada de las especificaciones del diseño o las restricciones de un lenguaje de programación pueden conducir a un código fuente muy liado que resulte difícil de probar y mantener. Más sutilmente, las características de un lenguaje de programación pueden influir en la forma de pensar, propagando diseños de software y estructuras de datos innecesariamente limitados.

Las características del lenguaje tienen un impacto directo sobre la calidad y la eficiencia de la traducción.

Desde un punto de vista ingenieril las características de los lenguajes de programación se centran en las necesidades que puede tener un proyecto específico de desarrollo de software. Aunque se podrían derivar esotéricos requerimientos para el código fuente, se pueden establecer un conjunto general de características de ingeniería: 1) facilidad de traducción del diseño al código; 2) eficiencia del compilador; 3) portabilidad del código fuente; 4) disponibilidad de herramientas de desarrollo y 5) facilidad de mantenimiento.

El paso de codificación comienza tras haber definido, revisado y modificado en caso necesario el diseño. En teoría, la generación de código fuente a partir de las especificaciones del diseño detallado deberá ser algo directo.

El grado de facilidad de la traducción del diseño al código proporciona una indicación de como se aproxima un lenguaje de programación a la representación del diseño.

Por otro lado los rápidos avances en velocidad del procesador y densidad de memoria ha comenzado a disminuir la necesidad de "código super eficiente", aunque muchas aplicaciones requieren programas rápidos y "ajustados". Las críticas actuales a los compiladores de lenguajes de alto orden van dirigidas a la incapacidad de producir código ejecutable rápido y ajustado.

La portabilidad del código fuente es una característica de los lenguajes de programación se puede interpretar de la siguiente forma: El código fuente

puede ser transportado de un procesador a otro y de un compilador a otro sin ninguna o muy pocas modificaciones.

La disponibilidad de herramientas de desarrollo puede acortar el tiempo requerido para la generación del código fuente y puede mejorar la calidad del código.

La facilidad de mantenimiento del código fuente es críticamente importante para cualquier esfuerzo no trivial de desarrollo de software. El mantenimiento no se puede llevar a cabo hasta que no se entienda el software. La documentación del diseño proporciona un fundamento para la facilidad de comprensión, ya que el código fuente final debe ser leído y modificado de acuerdo con los cambios en el diseño.

5.2.2 Elección de un lenguaje

La elección de un lenguaje de programación para un proyecto específico debe tener en cuenta las características anteriormente vistas. Sin embargo, el problema asociado con la elección puede desaparecer si sólo se dispone de un lenguaje o si el cliente demanda uno en particular.

Entre los criterios que se aplican durante la evaluación de los lenguajes disponibles están: 1)área de aplicación general; 2) complejidad algorítmica y computacional; 3) entorno en el que se ejecuta el software; 4) consideraciones de rendimiento 5) complejidad de las estructuras de datos, y 6) disponibilidad de un buen compilador o compilador cruzado. El área de aplicación de un proyecto es el criterio que más se aplica durante el proceso de selección del lenguaje.

En nuestro caso sólo se dispone de un lenguaje en particular y el cual como se ha mencionado anteriormente es Power Builder.

5.2.3 Clases de Lenguajes

Existen cientos de lenguajes de programación que han sido aplicados en uno u otro momento de algún esfuerzo serio de desarrollo de software. Dado que cualquier clasificación de lenguajes de programación permanecerá abierta a debate, en este trabajo se presentará el desarrollo de las cuatro generaciones de los lenguajes que en cierta forma corresponden a la evolución histórica de los lenguajes de programación. La figura 5.5 ilustra esta clasificación.

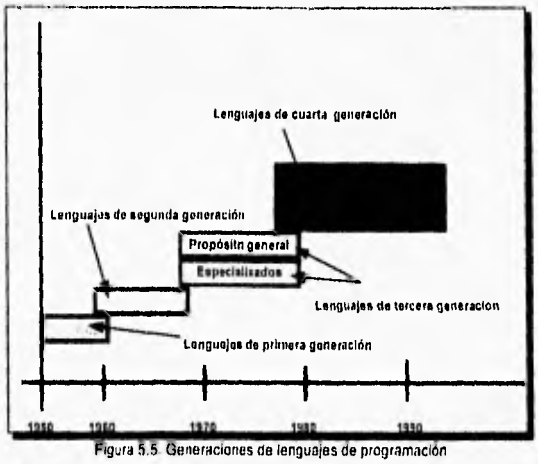


Figura 5.5 Generaciones de lenguajes de programación

5.2.3.1 Primera Generación de Lenguajes

La primera generación de lenguajes se remonta a los días en que se codificaba a nivel de máquina. El código máquina más conocido es el lenguaje ensamblador, el cual representa a la primera generación de lenguajes. Estos lenguajes dependientes de la máquina muestran el menor nivel de abstracción con el que se puede representar un programa.

Existen tantos lenguajes ensambladores como arquitecturas de sus procesadores con sus correspondientes conjuntos de instrucciones. Desde un punto de vista de la ingeniería de software, esos lenguajes sólo se deben de usar cuando un lenguaje de alto orden no satisfaga los requerimientos o no este disponible.

5.2.3.2 Segunda Generación de Lenguajes

La segunda generación de lenguajes fue desarrollada a finales de los 50's y principios de los 60's y ha servido como base de todos los lenguajes de programación modernos. La segunda generación de lenguajes está caracterizada por su amplio uso, la enorme cantidad de bibliotecas de software y la gran familiaridad y aceptación. Hay pocas discusiones sobre que FORTRAN, COBOL, ALGOL y BASIC son lenguajes de base debido a su madurez y su aceptación.

FORTRAN ha permanecido a lo largo de casi 30 años de críticas y sigue siendo el primer lenguaje de programación en el ambiente científico y de ingeniería. COBOL al igual que FORTRAN, ha alcanzado la madurez y es el

lenguaje aceptado como "standard" para aplicaciones de procedimiento de datos comerciales. Aunque el lenguaje ha sido a veces criticado por su falta de unidad, tiene excelentes posibilidades de definición de datos, es muy auto documentado y proporciona soporte para un gran rango de técnicas algorítmicas relativas al procesamiento de datos en los negocios. ALGOL es el predecesor de muchos lenguajes de tercera generación y ofrece un repertorio extremadamente rico de construcciones procedimentales y de tipificación de datos. ALGOL ha sido extensamente usado en Europa, pero ha encontrado poca ayuda en América (exceptuando el entorno académico). BASIC es un lenguaje originalmente diseñado para enseñar programación en modo de tiempo compartido. El lenguaje parecía destinado a quedarse obsoleto a principios de los 70, pero con la llegada de las computadoras personales ha vuelto a nacer.

5.2.3.3 Tercera Generación de Lenguajes

Los lenguajes de tercera generación, también denominados lenguajes de programación moderna o estructurada, están caracterizados por su potentes posibilidades procedimentales y de estructuración de datos. Los lenguajes de esta clase se pueden dividir en dos categorías, lenguajes de alto orden de propósito general y lenguajes especializados.

Lenguajes de alto orden de propósito general: Esto lenguajes tienen diferentes tipos de datos y permiten la tipificación de datos, permiten el uso de subprogramas y permiten una amplia gama de estructuras de control. El lenguaje de alto orden de propósito general mas antiguo es ALGOL, el cual ha servido como modelo para otros lenguajes de esta categoría. Sus descendientes, PL/1, Pascal, Modula-2, C y Ada, han sido adoptados como lenguajes con potencial para un gran espectro de aplicaciones (para uso en el área de ciencias e ingeniería, de productos empotrados, comerciales y/o sistemas).

Lenguajes Especializados: Los lenguajes especializados están caracterizados por su usual formulación sintáctica que ha sido especialmente diseñada para una aplicación particular. Actualmente se usan cientos de lenguajes especializados. En general estos lenguajes tienen una base de usuarios mucho menor que los lenguajes de propósito general. Entre los lenguajes que han encontrado aplicación en la comunidad de ingeniería del software están Lisp, PROLOG, Smalltalk, APL y FORTH.

5.2.3.4 Lenguajes de Cuarta Generación

A lo largo de la historia del desarrollo del software siempre se ha intentado generar programas de computadora en cada vez mayores niveles de abstracción. Los lenguajes de la primera generación trabajan a nivel de

instrucciones máquina, el menor nivel de abstracción posible. Los lenguajes de segunda y tercera generación han subido el nivel de representación de programas de computadora, pero aún hay que especificar distintos procedimientos algorítmicos perfectamente detallados. Durante la década pasada, los lenguajes de cuarta generación (L4G) han elevado aún más el nivel de abstracción.

Los lenguajes de cuarta generación contienen una sintaxis distinta para la representación del control y para la representación de las estructuras de datos. Sin embargo, un L4G representa esas estructuras en un mayor nivel de abstracción eliminando la necesidad de especificar detalles algorítmicos.

Los lenguajes de cuarta generación combinan características procedimentales y no procedimentales, o sea, el lenguaje permite al usuario especificar condiciones con sus correspondientes acciones (componente procedimental) mientras que pidiendo al mismo tiempo al usuario que indique el resultado deseado (componente no procedimental) para encontrar los detalles procedimentales aplicando su conocimiento del dominio específico.

Dentro de los diferentes tipos de L4G existen las tres siguientes categorías:

Lenguaje de Petición: Hasta ahora, la gran mayoría de los L4G se han desarrollado para ser usados conjuntamente con aplicaciones de bases de datos. Estos lenguajes de petición permiten al usuario manipular de forma sofisticada la información contenida en las bases de datos previamente creadas. Algunos lenguajes de petición tienen una sintaxis compleja que no es más sencilla que la de los lenguajes de tercera generación.

Generadores de Programas: Los generadores de programas presentan otra clase de L4G, aunque algo más sofisticados. Más que basarse en una base de datos predefinida, un generador de programas permite al usuario crear programas en un lenguaje de tercera generación usando notablemente menos sentencias. Estos lenguajes de programación de muy alto nivel hacen un fuerte uso de la abstracción de datos y procedimientos. Desgraciadamente para los que trabajan en el dominio de sistemas y de productos de ingeniería, la mayoría de los generadores de programas se centran exclusivamente en aplicaciones de sistemas de información de negocios y generan programas en COBOL.

Otros L4G: Aunque los lenguajes de petición y los generadores de programas son los L4G más comunes, existen otras categorías. Los lenguajes de soporte a la toma de decisiones permiten que los no programadores lleven a cabo una gran variedad de análisis (¿qué pasa si?) que van desde los simples modelos de hojas de cálculo bidimensionales hasta los sofisticados sistemas de

modelos estadísticos y de investigación operativa. Los lenguajes de prototipos se han desarrollado para asistir en la creación de prototipos facilitando la creación de interfaces para el usuario y de diálogos, además de proporcionar medios para el modelado de datos.

5.2.4 La Herramienta de Cuarta Generación Power Builder

Power Builder es un L4G del tipo de lenguajes de petición que se definió en el punto anterior. El entorno de programación queda definido por el grupo de herramientas que se muestra en la siguiente figura 5.6.

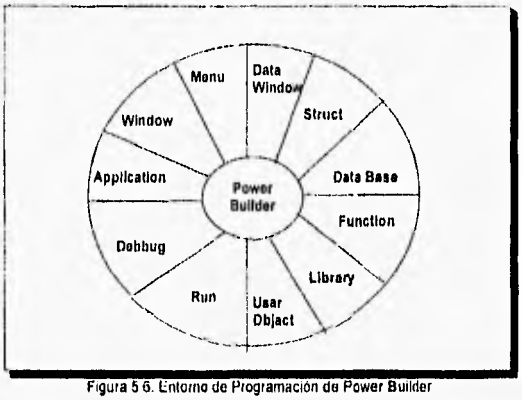


Figura 5.6. Entorno de Programación de Power Builder

Power Builder además es una herramienta orientada a la programación orientada a objetos y a eventos. Se dice que es orientada a la programación a objetos por que maneja la herencia, polimorfismo y creación de clases, aunque no es un lenguaje puro de programación orientada a objetos como lo es "C ++". Y es orientado a eventos por que se programa y responde dependiendo del evento realizado (Click, Doble Click, Close, Open) a un objeto definido de Power Builder.

Las herramientas proporcionan una total congruencia entre el desarrollo de aplicaciones y el manejo de la información de la base de datos.

Las herramientas que definen el entorno de programación son las siguientes:

Módulo Application: Este módulo sirve para crear una nueva aplicación en Power Builder o para seleccionar una aplicación ya existente y poder trabajar con ella. Además de que se define las librerías que se utilizan para cada aplicación.

Módulo de Window: En esta herramienta se definen las características de las ventanas o pantallas que manejaran los programas. Las ventanas utilizan una serie de elementos propios de Power Builder y de la interfaz windows, estos elementos son Command Buttons, Radio Buttons, Check Box, Data Windows, Picture Buttons, Data Drop List Box, Group Box, etc. Los elementos anteriores nos sirven para el diseño de las ventanas utilizadas en la aplicación, cada elemento tiene sus propios atributos y eventos los cuales se programan según la aplicación lo requiera. Existen diferentes tipos de ventanas (Main Window, Response Window, etc.), el tipo de ventana se selecciona cuando se crea la ventana, cada tipo de ventana tiene sus propios atributos y eventos que se programan de acuerdo a la aplicación. Además de que este módulo es la principal interfaz entre el usuario y Power Builder.

Módulo De Menús. Este módulo permite crear con rapidez y de una manera sencilla menúes del tipo popup los cuales pueden ser pegados a las ventanas que permitan el uso de menúes, es decir, los menúes creados son asociados a las ventanas. Un menú no tiene razón de existir si no es asociado a una ventana.

Módulo de Data Windows. Es la herramienta que realmente hace importante a Power Builder, debido a que esta herramienta es , asociada a las tablas existentes en la base de datos, existen diferentes tipos de data windows y tiene diversos usos, por ejemplo mostrar los datos existentes en una tabla, permitir dar altas, bajas, cambios, generar reportes de información, generar reportes gráficos, etc. Además para realizar las selecciones de datos, no es indispensable saber SQL, ya que el código SQL es generado automáticamente.

Módulo Struct. El módulo struct nos sirve para definir estructuras de datos, similares a las que se pueden generar en otros lenguajes como Pascal o C y las cuales se conocen como registros (record), generalmente son asociadas a un objeto o a una unidad.

Módulo Database. Al entrar a este módulo es entrar al administrador de la base de datos, aquí podemos crear nuevas bases de datos, crear tablas y vistas, modificar tablas, agregar registros a una tabla, borrar registros, manipular datos, ejecutar sentencias en SQL, etc. Este módulo hace que Power Builder sea poderoso, debido a que aparte de que el lenguaje es un L4G cuenta con su propio DBMS.

Módulo Function. Este módulo nos sirve para construir funciones que necesite el programador, o modificar funciones ya existentes, Power Builder cuenta con 300 funciones ya definidas.

Módulo Library. Esta herramienta nos sirve para crear, organizar y dar mantenimiento a las librerías ya existentes, una librería es donde se guardan los objetos creados para una aplicación.

Módulo User Object. Sirve para definir objetos del usuario y usarlos en las ventanas principalmente, los objetos que define el usuario son los que proporciona Power Builder, pero con la característica de que el usuario los modifica.

Módulo Run. Ejecuta la aplicación actual que tenga Power Builder cargada.

Módulo Debug. Este módulo es de gran utilidad cuando se tiene que revisar alguna falla en la operación de algún programa, ya que permite seguir los programas paso a paso e ir checando los valores de las variables asignadas en el programa.

5.2.5 Uso de Power Builder en el Sistema de Agente de GOSAC

El uso de un lenguaje L4G como lo es Power Builder reduce de manera significativa el tiempo de programación de los sistemas de información, siempre y cuando se haga un buen uso de este lenguaje y de explotar al máximo las herramientas y módulos que integran al lenguaje.

La creación de prototipos, reportes y actualizaciones a la base de datos son creados de una forma bastante rápida y es aquí donde más se siente e impacta en el tiempo de programación y la facilidad que se tiene para realizarlos.

Después de haber creado la base de datos, los pasos que se siguieron para la programación del sistema fueron los siguientes.

1.- Creación de una nueva aplicación (en el módulo aplication)

2.- Creación de las ventanas del sistemas
Para cada ventana:

2.1.- Creación de los menúes de las ventanas que lo requieran

2.2.- Creación de las Data Windows si es que se necesitan para el mantenimiento de la información y para los reportes que se generan

2.3.- Creación de los procesos con las sentencias del lenguaje Power Builder en los eventos requeridos de los objetos de la ventana.

2.4.- Compilar los programas

3.- Compilar y Generar el ejecutable

Las data window's nos permiten manejar la información contenida en las tablas de la base de datos de una manera fácil y permitiendo la creación rápida de la aplicación, ya que el programador se centra únicamente en la lógica de los procesos y esto es debido a que con la utilización de las data window's ya no se tiene que preocupar por la definición de las tablas, sus accesos, los posibles errores que puedan ocurrir al accesarlas, los campos que conforman las tablas, la definición de los campos, la validación de entrada de datos, las máscaras, la distribución de los campos en la pantalla o reporte, los cortes de control, la relación con otras tablas, etc.

El lenguaje Power Builder, tiene gran similitud con los lenguajes de tercera generación y en particular con "C" o Pascal, incluso algunos de sus comandos son los mismos, así que el lenguaje se hace familiar al programar con el.

Como el L4G está enfocado a la creación de sistemas de información, éste tiene un gran potencial en cuanto a los comandos requeridos para manejar información, lo cual hace que los programas sean cortos en código en comparación con otros lenguajes.

Un punto importante del que no hemos hablado, es que entre mayor grado de conocimiento y practica del lenguaje se tenga el tiempo de programación se reduce en forma considerable. Ya que como se dijo anteriormente es similar a "C" o "Pascal" pero no es igual debido a que tiene un enfoque especial y cuenta con otras instrucciones que son complejas y difíciles de entender por lo que en la mayoría de los casos lleva tiempo entender su uso y más si es la primera vez que se trabaja con una herramienta como ésta.

Otro aspecto importante de Power Builder es que genera información detallada de los elementos utilizados en el desarrollo de una aplicación y que sirven para la documentación del sistema.

Para tratar de explicar y tener una mejor visión de Power Builder a continuación presentamos un ejemplo sencillo utilizando la información que genera Power Builder.

El ejemplo que se presenta es el programa para el mantenimiento de el catálogo de compañías, en este programa se registra y actualiza la tabla compañías dando altas, bajas y cambios a la información existente en dicha tabla, además de que puede buscar y ordenar la información presentada en pantalla de distintas formas. La documentación generada es la que a continuación se presenta y explica.

En la página I del documento generado observamos la ventana (window) w_companias, así como todos los objetos que forman parte de ella. La información generada nos da las características y atributos de la ventana y de sus objetos. En esta página observamos los atributos de la ventana, las variables que utiliza la ventana y el inicio del programa (al programa se le conoce como script) para el evento de abrir la ventana (open event), el programa sirve para inicializar los objetos que interviene en la ventana y cargar los datos a la data window (dw_desplegar) para que el usuario los manipule.

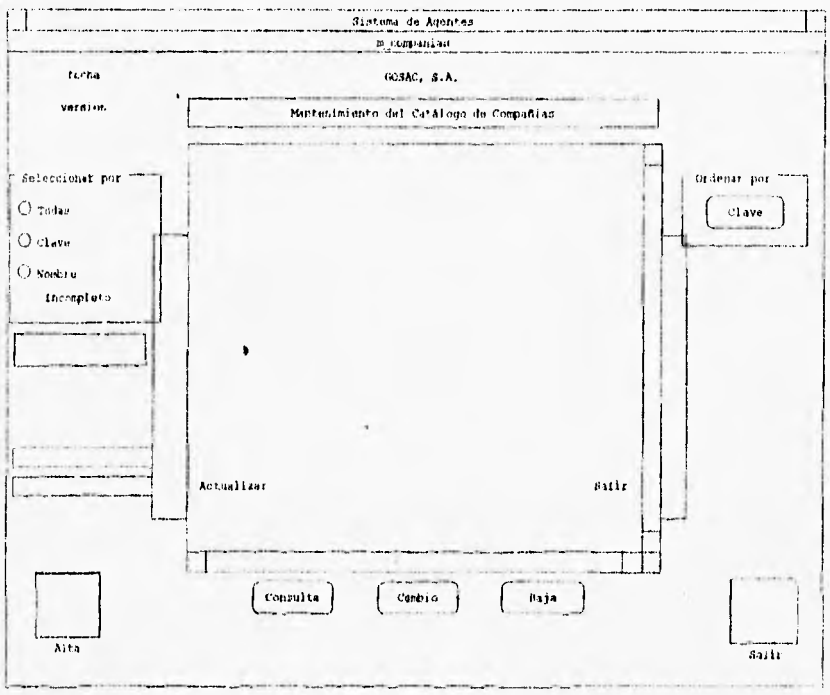
Continuando con la página II observamos la continuación del script para el evento abrir la ventana y las características y atributos para los objetos dw_2 (sirve para asociarle una data window de búsqueda), la dw_1 (sirve para asociarle una data window de búsqueda), el botón de comando (command button) cb_salir_ventana. Para este último elemento nos muestra el script asociado al evento clicked (clicked event), el script llama a la función habilita_w_companias y cambia los atributos a algunos objetos para ponerlos visibles o invisibles y ponerlos habilitados o deshabilitados.

En la página III sigue la continuación del script anterior, además encontramos las características y los atributos para el botón de comando cb_actualizar así como su script, el script sirve para actualizar los datos en la data window dw_desplegar. Después del script podemos ver las características y atributos del objeto del tipo texto estático (static text) st_1.

Al inicio de la página IV observamos las características, atributos y el script para el evento de modificación (modified event) del objeto de tipo de edición de línea (single line edit) sle_texto_buscar. El script recibe como parámetro una clave o parte de una cadena de caracteres para buscar, cargar y presentar la información en la pantalla. Continuando con los elementos que hay en la página, también encontramos los atributos del botón radial (radio button) rb_nombre y su script para el evento clicked. El script habilita y pone visibles algunos controles para introducir el texto que se quiere buscar.

En la página V, al inicio se encuentra la continuación del script anterior para que posteriormente encontremos lo atributos referentes al botón radial rb_clave, el cual al igual que el anterior sirve para habilitar controles para introducir la clave de búsqueda de información. Por último en la misma página encontramos los atributos para la data window dw_desplegar (es la que nos sirve

Library: c:\agentes\agentes.pbl
Date: 19/11/95 Time: 05:19:48



Window: w_companias

```
X = 46              Y = 17              Width = 2826        Height = 1861
Visible = true      Enabled = true      TitleBar = true
Title = "Sistema de Agentes"            MenuName = "m_companias"
ControlMenu = true  MaxBox = true       Resizable = true    WindowType = main!
WindowState = normal!                   BackColor = 12632256
```

```
Instance Variables
boolean baja=false, cambio=false, consulta=false
boolean alta=false, actualizacion=false
int seleccion=0
int clave1,compa
long renglon

End of Instance Variables


Script for: open  event
st_fecha.text=string(today(),"dd/mm/yy")
```

Library: c:\agentes\agentes.pbl
Date: 19/11/95 Time: 05:19:48

```
st_version.text=version
st_compania.text=compania
dw_tabular.settransobject(SQLCA)
dw_desplegar.settransobject(SQLCA)
dw_tabular.retrieve()

if nivel_u=1 then
   cb_cambio.enabled=false
   cb_baja.enabled=false
   pb_1.enabled=false
   m_companias.m_accin.m_cambio.enabled=false
   m_companias.m_accin.m_baja.enabled=false
   m_companias.m_accin.m_alta.enabled=false
end if
if nivel_u=2 then
   cb_baja.enabled=false
   m_companias.m_accin.m_baja.enabled=false
end if
pb_1.picturename=path + 'nuevo.bmp'
pb_1.disabledname=path + 'nuevo.bmp'


pb_2.picturename=path + 'salir.bmp'
pb_2.disabledname=path + 'salir.bmp'



End of Script
```

DataWindow: dw_2

```
X = 23              Y = 1169            Width = 494          Height = 53
TabOrder = 30       DataObject = "dw_companias_incompleto"  Border = true
LiveScroll = true   BorderStyle = stylebox!
```

DataWindow: dw_1

```
X = 23              Y = 1089            Width = 494          Height = 57
TabOrder = 20       DataObject = "dw_companias_clave"       Border = true
LiveScroll = true   BorderStyle = stylebox!
```

CommandButton: cb_salir_ventana

```
X = 1966            Y = 1149            Width = 270          Height = 89
TabOrder = 0        Text = "Salir"

Script for: clicked  event
habilita_w_companias(true)
```

```
dw_desplegar.enabled=false
dw_desplegar.visible=false
cb_salir_ventana.enabled=false
cb_salir_ventana.visible=false
cb_actualizar.enabled=false
cb_actualizar.visible=false
if alta or baja or cambio then
   rb_todos.triggerevent(clicked!)
end if
baja=false
consulta=false
cambio=false
alta=false
actualizacion=false


End of Script
```

CommandButton: cb_actualizar

```
X = 659              Y = 1149             Width = 275          Height = 93
TabOrder = 0         Text = "Actualizar"

Script for: clicked  event
int numero
if baja then
   dw_desplegar.deleterow(0)
end if
commit;
if dw_desplegar.update()=1 then
   commit;
   cb_salir_ventana.triggerevent(clicked!)
else
   rollback;
end if

End of Script
```

StaticText: st_1

```
X = 124              Y = 641              Width = 293          Height = 73
TabOrder = 0         Visible = true       Text = "Incompleto"
TextColor = 33554432                      BackColor = 12632256
Alignment = left!    FillPattern = solid!
```

SingleLineEdit: sle_texto_buscar
X = 28 Y = 777 Width = 462 Height = 89
TabOrder = 10 Limit = 0 Enabled = true Border = true
TextColor = 33554432 BackColor = 12632256
TextCase = upper! BorderStyle = stylelowered!

Script for: modified event
```
int clave
string nombre

consulta=false
cambio=false
alta=false
dw_desplegar.visible=false
cb_salir_ventana.visible=false
cb_actualizar.visible=false

choose case seleccion
case 0
   clave=integer(sle_texto_buscar.text)
   dw_tabular.dataobject="dw_companias_clave"
   dw_tabular.settransobject(SQLCA)
   dw_tabular.retrieve(clave)
case 1
   nombre=sle_texto_buscar.text
   dw_tabular.dataobject="dw_companias_incompleto"
   dw_tabular.settransobject(SQLCA)
   dw_tabular.retrieve(nombre)
end choose

End of Script
```

RadioButton: rb_nombre
X = 42 Y = 569 Width = 311 Height = 73
TabOrder = 0 Visible = true Enabled = true Text = "Nombre"
Automatic = true TextColor = 33554432
BackColor = 12632256 BorderStyle = stylelowered!

Script for: clicked event
```
m_companias.m_seleccin.m_nombreincompleto.check()
m_companias.m_seleccin.m_Clave.checked=false
m_companias.m_seleccin.m_todos.checked=false
rb_nombre.checked=true
sle_texto_buscar.visible=true
```

```
sle_texto_buscar.text=""
seleccion=1
sle_texto_buscar.setfocus()

End of Script
```

RadioButton: rb_clave

```
X = 42              Y = 481             Width = 257         Height = 73
TabOrder = 0        Visible = true      Enabled = true      Text = "Clave"
Automatic = true    TextColor = 33554432
BackColor = 12632256                    BorderStyle = stylelowered!

Script for: clicked  event
m_companias.m_seleccin.m_clave.check()
m_companias.m_seleccin.m_todos.checked=false
m_companias.m_seleccin.m_nombreincompleto.checked=false
rb_clave.checked=true
sle_texto_buscar.visible=true
sle_texto_buscar.text=""
seleccion=0
sle_texto_buscar.setfocus()

End of Script
```

DataWindow: dw_desplegar

```
X = 513             Y = 505             Width = 1847        Height = 781
TabOrder = 0        DataObject = "dd_companias"             LiveScroll = true
BorderStyle = stylebox!

Script for: dberror  event
if error_datos(dw_desplegar.dberrorcode())=1 then
   cb_salir_ventana.triggerevent(clicked!)
end if
dw_desplegar.setactioncode(1)

End of Script


Script for: itemchanged  event
string prueba, columna
int texto

texto=integer(gettext())
```

```
columna=dw_desplegar.getcolumnname()

if columna="cve_cia" then
   select nom_cia
   into :prueba
   from companias
   where cve_cia=:texto;
   if prueba<>'' then
//    messagebox("MENSAJE","La llave ya existe.")
      dw_desplegar.setactioncode(1)
   end if
end if

if columna="ext_tel" then
   cb_actualizar.setfocus()
end if

End of Script



Script for: retrieverow  event
habilita_w_companias(false)
if consulta then
   dw_desplegar.enabled=false
   dw_desplegar.visible=true
   cb_salir_ventana.enabled=true
   cb_salir_ventana.visible=true
   cb_actualizar.enabled=false
   cb_actualizar.visible=false
else
//CAMBIO  Y BAJA
   dw_desplegar.enabled=true
   dw_desplegar.visible=true
   cb_salir_ventana.enabled=true
   cb_salir_ventana.visible=true
   cb_actualizar.enabled=true
   cb_actualizar.visible=true
   //DESHABILITA EL CAMPO CLAVE
   dw_desplegar.settaborder("cve_cia",0)
 if cambio then
      dw_desplegar.setfocus()
   end if
end if

End of Script
```

StaticText: st_version
X = 42 Y = 117 Width = 458 Height = 73
TabOrder = 0 Visible = true Text = "version"
TextColor = 33554432 BackColor = 12632256
Alignment = center! FillPattern = solid!

StaticText: st_fecha
X = 42 Y = 29 Width = 458 Height = 73
TabOrder = 0 Visible = true Text = "fecha"
TextColor = 33554432 BackColor = 12632256
Alignment = center! FillPattern = solid!

StaticText: st_10
X = 2510 Y = 1625 Width = 247 Height = 61
TabOrder = 0 Visible = true Text = "Salir"
TextColor = 33554432 BackColor = 12632256
Alignment = center! FillPattern = solid!

PictureButton: pb_2
X = 2510 Y = 1449 Width = 229 Height = 177
TabOrder = 0 Visible = true Enabled = true Cancel = true
HTextAlign = center! VTextAlign = bottom!

Script for: clicked event
close(parent)

End of Script

StaticText: st_9
X = 129 Y = 1613 Width = 206 Height = 53
TabOrder = 0 Visible = true Text = "Alta "
TextColor = 33554432 BackColor = 12632256
Alignment = center! FillPattern = solid!

PictureButton: pb_1
X = 106 Y = 1433 Width = 229 Height = 177
TabOrder = 0 Visible = true Enabled = true
HTextAlign = center! VTextAlign = bottom!

Script for: clicked event
alta=true

```
habilita_w_companias(false)
//DESMARCA EL RENGLON SELECCIONADO
dw_tabular.setredraw(false)
dw_tabular.selectrow(0,false)
dw_tabular.setredraw(true)
//HABILITA LA DW DE DESPLEGAR Y LOS BOTONES: SALIR Y ACTUALIZAR
dw_desplegar.reset()
dw_desplegar.enabled=true
dw_desplegar.visible=true
dw_desplegar.reset()
cb_actualizar.visible=true
cb_actualizar.enabled=true
cb_salir_ventana.enabled=true
cb_salir_ventana.visible=true
//HABILITA EL CAMPO CLAVE
dw_desplegar.settaborder("cve_cia",1)
//INSERTA EL REGISTRO NUEVO
dw_desplegar.insertrow(0)
dw_desplegar.setfocus()


End of Script
```

RadioButton: rb_todos

```
X = 42              Y = 393             Width = 257         Height = 73
TabOrder = 0        Visible = true      Enabled = true      Text = "Todas"
Automatic = true    Checked = true      TextColor = 33554432
BackColor = 12632256                    BorderStyle = stylelowered!
```

```
Script for: clicked  event
m_companias.m_seleccin.m_todos.check()
m_companias.m_seleccin.m_clave.checked=false
m_companias.m_seleccin.m_nombreincompleto.checked=false
rb_todos.checked=true
dw_tabular.dataobject="dw_companias"
dw_tabular.settransobject(SQLCA)
dw_tabular.retrieve()
sle_texto_buscar.visible=false

End of Script
```

StaticText: st_7
```
X = 636             Y = 129             Width = 1623        Height = 81
TabOrder = 0        Visible = true
Text = "Mantenimiento del Catálogo de Compañias"
TextColor = 16777215                    BackColor = 8421376 Border = true
BorderColor = 0     BorderStyle = styleshadowbox!           Alignment = center!
FillPattern = solid!
```

StaticText: st_compania
```
X = 622             Y = 33              Width = 1646        Height = 73
TabOrder = 0        Visible = true      Text = "GOSAC, S.A."
TextColor = 33554432                    BackColor = 12632256
Alignment = center! FillPattern = solid!
```

CommandButton: cb_presentacion_clave
```
X = 2423            Y = 397             Width = 266         Height = 89
TabOrder = 0        Visible = true      Enabled = true      Text = "Clave"
Default = true

Script for: clicked  event
dw_tabular.setsort("1A,2A")
dw_tabular.sort()

End of Script
```

CommandButton: cb_baja
```
X = 1729            Y = 1457            Width = 279         Height = 89
TabOrder = 0        Visible = true      Enabled = true      Text = "Baja"

Script for: clicked  event
string serie,gerencia,prueba
double num_agt
if not actualizacion then
   return
end if
baja=true

//DESMARCA EL RENGLON SELECCIONADO
dw_tabular.setredraw(false)
dw_tabular.selectrow(0,false)
dw_tabular.setredraw(true)
select cve_tipo_afian
```

```
   into :prueba
   from ciatipfza
   where cve_cia=:compa;

select cve_agt_afian
   into :num_agt
   from ciaagt
   where cve_cia=:compa;

select nom_cia_ger
   into :gerencia
   from cias_gcias
   where cve_cia=:compa;

select serfol
   into :serie
   from cias_serfol
   where cve_cia=:compa;

if serie <> '' or gerencia <> '' or prueba <> '' or num_agt <> 0 then
   Messagebox("Mensaje","La Compañia no se puede dar de baja por que &
   existen datos con ella en otros registros")
else
   //HABILITA LA DW DE DESPLEGAR Y LOS BOTONES: SALIR Y ACTUALIZAR
   if dw_desplegar.retrieve(compa)<= 0 then
messagebox("ERROR","El registro ha sido actualizado por otro usuario",stop
sign!,ok!)
   end if
end if


End of Script
```

CommandButton: cb_cambio

```
X = 1299          Y = 1457          Width = 279       Height = 89
TabOrder = 0      Visible = true    Enabled = true    Text = "Cambio"

Script for: clicked  event
if not actualizacion then
   return
end if
cambio=true

//DESMARCA EL RENGLON SELECCIONADO
dw_tabular.setredraw(false)
dw_tabular.selectrow(0,false)
```

```
dw_tabular.setredraw(true)
//HABILITA LA DW DE DESPLEGAR Y LOS BOTONES: SALIR Y ACTUALIZAR
if dw_desplegar.retrieve(compa)<= 0 then
messagebox("ERROR","El registro ha sido actualizado por otro usuario",stopsig
n!,ok!)
end if


End of Script
```

CommandButton: cb_consulta

```
X = 865             Y = 1457            Width = 279         Height = 89
TabOrder = 0        Visible = true      Enabled = true      Text = "Consulta"

Script for: clicked  event
if not actualizacion then
   return
end if
consulta=true

//DESMARCA EL RENGLON SELECCIONADO
dw_tabular.setredraw(false)
dw_tabular.selectrow(0,false)
dw_tabular.setredraw(true)

// PERMITE VER LA DW PARA DESPLEGAR Y EL BOTON DE SALIR,
// PERO NO HABILITA LA DW NI EL BOTON DE ACTUALIZAR
if dw_desplegar.retrieve(compa)<= 0 then
messagebox("ERROR","El registro ha sido actualizado por otro usuario",stopsig
n!,ok!)
end if

End of Script
```

DataWindow: dw_tabular

```
X = 636             Y = 253             Width = 1637        Height = 1181
TabOrder = 0        Visible = true      Enabled = true
DataObject = "dw_companias"             HScrollBar = true   VScrollBar = true
Border = true       LiveScroll = true   BorderStyle = stylebox!

Script for: clicked  event
// OBTIENE EL RENGLON DESEADO
renglon=dw_tabular.getclickedrow()
```

Library: c:\agentes\agentes.pbl
Date: 19/11/95 Time: 05:19:48

```
//SI EL NUMERO ES VALIDO, OBTIENE LA CLAVE Y LO MARCA
if renglon>0 then
   compa=dw_tabular.getitemnumber(renglon,1)

   actualizacion=true
   dw_tabular.setredraw(false)
   dw_tabular.selectrow(0,false)
   dw_tabular.selectrow(renglon,true)
   dw_tabular.setredraw(true)
end if

// INICIALIZA LAS BANDERAS
baja=false
consulta=false
cambio=false
alta=false

//DESHABILITA LA DW PARA DESPLEGAR Y LOS BOTONES: SALIR Y ACTUALIZAR
dw_desplegar.enabled=false
dw_desplegar.visible=false
cb_salir_ventana.enabled=false
cb_salir_ventana.visible=false
cb_actualizar.enabled=false
cb_actualizar.visible=false


End of Script



Script for: doubleclicked  event
dw_tabular.triggerevent(clicked!)
cb_consulta.triggerevent(clicked!)

End of Script



GroupBox: gb_2
X = 2341            Y = 309            Width = 426          Height = 225
TabOrder = 0        Visible = true     Enabled = true
Text = "Ordenar por"                   TextColor = 33554432
BackColor = 12632256                   BorderStyle = stylelowered!


GroupBox: gb_1
X = 10              Y = 309            Width = 531          Height = 437
TabOrder = 0        Visible = true     Enabled = true
Text = "Seleccionar por"               TextColor = 0
BackColor = 12632256                   BorderStyle = stylelowered!
```

para dar altas, bajas, cambios y consultas), así como el script para el evento de detección de error (dberror event) el cual dispara al botón cb_salir_ventana. La dw_desplegar tiene varios scripts asociados como el anterior y el asociado al evento de cambio de información (itemchanged event), el cual nos sirve para validar la clave de la compañía.

La continuación del script anterior lo encontramos en la página VI, además de un nuevo script para el evento de cargar un renglón de información (retrieverow event), el script cambia los atributos de los botones, dependiendo de la operación que se quiera realizar (cambio, baja o consulta), el evento es también asociado a la data window dw_desplegar.

La página VII nos muestra los atributos para los textos estáticos st_version, st_fecha y st_10. Los atributos y el script para el botón con imagen (picture button) pb_2 (salir), es lo que sigue, el script es para el evento clicked y lo que realiza es cerrar la ventana. Los atributos del texto estático st_9 también son mostrados aquí. Por último se muestran los atributos del botón con imagen pb_1, el cual es el de alta de información, el script asociado al botón es para el evento clicked y lo que realiza es inicializar la data window dw_desplegar para que reciba la nueva información e insertarla en la base de datos.

La continuación del script anterior se encuentra en la página VII y después nos muestra los atributos para el botón radial rb_todos y el script asociado para el evento clicked, el script carga la información a la dw_tabular y habilita, deshabilita, pone visibles e invisibles algunos botones de control.

La página IX muestra los atributos para los textos estáticos st_7 y st_compania, Además presenta los atributos para el botón de comando cb_presentacion_clave y el script asociado al evento clicked. El script lo único que realiza es ordenar la información que se encuentra en la data window dw_tabular de acuerdo al primer y segundo campo. Por ultimo muestra los atributos y el script asociado al evento clicked del botón de comando cb_baja. El script verifica que la compañía que se va a dar de baja no comparta datos con alguna otra tabla para poder darla de baja.

La continuación del script se encuentra en la página X, después del script encontramos los atributos y el script asociado al botón de comando cb_cambio. El script es asociado al evento clicked y lo que realiza es cargar los datos del renglón seleccionado en la data window dw_tabular a la data window dw_desplegar para poder modificarlos.

En la página XI se muestra la continuación del script anterior y los atributos del botón de comando cb_consulta. El script asociado al evento clicked carga los datos del renglón seleccionado a la data window dw_desplegar para que sea consultada. Por último encontramos los atributos para la dw_tabular y el script

para el evento clicked. El script obtiene la información del renglón seleccionado, marca el renglón, inicializa banderas y por último habilita y deshabilita algunos botones de control.

El script anterior termina en la página XII y nos muestra el script para el evento doble clicked (double clicked) también asociado a la dw_tabular, el cual dispara el script del botón de comando cb_consulta. Para terminar observamos los atributos que tiene las cajas de agrupamiento (group box) gb_2 y gb_1.

El ejemplo anterior nos ilustra la forma de como se programa el sistema para la empresa GOSAC, quizá a primera vista resulte enredado y complicado, pero realmente es bastante sencillo. Otro aspecto importante que cabe hacerse notar es que aparentemente es mucho código, pero no es así, ya que en la documentación generada, mucha información es referente a los atributos y es lo que hace que se vea mucha información.

CAPÍTULO 6

Pruebas y Puesta en Marcha

Es mejor utilizar nuestra propia cabeza
un par de minutos que un
par de dias una computadora

Francis Crick
Biofisico y Premio Novel Británico

El hombre razonable se adapta al mundo;
el irrazonable intenta adaptar el mundo
asi mismo. Así pues, el progreso depende
del hombre irrazonable.

George Bernard Show

CAPÍTULO 6

Pruebas y Puesta en Marcha

6.1 Pruebas del Sistema

La prueba del software es un elemento crítico para la garantía de la calidad del software y además representa un último repaso a las especificaciones del diseño y la codificación.

La prueba presenta una interesante anomalía para los ingenieros de software ya que en las fases anteriores de definición y de desarrollo, lo que intenta el ingeniero es construir el software partiendo de un control abstracto y llegando a una implementación tangible. Posteriormente al llegar la etapa de prueba el ingeniero crea una serie de casos de prueba que intentan demoler y acabar con el software que ha sido construido. De hecho, la prueba es uno de los pasos de la ingeniería de software que se puede ver como destructivo en lugar de constructivo.

6.1.1 Flujo de Información de la Prueba

El flujo de información para la prueba sigue el esquema que se describe en la figura 6.1. Se proporcionan dos clases de entradas al proceso de prueba: 1) una configuración del software que incluye la especificación de requerimientos del software, la especificación de diseño y el código fuente; y 2) una configuración de prueba que incluye un plan y procedimiento de prueba, casos de prueba y resultados esperados.

Se lleva a cabo la prueba y se evalúan los resultados, o sea, se comparan los resultados de la prueba con los esperados. Cuando se descubren datos erróneos estó implica que hay un error y empieza la depuración.

A medida que se va recopilando y evaluando los resultados de la prueba, comienza a vislumbrarse una medida cualitativa de calidad y fiabilidad del software. Si se encuentran serios errores que requieren modificaciones en el diseño, la calidad y la fiabilidad del software quedan en entredicho, siendo necesarias más pruebas. Si, por otro lado, el funcionamiento del software parece correcto y los errores que se encuentren son fácilmente corregibles, se puede decir que la calidad y fiabilidad del software son aceptables.

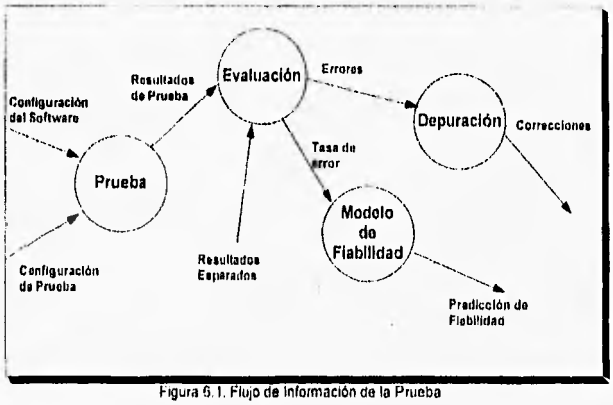


Figura 6.1. Flujo de información de la Prueba

Cualquier producto de ingeniería puede ser probado de una de dos formas: 1) conociendo la función específica para la que fue diseñado el producto, se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa; 2) conociendo el funcionamiento del producto, se pueden desarrollar pruebas que aseguren que "todas las piezas encajan"; o sea que, la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comportado de forma adecuada. La primera aproximación se denomina prueba de la caja negra y la segunda prueba de la caja blanca.

6.1.2 Técnicas de Prueba

Cuando se considera el software de computadora, la prueba de la caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. Los casos pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene. Una prueba de caja negra examina algunos aspectos del modelo fundamental del sistema sin tener mucho en cuenta la estructura lógica interna del software.

La prueba de caja blanca del software se basa en el minucioso examen de los detalles procedimentales. Se comprueban los caminos lógicos del software proponiendo casos de prueba que ejecuten casos específicos de condiciones y/o bucles. Se puede examinar el estado del programa en varios puntos para determinar si el estado real coincide con el esperado o afirmativo.

6.1.2.1 Prueba de la Caja Blanca

La prueba de la caja blanca es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar los casos de prueba. Mediante los métodos de prueba de caja blanca el ingeniero puede derivar casos de prueba que: 1) garanticen que se ejercitan por lo menos una vez todos los caminos independientes de cada módulo; 2) se ejercitan todas las decisiones lógicas en sus caras verdadera y falsa; 3) se ejecutan todos los bucles en sus límites y con sus límites operacionales, y 4) se ejercitan todas las estructuras de datos internas para asegurar su validez.

Quizá en este punto sea valido preguntarse ¿por qué gastar tanto tiempo en minusciosidades en lugar de gastarlo en el aseguramiento de haber alcanzado los requerimientos del programa?. Pero las justificaciones se dan a continuación.

- Los errores lógicos y las suposiciones incorrectas son inversamente proporcionales a la probabilidad de que se ejecute un camino del programa. Los errores tienden a reproducirse en nuestro trabajo cuando diseñamos e implementamos funciones, condiciones o controles que se encuentran fuera de lo normal.

- A menudo creemos que un camino lógico tiene pocas posibilidades de ejecutarse cuando, de hecho, se puede ejecutar de forma regular.

- Los errores tipográfico son aleatorios. Cuando se traduce un programa a código fuente de un lenguaje de programación, es muy probable que se den algunos errores de escritura. Muchos serán descubiertos por los mecanismos de comprobación de sintaxis, pero otros permanecerán indetectados hasta que comience la prueba.

Las prueba de caja blanca se centran en la estructura de control del programa. Se derivan casos de prueba que aseguren que durante la prueba se han ejecutado por lo menos una vez todas las sentencias del programa y que se ejercitan todas las condiciones lógicas.

La prueba del camino básico, es una técnica de las denominadas de caja blanca, hace uso de grafos de programa (o matrices de grafo) para derivar el conjunto de caminos linealmente independientes que aseguren la cobertura. La prueba de bucles complementa a otras técnicas de la caja blanca proporcionando un procedimiento para ejercitar bucles de distintos grados de complejidad.

6.1.2.2 Prueba de la Caja Negra

Los métodos de prueba de la caja negra se centran en los requerimientos funcionales del software. La prueba de la caja negra permite al ingeniero del software derivar conjuntos de condiciones de entrada que ejerciten completamente todos los requerimientos funcionales de un programa. La prueba de la caja negra no es una alternativa a las técnicas de prueba de la caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de la caja blanca.

La prueba de la caja negra intenta encontrar errores de las siguientes categorías: 1) funciones incorrectas o ausentes; 2) errores de interfaz, 3) errores en estructuras de datos, o en accesos a bases de datos externas; 4) errores de rendimiento; 5) errores de inicialización y determinación.

A diferencia de las pruebas de caja blanca, la prueba de caja negra tiende a ser aplicada durante posteriores fases de prueba. Ya que la prueba de la caja negra intencionadamente ignora la estructura de control, concentra su atención en el dominio de la información.

Las pruebas de caja negra son diseñadas para validar los requerimientos funcionales sin fijarse en el funcionamiento interno de un programa. Las técnicas de prueba de la caja negra se centran en el dominio de información de un programa de forma que se proporcione una cobertura completa de prueba. La partición equivalente divide el dominio de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. El análisis de valores límite, prueba la habilidad del programa para manejar datos que se encuentren en los límites aceptables. Los grafos de causa efecto se incluyen en una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones. Finalmente, la prueba de validación de datos asegura que los datos interactivos (conducidos por órdenes) sean procesados correctamente.

6.1.3 Estrategia de Prueba del Software

Una estrategia de prueba de software integra las técnicas de diseño de casos de prueba en una serie planificada que lleva a una construcción correcta del producto de software. Y lo que es más importante, una estrategia de prueba de software proporciona un plano para el programador, para la organización de control de calidad y para el cliente, un plano que describe: los pasos a llevar a cabo como parte de la prueba; cuando se deben de planificar y realizar estos pasos, y cuanto esfuerzo, tiempo y recursos serán requeridos.

Una estrategia de prueba de software debe ser suficientemente flexible para promover la creatividad y la adaptabilidad necesarias para adecuar la prueba

a todos los grandes sistemas basados en software. Al mismo tiempo, la estrategia debe ser suficientemente rígida para promover un razonable seguimiento de la planificación y la gestión a medida que progresa el proyecto.

Existen y se han propuesto varias estrategias de prueba de software en distintos libros y trabajos, pero todas estas pruebas proporcionan al ingeniero de software una plantilla para la prueba y todas tienen las siguientes características generales:

- La prueba comienza en el nivel de módulo y trabajan "hacia afuera" hacia la integración del sistema completo.

- En diferentes puntos son adecuadas a la vez distintas técnicas de pruebas.

- La prueba la lleva a cabo el que desarrolla el software y un grupo de prueba independiente.

- La prueba y la depuración son actividades diferentes, pero la depuración puede entrar en cualquier estrategia de prueba.

La estrategia de prueba que se sigue para el software producido se puede explicar al observar la figura 6.2 (Pasos de prueba del software). La prueba, en el contexto de la ingeniería de software, realmente es una serie de cuatro pasos que se llevan a cabo secuencialmente.

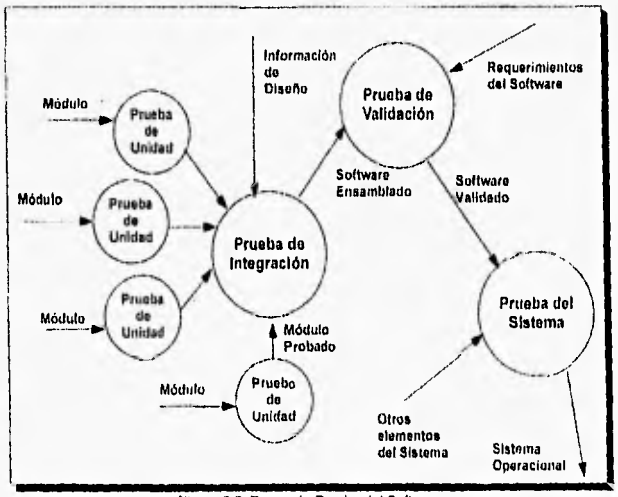


Figura 6.2 Pasos de Prueba del Software

Inicialmente la prueba se centra en cada módulo individual, asegurando que funcionen adecuadamente como una unidad. De ahí el nombre de prueba de unidad. La prueba de unidad hace un uso intensivo de las técnicas de prueba de la caja negra, ejercitando caminos específicos de la estructura de control del módulo para asegurar un alcance completo y una detección máxima de errores. A continuación se deben ensamblar o integrar los módulos para formar el paquete de software completo. La prueba de integración se dirige a todos los aspectos asociados con el doble problema de verificación y de construcción del programa. Durante la integración, las técnicas que más prevalecen son las de diseño de casos de prueba de la caja negra, aunque en algunos casos se pueden llevar a cabo unas pocas pruebas de caja blanca para asegurarse de que se cubren los principales caminos de control. Por último se deben comprobar los criterios de validación. La prueba de validación proporciona una seguridad final de que el software satisface todos los requerimientos funcionales y de rendimiento. Durante la validación se usan exclusivamente técnicas de prueba de caja negra. El fin de la prueba es la prueba del sistema y aquí se verifica que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema total.

Prueba de Unidad

La prueba de unidad centra el proceso de verificación en lo que es la menor unidad del software "El Módulo". Usando la descripción del diseño, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo. La prueba de unidad siempre está orientada a la caja blanca, y éste paso se puede llevar a cabo en paralelo para múltiples módulos.

En la prueba de unidad se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad del programa que esta siendo probada. Se examinan también las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente y conservan su integridad durante todos los pasos de ejecución del algoritmo. Se prueban además las condiciones límite para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento. Se ejercitan todos los caminos independientes de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez. Y finalmente se prueban todos los caminos de manejo de errores.

Prueba de Integración

El problema que aquí se analiza es cuando los módulos independientes interaccionan en conjunto, ya que los datos se pueden perder en una interfaz, un módulo puede tener un efecto adverso sobre otro, las

subfunciones cuando se combinan pueden no producir la función principal deseada, la imprecisión aceptada individualmente puede crecer hasta niveles inaceptables; y las estructuras de datos globales pueden presentar problemas.

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que al mismo tiempo se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es escoger los módulos probados en unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.

Para llevar a cabo la prueba se utiliza el proceso de integración incremental en donde el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y de corregir, es más probable que se puedan probar completamente las interfaces y se pueda aplicar una aproximación de prueba sistemática.

La prueba para el software producido se llevó a acabo mediante la integración descendente, la cual como su nombre implica, empieza la construcción y la prueba con los módulos atómicos (o sea, los módulos de niveles más bajos de la estructura del programa). Dado que los módulos son integrados de abajo hacia arriba, el procesamiento requerido de los módulos subordinados siempre está disponible y se elimina la necesidad de resguardos.

Pruebas de Validación

Una vez que el software se encuentra ensamblado y se han corregido los errores de interfaces, la prueba que sigue es la de validación. La validación se logra cuando el software funciona de acuerdo con las expectativas razonables del cliente, las expectativas razonables están definidas en la especificación de requerimientos.

La validación de software se consigue mediante una serie de pruebas de caja negra que demuestran la conformidad con los requerimientos. El plan de prueba traza las pruebas que se deben llevar a cabo y un procedimiento de prueba para definir los casos de prueba específicos que serán usados para demostrar la conformidad con los requerimientos. Tanto el plan como el procedimiento están diseñados para asegurar que se satisfacen todos los requerimientos funcionales y que se alcanzan todos los requerimientos de rendimiento.

Prueba del Sistema

Cuando el software es incorporado a otros elementos del sistema como son un nuevo hardware o información, se realiza una serie de pruebas de integración del sistema y de validación.

La prueba del sistema realmente está conformada por una serie de pruebas diferentes cuyo propósito primordial es la ejercitar profundamente el sistema. Aunque cada prueba tiene un propósito distinto, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas.

La pruebas mencionadas anteriormente son: 1) la prueba de recuperación, para forzar a un fallo del software de muchas formas y verificar que la recuperación se lleva a cabo apropiadamente; 2) prueba de seguridad, verifica que los mecanismos de protección incorporados al sistema lo protegerán, de hecho, de la penetración impropia; 3) prueba de resistencia, para enfrentar a los programas con situaciones anormales; 4) pruebas de rendimiento, la cual esta diseñada para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado.

6.2 Puesta en Marcha del Sistema

La puesta en marcha del sistema es el proceso de colocar el sistema de información desarrollado para ponerlo en operación normal en las actividades de la empresa. Ésto se realiza una vez que el producto de software se encuentra totalmente desarrollado y aprobado por las especificaciones con el cual fue creado, entonces lo único que falta es insertarlo en el ambiente operativo de la empresa e iniciar las operaciones.

La puesta en marcha del sistema es de suma importancia para el éxito del mismo, ya que un inicio de operaciones deficiente o mal implementado puede causar el fracaso inicial del producto desarrollado. Para el proceso de puesta en marcha se deben considerar todos los factores que influyen en la operación de un sistema como los siguientes: factor humano (capacitación para los usuarios), factor procedimental (implementación de procedimientos para administrar la operación del sistema), factor hardware (comunicaciones, cableado, etc.), factor de conversión de sistemas (migración de información entre el sistema anterior y el nuevo), etc.

Los factores que consideramos para la puesta en marcha del sistema son los factores de capacitación a los usuarios y la carga inicial de datos, dichos

factores los consideramos parte del proyecto de desarrollo e implantación del producto de software para la empresa GOSAC.

6.2.1 Capacitación a Usuarios

Dado que la capacitación es un factor para el éxito del producto de software se pone un especial énfasis en dicho factor, ya que incluso el mejor sistema puede tener éxito o fallar debido a la forma en que se opera y se utiliza. Por lo anterior la calidad con la que se de la capacitación a los usuarios que manejaran al sistema ayuda o dificulta el éxito de la puesta en marcha de un sistema de información. Las personas que se verán involucradas con el sistema deben conocer en máximo detalle las funciones que realiza el sistema, así como la forma de utilizarlo y lo que hace y deja de hacer.

La mayor parte de la capacitación de los usuarios radica en la operación del sistema mismo aunque en ocasiones la capacitación abarca el empleo del equipo y en estos casos los usuarios deberán recibir primero la capacitación respecto a la operación del equipo.

Dentro de la capacitación, se destacan las actividades de manejo de datos que reciben la máxima atención durante el proceso de capacitación de usuarios y los cuales son los siguientes, entrada de nuevos datos (es la inserción de nuevos datos a los ya existentes); edición de datos (como cambiar datos ya existentes) y la consulta de datos (búsqueda de datos almacenados previamente). La parte más importante del uso del sistema incluye el conjunto de actividades descritas anteriormente, por lo que la capacitación ocupa la mayor parte de su tiempo en estas actividades.

Los dos aspectos más importantes que se manejaran en la capacitación al usuarios son: 1) la familiaridad con el sistema de procesamiento, es decir, el equipo que se usa; 2) la capacitación en el empleo de la aplicación, es decir, en el software que recibe los datos, los procesa y genera un resultado. Es importante resaltar que si se tienen carencias en cualquier aspecto de la capacitación probablemente se terminará por hacer fracasar la utilización del sistema en la operación de la empresa.

6.2.1.1. Capacitación en el uso del sistema para GOSAC.

Dado que en la empresa GOSAC el sistema será manejado por tres tipos de usuarios, entonces la capacitación será enfocada a estos tres tipos: capturistas, administrativos y directivos.

La capacitación a los usuarios, como se mencionó anteriormente, incluye aspectos de manejo de equipo de cómputo, manejo de impresoras y aspectos del funcionamiento del producto de software. Es importante intercalar

procedimientos administrativos internos con la parte del sistema que soporta la operación. Los temas generales que forman parte de la capacitación básica para todos los usuarios de GOSAC son:

1.- Aspectos básicos del manejo de PC's e Impresoras.

2.- Ciclo operativo de la fianza en general: control de folios, emisiones de fianzas, movimientos de fianzas, primas por cobrar, primas por pagar y comisiones.

3.- Aspectos generales del sistema: antecedentes, esquema básico operativo, impacto en la operación, apoyo al control de las actividades de la empresa.

Así, los anterior puntos integran la capacitación básica que se debe aplicar a todos los usuarios de la empresa con el fin de adquirir los conocimientos mínimos para considerarse parte activa del proyecto de sistemas de la empresa GOSAC.

Una vez impartido el curso básico a los integrantes de la empresa que estarán involucrados con el sistema se agruparán de acuerdo a la actividad que tendrán entorno al sistema, es decir, de acuerdo al tipo de usuario que hayan sido catalogados. Para cada grupo de usuarios se desarrollan cursos de capacitación, de manera general, los puntos tratados en la capacitación de los 3 tipos diferentes de usuarios son:

1.- Usuario Tipo Capturista

- Manejo de PC e Impresora

- Actividades propias dentro del ciclo de operación diaria, procedimientos administrativos a seguir y relación con el sistema de información

- Módulo de primas por cobrar

- Módulo de primas por pagar

- Módulo de comisiones

- Módulo de catálogos

2.- Usuarios tipo administrativo (Agentes)

- Manejo de PC's

- Actividades propias dentro del ciclo de operación diaria, procedimientos administrativos a seguir y relación con el sistema de información

- Módulo de Folios

- Módulo de comisiones

- Obtención de información del módulo de fianzas

- Módulo de catálogos

- Módulo de solvencias

3.- Usuario tipo Directivo

- Manejo de PC's e impresoras

- Ciclo de operación diaria y puntos de control o supervisión

- Obtención de información del módulo de fianzas y de proceso de acumulación

- Módulo de Catálogos

- Módulo de Solvencias

La capacitación se imparte durante la instalación del software a la empresa GOSAC y ésta incluye aspectos tanto teóricos como prácticos usando como ejemplos tipo de información con la que se trabaja a diario, pero sin llegar ésta a ser real.

Posteriormente para el inicio de operaciones con el sistema, se vuelve a impartir la capacitación a los usuarios, pero con la diferencia de que los ejemplos ya contienen la información verdadera y real con la que se estará trabajando.

6.2.3 Conversión de información al sistema desarrollado

El proceso de conversión o migración es el cambio de la información del sistema antiguo al nuevo sistema. Si el sistema de cómputo fue creado para sustituir otro, se debe realizar un proceso de migración, el cual traduce la información del sistema anterior a los formatos que establece el nuevo sistema; si por el contrario el sistema anterior es un sistema manual de control, la conversión será entonces cargar (registrar) toda la información necesaria para el buen funcionamiento del nuevo sistema automatizado.

Dado que el sistema anterior era manual, entonces para el caso nuestro es necesario cargar la información que se requiera al sistema nuevo.

La carga de la información será entonces seleccionar la información necesaria de los folders, documentos y papelería en general de los archivos de GOSAC a los módulos del nuevo sistema para posteriormente generar reportes y compararlos con los resultados esperados.

Posteriormente existen algunos datos que hay que complementar dentro del sistema como por ejemplo la información referente a los usuarios, a la paridad de las monedas, etc.

Una vez realizado lo anterior y al iniciar operaciones con el nuevo sistema, las fallas en el proceso de carga de información y la inadecuada capacitación se verán reflejadas en los problemas que la empresa GOSAC tengan para lograr la adecuada operación del sistema.

Por último la utilización y aceptación que el sistema tenga por los usuarios del sistema en la empresa nos podrá indicar si el sistema fue un éxito o un fracaso.

CONCLUSIONES

Vive como si esperaras vivir
hasta los cien años, pero
estuvieras listo a morir mañana.

Ann Lee.

La mitad de nuestras equivocaciones
nacen de que cuando debemos pensar
sentimos, y cuando debemos sentir,
pensamos.

Proverbio británico

CONCLUSIONES

Después de haber desarrollado e implementado el sistema de información que se propuso desde el principio del presente trabajo, así como la metodología sugerida, podemos decir y evaluar los resultados que se obtuvieron, así como otros aspectos involucrados durante el desarrollo del presente trabajo.

La utilización y el manejo de un nuevo enfoque de programación, como es el caso de la programación por eventos, hace cambiar la manera y forma de pensar en cuanto se refiere a técnicas de programación, las herramientas que nos ayudan en la programación estructurada como son los diagramas de flujo, resultan en la mayoría de los casos inaplicables en la programación por eventos, mientras que algunas técnicas y herramientas de la programación orientada a objetos si son aplicables.

Para enfrentar los problemas en cuanto a la nueva forma de programar, es necesario cambiar la manera tradicional de pensar y además se necesita conocer el enfoque propuesto en la programación orientada a objetos, así como conocer perfectamente los eventos que incluye cada objeto que se utilice en la aplicación, ya que en la programación por eventos, se programa cada evento necesario o requerido de un objeto, y cada objeto puede tener uno o más eventos asociados, es por ello, que se tiene que conocer y tener bien definido lo que realizaran los eventos de cada objeto.

Como se mencionó anteriormente los conocimientos básicos de lo que es la programación orientada a objetos es de gran ayuda, ya que realizando una analogía con la programación orientada a eventos, se puede llegar a entender más fácil la naturaleza de este tipo de programación.

En mi opinión particular, al trabajar con la programación orientada a objetos y a eventos se tiene una forma de concebir e implementar las aplicaciones de una forma más natural y más sencillo que con la programación tradicional.

Otro punto resaltable es el impacto causado mediante el uso de una interfaz gráfica, que da como resultado un ambiente amigable, una forma sencilla

de intuir las operaciones de la aplicación, facilitar el manejo de la navegación dentro de la aplicación, etc. La interfaz usada en la aplicación es compatible a las que utiliza Windows y ésta nos permite entre otras cosas tener la posibilidad de usar múltiples aplicaciones (ventanas) y tenerlas disponibles al mismo tiempo, la aplicación desarrollada funciona bajo este esquema, es decir, se pueden tener diversos módulos del sistema disponibles a la vez para comodidad del usuario.

Los íconos e imágenes utilizadas en la aplicación deben ir enfocadas a la intuitividad, es decir, al poner un ícono no sólo se pone para que se vea bonito el sistema, el ícono se pone para que ayude al usuario a intuir que es lo que realiza o hace el módulo o parte de la aplicación en la que se encuentra incluido el ícono.

Otra ventaja que podemos apuntar es que si el usuario había trabajado con Windows o con aplicaciones que corrieran bajo este esquema, el manejo del nuevo sistema es relativamente sencillo, ya que aparte de usar botones con íconos para la ejecución de operaciones, también se cuenta con menús para que el usuario escoja cualquiera de las dos formas para ejecutar las operaciones en la aplicación y como se dijo anteriormente las ventanas principales de la aplicación pueden minimizarse y maximizarse para poder utilizar otro módulo que se requiera del sistema o cualquier otra aplicación que se desee y que este dentro de Windows.

Pasando a otro punto, la reducción en el tiempo al usar el sistema es considerable, ya que la información requerida se encuentra totalmente disponible y centralizada dentro del sistema desarrollado y no se tiene que estar buscando la información en folders y cajones en la oficina como anteriormente se realizaba, ahora sólo basta con revisar los catálogos dentro del sistema para tener la información requerida. Otro aspecto que también contribuye a la reducción de tiempo es el cálculo automático de comisiones, i.v.a., primas, etc.; que el sistema realiza y es principalmente en el módulo de fianzas donde el impacto de la automatización de cálculos se hace sentir y así facilitar al usuario el manejo de dichos cálculos que anteriormente resultaban ser engorrosos y complicados.

El manejo de la información de las fianzas y sus movimientos resultan fáciles y sencillos debido al análisis que se tuvo, ya que la asociación de fianzas y movimientos se realizan de forma natural dentro del sistema y no se tienen que hacer o involucrar información que sea redundante e innecesaria.

Los reportes de información que genera el sistema son de gran importancia para la empresa, los agentes y las compañías afianzadoras. Primeramente por que los reportes de producción nos indica que también o mal se encuentra la producción de la empresa, también nos permite visualizar a que compañía afianzadora es a la que más productos se le venden, y por último nos sirve como parámetro para ver el desempeño realizado por los agentes de GOSAC. No sólo los reportes anteriores son importantes, sino también los de inventario de fianzas, ya que nos permiten ver cuales son las fianzas que siguen y continúan en vigor y que además pueden generar algún movimiento. Por último los reportes de vencimiento de vigencia de fianza son de suma importancia para GOSAC y para los agentes ya que ellos permiten avisar con tiempo al cliente del estado en el que se encuentra la fianza, y así poder realizar un nuevo movimiento.

El último tipo de reportes que se incluyó en el sistema y que causó el mayor impacto fueron los reportes de productos, estos reportes son gráficas que muestran los ingresos generados y el número de movimientos efectuados en cada mes del año de un movimiento específico. El impacto generado fue por el hecho de presentar la información en forma de gráfica, en este caso fueron gráficas de barras y de pie.

El presentar la información de esta manera permite tener una mejor visualización de que es lo que está pasando con los productos (movimientos) que maneja la compañía y así poder realizar un análisis más rápido y sencillo de los productos.

El uso de un lenguaje de cuarta generación nos permite y facilita la creación de prototipos de manera rápida, está repercute en el sentido de que el tiempo que se tarda el análisis y diseño es recuperado en la etapa de desarrollo y construcción del sistema.

La principal ventaja de la situación anterior es en realizar con más tiempo disponible un muy buen análisis y un muy buen diseño para no tener problemas posteriores y por último implementar el sistema de forma rápida y sencilla con las ventajas otorgadas por el L4G.

Para el desarrollo es importante el conocimiento y tiempo que los desarrolladores tengan con el lenguaje de programación, ya que entre más se conozca el lenguaje de programación, la explotación del mismo sobre la aplicación será mejor y más eficiente. Si por el contrario no se tiene un buen

conocimiento del lenguaje de programación, el desarrollador estará limitado y no podrá hacer un buen uso de los recursos que ofrece el lenguaje de programación.

El uso de un L4G simplifica bastante la tarea de codificación, pero el buen uso del L4G lleva tiempo y no es tan fácil adaptarse a la herramienta debido a la complejidad de algunas instrucciones que sólo se llegan a comprender con el tiempo.

Una disyuntiva que surgió en la etapa de construcción del sistema fue el de utilizar la herencia para crear las ventanas o recurrir a la técnica de copear las ventanas. La herramienta Power Builder nos proporciona la opción de herencia ya que fue construido basado en la programación orientada a objetos. La dificultad que se presenta al utilizar la herencia es que lo que se hereda no se puede suprimir ni modificar y aunque las ventanas son muy similares, no son iguales. Es por ello que se prefirió utilizar la técnica de copeo, es decir, se crea la ventana más compleja y que más detalle tenga, para posteriormente copearla o salvarla con otro nombre y modificarla según las necesidades de la aplicación.

La determinación de no utilizar la herencia se tomo por la razón anteriormente dicha, pero en otras palabras, es más fácil destruir y eliminar cosas que construir e implementarlas.

En el desarrollo de sistemas existen además de los aspectos anteriores, otros factores que intervienen y no se consideran muchas veces, como es el ingenio y la astucia para atacar y resolver los problemas que se presentan en la programación del sistema, existen ocasiones en que el problema tiene más de una solución, pero el escoger la mejor muchas veces depende de la implementación que haya realizado el programador, ya que tal vez las soluciones existentes nos arrojen el mismo resultado pero quizá una sea más rápida que otra, o una utilice demasiados recursos, por lo que la mejor solución será la que se enfoque a las necesidades del sistema y estó es fundamental para que el desempeño del sistema sea eficiente.

La combinación de las metodologías usadas para el desarrollo del sistema fue de gran importancia ya que la combinación de los modelos tradicionales del software con la utilización de herramientas de cuarta generación nos permitió generar el software que se necesitaba para la empresa GOSAC.

Al rescatar las principales ventajas que nos ofrece la metodología de generación de prototipos y combinarla con la metodología T4G y también rescatar sus principales ventajas, nos permitió elaborar un producto con mayores ventajas, que si hubiéramos seguido forzosamente una metodología individual.

El conocimiento que se tenga del funcionamiento del negocio es importante, ya que entre más conocimiento uno tenga de éste la implementación, el análisis y el diseño serán realizadas de manera concisa y evita el hacer cambios o modificaciones, debido a una mala interpretación por parte nuestra. Anteriormente el tiempo que se tenía no era suficiente para llegar a conocer bien el funcionamiento del negocio, pero ahora con la rapidez que se tiene al implementar el software, uno puede gastar más tiempo en el conocimiento del negocio y preparar un mejor análisis y diseño.

En general el éxito del sistema se debe a muchos factores pero principalmente si el análisis y diseño son de excelente calidad, éstos sientan las bases para que el desarrollo también lo sea y además teniendo las facilidades que proporciona una interfaz gráfica esto hace que el producto realizado sea de gran calidad y de gran beneficio para la empresa que lo utiliza, y en un futuro se pueda considerar el tratar de ampliar los horizontes y límites del sistema.

Para terminar podemos decir que los objetivos que se fijaron en un principio y durante el desarrollo del sistema se cumplieron y se realizaron satisfactoriamente aplicando los pasos y metodología que se explican en el presente trabajo

APÉNDICE 1

Diccionario de Datos
del Sistema

CAMPO	DEFINICIÓN	DESCRIPCIÓN
año_acum	entero(4)	Año de acumulación
au_comision	numérico(13,2)	Comisión generada por los aumentos durante un mes
au_monto	numérico(13,2)	Monto generado por los aumentos durante un mes
au_numero	numérico(8)	Número de aumentos realizadas en un mes
au_prima neta	numérico(13,2)	Prima generada por los aumentos durante un mes
auditado	caracter(1)	Balance auditado (S/N)
bis_fza	entero(1)	Bis de la fianza
bonificación	numérico(13,2)	Bonificación del movimiento
ca_comision	numérico(13,2)	Comisión generada por las cancelaciones durante un mes
ca_numero	numérico(8)	Número de cancelaciones realizadas en un mes
cap_cont	numérico(13,2)	Capital contable del cliente
cobertura	numérico(6)	Número de cobertura de la fianza
cod_ope	caracter(1)	Código de Operación
cod_postal_cli	caracter(5)	Código postal del cliente
coment1	caracter(40)	Comentarios realizados por el analizador
coment2	caracter(40)	Comentarios realizados por el analizador
cred_cnsf	caracter(10)	Credencial de la comisión nacional de seguros y fianzas
cve_agt	entero(2)	Clave del agente de GOSAC
cve_agt_afian	entero(4)	Clave del agente en la compañía afianzadora en la que está registrado
cve_cia	entero(2)	Clave de la compañía afianzadora
cve_edo_poliza	caracter(1)	Marca del estado en la que se encuentra la póiiza del folio
cve_edo_recibo	caracter(1)	Marca del estado en la que se encuentra el recibo del folio
cve_gcia	entero(4)	Clave de la gerencia de la compañía afianzadora
cve_giro	caracter(2)	Clave del giro del cliente
cve_grupo	caracter(2)	Grupo al que pertenece al cliente

CAMPO	DEFINICIÓN	DESCRIPCIÓN
cve_moneda	entero(2)	Clave del tipo de moneda
cve_oper	caracter(2)	Clave de la operación (movimiento) que se le realiza a una fianza
cve_tipo	caracter(2)	Clave del tipo de fianza que maneja GOSAC
cve_tipo_afian	entero(4)	Clave del tipo de la fianza por compañía
cve_usuarios	caracter(10)	Clave del usuario del sistema
derechos	numérico(13,2)	Importe de los derechos
domicilio_cli	caracter(50)	Domicilio del cliente
edo_fza	caracter(1)	Estado de la fianza (V/C) vigente o cancelada
ex_comision	numérico(13,2)	Comisión generada por las expediciones durante un mes
ex_monto	numérico(13,2)	Monto generado por las expediciones durante un mes
ex_numero	numérico(8)	Número de expediciones realizadas en un mes
ex_prima neta	numérico(13,2)	Prima generada por las expediciones durante un mes
ext_contacto	caracter(5)	Extensión telefónica del contacto
ext_tel_cli	caracter(5)	Extensión telefónica del cliente
fec_analisis	fecha	Fecha del análisis realizado al cliente
fec_anul_mov	fecha	Fecha de anulación del movimiento
fec_edo_poliza	fecha	Fecha del último cambio de estado en la póliza del folio
fec_edo_recibo	fecha	Fecha del último cambio de estado en el recibo del folio
fec_emis_ult_mov	fecha	Fecha de emisión de último movimiento
fec_emision	fecha	Fecha de emisión de la fianza
fec_fin_vig	fecha	Fecha de fin de vigencia de la fianza
fec_fin_vig_mov	fecha	Fecha de fin de vigencia del movimiento
fec_ini_ult_mov	fecha	Fecha de inicio de vigencia del último movimiento
fec_ini_vig	fecha	Fecha de inicio de vigencia de la fianza
fec_ini_vig_mov	fecha	Fecha de inicio de vigencia del movimiento

CAMPO	DEFINICIÓN	DESCRIPCIÓN
fec_ref_cred	fecha	Fecha de referencia de la credencial
fec_reg_com	fecha	Fecha de registro de la comisión
fec_reg_mov	fecha	fecha de registro del movimiento
fec_registro	fecha	Fecha de registro de la fianza
fecha_cobro_abo	fecha	Fecha en la que se realizó el abono
gastos	numérico(13,2)	Importe de los gastos
gastos_cia	numérico(8,2)	Importe de los gastos de la compañía
Imp_abono	numérico(13,2)	Importe del abono
imp_comision	numérico(13,2)	Importe de la comisión
imp_maquila	numérico(8,2)	Importe de los gastos de maquila (papelería)
imp_maquila	numérico(13,2)	Importe de la maquila
inmuebles	numérico(13,2)	Valor de los inmuebles con los que cuenta el cliente
iva_gcia	num(4)	Porcentaje del I.V A. que cobra la gerencia
iva_mov	numérico(13,2)	I.V.A. del movimiento
linea_gtia	numérico(13,2)	Linea de garantia permitida por GOSAC
marca_cobrada	caracter(1)	Marca si ya se cobró (S/N)
marca_cont	caracter(1)	Marca de contabilidad (S/N)
marca_disp	caracter(1)	Marca si está disponible (S/N)
marca_pagada	caracter(1)	Marca si ya se pagó (S/N)
mes_acum	entero(2)	Número de mes de acumulación
monto_fza	numérico(13,2)	Monto de la fianza
monto_mov	numérico(13,2)	Monto del movimiento
nivel_usuario	entero(1)	Nivel asignado para el uso del sistema
nom_agt	caracter(40)	Nombre del agente de GOSAC
nom_benef	caracter(40)	Nombre del beneficiario

CAMPO	DEFINICIÓN	DESCRIPCIÓN
nom_cia	caracter(40)	Nombre de la compañía afianzadora
nom_contacto	caracter(40)	Nombre del contacto de la compañía
nom_grupo	caracter(40)	Descripción del grupo
nom_moneda	caracter(40)	Nombre de la moneda
nom_oper	caracter(15)	Nombre de la operación
nom_tipo	caracter(40)	Descripción del tipo de fianza
nom_usuario	caracter(40)	Nombre del usuario del sistema
num_autorizacion	numérico(8)	Número de autorización
num_folio	numérico(8)	Número de folio de las formas de operación por compañía
num_fza	numérico(8)	Número de la fianza
num_liquidación	caracter(10)	Número con el que se liquida la comisión
num_poliza	numérico(8)	Número de póliza del folio
num_recibo	numérico(8)	Número de recibo del folio
origen_fol	caracter(1)	Forma en la que se originó el folio (N/E), normal o externo
otras_gtias	numérico(13,2)	Valor de otras garantías que pueda aportar
paridad	numérico(8,2)	Paridad de la moneda con respecto al nuevo peso mexicano
password	caracter(8)	Password de acceso al sistema del usuario
por_com	num(4)	Porcentaje de la comisión del tipo de fianza
por_com_ca	entero(4)	Porcentaje que cobra el agente por el movimiento de cancelación
por_tar	num(4)	Porcentaje de la tarifa del tipo de fianza
pri_fut_mov	numérico(13,2)	Prima de los futuros del movimiento
pri_mov	numérico(13.2)	Prima del movimiento
prima_fza	numérico(13.2)	Prima de la fianza
prima_total	numérico(13.2)	Prima total calculado
re_comision	numérico(13.2)	Comisión generada por las renovaciones durante un mes

CAMPO	DEFINICIÓN	DESCRIPCIÓN
re_monto	numérico(13,2)	Monto generado por las renovaciones durante un mes
re_numero	numérico(8)	Número de renovaciones realizadas en un mes
resp_acum	numérico(13,2)	Responsabilidades acumuladas que tiene el cliente con GOSAC
rfc_benef	caracter(13)	Registro federal de causante que sirve como clave del beneficiario
rfc_cliente	caracter(13)	Registro federal de causante que sirve como clave del cliente
serfol	caracter(2)	Series de las formas de operación por compañía
serfol_p	caracter(2)	Serie de folio de la póliza
serfol_r	caracter(2)	Serie de folio del recibo
suma_cobro	numérico(13,2)	Suma del cobro de prima del movimiento
suma_pago	numérico(13,2)	Suma del pago de la prima del movimiento
tel_cli	caracter(10)	Teléfono del cliente
tel_contacto	caracter(10)	Teléfono del contacto
tipo_forma	caracter(1)	Tipo de la forma de los folios (P,R,A), póliza, recibo o ambas
verif_fol	caracter(1)	Marca de verificación de folio (S/N)
verif_p	caracter(1)	Marca de verificación de la poliza
verif_r	caracter(1)	Marca de verificación del recibo

APÉNDICE 2

Tablas del Sistema

Tabla: Relación de Fianzas en 3FN
cve_cia
cve_gcia
cve_agt_afian
num_fza
bis_fza
cobertura
cve_agt
cve_tipo
cve_tipo_afian
rfc_benef
rfc_cliente
fec_emision
fec_registro
fec_ini_vig
fec_fin_vig
cve_moneda
referencia
monto_fza
prima_fza
fec_emis_ult_mov
fec_ini_ult_mov
edo_fza

Tabla: Relación de Compañías en 3FN
cve_cia
nom_cia
gastos
imp_maquila
contacto
telefono_cli
ext_tel_cli

Tabla: Relación de Agentes en 3FN
cve_agt nom_agt cred_cnsf fec_ref

Tabla: Relación de Compañías-Gerencias en 3FN
cve_cia **cve_gcia** nom_cia_ger iva_gcia

Tabla: Relación de Compañías-Agentes en 3FN
cve_cia **cve_agt** cve_agt_afian por_com_ca

Tabla: Relación de Tipos de Fianza en 3FN
cve_tipo nom_tipo

Tabla: Relación de Compañías-Tipos de Fianza en 3FN
cve_cia
cve_tipo
cve_tipo_afian
por_tar
por_com

Tabla: Relación de Monedas en 3FN
cve_moneda
nom_moneda
paridad
fec_paridad

Tabla: Relación de Beneficiarios en 3FN
rfc_benef
nom_benef

Tabla: Relación de Clientes en 3FN
rfc_cliente
nom_cliente
cve_giro
domicilio_cli
cod_postal_cli
telefono_cli
ext_tel_cli
contacto
cve_grupo

Tabla: Relación de Solvencias en 3FN
rfc_cliente
cap_cont
auditado
inmuebles
otras_gtias
linea_gtia
resp_acum
fec_analisis
coment1
coment2

Tabla: Relación de Grupos en 3FN
cve_grupo
nom_grupo
gastos
imp_maquila
contacto
telefono_cli
ext_tel_cli

Tabla: Relación de Folios en 3FN
cve_cia
serfol
num_folio
verif
cve_edo_recibo
cve_edo_poliza
fec_edo_recibo
fec_edo_poliza
fec_registro
origen

Tabla: Relación de Acumulado Principal en 3FN
cve_cia
cve_agt
cve_tipo
mes
anio
ex_numero
ex_monto
ex_prima_neta
ex_comision
re_numero
re_monto
re_prima_neta
re_comision
au_numero
au_monto
au_prima_neta
au_comision
ca_numero
ca_comision
ca_comision

Tabla: Relación de Usuarios en 3FN
cve_usuario
password
nom_usuario
nivel_usuario

Tabla: Relación de Compañías-Series de Folio en 3FN
cve_cia
serfol
tipo_forma

Tabla: Relación de Movimientos en 3FN
cve_cia
cve_gcia
cve_agt_afian
num_fza
bis_fza
cobertura
num_recibo
verif_r
cve_oper
cve_tipo
serfol
num_poliza
verif_p
serfol_p
cod_ope
monto_mov
pri_mov
pri_fut_mov
derechos
gastos
iva
bonificacion
prima_total
fec_emis_mov
fec_ini_vig_mov
fec_fin_vig_mov
rfc_cliente
fec_reg_mov
marca_cont
fec_anul_mov
num_autorizacion
cve_agt
cve_tipo_afian
suma_cobro
suma_pago
marca_cobrada
marca_pagada
cve_grupo

Tabla: Relación de Comisiones en 3FN
cve_cia
cve_gcia
cve_agt_afian
num_fza
bis_fza
cobertura
num_recibo
verif_r
cve_oper
marca_disp
imp_comision
imp_maquila
fec_reg_com
fec_pago_com
num_liquidacion
cve_tipo
cve_agt

Tabla: Relación de Cobros en 3FN
cve_cia
cve_gcia
cve_agt_cia
num_fza
bis_fza
cobertura
num_recibo
verif_r
cve_oper
importe_abono
fecha

BIBLIOGRAFÍA

El Chiste no es llegar hasta arriba
sino quedarse ahí toda la vida.

Alejandro Lora

Bibliografía

* Edward Yourdon
Análisis Estructurado Moderno I/E
Prentice Hall Hispanoamericana, S.A. 1993 México

* C. J. Date
Introducción a los sistemas de bases de datos
Addison Wesley Iberoamericana, S.A. 1986 Wilmington, Delaware, E.U.A.

* Charette, Robert N.
Software Engineering enviroments concepts and technology
Prentice Hall 1ra edc. U.S.A. 1988

* Teague, Lavette C., Podgeon, Christopher W.
Structured Analysis Methods for computer information system
McMillan Publishing, 2da edc, U.S.A. 1985

* Database and Client/Server Solutions DBMS
Vol.7, No 5.
Mayo 1994

* Fairley, Richard.
Ingeniería de Software
McGraw Hill, 1ra edc. México, 1988.

* Sanders, Donald H.
Informática: Presente y Futuro
McGraw Hill, 3ra edc. México 1988.

* Lomb, David Alex.
Software Engineering: Planning for change.
Prentice Hall, 1ra edc,. U.S.A 1988.

* Adams, David R., Wagner, Gerald E.
Computer Information System: An Introduction
South Western Publishing,. 2da edc,. U.S.A. 1986.

* Tremblay, Jean Paul, De Dourek John M, Bunt Richard B.
An Introduction to Computer Sciene: An algorithmic approach. Pascal Edition
Mcgraw Hill, U.S.A. 1989

* Pressman, Roger.
Ingeniería de Software, Un enfoque práctico.
Prentice Hall, 3ra edc. España 1977

* Kendall, Kenneth E. Kendall, Julie E.
Análisis y Diseño de sistemas.
Prentice Hall, 1ra edc, México 1991

* Soluciones Avanzadas
Año 1, Número 6
Diciembre de 1993