UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

DESARROLLO DE UN SISTEMA DE ADJUDICACION DE BIENES COMO UNA SOLUCION CLIENTE-SERVIDOR

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N

JOSE OSCAR MIRANDA AGUILAR
JOSE RENE COTA BARAJAS
JOSE ANGEL REYES LEYVA
ALBERTO LAMADRID MARTINEZ

Director de Tesis: ING. LAURA SANDOVAL MONTAÑO

Noviembre 1997

Gracias a Dios por darme la actitud, paciencia y tenacidad para superarme en la vida, porque a pesar de las adversidades siempre me diste la valentía y fuerza suficiente para seguir siempre adelante.

Gracias Mamá, por que siempre has dado lo mejor de ti, por soñarme siempre en grande y por cultivar en mi todos los valores que un hombre de bien necesita para triunfar en la vida.

Gracias Familia, ya que sin su cariño, comprensión y cooperación, no hubiese podido realizar algo tan grande y significativo para mí. Por cierto, espero de todos lo mejor para nuestra Mamá y nuestra armonía familiar.

Gracias Marlem, por el amor, respeto, paciencia, consejos y ánimos que me diste para poder hacer realidad este logro de mi carrera profesional. Te Amo.

Gracias a todas las personas que han colaborado conmigo en todos los momentos de mi vida, ya que de TODOS he aprendido siempre algo, y lo único que quisiera pedir es que no nieguen nunca una oportunidad, porque siempre habrá una persona que valorará y aprovechará la confianza depositada en ella.

ALBERTO LAMADRID MARTÍNEZ.

Este trabajo se apoya en ciertos valores de tres pilares que han sido parte fundamental en mi vida y que sin ellos no hubiera sido posible la culminación de este sueño.

A mi madre Elda Aguilar por su valentia y determinación en los momentos mas dificiles, por enseñarme a seguir adelante pese a las adversidades y sobre todo, a tener la convicción de que todo lo que uno se propone es posible.

A mi padre José T. Miranda por el enorme esfuerzo y sacrificio que implica sacar adelante una familia, y por darme la oportunidad de aprovechar de la mejor manera posible los recursos que puso a mi alcance.

A mi tía Ana Aguilar por la fuerza interna que ha demostrado al nunca darse por vencida y por su lucha constante por aquellos ideales que se tienen.

Por último deseo agradecer a la U.N.A.M. y a la Facultad de Ingeniería por brindarme la oportunidad de formarme dentro de ese ambiente tan especial que rodea al campus universitario, pero sobre todo, por permitir que mi historia en ella pasara a formar parte de un pequeño capítulo dentro en un amplio libro lleno de tradición.

Gracias

OSCAR MIRANDA AGUILAR

A mis padres, María Guadalupe y José René.

A mis hermanos, Jorge, Carmen, Edgar, Nidia, Segio y Blas.

A mi tía, Tere.

A mi esposa, Yadira.

Porque todos han participado conmigo en el logro de esta importante meta.

Muchas gracias por su amor, guia y apoyo.

JOSÉ RENÉ COTA BARAJAS

A Dios, por damos la oportunidad de vivir y realizarnos como seres humanos.

A mis padres, por enseñarme los valores esenciales de la vida, de la familia y del espiritu.

A mis hijas, por ser el motor de mi existencia.

A mis maestros, por su conocimiento y ejemplo.

A Batli, por ser el duende que le da vida a mi imaginación.

MARCOS ALONSO AYLLON

A mis papás, María del Rosario y José Alfonso, por haberme guiado a iniciar este proyecto.

A mi esposa y mis dos hijas, Fabiola, Génesis y Ximena, por haberme motivado a concluirlo.

Muchísimas gracias.

JOSÉ ANGEL REYES LEYVA

CONTENIDO.

INTRODUCCION.

La tecnología de información en nuestros días es algo cada vez más común, que se extiende en todas las áreas: microempresas, empresas medianas, macroempresas, hogar, personal, convirtiéndose en algo de uso cotidiano. Contar con tecnología de información representa una ventaja competitiva, una necesidad de los negocios para mantenerse y crecer, y para mejorar la calidad de los servicios que demanda nuestra sociedad de hoy. Los cambios más importantes para fortalecer los negocios se están dando mediante la automatización de los procesos.

La Tecnología Cliente-Servidor es una tendencia en la actualidad que se complementa con los grandes sistemas centralizados en los Mainframes. La tecnología Cliente-Servidor se compone de un conjunto de "Clientes" en los que se encuentra la lógica aplicativa e interfaz de usuario, y el "Servidor" que satisface los requerimientos de datos realizados por los "Clientes". Esta tecnología es desarrollada mediante el uso de objetos programados por eventos, lo que resulta en más consistencia en el diseño y en la interfaz de usuario, desarrollo más ágil de aplicaciones, reusabilidad de objetos y código, etc. El Sistema "Administración de Bienes Adjudicados" se desarrollará mediante el uso de tecnología Cliente-Servidor.

El desarrollo del Sistema "Administración de Bienes Adjudicados" pretende proporcionar a la dirección de Bienes Adjudicados de la Banca, una herramienta que facilite y optimice las distintas actividades que realiza automatizando sus funciones y procesos, incrementando así la productividad de cada uno de los integrantes del área. Esto contribuye también a un mejor servicio a los clientes interesados en comprar estos Bienes.

La Adjudicación de Bienes es un proceso que se da en la Banca cuando un cliente por alguna razón deja de pagar un crédito. El simple hecho de adjudicar un Bien ya representa un problema para el Banco pues tiene que administrarlo y venderlo. En nuestro país, como consecuencia de la reciente crisis económica a finales de 1994, se incrementó fuertemente la cantidad de deudores con problemas que dejaron de pagar sus créditos. Esto representó una especie de caos en el área encargada de administrar y vender estos Bienes, ya que con las herramientas que contaban y con la necesidad de realizar algunos procesos manuales enfrentaron problemas debido al crecimiento de información.

El desarrollo de este sistema pretende proporcionar información en tiempo real para apoyar la toma de decisiones a nivel dirección, así como una operación y prestación de servicios más rápida, eficiente y segura. Esto se reflejará finalmente en menores gastos operativos, administrativos y los relacionados con los Bienes (tenencia, vigilancia, almacenaje, predial, luz, teléfono, etc.).

Las empresas tienden a una automatización gradual de los procesos de las distintas áreas que las conforman. Asignan presupuestos dependiendo de sus necesidades y prioridades, pero sus objetivos a mediano y largo plazo es la total automatización. Para lograr esto, se debe tener una visión global y una estrategia bien analizada.

Una estrategia de automatización global mal definida puede resultar en diseños deficientes de las aplicaciones, falta de integridad entre las aplicaciones de las distintas áreas, sobrecarga de información y datos conflictivos, además de altos costos de mantenimiento.

El desarrollo del Sistema "Administración de Bienes Adjudicados" tiene relación con otras áreas de la Banca, por lo que es importante considerar la información que deberá recibir y enviar a las mismas. Es importante mencionar que forma parte de la estrategia global de automatización de procesos de las distintas áreas de la Banca.

Nuestro trabajo tiene un sentido profesional muy amplio, ya que contribuimos al desarrollo de tecnología de información en nuestro país, aportando herramientas de productividad a los usuarios del sistema y por consecuencia al área que hará uso del mismo. También apoyamos la competitividad de la empresa mediante un mejor servicio a sus clientes. También apoyamos a la sociedad en general mediante la percepción de nueva tecnologia y un incremento general de su cultura informática.

En este trabajo presentamos una serie de capítulos cuya descripción general presentamos a continuación:

En el primer capítulo hablamos de la arquitectura Cliente-Servidor, sus conceptos, sus realidades. Explicamos los distintos modelos, las ventajas que representa para una empresa el desarrollo de aplicaciones con este tipo de tecnología, así como los costos que esto implica. Pretendemos que con el contenido de este capítulo quede claro a cualquier lector un panorama general de esta tecnología.

El segundo capítulo menciona algunas metodologías para el desarrollo de sistemas, el éxito en el desarrollo de un sistema depende mucho del uso de una metodología, existen distintos autores y opiniones. En este capítulo presentamos alternativas, sin embargo pensamos que un buen complemento para estas metodologías es la creatividad y criterio de los desarrolladores de software.

En el tercer capítulo planteamos la problemática y nuestra propuesta de solución. Explicamos el resultado de nuestra interacción con el cliente, entender su problemática, sus procesos actuales y sus requerimientos, en orden de, elaborar una propuesta de solución que las satisfaga.

El cuarto capítulo presenta el análisis y diseño del sistema. El análisis consiste en tener claros los requerimientos de los futuros usuarios del sistema. Desarrollo de diagramas que expliquen los procesos y flujos de información, así como la identificación de funciones y procesos. El diseño consiste en elaborar un modelo técnico de lo que será el sistema, especificando la base de datos, los procesos y funciones. Además diseña la interfaz gráfica de usuario. Esta es una de las etapas más importantes en el desarrollo de sistemas y se basa en una metodología, en nuestro caso orientada a objetos.

El quinto capítulo explica la construcción y puesta en producción del sistema. En la construcción interviene el uso de un lenguaje de programación y el ambiente de desarrollo. Las actividades relacionadas con esta etapa son la programación, pruebas unitarias e integrales. La puesta en producción del sistema involucra tanto la configuración del hardware donde funcionará el sistema, así como la instalación del software probado y aceptado por el área usuaria.

El sexto capítulo menciona los distintos tipos de mantenimiento que requieren los sistemas en producción. Una vez que un sistema se ha puesto en producción, siempre requiere de cambios de distinta naturaleza: Aumentativo, Perfectivo, Correctivo y Adaptativo. Normalmente estos cambios se requieren durante toda la vida activa de los sistemas.

En el séptimo y último capítulo expresamos nuestras conclusiones tanto del proceso de desarrollo como del mismo sistema, además de la percepción por parte de nuestro cliente para este proyecto.

1. ARQUITECTURA CLIENTE/SERVIDOR.

1.1. Fundamentos Cliente/Servidor

Conceptos

Cliente/Servidor es un modelo para construir sistemas de información partiendo de una diversidad de tecnologías que distribuyen aplicaciones, datos o servicios a través de múltiples procesadores. El término Cliente/Servidor implica la relación entre unos procesos que inician solicitudes de servicios (Clientes) y otros procesos que responden a ellas (Servidores), y que se ejecutan en el mismo procesador o en otro distinto de la red. El modelo Cliente/Servidor permite una clara separación de funciones basada en el concepto de servicio, posibilitando situar las aplicaciones en la plataforma más adecuada, dentro de un entorno de sistemas abiertos y heterogéneos, optimizando así los beneficios de la empresa.

Además, se podrán redistribuir las funciones y datos en caso de que cambie la organización o los procesos de negocio.

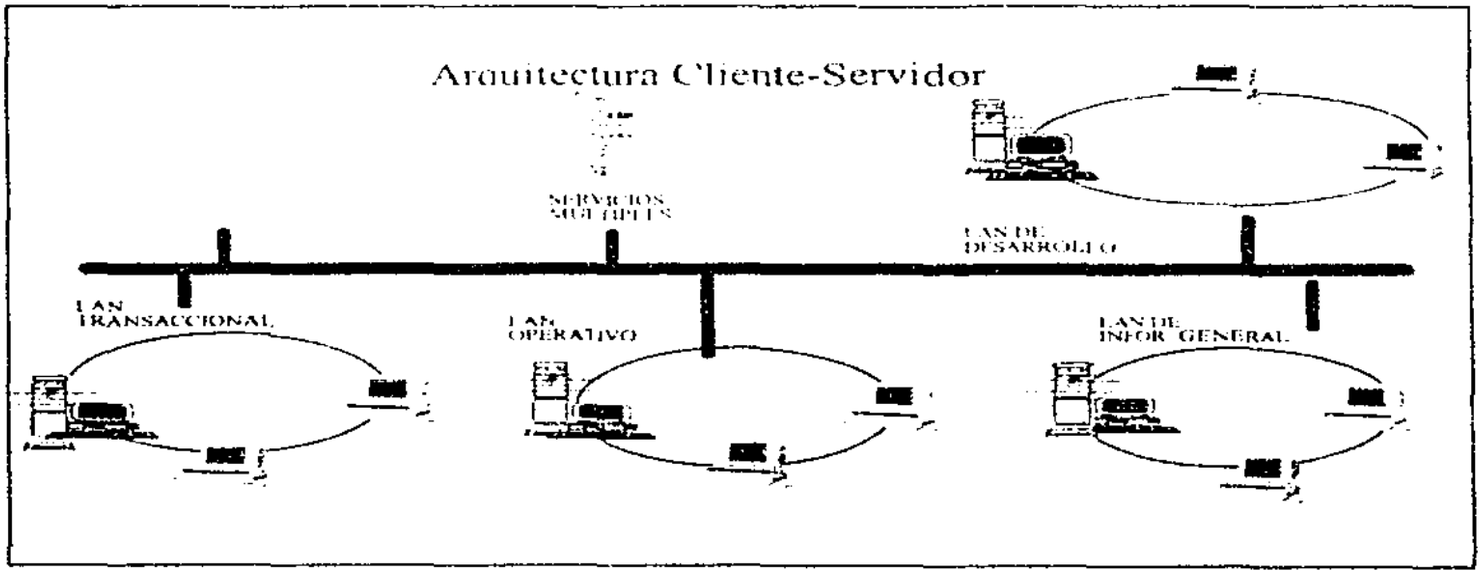


Fig. 1.1. Múltiples servidores en ambientes heterogéneos bajo una arquitectura cliente/servidor.

Objetivos

El propósito es permitir al usuario el acceso a aplicaciones y datos disponibles situado en cualquier lugar la red. A continuación se resumen los objetivos más importantes de Cliente/Servidor:

Localización Transparente: El servidor es un proceso que puede residir en la misma máquina que el proceso cliente o en otra diferente de la red. El usuario no necesita saber dónde se encuentran situados los servicios (Servidores), las aplicaciones o los datos. Del mismo modo, la aplicación cliente se codifica con independencia de la localización de los servidores. Es a través de Middleware Cliente/Servidor que dichos clientes obtienen los servicios necesarios para localizar e interoperar con los servidores.

Recursos compartidos: Un servidor puede servir a muchos clientes al mismo tiempo, controlando, además, el acceso a los recursos compartidos.

Escalabilidad de la aplicación: El objetivo es conseguir que la aplicación pueda ampliarse horizontal o verticalmente. Escalabilidad horizontal significa añadir o suprimir estaciones de trabajo/procesadores. Escalabilidad vertical significa migrar a servidores mayores o menores, o de un tipo diferente. Esto puede ser necesario por razones de capacidad, o porque las funciones y datos hayan de ser redistribuidos a procesadores distintos, si se rediseñan los procesos de negocio. Para conseguir esto, es necesario que el código de la aplicación sea portable. La elección del hardware para ejecutar la aplicación, o partes de ella, deben poder realizarse en tiempo de ejecución y no en el momento del diseño e implantación.

Interoperabilidad: Permite que los usuarios trabajen con aplicaciones y datos distribuidos en distintos procesadores de una red que pueden no tener una arquitectura similar. El Middleware ideal es independiente de las plataformas de hardware o de los sistemas operativos. El usuario debe poder combinar distintos procesos cliente y servidor, ejecutables, probablemente, en plataformas diferentes que interaccionan entre sí, ya sean del mismo o de distintos proveedores.

Evolución

Durante los últimos años, la implantación de sistemas Cliente/Servidor ha sufrido un proceso evolutivo. A continuación se describen las características de las distintas épocas :

Primera época

Desde hace algunos años está teniendo lugar de forma creciente la instalación de computadoras personales y estaciones de trabajo, al principio de forma aislada y posteriormente interconectados mediante redes de área local.

En estas redes, los usuarios comienzan a compartir datos, software, etc. Por otro lado, empresas cuya estructura informática estaba basada en grandes computadoras, conectan estación de trabajo primero en modo emulación pero también para distribuir aplicaciones entre el mainframe y la estación de trabajo en proceso cooperativo, que distribuye parte de una aplicación de forma fija. Estos desarrollos utilizan un entorno de comunicaciones consistente y homogéneo, y tienen generalmente una distribución de funciones fija entre el host y la estación de trabajo. El control de la aplicación y la mayor parte de la lógica de la aplicación se realizan en el mainframe. La interfaz de usuario está basada en caracteres o en gráficos.

Segunda época

Comienzan a proliferar las herramientas de desarrollo para aplicaciones Cliente/Servidor; su utilización, en principio casi exclusivamente GUIs (Graphic User Interface), se expande a otras más generales. Muchos proveedores de DBMS (Data Base Management System) ofrecen métodos y herramientas que abarcan todos los aspectos del desarrollo de aplicaciones. Los sistemas de grupo de trabajo evolucionan desde la ejecución de aplicaciones de productividad a sistemas que ejecutan aplicaciones críticas para el negocio. El Downsizing se hace popular; muchos proveedores son partidarios de soluciones basadas en una Red de Area Local, eliminando los sistemas basados en un mainframe. Los Sistemas Operativos de Red, como Novell Netware o Banyan Vines, representan un paso más en compartición de recursos, facilitando una completa infraestructura de servicios, muchos de ellos propios del middleware, como directorio, nominación y seguridad.

Situación Actual

Tipicamente, las estaciones de trabajo operan en un entorno de aplicaciones múltiples y el usuario controla la ejecución mediante una Interfaz Grafica (GUI) que, a menudo, está orientada a objetos. La integración de esas aplicaciones de la empresa con otras disponibles comercialmente (como por ejemplo, hojas de cálculo, procesadores de texto, correo electrónico, etc.) se consigue mediante GUIs (Graphic User Interface), compartiendo e intercambiando datos entre los programas utilizando tecnologías como DDE (Dynamic Data Exchange) y OLE (Object Linking and Embeding).Esta es la era del proceso en red, ("Network Computing"), en la cual todas las computadoras de una empresa -maiframes, minicomputadoras y computadoras personales- están conectadas.

Aplicaciones y datos están distribuidos mas allá de una Red Local.

El objetivo es tener aplicaciones operando a través de múltiples plataformas de red, con acceso transparente a funciones, datos y recursos del sistema, dondequiera que se encuentren.
Para conseguir esto, existen dos arquitecturas: (1) La arquitectura de aplicaciones de dos niveles, basada principalmente en Sistemas de Gestión de Bases de Datos, y (2) la arquitectura de tres niveles, que implica servicios de Gestión de Transacciones. Estas dos arquitecturas se describen con mayor profundidad más adelante.

Se toma conciencia de que la batalla en Sistemas Abiertos se ha desplazado desde la plataforma de hardware al Middleware, ya que cada vez surgen más productos y herramientas de desarrollo que proporcionan una infraestructura Cliente/Servidor que deben proporcionar interfaces abiertas que permitan la coexistencia heterogénea de diferentes desarrollos.

El Futuro

Algunos expertos predicen que en el futuro se pondrá cada vez mayor empeño en facilitar el uso de los sistemas y en la posibilidad de acceder a los datos independientemente de dónde se encuentre uno o dónde viaje (ubicuidad y movilidad de procesos).

Aplicaciones Cliente/Servidor

Características : Como se refleja en la siguiente figura, una aplicación Cliente/Servidor se compone de varios procesos cliente y servidor que se pueden distribuir en una red. El Middleware Cliente/Servidor conecta los procesos que componen la aplicación. Este término, ya muy

extendido, se utiliza para describir aquellas funciones que son necesarias para proporcionar la transparencia de localización.

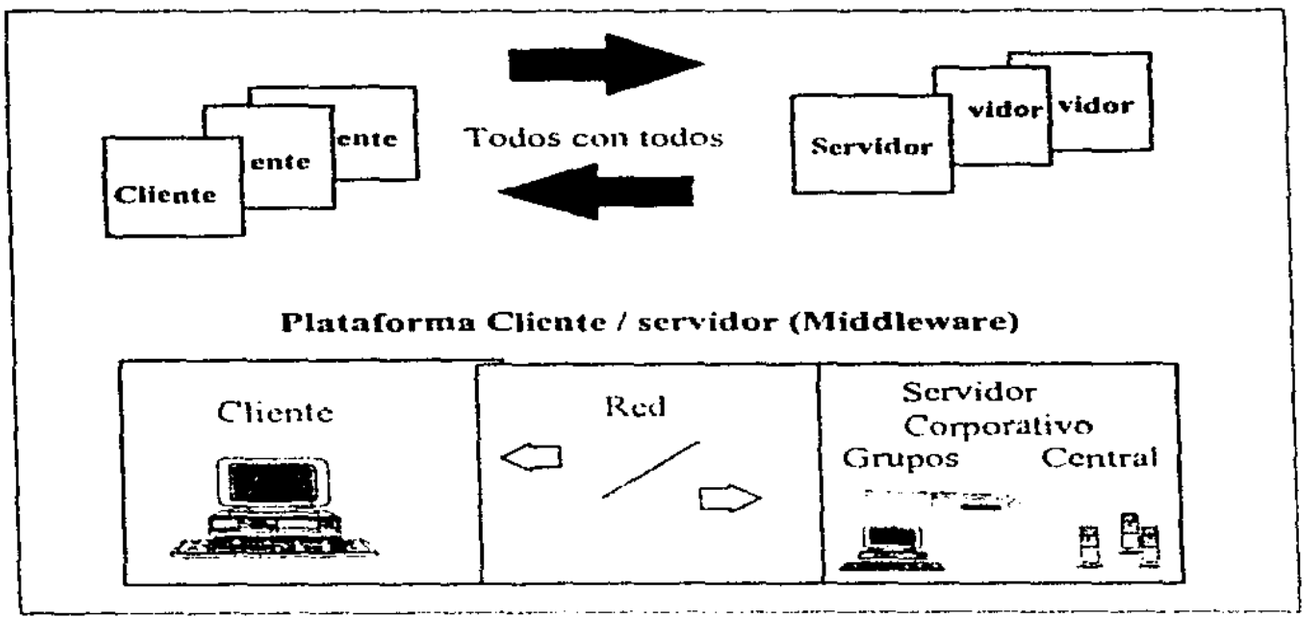


Fig. 1.2. Componentes de una aplicación cliente/servidor.

Es a través del Middleware que estos procesos se conectan e interactúan constituyendo aplicaciones. El Middleware agrupa una serie de funciones y servicios (directorios, seguridad, tiempo... etc.) no incluidos en las aplicaciones ni en el sistema operativo, que aíslan al programador de tener que ocuparse de los aspectos de bajo nivel propios de sistemas distribuidos: comunicaciones, directorios, integridad, etc.

Mientras que el Middleware define las plataformas y el nivel de transparencia de localización, las verdaderas ventajas de implantar Cliente/Servidor sólo se pueden conseguir mediante un adecuando diseño del sistema y de la aplicación, y con una acertada elección de los elementos de desarrollo (herramientas, lenguajes). El objetivo es conseguir una total flexibilidad para, en caso necesario, distribuir las distintas partes de una aplicación entre varios procesadores, sin que ésta tenga que modificarse.

Para esto, una aplicación Cliente/Servidor debe disponer de los siguientes atributos:

Separación de funciones: Para poder distribuir parte de una aplicación entre los procesadores, su lógica debe dividirse en los siguientes módulos funcionales:

- Lógica de presentación
- Lógica de negocio
- Lógica de datos.

Si se diseñan y se implantan de manera apropiada, los módulos de función pueden distribuirse y redistribuirse en distintos procesadores, y los flujos de trabajo pueden cambiarse en función de las necesidades del negocio. Además, se pueden elegir herramientas específicas y/o tecnologías para implantar los distintos tipos de funciones. La elección del Middleware Cliente/Servidor determina, a su vez, el nivel de portabilidad e interoperabilidad deseado.

Encapsulación de Servicios: Con objeto de que el acceso a dichos módulos funcionales sea conocido y accesible por clientes y servidores, las funciones deben encapsularse, es decir, rodearse de una interfaz bien definida, accesible mediante mensajes o llamadas definidas
Internamente, la función podrá cambiarse o mejorarse, incluso utilizando diferentes herramientas o lenguajes sin que los clientes tengan que ser notificados.
Un servidor, por tanto, puede ser considerado como un objeto. Este objeto, si se diseña correctamente, puede ser compartido por muchos procesos de clientes distintos y diferentes aplicaciones (reusabilidad). Los procesos servidores no están dedicados a procesadores físicos.

Portabilidad: Como se mencionaba anteriormente, las diferentes "piezas" de la aplicación pueden necesitar ser colocadas en diversos procesadores de una red. Para que esto sea posible, es necesario que dichas piezas sean desarrolladas con atributos de portabilidad que permitan la ubicación en sistemas de diferentes arquitecturas y fabricantes con mínimos o nulos cambios. La perfecta solución Cliente/Servidor es independiente del hardware o de los Sistemas Operativos, y debe permitir la combinación de diferentes plataformas.

Operación Síncrona/Asíncrona: Múltiples clientes pueden acceder a múltiples servidores. Los clientes siempre inician la petición; los servidores esperan pasivamente las peticiones de los clientes. Un cliente puede iniciar una petición en modo síncrono o asíncrono. El proceso síncrono significa que el cliente espera hasta que el servidor ha cumplido la petición, o hasta que haya expirado un plazo determinado.

En el modo asíncrono, el proceso cliente continúa en cuanto el servidor acusa recibo de la petición. Una vez terminado el proceso servidor, se notifica al cliente. El modo asíncrono es útil para procesos largos, por ejemplo la impresión de un archivo. En algunos casos especiales el proceso servidor conlleva la notificación asíncrona al proceso cliente. Ejemplo de este tipo son los servidores que soportan dispositivos de Entrada/Salida, como impresoras o interfaces gráficas de usuario.

De aplicaciones verticales a horizontales

En el pasado, las aplicaciones se construían, principalmente de forma vertical y sin ningún tipo de integración. Una aplicación vertical estaba formada por código y unos datos asociados. Todas utilizaban sus propios datos, lo que, en la mayoría de los casos, desembocaba en duplicidad de datos y falta de sincronización entre las bases de datos. Los problemas que esto originaba son bien conocidos. Desde entonces, se han hecho grandes esfuerzos por integrar aplicaciones a nivel de datos para evitar estas situaciones.

Una de las promesas más importantes del modelo Cliente/Servidor es que no sólo se puede lograr la integración de las aplicaciones a nivel de datos, sino también a nivel de funciones. Sin embargo esto sólo se conseguirá si las aplicaciones se diseñan adecuadamente. El modelo Cliente/Servidor pondrá fin a las aplicaciones monolíticas en las que los datos y el código que

utilizan están situados en un mismo lugar, ya sea un mainframe, un sistema intermedio o una computadora personal. La distribución de las funciones de la aplicación y la adopción de la orientación a objetos, tienen un importante impacto en la estructura y el diseño de esas aplicaciones.

Como se ha explicado anteriormente, una aplicación se puede descomponer en muchos procesos cliente y servidor autocontenidos. Algunos se ocupan de la interacción con el usuario a través de objetos o gráficos, y otros ejecutan funciones encapsuladas, relativas al negocio, que pueden invocarse desde diversos procesos cliente pertenecientes a aplicaciones diferentes. Para conseguir esto hay que diseñar la aplicación subdividiéndola en funciones autocontenidas que luego pueden desarrollarse como funciones encapsuladas.

La siguiente figura muestra un ejemplo del sector bancario en el que se aprecia la división lógica de una aplicación, adecuada para un entorno de proceso distribuido. Representa la emisión de una orden de compra y venta de acciones como parte de la aplicación de los agentes de cambio y bolsa del banco.

Orden: Un cliente de un banco da una orden de venta de acciones en bolsa a un determinado precio, y desea reinvertir el dinero en otras acciones.

Operación: La orden de este cliente se divide en dos operaciones: la venta y la compra de las acciones respectivas en bolsa. Existe una conexión directa entre las dos, ya que la segunda de ellas solamente se puede ejecutar si la primera se realiza con éxito.

Transacción: Cada una de las operaciones de bolsa se divide, a su vez, en transacciones individuales, como por ejemplo:

- Cargo y abono, en la cuenta bancaria del cliente.
- Abono y cargo, en la cuenta del agente de cambio
- Depósito/Disposición de las acciones desde/a la cuenta del cliente y desde/a la cuenta del agente.

Nótese que el proceso de cada una de las transacciones está relacionado, es decir, todas las interacciones de una transacción debe completarse antes de realizar actualizaciones en los datos. Además, todas las transacciones que componen la operación deben controlarse de la misma manera.

Funciones: Durante el proceso de las transacciones indicadas, seguramente se ejecutarán las siguientes funciones:

- Acceso a los datos del cliente.
- Acceso y actualización de datos de la cuenta.
- Acceso y actualización de datos de acciones.
- Acceso a datos del mercado de valores, etc.

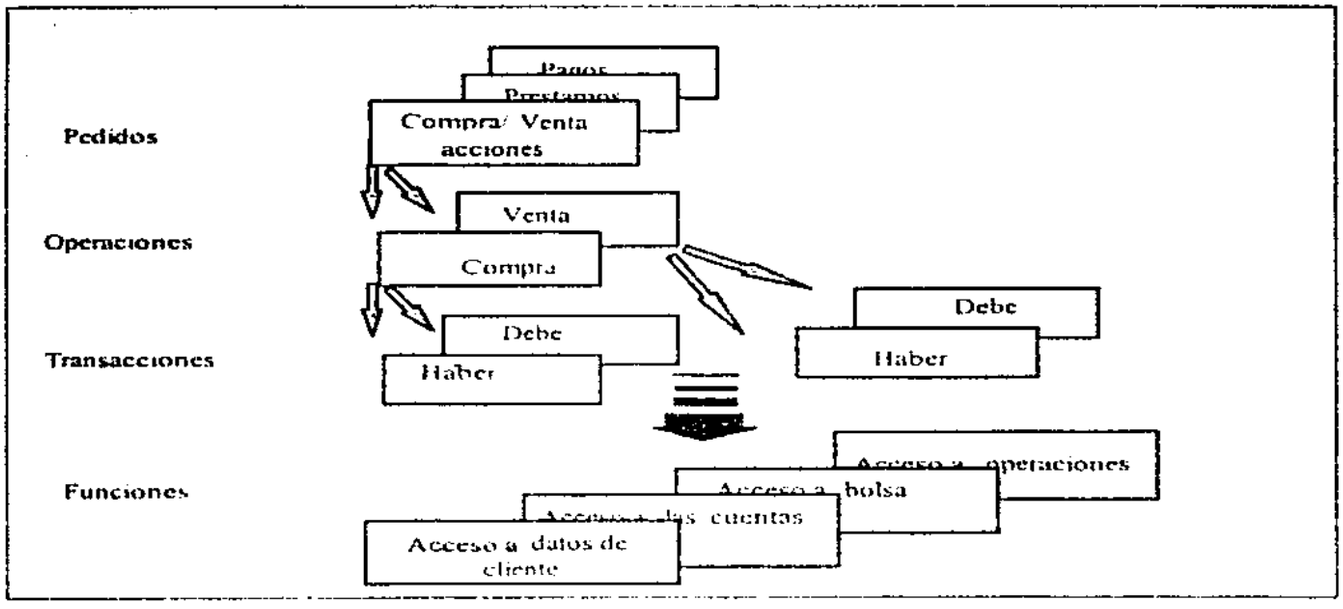


Fig. 1.3. Ejemplo de transacciones y funciones en un banco.

El resultado son unas funciones básicas que, combinadas, constituyen los procesos de negocio para realizar la aplicación. Es fácil darse cuenta que las funciones que intervienen en el proceso del pedido anterior se pueden utilizar igualmente como parte de otras operaciones, por ejemplo:

- El acceso a los datos del cliente puede utilizarse en cualquier transacción o aplicación.
- La transacción final que creó el débito o crédito de la base de datos de la cuenta del cliente, puede invocarse igualmente cuando el cliente retira dinero de un Cajero Automático, cuando se deposita dinero en el banco o cuando se deposita dinero mediante una transferencia.

Si esas funciones se desarrollan como funciones servidor encapsuladas, serán reutilizables.

Entorno de Aplicaciones Incremental

En un entorno Cliente/Servidor las aplicaciones tradicionales ya no existen como tales. Las funciones base definidas durante el proceso de diseño de la aplicación se traducirán en procesos cliente y servidor. El resultado será un Entorno de Aplicación que contiene componentes distribuibles. Los componentes distribuibles pueden ser clientes, que hacen de interfaz con el mundo exterior, o servidores que proporcionan servicios a los clientes y que pueden solicitar servicio de otros servidores, todos distribuidos a través de diversos procesadores en la red. Los servidores pueden ser funciones de negocio encapsulados desarrolladas por la empresa, pero también pueden ser aplicaciones estándar disponibles en el mercado como, por ejemplo, un sistema experto, una hoja de cálculo o un procesador de textos.
En este entorno, añadir una nueva aplicación significa:

- Añadir un nuevo proceso cliente
- El nuevo cliente llama a muchos servidores ya existentes
- Posible ligera modificación en alguno de los servidores.

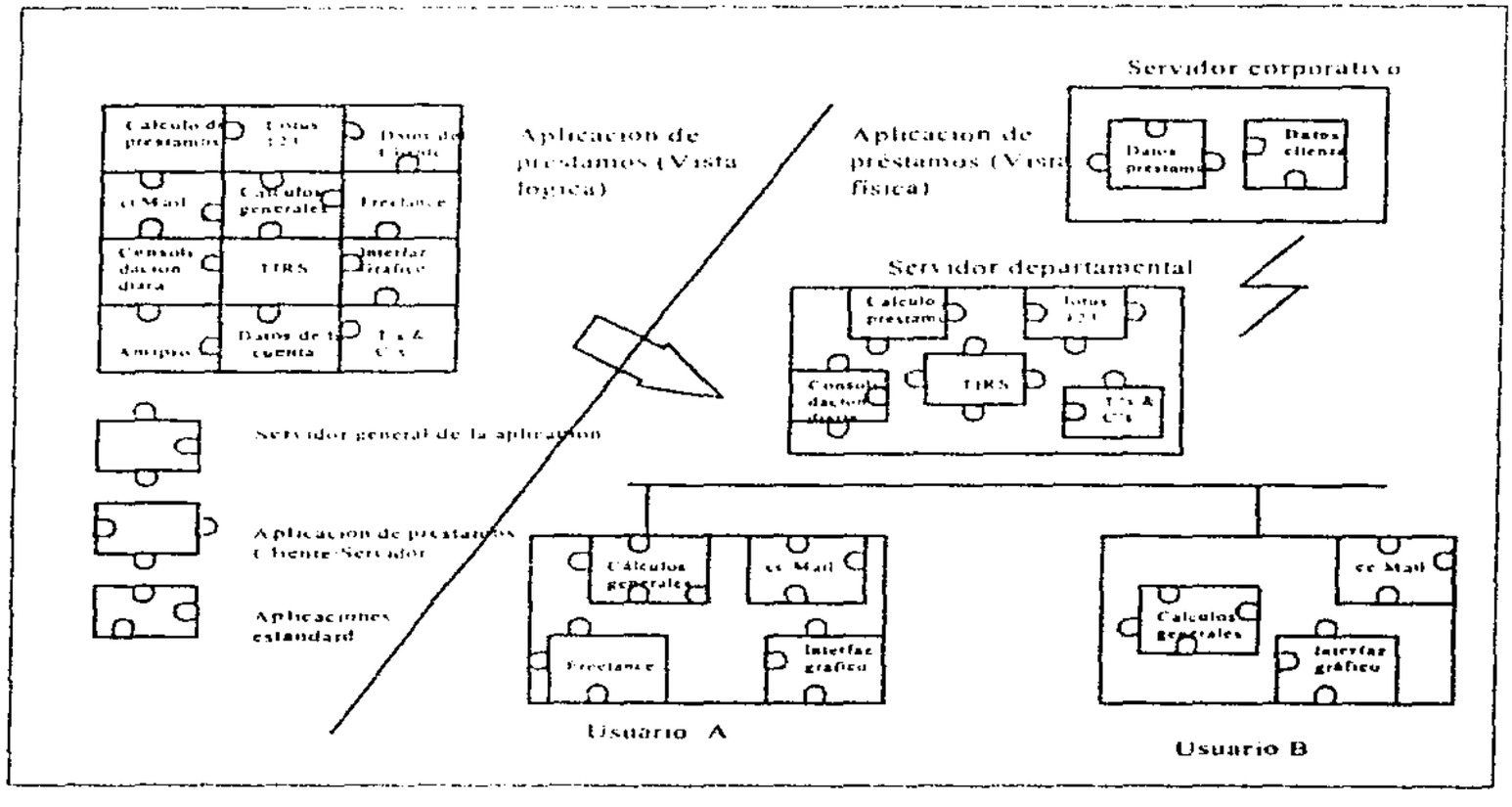
- Añadir un nuevo servidor, que también podrá ser utilizado en otros aplicaciones en el futuro.
- Es previsible que se cambien algunos de los procesos cliente existentes para que se beneficien de los nuevos servidores de aplicación.

Nota: Aunque se añaden nuevas funciones al servidor, ello no impacta en los procesos cliente existentes que utilizan ese servidor, siempre que éste continúe soportando la misma interfaz de mensajes.

Debido a estos atributos, el entorno también puede denominarse Entorno de Aplicaciones Incremental.

Aplicaciones Distribuidas Integradas

En un entorno Cliente/Servidor, las aplicaciones se componen de partes de desarrollo propio y de aplicaciones disponibles comercialmente. La integración se consigue a través de la Interfaz Gráfica de Usuario, compartiendo datos comunes a mediante comunicación directa e intercambio entre aplicaciones. Estas dos últimas posibilidades ocultan al usuario todos los aspectos relativos al sistema y proporcionan un acceso transparente y un fácil uso de las aplicaciones de negocio (Imagen de Sistemas Único). Las funciones de la aplicación se distribuyen entre los procesadores de la red.



Fig,. 1.4. Aplicaciones distribuidas integradas.

La figura anterior muestra cómo una aplicación, en este caso de créditos bancarios, se traduce en términos de componentes cliente y servidor distribuidos en una red.

La visión lógica de la parte izquierda muestra que la aplicación se compone de programas de desarrollo propio y de las siguientes aplicaciones estándar: Lotus 1-2-3, Freelance, Lotus AmiPro y TIRS (sistema basado en reglas de conocimiento). Aunque las funciones de estos programas están integradas de forma lógica en la aplicación de créditos, los mismos programas se utilizan independientemente en otras aplicaciones.

Las partes de la aplicación de créditos que son de desarrollo propio se componen de procesos cliente y servidor. En este ejemplo se muestra un parte GUI (Graphic User Interface) de Créditos que es responsable de la interacción con el usuario. También se muestran funciones de la aplicación de créditos, como, por ejemplo, Cálculo de Créditos y Datos de la Cuenta de Créditos que han sido desarrollados como servidores de la aplicación.

Otras partes de desarrollo propio son también de uso general por otras aplicaciones. Entre ellas se incluyen:

- Datos del Cliente: acceso y actualización de un archivo de información.
- Términos y Condiciones: una función que proporciona información sobre todos los términos y condiciones de las ofertas del banco.
- Cálculos Generales: una función de cálculos financieros que se utiliza en aplicaciones distintas.
- Proceso de Final de Día: contiene todas las funciones de aplicación que deben ser ejecutadas al final del día.

Los componentes que constituyen la aplicación del usuario se distribuyen de la forma siguiente:

- La Estación de Usuario Uno tiene la Función Cliente de Créditos, la función de Cálculos Generales, cc:Mail y Freelance
- La Estación de Usuario Dos tiene la función Cliente de Créditos, la función de Cálculos Generales y cc:Mail.
- El Servidor Departamental tiene TIRS, Lotus 1-2-3, un servidor de Cálculo de Créditos, el servidor de Proceso de Final de Día y el servidor de Términos y Condiciones.
- El Servidor Corporativo tiene los Datos de Cliente y el servidor de Datos Contables.

Los procesos cliente y servidor se conectan a través de Middleware. El cliente (la parte GUI) interacciona con el usuario e invoca el correspondiente servidor de aplicación. La conexión entre los componentes de la aplicación de crédito de desarrollo propio y las aplicaciones estándar se realiza mediante una interfaz DDE (Dynamic Data Exchange). Algunos productos Middleware Cliente/Servidor soportan acceso remoto DDE transparente. Esto implica que la aplicación que utiliza la interfaz DDE no necesita conocer la ubicación de la que solicita servicios.

Uno de los retos más importantes de las organizaciones de sistemas de información es materializar una infraestructura de tecnología informática que permita desarrollar este tipo de entorno de aplicaciones. Para conseguirlo, será necesario:

- Seleccionar los productos de middleware más adecuados
- Seleccionar herramientas de desarrollo de aplicaciones apropiadas
- Diseñar las aplicaciones para un entorno distribuido.

Tendencia hacia un mayor control por el usuario
En un entorno de proceso distribuido, el control de la aplicación pasa a la Estación de Trabajo Programable y al usuario. Típicamente, la estación de trabajo opera con múltiples aplicaciones y el usuario controla su ejecución manipulando iconos de la interfaz gráfica de usuario. Desde ahí se accede a los servicios locales de la red departamental o a los servicios remotos de una WAN. Los procesadores de la LAN (Local Area Network) y WAN (Wide Area Network) que forman parte del sistema se convierten en proveedores de servicios para las aplicaciones de la estación de trabajo.

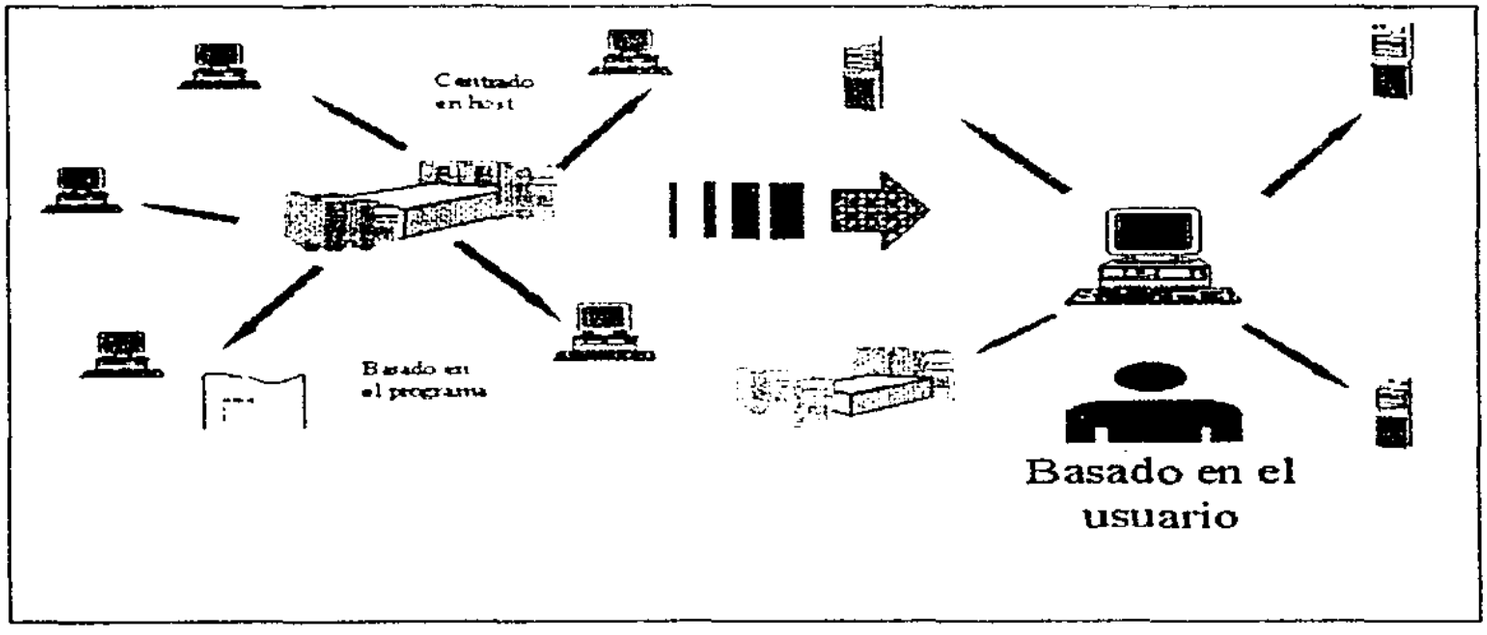


Fig. 1.5. Sistemas basados en el usuario.

Las consecuencias son las siguientes:
Dispositivos de Seguridad a nivel de la estación de trabajo: Para garantizar la seguridad de integridad de los datos, la estación de trabajo debe disponer de los dispositivos de seguridad necesarios para proceso local, para acceso a datos locales y para aplicaciones y datos de la red.
Estos dispositivos incluyen funciones para identificación y autentificación del usuario, control de acceso y, opcionalmente, para proteger los datos transmitidos por la red. Todos ellos son adicionales a los dispositivos de seguridad ya disponible, por ejemplo, de manera centralizada en el Host.

Gestión y Control del Flujo de Trabajo: En Cliente/Servidor, las funciones (servidor) se constituyen como entidades autónomas, que el usuario puede escoger de manera aleatoria. Dado que los procesos de negocio exigen una cierta secuencia en su ejecución (flujo de trabajo), es necesario que esas funciones independientes se liguen en esa misma secuencia.

Por ejemplo, el proceso de aprobación de un préstamo en un banco requiere que se sigan unos procedimientos estrictos. No se puede dar el dinero antes de reunir la información necesaria y de que la persona responsable haya autorizado la petición. Es muy probable que las autorizaciones dependan de la cantidad en cuestión. Grandes cantidades de dinero quizá deban ser autorizadas por la oficina regional o por la central, mientras que las cantidades pequeñas podrían ser autorizada por el personal encargado de la oficina local.

El código que controle la ejecución secuencial de proceso de negocio debe estar fuera de la lógica de las funciones, con el fin de evitar que cambios en los procesos de negocio, obliguen a la recodificación de esas funciones. Como se verá más adelante, este flujo de trabajo se debe integrar en lo que llama Servicios de Aplicación, dotando al sistema de la máxima flexibilidad.

Modelos de distribución Cliente/servidor

No existen reglas fijas sobre dónde dividir una aplicación. Como norma general, las funciones y los datos deben situarse lo más cerca posible del lugar donde sean necesarios para la operativa empresarial.

El objetivo de las aplicaciones Cliente/Servidor es conseguir una distribución de funciones entre los procesadores de una red departamental y/o WAN (Wide Area Network) que permita el uso óptimo de los recursos. Dependiendo de qué funciones se distribuyan, distinguiremos los tres modelos de distribución siguientes:

Presentación:

- Presentación Distribuida: que comparte la presentación entre el servidor y el cliente.
- Presentación Remota: que aísla la presentación del resto de la aplicación.

Datos:

- Datos Distribuidos: que comparte los datos entre el servidor y el cliente
- Datos Remotos: que separa el acceso a los datos del resto de la aplicación

Lógica

- Lógica Distribuida: que permite asignar las funciones de la aplicación a distintos procesadores.

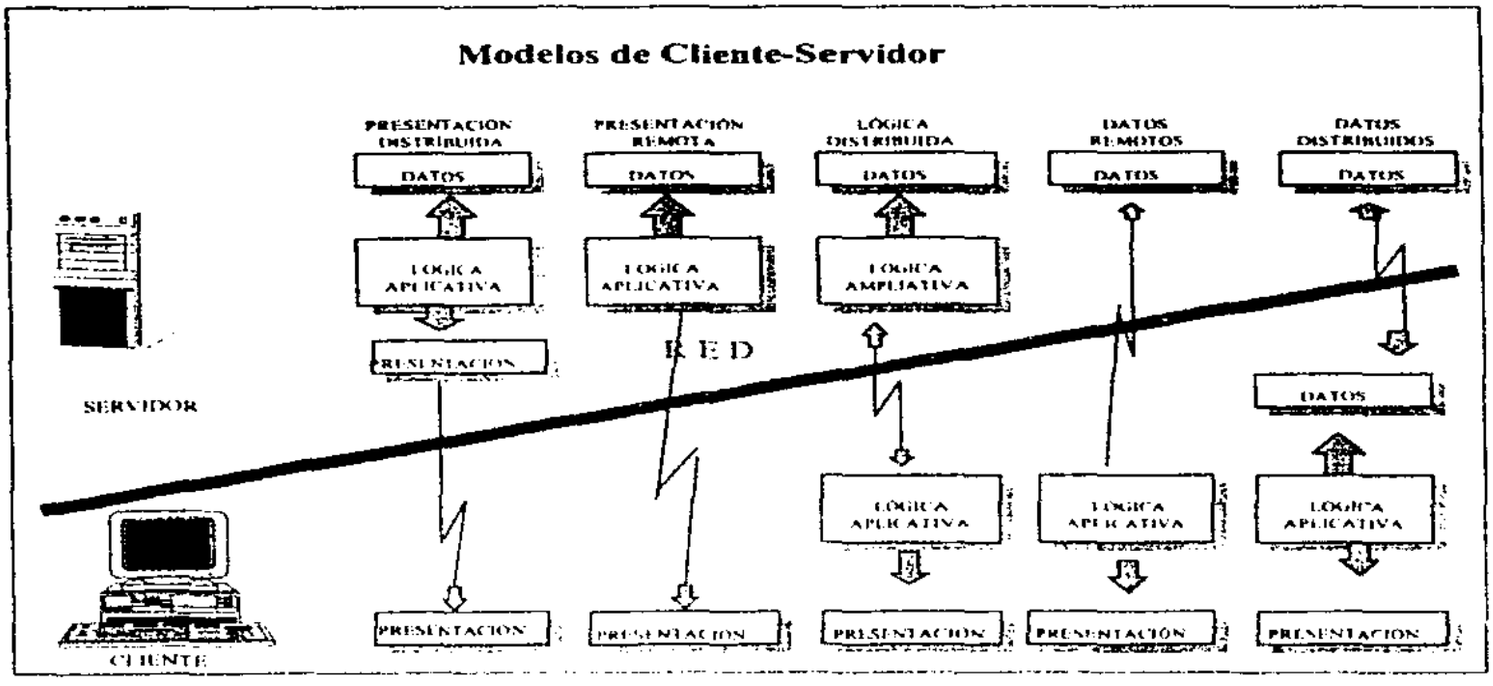


Fig. 1.6. Modelo de aplicaciones cliente/servidor

La figura anterior muestra el modelo conceptual de las posibilidades de distribución de aplicaciones. Esta es una visión simplificada y se utiliza para explicar las características básicas de los distintos modelos. Según ya hemos visto, una aplicación Cliente/Servidor se compone de un número determinado de procesos cliente y servidor. Generalmente, los procesos cliente se ejecutan en una estación de trabajo y solicitan servicios de múltiples procesos servidor, locales y remotos, utilizando cualquiera de los tres modelos. En realidad, es muy normal encontrar múltiples modelos de distribución dentro de una sola aplicación, y, entorno de un mismo sistema. Debido a las nuevas exigencias de los negocios, es posible que, con el tiempo se requieran otros modelos de distribución diferentes del inicialmente elegido. Por ello habrá que prestar particular atención a que el Middleware seleccionado, soporte todos los modelos posibles.

Presentación Remota

En este modelo la parte de la aplicación que se ocupa de la presentación de los datos está situada remotamente respecto de la lógica de negocio y los datos de la aplicación. La lógica de la presentación se desarrolla generalmente con alguna de las herramientas de desarrollo Cliente/Servidor. La Presentación Remota en su forma más simple puede consistir en una interfaz gráfica de usuario para acceder a programas de proceso de transacciones ya existentes; alternativamente, se pueden realizar, como parte de la Presentación Remota, todos los diálogos con el usuario, incluyendo la validación de datos y la iniciación de la lógica de negocio y el acceso a datos.

Otra forma de Presentación Remota, para arquitecturas del mundo UNIX, es el X-Windows System. Este sistema aisla la presentación y la gestión de ventas del resto de la

aplicación. El programador de la aplicación no tiene que conocer el lugar donde se ejecutan las funciones, es decir, las funciones X-Cliente y X-Servidor pueden ejecutarse en el mismo procesador o en procesadores diferentes. El terminal X-Windows no tiene lógica de interfaz de usuario pero proporciona servicios de presentación para OSF/Motif, X-Windows Manager, AIX-Windows Manager, etc.; la aplicación cliente tiene toda la lógica de interfaz de usuario y supervisa los sucesos.

Datos Remotos

Cuando se distribuyen los datos hacia los clientes, su acceso y manipulación quedan separados del resto de la aplicación.

Archivos distribuidos: Se utilizan principalmente para compartir los recursos de una red. Esto es la forma más sencilla de proceso distribuido que consiste en una redirección física de las solicitudes de lectura y escritura con muchos mensajes fluyendo por la red para encontrar los datos pedidos. Lo que es común a este tipo de servidores es la lógica de la aplicación, que siempre se ejecuta en el proceso cliente, por ejemplo, en una estación de trabajo de usuario. Las exigencias sobre el sistema por parte del procesador que gestiona el recurso son notablemente mayores que las del relativamente pequeño código cliente que se ejecuta en la estación de trabajo o procesador solicitante. Existe una relación asimétrica entre el cliente y el servidor.

Bases de Datos distribuidas: Permiten utilizar de forma más eficiente el procesador que controla la base de datos. El proceso cliente pasa las peticiones SQL en forma de mensajes al servidor de la base de datos. Todas las operaciones de búsqueda se ejecutan en el servidor y únicamente se devuelve el resultado a la aplicación invocante. Un posible inconveniente es que el resultado de una operación de búsqueda de la base de datos pueden ser múltiples filas de datos que, en un entorno WAN, puede tener implicaciones negativas sobre el rendimiento. Típicamente, la lógica de aplicación se ejecuta en el procesador cliente, por ejemplo, en una estación de trabajo de usuario.

También aquí, las exigencias sobre el sistema del procesador que gestiona el recurso son muy superiores a las del relativamente pequeño código cliente que se ejecuta en la estación de trabajo o procesador solicitante. También existe una relación asimétrica entre el cliente y el servidor, aunque comparado con los Archivos Distribuidos, el servidor ejecuta considerablemente más funciones.

Los servidores de bases de datos son la base fundamental de los sistemas de soporte de decisiones que requieren preguntas no planificadas e informes flexibles.

Tanto los Archivos Distribuidos como las Bases de Datos Distribuidas son funciones distribuidas transparente, cubiertas normalmente por el software del sistema, que están totalmente fuera de la lógica de la aplicación. El programador no tiene que saber que algunas de esas funciones no se ejecutan en la misma estación de trabajo o procesador.

Lógica distribuida

Este modelo proporciona la máxima flexibilidad y permite a los departamentos de desarrollo un control total sobre dónde situar las funciones en la red. Es necesario evitar que la distribución de las funciones de la aplicación quede fija por diseño. Cuando esto sucede, para volver a

redistribuir funciones y datos casi siempre son necesarios importantes modificaciones o volver a rediseñar la aplicación.

Un proceso cliente invoca a un proceso servidor que reside en el mismo procesador o en otro procesador de la red. El servidor puede realizar funciones relativas al negocio y acceder a archivos o bases de datos, devolviendo el resultado al proceso cliente que lo ha invocado. El cliente recibe el resultado sin saber si procede del mismo procesador o de otro distinto. Un servidor de aplicaciones puede, a su vez, actuar como cliente, solicitando funciones de otros servidores de la red.

Arquitecturas de Aplicaciones de Dos y Tres Niveles

Las aplicaciones Cliente/Servidor pueden clasificarse en aplicaciones de dos y de tres niveles. En esencia, es una diferenciación basada en la forma en que la aplicación se distribuye entre el cliente y el servidor. La sección siguiente describe sus características.
Téngase en cuenta que esta es una filosofía de diseño, y que los dos o los tres niveles describen estructuras lógicas, no físicas. Véase la figura anterior.

El Middleware Cliente/Servidor y/o las herramientas de desarrollo de aplicaciones utilizadas, normalmente dictan el tipo de distribución y el nivel de flexibilidad, así como los procesadores soportados.

Aplicaciones de Dos Niveles

Características: Como su nombre indica, la aplicación se divide en partes que se ejecutan en una estación cliente y en un servidor. Los servicios de presentación, la lógica de presentación y la lógica de aplicación y de acceso a datos es, normalmente, un bloque monolítico que se ejecuta en la estación cliente, mientras la base de datos está situada físicamente en el servidor.

Estos desarrollos se construyen sobre sistemas de gestión de bases de datos, también llamados a veces servidores de bases de datos. Esto es a lo que muchos se refieren cuando hablan de Cliente/Servidor. El formato del mensaje entre cliente y servidor es, invariablemente, algún tipo de SQL.

Algunos sistemas de gestión de bases de datos ofrecen Stored Procedures y Triggers (Procedimientos almacenados y Disparadores) que sirven para desarrollar la lógica de la aplicación en el servidor. Los proveedores afirman que esto es funcionalmente equivalente a una arquitectura en tres niveles, sin embargo, estos desarrollos no proporcionan una verdadera separación de la lógica de la aplicación, ya que ésta está ligada al gestor de base de datos. Por tanto, esas funciones no se pueden distribuir a otros procesadores de la red.

Utilización Típica: Sistemas de soporte de decisiones. Alternativamente, este tipo de arquitectura también es adecuada para nuevas aplicaciones departamentales. Permite que los departamentos de usuarios creen sus propias aplicaciones utilizando herramientas de interfaz de usuario.

Ventajas potenciales: Rápido desarrollo de nuevas aplicaciones en un entorno de alta productividad.

Desventajas potenciales: Falta de flexibilidad para distribuir funciones y datos, y aplicaciones poco escalables. Integridad de datos: probablemente sólo están protegidos los recursos de la base de datos, los otros recursos tienen que ser gestionados por la aplicación. Implicaciones en el rendimiento, ya que la lógica de la aplicación está físicamente ligada al gestor de bases de datos que (normalmente) sólo soporta SQL dinámico.

Aplicaciones de Tres Niveles

Características: Los funciones de la aplicación están divididas en lógica de presentación, lógica de aplicación o de negocio, y lógica de datos. Cada una de estas partes puede distribuirse entre los múltiples procesadores de la red. Probablemente, los servicios y la lógica de presentación se ejecutarán en la estación cliente, mientras que la lógica de aplicación y la lógica de datos pueden repartirse entre diferentes procesadores. Separando la lógica de aplicación puede conseguirse un nivel de flexibilidad que no es posible con las aplicaciones a dos niveles.

Como resultado, pueden existir servidores de aplicación o de transacciones que ejecuten funciones accesibles por muchos procesos cliente con distintas formas posibles de interacción con el usuario. Los servidores de aplicación se desarrollan con funciones encapsuladas, reutilizables, con un cometido e interfaz bien definidos.

El Middleware Cliente/Servidor provee la infraestructura necesaria (directorios, seguridad, nominación y servicios de tiempo, así como servicios de comunicación) para permitir la interacción entre cliente y servidor de forma transparente. Esto permite mover las funciones de aplicación de un procesador a otro sin modificaciones. La elección del lenguaje de programación o de la herramienta de desarrollo puede determinar el nivel de portabilidad.

Utilización típica: Muy aptas para aplicaciones corporativas que soportan procesos de negocio que pueden sufrir modificaciones. Muy potentes y adecuadas para aplicaciones grandes, dinámicas o complejas.

Ventajas potenciales: Máximo nivel de flexibilidad en términos de distribución (y redistribución) de funciones y datos. Las aplicaciones resultantes son escalables, flexibles y abiertas. Los servidores de aplicación resultantes son reutilizables.

Pueden seleccionarse las herramientas y lenguajes más adecuados para cada una de las partes de la aplicación (utilizando los conocimientos existentes). Puede conseguirse un cambio gradual hacia nuevas tecnologías, por ejemplo, Orientación a Objetos. Los elementos de gestión de transacciones garantizan la integridad de datos en todos los recursos utilizados por la aplicación.

Desventajas potenciales: Muy alta complejidad de desarrollo, requiere una inversión inicial en el diseño del sistema

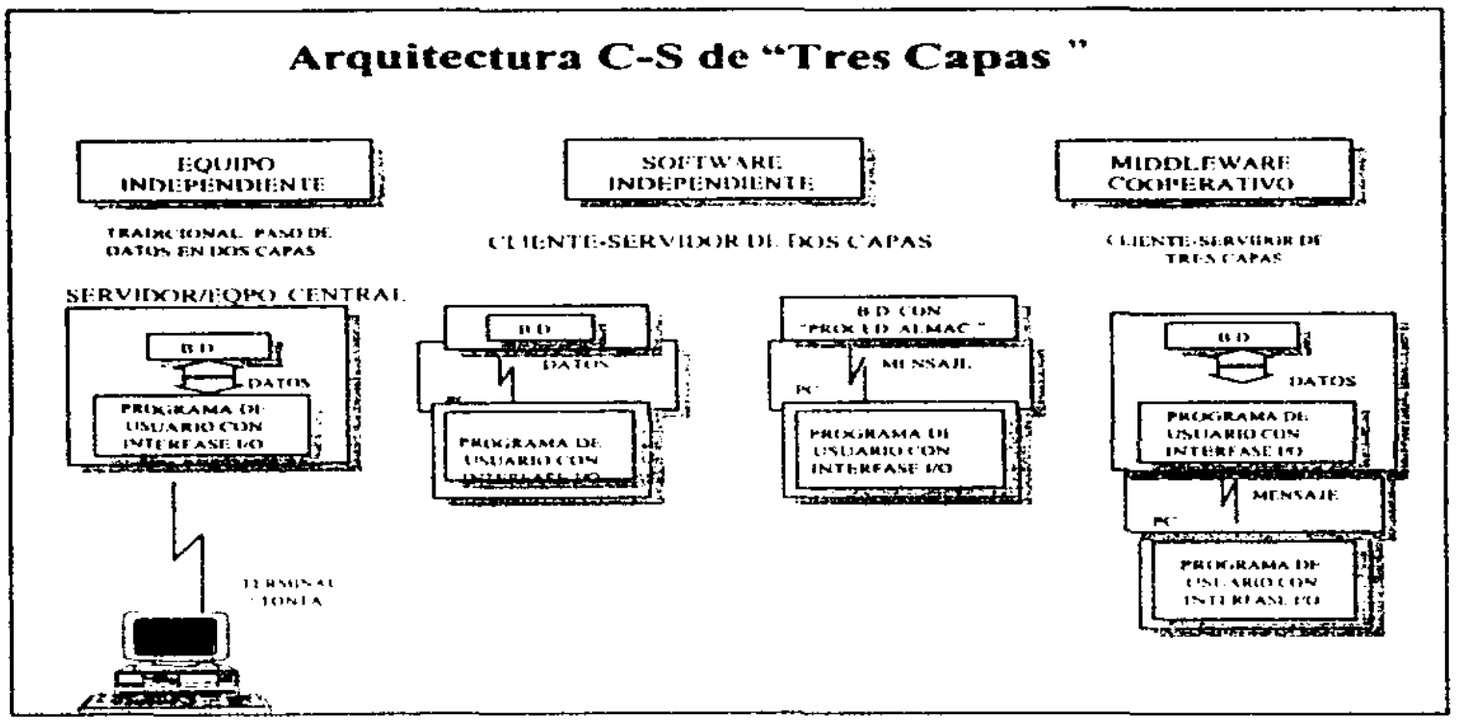


Fig. 1.7. Aplicaciones Cliente/Servidor de Tres Capas.

Ejemplos

La figura anterior muestra un ejemplo de dos y tres niveles, utilizando el mismo conjunto de productos. Aunque algunos productos tienen ciertas restricciones en cuanto al soporte de aplicaciones a dos y tres niveles, con frecuencia el nivel de flexibilidad es más una cuestión de arquitectura de la aplicación, que una característica propia de los productos.

Gestión de Transacciones

En el entorno Cliente/Servidor, el control de la aplicación se desplaza hacia el usuario. Los procesos cliente interaccionan con el usuario quien, a través de ellos puede solicitar servicios que pueden estar situados en el mismo procesador o en otro de la red. La lógica del negocio está implantada como servidores de aplicación, algunos de ellos con acceso a recursos externos, como por ejemplo, archivos planos, archivos jerárquicos, bases de datos relacionales, impresoras u otros dispositivos. Cada Gestor de Recursos es responsable de la integridad de su recursos incluyendo la recuperación física o los daños lógicos. Sin embargo, la integridad de los datos también debe garantizarse a través de todos los recursos utilizados mientras se ejecuta una transacción de negocio. Esto requiere que la transacción que se va a proteger tenga un comienzo lógico y un punto final definido con claridad, entre los cuales se protejan y se controlen los accesos a recursos. El comienzo y el final están, seguramente, dentro del proceso cliente.

Las funciones de gestión de transacciones son vitales en un proceso distribuido. Normalmente las transacciones se inician y finalizan en la estación de trabajo del usuario. Un soporte de "two-phase commit" como parte de un Sistema de Gestión de Bases de Datos puede no ser suficiente

ya que la base de datos no conoce todos los recursos involucrados en la ejecución de la transacción de negocio.

1.2.Sistemas Abiertos

Objetivos

Existen muchas definiciones de Sistemas Abiertos; entre ellas destacamos la de IEEE:
Un conjunto completo y consistente de estándares internacionales de tecnología de información y de estándares funcionales, que especifica interfaces, servicios formatos de soporte para conseguir la interoperabilidad y portabilidad de aplicaciones, datos y personas.

Según ISO: Todo el conjunto de interfaces, servicios y formatos de soporte, además de otros aspectos de usuarios, para la interoperabilidad y la portabilidad de aplicaciones, datos o personas, según se especifica en los estándares y perfiles de tecnología informática.

La apertura del sistema y de la solución es uno de los requisitos fundamentales que las empresas tratan de observar cuando piensan en Cliente/Servidor.
Para el propósito de este documento nos referimos a apertura en el contexto siguiente:

Libertad para seleccionar el sistema más adecuado. Una de las promesas de Cliente/Servidor es que pueda elegirse la plataforma de sistema más adecuada. Esto incluye estaciones de trabajo y servidores departamentales y corporativos. Dependiendo de las necesidades específicas, es probable que en una instalación se requieran procesadores de distintos tipos o arquitecturas.
Así, un banco puede decidir utilizar procesadores Intel como servidores para sus sucursales medianas y pequeñas, mientras que en las grandes sucursales y departamentos se instalan procesadores RISC. De forma similar, las estaciones de trabajo para las aplicaciones bancarias pueden funcionar en OS/2 y Windows mientras que las salas de contratación pueden disponer de estaciones basadas en Unix

Sucesivas tecnologías de hardware podrán incorporarse con el tiempo. Por ejemplo, podrá utilizarse multimedia para estaciones de trabajo manejadas por los mismos clientes en lugar de los empleados. En los bancos, compañías de seguros, líneas aéreas y de ferrocarriles hay una tendencia creciente a proporcionar estaciones de trabajo de autoservicio. Aunque es conveniente la no proliferación de diferente hardware y software, a veces es inevitable.
El logro de este objetivo es fruto de la portabilidad e interoperabilidad de las aplicaciones.

Protección de la inversión. La mayoría de las empresas han hecho grandes inversiones en los sistemas de aplicación que tienen instalados, así como en formación. Como es impensable una brusca sustitución, habrán de buscarse enfoques graduales que permitan la introducción de nuevas aplicaciones que utilicen el modelo Cliente/Servidor, mientras la estructura existente continúa funcionando.

La clave para proteger las inversiones es la interoperabilidad entre las viejas aplicaciones y las nuevas o diferentes, funcionando incluso en sistemas dispares.

Libertad para elegir el modelo de distribución Cliente/Servidor. Un buen sistema Cliente/Servidor debe servir para apoyar a la organización y a los procesos comerciales y debe adaptarse a ellos. Desde un punto de vista tecnológico, la solución debe construirse de forma que se pueda modificar cuando sea necesario, sin que las aplicaciones se vean afectadas. La elección de uno o más modelos de distribución Cliente/Servidor debe basarse en razones del negocio, y no en la capacidad o funcionalidad de una herramienta de desarrollo o de un entorno de ejecución.
Para disponer la flexibilidad necesaria, la infraestructura Cliente/Servidor ideal deberá ser capaz de soportar todos los modelos de distribución Cliente/Servidor, es decir, presentación, datos y función distribuidos. Además debe acoger las tecnologías actuales, pero también las nuevas tecnologías, como Orientación a Objetos o multimedia, cuando estén disponibles y su uso se estime oportuno.

Capacidad para explotar las aplicaciones estándar. Muchas empresas adoptan la estrategia de comprar aplicaciones en lugar de desarrollarlas. Frecuentemente, sin embargo, han de desarrollar algunas funciones, bien por no estar disponibles comercialmente, o porque suponen una ventaja competitiva.

No es extraño encontrar una situación similar a la siguiente: las aplicaciones existentes se mejoran funcionalmente utilizando una herramienta de interfaz gráfica de usuario y con software de productividad personal como hojas de cálculo, procesadores de textos o correo electrónico. Para la contabilidad financiera, contabilidad general, cuentas a cobrar y a pagar, etc., se utilizan paquetes como SAP R3, mientras que aplicaciones específicas del cliente se desarrollan con diversas herramientas orientadas a objetos.

En este contexto es importante una infraestructura común Cliente/Servidor que permita integrar aplicaciones para conseguir una interfaz de usuario homogénea.

1.3.Diseño de Aplicaciones Cliente/Servidor

La total flexibilidad relativa a la distribución de funciones y datos en un entorno Cliente/Servidor, no se consigue fácilmente sin costos. Las aplicaciones tienen que diseñarse adecuadamente. Anteriormente hemos visto cómo una aplicación puede dividirse en funciones independientes y cómo éstas se trasladan a procesos cliente y servidor, enfatizando la importancia de la separación de funciones.

Por lo tanto, en cualquier aplicación bien diseñada, las funciones siguientes deben desarrollarse como componentes individuales según sus características, es decir funciones relativas a

- Lógica de presentación.
- Lógica de negocio.
- Lógica de datos.

Otro conjunto de funciones llamado Servidor de Aplicación que se describen aquí mismo.

Lógica de presentación, de negocio y de datos.

Lógica de presentación: Se refiere a la interacción entre la lógica de negocio y los gestores de recursos de presentación, y que se ocupan de la presentación de información en pantallas, impresoras, multimedia, etc. Separando la presentación de la lógica de negocio y de datos se consigue el máximo nivel de portabilidad y distribución. Por razones prácticas esta parte se ejecuta normalmente en la estación de trabajo del usuario, mientras que las funciones de la lógica de negocio y de datos pueden distribuirse entre distintos procesadores.

La lógica de presentación aísla a la lógica de negocio de las particularidades del software de gestión de presentaciones. Se ocupa de comprobación formal de datos, así como de validaciones específicas de la aplicación. Puede tener interfaz directa con el correspondiente software de gestión de presentación.

Alternativamente, puede utilizarse una herramienta de Interfaz Gráfica de Usuario (GUI) de alto nivel, que produce una lógica de presentación encapsulada (Servidor GUI). Una clara separación de la lógica de negocio facilita el cambio de la herramienta de desarrollo elegida para implantar la lógica de presentación, sin que la lógica de negocio o de datos se vean afectadas.

Lógica de negocio: Contiene las funciones de la aplicación y es la razón por la que existe ésta. La lógica de negocio debe estructurarse de forma que corresponda a funciones (u objetos) que el usuario pueda invocar a través de la GUI (Graphic User Interface).

Las funciones de la lógica de negocio deben ser desarrolladas como servidores independientes, con interfaces bien definidas y proporcionando un conjunto limitado de funciones. Esto se percibe como una encapsulación a nivel de aplicación. El resultado son unos servidores de aplicación que pueden invocarse desde distintos procesos cliente, independientemente del tipo de interacción con el usuario o de interfaz (es decir, que esté basada en caracteres o en gráficos), o desde otros servidores de aplicación. Si la función se desarrolla como un servidor de aplicación, puede conseguirse una clara separación entre la lógica de presentación y la de negocio, y los servidores de aplicación podrán luego utilizarse para futuras aplicaciones. Se pueden desarrollar servidores de lógica de negocio utilizando lenguajes de procedimientos u Orientados a Objetos. En un entorno de aplicaciones pueden coexistir servidores escritos con distintos lenguajes de programación.

Lógica de Datos: Se refiere a la interacción entre la lógica de negocio y los gestores de recursos de datos; trata de la recuperación y actualización de datos, soportando archivos planos, jerárquicos, relacionales y base de datos orientadas a objetos.

La lógica de datos (también llamada servicios de acceso a datos) está ligada a la Interfaz de los Programas de Aplicación (API) para archivos planos o jerárquicos y para la base de datos relacionales u Orientadas a Objetos.

Mediante la separación de funciones, pueden elegirse los lenguajes de programación más adecuados para desarrollar cada una de las partes. A medida que hay más disponibilidad de las herramientas de programación Orientadas a Objetos (OO), así como las librerías de clases de aplicaciones, se han podido utilizar para desarrollar nuevas funciones de la lógica de negocio. Los

servidores escritos con lenguajes procedurales pueden acceder a servicios desarrollados con herramientas de programación OO. De manera inversa, una nueva aplicación que utilice técnicas OO puede acceder a servicios existentes que hubieran sido escritos con lenguajes procedurales.
La separación de funciones también permite sustituir tecnologías antiguas por otras nuevas, sin afectar al conjunto de la aplicación. Pueden escribirse nuevos servidores en un lenguaje de programación Orientado a Objetos, de la misma manera que una interfaz de usuario activada por acciones "event drive", puede sustituirse por una interfaz de usuario OO cambiando únicamente la lógica de presentación. El acceso a archivos planos puede sustituirse por accesos a una base de datos relacional cambiando la lógica de datos.

Una neta separación de funciones posibilita la distribución de la aplicación en un modelo Cliente/Servidor. Las funciones pueden ubicarse en los procesadores más adecuados de la red y pueden cambiarse de lugar cuando cambie el entorno.

Servicios de Aplicaciones

Ámbito

En todas las aplicaciones hay un conjunto de funciones que no deben desarrollarse como parte de la lógica de negocio o de datos, ya que tendrían un impacto negativo sobre la portabilidad y, por tanto, sobre la flexibilidad para distribuir funciones y datos. Las que describimos a continuación, deben desarrollarse como parte de un grupo distinto, denominado Servidor de Aplicaciones

Funciones específicas de entorno operativo.

La programación que depende del Sistema Operativo no debe incluirse en la lógica de presentación, de negocio o de datos, ya que en caso contrario el código resultante no podrá portarse a una plataforma distinta. Esto también es válido para cualquier código de aplicación que trate del proceso por excepción y del tratamiento de errores con gestores de recursos. Un ejemplo que puede servirnos es el redireccionamiento de una salida de impresión a un dispositivo alternativo, si el dispositivo inicial no está disponible (reconfiguración dinámica).

Enlace para gestionar la disponibilidad y rendimientos. Utilizando este enfoque estas funciones no son parte de la aplicación real y pueden sustituirse fácilmente cuando estén disponibles funciones más estándar.

Servicios comunes a múltiples aplicaciones.

Funciones operativas, como el arranque y cierre de estaciones de trabajo y servidores, proporcionando un acceso único para todas la aplicaciones.

Funciones relacionadas con la gestión del sistema, que incluyen la reserva, archivo y restauración de datos locales. El medio donde se realiza el respaldo (backup) y el archivo, así como los procedimientos correspondientes pueden pertenecer al entorno del sistema específico.
Servicios de importación y exportación de datos, que ofrecen copias extractadas de las bases de datos de reserva y todas las funciones requeridas para sincronización de bases de datos.

Control de sucesos y flujo de trabajo de los procesos de negocio.

Los necesarios controles de flujo de un proceso de negocio (aplicación, transacción) para garantizar que se realizan determinados pasos en una correcta secuencia para evitar cambios en el

programa, el control de flujos no debe incluirse en la lógica de presentación, de negocio o de datos.

Las acciones a tomar a nivel de aplicación cuando no está disponible un servidor. Normalmente, el proceso no puede terminarse si un componente o servidor está ocupado en otra tarea. Frecuentemente se necesitan funciones de proceso específicas de la aplicación para afrontar estas condiciones especiales. Dependen de la naturaleza del problema y de las necesidades de disponibilidad de las funciones de aplicación.

En esta categoría también se encuentra la gestión de sucesos asíncronos, tales como mensajes no solicitados (broadcasts) y la invocación de la función correspondiente para gestionar la solicitud.

Funciones de Monitoreo:

- Gestión de tareas y equilibrado de carga de programas, si no la proporciona el software del sistema.
- Gestión de errores lógicos incluyendo el registro de las condiciones de excepción.
- Enlaces que pueden ser necesarios para supervisar la disponibilidad y rendimiento de componentes individuales del sistema.

Consideraciones de diseño

Los Servicios de la Aplicación aíslan la lógica de presentación, de negocio y de datos del entorno operativo. Los cambios en este entorno suponen cambios sólo en un lugar, y no en los múltiples componentes que pueden verse afectados.

Los Servicios de Aplicación son un grupo lógico de funciones. Si se aíslan en un único lugar las funciones anteriores, normalmente desarrolladas como parte de la aplicación, se mejora el mantenimiento. Contienen código escrito por el usuario para aspectos específicos de la aplicación que utilizan software estándar disponible.

Cuando se eligen la arquitectura de las aplicaciones y los principios de diseño, una de las decisiones básicas es si se debe permitir a los procesos cliente y servidor que invoquen directamente a los gestores de recursos, o bien tener todas esas APIs en un solo lugar, es decir, en los Servicios de Aplicación. La ventaja de este enfoque es que las APIs pueden cambiarse más fácilmente, ya que las modificaciones se realizan en un único lugar (los Servicios de Aplicación) en lugar de hacerlo en muchos módulos funcionales.

Los criterios que pueden ayudar a tomar esta decisión son:

Estándares. Se puede decir no utilizar directamente APIs que no se ajusten a estándares. En ese caso, habrá que diseñar y desarrollar funciones de Servicios de Aplicación específicas del cliente. Estas funciones sirven para relacionar las APIs proporcionadas por el gestor de recursos de que se trate con las APIs independientes del producto. Esta técnica ha sido utilizada durante años por muchos clientes desarrollando subrutinas que realizan funciones similares.

Soporte de más alto nivel. Cuando el API proporcionado por el gestor de recursos se considera demasiado complejo para los que desarrollan aplicaciones, se pueden disponer de funciones de

mayor nivel como parte de los Servicios de Aplicación para aislar a los programadores de esta complejidad.

De forma similar, pueden desarrollarse funciones de aplicación estándar que sean transparentes a las funciones de la lógica de presentación, de negocio y de datos. Como ejemplo citaremos el registro automático de sucesos o el mantenimiento de un seguimiento de auditoría, y la lógica de aplicación para realizar procesos alternativos, si un recurso o servidor no está disponible.

Duración prevista. Cuando se puede prever que se migrará de un gestor de recursos a otro en el próximo futuro, es también aconsejable desarrollar funciones de Servicios de Aplicación específicas del cliente. Por ejemplo, si se contempla el uso de una base de datos Orientada a Objetos en algún momento del tiempo, puede ser aconsejable no utilizar SQL directamente en las aplicaciones, y en su lugar proporcionar funciones de acceso a datos de alto nivel como parte de los Servicios de Aplicación.

Grado de independencia. Para disfrutar de una total independencia de las APIs proporcionadas por un gestor de recursos, también se pueden desarrollar funciones de Servicios de Aplicación específicas del usuario.

Relación con la Plataforma Operativa.
Las funciones de la lógica de presentación, de negocio y de datos, si se desarrollan con lenguajes o herramientas adecuados, son portables y pueden situarse en el procesador más adecuado para su ejecución. Según se define en los Servicios de Aplicación, en cada procesador también hay un grupo de funciones. El Middleware aísla los elementos de la aplicación de los aspectos de distribución relativos al acceso de recursos locales y remotos (dispositivos, programas, datos). Estos elementos se representan en la parte superior de la figura.

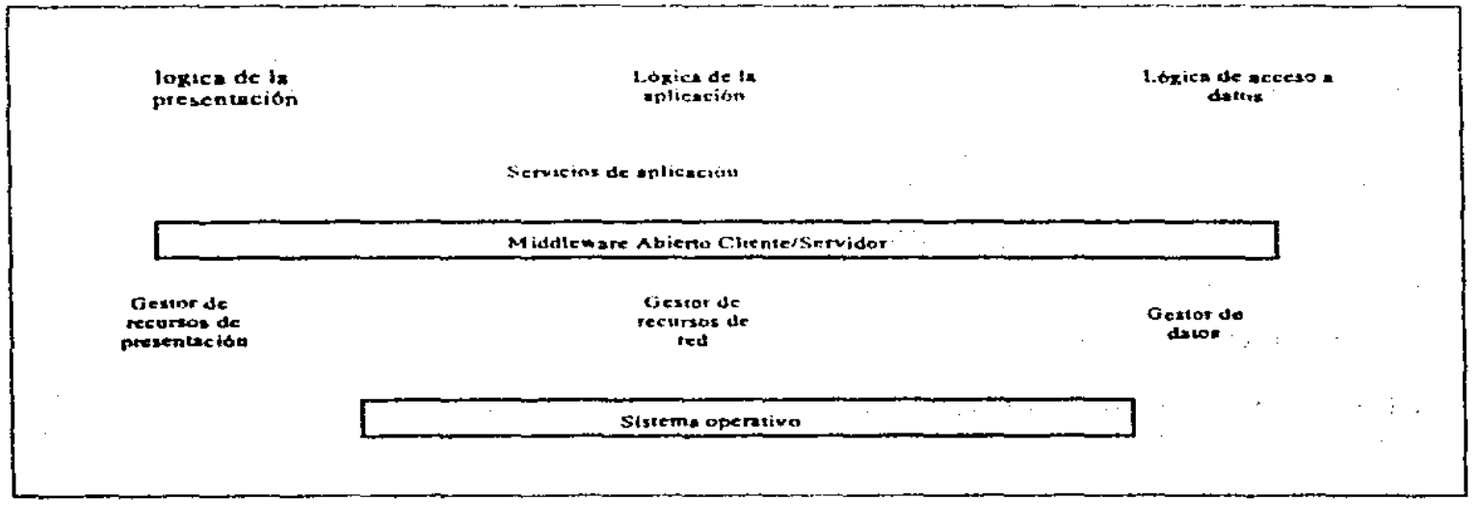


Fig. 1.8. Relación con la plataforma operativa.

1.4.Entorno de Desarrollo de Aplicaciones

Respecto a esta importante área, podemos afirmar que no existe una única solución válida universal para todas las problemáticas posibles, ya que las situaciones son muy diferentes en cada empresa; distintos Sistemas Operativos, distintos modelos de Distribución Cliente/Servidor o empleo de diferentes tecnologías. Algunas compañías optan por trabajar con software ya probado, mientras que otras prefieren utilizar el software más reciente. Hay quienes han realizado grandes inversiones en herramientas y no están dispuestos a abandonarlas. Lo que es aceptable en una situación puede no serlo en otra. Sólo teniendo en cuenta las necesidades y eligiendo los productos de forma conciente, estaremos seguros de optar por la mejor solución.

Enseguida se explica la relación entre la Plataforma Operativa y el Entorno de Desarrollo de Aplicaciones. Todo ello sin olvidar que la generalización ayuda a comprender los principios y las características, pero puede ser incorrecta con determinados productos.

Relación con la plataforma operativos

El Entorno de Desarrollo de Aplicaciones y la Plataforma Operativa, según se ha visto en el capítulo anterior, están estrechamente ligados. No se pueden considerar de forma independiente por las razones siguientes:

- Una Plataforma Operativa para la que no existan herramientas de desarrollo, difícilmente puede ser útil. Por lo tanto, al definirla han de tenerse en cuenta los aspectos de Desarrollo de Aplicaciones y viceversa.
- Algunas herramientas de desarrollo llevan integradas partes de la Plataforma Operativa o incluso pueden llegar a predeterminar una en concreto. Esto puede conducir a un sistema muy cerrado y propietario si se produce alguna de las siguientes situaciones:
  - La comunicación entre los procesos cliente y servidor se basa en protocolos de transporte de bajo nivel. En este caso, el producto puede imponer el protocolo que ha de utilizarse y los sistemas que deben interconectarse.
  - La interfaz entre los procesos cliente y servidor no es externa. En este caso, un proceso servidor no puede ser invocado por un proceso cliente construido con una herramienta diferente.
  - Los directorios y el sistema de seguridad se desarrollan de forma propietaria. En el mejor de los casos, esto ocasiona una complejidad añadida si requiere la utilización de otros productos Middleware. El caso más desfavorable se produce cuando las funciones Middleware de un producto limitan el tipo de sistema y de servicios de comunicación soportados.
  - Muchas herramientas de desarrollo soportan únicamente un subconjunto de los distintos modelos de Distribución Cliente/Servidor, o solamente las soluciones Cliente/Servidor que hemos denominado de dos niveles.

Cuando se define una infraestructura Cliente/Servidor el objetivo es seleccionar plataformas y herramientas que proporcionen flexibilidad para los distintos Modelos de Distribución y soporten un Entorno de Aplicaciones Incremental, en el que puedan desarrollarse nuevas aplicaciones añadiendo más procesos cliente y servidor.

El proceso de desarrollo

Anteriormente hemos mencionado dos conceptos básicos para conseguir los beneficios derivados de Cliente/Servidor:

- Estructura de la Aplicación. Uno de los objetivos de la separación estricta de los componentes de la lógica de presentación, lógica del negocio, y lógica de datos es que puedan distribuirse en una red. Esto implica que los componentes se desarrollen y se prueben como entidades distintas. Ello es crucial dado que los tres componentes son de naturaleza diferente y se necesitan herramientas distintas para construirlos.
- Entorno de Aplicaciones. Al implantar un Entorno de Aplicaciones Incremental, los procesos cliente y servidor (que son funciones del negocio encapsuladas) se desarrollan como entidades separadas.

Esto también resulta prioritario, ya que, con el tiempo, se dispondrá de nuevas tecnologías y surgirán nuevas herramientas que las soporte. Una correcta infraestructura Cliente/Servidor proporcionará el marco adecuado para poder usar herramientas de desarrollo de diversas procedencias.

Para aprovechar las ventajas que ofrece el modelo Cliente/servidor, probablemente se utilizarán distintas herramientas o lenguajes para desarrollar las diversas partes de una aplicación. Seleccionar las herramientas de desarrollo adecuadas no garantiza que la aplicación se diseñe de acuerdo con los principios de separación de funciones y encapsulación. De igual importancia son la arquitectura y diseño de la aplicación. Probablemente también sea necesario un nuevo proceso de desarrollo que dé cabida a un enfoque iterativo.

Proceso en cascada. El enfoque tradicional para el desarrollo de aplicaciones se ha descrito como proceso en cascada. Los pasos de definición de necesidades, análisis, diseño, desarrollo y prueba aparecen en una determinada secuencia; un paso sólo puede comenzarse una vez acabado otro.
En realidad, cada paso frecuentemente vuelve a realimentar los pasos anteriores. Algunos productos son muy rígidos en cuanto a que el paso siguiente de un ciclo sólo puede llevarse a cabo si se ha terminado el anterior (los resultados del paso anterior son los datos de entrada al siguiente).

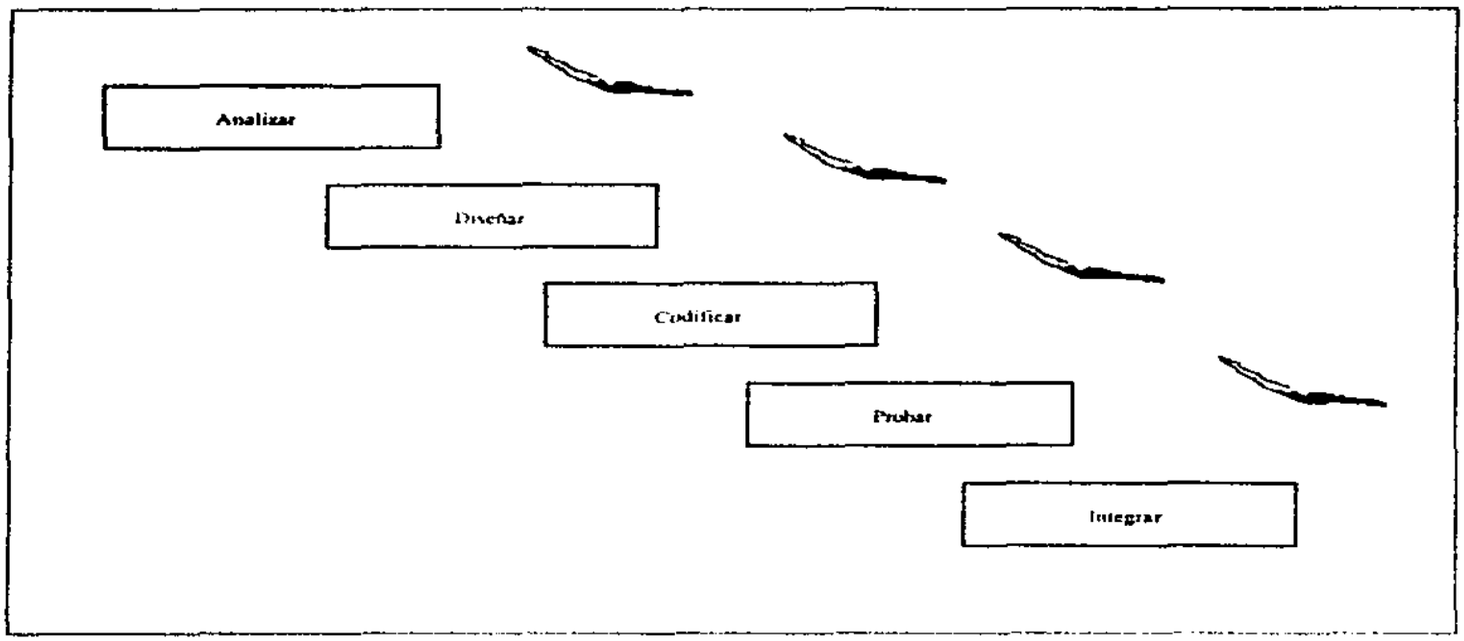


Fig. 1.9. Proceso de Desarrollo en Cascada.

Los requerimientos de la aplicación permanecen fijos pasadas las primeras fases del proceso de desarrollo. Sin embargo, los requerimientos del negocio son mucho más dinámicos. Cuando una aplicación tarda 2 ó 3 años en desarrollarse, lo más probable es que al final no cubra todas las necesidades existentes. Por lo tanto, comienza el mantenimiento.

Como una de las razones de la existencia de Clientes/Servidor es la necesidad de una inmediata adaptación al mundo de los negocios, este enfoque no es precisamente, el adecuado para nuestro propósito. Es demasiado rígido y lento para responder al creciente ritmo de cambio de la empresa actual.

Muchos estudios han demostrado que más del 70% del tiempo de desarrollo y de los recursos se invierten en mantener las aplicaciones existentes. Esto no incluye solamente corregir fallas o defectos, sino añadir funciones que deberían haber estado desde el comienzo o funciones que cubran posibles modificaciones en los requerimientos.

Proceso iterativo. La nueva tendencia es utilizar un diseño iterativo, también conocido como proceso en espiral. Las herramientas Cliente/Servidor y Orientadas a Objetos son muy adecuados para este tipo de procesos, en los que, una vez terminada la etapa de análisis, los programadores trabajan directamente con el usuario utilizando prototipos que se generan muy rápidamente. El estudio de necesidades en profundidad y la documentación no se hace hasta más tarde. Las herramientas facilitan este esquema. El diseño iterativo permite que el desarrollo de la aplicación evolucione incrementalmente de un paso al siguiente, volviendo a comprobar los resultados. En este contexto, algunas veces se utiliza la frase "diseñar un poco", "codificar un poco", "probar un

poco". La ventaja derivada es una gran participación del usuario, con la consiguiente mayor rapidez en la puesta en explotación de la aplicación, incluyendo una mayor riqueza funcional.

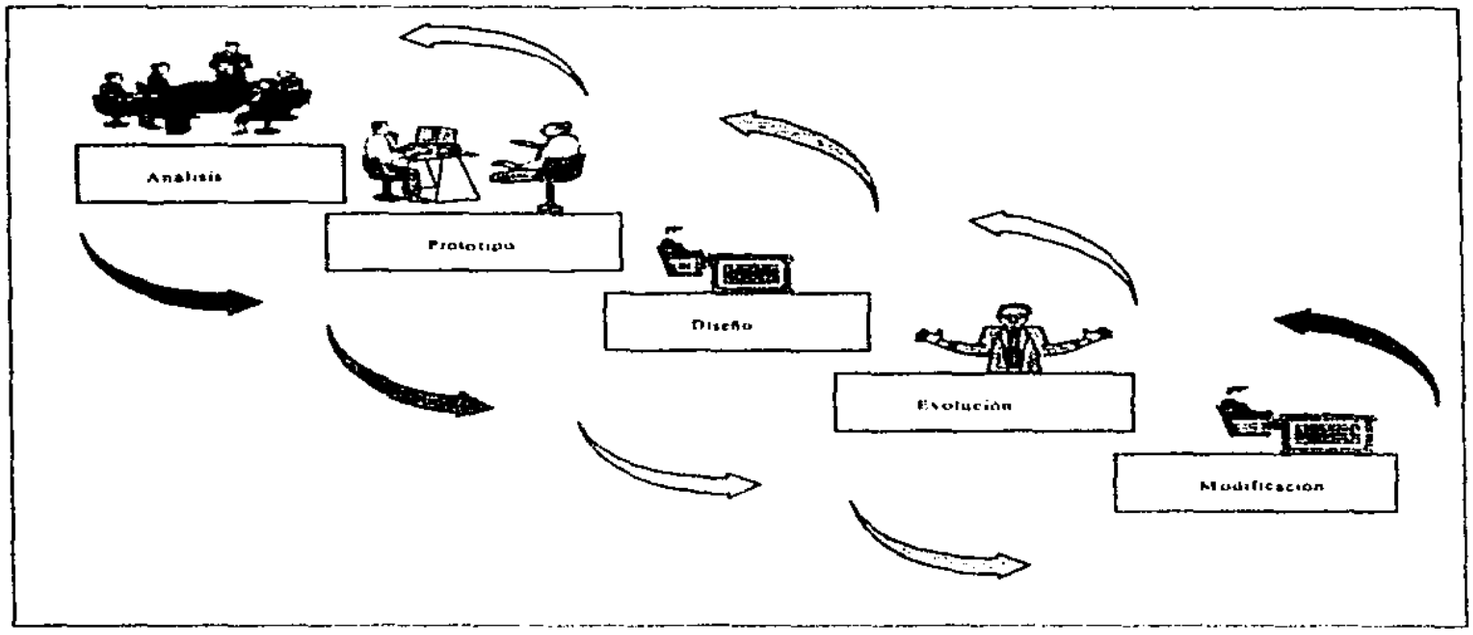


Fig. 1.10. Proceso de Desarrollo Iterativo.

Este proceso se adapta mejor que el esquema en cascada a la necesidad de dar respuesta rápida a las necesidades del negocio. Es muy adecuado para construir el Entorno de Aplicaciones Incremental. A medida que se construye y se prueba, esta técnica permite la puesta en marcha de componentes más experimentados, que han sido mejorados mediante iteraciones del proceso de desarrollo.

La construcción de prototipos se ha convertido en uno de los pasos más importantes en el proceso de diseño de aplicaciones, hasta el punto de que estos componentes no se consideran como "desechables" al final, sino que al pertenecer a aplicaciones desarrolladas iterativamente, su existencia contribuye a que los proyectos se realicen por fases de una forma más natural.

Debe utilizarse pues, las mismas herramientas para los prototipos que para el desarrollo de las aplicaciones de producción, y no se debe descuidar el trabajo de diseño y arquitectura de la aplicación. Uno de los retos de los prototipos es convencer al usuario de que el proceso no se acaba una vez que ha visto un prototipo del "front-end" de la aplicación. Aquí se aplica la regla 80/20, es decir, que el 80% del trabajo se realiza en el 20% del tiempo, mientras que el otro 20% de trabajo restante requiere el 80% del tiempo.

Cliente/Servidor y Orientación a Objetos.

En primer lugar, cabe afirmar que Cliente/Servidor y Orientación a Objetos (OO) no tienen nada que ver entre si; las aplicaciones Cliente/Servidor se pueden diseñar con características que no son Orientadas a Objetos; de la misma manera, una aplicación Orientada a Objetos puede no tener ningún atributo Cliente/Servidor. Sin embargo, los dos conceptos funcionan bien conjuntamente. Seguramente al desarrollar aplicaciones Cliente/Servidor, se introducirán y utilizarán progresivamente herramientas y técnicas Orientadas a Objetos. En cualquier caso, OO es una importante tecnología que no se puede ignorar; una infraestructura Cliente/Servidor y el Entorno de Desarrollo de Aplicaciones deben ser capaces de incorporar a las tecnologías OO, aunque no se utilicen inicialmente.

Dentro de este mundo se pueden identificar varias disciplinas: Interfaces de Usuario (OOUI), Diseño OO, Programación OO y Bases de Datos OO, las cuales son, en cierto modo, independientes entre si. Un purista OO las usaría todas; sin embargo, algunas de ellas pueden combinarse con las tradicionales equivalentes.

Interfaces de Usuario Orientadas a Objetos (OOUI) No siempre, aunque con mucha frecuencia, las aplicaciones Cliente/Servidor se construyen con Interfaces Gráficas de Usuario (GUI) en lugar de las tradicionales interfaces basadas en caracteres. Una OOUI es una clase especial de GUI (Graphic User Interface) basada en el concepto objeto-acción (en lugar de acción-objeto) e incorpora las siguiente características generales:

- El usuario selecciona objetos mediante un mouse y lo "arrastra" hacia el objetivo.
- El usuario obtiene una respuesta inmediata a las acciones seleccionadas.
- La interfaz no tiene estados o modos que alteren el significado de las acciones del usuario.
- Los objetos se presentan al usuario WYSIWYG (del inglés "What You See Is What You Get", es decir, lo que se ve es lo que se obtiene).
- Los objetos y el significado de las acciones son consistentes, tanto dentro de una aplicación como entre varias.

Los objetos se asemejan a los objetos del mundo real a través de un modelo conceptual subyacente, es decir, la visión del usuario de lo que ocurre. Como ejemplo de una sintaxis objeto-acción, consideremos la supresión de un archivo llamado documento.1st. Con una interfaz de usuario de un sistema operativo típico, el mandato puede ser "erase documento.1st". Primero se especifica la acción, luego el objeto. Con una interfaz de usuario orientada a objetos, el usuario borra el archivo seleccionado el objeto y arrastrándolo hasta un icono con el aspecto de tirar un papel a la papelera. Esta analogía ofrece simplificación al usuario, aunque la acción implique dos objetivos.

Nótese que una GUI (Graphic User Interface) no está orientada a objetos por definición. Las primeras aplicaciones con una GUI estaban más orientadas a tareas que a objetos. Las investigaciones han demostrado que una OOUI es más intuitiva y productiva para los usuarios. Uno de los criterios de selección de una herramientas GUI debería ser si soporta o no, la construcción de una OOUI.

1.5.Cliente/Servidor, un nuevo paradigma

Aquí resumimos las cuestiones derivadas del cambio de paradigma que supone el modelo Cliente/Servidor. La mayoría de ellas, si no todas, son cruciales para el éxito de la implantación Cliente/Servidor, pero normalmente no se tratan de forma adecuada si el departamento de Sistemas de Información no participa activamente en los proyectos Cliente/Servidor. En general, a los departamentos de usuarios y a la línea de negocio no les preocupa estos aspectos.

Si se quiere evitar la proliferación de distintos sistemas y plataformas de aplicación y la incompatibilidad entre los diversos desarrollos, el departamento informático debe asumir la responsabilidad corporativa en la selección de:

- Plataformas del sistema (por ejemplo procesadores para los servidores y estaciones de trabajo para los usuarios, así como sistemas operativos soportados).
- Estándares, formatos y protocolos aplicables al hardware y software del sistema y aplicaciones.
- Software para los componentes de Middleware de las Plataformas Operativas y Herramientas de Desarrollo de Aplicaciones.
- Software de aplicación estándar disponible en el mercado.

Informática Dirigida al Usuario.

La tendencia imparable de un mayor control de la aplicación por parte del usuario, modifica las exigencias de infraestructura relativas a:

- Seguridad, que incluye aspectos de identificación de usuarios, control de accesos, confidencialidad de datos en las estaciones de trabajo, servidores y red.
- Medidas organizativas respecto a responsabilidad y propiedad de los datos en las estaciones de usuario y en los servidores.
- La necesaria normativa que propicie la integración de las aplicaciones de desarrollo propio con las estándar. Esto conlleva la definición de arquitecturas de aplicación y estándares que deberían incluir normas sobre esa integración.
- Infraestructura de soporte necesario para la gestión operativa, contemplando:
  - Arranque y parada de las estaciones de trabajo y servidores
  - Proceso de fin de día
  - Copia de seguridad y archivo así como la recuperación de datos
  - Dispositivos de ayuda y Help Desk
  - La necesidad de gestión y mantenimiento de aplicaciones en un sistema distribuido.

Aplicaciones Cliente/Servidor.

El desarrollo de aplicaciones Cliente/Servidor que ofrezcan una total flexibilidad en términos de función y de ubicación de datos requiere nuevos planteamientos y tiene implicaciones en el departamento de sistemas de información. Respecto a la transición de aplicaciones monolíticas a Cliente/Servidor, cabe concluir recordando los siguientes puntos:

- El modelado de los procesos de negocio es clave para identificar las funciones de la aplicación que pueden implantarse y agruparse como servidores de aplicación (funciones de lógica de negocio encapsuladas)
- Las arquitecturas de aplicaciones y los principios de diseño deben establecerse antes de desarrollar la primera aplicación. La flexibilidad y apertura de la aplicación Cliente/Servidor resultante depende de ello en gran medida.
- La ubicación de funciones y datos, así como la disponibilidad y el rendimiento tendrán una dimensión diferente. Incluso en un entorno Cliente/Servidor flexible, en algún momento habrá que tomar decisiones relativas a la ubicación de funciones y datos. Las consideraciones sobre disponibilidad y rendimiento deben estar presentes en todas las etapas del proceso de desarrollo.
- Pueden ser necesarias nuevos profesionales expertos en el desarrollo de interfaces gráficas de usuario, en el diseño y desarrollo de servidores de lógica de negocio y en la Plataforma Operativa. Esto puede exigir el uso de diferentes herramientas para desarrollar los distintos componentes de las aplicaciones.
- La validación de las aplicaciones y su distribución a través de la red requiere nuevos procedimientos, ya que estamos frente a un conjunto compuesto por numerosos procesos cliente y servidor que han sido probados individualmente.

Responsabilidades de la organización de Sistemas de Informática

Definición de una infraestructura Cliente/Servidor: No existen soluciones estándar Cliente/Servidor, aunque pueden existir bloques que se puedan reutilizar. Ni siquiera se puede asegurar, de forma general, que un modelo de distribución, una herramienta o una plataforma sean los mejores. La solución correcta depende, en gran medida de:

- Los objetivos de negocio de la empresa, del grado de aceptación de las nuevas tecnologías.
- Los recursos disponibles actualmente de hardware, software y conocimientos.

El levantamiento de una infraestructura técnica que se aparte de las necesidades del negocio está condenada al fracaso. Los factores que determinan la solución Cliente/Servidor son:

- Las necesidades del negocio y cómo las tecnologías informáticas pueden soportarlas para conseguir los objetivos comerciales.
- El marco de tiempo disponible para la nueva puesta a punto.
- El estudio económico.
- El impacto sobre la organización y los conocimientos requeridos.
- Las arquitecturas y estándares ya adoptados por la compañía.

Pocas empresas pueden permitirse sustituir los sistemas y aplicaciones existentes en un solo paso. Normalmente es necesario un proceso gradual de implantación de las nuevas aplicaciones, lo cual exige:

- Una infraestructura Cliente/Servidor capaz de acomodar las aplicaciones existentes junto con las nuevas aplicaciones Cliente/Servidor.
- Estándares corporativos de tecnología informática.

- Una arquitectura de aplicaciones que permita desarrollar y poner en servicio nuevas aplicaciones, mientras siguen funcionando las existentes, preferiblemente sin tener que modificarlas.

Plataforma Operativa: La plataforma deberá soportar todos los Modelos de Distribución Cliente/Servidor, todos los Servicios de Comunicación, y deberá utilizar preferentemente componentes estándar de la industria para los Servicios de Sistemas Distribuidos. Los desarrollos propios deben coexistir con las aplicaciones estándar y su integración deberá ser imperceptible para el usuario. Igualmente, podrán acomodarse programas escritos utilizando diferentes tecnologías y herramientas.

Entorno de Desarrollo de Aplicaciones: Debe elegirse después de la Plataforma Operativa. Aunque es conveniente evitar una proliferación de herramientas de desarrollo, se garantizará que el enlace entre éstas y el Middleware no sea excesivamente rígida. Será posible utilizar diferentes herramientas para desarrollar parte de una aplicación. Un Entorno de Aplicación debe posibilitar la coexistencia de procesos cliente y servidor desarrollados con distintos lenguajes de programación y/o herramientas, así como utilizar distintas tecnologías (por ejemplo, lenguaje procedural, lenguaje orientado a objetos, multimedia), y que han sido puestas en explotación en distintos momentos.

Administración de sistemas: Estas funciones aumentan considerablemente el coste de una solución, pero no se pueden evitar. Siempre debe adaptarse a las necesidades de la organización y al decidir la plataforma operativa y el entorno de desarrollo, es decir, en las primeras fases de la definición de la solución, merece la pena considerar los aspectos siguientes:

- ¿ Qué necesitamos gestionar?
- ¿Dónde estarán situados los procesadores y estaciones de trabajo?
- ¿Cuántos tipos distintos se soportarán?
- ¿Qué tipo de soporte es necesario y quién lo proporciona?

Cómo definir una infraestructura Cliente/Servidor. Si no se acomete el trabajo de definir una infraestructura Cliente/Servidor se corre el riesgo de que surjan en la empresa una serie de soluciones Cliente/Servidor aisladas.

No es en absoluto recomendable el intento de una infraestructura completa desde el principio, ya que las Tecnologías pueden no responder a tiempo a las prioritarias necesidades del negocio. El enfoque más adecuado está en un sistema y una plataforma de aplicación conceptuales, y una arquitectura, construida incrementalmente y ampliada a medida que se desarrollan nuevas aplicaciones.

La Plataforma Operativa, el Middleware y el entorno de Desarrollo de Aplicaciones están relacionadas entre sí. Las necesidades de apertura pueden condicionar la elección de la plataforma o de Middleware, de igual manera que lo condiciona una determinada herramienta de desarrollo. El software de aplicación puede influir en la plataforma del sistema y el tiempo disponible para la primera aplicación puede implicar algún tipo de compromiso. Por lo tanto, es necesario fijar los objetivos, y el modo de conseguirlos en cada caso concreto. Una Metodología de Infraestructura

para Sistemas Distribuidos que permita definir una infraestructura para el sistema Cliente/Servidor y evalúe la puesta en marcha del proyecto sobre una base racional.

El enfoque estructurado de dicha Metodología comprende los pasos siguientes:

- Captación de las necesidades. Definir, analizar y evaluar, aunando los requerimientos del negocio con las aportaciones tecnológicas.
- Diseño conceptual en el que se sitúan los principales bloques funcionales y de datos del sistema, mostrando la relación y comunicación entre ambos.
- Detalle de los principales componentes funcionales, selección de procesos, determinando los principios que deben aplicarse a la selección de software o diseño de los módulos.
- Al final de los tres pasos anteriores, se definen los conceptos del sistema y la infraestructura tecnológica, sin concretar, todavía, en productos a plataformas específicas.

Consideraciones sobre la Implantación Existen tres formas de abordar la implantación de soluciones Cliente/Servidor:

- Construir la infraestructura y descuidar las aspectos del negocio a corto plazo (Opción 1)
- Centrarse en soluciones de negocio a corto plazo e ignorar la infraestructura (Opción 2)
- Diseñar la infraestructura e implantarla incrementalmente a medida que se desarrollan las soluciones de negocio. (Opción 3)

La primera opción puede acarrear dos problemas: la contratación de servicios a proveedores externos a la empresa, y la implantación incontrolada de soluciones por parte de los usuarios. Aunque puede ser una solución razonable a corto plazo, provocarán, a medio y largo plazos, caos e incompatibilidades entre sistemas y soluciones distintas, debido a la falta de una infraestructura corporativa Cliente/Servidor.

El peligro inherente en la opción 2 es que las plataformas, el Middleware y las herramientas pueden cubrir las necesidades de la primera aplicación abordada, pero pueden no ser adecuados para las siguientes y, por tanto, resulta una mala inversión.

La opción 3 exige cierto trabajo inicial de diseño y arquitectura, pero garantizará una sólida infraestructura, a corto y medio plazos, ya que tiene en cuenta las necesidades generales, aunque sólo se desarrollan los componentes a medida que se necesitan. Esta opción es, obviamente, la más adecuada ya que combina las ventajas de las opciones 1 y 2. Equivale a construir una casa basándose en un plano, pero dejando sin terminar algunas habitaciones del primer piso.

Proceso de planificación de Sistemas de Información

Necesidad de un nuevo proceso. Hay que revisar y adaptar el proceso de planificación informática. El rápido cambio de las tecnologías acorta el ciclo de vida de los productos. Ya no son posibles los ciclos de planificación a 2-3 años. El tiempo transcurrido desde la concepción de un nuevo producto o servicio hasta su puesta en mercado se acorta cada vez más. El departamento de sistemas de información tiene que adaptarse. La infraestructura informática y el proceso de desarrollo tendrán que posibilitar la rápida introducción de nuevas aplicaciones, ya que en caso contrario, la empresa corre el riesgo de desaparecer.

Cartera de aplicaciones. Deben examinarse detenidamente las aplicaciones en cartera. Es muy probable que las necesidades formuladas hace uno o dos años ya no sean válidas. Hay que establecer nuevas prioridades con base en los objetivos de negocio de la empresa. El departamento de sistemas de información seguirá teniendo la responsabilidad del desarrollo de aplicaciones corporativas, así como de la gestión de los datos de la empresa. Sin embargo, una infraestructura Cliente/Servidor definida adecuadamente permitirá utilizar aplicaciones estándar seleccionadas por el departamento de informática o por los departamentos de usuarios.

Definición de responsabilidades

Propiedad. Es necesario una clara definición de la "línea de confianza" que incluya los propietarios de aplicaciones y datos corporativos departamentales de usuarios. Esta definición tiene una gran transcendencia. Regula hasta qué punto los departamentos y usuarios pueden añadir sus propias aplicaciones y datos. Estos afectan a la definición de configuraciones. No es razonable traspasar al departamento informático la responsabilidad de garantizar la disponibilidad y tiempo de respuesta, mientras se disponga de una total libertad para ejecutar aplicaciones adicionales en la estación de trabajo y/o servidores.

Evolución en el desarrollo de aplicaciones

Participación del Usuario. En el pasado, el departamento de Sistemas de Información desarrollaba aplicaciones de acuerdo con las especificaciones aportadas por los usuarios. Típicamente, ambos grupos sólo se reunían de tarde en tarde, una vez fijadas las especificaciones, y solamente para discutir sobre quién tenía la responsabilidad en los retrasos, del mal funcionamiento, etc. No es extraño que los profesionales de hoy en día sean escépticos sobre el grado de interés y la productividad de las organizaciones de desarrollo de aplicaciones que dependen de Sistemas de Información. Dentro de la tendencia de una creciente partición del usuario en el desarrollo de nuevas aplicaciones es fundamental que los usuarios y el departamento de sistemas de información puedan colaborar más estrechamente, de forma que las aplicaciones puedan ser construidas por equipos que trabajan con unos objetivos comunes de desarrollo rápido, buen rendimiento y alta calidad.

Aspectos organizativos

La introducción de nuevos métodos, herramientas y procesos de desarrollo conducen a nuevas profesiones especializadas:

- Arquitectos de sistemas cuya responsabilidad es desarrollar una arquitectura de aplicaciones corporativas y establecer principios de diseño.
- Especialistas en la definición e implantación de servidores de aplicación, para garantizar que tienen el nivel adecuado de granularidad y se adaptan a la necesidad de ser compartidos por distintos procesos cliente.

Si se utiliza tecnología de Orientación u Objetivo, se necesitan expertos en la definición y desarrollo de aplicaciones u objetivos de negocio.

El desarrollo de aplicaciones lo realizan equipos compuestos por personal de desarrollo y usuario que ahora comparten la responsabilidad con la organización de Sistemas de Información.

Junto a esta estrecha cooperación, es necesaria una estructura de dirección que permita tomar decisiones de forma compartida.

El entorno de Aplicaciones Incremental genera la necesidad de nuevos procedimientos de prueba, ya que cada componente tiene ahora que probarse aisladamente y como parte de la aplicación. El empaquetado del software de aplicación para su distribución y exige consideraciones especiales.

1.6.Costos y Beneficios del Modelo Cliente/Servidor

Costos

Los costos de la implantación de soluciones Cliente/Servidor no deben contemplarse sólo en términos absolutos, sino que deben medirse en función del beneficio que reporten los nuevos desarrollos. Como el precio de las Computadoras Personales ha bajado tanto en los últimos años, con frecuencia se comete el error de pensar que las soluciones Cliente/Servidor son económicas, o más económicas que las basadas en computadoras tradicionales. Estudios independientes indican que el coste de hardware y software, en un período de 5 años, representa solamente 20% de los costos totales. En el caso de sistemas distribuidos, el 80% restante son costos de infraestructura y gastos de explotación.

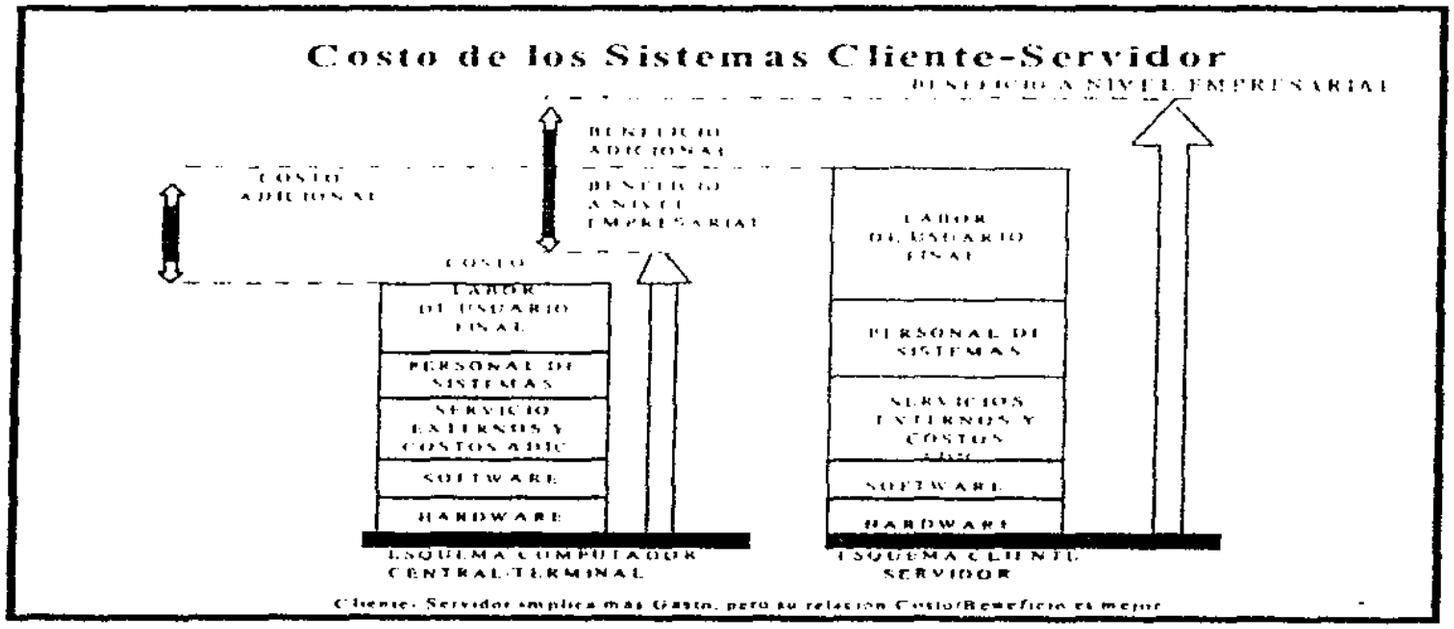


Fig. 1.11. Costos y Beneficios del Modelo Cliente/Servidor.

Salvo que se opte por una implantación muy a la ligera, descuidando aspectos claves, los costos iniciales de los sistemas distribuidos pueden ser substancialmente más elevados de lo previsto. Algunos de los aspectos que, a menudo, se olvidan son:

- Aumento de la complejidad debido a la conexión de múltiples sistemas heterogéneos.
- Diversidad de fuentes de datos.
- Nuevas plataformas (sistemas operativos) y operaciones.
- Inversión inicial en infraestructura.
- Formación y creación de especialistas en nuevas herramientas y metodologías.
- Arquitectura de aplicaciones y diseño para un entorno distribuido.
- Distribución de software/gestión de licencias.
- Nuevas guías de procedimientos operativos.
- Soporte técnico a los usuarios.

Beneficios
Los beneficios percibidos se encuadran, generalmente, en alguna de estas categorías:

- La productividad que se obtiene en las Estaciones de Trabajo Programables con Interfaz Gráficas de Usuario, que permite acceder e integrar aplicaciones muy intuitivamente.
- La abundancia de software disponible comercialmente, como por ejemplo procesadores de textos, hojas de cálculo, sistemas basados en el conocimiento, correo. etc.
- La cercanía del usuario a aplicaciones y datos que son necesarios para su actividad, compartiendo servicios (y costos).
- La disponibilidad de potencia de cálculo a nivel personal, sin la responsabilidad del mantenimiento del sistema y del software de aplicaciones.
- La disponibilidad de herramientas de desarrollo fáciles de usar, reduciendo la dependencia del departamento informático.

Los beneficios percibidos por la alta dirección seguramente estarán entre los siguientes:

- Un mejor ajuste del Sistema de Información a la organización y a los procesos de negocio. Cada tarea se puede ubicar en la plataforma que sea más eficaz, y los recursos informáticos se pueden ampliar progresivamente. Localización de funciones y datos donde sea necesario para la operativa diaria sin cambiar las aplicaciones.
- Mayor protección de activos informáticos e integración de los sistemas y aplicaciones ya existentes. Adición de nuevas funciones a las aplicaciones, cambiando solamente las partes afectadas, sin cambiar el conjunto (aplicaciones incrementales).
- Acceso a la información cuando y donde la necesitan los usuarios. Este es, probablemente, el mayor activo corporativo.
- Disponibilidad de aplicaciones estándar (comprar en vez de desarrollar).
- Libertad para migrar a plataformas de sistemas alternativos y usar servidores especializados.
- Una respuesta más rápida a las necesidades del negocio gracias a la disponibilidad de software de productividad personal y de herramientas de desarrollo fáciles de usar.

Un entorno de utilización más sencillo, que proporciona una mayor productividad. A largo plazo, las GUI reducen los costos asociados a educación y formación de los usuarios.

Los beneficios que se derivan de una aplicación Cliente/Servidor dependen, en gran medida, de las necesidades del negocio. Por ejemplo, renovar una aplicación existente añadiéndole una

Interfaz Gráfica de Usuario (GUI) podria ser una solución rentable en un determinado momento, pero puede no serlo en otro. No se debe generalizar.

1.7.Cliente/Servidor en Internet

En esta sección damos un panorama general de la estrecha relación que existe entre la plataforma cliente/servidor e Internet. Así mismo, se toma el punto de vista de un fabricante de herramientas de desarrollo muy importante como lo es Powersoft, con el fin de hablar de un caso específico del conjunto de herramientas y terminología que están surgiendo en el escenario de los sistemas de información. Por otro lado, es esta plataforma de desarrollo la que estamos utilizando para este proyecto.

La Internet - Una nueva Plataforma :

La Internet es una nueva plataforma de cómputo, como antes lo fue cliente/servidor. Al igual que las plataformas anteriores, Internet promete un nuevo enfoque para abordar de mejor manera ciertos requerimientos de los negocios. Mucha gente ve a Internet como un complemento muy importante de la arquitectura cliente/servidor - más evolución que revolución. Un artículo reciente en la revista Information Week puntualizó "La computación en Internet no es sólo una nueva arquitectura, es realmente una extensión natural de la computación cliente/servidor". Además de la naturaleza complementaria de la computación en Internet y cliente/servidor, qué aspectos más importantes hay en lo que respecta a implicaciones y beneficios de Internet para los usuarios, desarrolladores y administradores de sistemas de información.

Los usuarios actualmente se benefician de un acceso a la información sin precedentes, tanto desde fuentes internas (intranet) como externas (World Wide Web). (De hecho, dado que hay vastas cantidades de información disponibles en la red, la habilidad para encontrar exactamente lo que se está buscando se ha convertido en todo un arte). Adicionalmente a la accesabilidad de la información, los usuarios se están beneficiando también de la estandarización de las interfaces de usuario. La navegación "point-and-click" a través de documentos HTML (el "HyperText Markup Language" del World Wide Web) requiere muy poca capacitación, si es que se requiere.

Desde una perspectiva del desarrollador, un número de beneficios obligados son dirigidos a la migración hacia Internet. Compañías están reportando dramáticas ganancias en Retornos de Inversión y han mejorado la satisfacción de sus clientes. Como lo reportó Information Week, una persona en HBO puso una base de datos de recursos humanos en su intranet en sólo 7 días. Esta tarea aclaró un objetivo que había sido planeado para un proyecto de un año-hombre. No frecuentemente puede la tecnología provocar un avance tan dramático de productividad.

Los administradores de sistemas de información están también anticipando enormes ahorros en los costos de integración y mantenimiento de aplicaciones. La administración de aplicaciones centralizadas (con usuarios bajando archivos ejecutables vía la Internet, con base en sus necesidades específicas), permitirá revisiones más frecuentes y menos inoportunas.

El cambio a computación en Internet lleva consigo grandes promesas para el uso, desarrollo y administración de aplicaciones. ¿Qué papel jugarán las herramientas de desarrollo en esta arena ?

La Importancia de la Tecnología de Desarrollo en Internet :

La tecnología de herramientas de desarrollo para la Internet está en su infancia. La herramienta más popular para muchos desarrolladores es el procesador de palabras o editor de textos, lo cual es aceptable siempre y cuando el objetivo sea publicar páginas estáticas en el Web. Sin embargo, tal como los desarrollos cliente/servidor requieren herramientas sofisticadas para hacer programación más productiva, la entrega de contenido dinámico en el Web también requiere herramientas y técnicas sofisticadas. Los desarrolladores de Internet rápidamente se convencerán que elegir las herramientas de desarrollo apropiadas es crítico. Muchos asuntos que los desarrolladores tuvieron que enfrentar durante los inicios del desarrollo cliente/servidor están ocurriendo con la Internet, y entendiendo estos asuntos será mas fácil la transición a este nuevo modelo.

Un asunto es que las herramientas de desarrollo para Internet están aún sin probarse. Recordando el avance hacia cliente/servidor, montones de herramientas de desarrollo para Internet han sido anunciadas o están disponibles en versiones iniciales. Pero los desarrolladores requieren encontrar un ambiente completo que incluya inteligencia para la base de datos, componentes y facilidades de desarrollo en equipo. Afortunadamente, las herramientas requeridas para construir aplicaciones Web escalables y orientadas a datos, madurarán rapidamente porque estarán basadas en tecnologías largamente probadas para cliente/servidor.

Un segundo asunto respecto a las organizaciones de desarrollo es la dificultad en seleccionar una herramienta con viabilidad de largo plazo. Eventualmente, un puñado de herramientas aparecerá, y pudiera ser difícil de predecir cuál de ellas liderará el mercado y cuál simplemente desaparecerá. ¿Llegará a haber una acumulación de talento especializado en una herramienta en particular para soportar proyectos de misión crítica ? Los asuntos de maduración de productos y longevidad son extremadamente importantes, y sólo serán resueltos con el tiempo.

Mientras tanto, la administración de sistemas de información no debe andarse por las orillas. Los sistemas de información deben ir hacia adelante o arriesgarse a perder el reconocimiento que la computación en Internet promete.

La siguiente sección habla de la Internet como una extensión lógica de cliente/servidor y provee del marco de trabajo para el resto de este documento.

Cliente/Servidor y la Internet :

La computación en Internet descansa en la misma continuidad de cliente/servidor, con diferencias relativamente menores separándolos. El foco de la interfaz de usuario se ha movido de Windows a visualizadores o "browsers". Las redes se han estandarizado sobre el ya popular protocolo TCP/IP. La computación distribuida se ha expandido desde el simple dos-capas hasta servidores de aplicación y bases de datos distribuidas (tres-capas), y esta expansión continua hacia N-capas (N = cualquier número) con la adición de visualizadores y servidores de Web.

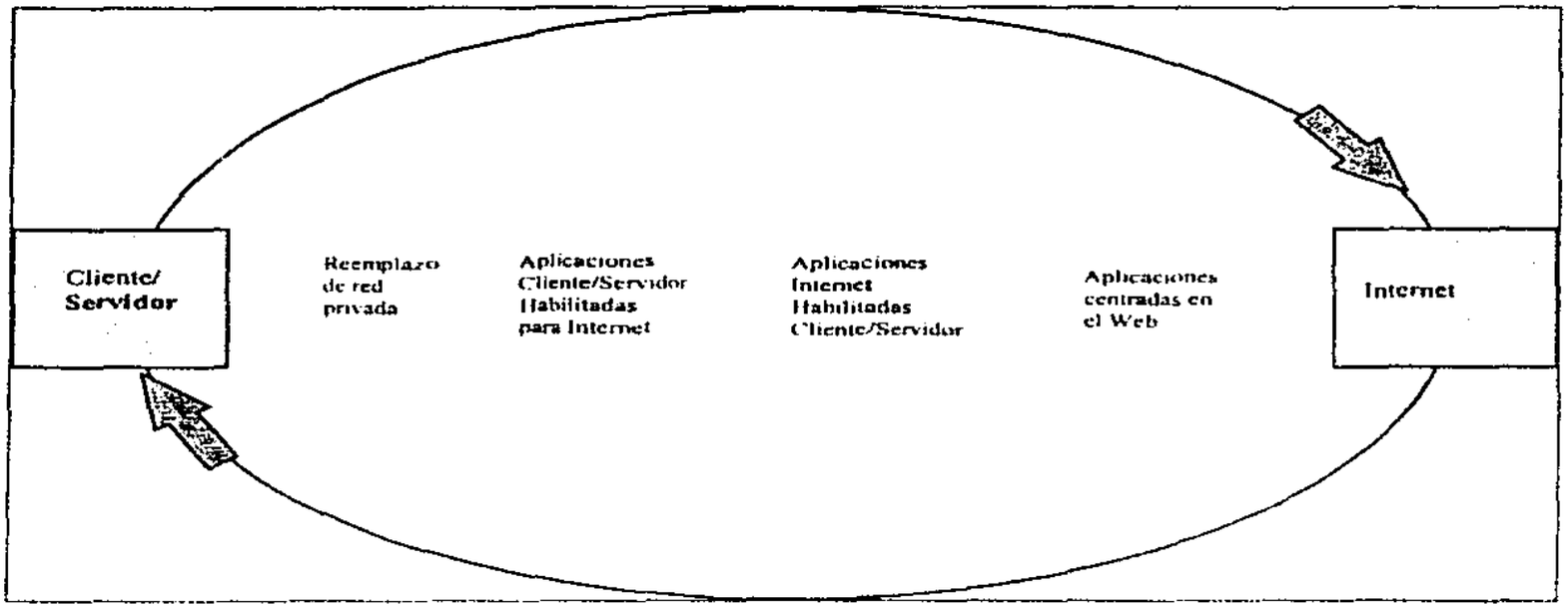


Figura 1.12. La computación cliente/servidor e Internet se basan en la misma continuidad

Muchos también creen que la computación cliente/servidor e Internet pueden coexistir y convertirse gradualmente en integrados. En una reciente reunión de 225 administradores de sistemas de información, 51% describió una arquitectura basada en Internet como "una extensión, paralela, o agregada a la tecnología cliente/servidor". Con el tiempo, las aplicaciones y herramientas cliente/servidor ganarán características que lo acercarán a la Internet, mientras que las tecnologías originales para Internet serán mejoradas con la funcionalidad cliente/servidor.

¿Qué significa esta liga entre cliente/servidor e Internet para los desarrolladores? Esto significa que las habilidades para cliente/servidor son extremadamente relevantes y valiosas en la transición del mercado a computación Internet. Los conocimientos en bases de datos, transacciones, concurrencia, seguridad, presentación de datos, interfaces de usuario, "middleware", y técnicas orientadas a objetos, es exactamente lo que se requiere para migrar Internet al siguiente nivel. Las habilidades y experiencia de desarrollo de aplicaciones sólidas ayudará a cruzar la barrera del uso predominante del Web en folletos electrónicos de mercadotecnia, hacia aplicaciones corporativas de negocios centradas en los datos y llenas de potencial.

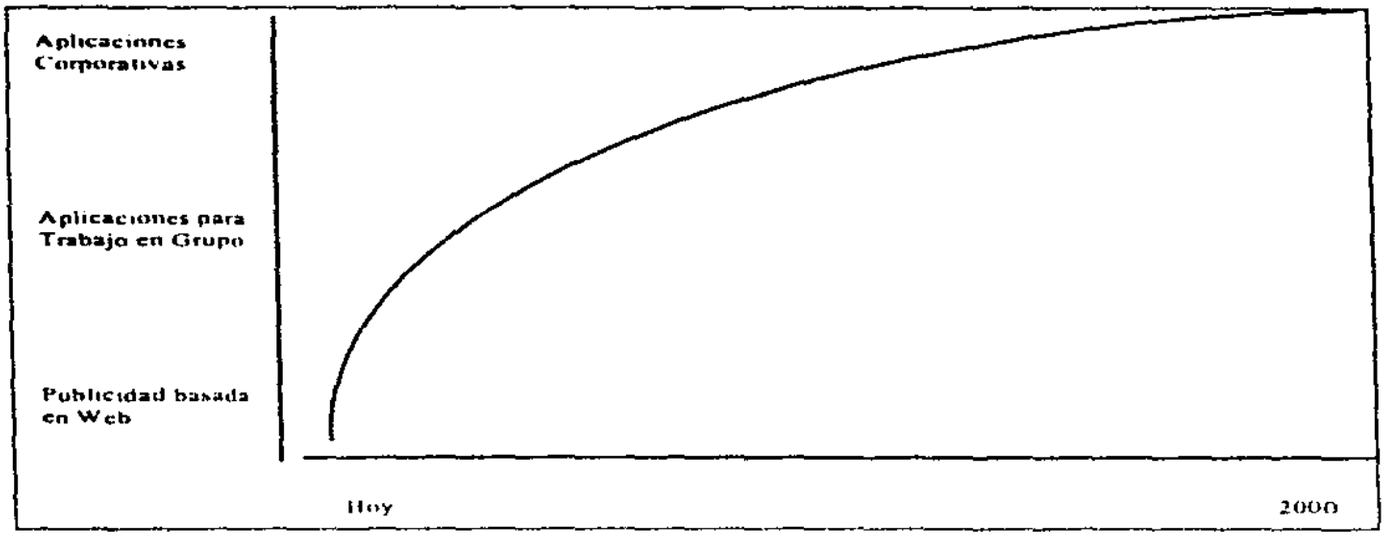


Figura 1.13. Los requerimientos de los usuarios y la ubicuidad de los visualizadores de Web en uso corporativo rapidamente empujarán al mercado, de un medio para publicar documentos, a una plataforma de trabajo en grupo y aplicaciones empresariales.

Desarrollo de Aplicaciones en Internet :

Como comentamos anteriormente, la Internet está aún madurando en el área de aplicaciones de negocios y tecnologías de desarrollo. El mercado de herramientas está muy fragmentado, y mientras muchas compañías lanzan sus productos como "la verdadera solución para Internet" para excitar a los capitalistas aventureros, la realidad es que un ambiente totalmente integrado de desarrollo para la Internet no existe todavía. Más importante aún, el mercado no ha dado su aval a ninguno de los ambientes específicos de desarrollo, dejando a los desarrolladores a la deriva. En estos términos, la respuesta es avanzar hacia cualquier estándar de facto que aparezca, como lo son HTML, Plug-Ins, Java, ActiveX Controls, etc.

A continuación abordamos el tema de desarrollo sobre Internet y cómo algunos fabricantes de software como Powersoft están permitiendo la creación de aplicaciones sofisticadas y orientadas a datos.

Aplicaciones cliente/servidor tradicionales habilitadas para Internet :

La inversión en aplicaciones cliente/servidor en los pasados años obligará a los administradores de desarrollo a mejorar sistemas existentes con capacidades para Internet en vez de reescribirlos completamente. Si recursos significativos han sido invertidos para crear e integrar una aplicación cliente/servidor de misión crítica, podría no tener sentido abandonar esta inversión para colocarla en un visualizador de Web. En vez de ésto, la adición de acceso a Internet para aplicaciones existentes puede ofrecer una mejor y más fácil entrega de la solución.

Por ejemplo, una compañía podría actualizar la aplicación de servicios al cliente para ganar acceso automático a la correspondencia para clientes codificada en GIF ó JPEG que resida en un servidor de Web interno. Combinado con acceso Intranet para documentos de políticas

corporativas y directorios de empleados, la adición de capacidades de visualizadores de Web para las aplicaciones existentes cliente/servidor mejora la productividad mientras que disminuye costos de impresión y distribución de documentos basados en papel.

WebViewer :

El WebViewer ActiveX (OLE) Control de Visual Components (la división de componentes de Powersoft) es una tecnología líder para activar en aplicaciones cliente/servidor capacidades de visualización y navegación en el Web. Con WebViewer, los eventos de los usuarios en una aplicación de PowerBuilder puede "disparar" la extracción de información basada en Web automáticamente, mejorando la productividad y utilidad.

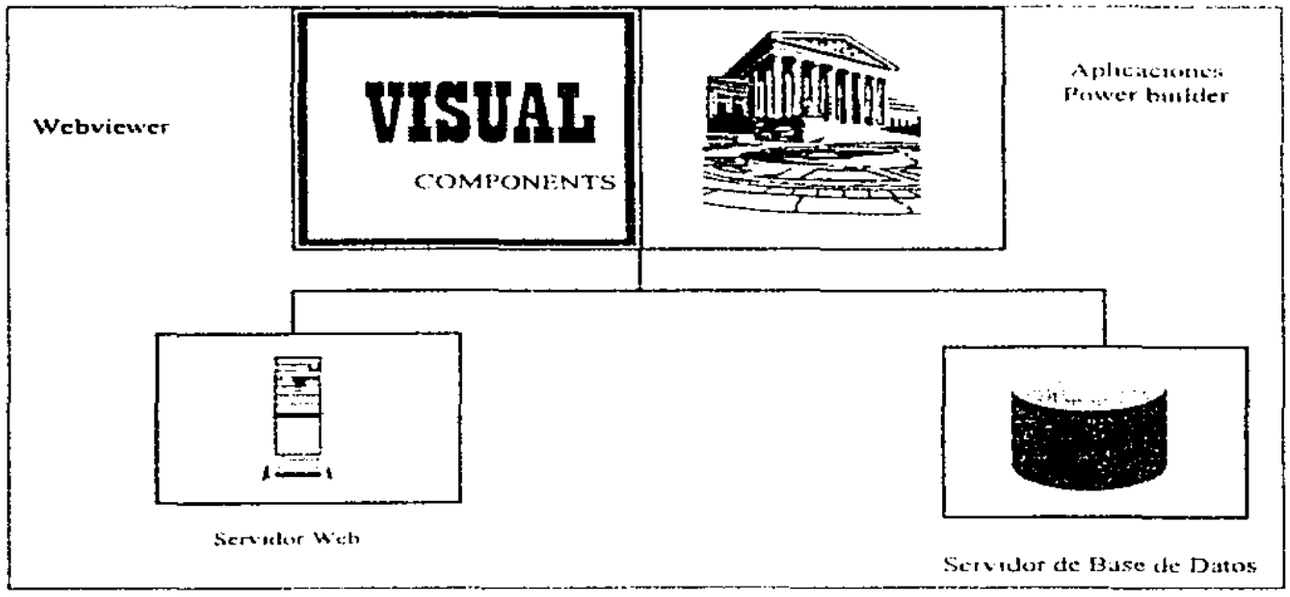


Fig. 1.13 : Aplicacion cliente/servidor escrita en PowerBuilder gana acceso al Web usando el Visual Components WebViewer OLE (Object Linking and Embeding) Control.

Adicionalmente al WebViewer control de Visual Components, los desarrolladores pueden utilizar objetos pre-construidos para transferencia de archivos (FTP, HTTP, y otros), conectando las aplicaciones cliente/servidor a servidores de Web y controlando el comportamiento de visualizadores estándares tales como Netscape Navigator y Microsoft Internet Explorer.

Agregar capacidades de Internet a aplicaciones cliente/servidor a través de WebViewer y otros objetos es una gran forma de extender el valor de las inversiones existentes. Adicionalmente, el poder de la Web para la obtención de información y manejo de documentos ha impulsado a la mayoría de las organizaciones a explorar o evaluar intranets ; muchos están encontrando que las intranet proveen más contenido dinámico de las bases de datos y de otras fuentes usando tecnología de visualizadores de Netscape, Microsoft y otros.

Enseguida nos referiremos a los retos de los grupos de sistemas de información proveyendo acceso al Web fácil de usar, a la vez que haciendo coincidir la funcionalidad e integridad de aplicaciones cliente/servidor.

Construyendo aplicaciones basadas en Visualizadores :

La publicación de documentos estáticos en el Web es fácil. La creación de aplicaciones de funcionalidad poderosa para correr en un ambiente de visualizador es otro asunto. Hay muchas formas de construir aplicaciones basadas en visualizador, tal como usar extensiones de visualizador preconstruidas (Off-the-shelf), desarrollando contenido ejecutable para correr en el visualizador, y desarrollando lógica de servidores de aplicaciones escalable. Powersoft proveer tecnología para cada uno de estos enfoques y puede ensamblar una solución que satisfaga las necesidades de la mayoría de los negocios.

Extensiones de Visualizador Preconstruidas (Off-the-shelf) :

Como un significativo crecimiento de la evolución de la programación orientada a objetos, los componentes reutilizables se están transformando rapidamente hacia la programación para el Web. Los componentes de visualizadores vienen hoy en día en dos formas : Netscape Plug-Ins, and Microsoft ActiveX Controls. Powersoft tienen varias tecnologías que incluyen estos estándares de facto, incluyendo la hoja de cálculo Formula One/Net, y el PowerBuilder DataWindows Viewer.

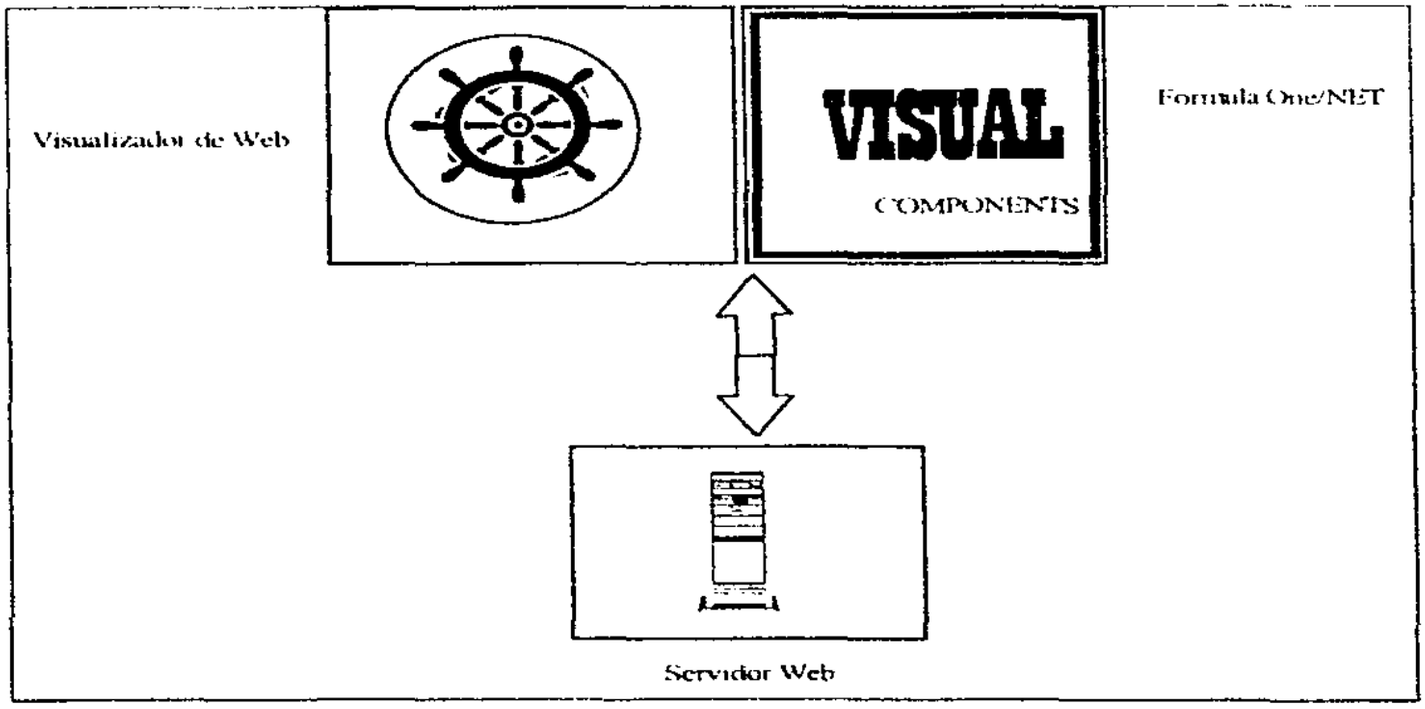


Fig. 1.14. Extensiones de Visualizadores Precontruidos - tal como el Plug-In para hojas de cálculo de Fórmula One/NET, acelera el desarrollo de aplicaciones para Web altamente funcional y visualmente sofisticada.

Fórmula One/NET

Muchas estrategias de desarrollo para el Web apalancarán inversiones en desarrollo de componentes para velocidad, tiempo de integración y ahorro en costos. Por ejemplo, si un banco desea ofrecer funcionalidad gráfica e interactiva para calcular el pago de préstamos, ellos podrían desarrollar hojas de cálculo para este propósito. Esto puede ser hecho incorporando el Plug-In de compatibilidad de Excel con Formula One/NET hecho por Powersoft's Visual Components Group.

DataWindow Viewer

La presentación de los "queries" de bases de datos y reportes sobre el Web es realizada tipicamente hoy en día a través de la generación de HTML. Para aplicaciones que requieren presentación más sofisticada de datos, Powersoft también provee de tecnología DataWindows con PowerBuilder y Optima++. El DataWindow Viewer, pensado para estar disponible en el futuro cercano como un ActiveX Control y como un Plug-In, permitirá el despliegue sofisticado de datos generados durante la ejecución de aplicaciones de PowerBuilder y Optima++.

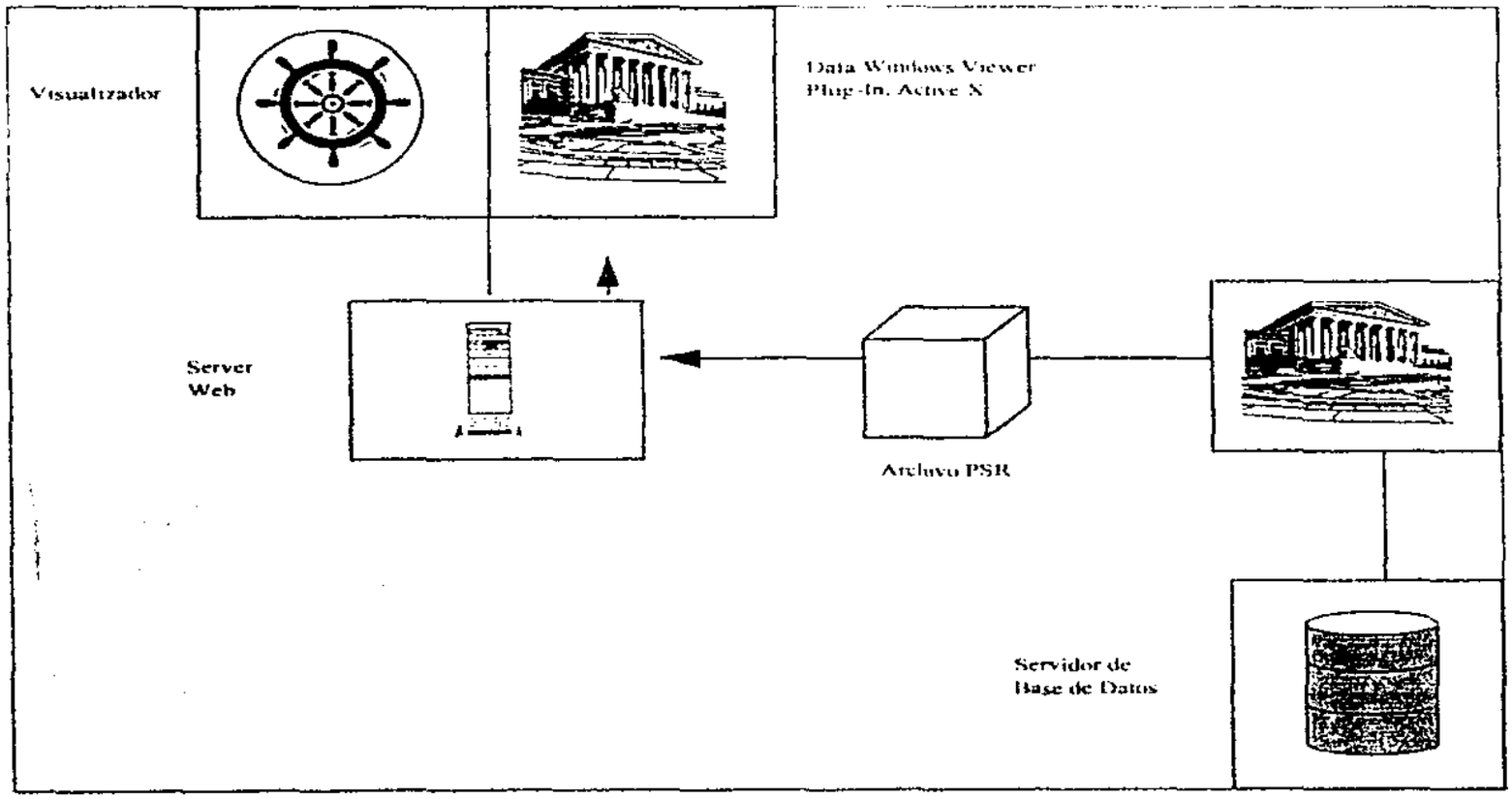


Fig. 1.15. El componente DataWindow Viewer de Powersoft habilita presentaciones de datos sofisticados para el Web.

El DataWindow Viewer lee un archivo PSR (Powersoft Report Format) el cual contiene tanto los datos como el formato de la presentación, y lo despliega en un visualizador. El visualizador de DataWindow Viewer soporta formatos de despliegue sofisticados tales como tabuladores-cruzados, mapas, gráficas, y formas de negocio (cross-tab, chart, graphics, business forms).

Desarrollo de Contenido Ejecutable para el Visualizador

Como los usuarios demandan aplicaciones para Internet más funcionales y robustas, los desarrolladores encontrarán cada vez más pesado programar en lenguajes simples y de HTML. Los ambientes de desarrollo altamente productivos serán introducidos, lo cual permitirá el desarrollo de controles de lógica de aplicaciones e interfaz de usuario. Estos incluirán generación de código robusto y herramientas para la construcción y prueba de aplicaciones para producción.

PowerBuilder Window Plug-In

PowerBuilder es el ambiente de desarrollo de aplicaciones de cuarta generación desarrollado por Powersoft y ampliamente reconocido en el mercado. Con la liberación de PowerBuilder 5.0, los desarrolladores tendrán un nuevo medio para agregar lógica al lado del cliente en las aplicaciones de negocios sobre Web. Dos mecanismos primarios son el PowerBuilder Plug-In, y la futura habilidad para generar controles ActiveX de PowerBuilder.

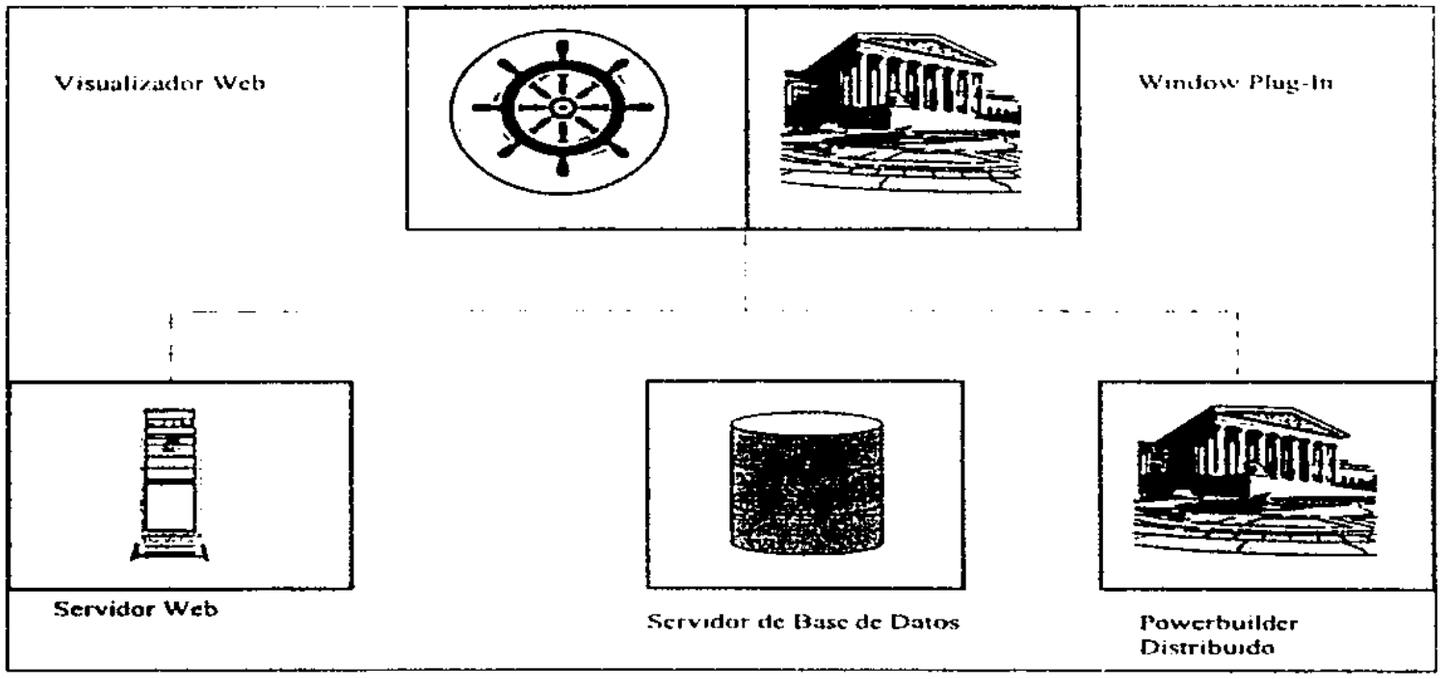


Fig 1.16. El PowerBuilder Windows Plug-In habilita una funcionalidad completa en aplicaciones cliente/servidor dentro de un visualizador.

Más funcional que el DataWindow Plug-In, el PowerBuilder Window Plug-In habilitará funcionalidad completa en aplicaciones cliente/servidor para correr dentro de un visualizador con capacidades de Plug-In.DataWindows, PowerScript (lenguaje para escribir aplicaciones robustas de Powersoft) y controles Windows son todos soportados. Adicionalmente, futuras versiones de PowerBuilder permitirán la generación de controles ActiveX para el uso en Internet en aplicaciones cliente/servidor.

Optima++

Quizás las más recomendada tecnología para crear lógica aplicativa en Internet es Java. Uno de los propósitos de Java es habilitar de inteligencia el procesamiento del lado del cliente y además de manejo de eventos para mejorar el desempeño y funcionalidad interactivos . Esto es realizado a través de Java "applets" - miniaplicaciones que son bajadas del Web server y ejecutadas en un visualizador-. Powersoft soporta desarrollo de Java applets a través de Optima++, el ambiente de desarrollo RAD C++ visual de Powersoft. Optima++ es un ambiente poderoso que combina la productividad y el poder de ensamblado de componentes RAD con el desempeño del C++, entregando capacidades para construir aplicaciones de tercera generación a los principales desarrolladores.

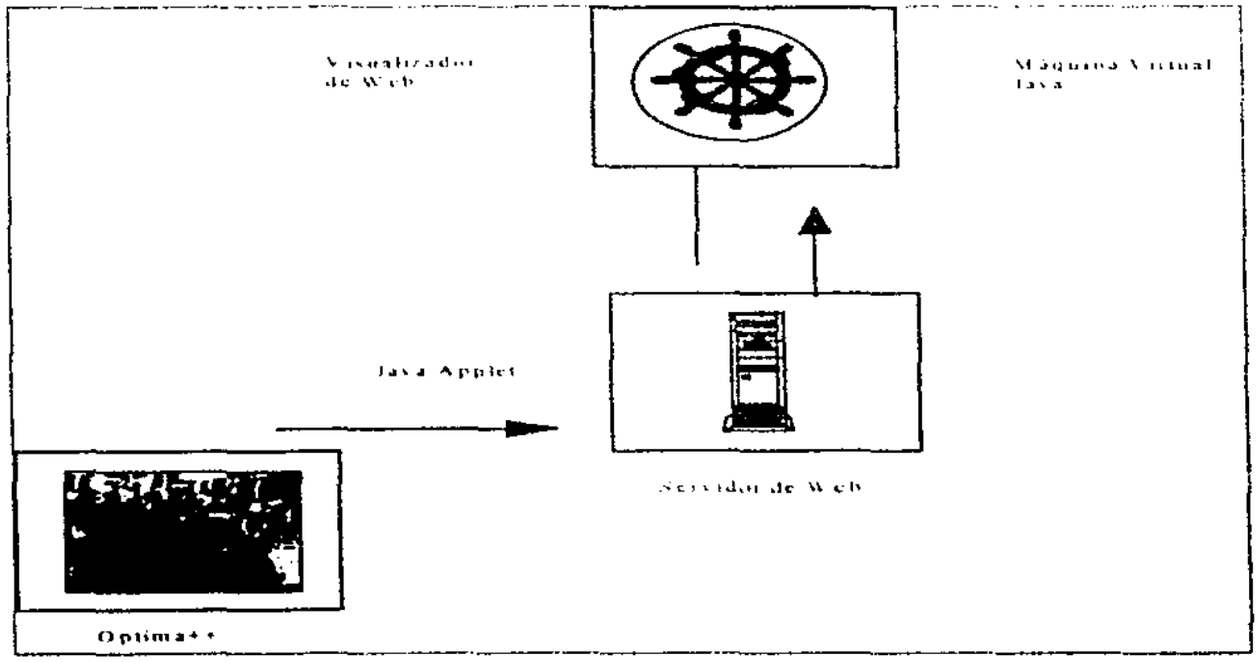


Fig. 1.17. Optima++ permitirá el desarrollo de Java applets.

En el futuro cercano , Optima++ soportará la creación de Java applets. Al mismo tiempo, Optima++ incluirá componentes que habilitará aplicaciones tradicionales cliente/servidor para Internet y simplificará la creación de servidores de aplicación tradicionales para Internet usando CGI, NSAPI, y ISAPI. Usando Optima++, tú serás capaz de crear tanto el lado cliente como el lado servidor de una aplicación Internet y administrarlos en el mismo proyecto.

Desarrollo de Servidores de Aplicaciones Escalables :

Además de la productividad del programador, la escalabilidad se convertirá en un asunto importante cuando las aplicaciones en producción - especialmente los sistemas internos de misión crítica- sean mirados al Web. Para facilitar la migración de estas aplicaciones de negocios, las compañías requerirán la habilidad de generar lógica de aplicación basada en servidores y compiladas. Como con el desarrollo de contenido para el visualizador, Powersoft ofrece tecnologías 3GL, 4GL y desarrollo de componentes. Powersoft ofrece PowerBuilder y Optima++ para el desarrollo de funcionalidad de servidores de aplicaciones Web.

Servidores de Aplicaciones PowerBuilder :

Con las capacidades distribuidas introducidas en PowerBuilder 5.0, los desarrolladores pueden crear servidores de aplicaciones tradicionales que distribuyen lógica y procesamiento a través de la red para una mayor escalabilidad. Las aplicaciones pueden regresar el resultado procesado del lado del Servidor al visualizador a través de CGI o el API nativo del servidor Web (p.e. NSAPI e ISAPI). Los resultados son regresados en una de dos formas : HTML, o un archivo recuperable y leíble vía un DataWindow Plug-In.

Web.PB:

Un nuevo componente, Web.PB, provee la conexión desde el servidor Web estándar a los objetos de aplicación de PowerBuilder (DPB). Con Web.PB, un visualizador de Web puede invocar funciones DPB que se ejecutarán detrás del servidor Web y el "firewall". Powerscript, el lenguaje de scripts de cuarta generación de Powersoft, permite incrustar sintaxis HTML y regresa los resultados procesados a través del scòvidor Wec para presentarlos en el visualizador.

Además de incrustar HTML en PowerScript, PowerBuilder 5.0 incluye soporte para la generación de archivos HTML estándar desde un DataWindow (o desde un DataWindow no visual conocido como un DataStore). La salida de DataWindow automaticamente generá tablas HTML, las cuales son salvadas en un archivo y puestas a disposición a través de un visualizador Web. Como se mencionó anteriormente el PowerBuilder DataWindow Plug-In o el ActéveX Control leerá un archivo PSR generado por un DataWindow o un DataStore no visual y desplegará fuentâs sofisticadas, mapas y gráficas.

Servidor de Aplicaciones Optima++ :

Optima++ puede ser usado para crear tanto servidores de aplicaciones EXE como DLL que iiteractuen direcôamente con servidores Web. En un futuro cercano, Optima++ incluirá componentes äe serviäores de aplicacéones los cuales ocultarán los detalles de los diferentes APIs de servidores Web y también permitirán construir servidores de aplicaciones distribõidas. Similar a las capácidades distribuidas introducidas en PowerBuilder 5.0, Optima++ puede crear servidores de aplicaciones tradicionales que distribuyan lógica y procesamiento a través de la red para una mayor escalabilidad. Los resultados de procesamiento del lado del servidor pueden ser regresados al vésualizador a través de CGI del ÁPI nativo del servidor Web (p.e. NSAPI e ISAPI).

Adicionalmente, Optima++ contiene la tecnología DataWindow creada por PowerBuilder. Esto permite a Optima++ tener el mismo "Salvar como HTML" y soporte disponible de DataWindow Plug-In/ActiveX para servidores de aplicación PowerBuilder. Juntos, PowerBuilder

y Optima++ proveen un rango completo de funcionalidad y desempeño para construir servidores de aplicación escalables Web.

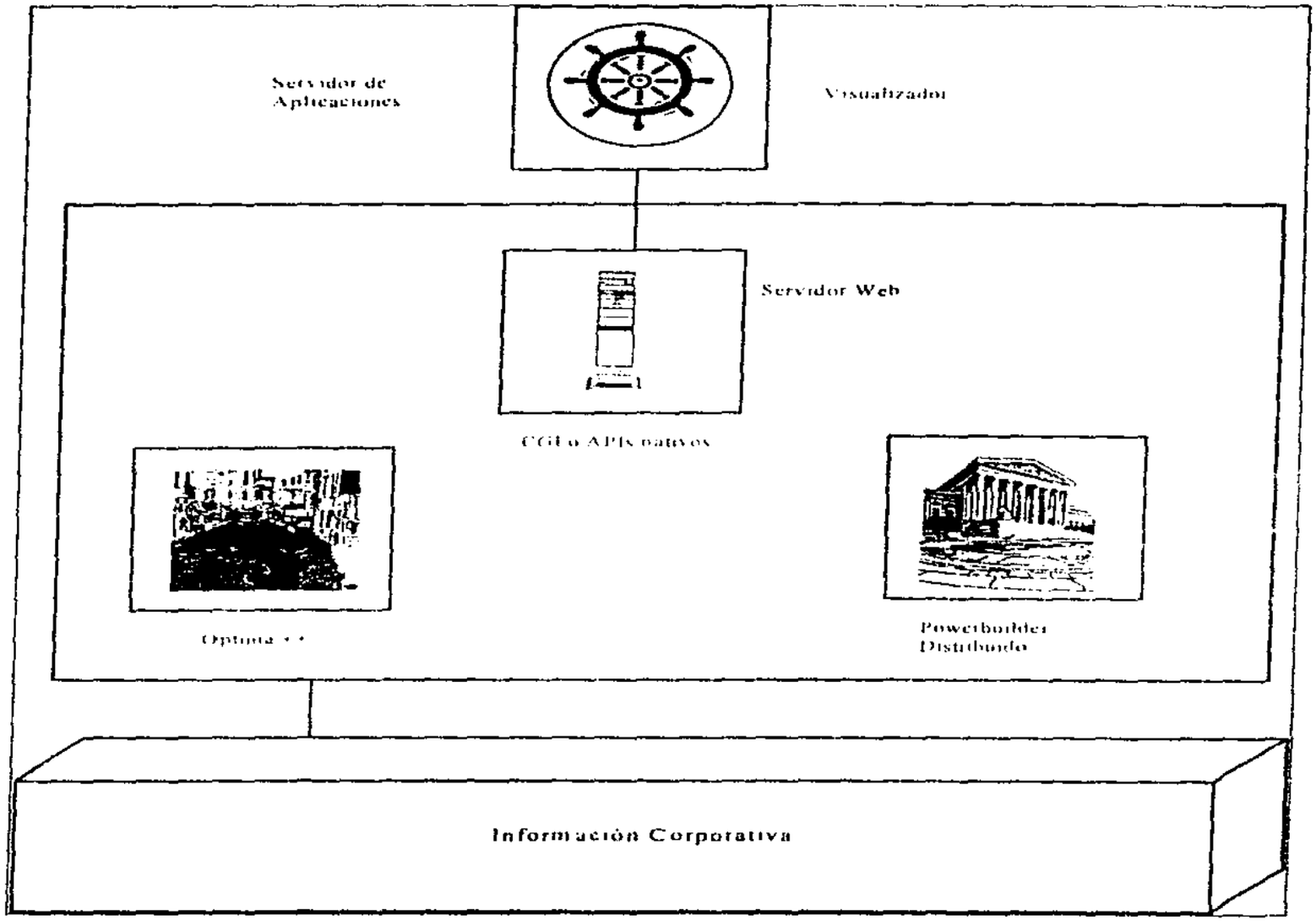


Fig. 1.18. Oôtima++ ù PowerBuilder Distribuido soportan el desarrollo de servidores de aplicación escalables.

Adminkstración de Fuentes de Páginas Web :

Después de que una organización elige las tecnologías que permitirán el desarrollo de aplicaciones escalables para la Web, la atención puede ser turnada a ajustar el proceso de desarrollo. Como la mayoría de loó desarrolladores en una organizãción se ven envueltos en un desarrollo de Internet, hay una gran necesidad de control de fuentes, versiones, y capacidades de trabajo en grupo.

ObjectCycle

ObjectCycle de Powersoft, incluido con PowerBuilder 5.0 edición Entârprise, permite desarrollo en equipo para aplicaciones Web manejando pâçinas HTML, objetos gráficos (GIF, JPEG), applets, y otrís componentes. ObjectCycle maneèa un chequeo de entrada y otro de salida para objetos, control de versión de objetos, y usos de almacenamiento en RDBMS de objetos para alto desempeño y robustos.

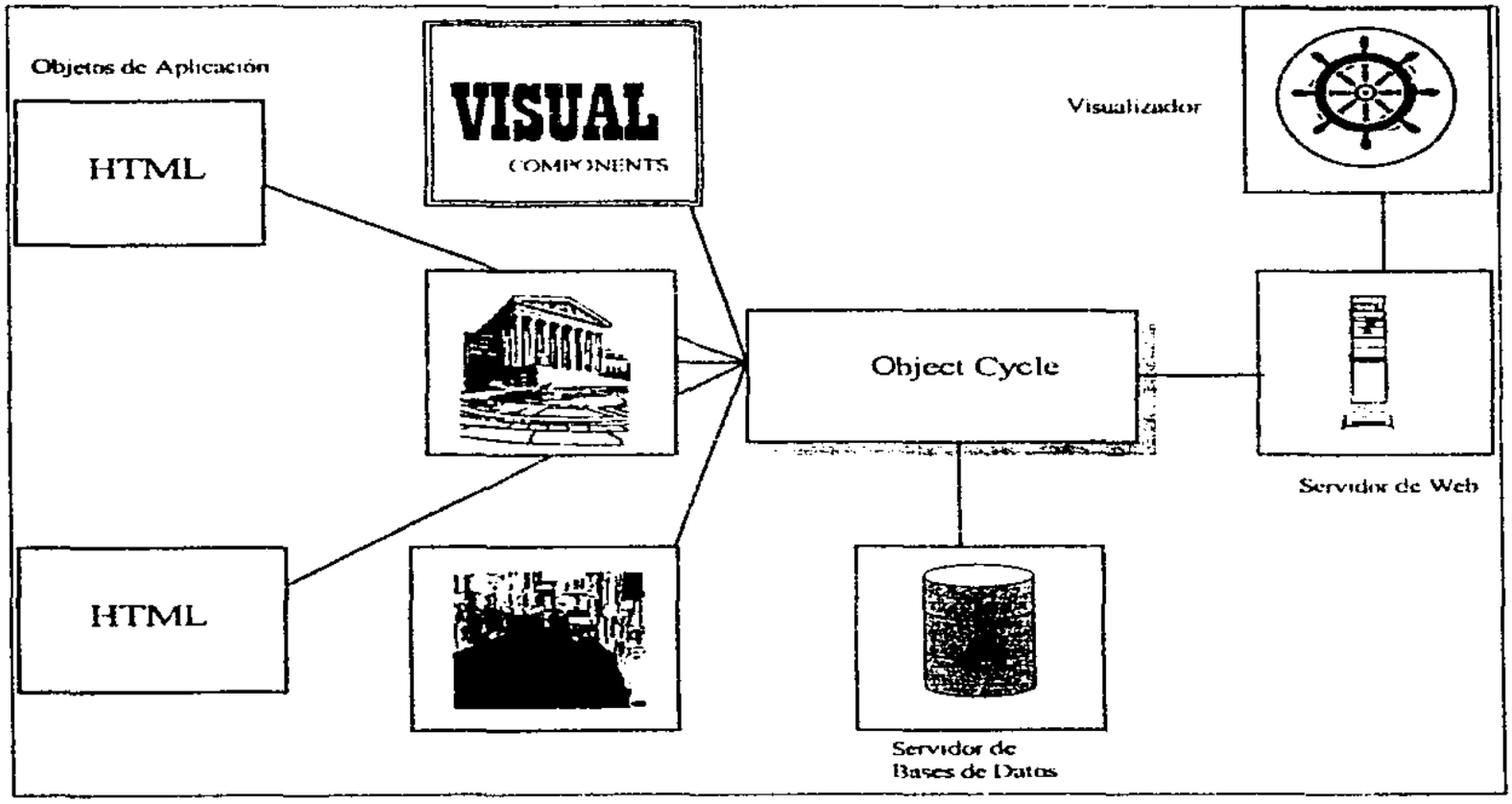


Fig. 1.10. ObjectCycle es un administrador universal de objetos para desarrollo en equipo de aplicaciones Web.

Finalmente, Powersoft está comprometido hacer por Internet lo que hizo por cliente/servidor, lo cual fue para levantar el nivel de abstracción los desarrolladores construyendo aplicaciones de negocio, resultando en productividad incrementada. Comparado con PowerSoft, ninguna otra compañía tiene la amplitud de tecnologías y la profundidad de experiencia en construir aplicaciones de negocio escalables. Como la Internet continúa madurando como una plataforma de computación de negocios, Powersoft continuará su liderazgo proveyendo una amplia gama de soluciones de desarrollo para construir aplicaciones serias de negocio.

2. INGENIERIA DE SOFTWARE Y METODOLOGIAS.

2.1 El Software

Hace veinte años, menos del 1 por 100 de la gente podría describir de forma inteligente lo que significaba el "software de computadora". Hoy, la mayoría de los profesionales y muchas personas en general creen que entienden el software. Pero la verdad es que ésto es muy cuestionable.

Una descripción del software de un libro de texto puede tener la siguiente forma: "Software: (1) instrucciones (programas de computadora) que cuando se ejecutan proporcionan la función y el comportamiento deseado. (2) estructuras de datos que facilitan a los programas manipular adecuadamente la información, y (3) documentos que describen la operación y el uso de los programas". No hay duda de que podrían ofrecerse otras definiciones más completas, pero nosotros necesitamos algo más que una definición formal.

Características del software

Para poder comprender lo que es el software (y consecuentemente la ingeniería del software), es importante examinar las características del software que lo diferencian de otras cosas que los hombres pueden construir. Cuando se construye hardware, el proceso creativo humano (análisis, diseño, construcción, prueba) se traduce finalmente en una forma física. Si construimos una nueva computadora, nuestro boceto inicial, diagramas formales de diseño y prototipo de prueba, evolucionan hacia un producto.

El software es un elemento del sistema que es lógico, en lugar de físico. Por tanto, el software tiene unas características considerablemente distintas a las del hardware.

El software se desarrolla, no se fabrica en un sentido clásico.

Aunque existen algunas similitudes entre el desarrollo de software y la construcción del hardware, ambas actividades son fundamentalmente diferentes. En ambas actividades la buena calidad se adquiere mediante un buen diseño, pero la fase de construcción del hardware puede introducir problemas de calidad que no existen (o son facilmente corregibles) en el software. Ambas actividades dependen de las personas, pero la relación entre la gente dedicada y el trabajo realizado es completamente diferente para el software. Ambas actividades requieren la construcción de un "producto", pero los métodos son diferentes.

Los costos del software se encuentran en la ingeniería. Esto significa que los proyectos de software no se pueden gestionar como si fueran proyectos de fabricación.
En la pasada década se ha trabajado en la literatura el concepto de "fábrica de software". Es importante tener en cuenta que este término no implica que la fabricación del hardware y el desarrollo del software sean equivalentes. En vez de ello, el concepto de fábrica de software recomienda el uso de herramientas para el desarrollo automático del software (véase la parte).

El software no se "estropea".

La Fig. 2.1 describe, para el hardware, la proporción de fallos como una función del tiempo. Esa relación, denominada frecuentemente "curva de bañera", indica que el hardware exhibe relativamente muchas fallas al principio de su vida (estos fallos son atribuibles normalmente a defectos del diseño o de la fabricación); una vez corregidos los defectos, la tasa de fallos cae hasta un nivel estacionario (bastante bajo, con un poco de optimismo) donde permanece durante un cierto período de tiempo. Sin embargo, conforme pasa el tiempo, los fallos vuelven a presentarse a medida que los componentes de hardware sufren los efectos acumulativos de la sociedad, la vibración, los malos tratos, las temperaturas extremas y muchos otros males externos. Sencillamente, el hardware comienza a estropearse.

El software no es susceptible a los males del entorno que hace que el hardware se estropee. Por tanto, en teoría, la curva de fallos para el software tendría la forma que muestra la Fig. 2.2 Los defectos no detectados harán que falle el programa durante las primeras etapas de su vida. Sin embargo, una vez que se corrigen, suponiendo que no se introducen nuevos errores, la curva se aplana, como muestra la figura. Esa figura es una gran simplificación de los modelos reales de fallos de software. Sin embargo, la implicación es clara: "el software no se estropea, pero se deteriora".

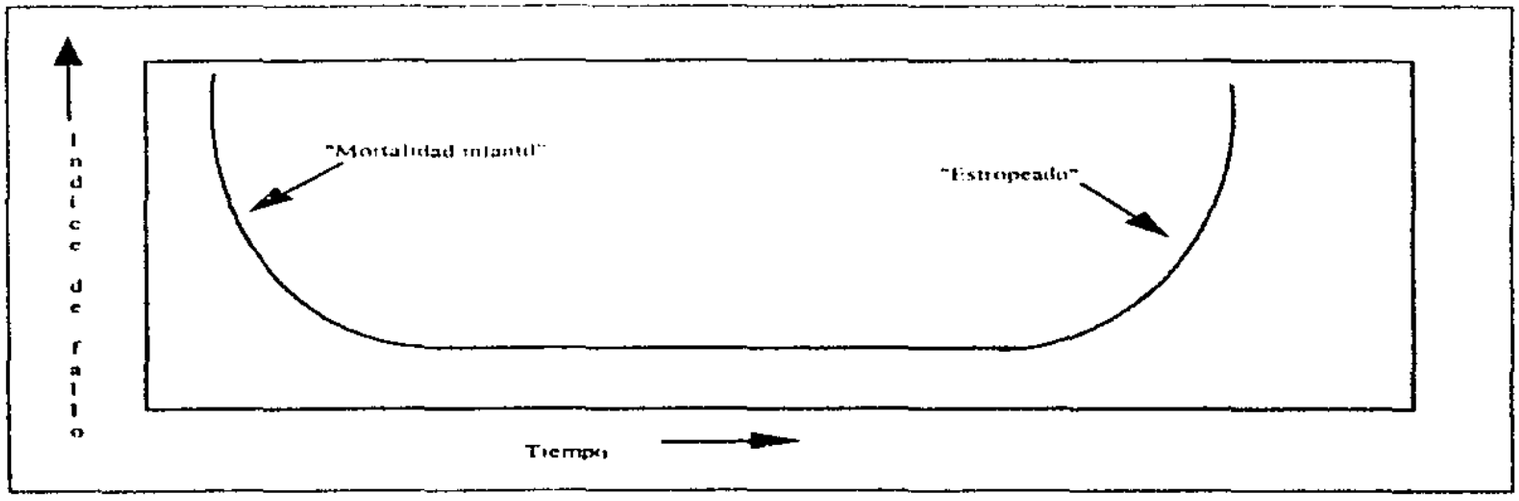


Fig. 2.1. Curva de fallos del hardware.

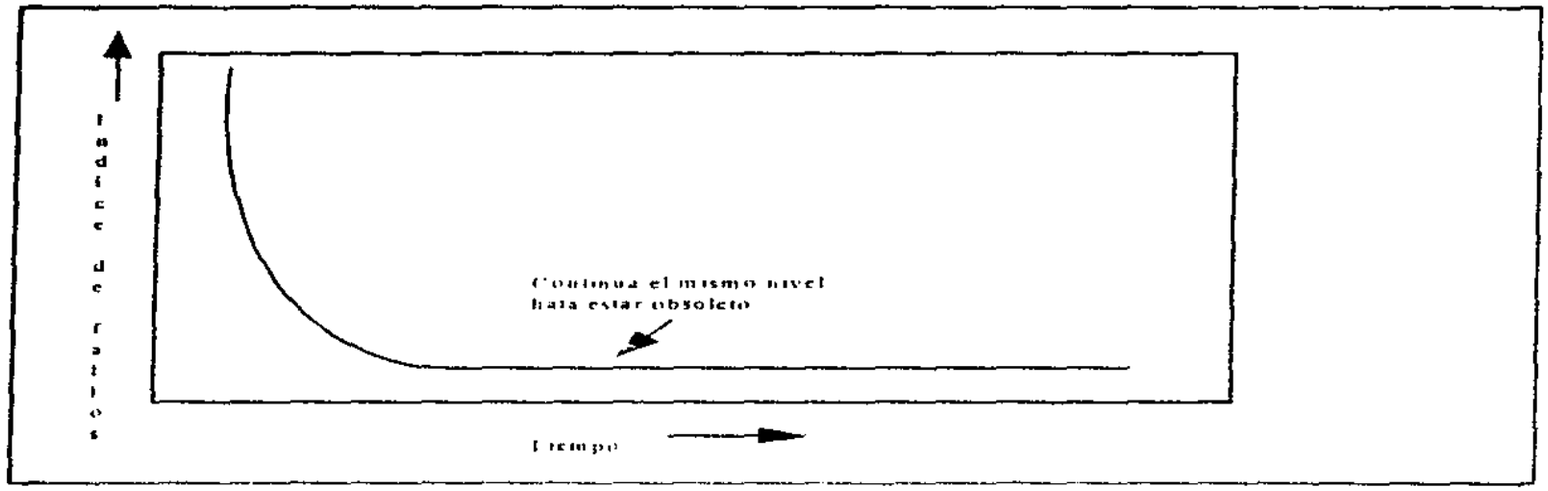


Fig. 2.2 Curva de fallos del software (idealizada).

Esto, que parece una contradicción, puede comprenderse mejor considerando la Fig. 2.3. Durante su vida, el software sufre cambios (mantenimiento). Conforme se hacen los cambios, es bastante probable que se introduzcan nuevos defectos, haciendo que la curva de fallos tenga picos como se ve en la Fig. 2.3. Antes de que la curva pueda volver al estado estacionario original, se solicita otro cambio, haciendo que de nuevo se cree otro pico. Lentamente, el nivel mínimo de fallos comienza a crecer; el software se estropea, se sustituye por una "pieza de repuesto". No hay piezas de repuesto para el software. Cada fallo en el software indica un error en el diseño o en el proceso mediante el que se tradujo el diseño a código máquina ejecutable. Por tanto, el mantenimiento del software tiene una complejidad considerablemente mayor que la del mantenimiento del hardware.

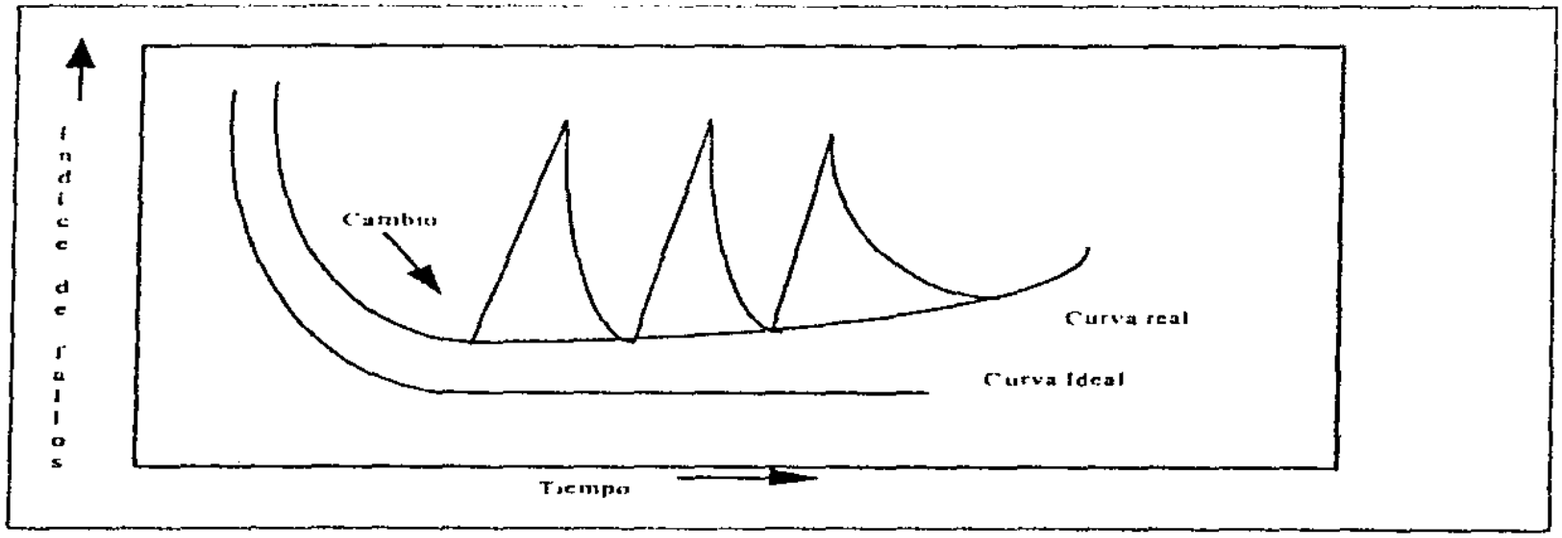


Fig. 2.3. Curva real de fallos del software

La mayoría del software se construye a medida, en vez de ensamblar componentes existentes.

Consideremos la forma en la que se diseña y se construye el hardware de control para un producto basado en microprocesador. El ingeniero de diseño construye un sencillo esquema de la circuitería digital, hace algún análisis fundamental para asegurar que se realiza la función adecuada y va al catálogo de ventas de componentes digitales existentes. Cada circuito integrado (frecuentemente llamado un "CI" o "pastilla") tiene un número de pieza, una función definida y válida, una interfaz bien definida y un conjunto estándar de criterios de integración. Después de seleccionar cada componente, puede solicitarse la compra.

Por desgracia, los diseñadores del software no disponen de esa comodidad que acabamos de describir. Con unas pocas excepciones, no existen catálogos de componentes de software. Se puede comprar software ya desarrollado, pero sólo como una unidad completa, no como componentes que puedan reensamblarse en nuevos programas. Aunque se ha escrito mucho sobre "reusabilidad del software", sólo estamos comenzando a ver las primeras implementaciones con éxito de este concepto.

Componentes de software

El software de computadora es información que existe en dos formas básicas: componentes no ejecutables en la máquina y componentes ejecutables en la máquina. Por lo que sólo contemplaremos los componentes de software que conducen directamente a instrucciones máquina ejecutables.

Los componentes de software se crean mediante una serie de traducciones que hacen corresponder los requisitos del cliente con un código ejecutable en la máquina. Se traduce un modelo (prototipo) de requisitos a un diseño. Se traduce el diseño del software a una forma en un lenguaje que especifica las estructuras de datos, los atributos procedimentales y los requisitos que atañen al software. La forma en lenguaje es procesada por un traductor que la convierte en instrucciones ejecutables en la máquina.

La reusabilidad es una característica importante para un componente de software de alta calidad. Es decir, el componente debe diseñarse e implementarse para que pueda volver a usarse en muchos programas diferentes. En los años sesenta se construyeron bibliotecas de subrutinas científicas reutilizables en una amplia serie de aplicaciones científicas y de ingeniería. Esas bibliotecas de subrutinas reutilizaban de forma efectiva algoritmos bien definidos, pero tenían un dominio de aplicación limitado. Hoy en días, hemos extendido nuestra visión de reusabilidad para abarcar no sólo los algoritmos, sino también tanto datos como procesos en un único paquete (frecuentemente llamado clase u objeto), permitiendo al ingeniero de software crear nuevas aplicaciones a partir de trozos reusables. Por ejemplo, las interfaces interactivas de hoy en día se construyen frecuentemente a partir de componentes reusables que permiten la creación de ventas gráficas, de menús emergentes y de una amplia variedad de mecanismos de interacción. Las estructuras de datos y los detalles de procesamiento requeridos para construir la interfaz están contenidos en una biblioteca de componentes reusables orientados a la construcción de interfaces. Los componentes de software se construyen mediante un lenguaje de programación que tiene un vocabulario limitado, una gramática definida explícitamente y reglas bien formadas de sintaxis y semántica. Estos atributos son esenciales para la traducción por la máquina. Las clases de lenguajes que se utilizan actualmente son los lenguajes máquina, los lenguajes de alto nivel y los lenguajes no procedimentales.

Los lenguajes máquina son una representación simbólica del conjunto de instrucciones de la CPU. Si un buen programador produce programas mantenibles y bien documentados, puede utilizar el lenguaje máquina para hacer un uso extremadamente eficiente de la memoria y para "optimizar" la velocidad de ejecución del programa. Si el programa está mal diseñado y tiene poca documentación, el lenguaje máquina puede convertirse en una pesadilla.

Aunque hoy se utilizan cientos de lenguajes de programación, poco más de una decena son lenguajes de programación de alto nivel con una gran aceptación de la industria. Después de casi treinta años desde su aparición, lenguajes como COBOL y FORTRAN siguen utilizándose mucho en la actualidad. Los lenguajes de programación modernos (lenguajes que soportan directamente las prácticas modernas para el diseño procedimental de datos) tales como Pascal, C y Ada se utilizan ampliamente. Los lenguajes orientados a los objetivos como C++, Object Pascal, Eiffel y otros, están ganando cada vez más seguidores.

Los lenguajes especializados (diseñados para ámbitos de aplicación específicos), como APL, LISP, OPS5 y lenguajes descriptivos para redes neuronales artificiales, están teniendo cada vez mayor aceptación conforme las nuevas aplicaciones pasan de los laboratorios a la práctica.

El código máquina, los lenguajes ensambladores (nivel máquina) y los lenguajes de programación de alto nivel son normalmente considerados como las "tres primeras generaciones" de lenguajes de computadora. Con cualquiera de esos lenguajes, el programador ha de preocuparse tanto de la especificación de la estructura de la información como de la de control del propio programa. Por ello, los lenguajes de las tres primeras generaciones se denominan Lenguajes procedimentales.

En la década pasada, apareció un grupo de lenguajes no procedimentales o de cuarta generación. En vez de requerir que quien desarrolla el software especifique los detalles procedimentales, un programa en un lenguaje no procedimental es la "especificación del resultado deseado, en vez de la especificación de la acción requerida para conseguir el resultado". El software de soporte traduce la especificación del resultado en un programa máquina ejecutable. Hasta la fecha, los lenguajes de cuarta generación se han utilizando en aplicaciones de bases de datos y en otras áreas de procesamiento de datos para negocios.

Aplicaciones del Software.

El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales (es decir, un algoritmo). Excepciones notables a esta regla son el software de los sistemas expertos y de redes neuronales. Para determinar la naturaleza de una aplicación de software, hay dos factores importantes que se deben considerar: el contenido y el determinismo de la información. El contenido se refiere al significado y a la forma de la información de entrada y de salida. Por ejemplo, muchas aplicaciones bancarias usan unos datos de entrada muy estructurados (una base de datos) y producen "informes" con determinados formatos. El software que controla una máquina automática (por ejemplo, un control numérico) actúa sobre elementos de datos discretos con una estructura limitada y produce órdenes concretas para la máquina en rápida sucesión.

El determinismo de la información se refiere a la predecibilidad del orden y del tiempo de llegada de los datos. Un programa de ingeniería acepta datos que están en un orden predefinido, ejecuta el algoritmo sin interrupción por condiciones externas y produce una salida que depende de una función del entorno y del tiempo. Las aplicaciones con estas características se dice que son indeterminadas.

Algunas veces es difícil establecer categorías genéricas para las aplicaciones del software que sean significativas. Conforme aumenta la complejidad del software, es más difícil establecer compartimentos nitidamente separados. Las siguientes áreas del software indican la amplitud de las posibilidades de aplicación.

Software de sistemas. El software de sistemas es un conjunto de programas que han sido escritos para servir a otros programas. Algunos programas de sistemas (p. ej.: ciertos componentes del sistema operativo, utilerías de manejo de periféricos, procesadores de telecomunicaciones) procesan datos en gran medida indeterminados. En cualquier caso, el área del software de sistemas se caracteriza por una fuerte interacción con el hardware de la computadora; una gran utilización por múltiples usuarios; una operación concurrente que requiere una planificación, una compartición de recursos y una sofisticada gestión de procesos; unas estructuras de datos complejas y múltiples interfaces externas.

Software de tiempo real. El software que mide/analiza/controla sucesos del mundo real conforme ocurren, se denomina de tiempo real. Entre los elementos del software de tiempo real se incluyen: un componente de adquisición de datos que recolecta y da formato a la información recibida del entorno externo, un componente de análisis que transforma la información según lo requiera la aplicación, un componente de control/salida que responda al entorno externo y un componente de monitoreo que coordina todos los demás componentes, de forma que pueda mantenerse la respuesta en tiempo real (típicamente en el rango de 1 milisegundo a 1 minuto). Hay que tener en cuenta que el término "tiempo real " tiene un significado diferente de "interactivo" o "tiempo compartido". Un sistema de tiempo real debe responder dentro de unas ligaduras estrictas de tiempo. El tiempo de respuesta de un sistema interactivo (o de tiempo compartido) puede ser normalmente sobrepasado sin que se produzca ningún desastre.

Software de gestión. El procesamiento de información comercial constituye la mayor de las áreas de aplicación del software. Los "sistemas discretos" (por ejemplo: nóminas, cuentas pagar/cobrar, inventarios, etc.) han evolucionado hacia el software de sistemas de información de gestión (SIG), que accede a una o más bases de datos grandes que contienen información comercial. Las aplicaciones en esta área reestructuran los datos existentes en orden a facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamientos de datos, las aplicaciones de software de gestión también realizan cálculo interactivo (por ejemplo: el procesamiento de transacciones en punto de ventas).

Software de ingeniería y científico. El software de ingeniería y científico está caracterizado por los algoritmos de "manejo de números." Las aplicaciones van desde la astronomía a la vulcanología, desde el análisis de la presión de los automotores a la dinámica orbital de las lanzaderas espaciales y desde la biología molecular a la fabricación automática. Sin embargo, las nuevas aplicaciones del área de ingeniería/ciencia se han alejado de los algoritmos convencionales numéricos. El diseño asistido por computadora (de inglés, CAD), la simulación de sistemas y otras aplicaciones interactivas, han comenzado a tomar características del software de tiempo real e incluso del software de sistemas.

Software empotrado. Los productos inteligentes se han convertido en algo común en casi todos los mercados de consumo e industriales. El software empotrado reside en memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El

software empotrado puede ejecutar funciones muy limitadas y curiosas (por ejemplo: el control de las teclas de un horno de microondas) o suministrar una función significativa y capacidad de control (por ejemplo: funciones digitales en un automóvil, tales como control de la gasolina, sistemas de frenado, etc.)

Software de computadoras personales. El mercado del software de las computadoras personales ha germinado en la pasada década. El procesamiento de textos, las hojas de cálculo, los gráficos por computadora, entretenimientos, gestión de bases de datos, aplicaciones financieras, de negocios y personales y redes o acceso a bases de datos externas son sólo algunas de los cientos de aplicaciones. De hecho, el software de las computadoras personales continúa representando uno de los diseños del software más innovadores en el campo del software.

Software de inteligencia artificial. El software de inteligencia artificial (IA) hace uso de algoritmos no numéricos para resolver problemas complejos para los que no son adecuados al cálculo o el análisis directo. Actualmente, el área más activa de la IA es la de los sistemas expertos, también llamadas sistemas basados en el conocimiento. Otras áreas de aplicación para el software de IA son los juegos. En los últimos años ha surgido una nueva rama de software de IA llamada redes neuronales artificiales . Una red neuronal simula la estructura de proceso del cerebro (las funciones de la neurona biológica) y a la larga puede llevar a una clase de software que pueda reconocer patrones complejos y aprender de "experiencia" pasada.

2.2 Paradigmas de la Ingeniería del Software

Los males en el desarrollo del software no van a desaparecer de la noche a la mañana. Reconocer los problemas y sus causas y demoler los mitos del software son los primeros pasos hacia las soluciones. Pero las propias soluciones tienen que proporcionar asistencia práctica a la persona que desarrolla software, mejorar la calidad del software y, por último, permitir al mundo del software mantenerse en paz con el mundo del hardware.

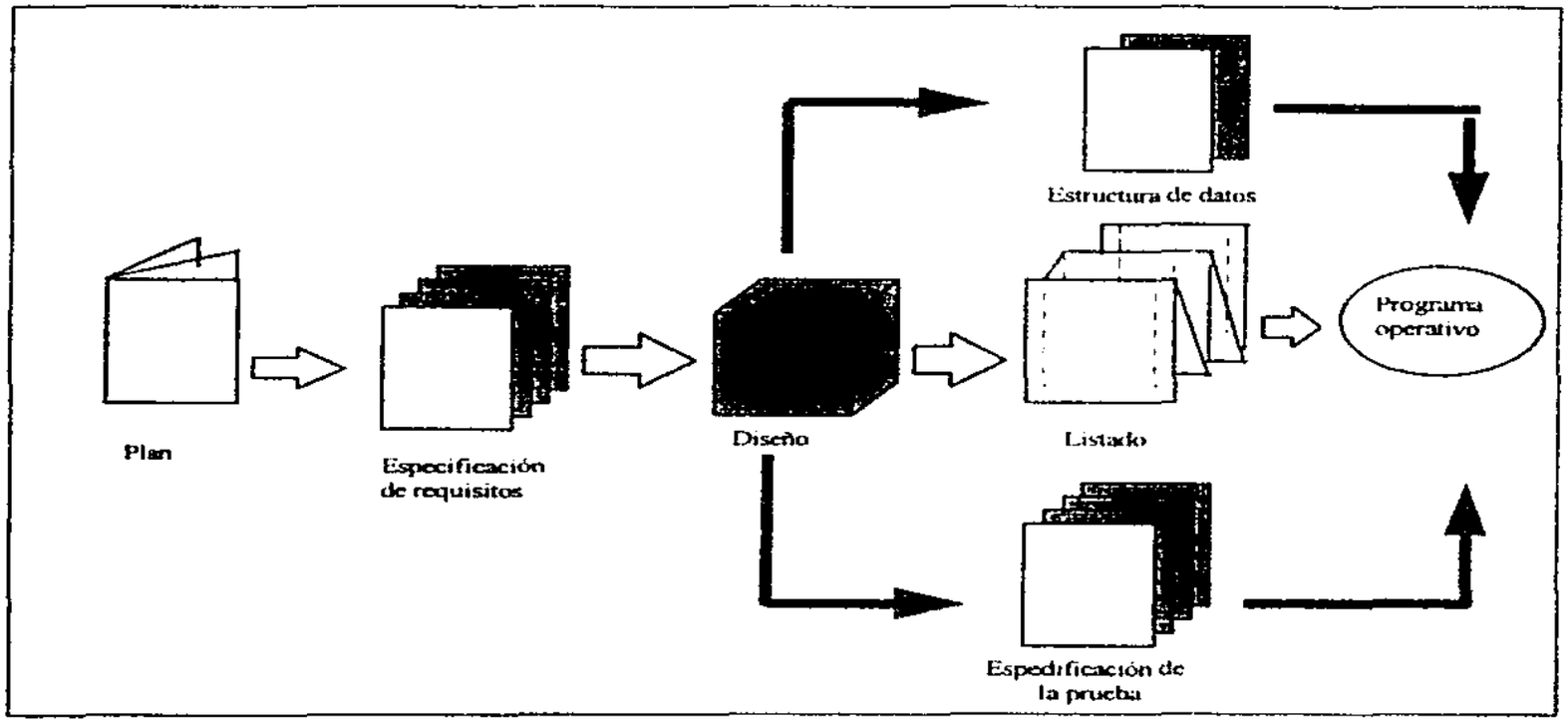


Fig. 2.4. La configuración del software

No existe un único enfoque mejor para solucionar el mal del software. Sin embargo, mediante la combinación de métodos completos para todas las fases del desarrollo del software, mejores herramientas para automatizar estos métodos, bloques de construcción más potentes para la implementación del software, mejores técnicas para la garantía de calidad del software y una filosofía predominante para la coordinación, control y gestión, podemos conseguir una disciplina para el desarrollo del software "una disciplina llamada ingeniería del software".

Ingeniería del software: una definición

Una de las primeras definiciones de ingeniería del software fue la propuesta por Fritz Bauer en la primera conferencia importante dedicada al tema: El establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales.

Aunque se han propuesto muchas más definiciones generales, todas refuerzan la importancia de una disciplina de ingeniería para el desarrollo del software. La ingeniería del software surge de la ingeniería de sistemas y de hardware. Abarca un conjunto de tres elementos clave - métodos, herramientas y procedimientos - que facilitan al gestor controlar el proceso del desarrollo del software y suministrar a los que practiquen dicha ingeniería las bases para construir software de alta calidad de una forma productiva.

Los **métodos** de la ingeniería del software indican "cómo" construir técnicamente el software. Los métodos abarcan un amplio espectro de tareas que incluyen: planificación y estimación de proyectos, análisis de los requisitos del sistema y del software, diseño de estructuras de datos, arquitectura de programas y procedimientos algorítmicos, codificación, prueba y mantenimientos. Los métodos de la ingeniería del software introducen frecuentemente una notación especial orientada a un lenguaje gráfico y un conjunto de criterios para la calidad del software.

Las **herramientas** de la ingeniería del software suministran un soporte automático o semiautomático para los métodos. Hoy existen herramientas para soportar cada uno de los métodos mencionados anteriormente. Cuando se integran las herramientas de forma que la información creada por una herramienta pueda ser usada por otra, se establece un sistema para el soporte del desarrollo del software, llamada ingeniería del software asistida por computadora (del inglés, CASE). CASE combina software, hardware y bases de datos sobre ingeniería del software (una estructura de datos que contenga la información relevante sobre el análisis, diseño, codificación y prueba) para crear un entorno de ingeniería del software análogo al diseño/ingeniería asistido por computadora, CAD/CAE (de las siglas en inglés) para el hardware.

Los **procedimientos** de la ingeniería del software son el pegamento que junta los métodos y las herramientas y facilita un desarrollo racional y oportuno del software de computadora. Los procedimientos definen la secuencia en la que se aplican los métodos, las entregas (documentos, informes, formas, etc.) que se requiere, los controles que ayudan a asegurar la calidad y coordinar los cambios y las directrices que ayudan a los gestores de software a evaluar el progreso.

La ingeniería del software está compuesta por una serie de pasos que abarcan los métodos, las herramientas y los procedimientos antes mencionados. Estos pasos se denominan frecuentemente paradigmas de la ingeniería del software. La elección de un paradigma para la ingeniería del software se lleva a cabo de acuerdo con la naturaleza del proyecto y de las aplicación, los métodos y herramientas a usar y los controles y entregas requeridos. Tres son los paradigmas que se han tratado ampliamente; son los que se describen a continuación.

El ciclo de vida clásico.

La Fig. 2.5 ilustra el paradigma del ciclo de vida clásico para la ingeniería del software. Algunas veces llamado "modelo de cascada", el paradigma del ciclo de vida exige un enfoque sistemático y secuencial del desarrollo del software que comienza en el nivel del sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento. Modelado a partir del ciclo convencional de una ingeniería, el paradigma del ciclo de vida abarca las siguientes actividades:

Ingeniería y análisis del sistema. Debido a que el software es siempre parte de un sistema mayor, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego

asignando algún subconjunto de estos requisitos al software. Este planteamiento del sistema es esencial cuando el software debe interrelacionarse con otros elementos, tales como hardware, personas y bases de datos. La ingeniería y el análisis del sistema abarcan los requisitos globales a nivel del sistema con una pequeña cantidad de análisis y de diseño a un nivel superior.

Análisis de los requisitos del software. El proceso de recopilación de los requisitos se centra e intensifica especialmente para el software. Para comprender la naturaleza de los programas que hay que construir, el ingeniero de software ("analista") debe comprender el ámbito de la información del software, así como la función, el rendimiento y las interfaces requeridos. Los requisitos, tanto del sistema como del software, se documentan y se revisan con el cliente.

Diseño: El diseño del software es realmente un proceso multipaso que se enfoca sobre cuatro atributos distintos del programa: la estructura de los datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz. El proceso de diseño traduce los requisitos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación. Al igual que los requisitos, el diseño se documenta y forma parte de la configuración del software.

Codificación: El diseño debe traducirse en una forma legible para la máquina. El paso de codificación realiza esta tarea. Si el diseño se realiza de manera detallada, la codificación puede realizarse mecánicamente.

Prueba: Una vez que se ha generado el código, comienza la prueba del programa. La prueba se centra en la lógica interna del software, asegurando que todas las sentencias se han probado, y en las funciones externas, realizando pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren.

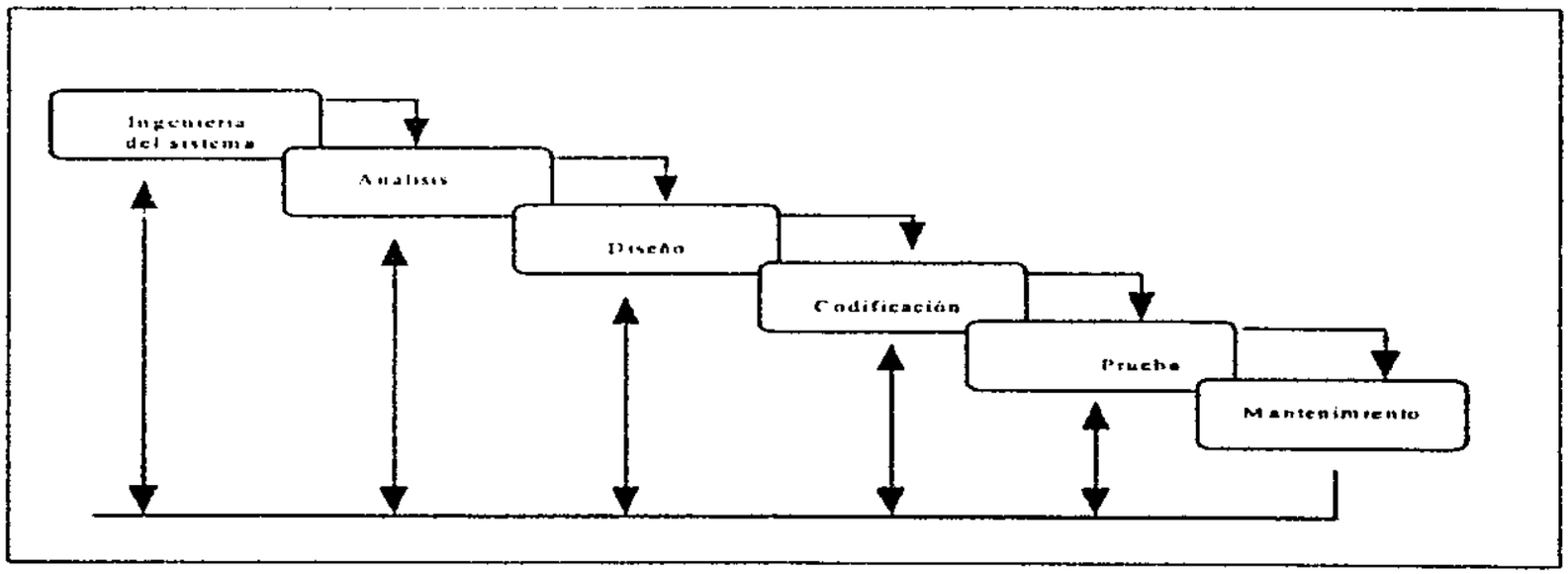


Fig. 2.5. El ciclo de vida clásico

Mantenimiento: El software, indudablemente, sufrirá cambios después de que se entregue al cliente (una posible excepción es el software empotrado). Los cambios ocurrirán debido a que se hayan encontrado errores, a que el software debe adaptarse a cambios del entorno externo (por ejemplo, un cambio solicitado debido a que se tiene un nuevo sistema operativo o dispositivo periférico), o debido a que el cliente requiera ampliaciones funcionales o del rendimiento. El mantenimiento del software aplica cada uno de los pasos procedentes del ciclo de vida a un programa existente en vez de a uno nuevo.

El ciclo de vida clásico es el paradigma más antiguo y más ampliamente usado en la ingeniería del software. Sin embargo, con el paso de unos cuantos años, se han producido críticas al paradigma, incluso por seguidores activos, que cuestionan su aplicabilidad a todas las situaciones. Entre los problemas que se presentan algunas veces, cuando se aplica el paradigma del ciclo de vida clásico, se encuentran:

- Los proyectos reales raramente siguen el flujo secuencial que propone el modelo. Siempre hay iteraciones y se crean problemas en la aplicación del paradigma.
- Normalmente, es difícil para el cliente establecer explícitamente al principio todos los requisitos. El ciclo de vida clásico lo requiere y tiene dificultades en acomodar posibles incertidumbres que pueden existir al comienzo de muchos proyectos.
- El cliente debe tener paciencia. Hasta llegar a las etapas finales del desarrollo del proyecto, no estará disponible una versión operativa del programa. Un error importante no detectado hasta que el programa esté funcionando puede ser desastroso.

Cada uno de estos problemas es real. Sin embargo, el paradigma clásico del ciclo de vida tiene un lugar definido e importante dentro del trabajo realizado en ingeniería del software. Suministra una plantilla en la que pueden colocarse los métodos para el análisis, diseño, codificación, prueba y mantenimiento. Además, veremos que los pasos del paradigma clásico del ciclo de vida son muy similares a los pasos genéricos aplicables a todos los paradigmas de ingeniería del software. El ciclo de vida clásico sigue siendo el modelo procedimental más ampliamente usado por los ingenieros del software. A pesar de sus inconvenientes, es significativamente mejor que desarrollar el software sin guías.

Construcción de prototipos.

Normalmente un cliente define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, proceso o salida. En otros casos, el programador puede no estar seguro de la eficiencia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma en que debe realizarse la interacción hombre-máquina. En éstas y muchas otras situaciones, puede ser mejor método de ingeniería del software la construcción de un prototipo.

La construcción de prototipos es un proceso que facilita al programador la creación de un modelo del software a construir. El modelo tomará una de las tres formas siguientes: (1) un prototipo en papel o un modelo basado en PC que describa la interacción hombre-máquina, de forma que facilite al usuario la comprensión de cómo se producirá tal interacción; (2) un prototipo que implemente algunos subconjuntos de la función requerida del programa deseado, o (3) un programa existente que ejecute parte o toda la función deseada, pero que tenga otras características que deben ser mejoradas en el nuevo trabajo de desarrollo.

La Fig. 2.6 muestra la secuencia de sucesos del paradigma de construcción de prototipos. Como en todos los métodos de desarrollo de software, la construcción de prototipos comienza con la recolección de los requisitos. El técnico y el cliente se reunen y definen los objetivos globales para el software, identifican todos los requisitos conocidos y perfilan las áreas en donde será necesario una mayor definición. Luego se produce un "diseño rápido". El diseño rápido se enfoca sobre la representación de los aspectos del software visible al usuario (por ejemplo, métodos de entrada y formatos de salida). El diseño rápido conduce a la construcción de un prototipo. El prototipo es evaluado por el cliente/usuario y se utiliza para refinar los requisitos del software a desarrollar. Se produce un proceso interactivo en el que el prototipo es "afinado" para que satisfaga las necesidades del cliente, al mismo tiempo que facilita al que lo desarrolla una mejor comprensión de lo que hay que hacer.

Idealmente, el prototipo sirve como mecanismo para identificar los requisitos del software. Si se va a construir un prototipo que funcione, el realizador intenta hacer uso de fragmentos de programas existentes o aplica herramientas (por ejemplo, generadores de informes, gestores de ventas, etc.) que faciliten la rápida generación de programas que funcionen.

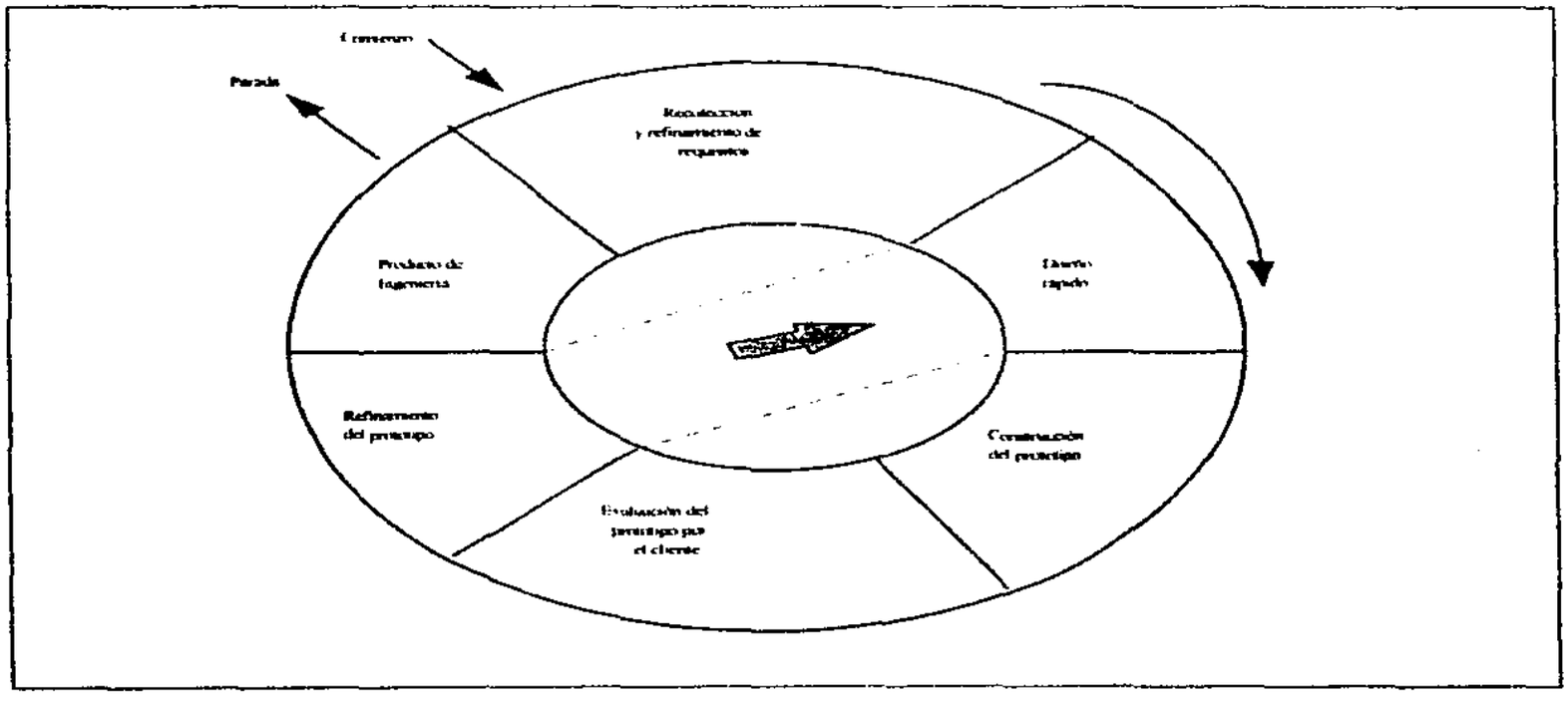


Fig. 2.6. Creación de prototipos

¿Pero qué debemos hacer con el prototipo cuando ya ha servido para el propósito establecido?

El prototipo puede servir como "primer sistema" - Pero esto puede ser visión idealizada. Al igual que en el ciclo de vida clásico, la construcción de prototipos como paradigma para la ingeniería del software, puede ser problemática por las siguientes razones:

- El cliente ve funcionando lo que parece ser una primera versión de software, ignorando que el prototipo se ha hecho con "plastilina y alambres", ignorando que, por las prisas en hacer que funcione, no hemos considerado los aspectos de calidad o de mantenimiento del software a largo plazo. Cuando se le informa de que el producto debe ser reconstruido, el cliente se vuelve loco y solicita que se apliquen "cuantas mejoras" sean necesarias para hacer el prototipo un producto final que funcione. El gestor de desarrollo del software cede demasiado a menudo.
- El técnico de desarrollo, frecuentemente, impone ciertos compromisos de implementación con el fin de obtener un prototipo que funcione rápidamente. Puede que utilice un sistema operativo o un lenguaje de programación inapropiados, simplemente porque ya está disponible y es conocido; puede que implemente ineficientemente un algoritmo, sencillamente para demostrar su capacidad. Después de algún tiempo, el técnico puede haberse familiarizado con esas elecciones y haber olvidado las razones por las que eran inapropiadas. La elección menos ideal forma ahora parte integral del sistema.

Aunque pueden aparecer problemas, la construcción de prototipos es un ***paradigma efectivo*** para la ingeniería del software. La clave está en definir al comienzo las reglas del juego; esto es, el cliente y el técnico deben estar de acuerdo en que el prototipo se construya para servir sólo como un mecanismo de definición de los requisitos. Posteriormente, ha de ser descartado (al menos en parte) y debe construirse el software real, con los ojos puestos en la calidad y en el mantenimiento.

El modelo en espiral

El modelo en espiral para la ingeniería del software ha sido desarrollado para cubrir las mejores características tanto de ciclo de vida clásico, como de la creación de prototipos, añadiendo al mismo tiempo un nuevo elemento: el análisis de riesgo, que falta en esos paradigmas. El modelo, representado mediante la espiral de la Fig. 2.7, define cuatro actividades principales, representadas por los cuatro cuadrantes de la figura:

- Planificación: determinación de objetivos, alternativas y restricciones.
- Análisis de riesgo: análisis de alternativas e identificación/resolución de riesgos.
- Ingeniería: desarrollo del producto de "siguiente nivel".
- Evaluación del cliente: valoración de los resultados de la ingeniería.

Un aspecto intrigante del modelo en espiral se hace evidente cuando consideramos la dimensión radial representada en la Fig. 2.7. Con cada iteración alrededor de la espiral (comenzando en el centro y siguiendo hacia el exterior), se construyen sucesivas versiones del software, cada vez más completas. Durante la primera vuelta alrededor de la espiral se definen los objetivos, las alternativas y las restricciones, y se analizan e identifican los riesgos. Si el análisis de riesgo indica que hay una incertidumbre en los requisitos, se puede usar la creación de prototipos en el cuadrante de ingeniería para dar asistencia tanto al encargado del desarrollo como al cliente. Se pueden usar simulaciones y otros modelos para definir más el problema y refinar los requisitos.

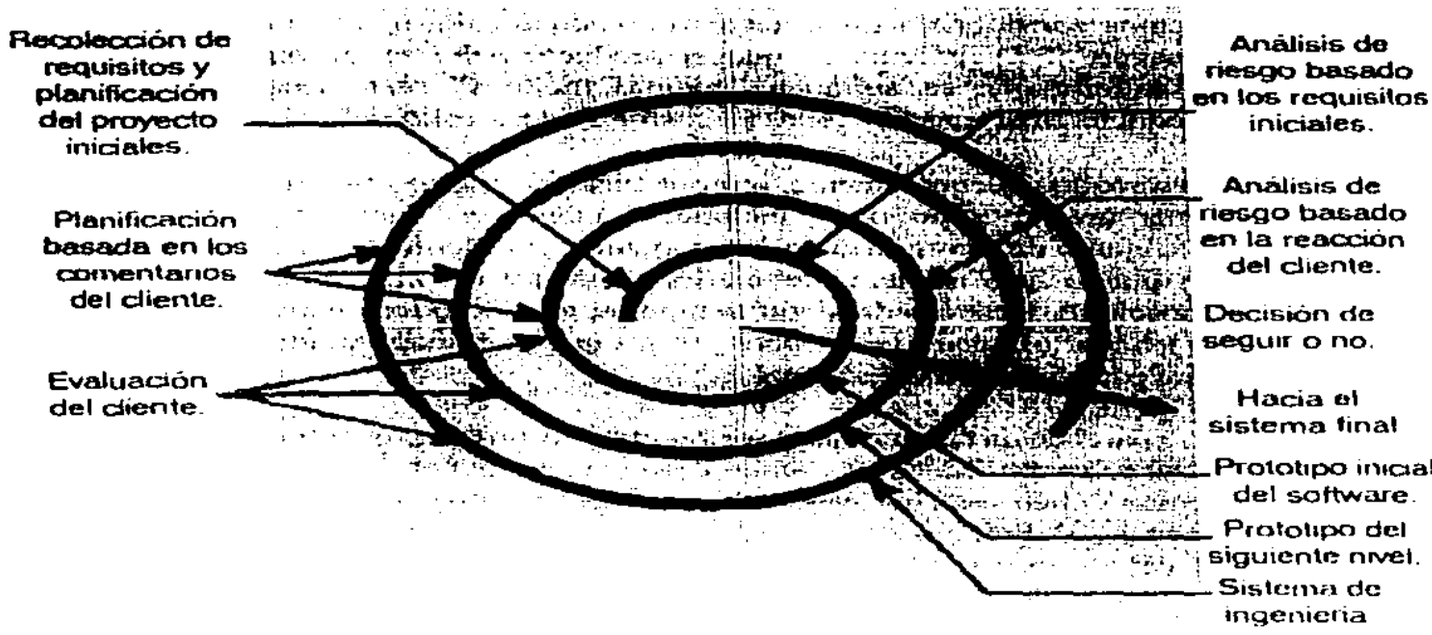


Fig. 2.7 El modelo en espiral.

El cliente evalúa el trabajo de ingeniería (cuadrante de evaluación del cliente) y sugiere modificaciones. Con base en los comentarios del cliente se produce la siguiente fase de planificación y de análisis de riesgo. En cada bucle alrededor de la espiral, la culminación del análisis de riesgo resulta en una decisión de "seguir o no seguir". Si los riesgos son demasiado grandes, se puede dar por terminado el proyecto.

Sin embargo, en la mayoría de los casos, se sigue avanzando alrededor del camino de la espiral, y ese camino lleva a los desarrolladores hacia fuera, hacia un modelo más completo del sistema, y, al final, al propio sistema operacional. Cada vuelta alrededor de la espiral requiere ingeniería (cuadrante inferior derecho), que se puede llevar a cabo mediante el enfoque del ciclo de vida clásico o de la creación de prototipos. Debe tenerse en cuenta que el número de actividades de desarrollo que ocurren en el cuadrante inferior derecho aumenta al alejarse del centro de la espiral.

El paradigma del modelo en espiral para la ingeniería del software es actualmente el enfoque más realista para el desarrollo de software y de sistemas a gran escala. Utiliza un enfoque "evolutivo" para la ingeniería del software, permitiendo al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo. Utiliza la creación de prototipos como un mecanismo de reducción del riesgo, pero, lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de creación de prototipos en cualquier etapa de la evolución del producto.

Mantiene el enfoque sistemático correspondiente a los pasos sugeridos por el ciclo de vida clásico, pero incorporándola dentro de un marco de trabajo interactivo que refleja de forma más realista el mundo real. El modelo en espiral demanda una consideración directa de riesgos técnicos en todas las etapas del proyecto y, si se aplica adecuadamente, debe reducir los riesgos antes de que se conviertan en problemáticos. Pero, al igual que otros paradigmas, el modelo en espiral no es la panacea. Puede ser difícil convencer a grandes clientes (particularmente en situaciones bajo contrato) de que el enfoque evolutivo es controlable. Requiere una considerable habilidad para la valoración del riesgo, y cuenta con esta habilidad para el éxito. Si no se descubre un riesgo importante, indudablemente surgirán problemas. Por último, el modelo en sí mismo es relativamente nuevo y no se ha usado tanto como el ciclo de vida o la creación de prototipos. Pasarán unos cuantos años antes de que se pueda determinar con absoluta certeza la eficacia de este importante nuevo paradigma.

Técnicas de cuarta generación

El término "técnicas de cuarta generación" (T4G) abarca un amplio espectro de herramientas de software que tienen algo en común: todas facilitan, al que desarrolla el software, la especificación de algunas características del software a alto nivel. Luego, la herramienta genera automáticamente el código fuente basándose en la especificación del técnico. Cada vez parece más evidente que cuanto mayor sea el nivel en el que se especifique el software, más rápido se podrá construir el programa. El paradigma T4G para la ingeniería del software se orienta hacia la posibilidad de especificar el software a un nivel más próximo al lenguaje natural o en una notación que proporcione funciones significativas.

Actualmente, un entorno para el desarrollo de software que soporte el paradigma T4G puede incluir todas o algunas de las siguientes herramientas: lenguajes no procedimentales para consulta o bases de datos, generación de informes, manipulación de datos, interacción y definición de pantallas, generación de código, y facilidades gráficas de alto nivel y facilidades de hoja de cálculo. Todas estas herramientas están disponibles, pero sólo para ámbitos de aplicación muy específicos. Actualmente, no existe un entorno T4G que puede aplicarse con igual facilidad a todas las categorías de aplicaciones del software descritas anteriormente.

En la Fig. 2.8 se describe el paradigma para la ingeniería del software. Al igual que otros paradigmas, T4G comienza con el paso de recolección de requisitos. Idealmente, el cliente describe los requisitos que son, traducidos directamente a un prototipo operativo. Sin embargo, en la práctica no se puede hacer eso. El cliente puede no estar seguro de lo que necesita; puede ser ambiguo en la especificación de hechos que le son conocidos y puede no desear o ser incapaz de especificar la información en la forma en que una herramienta T4G puede aceptarla. Además, las herramientas actuales de T4G no son lo suficientemente sofisticadas como para entender el lenguaje "natural", y no lo serán por algún tiempo. En este momento el diálogo cliente-técnico descrito por los otros paradigmas sigue siendo una parte esencial del enfoque T4G.

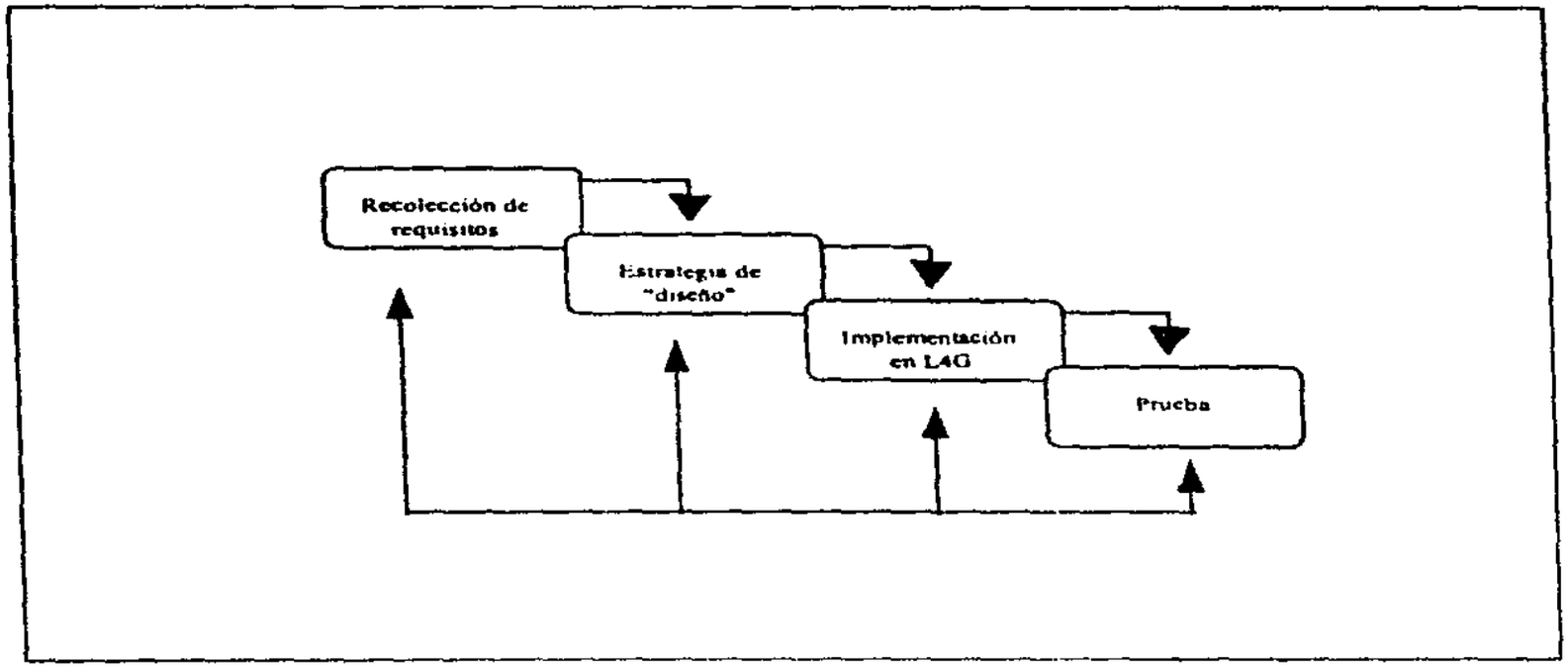


Fig. 2.8 Técnicas de cuarta generación

Para aplicaciones pequeñas, se puede ir directamente desde el paso de recolección de requisitos al paso de implementación, usando un lenguaje de cuarta generación no procedimental (L4G) . Sin embargo, es necesario un mayor esfuerzo para desarrollar una estrategia de diseño para el sistema, incluso si se utiliza un L4G. El uso de T4G sin diseño (para grandes proyectos) causará las mismas dificultades (poca calidad, mantenimiento pobre, mala aceptación por el cliente) que se encuentran cuando se desarrolla software mediante los enfoques convencionales.
La implementación mediante un L4G permite, al que desarrolla el software, centrarse en la representación de los resultados deseados, que es lo que se traduce automáticamente en un código fuente que produce dichos resultados. Obviamente, debe existir una estructura de datos con información relevante y a la que el L4G pueda acceder rápidamente.

Para transformar una implementación T4G en un producto, el que lo desarrolla debe dirigir una prueba completa, desarrollar una documentación con sentido y ejecutar el resto de actividades de "transición" requeridas en los otros paradigmas de ingeniería del software. Además, el software desarrollado con T4G debe ser construido de forma que facilite la realización del mantenimiento de forma expeditiva.

Ha habido mucha controversia y un considerable debate sobre el uso del paradigma T4G. Los defensores aducen reducciones drásticas en el tiempo de desarrollo del software y una mejora significativa en la productividad de la gente que construye el software. Los detractores aducen que las herramientas actuales de T4G no son más fáciles de utilizar que los lenguajes de programación, que el código fuente producido por tales herramientas es "ineficiente" y que el mantenimiento de grandes sistemas de software desarrollados mediante T4G, es cuestionable.

Ambas posturas tienen su parte de razón. Aunque es difícil separar los hechos de las suposiciones (existen pocos estudios controlados hasta el momento), es posible resumir el estado actual de los métodos de T4G.

- Con muy pocas excepciones, el ámbito de aplicación actual de las T4G está limitado a las aplicaciones de sistemas de información de gestión, concretamente al análisis de información y la obtención de informes relativos a grandes bases de datos. Sin embargo, las nuevas herramientas CASE soportan ahora el uso de T4G para la generación automática de "esquema de código" para aplicaciones de ingeniería y de tiempo real.
- Los datos preliminares recogidos en compañías que usan T4G parecen indicar que el tiempo requerido para producir software se reduce mucho para aplicaciones pequeñas y de tamaño medio, y que la cantidad de análisis y diseño para las aplicaciones pequeñas, también se reduce.
- Sin embargo, el uso de T4G para grandes trabajos de desarrollo de software exige el mismo o más tiempo de análisis, diseño y prueba (actividades de ingeniería del software), perdiéndose así un tiempo sustancial que se ahorra mediante la eliminación de la codificación.

Resumiendo, las técnicas de cuarta generación ya se han convertido en una parte importante del desarrollo de software en el área de aplicaciones de sistemas de información y probablemente llegarán a usarse bastante en aplicaciones de ingeniería y de tiempo real durante la segunda mitad de la década de los noventa. Como muestra la Fig. 2.9, la demanda de software continuará creciendo durante el resto de este siglo, pero los métodos y los paradigmas convencionales contribuirán probablemente cada vez menos al desarrollo del mismo. Las técnicas de cuarta generación llenarán el hueco existente.

Combinación de paradigmas

Frecuentemente, se describen los paradigmas de la ingeniería del software, tratados en las secciones anteriores, como métodos para la ingeniería del software en lugar de como métodos complementarios. En muchos casos, los paradigmas pueden y deben combinarse de forma que puedan utilizarse las ventajas de cada uno de ellos en un único proyecto. El paradigma del modelo en espiral lo hace directamente, combinado la creación de prototipos y algunos elementos del ciclo de vida clásico, en un enfoque evolutivo para la ingeniería del software. Pero ninguno de los paradigmas pude servir como base en la cual se integren los demás.

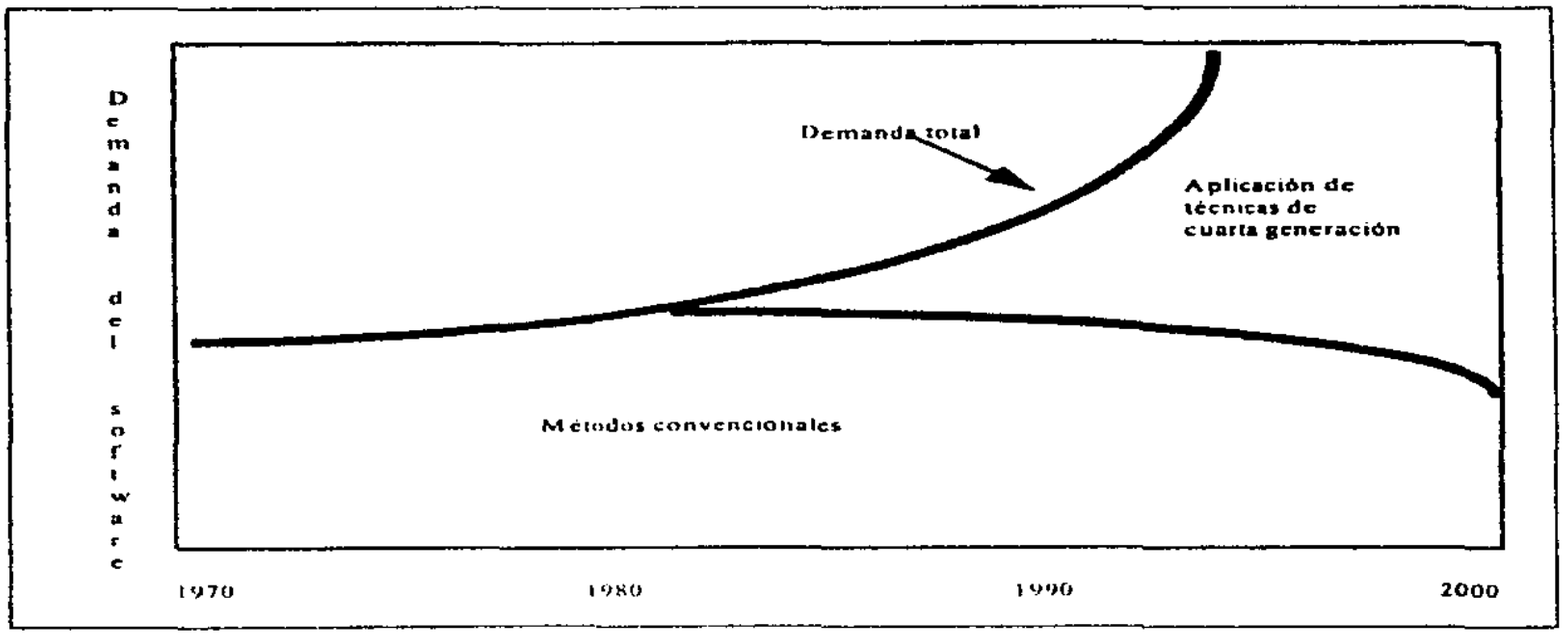


Fig. 2.9. Naturaleza mutable del desarrollo de software

La Fig. 2.10 muestra cómo pueden combinarse los tres paradigmas mencionados durante un trabajo de desarrollo de software. En todos los casos, el trabajo comienza con la determinación de objetivos, alternativas y restricciones paso que a veces se llama recolección preliminar de requisitos. A partir de ese punto se puede tomar cualquiera de los caminos indicados en la Fig. 2.10. Por ejemplo, se pueden seguir los pasos del ciclo de vida clásico (camino más a la izquierda), si se puede especificar completamente el sistema al principio de todo. Si los requisitos son inciertos, se puede usar un prototipo para definir mejor los requisitos. Usando el prototipo como guía, el que desarrolló puede después volver a los pasos del ciclo de vida clásico (diseño, codificación y prueba). Alternativamente, el prototipo puede evolucionar hacia el sistema o producir, con una vuelta al paradigma de ciclo de vida en el momento de la etapa de prueba. Se pueden usar las técnicas de cuarta generación para implementar el prototipo o para implementar el sistema durante el paso de codificaciones de ciclo de vida. También se puede usar un L4G junto con el modelo en espiral en los pasos de creación de prototipos o de codificación.

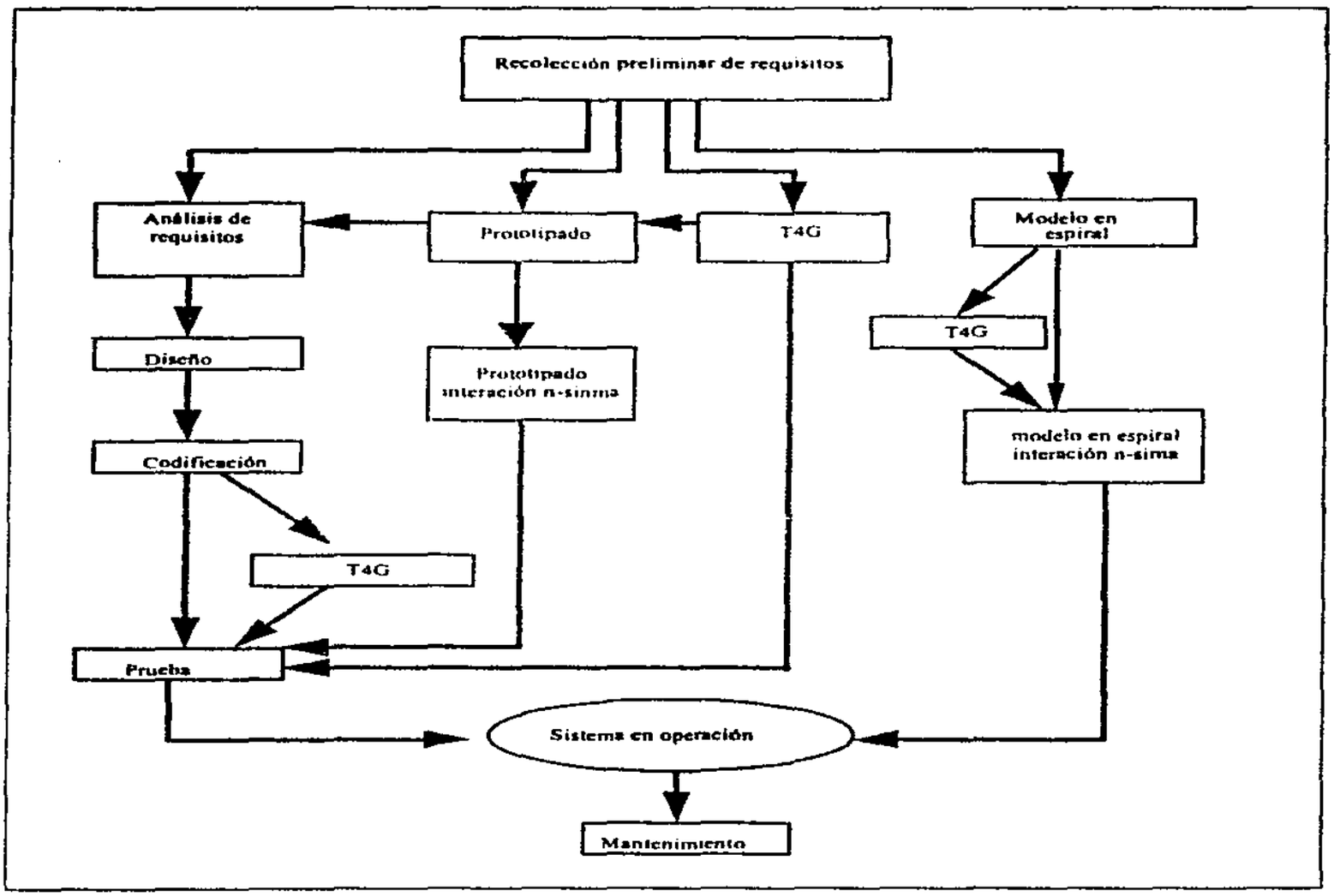


Fig. 2.10. Combinación de Paradigmas

No hay necesidad de ser dogmático en la elección de los paradigmas para la ingeniería del software; la naturaleza de la aplicación debe dictar el método a elegir. Mediante la combinación de paradigmas, el todo puede ser mejor que la suma de las partes.

2.3 Análisis Estructurado, un Poco de Historia

Al igual que muchas de las contribuciones importantes a la ingeniería del software, el análisis estructurado no fue introducido en un sólo artículo o libro clave que incluyera un tratamiento completo del tema. Los primeros trabajos sobre modelos de análisis aparecieron a finales de los 60's y principios de los 70's, pero la primera aparición del enfoque de análisis estructurado fue como complemento de otro tema importante - el "diseño estructurado". Los investigadores necesitaban una notación gráfica para representar los datos y los procesos que los transforman. Esos procesos quedarían finalmente establecidos en una arquitectura de diseño.

El término "análisis estructurado" fue popularizado por DeMarco. En su libro sobre esta materia. DeMarco presentó y denominó los símbolos gráficos clave que permitirían a un analista crear modelos de flujo de información; sugirió heurísticas para utilizar esos símbolos; sugirió el uso de un diccionario de datos y narrativas de procesamiento como complementos a los modelos de ese nuevo método. En los años siguientes. Page-Jones, Gane y Sarson y muchos otros propusieron variaciones del enfoque de análisis estructurado. En todos los casos, el método se centraba en aplicaciones de sistemas de información y no proporcionaba una notación adecuada para los aspectos de control y de comportamientos de los problemas de ingeniería de tiempo real. A mediados de los 80's, comenzaron a hacerse dolorosamente evidentes las deficiencias del análisis estructurado (cuando se intentaba aplicar el método a aplicaciones orientadas a control). Las "ampliaciones" para tiempo real fueron introducidas por Ward y Mellor y, más tarde por Hatley y Pirbhai. Con esas ampliaciones, se consiguió un método de análisis más robusto que podría ser aplicado de forma efectiva a problemas de ingeniería. En la actualidad, se está intentando desarrollar una notación consistente y se están publicando tratamientos modernos que permitan acomodar el uso de herramientas CASE (ingeniería del software asistida por computadora).

2.4 Análisis Orientado a Objetos y Modelado de Datos

Conceptos Orientados a los Objetos:

Cualquier discusión sobre el análisis orientado a los objetos (AOO) debe comenzar estableciendo el término "orientación a los objetos", ¿Qué significa el punto de vista orientado a los objetos? ¿Por qué existe un método considerado orientado a los objetos? ¿Qué es un objeto? Existen muchas opiniones diferentes sobre las respuestas correctas para esas preguntas. En la discusión que sigue, intentaremos sintetizar las más conocidas.

Para comprender el punto de vista orientado a los objetos, consideremos un ejemplo de un objeto del mundo real la cosa sobre la que está ahora mismo sentado - una silla. Silla es un miembro (también se usa el término "instalación") de una clase de objetos mucho mayor que denominamos mueble. Se puede asociar un conjunto de atributos genéricos a cada objeto de la clase muebles. Por ejemplo, todo mueble tiene precio, dimensiones, peso, situación y color, entre muchos atributos posibles. Se aplican igualmente si estamos hablando de una mesa o de una silla, de un sofá o de una cómoda. Dado que silla es un miembro de la clase mueble, hereda todos los atributos definidos para la clase. En la Fig. 2.11 se ilustra esquemáticamente este concepto.

Una vez que se ha definido la clase, se pueden reutilizar los atributos creando nuevas instalaciones de la clase. Por ejemplo, supongamos que vamos a definir un nuevo objeto denominado mella (un cruce entre una mesa y una silla) que sea miembro de la clase mueble. Mella hereda todos los atributos de muebles.

Hemos intentado definir una clase describiendo sus atributos, pero, sin embargo, falta algo. Cada objeto de la clase mueble puede ser manipulado de muchas formas. Puede ser vendido y comprado, modificado físicamente (por ejemplo, se le puede quitar una pata o pintarlo de dorado), o movido de un sitio a otro. Cada una de esas operaciones (otros términos que se usan son servicios o métodos) modifican uno o más atributos del objeto. Por ejemplo, si el atributo situación es realmente un elemento de datos compuesto que está definido como:
situación = edificio + piso + habitación
entonces, una operación denominada mover modificará uno o más de esos elementos de datos (edificio, piso o habitación) que componen el atributo situación. Para hacerlo, mover ha de ser "consciente" de la existencia de esos elementos de datos. Se puede utilizar la operación mover para una mesa o una silla, mientras que sean instalación de la clase mueble. Todas las operaciones válidas (p. eje.: comprar, vender, pesar) para la clase mueble están "conectadas" a la definición del objeto tal como muestra la Fig. 2.12, y son heredadas por todas las instancias de la clase.

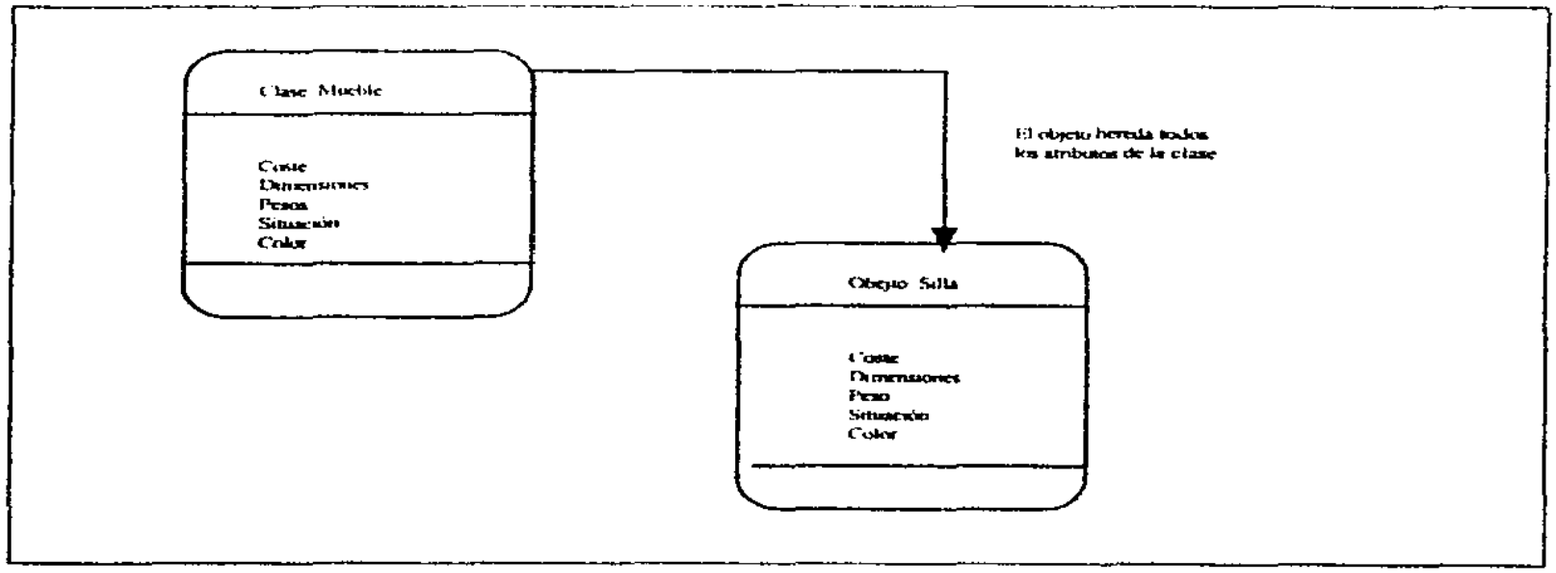


Fig. 2.11. Herencia de clase a objeto

El objeto silla (y en general todos los objetos) encapsulan datos (los valores de los atributos definidos para la silla), operaciones (las acciones que se aplican para cambiar los atributos de la silla), otros objetos (se pueden definir objetos compuestos), constantes (valores preestablecidos) y otra información relacionada. La encapsulación significa que toda esa información está empaquetada bajo un solo nombre y puede ser reutilizada como especificación o como componente de programa.

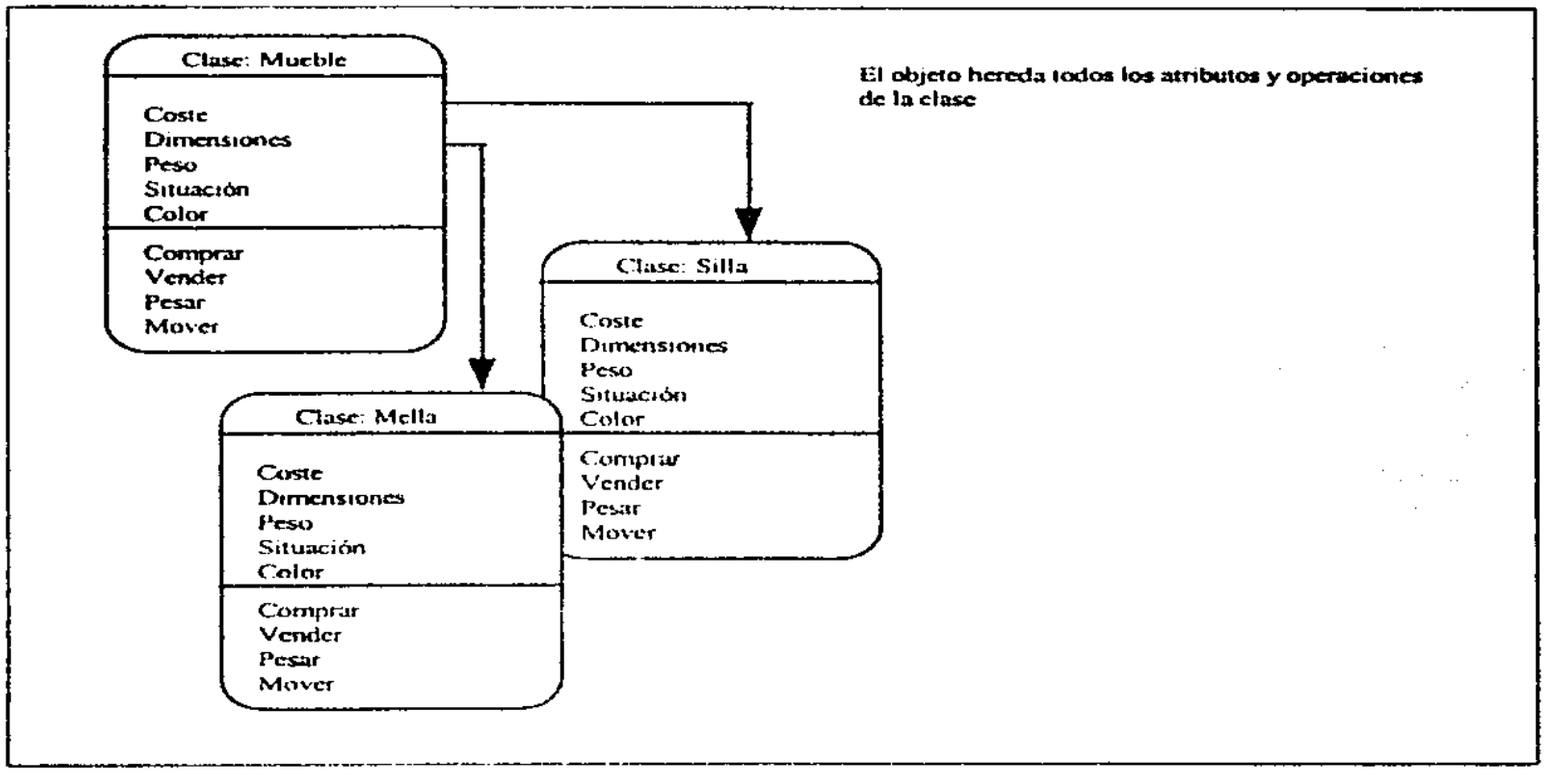


Fig. 2.12. Herencia de operaciones de clase u objeto.

Ahora que ya hemos introducido algunos conceptos básicos, resultará más adecuada una definición más formal de la "orientación a los objetos". Coad y Yourdon definen el término de la siguiente forma:

orientación a los objetos = objetos + clasificación + herencia + comunicación

Ya hemos introducido tres de esos conceptos. Pospondremos la discusión sobre la comunicación para un poco más adelante.

Identificación de objetos

Si miramos alrededor en una habitación, veremos un conjunto de objetos físicos que pueden ser identificados fácilmente, así como clasificados y definidos (en términos de atributos y operaciones). Pero como "observamos" el espacio del problema de una aplicación de software, resulta más complicado encontrar los objetos.

Podemos empezar a identificar objetos examinando la descripción del problema o llevando a cabo un "análisis gramatical" de la narrativa de procesamiento del sistema a construir. Determinamos los objetos subrayando cada nombre o cláusula nominal y añadiéndolo en una tabla. Se deben anotar los sinónimos. Si un objeto es necesario para implementar una solución,

entonces es parte del espacio de la solución; en caso contrario, cuando el objeto sólo es necesario para describir la solución, entonces es parte del espacio del problema. Pero, ¿qué debemos observar una vez que ya hemos aislado todos los nombres?

Los objetivos se manifiestan en una de las formas que se presentan en la Fig. 2.13. Los objetivos pueden ser:

- Entidades externas (p. ej.: otros sistemas, dispositivos, gente) que producen o consumen información a ser utilizada en el sistema basado en computadora.
- Cosas (p. eje.: informes, visualizaciones, cartas, señales) que son parte del dominio de información del problema.
- Ocurrencias o sucesos (p. ej.: una transferencia de una propiedad o la terminación de una serie de movimientos de un robot) que ocurren en el contexto de operación del sistema.
- Papeles que juegan las personas que interactúan con el sistema (p. ej.: gestor, ingeniero, vendedor).

Unidades organizativas (p.eje.: división, grupo, equipo) que son relevantes para la aplicación.

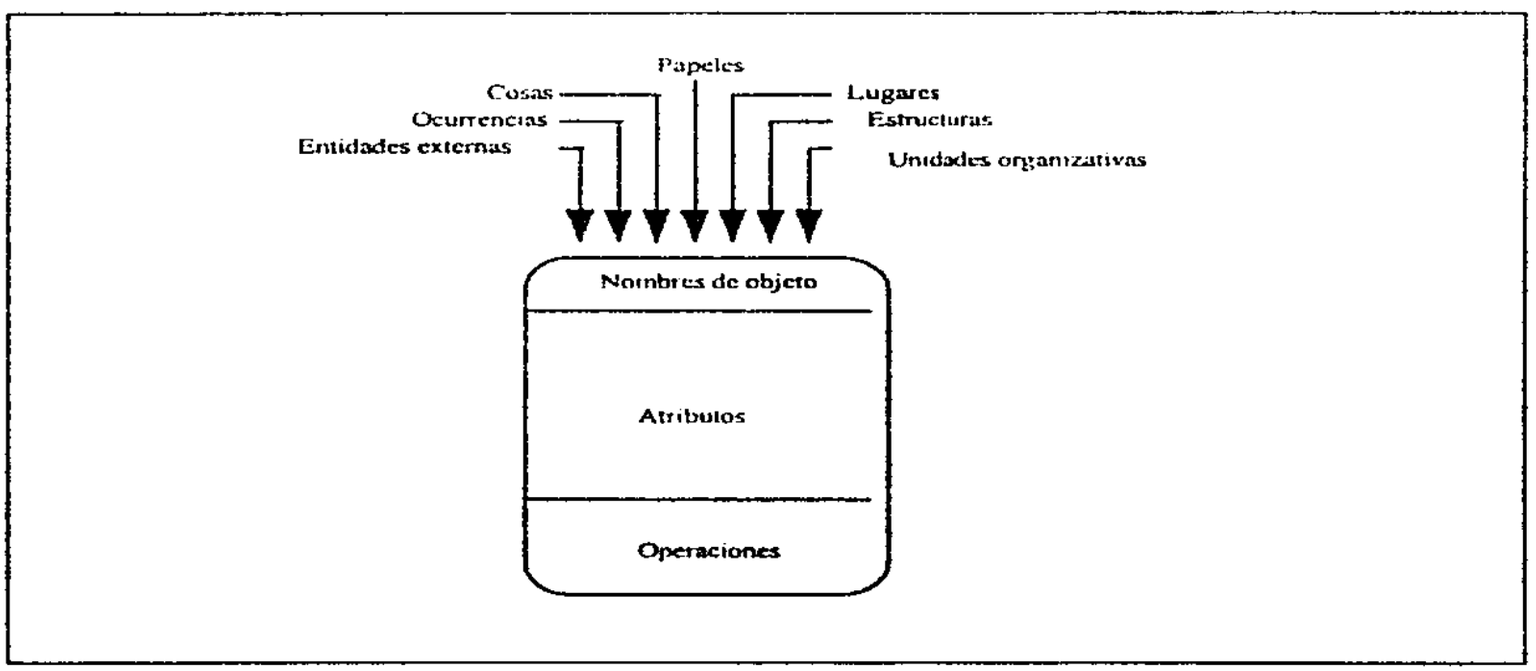


Fig. 2.13 Objetos

- **Lugares (p.ej.: sala de facturación o muelle de descarga) que establecen el contexto del problema y del funcionamiento general del sistema.**
- **Estructura (p. ej.: sensores, vehículos de cuatro ruedas o computadoras) que definen clases de objetos o, en caso extremos, clase de objetos relacionadas.**

También es importante darse cuenta de lo que son objetos. En general, un objeto no debe tener nunca un "nombre procedimental imperativo". Por ejemplo, si el equipo de desarrollo de un programa para un sistema de reconocimiento médico define un objeto con el nombre inversión de imagen, está cayendo en un sutil error. La imagen obtenida mediante el software por supuesto que puede ser un objeto (es una cosa que forma parte del campo de información). Inversión de la imagen es una operación que se aplica al objeto. Lo más normal será definir la inversión como una operación del objeto imagen, pero no ha de ser definida como un objeto aparte que signifique "imagen invertida". Como indica Cashman: " ...aunque la base de la orientación a los objetos es la encapsulación, todavía mantiene separados los datos de las operaciones sobre los datos."

Para ilustrar cómo se pueden identificar los objetos ya en las primeras etapas del análisis, vamos a hablar sobre ejemplo del sistema de seguridad. A continuación, presentamos la narrativa de procesamiento: El software HogarSeguro permite al propietario de la vivienda configurar el sistema de seguridad al instalarlo, controla todos los sensores conectados al sistema de seguridad e interactúa con el propietario a través de un teclado numérico y unas teclas de función que se encuentran en el panel de control de HogarSeguro que se muestra en la Fig. 2.14

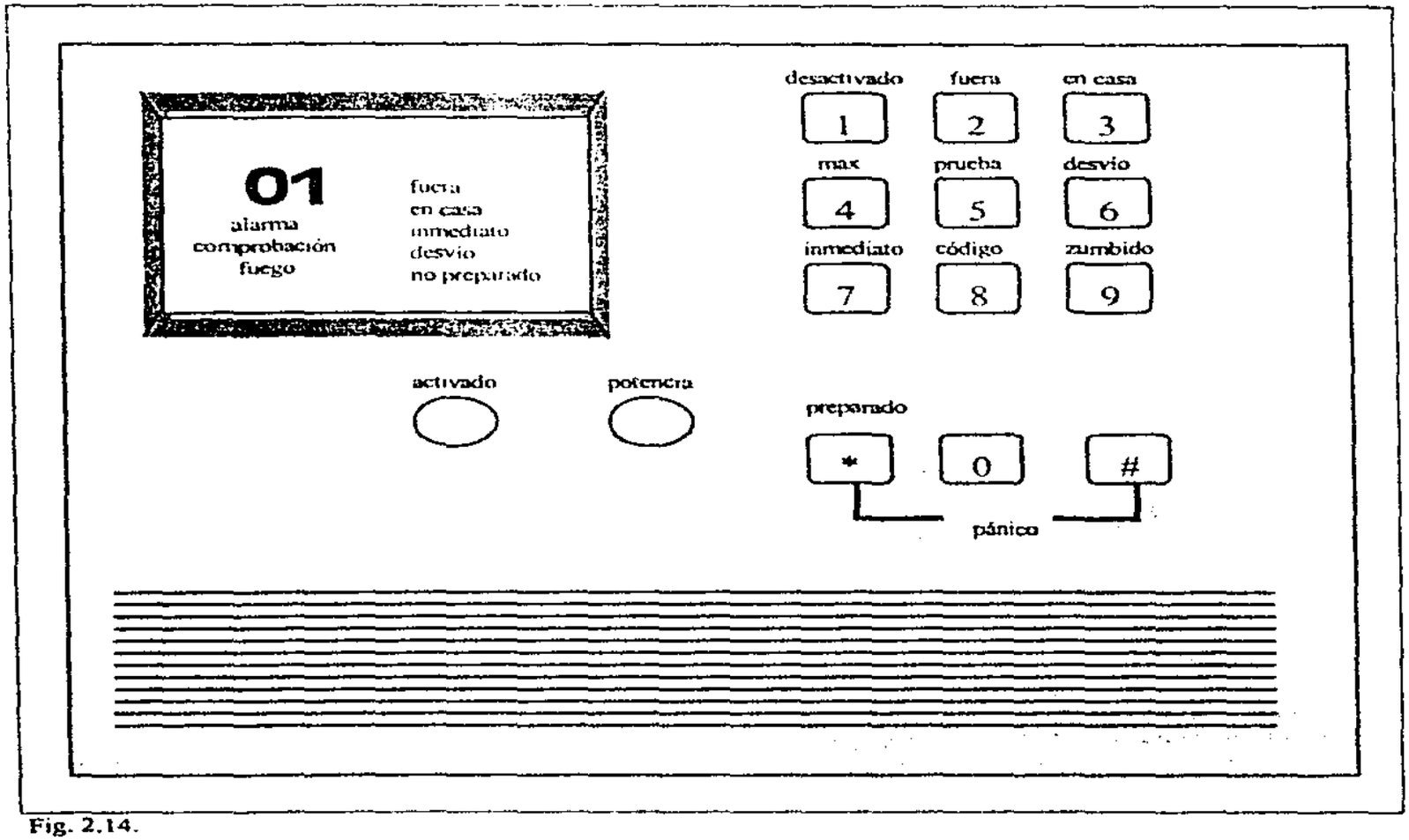


Fig. 2.14.

Durante la instalación, se usa el panel de control de HogarSeguro para "programar" y configurar el sistema. Cada sensor tiene asignado un número y un tipo; existente una contraseña maestra para activar y desactivar el sistema, y se introduce (n) un (os) teléfono (s) con qué contactar cuando se produce un suceso detectado por un sensor.Cuando el software detecta la sensorización de un suceso, hace que suene una alarma audible que está incorporada en el sistema. Tras un retardo, especificado por el propietario durante la configuración del sistema el programa marca un número de teléfono de un servicio de monitoreo, proporciona información sobre la situación e informa sobre la naturaleza del suceso detectado. Cada 20 segundos se volverá a marcar el número de teléfono hasta que se consiga establecer la comunicación.Toda la interacción con HogarSeguro está gestionada por un subsistema de interacción con el usuario que lee la información introducida a través del teclado numérico y las teclas de función, muestra mensajes de petición en un monitor LCD y muestra información sobre el estado del sistema en el monitor LCD.

Extrayendo los nombres, podemos proponer varios objetos potenciales.

Objeto/Clase potencial	Clasificación general
Propietario	Papel o entidad externa
Sensor	Entidad externa
Panel de control	
Instalación	Ocurrencia
Sistema (sinónimo: Sistema de seguridad)	Cosa
Número, tipo	No son objetos, son atributos de sensor
Contraseña maestra	Cosa
Número de teléfono	Cosa
suceso sensorizable	Ocurrencia
alarma audible	Entidad externa
servicio de monitoreo	Unidad organizativa o entidad externa

La lista anterior continuaría hasta que hayan sido considerados todos los nombres contenidos en la narrativa de procesamiento. Se puede observar cómo hemos identificado cada entrada de la lista como objeto potencial. Antes de tomar una decisión final, hemos de considerar cada uno de ellos más detenidamente.

Coad y Yourdon sugieren seis características selectivas que debe usar el analista para considerar la inclusión o no de cada objeto potencial en el modelo de análisis.

1. ***Información retenida.*** El objeto potencial será útil durante el análisis sólo si la información sobre el mismo ha de ser recordada para que el sistema pueda funcionar.
2. ***Servicios necesarios.*** El objeto potencial debe tener un conjunto de operaciones identificables que puedan cambiar de alguna forma el valor de sus atributos.
3. ***Múltiples atributos.*** Durante el análisis de requisitos, se debe centrar la atención en la información "principal", un objeto con un solo atributo puede resultar útil durante el diseño,

pero probablemente se represente mejor como atributo de otro objeto durante la fase de análisis.

4. ***Atributos comunes.*** Se pueden definir conjuntos de atributos para los objetos potenciales y aplicar así esos atributos a todas las ocurrencias del objeto.
5. ***Operaciones comunes.*** Se pueden definir conjuntos de operaciones para los objetos potenciales y aplicar así esas operaciones a todas las ocurrencias del objeto.
6. ***Requisitos esenciales.*** En el modelo de requisitos casi siempre se definirán como objetos las entidades externas que aparecen en el espacio del problema y que producen o consumen información esencial para el funcionamiento de cualquier solución que se desarrolle para el sistema.

Para que un objeto sea considerado como legítimo para ser incluido en el modelo de requisitos, el objeto potencial debe satisfacer todas (o casi todas) las características anteriores. La decisión de incluir o no un objeto potencial en el modelo de análisis puede resultar algo subjetiva, haciendo que una posterior evaluación pueda descartar o reinstanciar un determinado objeto. Sin embargo, el primer paso del AOO debe consistir en la definición de los objetos, debiendo llevar a cabo decisiones (incluso aunque sean subjetivas). Teniendo esto en cuenta, aplicamos las características de selección a la lista de objetos potenciales de HogarSeguro:

Objeto/Clase potencial	Número de características aplicable
propietario	rechazado: fallo 1 y 2, aunque 6 sea aplicable
sensor	aceptado: todas son aplicables
panel de control	aceptado: todas son aplicables
instalación	Rechazado
sistema (sinónimo: sistema de seguridad)	aceptado: todas son aplicables
número, tipo	rechazados: falla 3, son atributos de sensor
contraseña maestra	rechazado: falla 3
número de teléfono	rechazado: falla 3
suceso sensorizable	aceptado: todas son aplicables
alarma audible	aceptado: aplicables 2,3,4,5 y 6
servicio de monitoreo	rechazado: falla 1 y 2, aunque 6 sea aplicable.

Se debe observar que (1) la lista anterior no incluye todo - puede que haya que añadir otros objetos para completar el modelo; (2) algunos de los objetos rechazados se convertirán en atributos de los objetos aceptados (p. ej.: número y tipo son atributos de sensor y contraseña maestra y número de teléfono serán atributos de sistema); (3) descripciones diferentes del problema pueden llevar a diferentes "aceptaciones" y "rechazos" (p. ej.: si cada propietario tuviera su propia contraseña o fuera identificado por su voz, el objeto propietario satisfaceria las características 1 y 2 y hubiera sido aceptado).

Especificación de atributos

Los atributos describen un objeto que haya sido seleccionado para ser incluido en el modelo de análisis. Esencialmente, son los atributos los que definen un objeto - son los que aclaran el significado del objeto en el contexto del espacio del problema. Por ejemplo, si fuéramos a desarrollar un sistema que gestionara las estadísticas de béisbol para los jugadores profesionales, los atributos de objeto jugador serían bastante diferentes de los que tendría si ese objeto se usara en el contexto de un sistema de pensiones de jugadores profesionales de béisbol. En el primero, pueden ser relevantes atributos como el nombre, la posición, la media de bateo, el porcentaje de estancia en el campo de juego, los años en la liga o los partidos jugados. En el segundo caso, algunos de esos atributos seguirían siendo significativos, mientras que otros serían reemplazados por atributos como el salario medio, el crédito total, el beneficio mensual deseado, la dirección postal, etc.

Para desarrollar un conjunto significativo de atributos para un objeto, el analista debe estudiar de nuevo la narrativa de procesamiento (o descripción de alcance) para el problema concreto y seleccionar aquello de "pertenezca" de forma razonable al objeto. Además, se debe responder a la siguiente pregunta para cada objeto: ¿qué elementos de datos (elementos y/o compuestos) definen completamente ese objeto en el contexto del problema actual?

Para ilustrar esto, consideremos el objeto sistema definido para HogarSeguro. Anteriormente ya indicamos que el propietario puede configurar el sistema de seguridad para que refleje la información sobre los senderos, sobre la respuesta de la alarma, sobre la activación y desactivación, sobre identificación, etc.. Podemos representar esos elementos de datos compuestos de la siguiente forma:

- información de sensor = tipo de sensor + número de sensor + umbral de alarma
- información de respuesta de alarma = tiempo de retardo + número de teléfono + tipo de alarma
- información de activación/desactivación = contraseña maestra + número de intentos permitidos + contraseña temporal
- Información de identificación = ID del sistema + número de teléfono de verificación + estado del sistema

Cada uno de los elementos de datos que aparecen a la derecha de los signos de igualdad se puede definir más exahustivamente, pero por lo que a nosotros respecta, ya constituyen una lista de atributos razonable para el objeto sistema (Fig. 2.15)

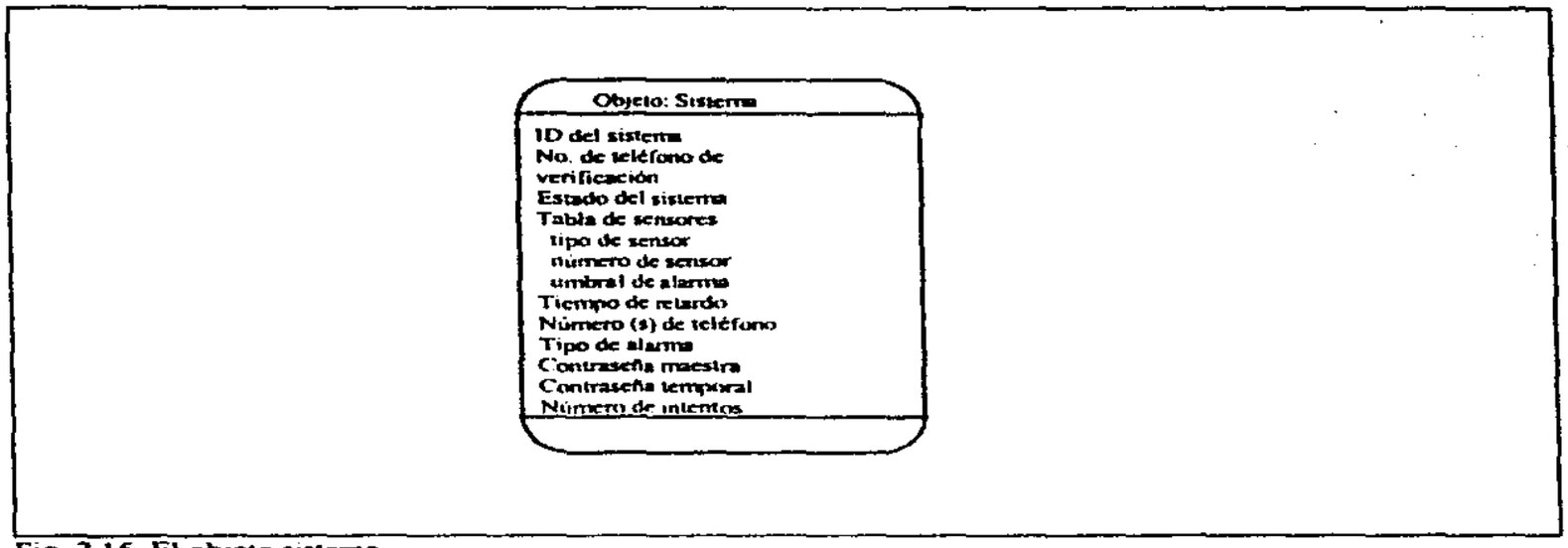


Fig. 2.15. El objeto sistema.

Definición de las operaciones

Una operación cambia un objeto de alguna forma. Más concretamente, cambia valores de uno o más atributos que están contenidos en el objeto. Por tanto, una operación debe tener "conocimiento" de la naturaleza de los atributos del objeto y deben ser implementadas de forma que se les permita manipular las estructuras de datos que hayan sido derivadas de los atributos.

Aunque existen muchos tipos distintos de operaciones, generalmente se pueden dividir en tres grandes categorías: (1) operaciones que manipulan los datos de alguna forma (p. ej.: adiciones, eliminaciones, cambios de formato, selecciones); (2) operaciones que realizan algún cálculo; (3) operaciones que monitorean un objeto frente a la ocurrencia de algún suceso de control.

Para realizar un "primer intento" de obtener un conjunto de operaciones para los objetos del modelo de análisis, el analista ha de volver a estudiar la narrativa de procesamiento (o descripción de ámbito) del problema y seleccionar aquellas operaciones que "correspondan" razonablemente a cada objeto. Para hacerlo, se lleva a cabo de nuevo un análisis gramatical y se aíslan los verbos. Algunos de los verbos serán operaciones legítimas y se podrán conectar facilmente con un objeto específico. Por ejemplo, de la narrativa de procesamiento de HogarSeguro, que se presentó anteriormente, podemos ver que "cada sensor tiene asignado un número y un tipo" o que "se programa una contraseña maestra para activar y desactivar el sistema ". Estas dos frases indican varias cosas:

- Que para el objeto sensor es relevante una operación de asignación.
- Que se aplicará una operación programar al objeto sistema.
- Que activar y desactivar son operaciones que se aplican al sistema (también que el estado del sistema debe ser ahora definido, usando la notación de diccionario de datos, como estado de sistema = activado / desactivado).

Tras una investigación más detallada, es probable que haya que dividir la operación programar en varias suboperaciones más específicas requeridas para configurar el sistema. Por ejemplo, programar implica: especificar números de teléfono, configurar las características del sistema (p. ej.: crear la tabla de senderos, introducir las características de la alarma) e introducir contraseñas. Pero, por ahora, especificamos programar como una sola operación.

Además del análisis gramatical, podemos comprender mejor las operaciones considerando la comunicación que ocurre entre los objetos. Los objetos se comunican mediante el paso de mensajes a otros objetos. Antes de continuar con la especificación de las operaciones, exploremos este asunto con más detalle.

Comunicación entre objetos

La definición de los objetos del contexto del modelo de análisis puede ser suficiente para establecer una base para el diseño, pero se debe añadir algo más (durante el análisis o durante el diseño) para que se pueda construir el sistema - se debe establecer un mecanismo para la comunicación entre los objetos. Este mecanismo se denomina mensaje.
En la Fig. 2.16 se muestra esquemáticamente el uso de mensajes. Cuatro objetos, A, B, C y D, se comunican entre sí mediante el paso de mensajes. Por ejemplo, si el objeto B requiere un procesamiento asociado con la operación op10 del objeto D, entonces le envía a D un mensaje que puede tener la siguiente forma:

mensaje: (destino, operación, argumentos) donde:

- "destino" define el objeto (en este caso, D) que recibe el mensaje
- "operación" refiere la operación que va a recibir el mensaje (op10)
- "argumentos" proporciona información requerida para que se pueda llevar a cabo la operación.

Como parte de la ejecución de la operación op10, el objeto D puede enviar un mensaje al objeto C de la forma:

mensaje (C,op08, <datos>)

C busca op08, la realiza y devuelve el control a D. La operación op10 termina y se devuelve el control a B.

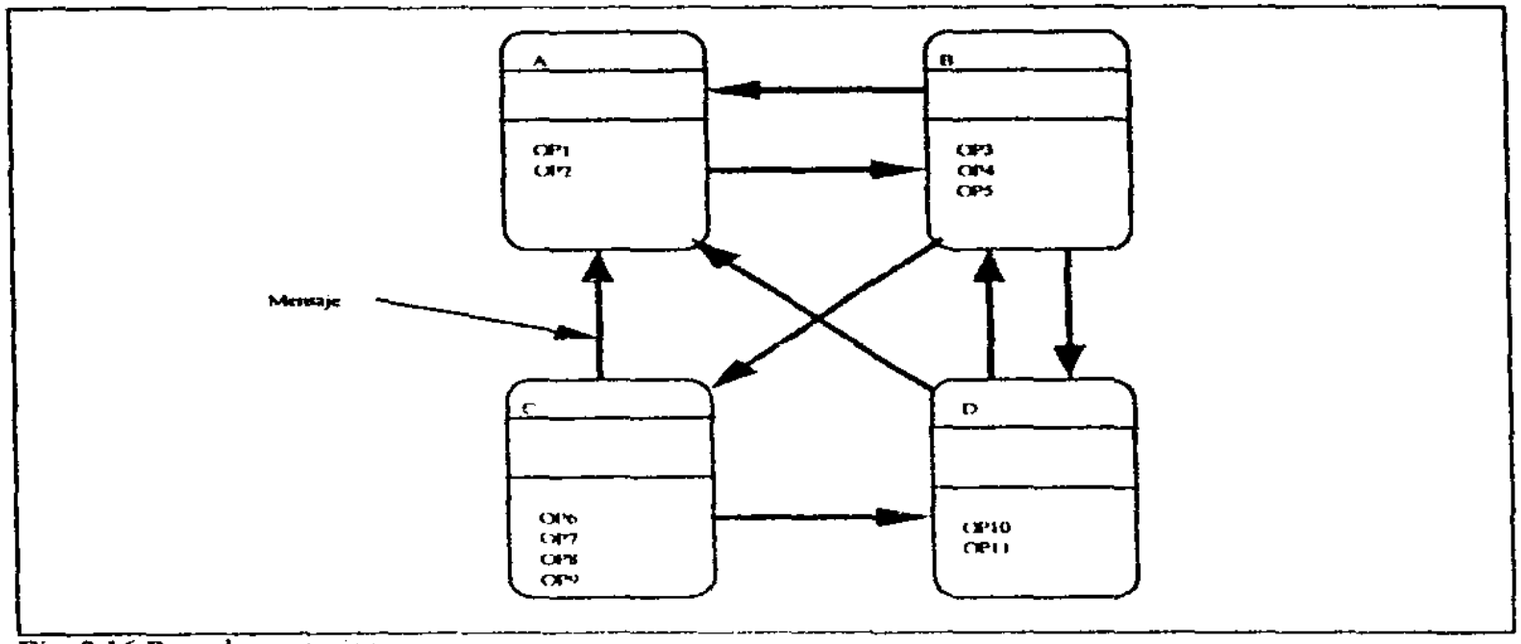


Fig. 2.16 Paso de mensajes

Cox describe el intercambio entre objetos de la siguiente forma:

A un objeto se le pide realice una de sus operaciones enviándole un mensaje que le indique qué es lo que tiene que hacer. El objeto receptor responde al mensaje escogiendo primero la operación que implementa el nombre del mensaje, ejecutando esa operación y devolviendo el control al emisor.

La mensajería es importante para la implementación de un sistema orientado a los objetos, pero no es necesario considerarla detalladamente durante el análisis de requisitos. De hecho, en esta etapa nuestro único cometido es usar el concepto como ayuda para la determinación de operaciones candidatas para un objeto concreto.

Fin de la definición del objeto

La definición de la operaciones es el último paso para tener la especificación de un objeto completa. Se puede determinar operaciones adicionales considerando el "historial" de un objeto y los mensajes que se pasan entre objetos definidos en el sistema. El historial genérico de un objeto puede ser definido reconociendo que se debe crear, modificar, manipular o leer en otras formas y posiblemente eliminar. Para el objeto sistema, se puede expandir ésto para reflejar actividades que ocurren durante su vida (en este caso, durante el tiempo que HogarSeguro esté en funcionamiento). Algunas de las operaciones pueden ser extraídas de la probable comunicación entre los objetos. Por ejemplo, suceso de sensor enviará un mensaje al sistema para que muestre la situación del suceso y el número; panel de control enviará un mensaje de reinicialización para actualizar el estado del sistema (un atributo); la alarma audible enviará un mensaje de petición; el

panel de control enviará un mensaje de modificación para cambiar uno o más atributos sin reconfigurar el sistema entero; suceso de sensor también enviará un mensaje para llamar el número de teléfono contenido en el objeto. Se pueden considerar otros mensajes y derivar de ellos operaciones La definición resultante para el objeto es la que se muestra en la Fig. 2.17. Para el resto de los objetos definidos para HogarSeguro se seguiría un enfoque similar. Una vez que se han definido los atributos y las operaciones para cada uno de los objetos hasta este nivel, se puede crear un modelo de AOO inicial (Fig. 2.18).

Modelo de Análisis Orientado a los Objetos

Se han sugerido varios esquemas de modelado diferentes para el AOO. Todos hacen uso de la definición de objetos presentada anteriormente, pero cada una introduce sus propia notación, heurísticas y filosóficas. Para ilustrar el enfoque de modelado del AOO, usaremos la notación adoptada por Coad y Yourdon.

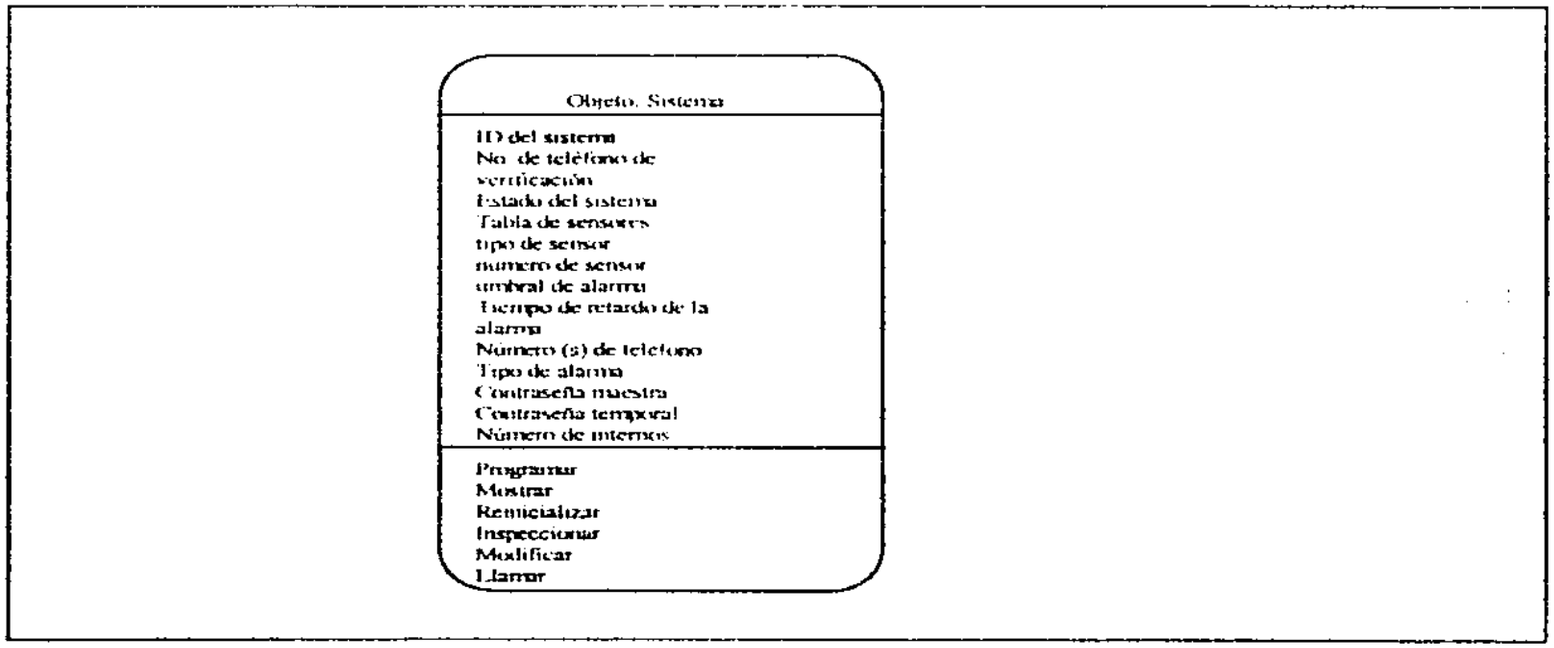


Fig. 2.17 El objeto sistema con sus operaciones.

El enfoque de AOO propuesto por Coad y Yourdon consiste en cinco pasos: (1) identificación de objetos; (2) identificación de estructura; (3) definición de temas; (4) definición de conexiones entre atributos e instancia; y (5) definición de conexiones entre operaciones y mensajes. En los párrafos que siguen se muestra una introducción de los mecanismos y notaciones de los pasos 2 al 5.

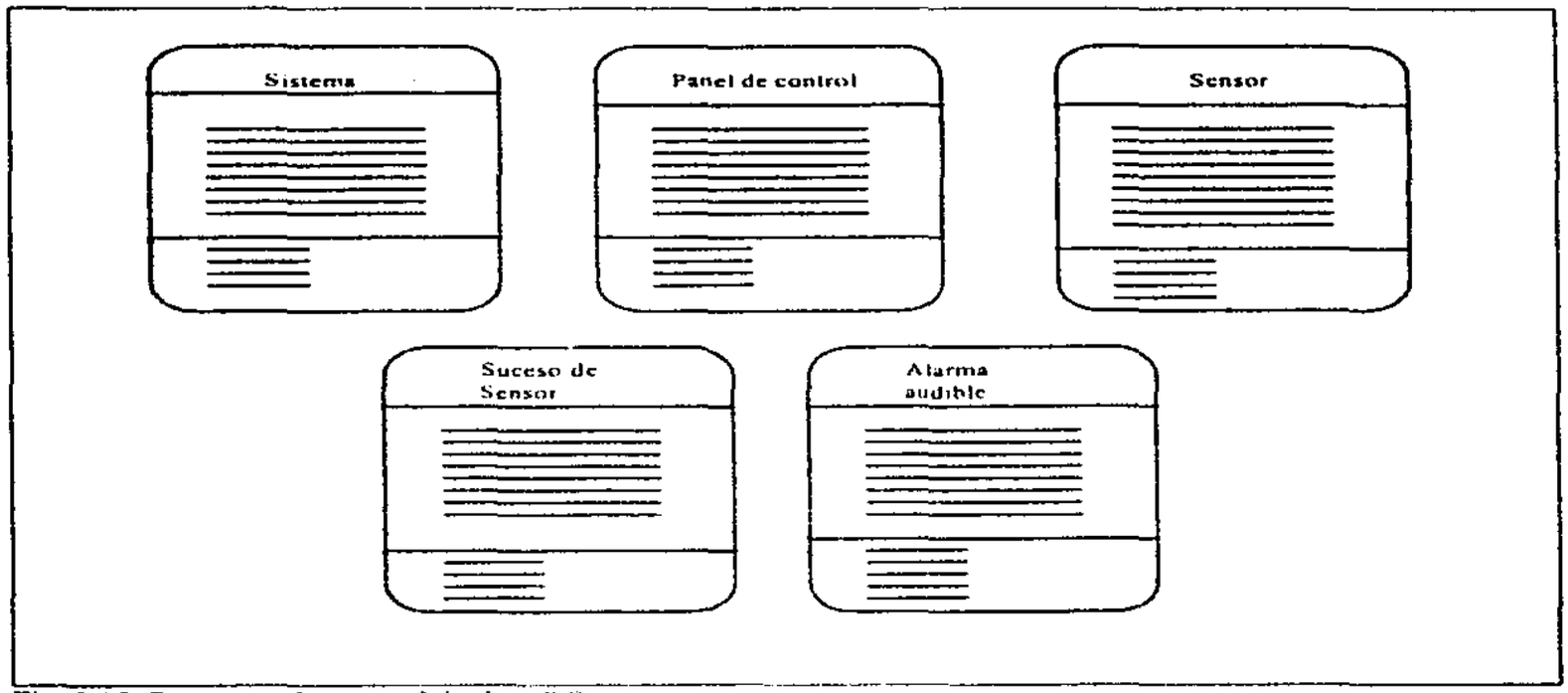


Fig. 2.18 Comienzo de un modelo de AOO.

Estructuras de clasificación y ensamblaje

Una vez que se han identificado los objetos, el analista pasa a centrarse en la estructura de clasificación. Se considera cada objeto como una "generalización" y luego como una "especialización". Es decir, se definen y se nombran las instancias de un objeto. Para ilustrar ésto, considere el objeto sensor definido para HogarSeguro y mostrado en la Fig. 2.19 En otros casos, puede que un objeto representado en el modelo inicial esté realmente compuesto por varias partes constitutivas que pueden ser definidas en si mismas como objetos. Las estructuras de este tipo se denominan estructuras de ensamblaje y se definen con la notación presentada en la Fig. 2.20. El triángulo implica la relación de ensamblaje. Se debe tener en cuenta que las líneas de conexión pueden ser ampliadas con símbolos adicionales (que no se muestran) que están adaptados de la notación del modelo entidad-relación que se discute más adelante.

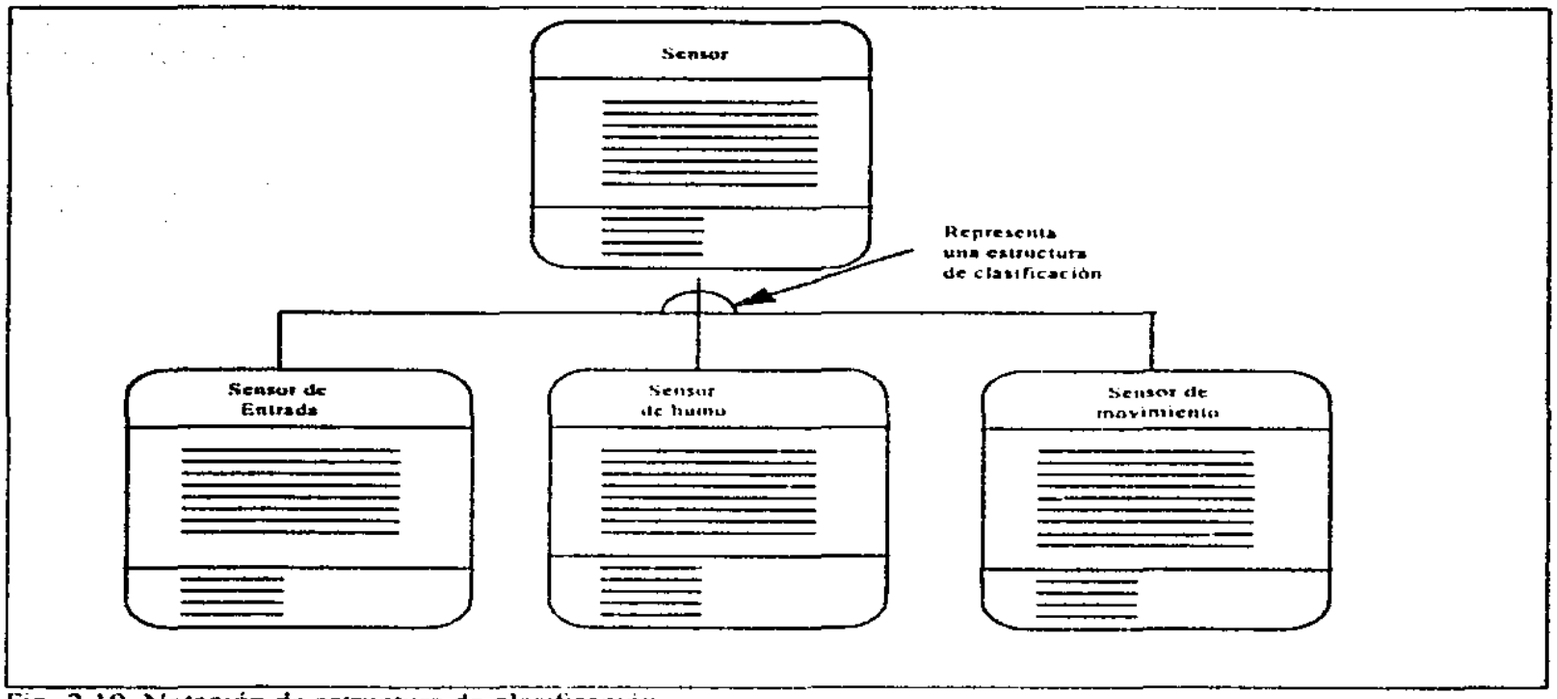


Fig. 2.19. Notación de estructura de clasificación

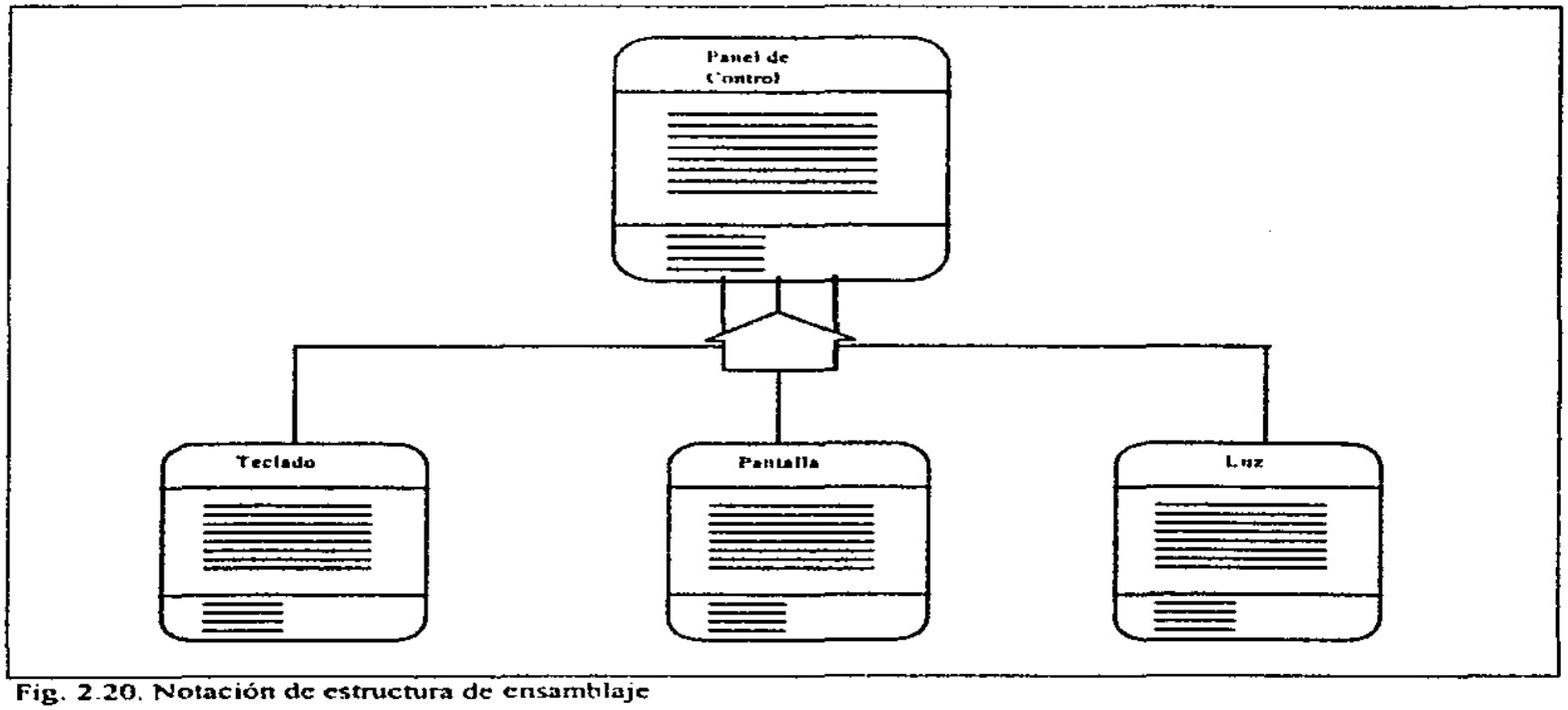


Fig. 2.20. Notación de estructura de ensamblaje

La representación estructural proporciona al analista un medio de partición del modelo de requisitos. La expansión de cada objeto proporciona los detalles necesarios para las revisiones y el subsiguiente diseño.

Definición de temas

Un modelo de AOO real puede tener cientos de objetos y docenas de estructuras. Por esta razón, es necesario definir alguna representación concisa que sea un compendio de los modelos de objeto y de estructura descritos anteriormente.

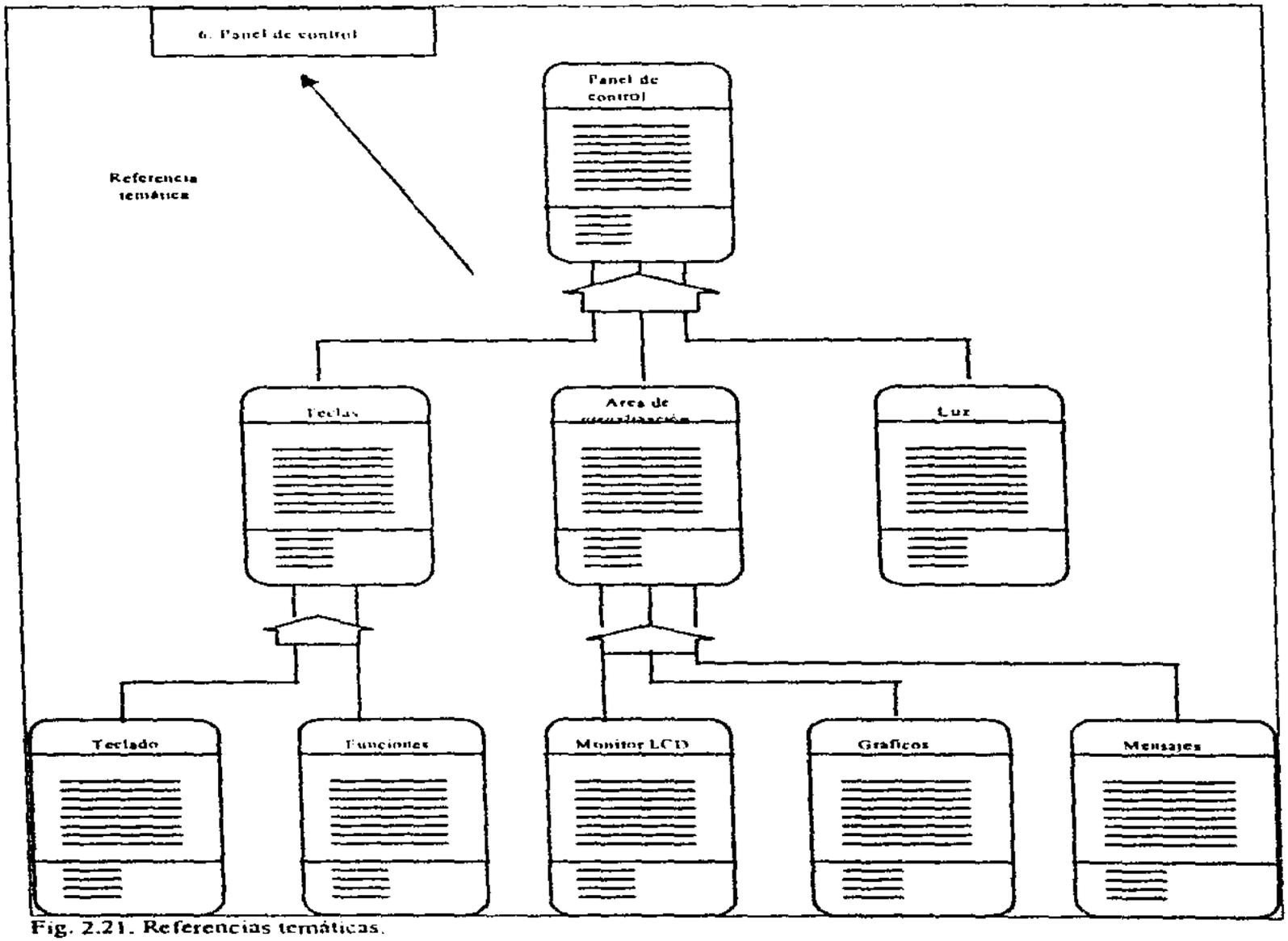


Fig. 2.21. Referencias temáticas.

Un tema no es más que una referencia o puntero a mayores detalles del modelo de análisis. Por ejemplo, supongamos que el panel de control de HogarSeguro es considerablemente más complejo que lo que implica la Fig. 2.20, conteniendo múltiples áreas de visualización, una disposición sofisticada de teclas y otras características. Se puede modelar como la estructura de ensamblaje de la Fig. 2.21. Si el modelo de requisitos general contiene docenas de estas estructuras (no es el caso de HogarSeguro), sería difícil asimilar la representación entera de una vez. Definiendo una referencia temática como se muestra en la Fig. 2.21, se puede referenciar la estructura entera con un sencillo icono (el rectángulo) Generalmente, se crean referencias temáticas para cualquier estructura que contenga más de cinco o seis objetos.
Al nivel más abstracto, el modelo de AOO contendrá sólo referencias temáticas como las que se muestran en la parte superior de la Fig. 2.22. Cada referencia se expandirá en una estructura. En la parte inferior de la Fig. 2.22 se muestran las estructuras de los objetos panel de control y sensor (Fig. 2.20 y 2.21) para los objetos sistema, suceso de sensor y alarma audible también habríamos creado estructuras en caso de que esos objetos requirieran más de cinco o seis objetos de clasificación o de ensamblaje.

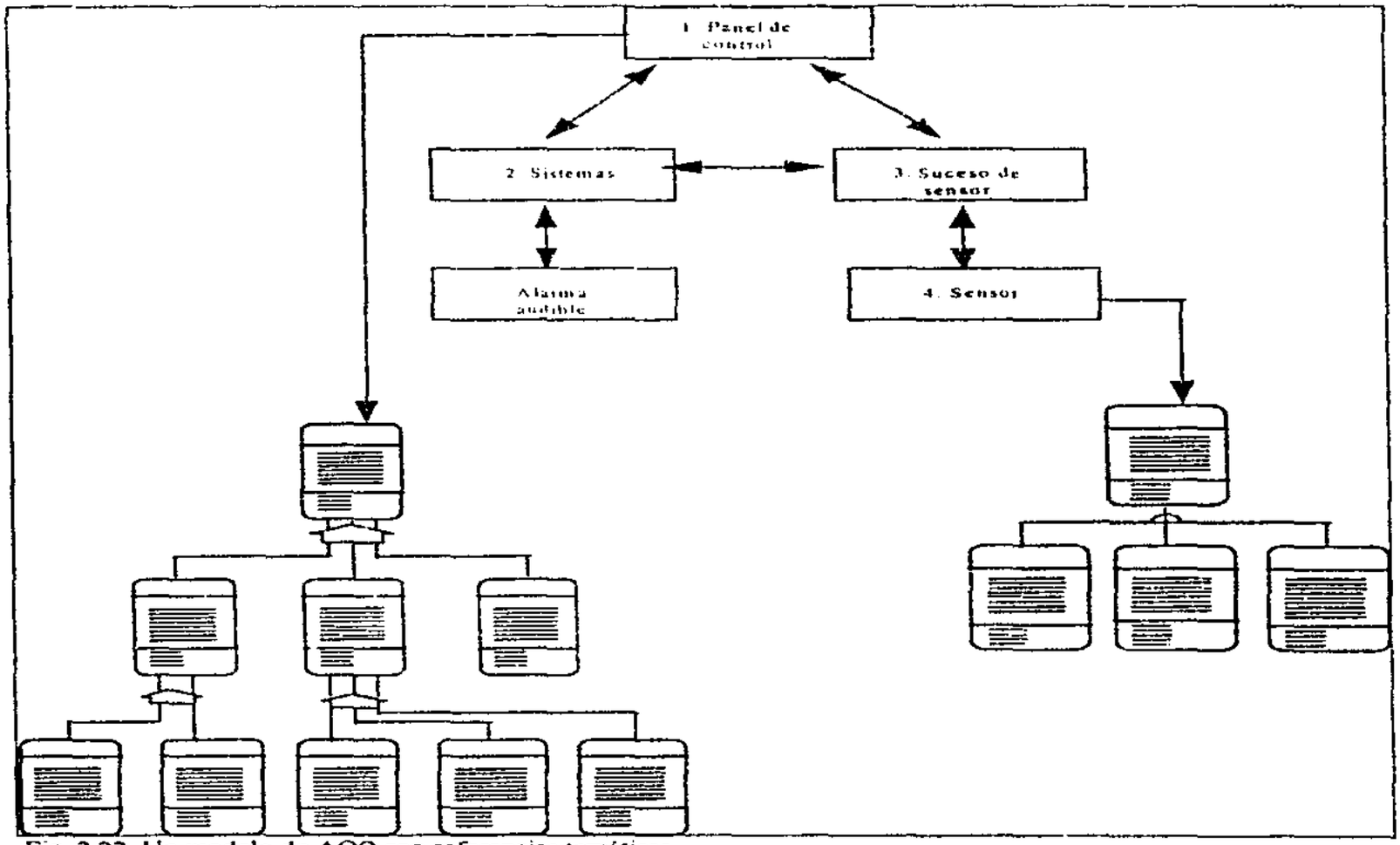


Fig. 2.22. Un modelo de AOO con referencias temáticas.

Las líneas bidireccionales de la parte superior de la Fig. 2.22 representan caminos de comunicación (mensajes) entre los objetos. Se derivan como una consecuencia de las actividades que se describen en la siguiente sección.

Conexiones de instancias y caminos de mensajes

Los objetos no existen en una burbuja, razón por la que el analista debe definir relaciones para cada objeto del modelo. Una conexión de instancia es una notación de modelado que indica que existe alguna relación importante. Por ejemplo, los senderos pueden generar sucesos de senderos y estos deben ser reconocidos por el sistema. Una vez que se han establecido las líneas, se evalúa cada extremo para determinar si lo que existe es una conexión simple (1:1) o una conexión múltiple (1: muchos). Para una conexión 1:1, la línea lleva una barra. Una conexión 1: muchos, aparece con el símbolo de triple bifurcación, que se puede ver en la Fig. 2.23b. Finalmente, se añade otra barra vertical si la conexión es obligada y un círculo si lo conexión es opcional. Por ejemplo, por lo menos un sensor y probablemente muchos senderos pueden llevar a un suceso de sensor (p. ej.: se detecta humo o se produce una rotura) en HogarSeguro. En cada situación concreta, se producirá 0 ó 1 suceso de sensor. Para cada conexión de instancia se sigue este tipo de evaluación. En la Fig. 2.23c se muestra la representación resultante.

Al representar las conexiones de instancias, el analista añade otra nueva dimensión al modelo AOO. No sólo se han identificado las relaciones entre objetos, sino que se han definido también los caminos de mensajes importantes. En la explicación de la Fig. 2.22 hicimos referencia a las fechas punteadas que conectaban los símbolos temáticos. Se trata de caminos de mensajes (a veces referidos como conexiones de mensajes): cada fecha implica un intercambio de mensajes entre objetos del modelo. Como ya señalamos anteriormente, los mensajes se moverán a través de los caminos de conexión de instancias; por tanto, las conexiones de mensajes se derivan directamente de las conexiones de instancias.

Tabla : Especificación de objeto

I. Nombre del objeto
II. Descripción de atributos
    Nombre del atributo
    Contenido del atributo
    Tipo /estructura de datos del atributo
I. Entrada externa al objeto
II. Salida externa del objeto
III. Descripción de operaciones
    Nombre de la operación
    Descripción de interfaz de la operación
    Descripción del procesamiento de la operación
    Aspectos de rendimiento
    Restricciones y limitaciones
I. Conexiones de instancia
II. Conexiones de mensajes

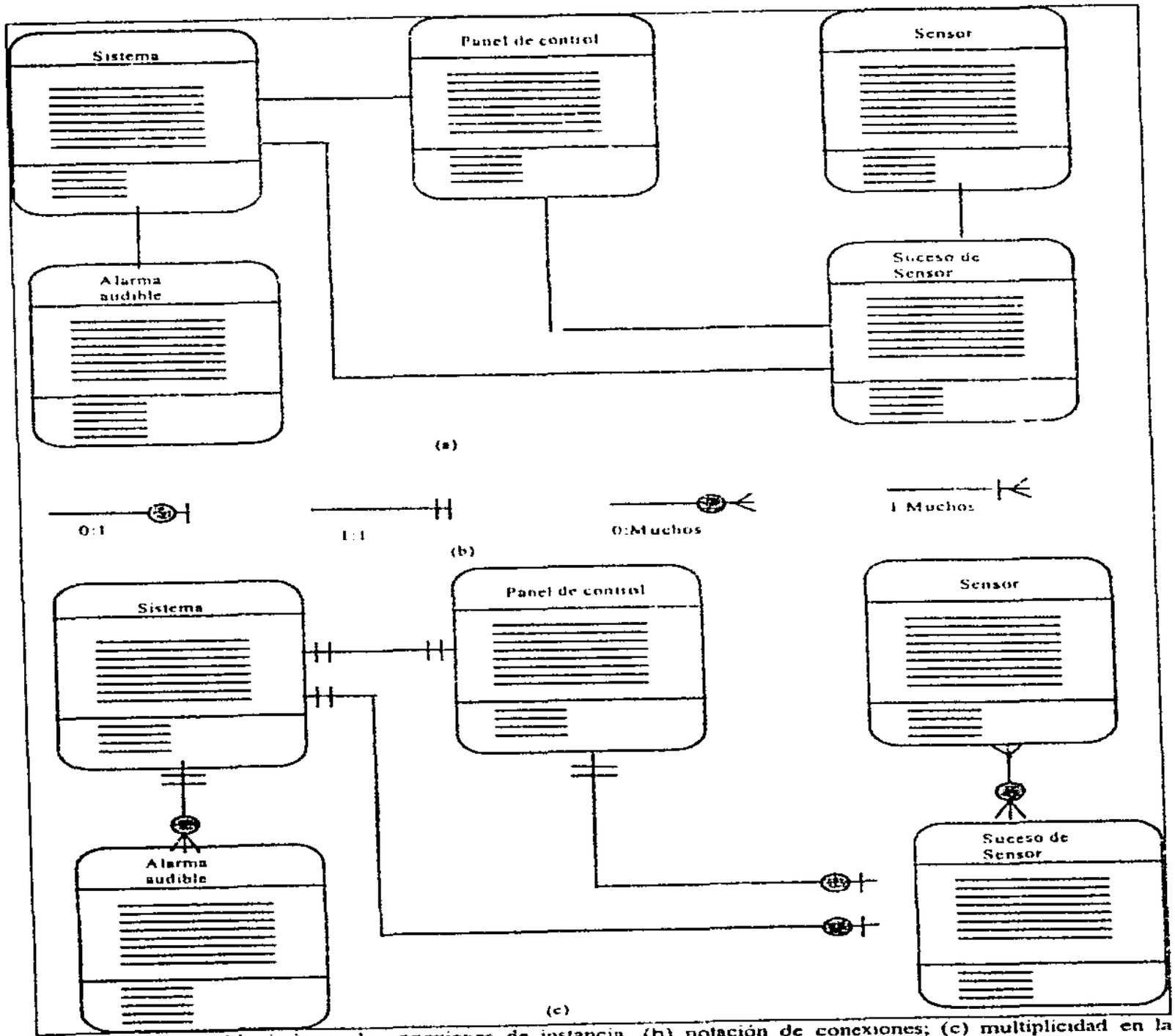


Fig. 2.23. (a) Establecimiento de conexiones de instancia, (b) notación de conexiones; (c) multiplicidad en la definición.

Al igual que los diagramas de análisis estructurado, la representación gráfica es la base del modelo de AOO y proporciona un excelente punto de apoyo para la creación de una especificación de requisitos del software. También se debe describir, mediante un formato textual, cada objeto representado gráficamente. En la tabla "Especificación de Objeto" se muestra una sugerencia de esquema para la descripción de objetos.

AOO y prototipos

Es importante indicar que el uso del AOO puede llevar a una creación de prototipos extremadamente efectiva, así como a técnicas de "ingeniería del software evolutiva". Se pueden catalogar en una biblioteca los objetos que han sido especificados y posteriormente implementados para el proyecto actual. Los objetos son inherentemente reutilizables y con el tiempo crecerá la biblioteca de objetos reusables. Cuando se aplica el AOO a nuevos proyectos, el analista puede trabajar en su especificación del sistema utilizando objetos existentes (implementados) que están contenidos en la biblioteca de objetos, en lugar de inventar objetos nuevos (esta práctica es muy común en el diseño de hardware; los ingenieros de sistemas especifican circuitos integrados existentes, en lugar de crear un diseño a medida). Por ejemplo, se pueden usar los objetos sensor y suceso de sensor descritos para HogarSeguro, sin cambios, en muchas aplicaciones de control y monitoreo de procesos. De forma similar, el objeto panel de control puede tener una gran aplicación en otros productos.

Usando objetos existentes durante el análisis, el tiempo de especificación se reduce substancialmente, pudiéndose crear rápidamente un prototipo del sistema especificado que pueda ser revisado por el cliente. Lo que es más importante, el prototipo puede evolucionar hasta un producto o sistema operativo debido a que los bloques constitutivos que lo conforman (los objetos) ya han sido probados en la práctica.

Modelado de Datos

Los conceptos de AOO presentados en las secciones anteriores son realmente extensiones de una técnica de análisis que se ha usado durante muchos años en aplicaciones de procesamiento masivo de datos. Esta técnica, denominada modelado de datos o modelado de información, se centra unicamente en los datos (y, por tanto, satisface uno de los principios fundamentales del análisis), representando una "red de datos" existente para su sistema dado. El modelado de datos es útil para aplicaciones en las que los datos y las relaciones que gobiernan los datos son complejos. A diferencia del enfoque de análisis estructurado, el modelado de datos considera los datos independientemente del procesamiento que transforma los datos.

La terminología del modelado de datos y algunas de sus notaciones gráficas son similares a las usadas en el AOO. Pero es importante reconocer ambos enfoques como distintos, con un punto de vista bastante diferente. Ambos usan el termino "objeto" pero su definición es mucho más limitada en el contexto del modelado de datos. Ambas describen relaciones entre los objetos, pero a modelado de datos no le preocupa cómo se establecen esas relaciones. Modelado de datos hace exactamente lo que su nombre indica -modelo de datos- sin molestarse por el procesamiento que se debe aplicar para transformar los datos. Es, por tanto, una técnica complementaria que sirve para una función de análisis especifica. Se puede cambiar con otro enfoque de modelado

que considere los aspectos de procesamiento, para formar un método de análisis de requisitos completo.

El modelado de datos se usa ampliamente en aplicaciones de bases de datos. Proporciona al analista y al diseñador de la base de datos una amplia visión de los datos y de las relaciones que gobiernan los datos. Dentro del contexto del análisis estructurado, se puede usar el modelado de datos para representar el contenido de los almacenes de datos y de las relaciones que existen entre ellos. Por ejemplo, un elemento de datos de una estructura de archivo (un almacén de datos) puede ser un puntero a otra estructura de archivo (un almacén de datos diferente). La notación del análisis estructurado no proporciona mecanismo directos para representar este tipo de relaciones.

Objetos de datos, atributos y relaciones

Cuando se usa el modelado de datos como técnica de análisis de requisitos, el analista comienza creando un modelo para los objetos de datos. Un objeto de datos se define de forma muy parecida a cómo se ha definido un objeto en AOO.

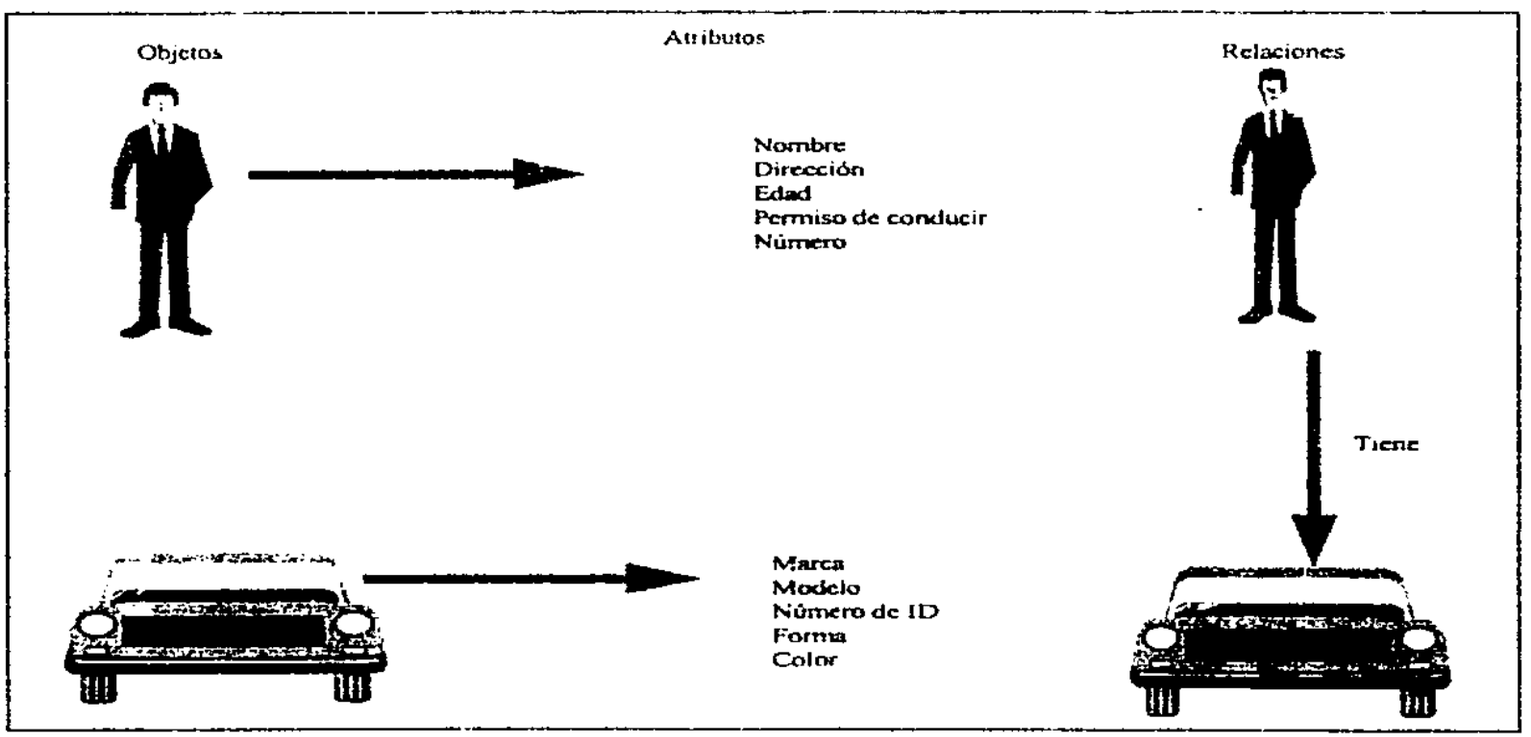


Fig. 2.24. Objetos de datos, atributos y relaciones.

Recordando la anterior explicación de los objetos, los objetos de datos pueden ser entidades externas, cosas, ocurrencias o sucesos, papeles, unidades organizativas, lugares o estructuras. Por ejemplo, una persona o un coche (Fig. 2.24) pueden ser vistos como objetos de

datos en el sentido de que pueden ser definidos en términos de conjuntos de atributos. Los objetos de datos están relacionados entre sí. Por ejemplo, una persona pude tener un coche, de forma que la relación "tener" denota una conexión específica entre la persona y el coche. Siempre es el contexto del problema que se está analizando el que define las relaciones.

Un objeto de datos no toma la misma forma que un objeto de los presentados anteriormente en este capítulo. Sólo encapsula datos - dentro de un objeto de datos no hay referencias a las operaciones que actúan sobre los datos. Por tanto, el objeto de datos puede ser representado como una tabla, tal como muestra la Fig. 2.25. La cabecera de la tabla refleja los atributos del objeto. En este caso, un coche se define en términos de la marca, el modelo, el número (ID#), la forma, el color y el propietario. El cuerpo de la tabla representa instancias específicas del objeto de datos. Por ejemplo, un Corvette de Chevy es una instancia del objeto de datos coche.

Los atributos de los objetos de datos los caracterizan de tres formas diferentes. Se puede usar para (1) dar nombre a una instancia del objeto de datos, (2) describir la instancia, o (3) hacer referencia a otra instancia de otra tabla. Además, se pueden definir uno a más atributos como identificador - es decir, el atributo identificador se convierte en una "clave" que se utilizará para encontrar una instancia del objeto de datos. En algunos casos, los valores de los identificadores son únicos, aunque no se trata de un requisito. Refiriéndonos al objeto de datos coche, un identificador razonable podría ser el número (ID#). Los atributos de los objetos de datos se determinan exactamente de la misma forma que se ha sugerido para los atributos definidos como parte de análisis orientado a los objetos.

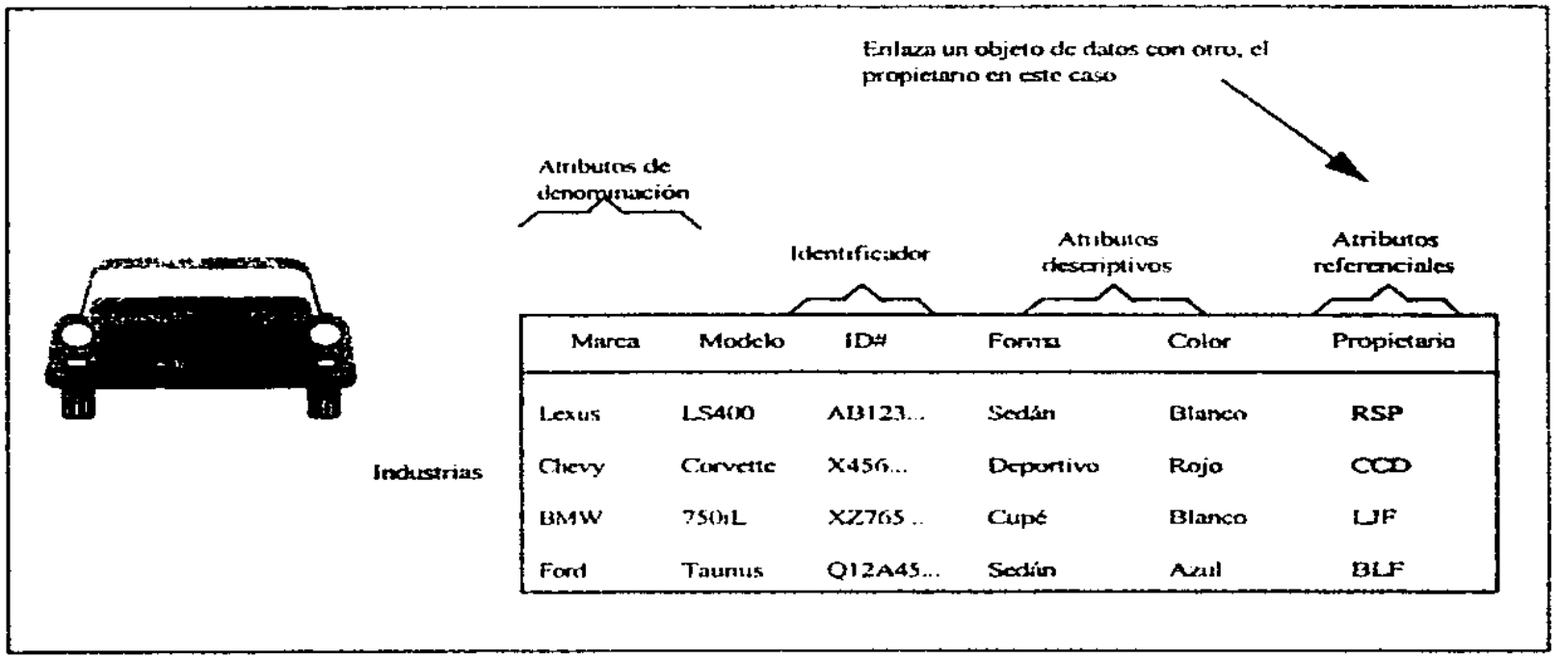


Fig. 2.25, Los objetos de datos como tablas

Se pueden "formalizar" las tablas de los objetos de datos aplicando una serie de reglas de normalización que producen un modelo relacional de los datos. Cuando se aplican sobre las tablas de objetos de datos las reglas de normalización, se reduce al mínimo la redundancia - o sea, se minimiza la cantidad de información que se necesita mantener para satisfacer un problema concreto. Se pueden resumir esas reglas de normalización de la siguiente forma:

1. Una instancia dada de un objeto de datos tiene un valor y sólo uno para cada atributo.
   Por ejemplo, esta regla requiere que en la tabla de choches (Fig. 2.25) sólo haya un propietario por cada instancia de coche. Si el Corvette de Chevy tuviera dos propietarios, habría que rediseñar el objeto de datos. Adicionalmente, se asume que la tabla tiene una entrada por cada atributo No debe haber "huecos" en la tabla. Esta regla nos ayuda a definir una relación, usando la jerga de los diseñadores de bases de datos.

2. Los atributos representan elementos de datos elementales; no contienen estructuras internas.
   Por ejemplo, si se hubiera definido forma como

   forma = número de puertas + altura + anchura + nombre
   nombre = coupe / sedán /sport /lujo

   tendríamos una violación de esta regla.
   Para resolver una situación en la que un atributo fuera un elemento de datos, tendríamos que definir la tabla de forma que cada uno de los componentes de forma estuvieran como atributos separados.

3. Cuando se usa más de un atributo para identificar un objeto de datos, hay que asegurarse de que los atributos descriptivos y referenciales representen "una característica de objeto completo y no una característica de algo que pueda ser identificado sólo por un parte del identificador.

Por ejemplo, supongamos que añadimos un nuevo identificador, venta, a la tabla de coche de la Fig. 2.25. Si continuamos modificando la tabla (la descripción del objeto de datos) y añadimos lugar de venta como atributo descriptivo, estaremos violando la regla 3, ya que lugar de ventas es un atributo de uno sólo de los identificadores - venta. No es un atributo de ID#, el otro identificador de coche.
Si se diseñan las tablas de acuerdo con estas tres reglas, se dice que el modelo de datos resultante está en la segunda forma normal.

4. Todos los atributos que no sean identificadores deben representar alguna característica de la instancia nombrada por el identificador del objeto de datos y describir algún otro atributo que no sea un identificador. Por ejemplo, si se añadiera un atributo nombre de pintura al objeto de datos coche, tendríamos una violación de la regla 4. Nombre de pintura es una característica de color, no de la instancia de coche identificada por el ID#.

Diagramas de entidad-relación
La notación principal del modelado de datos es el diagrama de entidad-relación (E-R). Varios autores han contribuido al desarrollo de la notación E-R. Todos ellos identifican un conjunto de componentes primarios para los diagramas E-R, objetos de datos, atributos, relaciones y varios

indicadores de tipo. El principal propósito del diagrama E-R es representar los objetos de datos y sus relaciones.
La notación de diagrama E-R es relativamente sencillo. Los objetos de datos se representan como rectángulos etiquetados. Las relaciones se indican mediante rombos. Las conexiones entre los objetivos de datos y las relaciones se establecen mediante variadas líneas especiales de conexión.
La relación entre los objetos de datos coche y fabricante se representaría como se muestra en la figura. Un fabricante construye uno o muchos coches. Dentro del contexto que implica el diagrama E-R, la especificación del objeto de datos coche (tabla del objeto de datos de la Fig. 2.26) sería radicalmente diferente de la especificación anterior (Fig. 2.26): Las líneas de conexión tienen el mismo significado que las presentadas en la Fig. 2.23b.Expandiendo el modelo, podemos representar un diagrama E-R excesivamente simplificado (Fig. 2.27) de los elementos de distribución del mercado del automóvil. Se han introducido nuevos objetos de datos, transportista y distribuidor. Además, aparecen nuevas relaciones - transporta, contrata, licencia y almacena - indicando cómo están asociados entre sí los objetos de datos que se muestran en la figura. Se deberían desarrollar las tablas de cada uno de los objetos de datos contenidos en el diagrama E-R de acuerdo con las reglas introducidas.

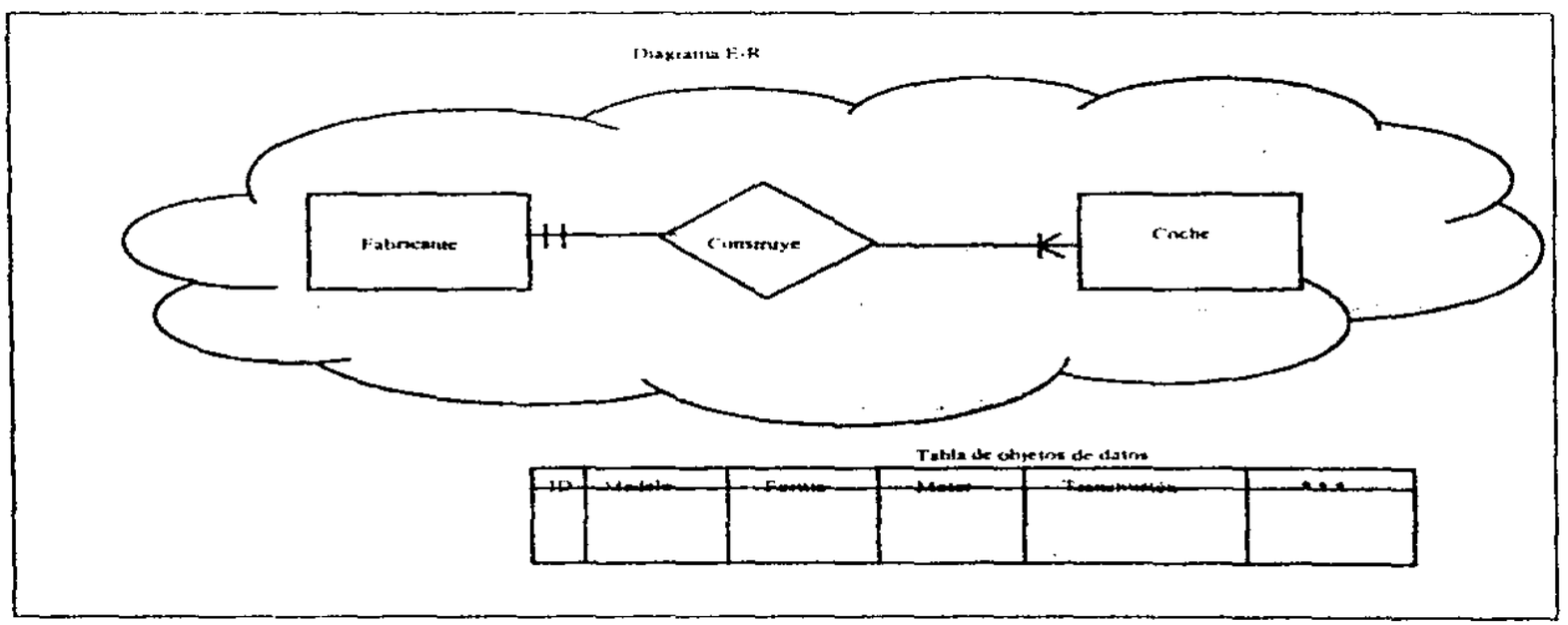


Fig. 2.26. Un sencillo diagrama E-R y una tabla de objetos.

Además de utilizar la notación básica del diagrama E-R presentada en las Fig. 2.26 y 2.27, el analista puede representar jerarquías de tipos de objetos de datos. Son análogas a la notación de estructura de los objetos del AOO que se presentó anteriormente. Por ejemplo, se puede categorizar el objeto de datos coche como nacional, europeo u oriental. La notación E-R que se muestra en la Fig. 2.28 representa esta categorización en forma de jerarquía.

La notación E-R proporciona un mecanismo para representar la asociación entre los objetivos. La Fig. 2.29 muestra cómo se presenta un objeto de datos asociativo. En la figura, los

objetos de datos que modelan los subsistemas individuales están asociados como uno con el objeto coche.

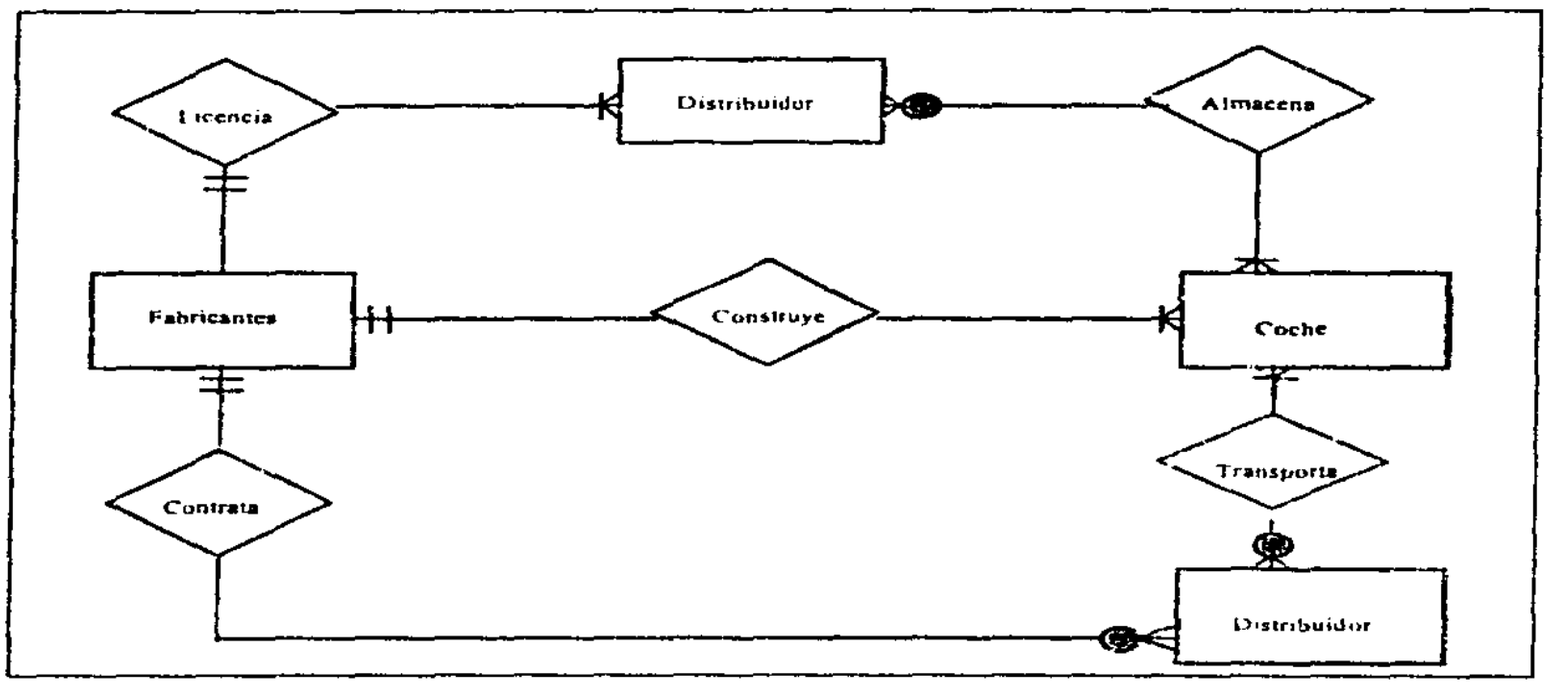


Fig. 2.27. Un diagrama E-R expandido

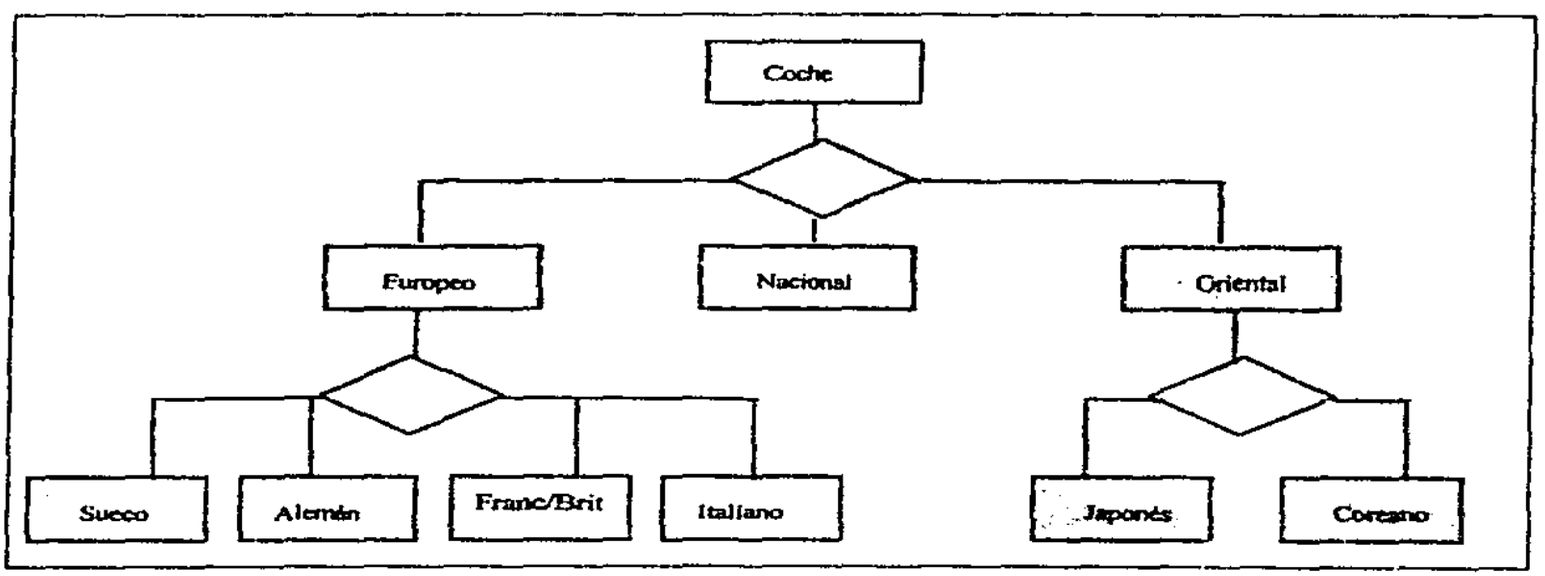


Fig. 2.28. Jerarquías de tipos de objetos de datos

El modelado de datos y el diagrama de entidad-relación se convierten para el analista en una notación concisa para examinar los datos dentro del contexto de la aplicación de procesamiento de datos. En la mayoría de los casos, se usa el enfoque de modelado de datos conjuntamente con el análisis estructurado, aunque se puede usar también para el diseño de bases de datos y como soporte de cualquier otro método de análisis de requisitos.

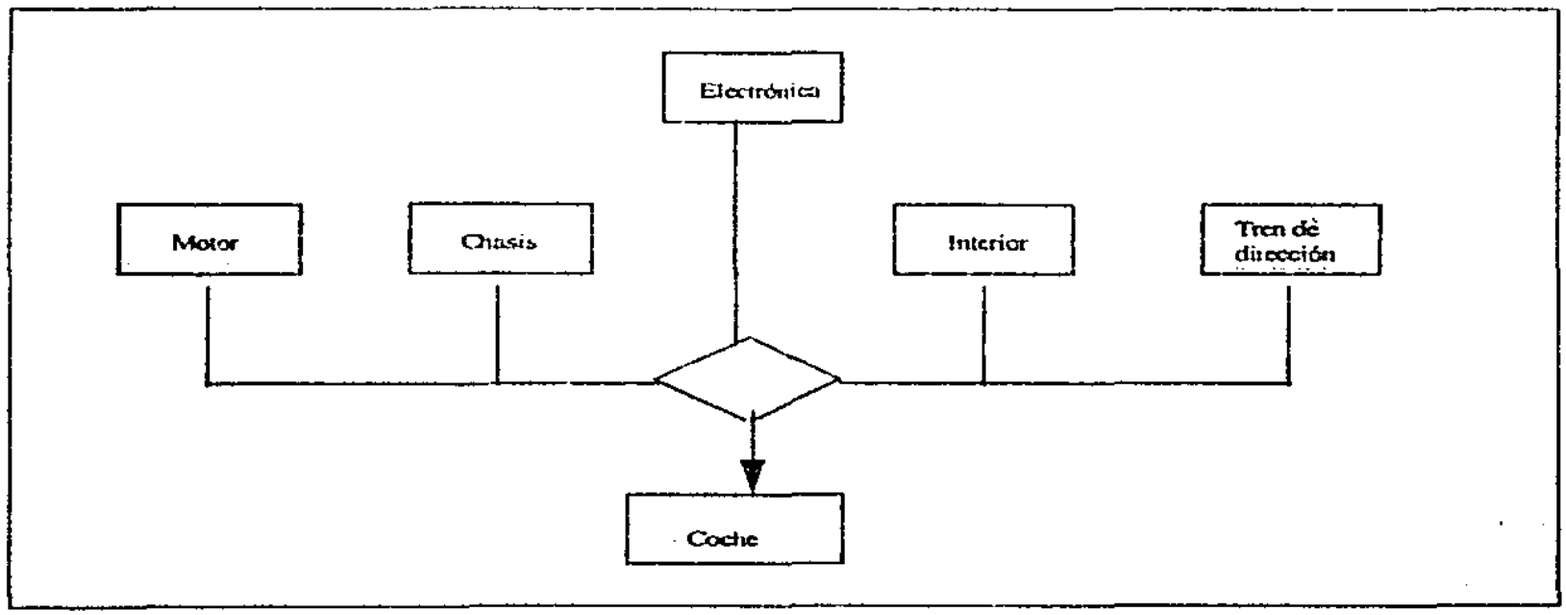


Fig. 2.29. Asociación de objetos

2.5 Fundamentos del Diseño del Software

El diseño es el primer paso de la fase de desarrollo de cualquier producto o sistema de ingeniería. Puede definirse 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 de representación de una entidad que se construirá más adelante. El proceso por el cual se desarrolla el modelo combina: la intuición y los criterios con base 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 permiten discernir sobre la calidad y un proceso de interacción que conduce finalmente a una representación del diseño final.

El diseño de software para computadoras, al igual que los métodos de diseño de ingeniería de otras disciplinas, cambia continuamente conforme aparecen nuevos métodos, mejores análisis y un conocimiento más amplio. A diferencia del diseño mecánico o electrónico, el diseño de software está en una etapa relativamente temprana de su evolución. Nos hemos tomado en serio el diseño del software (en vez de "programación" o "escritura de código") desde hace poco más de tres décadas. Por tanto, a las metodologías de diseño de software les falta profundidad, flexibilidad y la naturaleza cuantitativa asociada normalmente con las disciplinas de diseño de ingeniería más clásicas. Sin embargo, existen técnicas para el diseño de software, están disponibles criterios para calificar un diseño y pueden aplicarse notaciones de diseño.

Ingeniería del Software y Diseño del Software

El diseño del software se asienta en el núcleo técnico del proceso de ingeniería del software y se aplica independientemente del paradigma de desarrollo utilizado. Una vez que se han establecido los requisitos del software, el diseño del software es la primera de tres actividades técnicas -diseño, codificación y prueba. Cada actividad transforma la información de forma que finalmente se obtiene un software para computadora validado.El la Fig. 2.30 se muestra el flujo de información durante la fase de desarrollo. Los requisitos del programa, establecidos mediante los modelos de información, funcional y de comportamiento, alimentan el paso de diseño. Mediante alguna de las metodologías 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 las estructuras de datos que se van a requerir para implementar el software. El diseño arquitectónico define las relaciones entre los principales elementos estructurales del programa. El diseño procedimental transforma los elementos estructurales en una descripción procedimental de software. Se genera el código fuerte y, para integrar y validar el software, se llevan a cabo las pruebas.

Las fases de diseño, codificación y prueba absorben el 75 por 100 o más del costo de la ingeniería del software (excluyendo el mantenimiento). Es aquí donde se toman decisiones que

afectarán finalmente el éxito de la implementación del programa y, con igual importancia, a la facilidad de mantenimiento que tendrá el software.

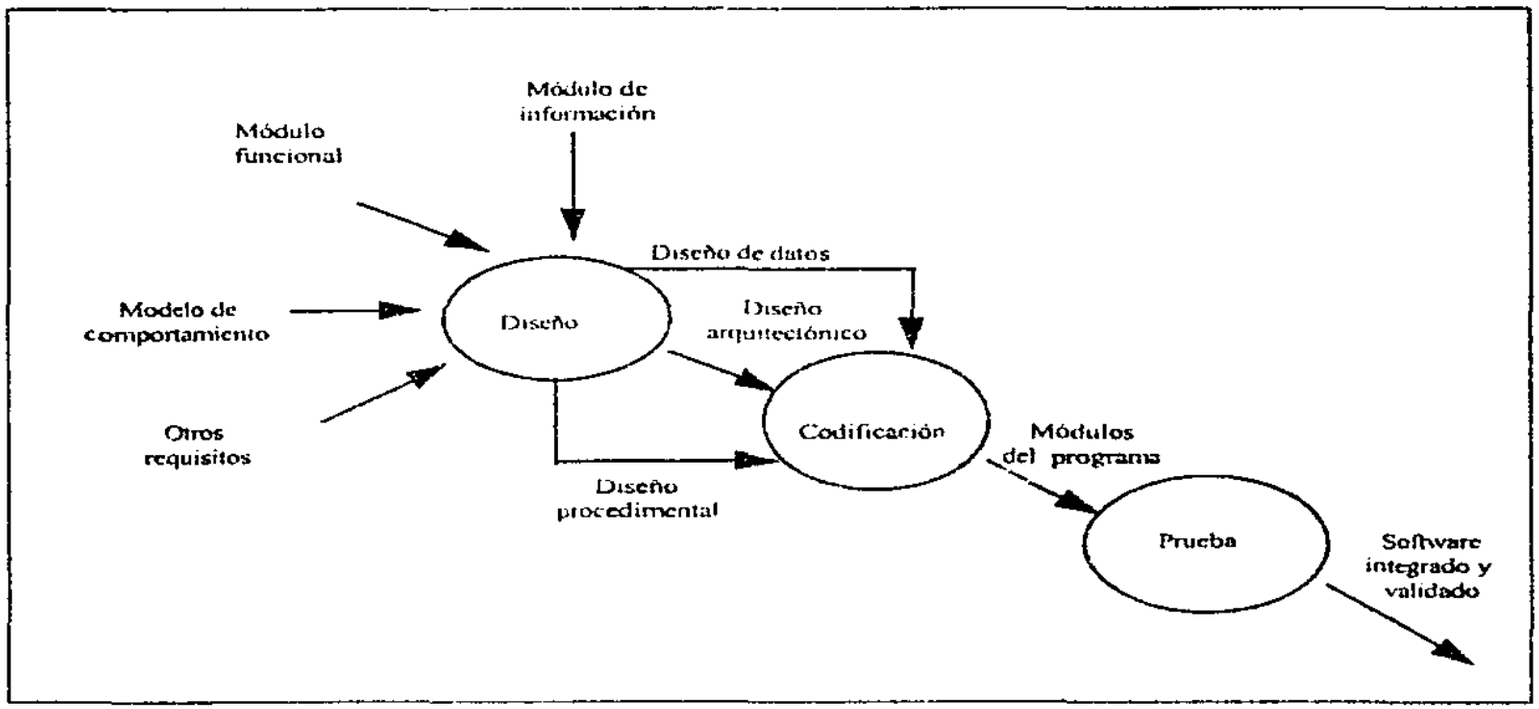


Fig. 2.30. Diseño del software e ingeniería del software.

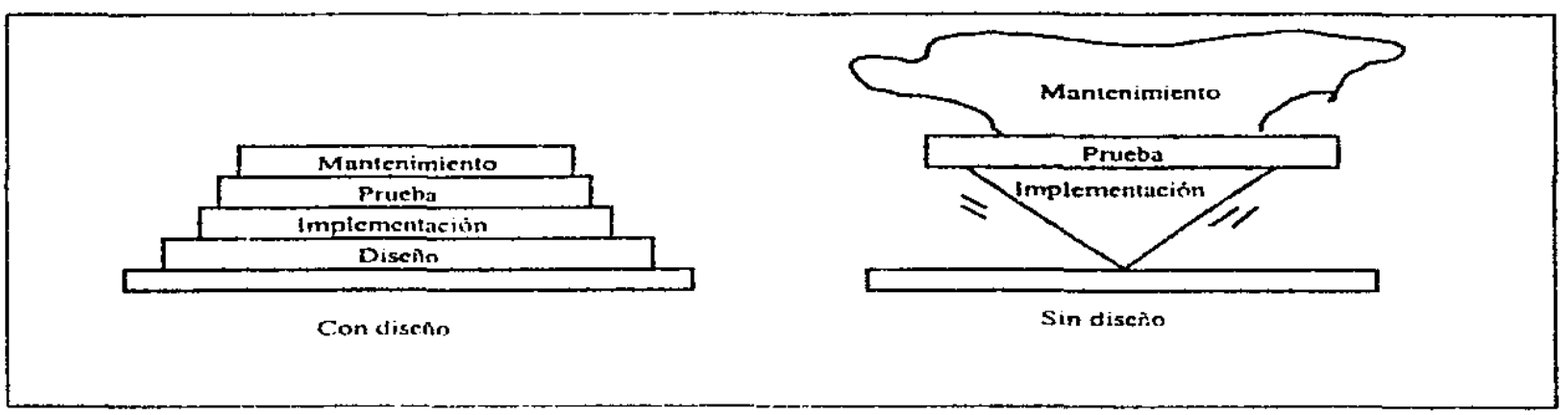


Fig. 2.31. Importancia del diseño

Estas decisiones se llevan a cabo durante el diseño de software, haciendo que sea un paso fundamental de la fase de desarrollo.

La importancia del diseño del software se puede sentar con una única palabra - calidad. El diseño es el proceso en el que se asienta la calidad del desarrollo del software. El diseño produce las representaciones del software de las que puede evaluarse su calidad. El diseño es la única forma mediante el cual podemos traducir con precisión los requisitos del cliente en un producto o sistema acabado. El diseño de software sirve como base (Fig. 2.31) de todas las posteriores etapas del desarrollo y de la fase de mantenimiento. Sin diseño, nos arriesgamos a construir un sistema inestable, un sistema que falle cuando se realicen pequeños cambios; un sistema que pueda ser difícil de probar; un sistema cuya calidad no pueda ser evaluada hasta más adelante en el proceso de ingeniería del software, cuando quede poco tiempo y se haya gastado ya mucho dinero.

El proceso del diseño.

El diseño del software es un proceso mediante el que se traducen los requisitos en una representación del software. Inicialmente, la representación describe una visión holística del software. Posteriores refinamientos conducen a una representación de diseño que se acerca mucho al código fuente.

Desde el punto de vista de la gestión del proyecto, el diseño del software se realiza en dos pasos. El diseño preliminar se centra en la transformación de los requisitos en los datos y la arquitectura del software. El diseño detallado se ocupa del refinamiento de la representación arquitectónica que lleva a una estructura de datos detallada y a las representaciones algorítmica del software.

Dentro del contexto de los diseños preliminar y detallado, se llevan a cabo varias actividades de diseño diferentes. Además del diseño de datos, del diseño arquitectónico y del diseño procedimental, muchas aplicaciones modernas requieren una actividad distinta de diseño de la interfaz. El diseño de la interfaz establece la disposición y los mecanismos para la interacción hombre-máquina. Las relaciones entre las vertientes técnica y de gestión del diseño se muestran en la Fig. 2.32

Diseño Orientado a Objetos

El diseño orientado a los objetos (DOO), al igual que otras metodologías de diseño orientadas a la información, crea una representación del campo del problema del mundo real y la hace corresponder con el ámbito de la solución, que es el software. A diferencia de otros métodos, el DOO produce un diseño que interconecta objetos de datos (elementos de datos) y operaciones de procesamiento en una forma que modulariza la información y el procesamiento, en lugar de dejar aparte el procesamiento. La naturaleza única del diseño orientado a los objetos queda reflejada en su capacidad de construir sobre tres pilares conceptuales importantes del diseño de software: abstracción, ocultamiento de información y modularidad. Todos los métodos de diseño intentan desarrollar software con esas tres características fundamentales, pero sólo el DOO proporciona un mecanismo que permite el diseñador conseguir las tres sin complejidad ni necesidad de compromisos.

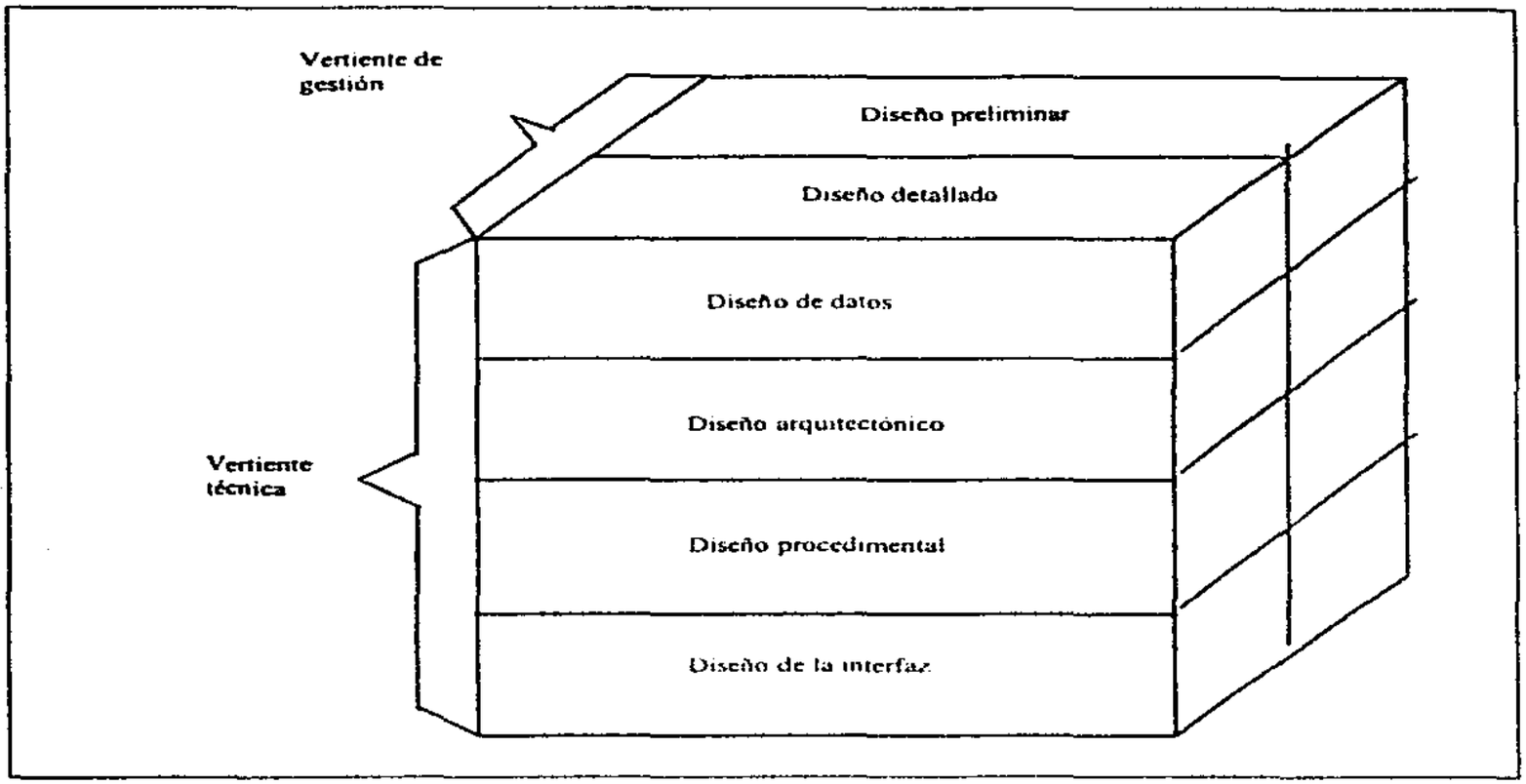


Fig. 2.32. Relaciones entre las vertientes técnica y de gestión del diseño.

Wiener y Sincovec resumen la metodología DOO de la siguiente forma:

Ya no es necesario que el diseñador del sistema haga corresponder el ámbito del problema con estructuras de datos y de control predefinidas que se encuentren en el lenguaje de implementación. En cambio, el diseñador puede crear sus propios tipos abstractos de datos y abstracciones funcionales y hacer corresponder el cambio del mundo real con esas abstracciones creadas por el propio programador. Esta correspondencia será la mayoría de las veces mucho más natural, ya que el rango de tipos abstractos de datos que puede inventar el diseñador es virtualmente ilimitado. Más aún, el diseño del software se desliga de los detalles de representación de los objetos de datos que se usan en el sistema. Así, se pueden cambiar muchas veces esos detalles de representación, sin que ello afecte al sistema de software global.

El análisis orientado a los objetos, el diseño orientado a los objetos y la programación orientada a los objetos comprenden un conjunto de actividades de ingeniería del software para la construcción del sistema orientado a los objetos. Como ocurría con el AOO, es prematuro decir que el diseño orientado a los objetos es un método maduro. Sin embargo, a medida que sigue creciendo la popularidad del enfoque orientado a los objetos se impone la utilización de métodos de diseño para crear sistemas orientados la los objetos.

2.3 Diseño Orientado a los Objetos

Orígenes del Diseño Orientado a los Objetos

Los objetos y las operaciones no son conceptos nuevos de programación, pero el diseño orientado a los objetos sí lo es. En los primeros días de la informática, los lenguajes ensambladores permitían a los programadores utilizar instrucciones máquina (operadores) para manipular elementos de datos (operandos). El nivel de abstracción que se aplicaba al ámbito de la solución era muy bajo.

Al aparecer los lenguajes de programación de alto nivel (p. ej.: FORTRAN, ALGOL, COBOL), se pudieron modelar los objetos y las operaciones del espacio del problema del mundo real con estructuras de datos y de control predefinidas que estaban disponibles como parte del lenguaje de alto nivel.

Durante los años setenta, aparecieron conceptos como la abstracción y el ocultamiento de información, emergiendo los métodos de diseño conducidos por los datos, aunque los desarrolladores de software todavía se centraban en los procesos y sus representaciones. Al mismo tiempo, los lenguajes de alto nivel (p.ej.: Pascal) introdujeron una variedad mucho más rica de estructuras y tipos de datos.

Mientras los lenguajes de alto nivel convencionales (lenguajes surgidos de la tradición de FORTRAN y ALGOL) evolucionaban, los investigadores trabajaban duramente en una nueva clase de lenguajes de simulación y de creación de prototipos, como SIMULA y Smaltalk. En esos lenguajes, el énfasis estaba en la abstracción de datos, y los problemas del mundo real eran representados por un conjunto de objetos de datos para los que se adjuntaba un conjunto correspondiente de operaciones. El uso de esos lenguajes era radicalmente diferente del uso de los lenguajes convencionales.

El enfoque que denominamos diseño orientado a los objetos ha evolucionado a lo largo de los últimos 20 años. Los primeros trabajos sentaron las bases al establecer la importancia de la abstracción, del ocultamiento de información y de la modularidad para la calidad del software.
Durante los años ochenta, la rápida evolución de los lenguajes de programación Smaltalk y Ada, seguida de un crecimiento explosivo del uso de los dialectos orientados a los objetos de C como C++ y Objetive-C, produjeron un interés inusitado en el DOO. En un temprano tratamiento de los métodos para conseguir el diseño orientado a los objetos, Abbott mostró "cómo el análisis de la descripción del problema y su solución en lenguaje natural se podía usar como guía para el desarrollo de la parte visible de un paquete útil. Un paquete contiene tanto datos como los procedimientos que operan sobre ellos, y el algoritmo concreto para un problema dado". Booch amplió a finales de los ochenta, el DOO se usaba en el diseño de aplicaciones de software que iban desde las animaciones gráficas en computadora hasta las telecomunicaciones. Actualmente, muchos creen que con la llegada del nuevo milenio, los métodos y los lenguajes de programación orientados a los objetos serán los que predominen.

Conceptos del Diseño Orientado a los Objetos
Al igual que otros métodos de diseño, el DOO introduce un nuevo conjunto de términos, notaciones y procedimientos para la derivación del diseño de software. En esta sección repasamos la terminología orientada a los objetos e introducimos algunos conceptos adicionales que son relevantes para el diseño.

Objetos, operaciones y mensajes
El funcionamiento del software se consigue al actuar sobre una estructura de datos (de un nivel de complejidad variable) uno o más procesos de acuerdo con un procedimiento de invocación definido por un algoritmo estático o por órdenes dinámicas. Para conseguir un diseño orientado a los objetos, debemos establecer un mecanismo para (1) representar la estructura de datos, (2) especificar el proceso y (3) llevar a cabo el procedimiento de invocación.

Un objeto es una componente del mundo real que se ha hecho corresponder con el campo de software. En el contexto de un sistema basado en computadora, un objeto típicamente es un productor o un consumidor de información. Por ejemplo, objetos típicos pueden ser máquinas, órdenes, archivos, monitores, interruptores, señales, cadenas alfanuméricas o cualquier otro lugar, persona, cosa, ocurrencia, papel (de comportamiento) o suceso. Cuando se hace corresponder un objeto con su realización software, consiste en una estructura de datos privada y en procesos, denominados operaciones, que pueden legítimamente transformar la estructura de datos. Las operaciones contienen unas construcciones procedimentales y de control que pueden ser invocadas mediante un mensaje - una petición al objeto para que realice alguna de sus operaciones.

La Fig. 2.33 muestra cómo representamos un objeto. El objeto sistema (parte del producto de seguridad HogarSeguro) muestra su estructura de datos privada y las operaciones asociadas.
El objeto también tiene una parte compartida, que es su interfaz. Los mensajes se mueven por la interfaz y especifican la operación que se desea realizar sobre el objeto, aunque no cómo se ha de realizar la operación. Es el objeto que recibe el mensaje el que determina cómo se implementa la operación requerida.

Al definir un objeto con parte privada y proporcionar mensajes para invocar el procesamiento adecuado, conseguimos el ocultamiento de información - es decir, dejar ocultos para el resto de los elementos del programa los detalles de implementación del objeto. Los objetos con sus operaciones proporcionan una modularidad inherente - es decir, los elementos del software (datos y procesos) están agrupados con un mecanismo de interfaz bien definido (en este caso, los mensajes)

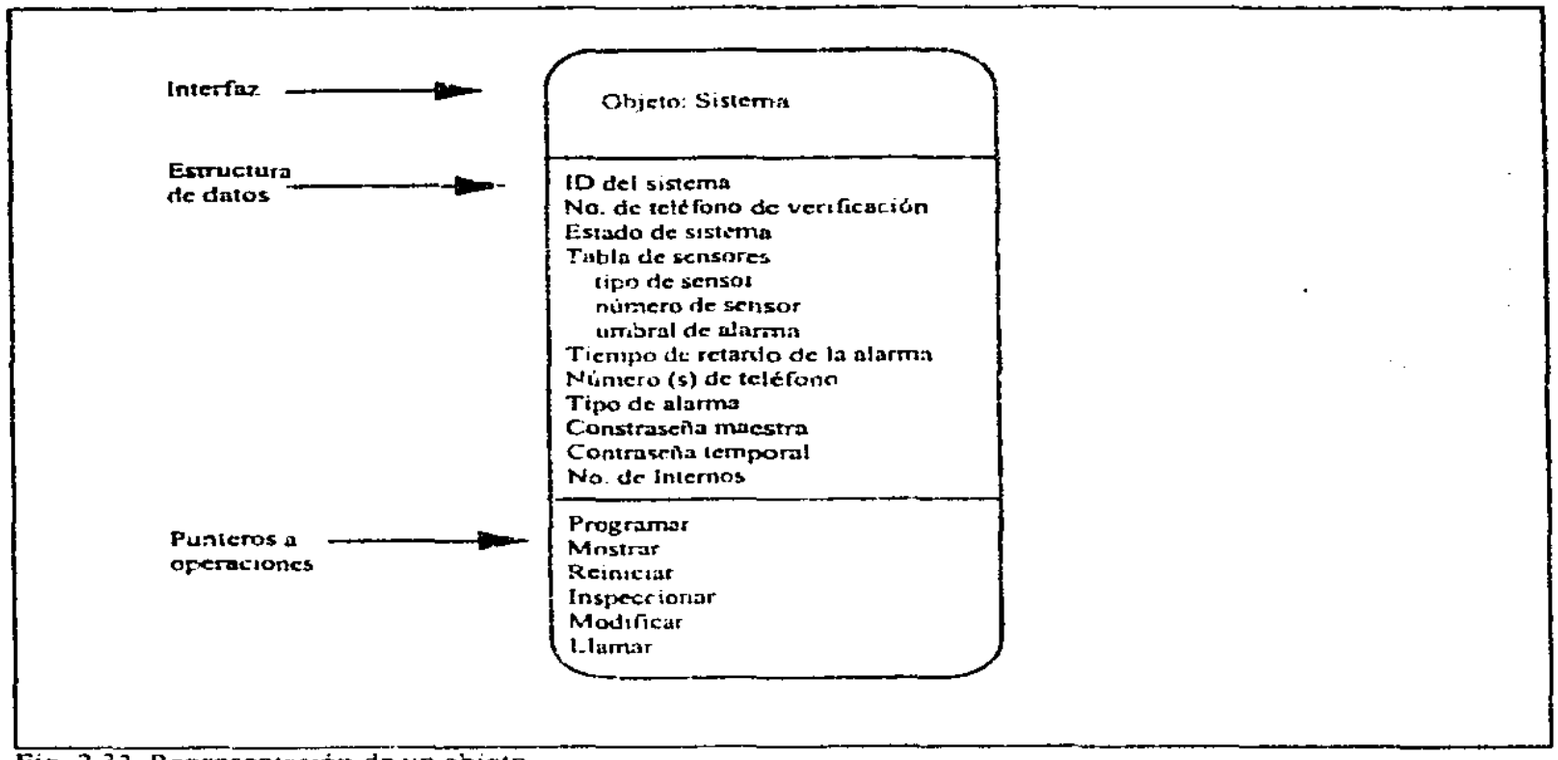


Fig. 2.33. Representación de un objeto.

Aspectos de diseño

Bertrand Meyer sugiere cinco criterios para juzgar la capacidad de un método de diseño de conseguir la modularidad y los relaciona con el diseño orientado a los objetos:

- *Descomponibilidad:* la facilidad con la que un método de diseño ayuda al diseñador a descomponer un gran problema en subproblemas más fáciles de resolver.
- *Componibilidad:* el grado en que un método de diseño asegura que los componentes del programa (módulos), una vez diseñados y construidos, pueden ser reusados para crear otros programas.
- *Comprensibilidad:* la facilidad con la que se puede comprender un componente de un programa sin tener que referenciar otra información ni otros módulos.
- *Continuidad:* la capacidad de realizar pequeños cambios en un programa y que esos cambios se manifiesten por si mismos con sólo unos cambios correspondientes en uno o unos pocos módulos.
- *Protección:* una característica arquitectónica que reduce la propagación de efectos laterales cuando se produce un error en un módulo dado.

A partir de estos criterios, Meyer sugiere la derivación de cinco principios básicos de diseño para arquitecturas modulares: (1) unidades lingüísticamente modulares; (2) pocas interfaces; (3)

interfaces pequeñas (acoplamiento débil); (4) interfaces explícitas; (5) ocultamiento de información.
Los módulos se definen como unidades lingüísticamente modulares cuando "corresponden a unidades sintácticas del lenguaje utilizado". Es decir, el lenguaje de programación que se vaya a utilizar debe soportar directamente la modularidad. Por ejemplo, si el diseñador crea una subrutina, cualquiera de los lenguajes de programación más populares (p. ej.: FORTRAN, C, Pascal) puede implementarla como una unidad sintáctica. Pero, si lo que se define es un paquete que contiene estructuras de datos y procedimientos, identificándolos como una única unidad, sería necesario un lenguaje como Ada (u otro lenguaje orientado a los objetos) para representar directamente ese tipo de módulo con la sintaxis del lenguaje.
Para conseguir un bajo acoplamiento, se debe minimizar la cantidad de información que se mueve a través de una interfaz ("intefaces pequeñas"). Siempre que los módulos deban comunicarse, lo deben hacer de una forma obvia y directa ("interfaces explícitas"). Por ejemplo, si los módulos X e Y se comunican a través de una zona global de datos, violarán el principio de interfaces explícitas, ya que la comunicación entre los módulos no resulta evidente para un observador externo.
Finalmente, conseguimos el principio de ocultamiento de información cuando toda la información de un módulo está oculta para un acceso desde el exterior, a menos que la información sea definida específicamente como <<información pública>>
Los criterios y los principios de diseño presentados en esta sección se pueden aplicar a cualquier método de diseño (p.ej.: los podemos aplicar al diseño estructurado). Sin embargo, como veremos, el método de diseño orientado a los objetos consigue cada uno de esos criterios de forma más eficiente que los otros enfoques, dando como resultado arquitecturas modulares que permiten conseguir los criterios de modularidad de forma más efectiva.

Clases, instancias y herencia

Muchos objetos del mundo físico tienen características razonablemente similares y realizan operaciones razonablemente similares. Si observamos la planta de fabricación de una fábrica de equipos pesados, veremos máquinas fresadoras, taladradoras y perforadas. Aunque cada uno de esos objetos es diferente, todos pertenecer a una clase superior denominada "máquinas herramienta". Todos los objetos de la clase máquinas herramienta tiene atributos comunes (p. ej.: corte, arranque, parada). Por tanto, clasificando un "torno" como miembro de la clase de máquinas herramienta, ya sabemos algo acerca de sus atributos y sus operaciones, incluso sin saber exactamente cuál es su función.

Las realizaciones en software de objetos del mundo real se clasifican de forma muy parecida. Todos los objetos son miembros de una clase mayor y heredan la estructura de datos privada y las operaciones que han sido definidas para esa clase. Dicho de otra forma, una clase es un conjunto de objetos que tienen las mismas características. Así, un objeto es una instancia de una clase mayor. Recordando el sistema de seguridad HogarSeguro presentado anteriormente, llevamos a cabo el análisis orientando a los objetos e identificamos clases candidatas. Una de esas clases es sensor. Para esa definición de clase, podemos instanciar objetos sensores específicos, sensor de entrada, sensor de humo y sensor de movimiento.

TYPE sensor de movimiento IS INSTANCE OF sensor; implica la herencia de los atributos de sensor.

Pero. ¿qué pasa si la herencia no es perfecta? Esta situación se produce cuando una instancia candidata de una clase comparte la mayoría, pero no todos, de los atributos de la clase y requiere todas las operaciones de la clase, así como otras adicionales, que sólo son relevantes para el candidato a miembro. Por ejemplo, supongamos que se va a añadir un nuevo tipo de sensor al sistema HogarSeguro. A diferencia de los sensores de humo, de entrada y de movimiento, el nuevo sensor, es "agresivo", es decir, realiza alguna acción cuando detecta un suceso. Un sensor de fuego alertará al panel de control de HogarSeguro, pero también esparcirá en el aire un producto químico para frenar el avance del fuego, durante un período de tiempo programado. Para dar cabida al sensor de fuego, creamos una subclase - esto es, heredamos los atributos y las operaciones de la clase sensor, pero los modificamos para ajustarlos a las características especiales de sensor de fuego. En la Fig. 2.34 se muestran la clase sensor original y la nueva subclase, denominada sensor agresivo.

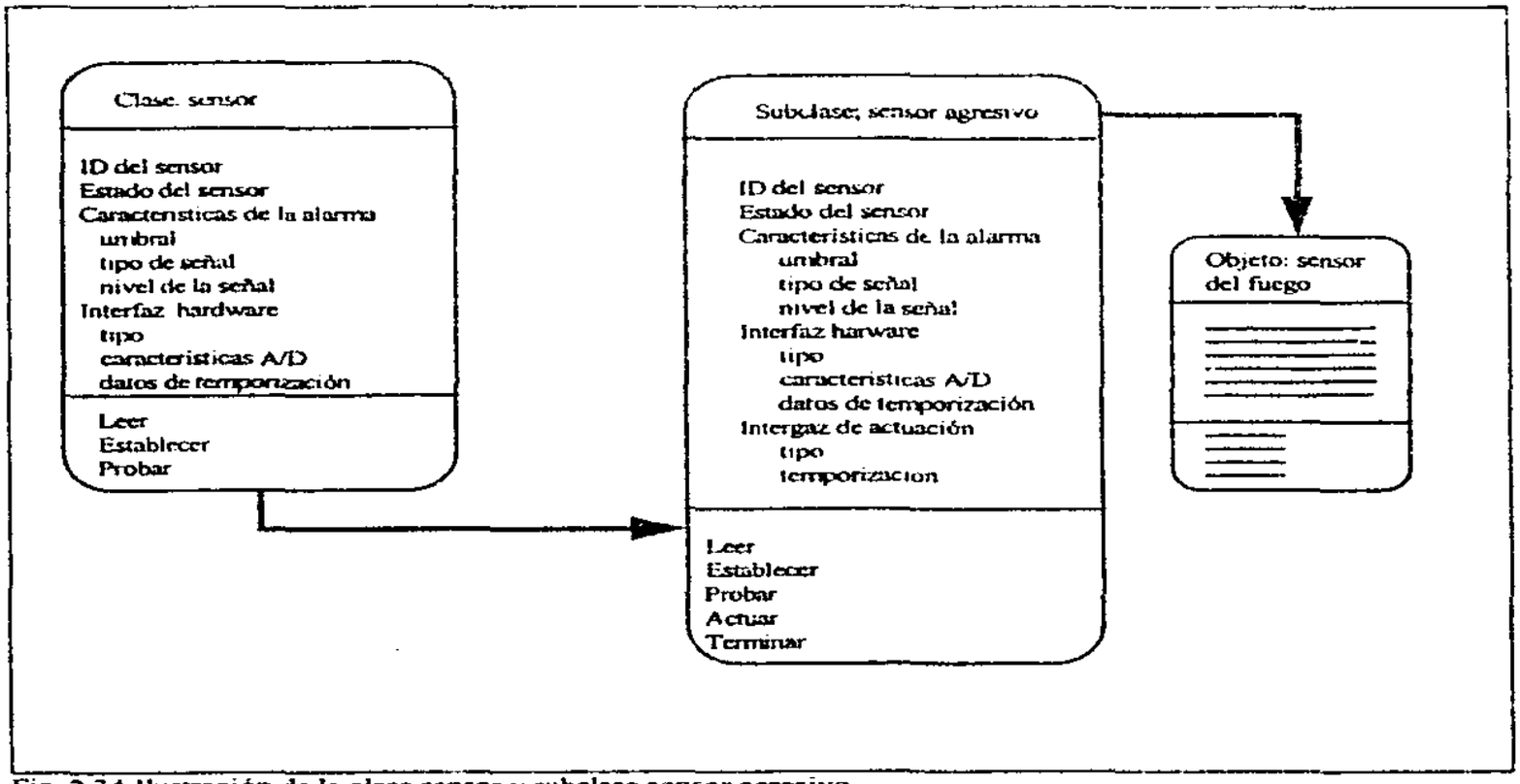


Fig. 2.34 Ilustración de la clase sensor y subclase sensor agresivo.

El uso de clases, subclases y herencia es de crucial importancia en la ingeniería del software moderna. La reutilización de componentes de software (lo que nos permite conseguir la composición) se lleva a cabo creando objetos (instancias) que se forman sobre los atributos y las operaciones existentes heredadas de una clase o de una subclase. Sólo es necesario especificar cómo son las diferencias entre el nuevo objeto y la clase, en lugar de definir todas las características del nuevo objeto.

A diferencia de otros conceptos de diseño que son independientes del lenguaje de programación que se utilice, la implementación de las clases, de las subclases y de los objetos varía de acuerdo con el lenguaje de programación que se utilice. Por esta razón, el anterior tratamiento genérico requerirá ciertas modificaciones para el contexto de cada lenguaje de programación específico. Por ejemplo en Ada, los objetos se implementan como paquetes y las instancias se obtienen mediante el uso de abstracciones de datos y tipos. Por otro lado, en el lenguaje de programación Smaltalk implementan directamente cada uno de los conceptos anteriormente descritos, convirtiéndolo en un lenguaje de programación verdaderamente orientado a los objetos.

Descripciones de los objetos

La descripción de diseño de un objeto (una instancia de una clase o de una subclase) puede tener dos formas distintas:

- Una descripción del protocolo que establece la interfaz del objeto, definiendo cada mensaje que puede recibir el objeto y las operaciones que realiza el objeto cuando recibe el mensaje.
- Una descripción de la implementación, que muestra los detalles de cada operación implicada en la recepción de un mensaje por el objeto. Los detalles de implementación incluyen la información sobre la parte privada del objeto, esto es, los detalles internos de la estructura de datos, y los detalles procedimentales que describen las operaciones.

La descripción del protocolo no es otra cosa que un conjunto de mensajes con sus correspondientes comentarios asociados. Por ejemplo, una parte de la descripción del protocolo del objeto sensor de movimiento (descrito anteriormente) podría ser:

MENSAJE (sensor de movimiento) -- leer: DEVUELVE ID del sensor, estado del sensor.
que describe el mensaje requerido para leer el sensor. De forma similar,

MENSAJE (sensor de movimiento) -- establecer: RECIBE ID del sensor, estado del sensor.
establece o inicializa el estado del sensor.

Para grandes sistemas son muchos mensajes, a menudo se pueden crear categorías de mensajes. Por ejemplo, para el sistema HogarSeguro las categorías de mensajes podrían ser: mensajes de configuración del sistema, mensajes de monitoreo, mensajes de sucesos, etc..

Una descripción de la implementación de un objeto proporciona los detalles internos ("ocultos") requeridos para su implementación, aunque no necesariamente para su invocación. Es decir, el diseñador del objeto proporciona una descripción de su implementación y, consecuentemente, crea los detalles internos del objeto. Sin embargo, otro diseñador o programador que utilice el objeto u otras instancias del objeto, sólo requiere la descripción del protocolo, no la descripción de la implementación.

Una descripción de la implementación está compuesta por la siguiente información: (1) una especificación del nombre del objeto y de la referencia a una clase; (2) una especificación de la estructura de datos privada, con indicación de los elementos de datos y sus tipos; (3) una descripción procedimental de cada operación o, alternativamente, referencias a esas descripciones

procedimentales. La descripción de la implementación debe contener suficiente información para que se puedan manejar de forma adecuada todos los mensajes descritos en la descripción del protocolo.

Cox caracteriza las diferencias entre la información contenida en la descripción del protocolo y la contenida en la descripción de la implementación, en términos de los "usuarios" y de los "suministradores" de los servicios. Un usuario de un "servicio" proporcionado por un objeto debe familiarizarse con el protocolo de invocación de ese servicio, es decir, con la forma de especificar qué es lo que desea. Al suministrador del servicio (el propio objeto), lo que le concierne es cómo proporcionar el servicio al usuario, es decir, los detalles de implementación. Este concepto, denominado encapsulación, se resume de la forma siguiente:

Un objeto proporciona encapsulación, en cuanto a que pone en servicio una estructura de datos y un grupo de procedimientos para accederla, de tal forma que los usuarios de esa facilidad puedan accederla a través de unas interfaces cuidadosamente documentadas, controladas y estandarizadas. Esas estructuras de datos encapsuladas, denominadas objetos, cuentan como datos activos a los que se les puede pedir que hagan cosas, mandándoles mensajes.

Métodos de Diseño Orientado a los Objetos

A veces es difícil distinguir claramente el análisis orientado a los objetos y el diseño orientado a los objetos. Esencialmente, el análisis orientado a los objetos (AOO) es una actividad de clasificación. Es decir, se analiza un problema con el fin de determinar clases de objetos que sean aplicables en el desarrollo de la solución. (El diseño orientado a los objetos que sean aplicables en el desarrollo de la solución). El diseño orientado a los objetos (DOO) permite al ingeniero de software indicar los objetos que se derivan de cada clase y las interrelaciones entre ellos. Además, el DOO debe proporcionar una notación que refleje las relaciones entre los objetos. En este capítulo también se pueden aplicar de igual forma la terminología, la notación y el enfoque usado en el AOO.

Los primeros intentos de describir un método de diseño orientado a los objetos no surgieron hasta principios de la década de los ochenta. Tanto Abbott como Booch establecen que el DOO debe comenzar con una descripción en lenguaje natural (p. ej.: español) de la estrategia de solución, mediante una realización en software, de un problema del mundo real. A partir de esa descripción, el diseñador puede identificar los objetos y las operaciones. Posteriores contribuciones de Schlaer y Mellor y de Coad y Yourdon introdujeron una notación más amplia para asistir a esa actividad y argumentaron que se trataba realmente de una actividad de análisis.
En su estado actual de evolución, los métodos de DOO combinan elementos de las tres categorías de diseño presentadas anteriormente: diseño de datos, diseño arquitectónico y diseño procedimental. Al identificar clases y objetos, se crean abstracciones de datos. Asociando operaciones a los datos, se especifican módulos y se establece una estructura para el software. Al desarrollar un mecanismo para la utilización de los objetos, (p. ej.: generación de mensajes) se describen las interfaces.

Los métodos de DOO iniciales están caracterizados por los siguientes pasos:

1. Definir el problema
2. Desarrollar una estrategia informal (narrativa de procesamiento) para la realización en software del campo del problema del mundo real.
3. Formalizar la estrategia mediante las siguientes subpasos:
   a) Identificar los objetos y sus atributos.
   b) Identificar las operaciones que se les puede aplicar a los objetos.
   c) Establecer interfaces que muestren las relaciones entre los objetos y las operaciones.
   d) Decidir aspectos del diseño detallado que proporcionen una descripción de la implementación de los objetos.
4. Volver a aplicar los pasos 2,3 y 4 recursivamente.

Se puede ver que los cuatro pasos (a excepción, posiblemente, del paso 3) se realizan durante el análisis de requisitos del software. Extendiendo esas actividades al diseño, se añaden los siguientes pasos:

5. Refinar el trabajo realizado en el AOO, buscando las subclases, las características de los mensajes y otros detalles de elaboración.
6. Representar la (s) estructura (s) de datos asociada (s) con los atributos del objeto.
7. Representar los detalles procedimentales asociados con cada operación.

En la siguiente sección, repasaremos los primeros cuatro pasos y consideraremos las actividades de diseño asociadas con los pasos del 5 al 7.

Definición de Clases y Objetos

"Definición" sólo es otro término para el análisis de requisitos. Para definir adecuadamente el sistema, se debe llevar a cabo la ingeniería del sistema de computadora para asignar la función al elemento de software del sistema. A continuación, se deben aplicar métodos de análisis de requisitos del software para identificar el ámbito de información y dividir el problema. Obviamente, los métodos de análisis que mejor se ajustan a un sistema que va a ser implementado con un enfoque orientado a los objetos, son los métodos de AOO.

La aplicación de los principios y métodos de análisis de requisitos permite al analista y al diseñador llevar a cabo dos subpasos necesarios: (1) describir el problema y (2) analizar y clarificar las limitaciones conocidas. La realización en software del problema del mundo real, independientemente de su tamaño o de su complejidad, debe describirse con una frase sencilla y gramaticalmente correcta. Obviamente, el nivel de abstracción puede ser muy alto, pero la frase sencilla que describe el problema "permite a los ingenieros de software que trabajan en el proyecto comprender el problema de forma sencilla y uniforme ".

Para ilustrar el paso de definición del problema (y todos los pasos siguientes), recordemos la narrativa de procesamiento de HogarSeguro ya presentada:

> El software HogarSeguro permite al propietario de la vivienda configurar el sistema de seguridad al instalarlo, controla todos los sensores conectados al sistema de seguridad e interactúa con el propietario a través de un teclado numérico y unas teclas de función que se encuentran en el panel de control de HogarSeguro.

Durante la instalación, se usa el panel de control de HogarSeguro para "programar" y configurar el sistema. Cada sensor tiene asignado un número y un tipo; existe una contraseña maestra para activar y desactivar el sistema, y se introduce(n) un(os) teléfono(s) con los que contactar cuando se produce un suceso detectado por un sensor.
Cuando el software detecta la sensorización de un suceso, hace que suene una alarma audible que está incorporada en el sistema. Tras un retardo, especificado por el propietario durante la configuración del sistema, el programa marca un número de teléfono de un servicio de monitoreo, proporciona información sobre la situación e informa sobre la naturaleza del suceso detectado. Cada 20 segundos se volverá a marcar el número de teléfono hasta que se consiga establecer la comunicación.
Toda la interacción con HogarSeguro esta gestionada por un subsistema de interacción con el usuario que lee la información introducida a través del teclado numérico y las teclas de función; muestra mensajes de petición e información sobre el estado del sistema en el monitor LCD. La interacción por teclado toma la siguiente forma:

Tras realizar el análisis gramatical de nombres y verbos, y eliminar los objetos candidatos que no satisfacen los criterios anteriores indicados, quedan seleccionados los objetos que se muestran en la Fig. 2.35.

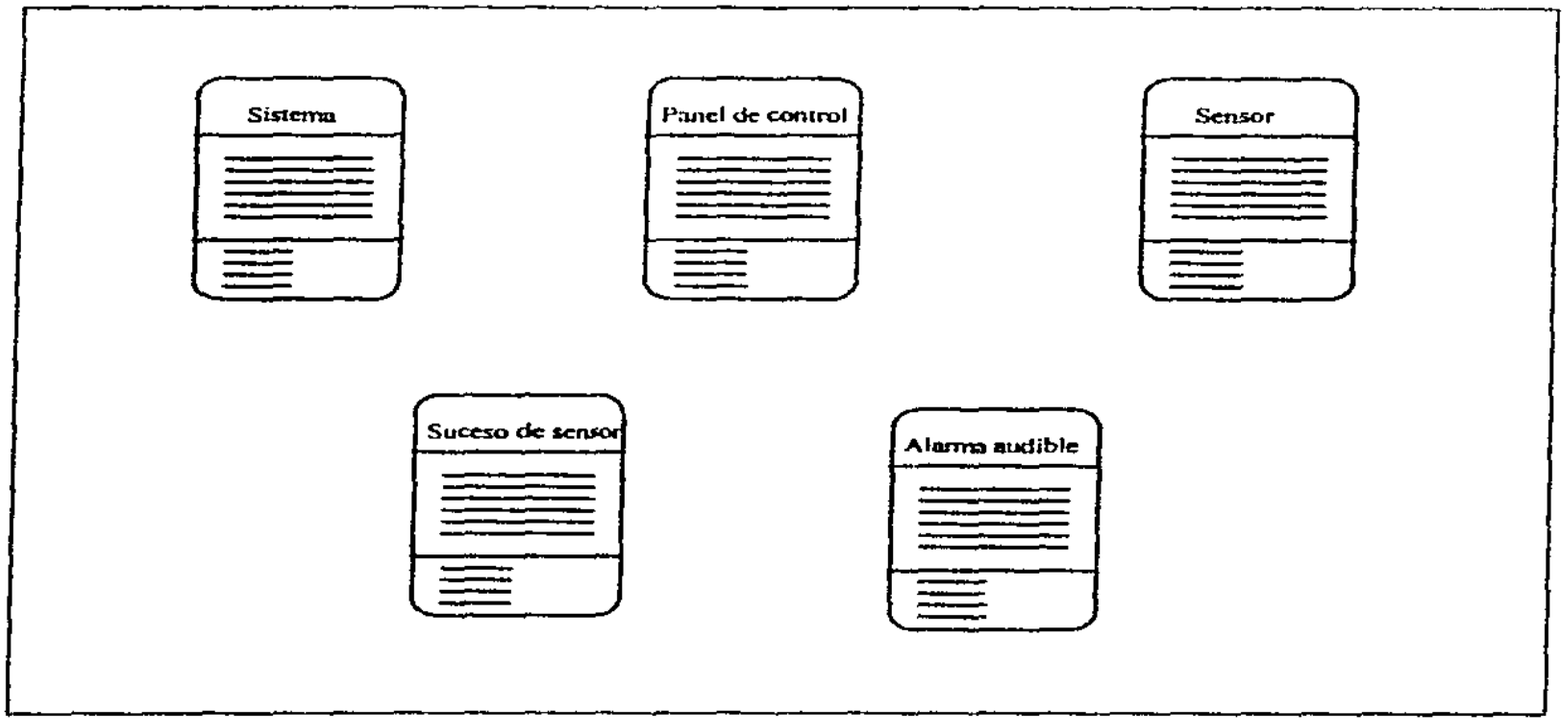


Fig. 2.35 Objetos seleccionados en el primer nivel de refinamiento.

Continuando con las actividades de análisis orientado a los objetos, se especifican los atributos y las operaciones de cada objeto (p. ej.: atributos y operaciones del objeto sistema que se muestra en la Fig. 2.33). Además, se siguen refinando los objetos definidos en el primer nivel de refinamiento, mediante la notación de estructura de ensamblaje ya descrita y repetida en la Fig. 2.36

En el corazón mismo del AOO reside la identificación de los objetos, sirviendo como base para el DOO. Abbott proporciona una valiosa argumentación:

La orientación de los objetos enfatiza la importancia de identificar de forma precisa los objetos y sus propiedades que serán manipuladas por un programa, antes de comenzar a escribir los detalles de esas manipulaciones. Sin esa cuidadosa identificación, será casi imposible precisar las operaciones a realizar y sus efectos deseados. Los nombres y las frases nominales de la estrategia informal son buenos indicadores de los objetos y sus clasificaciones (p. ej.: tipos de datos) de la solución del problema.

Así, el cometido del AOO es el de aislar todos los nombres y frases nominales contenidos en la narrativa de procesamiento que describe lo que ha de hacer el sistema.

En este momento, merece la pena fijarse en las categorías de nombre. A menudo, un nombre común es el nombre de una clase. Por ejemplo, sensor es un nombre común que describe una clase de dispositivos que se usan para detectar algún suceso. Los nombres propios son nombres de entes o cosas específicas. Por ejemplo, Ferrari F-40, Ford Taurus y Audi 100 son nombres propios o frases nominales de una clase que podríamos denominar automóvil. Un nombre abstracto o masivo es el nombre de una cantidad, de una actividad o de una medida. Por ejemplo, deportivo es un nombre masivo que se refiere a una colección de nombres propios de una clase referenciada por el nombre común automóvil.

Se puede utilizar esa categorización para ayudar a definir clases, subclases y objetos. A menudo, un nombre común representa una clase de objetos (una abstracción de datos). Un nombre propio representará una instancia de una clase. Un nombre abstracto o masivo (incluyendo unidades de medida) servirá para indicar características restrictivas o agrupamientos específicos del problema para los objetos o las clases. Sin embargo, es importante tener en cuenta que se deben utilizar el contexto y la semántica para determinar las categorías de los nombres. Una palabra puede ser un nombre común en un contexto, un nombre propio en otro y, en algunos casos, un nombre abstracto o masivo en un tercer contexto.

Es importante considerar todos los nombres y frases nominales, ya que serán de interés para la realización en el software final de la solución. Algunos objetos, como ya indicamos, residirán fuera de los límites del espacio de solución software. Otros objetos, aunque relevantes para el problema, podrán resultar redundantes o extraños al refinar la solución.

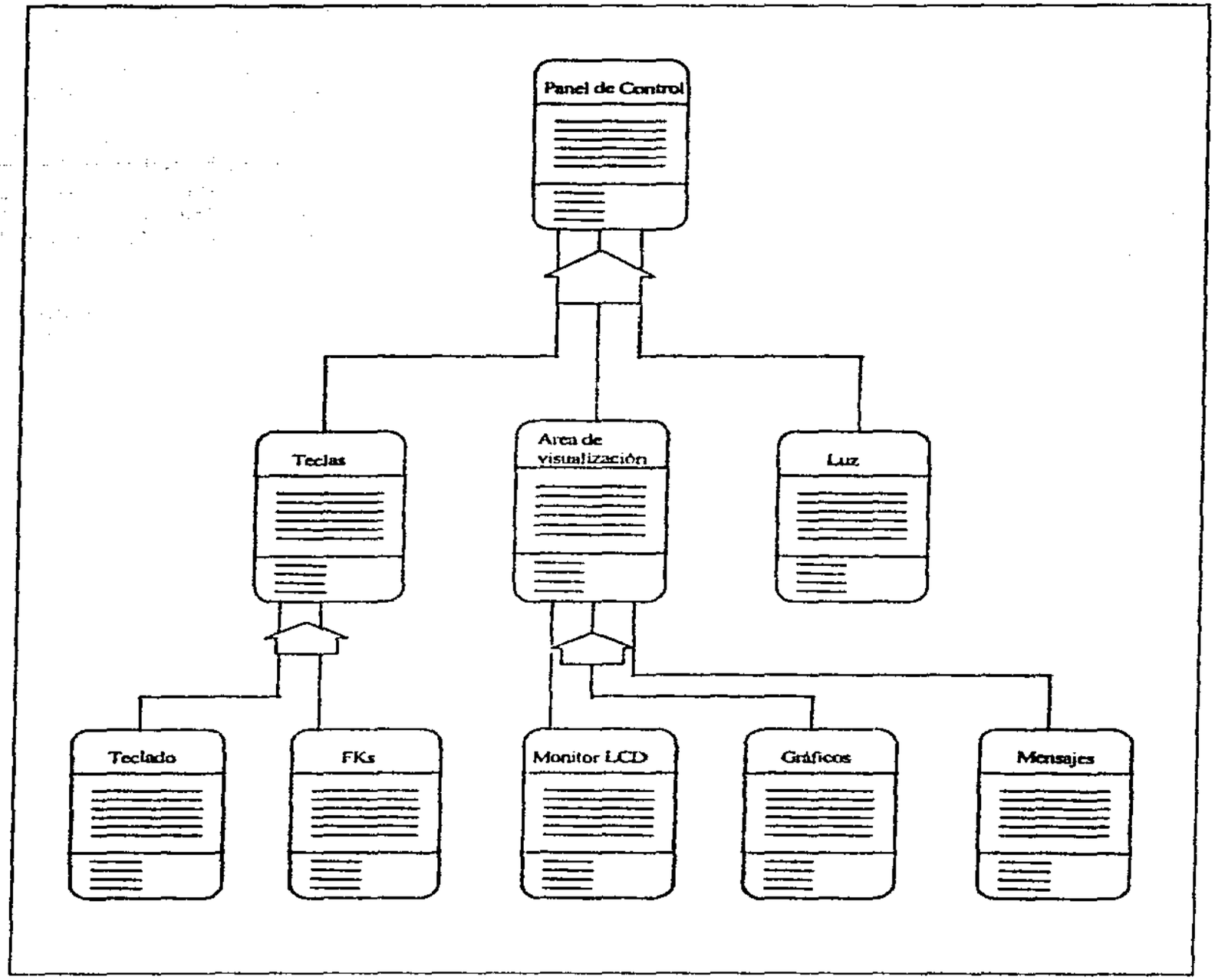


Fig. 2.36 Refinamiento de un objeto mediante una estructura de ensamblaje.

Refinamiento de las Operaciones

Una vez que se han identificado los objetos del espacio de la solución, el diseñador selecciona el conjunto de operaciones que actúan sobre los objetos. Las operaciones se identifican examinando todos los verbos de la estrategia informal (la narrativa de procesamiento).

Aunque existen muchos tipos distintos de operaciones, generalmente se pueden dividir en tres grandes categorías: (1) operaciones que manipulan los datos de alguna forma (p. ej.: adiciones, eliminaciones, cambios de formato, selecciones); (2) operaciones que realizan algún cálculo; (3) operaciones que monitorean un objeto frente a la ocurrencia de algún suceso de control.

Por ejemplo, la narrativa de procesamiento de HogarSeguro contiene los siguientes fragmentos de frase: "cada sensor tiene asignado un número y un tipo" y "se programa una contraseña maestra para activar y desactivar el sistema".
Estas dos frases indican varias cosas:

- Que para el objeto sensor es relevante una operación de asignación.
- Que se aplicará una operación programar al objeto sistema.
- Que activar y desactivar son operaciones que se aplican al sistema; también que el estado del sistema debe ser definido (usando la notación de diccionario de datos) como

estado del sistema = activado / desactivado

La operación programar quedó asignada durante el AOO, pero es durante el DOO cuando se refina en varias operaciones más específicas que son requeridas para configurar el sistema. Por ejemplo, tras dialogar con el ingeniero del producto, con el analista y, probablemente, con el departamento de ventas, el diseñador puede reelaborar la narrativa de procesamiento original y escribir lo siguiente para programar:

> Programar permite al usuario de HogarSeguro configurar el sistema una vez que ha sido instalado. El usuario puede (1) instalar números de teléfono; (2) definir retardos para las alarmas; (3) construir una tabla de sensores que contenga el ID, el tipo y la ubicación de cada sensor; y (4) cargar una contraseña maestra.

Por tanto, el diseñador ha refinado la operación simple programar y la ha sustituido por las operaciones instalar, definir, construir y cargar. Cada una de las nuevas operaciones entra a formar parte del objeto sistema, conoce las estructuras de datos internas que implementan los atributos del objeto y se invoca mandando al objeto mensajes de la forma :

MENSAJES (sistema) -- instalar : RECIBE número de teléfono;

que implica que, para proporcionar al sistema un número de teléfono de emergencia, se le debe enviar al objeto sistema el mensaje instalar.

Los verbos connotan acciones u ocurrencias. En el contexto de la formalización del DOO, no sólo consideramos los verbos, sino también las frases verbales descriptivas y los predicados (p.ej.: "es igual que") como posibles operaciones. Hasta que se haya refinado cada operación al mayor nivel de detalle se aplica recursivamente el análisis gramatical.

Componentes de Programa e Interfaces

Un aspecto importante de la calidad del diseño de software es la modularidad - esto es, la especificación de componentes de programa (módulos) que, combinados, forman el programa completo. El enfoque orientado a los objetos define el objeto como componente de programa que se encuentra enlazado con otros componentes (p. ej.: datos privados, operaciones). Pero la definición de objetos y operaciones no es suficiente. Durante el diseño, también debemos identificar las interfaces que existen entre los objetos y la estructura general (en sentido arquitectónico) de objetos.

Aunque un componente de programa es una abstracción de diseño, se puede representar dentro del contexto del lenguaje de programación que se va a utilizar para implementar el diseño. Para dar cabida al DOO, el lenguaje de programación que se use en la implementación debe permitir crear el siguiente componente de programa (modelado a partir de Ada):

```
PACKAGE    nombre-de-componente-de-programa IS
TYPE       especificación de los objetos de datos
      -
      -
      -
PROC       especificación de las operaciones asociadas ...
PRIVATE
PROC       operación.1 (descripción de la interfaz) IS
      -
      -
      -
END
PROC       operación (descripción de la interfaz) IS
      -
      -
      -
END
END        nombre-de-componente-de-programa
```

De acuerdo con el anterior LDP (lenguaje de diseño de programas) similar a Ada, un componente de programa queda especificado indicando tanto los objetos de datos como las operaciones. La parte de especificación del componente indica todos los objetos de datos (declarados con la sentencia TYPE) y todas las operaciones (PROC por "procedimiento") que actúan sobre ellos. La parte privada (PRIVATE) el componente proporciona los detalles ocultos de la estructura de datos y del procesamiento. En el contexto de la discusión anterior, el paquete (PACKAGE) es conceptualmente similar a los objetos tratados en este capítulo.

El primer componente de programa a identificar debe ser el módulo de más alto nivel desde el que se origina todo el procesamiento y del que evolucionan todas las estructuras de datos. Refiriéndonos de nuevo al ejemplo HogarSeguro considerado en las secciones anteriores, podemos definir el componente de programa de mayor nivel como:

PROCEDURE software HogarSeguro

Con la Fig. 2.13 como guía, podemos emparejar el componente software HogarSeguro con un diseño preliminar de los siguientes paquetes (objetos):

```
PACKAGE    sistema IS
        TYPE datos sistema
        PROC instalar, definir, construir, cargar
        PROC mostrar, inicializar, preguntar, modificar, llamar
        PRIVATE
                PACKAGE BODY sistema IS
                PRIVATE
                        sistema. id IS STRING LENGTH (8);
                        número de teléfono de verificación, número de teléfono, .....
                        IS STRING LENGTH (8);
                        tabla de sensores DEFINED
                                tipo de sensor ISSTRING LENGTH (2),
                                número de sensor, umbral de alarma IS NUMERIC;
                PROC instalar RECEIVES (número de teléfono)
                        (detalle de diseño para la operación instalar)
        -
        -
        -
END sistema
PACKAGE sensor IS
        TYPE datos de sensor
        PROC leer, establecer, probar
        PRIVATE
        PACKAGE BODY sensor IS
        PRIVATE
                sensor.id IS STRING LENGTH (8);
                estado de sensor IS STRING LENGTH (8);
                características de la alarma DEFINED
                        umbral, tipo de señal, nivel de señal IS NUMERIC;
                interfaz hardware DEFINED
                        tipo, características a/d, datos de temporización IS  NUMERIC;

END sensor
        -
        -
        -
END software HogarSeguro
```

Para cada uno de los componentes de programa del software HogarSeguro se especifican los objetos de datos y las correspondientes operaciones. Muchos de los detalles sobre la parte privada de los objetos de datos todavía no se han especificado, así como tampoco los detalles de las operaciones. Más adelante se considerarán esos detalles de implementación.

Una vez identificados los componentes de programa, estamos listos para examinar la evolución del diseño y evaluar la necesidad crítica de cambio. Es probable que la revisión de la definición "de primera mano" de los paquetes produzca modificaciones para añadir nuevos objetos o para volver a pasos anteriores del DOO (es decir, a la estrategia informal), con el fin de valorar la completitud de las operaciones que han sido especificadas.

Una Notación para el DOO

Anteriormente usamos una notación gráfica para representar los objetos, las operaciones, los mensajes y otras estructuras propuestas por Coad y Yourdon. Esta notación también va bien para las primeras etapas del diseño. Sin embargo, también se han propuesto otras notaciones que a menudo se encuentran en el campo industrial. En esta sección, repasaremos algunas de ellas.
Boock propone una notación que combina cuatro diagramas distintos para la creación del diseño orientado a los objetos. Un diagrama de clase refleja las clases y sus relaciones. Un diagrama de objetos representa objetos específicos (instancia de una clase) y los mensajes que pasan por ellos. Como parte del diseño físico, se asignan las clases y los objetos a componentes de software específicos. El diagrama de módulos, a veces denominados diagrama de Booch, sirve para ilustrar esos componentes de programa. Dado que muchos sistemas orientados a los objetos contienen varios programas que se puede ejecutar en varios procesadores distribuidos. Booch también sugiere una notación, denominada diagrama de procesos, que permite al diseñador reflejar cómo se asignan los procesos a procesadores concretos en un gran sistema.

Representación de relaciones entre clases y entre objetos.

Como alternativa a la notación ya sugerida, Booch sugiere diagramas que representan clases y objetos, así como las relaciones entre ellos. El diagrama de clases, ilustrado en la Fig. 2.37, indica la "arquitectura" de clases de un sistema. La herencia se representa con un símbolo de flecha y las conexiones de doble trazo indican que una clase utiliza la información contenida en otra clase. Los símbolos de los extremos de la notación usada (doble trazo) indican la cardinalidad. Por ejemplo, en la figura, para cada instancia de la clase A habrá una o más instancias (indicadas por el símbolo +) de la clase D. Además del propio diagrama de clases, se puede definir cada clase con una plantilla que incluya su nombre, sus atributos y sus operaciones, así como la información sobre la herencia, los parámetros y el comportamiento. Hay que tener en cuenta que la plantilla de la clase no es realmente una parte del diagrama, sino que se crea para complementarlo.

Las clases se definen como parte del diseño de un sistema y existen independientemente del comportamiento en ejecución del sistema. Sin embargo, los objetos se crean y se eliminan dinamicamente a medida que se ejecuta el sistema. Por esta razón, para el diseñador puede ser de un valor incalculable el disponer de una notación que capture la "semántica dinámica" o "imagen temporal" de un programa. El diagrama de objetos (Fig. 2.38) refleja cada instancia de una clase,

las operaciones que se le aplican y los mensajes que se le pasan. Las líneas que conectan los objetos terminan con un recuadro etiquetado que indica el tipo de datos que contiene el mensaje. La fecha paralela a la línea indica el tipo de sincronización entre los objetos (p. ej.: síncrono, asíncrono, retardado). Para cada objeto se crea una plantilla de objeto que define el objeto y los mensajes que se le pasan.

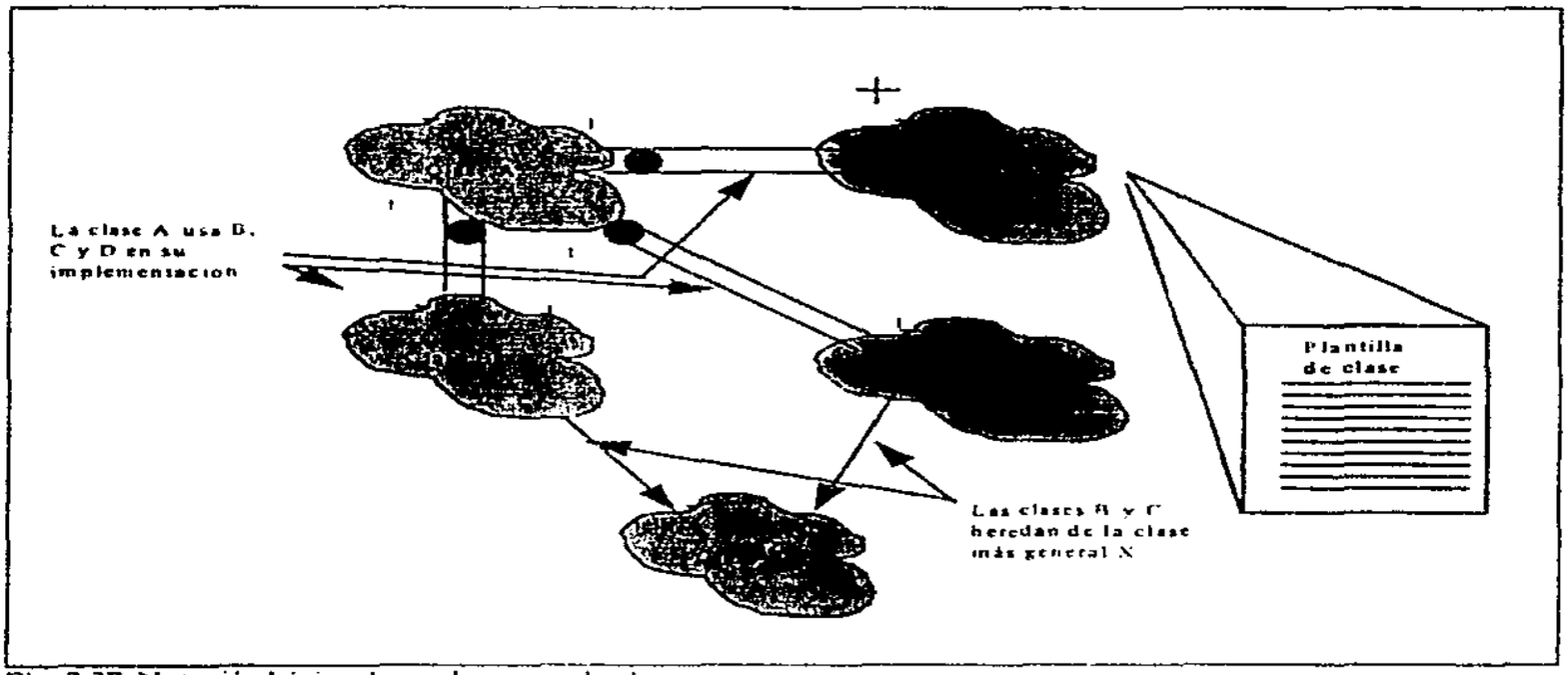


Fig. 2.37. Notación básica de un diagrama de clases.

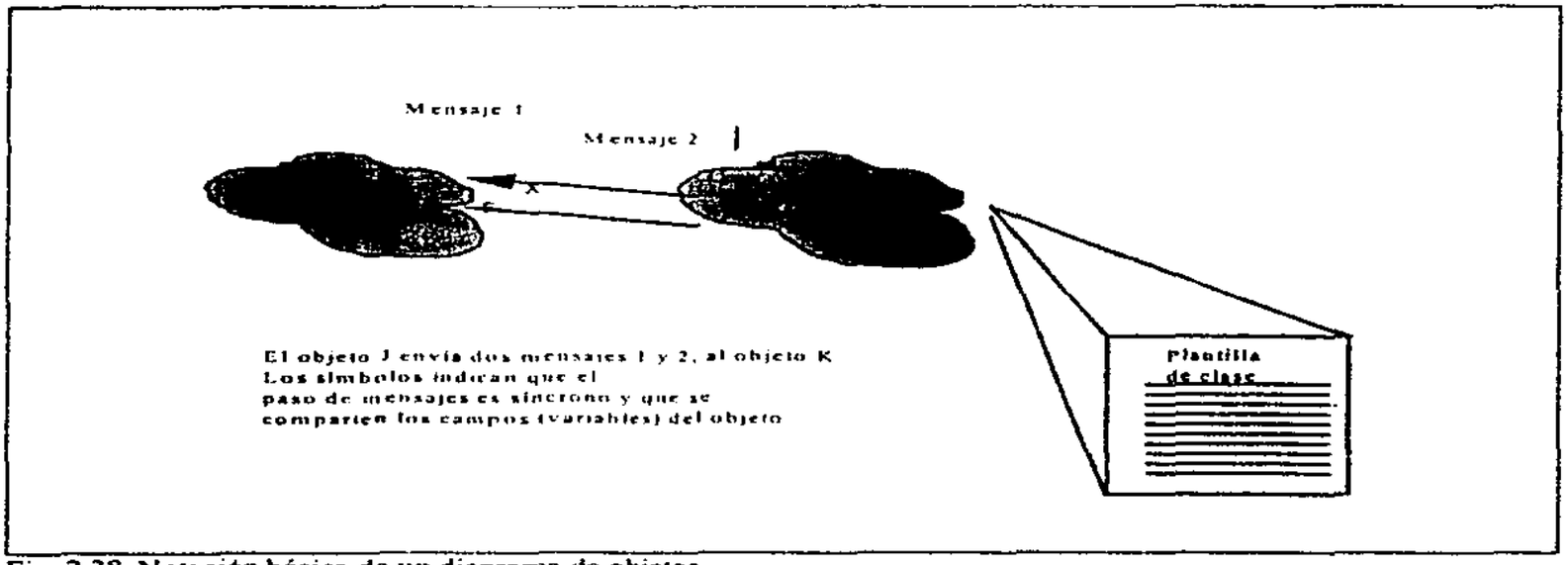


Fig. 2.38. Notación básica de un diagrama de objetos

Modularización del diseño

¿Cómo se define el concepto de módulo cuando se aplica el DOO? La respuesta a esta pregunta nos lleva a una notación de diseño que asigna clases y objetos a los componentes físicos de programa (módulos) y, por tanto, representa una visión de implementación de la arquitectura del programa. El diagrama de módulos representa un componente de programa (objeto) como una caja dividida en una parte de especificación (visible al exterior) y una parte privada (también denominada parte del cuerpo) que está oculta al exterior. En esta etapa, todavía no se especifican los detalles de implementación de la parte privada o cuerpo del paquete. Los objetos de datos se representan con un óvalo alargado, mientras que las operaciones que actúan sobre ellos se representan mediante rectángulos.

La Fig. 2.39 ilustra el uso de los diagramas de módulos para representar las dependencias entre los componentes de programa.

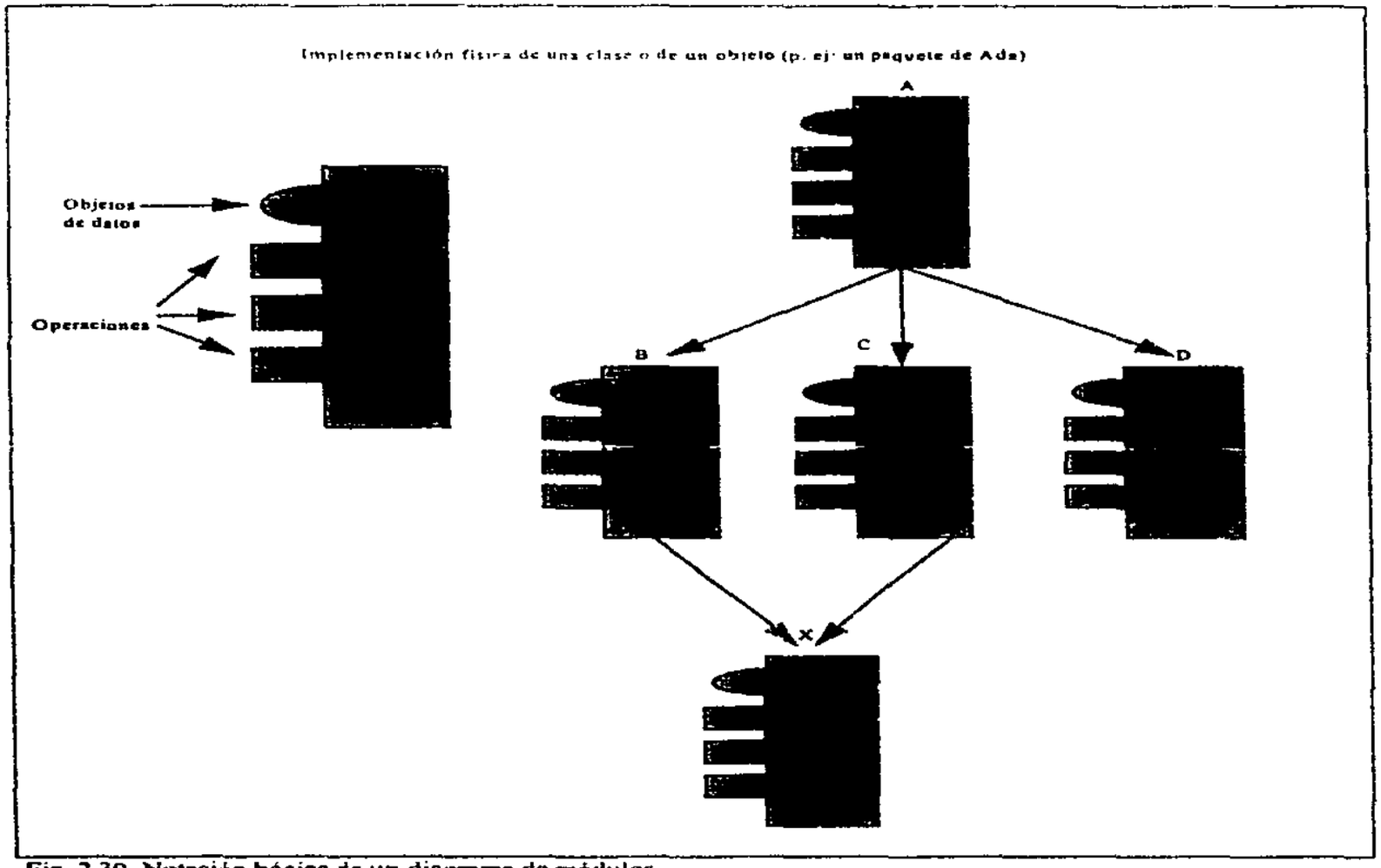


Fig. 2.39. Notación básica de un diagrama de módulos.

Las flechas de conexión implican dependencia; es decir, el paquete o componente en el origen de la flecha depende del paquete o componente de programa en la punta de la flecha. En la Fig. 2.39 el componente de programa de mayor nivel A, depende de los objetos y las operaciones contenidos en B, C y D, necesarios para satisfacer sus función.

El software de un sistema basado en computadora es ejecutado por uno o más procesadores (computadoras) que interactúan con otros dispositivos mediante una serie de conexiones definidas. El uso del diseño y de la implementación orientados a los objetos no altera esa situación. Por ello, se usa el diagrama de proceso (Fig. 2.40) para indicar los procesadores (caja sombreadas), los diapositivos (cajas en blanco) y las conexiones (líneas) que definen la arquitectura del hardware de un sistema. En el diagrama también se indican los procesos software que se ejecutan en cada procesador.

El uso de notaciones gráficas en el DOO no es esencial, pero proporciona una indicación de las relaciones entre clases y objetos, que falta en la representación en LDP. Se puede usar la notación propuesta por Booch para representar componentes de programa a un nivel de abstracción relativamente alto. Cuando comienza el diseño detallado de implementación, se abandona la notación gráfica y se usa LDP como representación de diseño.

Diseño Detallado para la Implementación

El paso de diseño detallado en el DOO es similar en muchos aspectos al diseño detallado en las otras metodologías de diseño. Se describen las interfaces con detalle; se refinan y se especifican las estructuras de datos; se diseñan los algoritmos para cada unidad del programa usando los conceptos fundamentales de diseño, tales como el refinamiento sucesivo y la programación estructurada. La diferencia clave del DOO es que se puede aplicar recursivamente en cualquier momento al proceso descrito en las secciones anteriores. De hecho, la definición recursiva de la estrategia de solución es esencial para conseguir un nivel de diseño y de abstracción de datos del que se pueda derivar el detalle de implementación. Se ha propuesto la siguiente directriz, para determinar cuándo es necesario la aplicación recursiva del DOO: "Si la implementación de una operación va a requerir una gran cantidad de código (más de 200 líneas.), entonces se toma la descripción del funcionamiento de la operación como declaración de un nuevo problema y se repite el proceso de DOO para ese nuevo problema".

Se puede utilizar la plantilla de diseño en LDP de un paquete como punto de partida del diseño detallado. Recordando la estructura general del paquete:

```
PACKAGE nombre-de-componente-de-programa IS
TYPE especificación de los objetos de datos
        -
        -
        -
PROC especificación de las operaciones asociadas
        -
        -
        -
        -
```

PRIVATE
    detalles de la estructura de datos del objeto
PACKAGE BODY nombre-de-componente-de-programa IS
PROC operación.i (descripción de la interfaz) IS

    -
    -
    -
    -

END
END nombre-de-componente-de-programa

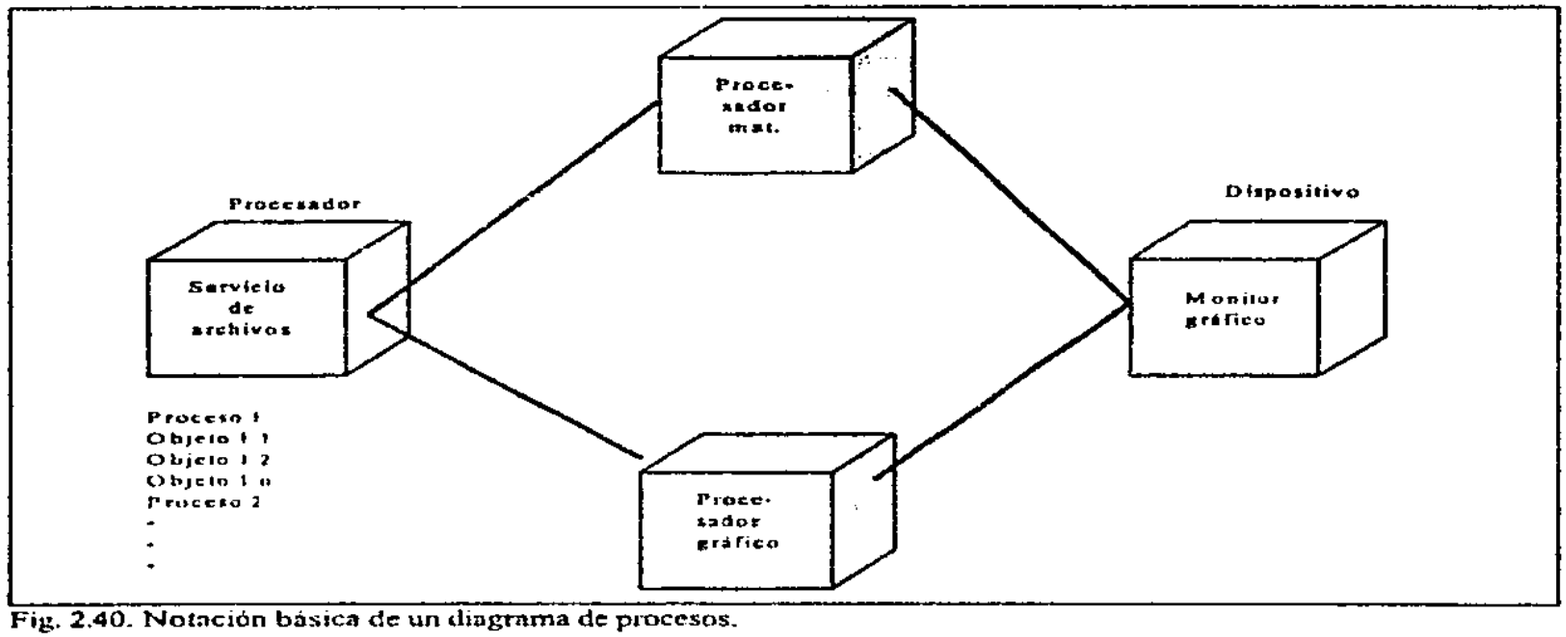


Fig. 2.40. Notación básica de un diagrama de procesos.

El diseño detallado completa toda la información requerida para implementar totalmente la estructura de datos y los tipos contenidos en la parte PRIVATE del paquete y todo el detalle procedimental contenido en el cuerpo del paquete (PACKAGE BODY).

Para ilustrar el diseño detallado de un componente de programa, reconsideremos el paquete sensor definido como parte del ejemplo HogarSeguro. En este punto del proceso de DOO, se puede representar sensor con el siguiente esquema de paquete:

PACKAGE sensor IS
TYPE datos de sensor
PROC leer, establecer, probar
PRIVATE
PACKAGE BODY sensor IS
PRIVATE

```
        sensor.id IS STRING LENGTH (8);
        estado de sensor IS STRING LEGTH (8);
        características de la alarma DEFINED
        umbral, tipo de señal, nivel de señal IS NUMERIC;
        interfaz hardware DEFINED
        tipo, características a/d, datos de temporización IS NUMERICA;

END sensor
```

Las estructuras de datos de los atributos del sensor ya han sido definidas. Por tanto, el primer paso es definir las interfaces de cada una de las operaciones asociadas con sensor:

```
PROC leer (sensor.id, estado del sensor: OUT);
PROC establecer (a características de la alarma, interfaz hardware: IN);
PROC probar (sensor.id, estado del sensor, características de la alarma: OUT);
```

El siguiente paso requiere un refinamiento sucesivo de cada operación asociada con el paquete sensor. Para ilustrar el refinamiento, desarrollemos una narrativa de procesamiento (una estrategia informal) para leer:

> Cuando el objeto sensor recibe un mensaje leer, se invoca el proceso leer. El proceso determina la interfaz y el tipo de señal, muestrea la interfaz del sensor, convierte las características A/D (analógica/digital) a un nivel de señal interno y compara el nivel de señal interno con un valor de umbral. Si se excede el umbral, se establece el estado del sensor o "suceso". De otro modo, se establece el estado del sensor a "in suceso". Si se detecta un error al muestrear el sensor, se establece el estado del sensor a "error".

Dada esa narrativa de procesamiento, se puede desarrollar una descripción en LDP para el proceso de lectura:

```
PROC leer (sensor.id, sensor.estado: OUT);
        señal.original IS BIT STRING
        IS (hardware.interfaz.tipo = "B")
THEN
        OBTENER (sensor, excepción: sensor.estado :=error)
                        señal.original;
        CONVERTIR señal.original A nivel.señal.interno;
        IF nivel.señal.interno> umbral
                THEN sensor.estado :="suceso";
                ELSE sensor.estado :="sin suceso";
        ENDIF
        ELSE (especificación del procesamiento para otros tipos de interfaces)
        ENDIF
        RETURN sensor.id, sensor.estado;
END leer
```

Se puede traducir la representación en LDP para la operación leer al lenguaje de programación apropiado. Se asume que las funciones OBTENER y CONVERTIR están disponibles como parte de la biblioteca del tiempo de ejecución.

Integración del DOO con el Análisis Estructurado y el Diseño Estructurado.

Dado el amplio uso del análisis estructurado (AE) y del diseño estructurado (DE) y el gran interés por el enfoque orientado a los objetos, surge una pregunta natural: ¿se pueden integrar estas dos estrategias de análisis y diseño para formar un método único? Actualmente todavía no hay una respuesta a esta pregunta: Algunos investigadores piensan que las estrategias son muy diferentes, mientras que otros creen que se pueden utilizar elementos de ambos métodos para desarrollar un modelo completo de un problema. En esta sección, investigaremos brevemente el espacio existente entre el AE/DE y el AOO/DOO, intentando indicar situaciones en las que pueden coexistir pacíficamente ambas técnicas.

Los elementos clave de AE/DE son el modelo de flujo, el diccionario de datos, la especificación de control, la especificación de procesos y el diagrama de entidad-relación (E-R) para el modelado de datos. A primera vista, podría parecer que el diagrama de flujo de datos podría proporcionar una buena conexión entre el AE/DE y AOO/DOO. Podría parecer que los elementos de datos - flujos (flechas) y almacenes (líneas dobles)- y los procesos ("burbujas") constituyen un modelo aproximado de los objetos y sus operaciones, respectivamente. Pero no es ese el caso.

Recordemos que la orientación a los objetos crea una representación del campo del problema del mundo real y la hace corresponder con el ámbito de la solución que es el software. Por esta razón, las representaciones de AE/DE que reflejan los elementos del mundo real relativo al problema proporcionarían un mejor puente con el AOO/DOO. Las entidades externas (cuadros) que representan a los productos y consumidores del flujo de datos serán posibles candidatos a objetos. Los objetos de datos definidos como parte del diagrama E-R también son candidatos a objetos. Es cierto que los DFDs de menor nivel podrían proporcionar una indicación de los atributos de algunos objetos y que lo procesos indicados en esos diagramas podrían ser operaciones aplicadas a los objetos. Pero un DFD no refleja un sistema desde el punto de vista orientado a los objetos.

La especificación de control (CSPEC) proporciona un modelo de comportamiento que puede ayudar al diseñador a comprender mejor el paso de mensajes en el sistema a los objetos. Finalmente, el diagrama de entidad-relación puede ayudar a aislar clases y subclases candidatas, así como las relaciones entre ellas.

Ward describe una correspondencia entre el AE/DE para sistemas de tiempo real y el diseño orientado a los objetos. Se puede representar una abstracción de datos (una clase) "como una transformación incorpórea con sus entradas y sus salidas; no es instanciada incorporándola al modelo de flujo ... tal transformación podría construirse teniendo en cuenta la reusabilidad, incluyendo datos y transformaciones adecuadas para muchos ámbitos de aplicación en los que aparezca la abstracción de datos": Esencialmente, lo que Ward sugiere es que se puede representar una clase o una subclase como una burbuja aparte. Los flujos hacia y desde esa burbuja siguen representando flujo de datos, pero implican operaciones asociadas a la clase. Para

instanciar un objeto, se incluye la burbuja independiente en el contexto del modelo de flujo. Por ejemplo, una clase sensor puede estar modelada como una abstracción de datos (utilizando un diagrama E-R) y representada por una burbuja aparte ("incorpórea"). Se podría instanciar un objeto de la clase sensor (p. ej.: sensor de humo) en un modelo de flujo específico, incluyendo una burbuja sensor de humo en el DFD.

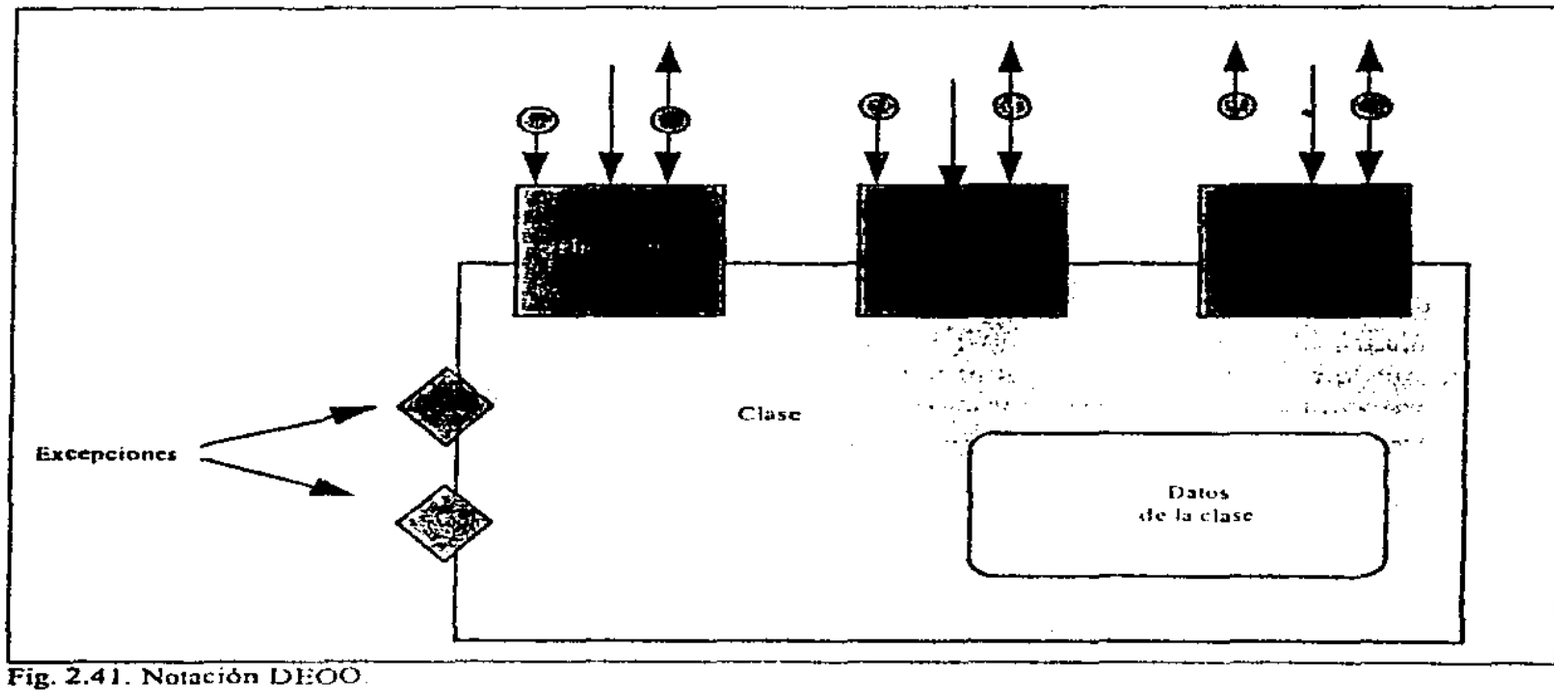


Fig. 2.41. Notación DEOO.

Wasserman ha sugerido una notación de diseño estructurado orientado a los objetos (DEOO), que engloba elementos del diagrama estructurado y el diagrama de Booch, así como los principios de definición de clases y de herencia. Los autores sugieren una notación que toma la forma que muestra la Fig. 2.41. El rectángulo grande se usa para definir una clase que usa las operaciones denotadas por los cuadros que se solapan con el rectángulo. Para mostrar los datos que entran y salen de las operaciones se usa una "notación de paso de parámetros" (los pequeños círculos con flechas que emanan de ellos), notación que ha sido a veces utilizada en la definición de diagrama de estructura. Las flechas gruesas indican el acoplamiento entre los módulos de la estructura del programa y los rombos indican condiciones de excepción que requieren un procesamiento de "casos especiales".

Algunos investigadores piensan que "no hay una oposición clara entre el AE/DE y el DOO", mientras que otros (los "puristas") creen que los métodos orientados a los objetos y el enfoque de diseño estructurado se excluyen mutuamente (y que el DOO es con mucho superior). Sólo el tiempo dirá si los dos valiosos métodos de diseño siguen caminos diferentes o si llegan a un matrimonio duradero.

3. PLANTEAMIENTO DE LA PROBLEMATICA Y PROPUESTA DE SOLUCION.

3.1 Antecedentes.

La tecnología de información comienza a ser una necesidad en las distintas áreas de las organizaciones. Contar con la información de manera oportuna y que ésta sea confiable es un objetivo general de las empresas, ya que la toma de decisiones con rapidez es un factor competitivo hoy en día, lo cual determina el crecimiento de las empresas.

Los sistemas informáticos bien diseñados representan la oportunidad de cambiar positivamente la forma de hacer las cosas en las empresas. Elimina vicios operativos, duplicidad de actividades, tareas repetitivas, pases laterales de información, resultados poco confiables, y por consecuencia, los altos riesgos operativos y administrativos.

El área de Bienes Adjudicados de la Banca ha experimentado un rápido crecimiento a raíz de la crisis económica que ha padecido nuestro país recientemente. Cuando un cliente de la banca no paga un crédito, ésta procede a adjudicarse legalmente el Bien. Una vez adjudicado su objetivo es venderlo, sin embargo mientras el Bien es vendido es necesario administrarlo.

Esta área de las instituciones financieras, en muchas de ellas, no cuenta con un sistema de información. Se apoyan en el uso de hojas de cálculo y procesadores de palabras, que no son suficientes para un control eficiente de las actividades operativas y administrativas del área. Es común que la información se encuentre duplicada en distintos archivos. Cuentan con un archivo diferente para cada uno de las distintas consultas y reportes que necesitan.

Las entradas y salidas de información, que se presentan cada vez que se adjudica un Bien o cada vez que se vende uno, se tienen que reflejar en cada uno de los distintos archivos mencionados. El control de movimientos prácticamente es manual, lo que resulta en poca confiabilidad de la información, lo cual afecta el proceso de toma de decisiones.

La cancelación de movimientos (adjudicación o venta) se tiene que reflejar también en cada uno de los distintos archivos donde se encuentra duplicada la información, de esta forma es fácil incurrir en errores por omisión o por la necesidad de capturar ciertos datos más de una vez.

Validar la información, así como buscar y corregir errores se vuelve una tarea muy complicada y prolongada, a la que se le tiene que dedicar muchas horas de trabajo, las cuales serían muy productivas en la realización de otras actividades.

Otro problema operativo lo representan los cierres mensuales. Son un conjunto de tareas en las que se invierte mucho tiempo y provoca normalmente fuertes retrasos de información y por consecuencia las decisiones se toman fuera de tiempo. Esto afecta seriamente el desarrollo de las empresas y en nuestro caso el resultado del área.

Nuevos requerimientos, tales como reportes, consultas y procesos son muy difíciles de desarrollar y proporcionar en forma oportuna, por lo que el área se tiene que conformar con una herramienta deficiente y sin potencial, sin la flexibilidad suficiente para soportar futuras necesidades y el crecimiento.

Otro punto muy importante es cuando aumenta la cantidad de información. Esto provoca un incremento en la pérdida de control y se multiplican los problemas del área. Se va perdiendo la eficiencia y productividad del área, lo que conduce a un caos de la misma. Las personas responsables directamente de realizar las actividades, cometen con mayor facilidad errores, además de que les hace falta cada vez más tiempo para realizarlas.

3.2 Análisis de Requerimientos.

Los requerimientos de la Banca para optimizar el desarrollo de las funciones del área de Bienes Adjudicados, se pueden resumir en lo siguiente:

> Desarrollar un sistema informático que satisfaga las necesidades actuales y futuras del área en cuanto al crecimiento, que sea flexible para integrarle nuevos requerimientos, que la información pueda consultarse en tiempo real y que optimice la forma de hacer las distintas actividades. Utilizando tecnología de vanguardia para el desarrollo de aplicaciones Cliente-Servidor.

Estos requerimientos generales se pueden subdividir en operativos, administrativos y de estrategia, y de procesos y funciones específicas a automatizar. Se requiere contar con una herramienta que represente un cambio positivo a todos los integrantes del área.

De manera más detallada podemos mencionar los siguientes requerimientos operativos:

- Integrar la información en una sola Base de Datos segura y consistente.
- Eliminar la duplicidad de la información.
- Eliminar la repetición de tareas.
- Eliminación de pases laterales de información.
- Consulta de la información en tiempo real.
- Obtención de reportes y estadísticas en cualquier momento, sin la necesidad de procesos especiales.
- Control fácil y confiable de Entradas y Salidas.
- Eliminación de tareas y flujo de documentos.

Los siguientes requerimientos administrativos y de estrategia:

- Proporcionar información oportuna que facilite la toma de decisiones.
- Mejor servicio a los clientes (compradores de los Bienes Adjudicados).
- Ventajas competitivas.
- Disminución de costos operativos y administrativos del área.
- Optimizar el uso del tiempo de los integrantes del área, incrementando así la productividad y la eficiencia.
- Incremento en las ventas, como consecuencia de un mejor servicio a los clientes y una mejor imagen de la empresa.
- Fomentar una nueva cultura informática a los integrantes del área.

Y los siguientes requerimientos de procesos y funciones a automatizar:

- Control de Registro de Bienes:
  - Adjudicados.
  - Dación en pago.
- Actualización de atributos importantes de los Bienes:
  - Revaluación.
  - Estatuas jurídico.
  - Corredor asignado.
  - Avalúos.
  - Posesión.
  - Escrituras.
  - Etc.
- Control de ventas de los Bienes y Seguimiento:
  - Ventas Totales.
  - Ventas con Anticipo.
  - Ventas Parciales.
  - Liquidaciones.
  - Quebrantos.
- Cierre Mensual.
  - Concentrados históricos.
  - Comparativos mensuales.
  - Análisis de información.
- Control de gastos de los Bienes.
  - Registro de gastos.
  - Reportes.
- Obtención de reportes y estadísticas:
  - Bienes disponibles (listos y no listos para su venta).
  - Relación de Bienes por diferentes criterios.
  - Entradas y Salidas de Bienes.
  - Afectaciones contables
  - Comportamiento de ventas.

- Antigüedad de la Adjudicación de los Bienes.
- Por su ubicación.
- Gráficas.
- Otros.

3.3 Estrategia de Solución y Viabilidad.

Después de conocer los requerimientos de los futuros usuarios del sistema, se necesita de una estrategia para lograrlos. Se deben definir todas las actividades necesarias para desarrollar el sistema, capacitación para su uso eficiente, e instalación para dejarlo en funcionamiento:

Actividades estratégicas para el desarrollo de la solución:

La primera actividad importante es conocer la situación actual del cliente, en cuanto a tecnología necesaria para el funcionamiento del sistema requerido:

- Análisis a detalle de la situación actual de la institución, especialmente del área de Bienes Adjudicados, tanto en hardware, comunicaciones y software.
- Definición del equipo de cómputo adecuado para el óptimo funcionamiento del sistema, tomando en cuenta el número de usuarios, transacciones y otros parámetros.
- Definición de la arquitectura Cliente/Servidor más adecuada.
- Integración de nuevo equipo de cómputo y software en caso de que se determine conveniente o necesario.

La segunda actividad es llevar a cabo el Análisis y Diseño del Sistema, lo que nos permitirá conocer a detalle el contenido y alcances del Sistema y nos proporcionará la información necesaria para construirlo:

- Análisis del Modelo Actual de Procesos.
- Análisis del Nuevo Modelo de Procesos.
- Definición de Entidades.
- Definición de Procesos y Funciones.
- Diseño de la Base de Datos.
- Diagrama de Entidad-Relación.
- Diagrama de Flujo de Datos.
- Elaboración del Prototipo.
- Especificación de Aplicaciones.
- Aprobación de la información de esta etapa.

La tercera actividad consiste en programar las distintas aplicaciones que conformarán el Sistema. Terminada esta etapa el sistema funcionará de acuerdo a lo especificado en la etapa de Análisis y Diseño:

- Preparación del ambiente de desarrollo del Sistema.
- Programación de rutinas comunes.
- Programación de las aplicaciones y pruebas unitarias.
- Diseño de pruebas integrales (casos de prueba).
- Pruebas Integrales al Sistema.
- Aprobación del funcionamiento correcto de las aplicaciones.

La cuarta actividad es instalar el sistema, es decir, el sistema quedará listo para ser utilizado, quedando como una poderosa herramienta para el desarrollo de las actividades del área y el logro eficiente de los objetivos:

- Generación del ambiente de producción.
- Generación de la Base de datos.
- Instalación del ejecutable del sistema (de acuerdo a la arquitectura cliente-servidor seleccionada).
- Definición y captura de los parámetros iniciales del sistema.
- Captura de catálogos.
- Estabilización.

La siguiente actividad es capacitar al personal del área en el uso del sistema. Otra capacitación importante es la técnica y va orientada al área de sistemas de la institución y explica cómo está construido el sistema

- Operativa (a los usuarios del sistema).
- Técnica (área de sistemas de la institución).

Una actividad importante después de que un sistema está en producción, es proporcionarle mantenimiento para que siempre esté actualizado y no disminuya el nivel de beneficios que aporta, sino que al contrario, los incremente. Los distintos servicios de mantenimiento son los siguientes:

- Perfectivo.
- Correctivo.
- Adaptativo.
- Aumentativo.

Personal involucrado y tiempos de desarrollo:

Para integrar esta solución, se requiere de un equipo de profesionistas que desarrollen las distintas fases y actividades del proyecto. Es muy importante contemplar los distintos perfiles profesionales que deben intervenir en el desarrollo de un sistema informático:

Coordinador o Líder del Proyecto.- Es el responsable de la conclusión exitosa del proyecto. Asigna, coordina y revisa las actividades de todos los integrantes de equipo de trabajo. Detecta, corrige o controla desviaciones. Debe contar con la experiencia administrativa y técnica para apoyar a la solución de cualquier problema que se presente durante el desarrollo del proyecto.

Analistas Programadores.- Esta área se encarga del análisis, diseño y programación de las aplicaciones que conforman el sistema. Deben cumplir con estándares, tanto para técnicas de programación como de estructura funcional de las aplicaciones. También deben de hacer un primer nivel de pruebas unitarias a sus aplicaciones.

Documentación.- La función de esta área es desarrollar las ayudas en-línea para los usuarios del sistema, así como, otras herramientas de asistencia para el usuario. También elaboran el manual técnico y el de usuario.

Calidad.- Esta área es responsable de efectuar las pruebas a las aplicaciones del sistema, tanto unitarias como integrales, además de vigilar que se apliquen los distintos estándares definidos para el desarrollo de las aplicaciones. Deben definir casos de prueba y resultados esperados. Además de llevar un control de los errores presentados.

Las distintas áreas mencionadas se relacionan entre sí, una de ellas construye la aplicación, otra hace las pruebas necesarias para garantizar la calidad del producto, y otra realiza la documentación asociada al sistema (ayudas y manuales). El responsable de que todo esto suceda es el líder del proyecto.

Estas actividades ingresan a un círculo cuyo objetivo principal es la Calidad del producto que se está desarrollando y la conclusión oportuna del mismo. Omitir alguna de estas actividades conduce muchas veces, a un producto incompleto, inconsistente y que por lo tanto no satisface las expectativas y necesidades del cliente. Además de que provoca la pérdida de control del tiempo y retrasos permanentes que hacen interminables los proyectos.

Tomando en cuenta la importancia de integrar a nuestro equipo de desarrollo las áreas mencionadas y de acuerdo al resultado de un análisis preliminar del Sistema para el Control de Bienes Adjudicados, se estiman los siguientes recursos para su desarrollo:

- Un Líder de proyectos.
- Dos Analistas programadores.
- Un Técnico documentador.
- Un técnico de calidad.

Los tiempos de desarrollo para cada una de las distintas actividades del proyecto son los siguientes:

ACTIVIDAD	TIEMPO ESTIMADO (Días Hábiles)
Análisis y Diseño	20
Programación.	20
Pruebas y ajustes	10
Capacitación	2
Instalación y Estabilizacion.	3

El tiempo total del desarrollo del sistema será de: 55 días hábiles.

Costo del desarrollo del Sistema.

Para obtener el costo total del desarrollo del sistema, debemos especificar el costo de cada uno de los distintos perfiles profesionales involucrados. El costo mensual indicado a continuación considera un promedio de 22 días hábiles por mes y 8 hrs de trabajo por día, lo cual representa un total de 176 hrs.

PERFIL PROFESIONAL	COSTO MENSUAL
Líder de Proyectos	30,000.00
Analista Programador.	23,000.00
Técnico de Documentación	10,000.00
Técnico de Calidad.	10,000.00

El costo mensual del sistema será: 73,000.00

Considerando 2.5 meses para el desarrollo total del sistema, el costo total del proyecto será de:

Costo total del Proyecto: $ 182,500.00

Hardware requerido.

El área de Bienes Adjudicados de la Banca, cuenta en promedio con cuatro integrantes. Este es el caso de nuestro cliente, por lo que se requiere del siguiente equipo de cómputo, el cual debe funcionar en red, bajo una arquitectura cliente-servidor de dos capas:

1 "Servidor" de Base de Datos.
Procesador Pentium, 32 MB RAM.
4 Computadoras "Cliente".
Procesador Pentium, 8 MB RAM.
5 Tarjetas para Red 10 Base T.
1 Concentrador de 12 puertos.

La arquitectura cliente-servidor de dos capas consiste en el "servidor" de base de datos y el programa de usuario con interfase I/O instalado en cada "cliente".

Software de Red y Administrador de Base de Datos.

Los componentes de la solución presentados a continuación son requeridos por nuestro cliente ya que forman parte de sus estándares para el desarrollo e instalación de aplicaciones. De cualquier manera consideramos que es una buena decisión ya que coincide con las tendencias actuales del mercado.

Sistema operativo de Red:
Windows NT para cinco usuarios.

Administrador de Base de Datos:
SQL-Anywhere para cinco usuarios.

Windows 95 para cada cliente.

Herramientas para el desarrollo del Sistema.

El lenguaje de programación que se utilizará para el desarrollo del sistemas es Power Builder de Powersoft. Es un lenguaje de cuarta generación orientado a objetos programados por eventos. También es un estándar de nuestro cliente.

Participación del cliente.

La participación del cliente y de manera más específica la de los futuros usuarios del sistema es indispensable para lograr con éxito el desarrollo del proyecto.

Debe proporcionar al equipo de desarrollo la información necesaria y participar en funciones de seguimiento, asignación de recursos, revisión y aceptación de las funciones del proyecto. También debe permitir el acceso a sus instalaciones al personal encargado de desarrollar el proyecto y en caso de ser necesario asignarles un lugar de trabajo.

Es también compromiso del Cliente adquirir los paquetes de software, licencias y equipo de cómputo, necesarios para el desarrollo, instalación y operación del Sistema. Cabe señalar que estos conceptos no son considerados en nuestra propuesta al cliente ya que cuenta con un área departamental que satisface estas necesidades.

Viabilidad del Proyecto.

La incorporación de tecnología de información tiene como resultado grandes beneficios, ya que apoya directamente la productividad en la ejecución de las funciones administrativas y operativas de las distintas áreas de las empresas.

Dependiendo del nivel de cultura informática del personal de una empresa, se observa que entre más bajo sea éste, mayor oposición se encuentra al cambio. Y las razones pueden ser de los

más variadas: temor generalizado al cambio, a ser desplazados, a dejar de hacer ciertas actividades, a no ser indispensables.

De cualquier manera es bueno comentar que la cultura informática cada día se extiende más, por lo que la tecnología de información comienza a ser una herramienta indispensable para el crecimiento tanto de las personas como de las empresas.

La Banca en México representa un sector económico muy fuerte y con una cultura informática de alto nivel. El área de Bienes Adjudicados de nuestro cliente, apoya el desarrollo e instalación del sistema ya que están seguros del apoyo que esta nueva herramienta representará para el desempeño de las distintas actividades del área, incrementando así, la eficiencia, productividad e imagen del área.

Otro problema es la inversión económica que representa integrar tecnología de información a las empresas. Es muy común que no se comprenda la relación COSTO/BENEFICIO, y por otro lado, no saber distinguir entre un gasto y una inversión.

Cliente-Servidor implica más inversión, pero su relación Costo/Beneficio es mejor.

Resumiendo podemos indicar los siguientes Costos para el desarrollo e instalación de este sistema:

- Hardware y software del "cliente".
- Hardware y software del "servidor".
- Desarrollo de la aplicación.
- Administrador de Base de Datos.
- Comunicaciones.
- Capacitación.
- Participación del usuario final.
- Soporte al usuario final.

La automatización de los procesos fortalece los negocios. El retorno de inversión en cliente-servidor se da principalmente con el usuario final ya que los apoya de la siguiente manera incrementando su productividad:

- Automatización de procesos manuales (uso de GUIs, reportes a la medida, explotación de la información oportuna, etc.).
- Independencia operativa.

Y la relación Costo/Beneficio es mayor:

- Mayores ingresos.
- Menores costos operativos sobre el manejo de los "Bienes".

En el caso de la Banca, tener que adjudicar un "Bien" representa ya una pérdida, ya que es resultado de algún crédito con problemas (el cliente no puede o no quiere pagar). Administrar un

"Bien" representa un gasto fuerte, además de otros gastos asociados como predial, agua, vigilancia, renta de bodegas, mantenimiento, tenencia, etc.

La situación ideal sería jamás tener que adjudicar un "Bien", pero una vez adjudicados, lo más importante es venderlos. Este sistema proporcionará un control eficiente de los "Bienes" y se podrá obtener información oportuna y confiable de los mismos: vendidos, pendientes de vender, prometidos en venta, asignados a corredor, etc. Se podrá determinar utilidad o pérdida al momento de venderlos, distintas formas de análisis de información, etc., proporcionando así elementos valiosos para la toma oportuna y correcta de decisiones.

En la actualidad, como resultado de la última crisis económica en nuestro país, la adjudicación de "Bienes" se ha venido incrementando considerablemente representando esto un gran problema para la Banca. Esto ha provocado que al personal de esta área le sea cada vez más difícil administrar los Bienes que reciben, así como definir las mejores estrategias para su oportuna comercialización, ya que no cuentan con la información que necesitan para apoyar la toma de decisiones, ni con la automatización de los procesos que les permita ser más productivos.

Un banco mediano puede estar adjudicando "Bienes" tanto muebles como inmuebles que representan millones de pesos por mes, por lo que la importancia de administrarlos adecuadamente y venderlos con rapidez se vuelven actividades estratégicas. Los gastos pueden ascender a cientos de miles de pesos mensuales y muchas veces no se tiene idea del nivel de gastos que esto representa.

De acuerdo a un análisis previo y muy general, el retorno de inversión puede ser de uno a seis meses, lo que representa un tiempo muy razonable, haciendo de este sistema un proyecto de alta viabilidad, por lo que la Dirección de Bienes Adjudicados de nuestro cliente ha tomado la decisión del desarrollo del Proyecto.

3.4 Análisis Operacional.

Los Bienes Adjudicados se clasifican en dos grandes grupos:

- Muebles, al que corresponden mobiliario, equipo, automóviles, camiones, aviones, obras de arte, joyas, etc.
- Inmuebles, al que corresponden casas, edificios, departamentos, terrenos, bodegas, naves industriales, etc.

Los Bienes tienen asociada información muy importante para su óptima administración y comercialización:

- Región y sucursal bancaria.
- Deudor.
- Tipo del Bien (Mueble o Inmueble)
- Clasificación por tipo de Bien (Muebles: automóvil, mobiliario, equipo, etc. Inmuebles: Casas, terrenos, edificios, etc.)
- Descripción detallada.

- **Ubicación.**
- **Fecha de Adjudicación.**
- **Importe o valor de Adjudicación.**
- **Fecha de Registro.**
- Situación Jurídica.
- Listo para la venta.
- Corredor asignado.
- Estatus.
- Fecha e Importe de Venta

La adjudicación de Bienes está relacionada con Deudores que dejaron de pagar un crédito bancario. A estos Bienes se les asigna un Valor de Adjudicación autorizado por la CNByV, y es este valor el que amortiza el importe de la deuda del cliente. Para esta área de Bienes Adjudicados no es importante la situación jurídica del cliente deudor, tampoco el importe pendiente de pago de la deuda. Su objetivo es la administración y comercialización de los Bienes que recibe.

Debido a que estos Bienes son resultado de créditos que terminaron con problemas, el proceso de venta siempre es de contado. En el caso de Bienes Inmuebles es común el pago de un anticipo para formalizar la venta, y al momento de escriturar se debe pagar el remanente. En el caso de ventas parciales de Bienes Muebles se debe pagar de contado la parte que se está vendiendo. No existe crédito en la venta de Bienes Adjudicados Muebles e Inmuebles

La Comisión Nacional Bancaria y de Valores (CNByV) regula y vigila todas las operaciones de adjudicación y venta de los Bienes. A este organismo se le debe informar todas las entradas y venta de Bienes. En el caso de ventas se le debe informar el importe de la venta e indicarle la existencia de ganancia o pérdida.

Para que los Bienes Adjudicados se puedan vender es necesario que estén listos para la venta. Un Bien está listo para la venta cuando cumple con los siguientes requisitos:

- Se tiene posesión del mismo
- Se tiene escrituras (bienes inmuebles) o factura (bienes muebles)
- Sin algún tipo de trámite jurídico.

Contablemente los Bienes Adjudicados se clasifican en tres cuentas:

- ***1601.*** Para Bienes Muebles.
- ***1602.*** Para Bienes Inmuebles.
- ***1603.*** Para prometidos en venta.

Los Bienes Prometidos en Venta son el resultado de la formalización de la venta de un Bien Inmueble mediante el pago de un anticipo. Esta operación o movimiento genera un registro con cuenta 1603 para el remanente pendiente de pago, y deberá ser liquidado al momento de la escrituración concluyendo así el proceso de la venta.

Todo movimiento (registro, venta total, venta parcial, salida, revaluaciones, anticipo, liquidación, quebranto, cancelaciones, etc.) que se efectúe sobre un Bien debe ser registrado, además de actualizar su estatus, fecha e importe del movimiento. Esto permitirá contar con una bitácora para cada Bien. También se debe permitir la cancelación de movimientos en caso de error.

Las salidas se presentan cuando un Bien pasa a ser parte de los activos del Banco, cuando se pierden, son donados. En este caso no existe el proceso de venta pero la salida debe ser reportada y contabilizada. Esta información también debe ser reportada a la CNByV.

Las revaluaciones permiten ajustar el importe o valor de adjudicación de un Bien, incrementándole o decrementándolo. Estos movimientos deben ser autorizados por la CNByV. Cuando pasa mucho tiempo sin venderse un Bien puede ser conveniente esta operación para efectos de pérdida o ganancia al momento de la venta.

Como resultado de la venta de un Bien Mueble o Inmueble, puede existir ganancia o pérdida con respecto al valor de adjudicación del Bien. En caso de pérdida o quebranto es necesaria una ficha de castigo por el valor de la pérdida. La ficha de castigo o quebranto es un documento que aporta el banco respaldando el importe de la pérdida.

Adjudicar un Bien representa gastos para el banco. En el caso de Bienes Inmuebles se requiere pagar predial, vigilancia, mantenimiento, agua, luz, etc. En el caso de Bienes Muebles se paga tenencia, almacenaje, mantenimiento, etc. Saber cuánto se gasta por la administración de un Bien es importante ya que puede determinar prioridades en el esfuerzo de venta. En realidad es estratégico para el banco saber cuánto le cuesta administrar los Bienes Adjudicados agrupándolos por sucursal, región, o todo el banco, además de clasificar estos gastos por distintos conceptos.

El sistema debe contar con un amplio módulo de información en tiempo real, que nos permita realizar consultas versátiles sobre los Bienes, comparativos mensuales de información, reportes contables, información para la CNByV, etc.

A continuación se presenta un Diagrama de Procesos del Sistema, así como, Diagramas de Estado para Bienes Muebles y Bienes Inmuebles:

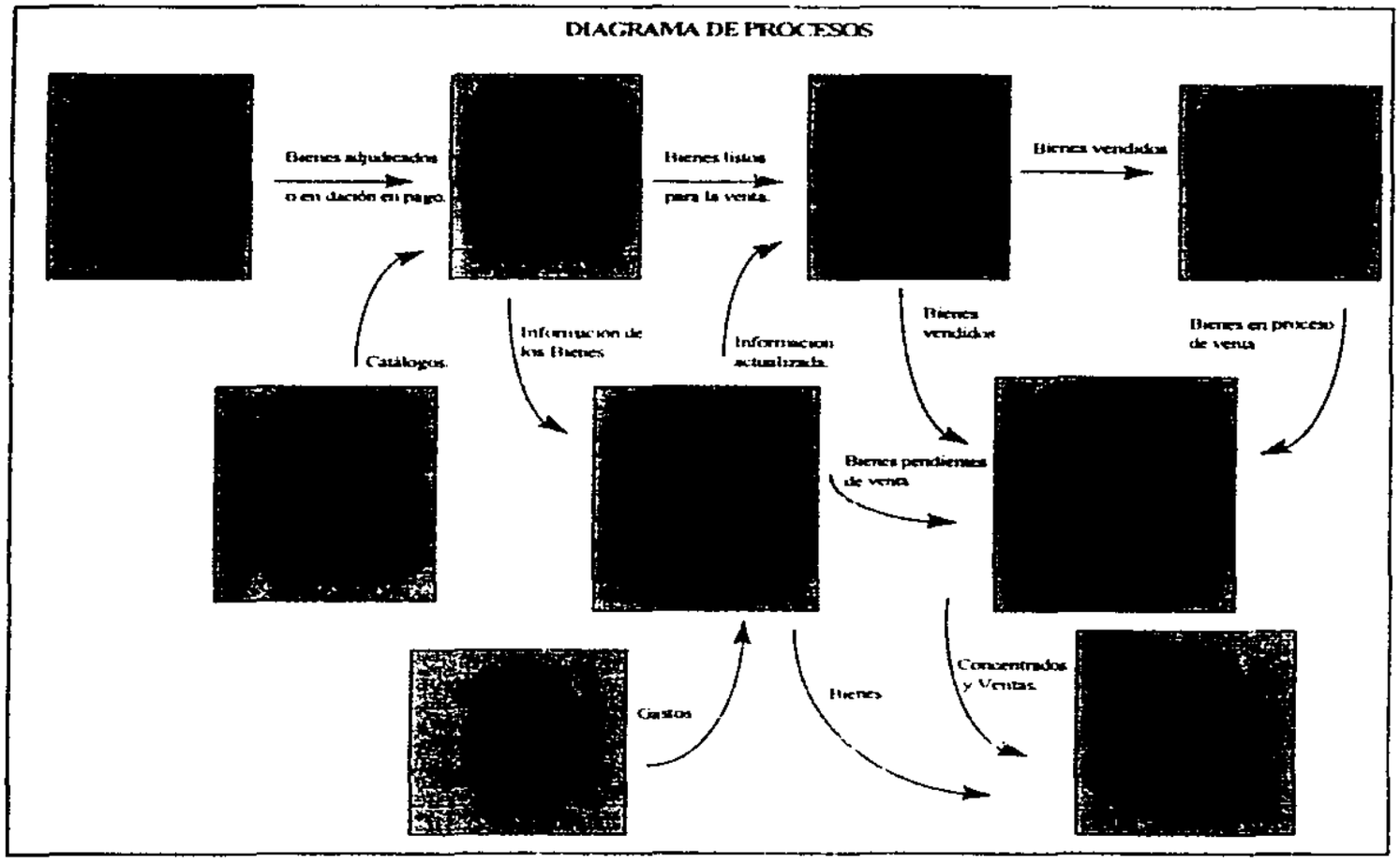


Fig. 3.2 Diagrama de Procesos del Sistema de Bienes Adjudicados.

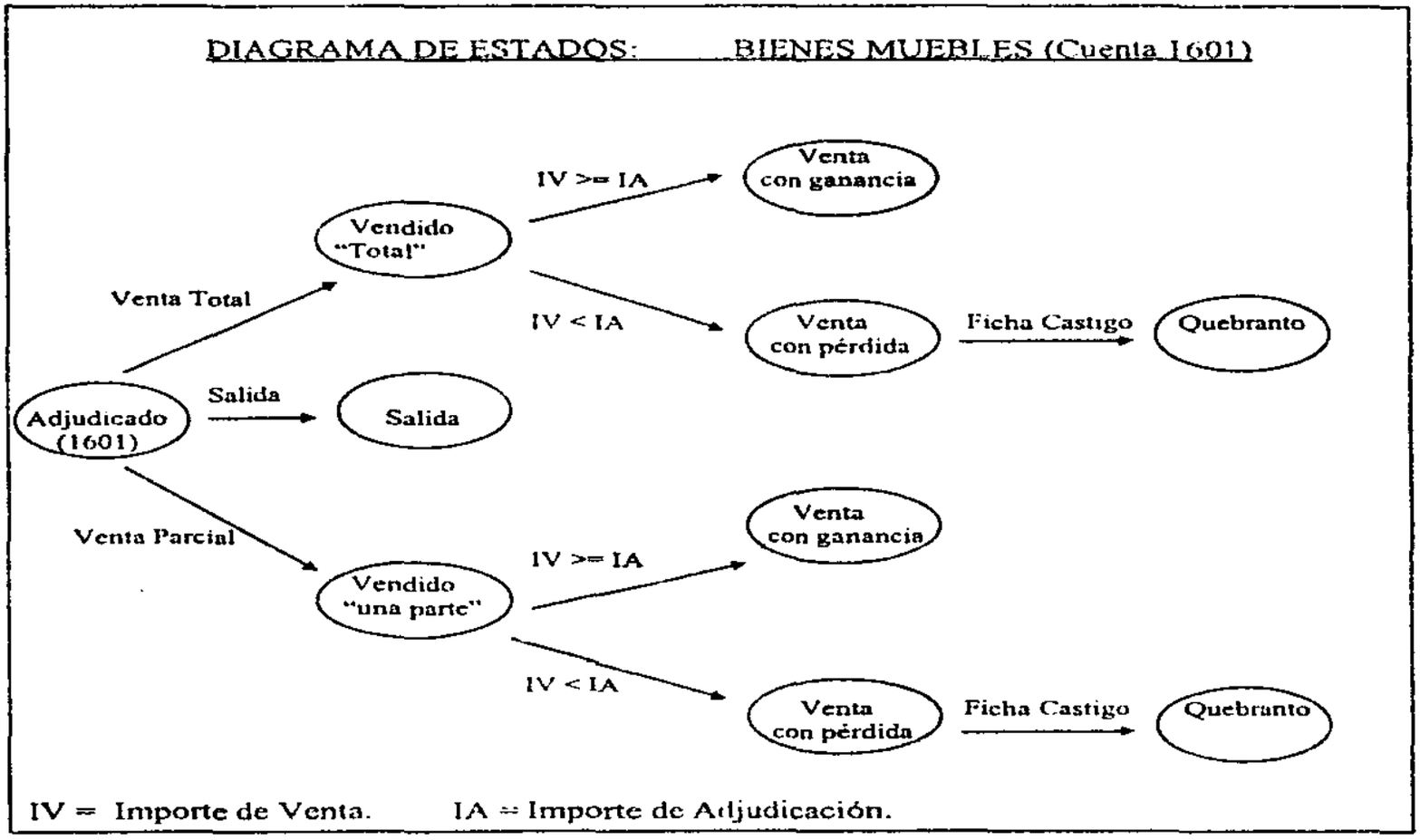


Fig. 3.2 Diagrama de Estados de Bienes Muebles.

Los Bienes Muebles (vehículos, camiones, muebles de hogar, de oficina, joyas, obras de arte, etc.) pueden ser vendidos totalmente o parcialmente. La venta parcial se presenta en el caso de lotes, por ejemplo, si se adjudica un conjunto de muebles o joyas se puede vender parte de estos a un comprador, de lo que quede se puede vender parte a otro comprador y así sucesivamente. Tanto en las ventas totales como en las parciales el pago debe hacerse en una sola exhibición.

La salida se presenta cuando un Bien no se vende, debido a que se decide que pase a formar parte de los activos del banco, que sea donado, etc. Las ventas pueden presentar utilidad o pérdida con respecto al valor de adjudicación. En caso de pérdida se requiere de una Ficha de Castigo para completar el proceso de venta.

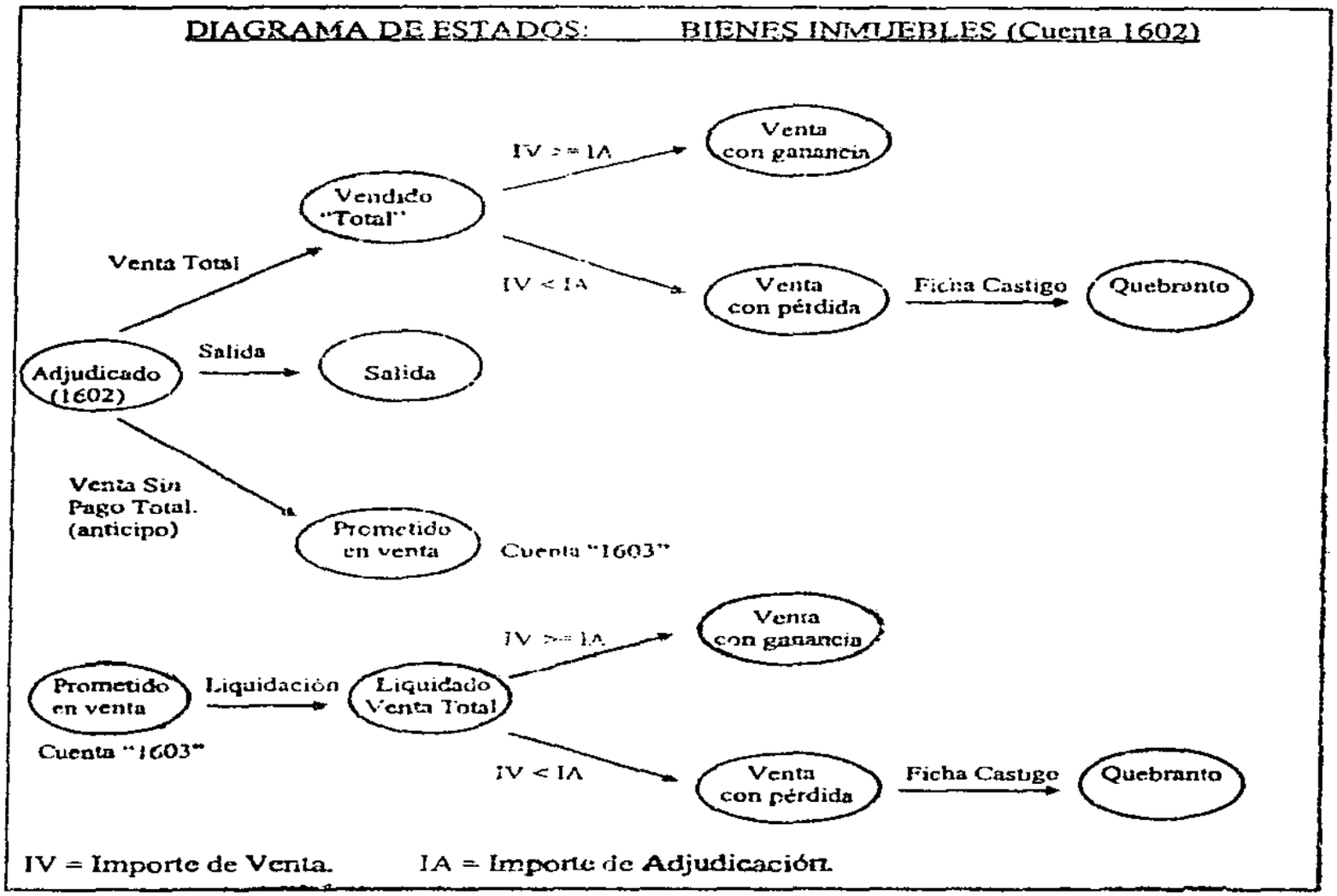


Fig. 3.3 Diagrama de Estados de Bienes Inmuebles.

Los Bienes Inmuebles (terrenos, casas, edificios, bodegas, departamentos, etc.) al venderse pueden ser pagados totalmente, o pueden ser vendidos mediante el pago de un anticipo. Cuando los Bienes están en este caso se les llama "prometidos en venta" y queda un remanente pendiente de pago. Este remanente posteriormente debe ser liquidado para concluir el proceso de la venta.

La salida se presenta cuando un Bien no se vende, debido a que se decide que pase a formar parte de los activos del banco, que sea donado, etc. Las ventas pueden presentar utilidad o pérdida con respecto al valor de adjudicación. En caso de pérdida se requiere de una Ficha de Castigo para completar el proceso de venta.

4. ANÁLISIS Y DISEÑO DEL SISTEMA.

4.1 Introducción.

Los métodos orientados a objetos para el análisis de requisitos del software permiten al analista obtener el modelo de un problema representando clases, objetos, atributos y operaciones como componentes principales de la modelización. Los puntos de vista orientados a los objetos combinan la clasificación de objetos, la herencia, de atributos y los mensajes de comunicación dentro del contexto de la notación de modelización.

Los objetos modelizan casi cualquier aspecto identificable del ámbito del problema: entidades externas, cosas, sucesos, papeles, unidades organizativas, y estructuras que pueden ser representados como objetos. Como punto importante, los objetos encapsulan datos y procesos. Las operaciones de procesamiento son parte del objeto y son iniciadas pasando un mensaje al objeto. Una definición de una clase forma la base para la reusabilidad en los niveles de modelización, diseño e implementación. Se pueden instanciar nuevos objetos de una clase.

El método de análisis orientado a los objetos proporciona una notación y un conjunto de heurísticas, para construir un modelo de Análisis Orientado a objetos (AOO). Para construir una especificación gráfica de un sistema basado en computadoras, se utilizan estructuras, temas, conexiones de instancia, y caminos de mensajes.

Se puede ver la modelización de datos como un subconjunto del AOO. Usando el diagrama entidad relación como una notación principal, la modelización de datos se centra en la definición de objetos de datos (objetos que no encapsulan el procesamiento) y en la manera en que están relacionados entre sí. La modelización de datos se usa en aplicaciones con procesamiento de datos masivos y se puede aplicar como notación complementaria para el análisis estructurado.

Con base en la descripción de la problemática que se tiene en el sector financiero y el diagrama de procesos que nos permite ver de forma específica el modo de operar de la información y las partes que se hayan involucradas, es posible abstraer del mundo real los siguientes objetos.

El desarrollo de este análisis pretende observar e identificar todas los componentes que son parte esencial en las operaciones que se realizan en las áreas destinadas para la administración de los bienes. Dentro de este análisis se podrán aplicar las técnicas que son utilizadas para el análisis de

especificaciones y requerimientos, así como obtener de manera concentrada en forma de diagramas el resumen del objetivo general del sistema.

4.2 Análisis del Diagrama de Procesos.

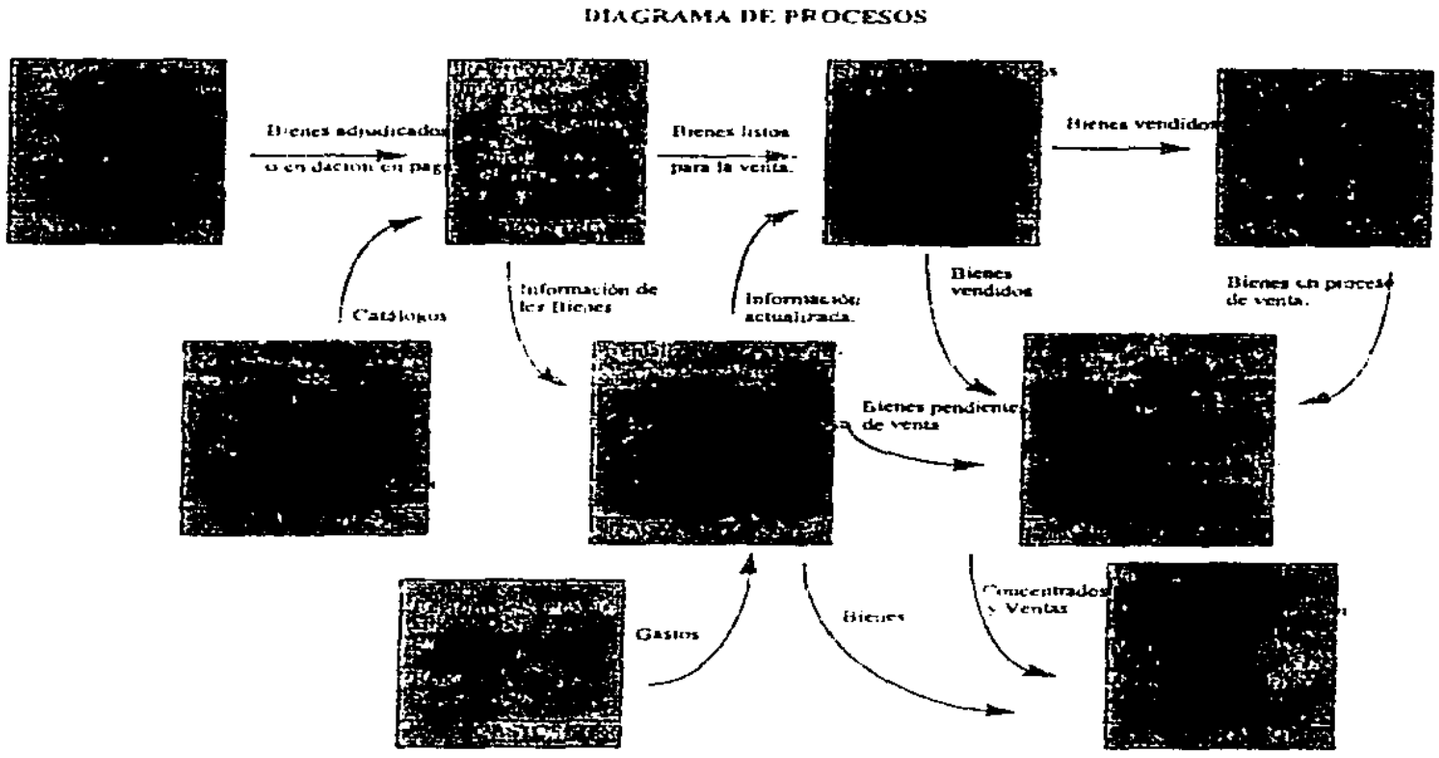


Fig.4.1. Diagrama de Análisis del Proceso.

De acuerdo a este diagrama podemos realizar la identificación de las entidades que componen las partes principales para la venta y administración de los bienes, a continuación se muestra la definición de las entidades detectadas.

Entidad Deudores.

Los deudores son todas aquellas entidades de las cuales se van a adjudicar los bienes, es decir, son aquellas personas fisicas o morales que tienen problemas por falta de pago de un crédito en el tiempo convenido entre las personas y la institución financiera, por lo tanto adjudica un bien o éstas lo dan en dación; esta entidad es utilizada como referencia, para el control y manejo adecuado de los bienes, así como la toma de decisiones y estadísticas. Todo bien que es ingresado al sistema tiene como referencia un deudor, dentro de las características de utilidad para la institución como referencias informativas son:

Nombre. El cual especifica de manera concreta qué persona física o moral se le adjudicó el bien, lo cual es una simple referencia porque la función primordial es administrar los bienes sin importar a qué persona se le adjudicó el bien.

Ubicación. Este atributo es de importancia, porque a través de éste se pueden realizar diferentes tipos de estadísticas por zonas geográficas, consultas de bienes disponibles por sucursales o regiones, que permiten realizar una administración de los bienes de forma más adecuada.

Tipo de Pago. El cual únicamente nos indica cuál fue la forma de entrada del bien a la institución, ya sea por adjudicación o por dación

Importe Capital. Es el importe por el cual se le está adjudicando el bien, o sea es el valor de la deuda.

Importe de Intereses Normales. Especifica el monto total de los intereses que fueron aplicados al importe del capital.

Importe de Intereses Moratorios. Los cuales son el resultado de no haber podido cubrir el monto de los intereses normales en el tiempo establecido.

Importe por otras causas. Que pueden ser de diversa índole, como son gastos de investigación, de operación, etc.

La forma de manipulación de los deudores común es:

Alta, es el registro de un nuevo deudor, al cual se le van a relacionar bienes.

Baja, implica la eliminación física del deudor, el cual puede ser dado de baja, porque ya no tiene bienes asociados, o bien por errores en los datos proporcionados con anterioridad.

Actualización, que permite realizar alguna modificación específica sobre los atributos del deudor, como lo es el nombre, o los importes de la deuda y/o intereses.

Depuración, que es un proceso que se realiza con frecuencia para desechar todas aquellas referencias que ya no se ocupan y de esa manera eliminarlo.
De acuerdo a la descripción anterior la definición de la entidad quedaría de la siguiente manera:

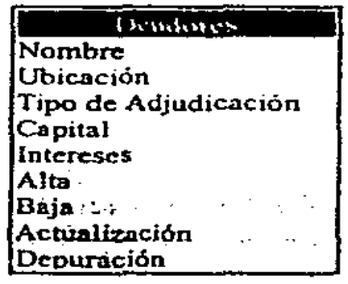


Diagrama de la Entidad Deudores

Entidad Bienes.

El propósito de las áreas de bienes adjudicados del sector financiero es la administración y comercialización de los bienes, porque estos bienes representan una pérdida dado que provienen de un crédito que tuvo problemas de pago, por lo tanto es necesario tratar de recuperar el valor del crédito lo antes posible a través de la venta del bien.

En el sector financiero, un bien es cualquier cosa susceptible de satisfacer una necesidad humana, y se clasifican en dos grandes grupos:

Bienes Muebles. Es todo bien que puede ser trasladado de un lugar a otro. Estos bienes pueden tener a su vez una subclasificación para poder diferenciarlos de acuerdo a su tipo en automóviles, obras de arte, electrónica, etc., una característica de este tipo de bienes es que pueden ser susceptibles de almacenaje y poseen una factura.

Bienes Inmuebles. Pertenecen a los bienes raíces, esto quiere decir que son una porción de terreno o construcción el cual tiene un área bien comprendida y tiene una ubicación bien definida, por lo tanto pertenece a un centro regional y a un estado. La clasificación de los bienes inmuebles pueden ser terrenos, casas, edificios, etc. Este tipo de bienes se asignan a un corredor, que realiza la promoción de venta del bien hacia personas físicas o morales.

En este momento se pueden observar una Clasificación de los Bienes, que es de vital importancia, ya que a través de ésta es posible tener un control más estricto de los bienes, así como poder obtener información de los bienes que son adquiridos.

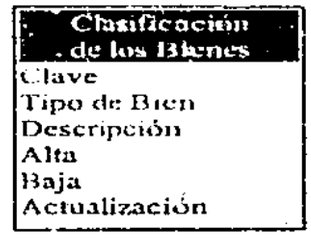


Diagrama de la Entidad Clasificación de Bienes

Para efecto de la administración de los bienes, se colectan una serie de propiedades que facilitan tanto la identificación como el cambio de la situación del bien.

Cuando un bien es adjudicado, se deben de tener en cuenta las siguientes características:

Deudor. Es indispensable saber de qué deudor proviene el bien, ya que este proporciona la información necesaria de la ubicación del bien.

No. Consecutivo de los bienes por deudor. Es un proporcionado por las personas responsables del registro de los bienes, que constituye una seriación de los bienes adjudicados a un solo deudor.

Tipo de Bien. Proporciona la cuenta contable, correspondiente al bien. Las cuentas pueden ser las siguientes de acuerdo a un estándar del sector financiero :

- 1601 corresponde a los bienes muebles
- 1602 corresponde a los bienes inmuebles
- 1603 corresponde a los bienes inmuebles prometidos en venta.

Descripción a detalle. Información con lujo de detalle sobre el bien adquirido.

Descripción resumida. Breve descripción del Bien.

Clasificación del bien. Proporcionar el subtipo de bien específico.

Fecha de adjudicación. Fecha en la que se realiza el registro de alta.

Importe de adjudicación. Valor con el cual se registra el bien.

Fecha de avalúo. Fecha en la que se realiza una valoración del bien .

Importe de avalúo. Importe asignado por el valuador del bien.

Asignado a corredor(Inmueble). Indica si el bien ya está siendo promocionado para su venta.

Nombre del corredor(inmueble). Nombre de la empresa de venta de Bienes Raíces.

Bodega en donde se encuentra el bien(mueble). Bodega en donde se encuentra almacenado el bien.

Estatus Jurídico. Permite saber si el bien adjudicado, se encuentra con problemas del tipo jurídico, que son de diferente índole dependiendo el tipo del bien.

Indicador de posesión del bien. El cual informa si el bien lo tiene físicamente la institución financiera.

Indicador de escrituras(Inmuebles). Indica si se cuenta con las escrituras del bien.

Indicador de factura(Muebles). Indica si se cuenta con las facturas del bien.

Indicador de seguro. Permite saber si el bien tiene seguro.

Indicador de fotografía. En caso de contar con fotografía del bien se marca.

Listo para la venta. Este atributo es verdadero si se cumplen con las siguientes condiciones :

- No tener conflicto jurídico
- Contar con facturas o escrituras
- Tener posesión del bien

Importe de venta. Monto con el cual se vendió el bien.

Fecha de venta. Fecha en la que el bien sea vendido.

Estatus del bien. Indica en qué situación se encuentra el bien.

Observaciones. Algunas indicaciones que se puedan añadir con respecto al bien.

Ubicación. Se debe de tener la referencia de dónde se encuentra el bien, en el caso de los bienes inmuebles deben de tener como residencia una bodega en donde se almacena temporalmente el bien, las bodegas normalmente son referenciadas por una descripción. Para los bienes inmuebles se debe de tener toda la información del domicilio de donde se encuentra localizado físicamente el bien.

Estos dos tipos de ubicaciones, comparten atributos comunes como son : Calle, Colonia, Código Postal, Ciudad, Estado, Responsable, Teléfono, Fax

Las ubicaciones de los bienes inmuebles, siempre van a estar relacionadas con el deudor, mientras que las bodegas siempre cuentan con una clave y descriptivo diferente para diferenciarlas entre sí.

Por lo tanto se pueden clasificar las entidades de la siguiente forma:

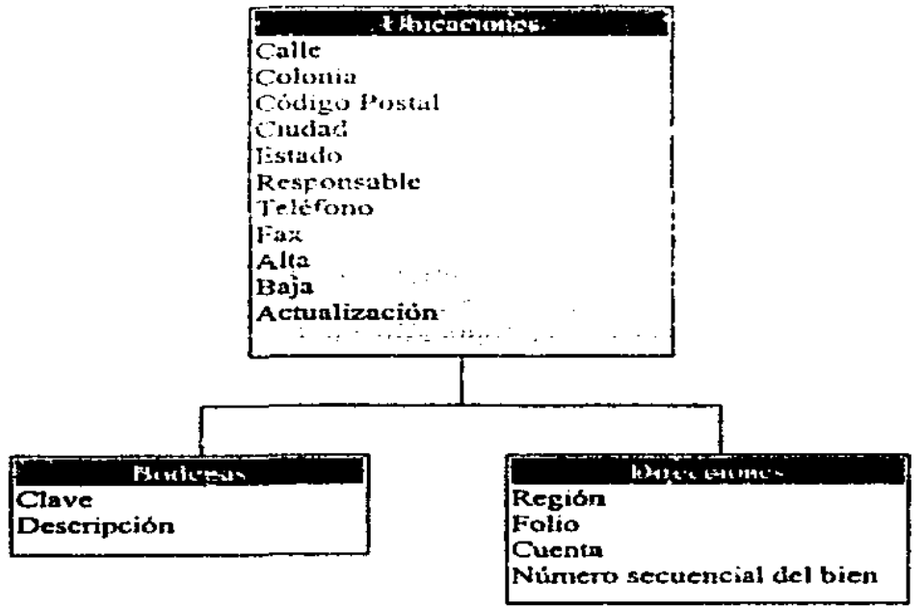


Diagrama de la Entidad Ubicaciones y sus derivaciones correspondientes.

Dentro de los procesos que se pueden identificar en la manipulación de los bienes se encuentran las siguientes:

Alta. Permite registrar un bien con todos o casi todos los atributos descritos con anterioridad.

Baja. Eliminar un registro que no este bien definido, o que no se vaya a utilizar, o bien ya haya sido vendido.

Actualización. Modificar algunos atributos que permitan registrar algún atributo que se haya cambiado de estatus o bien poner en el bien conceptos asociados para su utilización en otros procesos.

Gastos. En este proceso, se definen algunos gastos de administración para los bienes, que dependen del tipo de bien que sea, por ejemplo, a un bien inmueble se le deben de pagar los servicios de agua, luz, teléfono, etc., y a un bien mueble se pueden pagar gastos de almacén, transportación, etc.

Aquí vale la pena mencionar que este tipo de gastos normalmente son los mismos, para cada tipo de bienes, por lo tanto en este proceso se puede identificar otra entidad a la que podemos denominar la entidad de Gastos.

Para las diferentes características de los gastos de los bienes se pueden identificar los siguientes:

Tipo de bien. Para saber qué tipo de gasto se le va asociar.

Descripción y Clave de gasto. Para tener identificados los diferentes tipos de gastos.

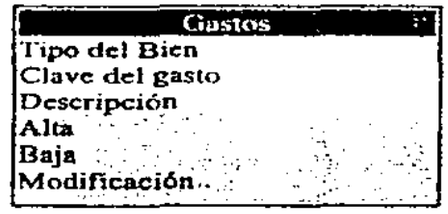


Diagrama de la Entidad Gastos

Esta entidad no es muy compleja, sin embargo es necesario definirla para efectos de una buena estructuración y una óptima administración de la información. Los gastos pueden ser de diferente índole y casi siempre son los mismos para el tipo de bien especificado, sin embargo no son los únicos o bien pueden surgir otros tipos de gastos diferentes, por esa razón, es necesario separarla y definirla como una entidad, que tenga la capacidad de poder crecer sin realizar una modificación estructural con relevancia.

Venta. Este proceso es de gran importancia porque a través de éste es donde se consigue la venta de los bienes, ya que la finalidad del sector financiero es lograr la recuperación del capital de los créditos que no fueron pagados y por lo tanto se pretenden vender los bienes en el menor tiempo posible.

Para que un bien pueda ser vendido, debe de cumplir con ciertas condiciones como son:

- Tener posesión del bien.
- Tener escrituras o facturas
- No tener ningún tipo de problema jurídico

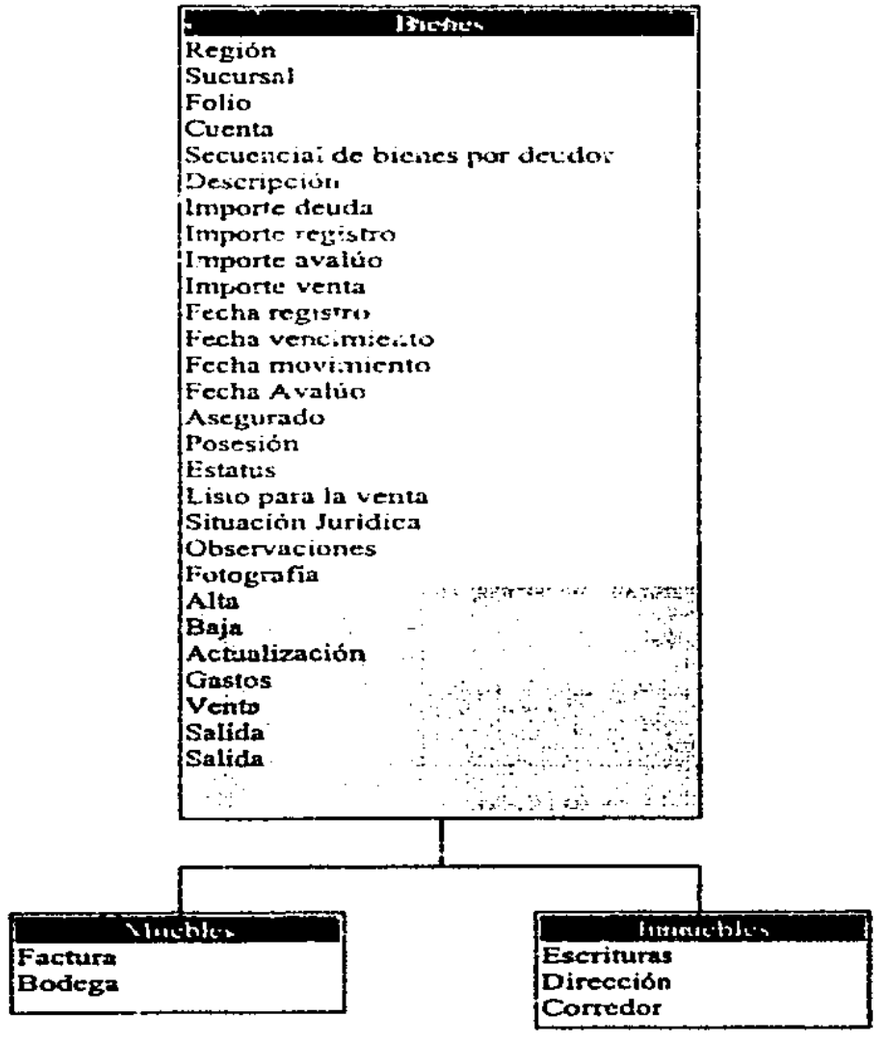


Diagrama de la Entidad Bienes y sus derivaciones

Dentro de nuestro análisis encontramos otros tipos de objetos que están de manera implícita, porque son parte esencial para que algunos objetos se encuentren bien clasificados y estructurados, este tipo de objetos se describen a continuación:

Entidad Centro Regional.

El centro regional es una entidad como un marco de estudio, un contexto territorial socioeconómico para la ordenación del territorio, la cual permite clasificar y obtener información de una manera más concisa sobre ciertas partes del país. En todas las instituciones del tipo financiero, este tipo de entidades es viable y utilizada. En general las características de esta entidad son:

Clave. Que define una entidad como única.

Descripción. Para identificar el centro regional con un nombre específico.

Alta. Para poder realizar la clasificación de la información por zonas regionales, es necesario ingresar una nueva región.

Baja. Cuando una región no es utilizada es posible realizar la eliminación del centro regional.

Actualización. Esta actualización solamente es para redefinir la descripción del centro regional.

Entidad Sucursales.

Las sucursales son las entidades que están asociados con los deudores, ya que en ésta es donde no se cubrió el valor del crédito por el cual se adjudican los bienes.

Una sucursal tiene asociada un centro regional, el cual es parte primordial para poder realizar el estudio del comportamiento de la relación de pérdidas y ganancias económicas por los diferentes sectores que constituyen nuestra región.

Los procesos asociados más relevantes y observados para las sucursales son:

Alta. La cual constituye realizar la identificación y definición de la sucursal, para su uso para propósitos de la administración de los bienes.

Baja. Cuando una sucursal ya no es utilizada o simplemente desaparece, es posible eliminarla, para ya no contar con información que no tiene validez.

Actualización. Para el caso de las sucursales es el cambio de descripción de la sucursal.

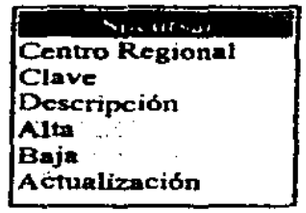


Diagrama de la Entidad Sucursal

Entidad Estatus Jurídico

El estatus jurídico es la situación legal que tiene el activo, este es un indicador si el bien tiene un problema del tipo legal. La clasificación de los problemas jurídicos son casi del mismo tipo, para cualquier bien, por lo tanto se utiliza realizando un catálogo de posibles causas de problemas jurídicos, las cuales comparten los siguientes atributos:

Clave. Para identificación del tipo de problema jurídico.

Descripción. Explicación del problema.

Los procesos que se utilizan en esta entidad son:

Alta. Para los nuevos tipos de problemas jurídicos.

Baja. Para eliminar las causas de problemas del tipo jurídico obsoletas.

Actualización. Para corregir o modificar la descripción del problema.

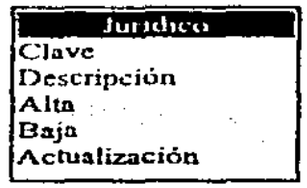
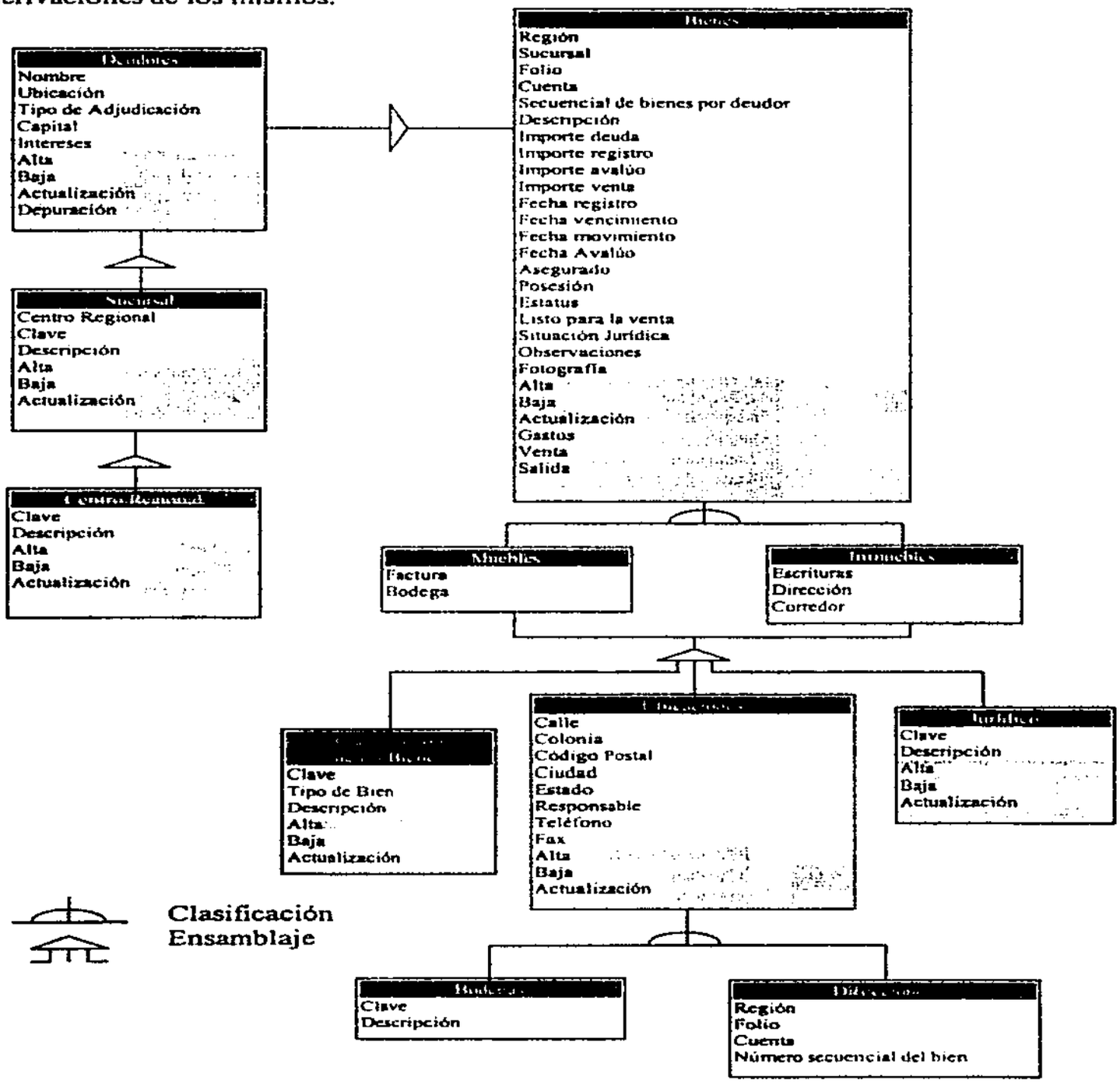


Diagrama de la Entidad Estatus Jurídico

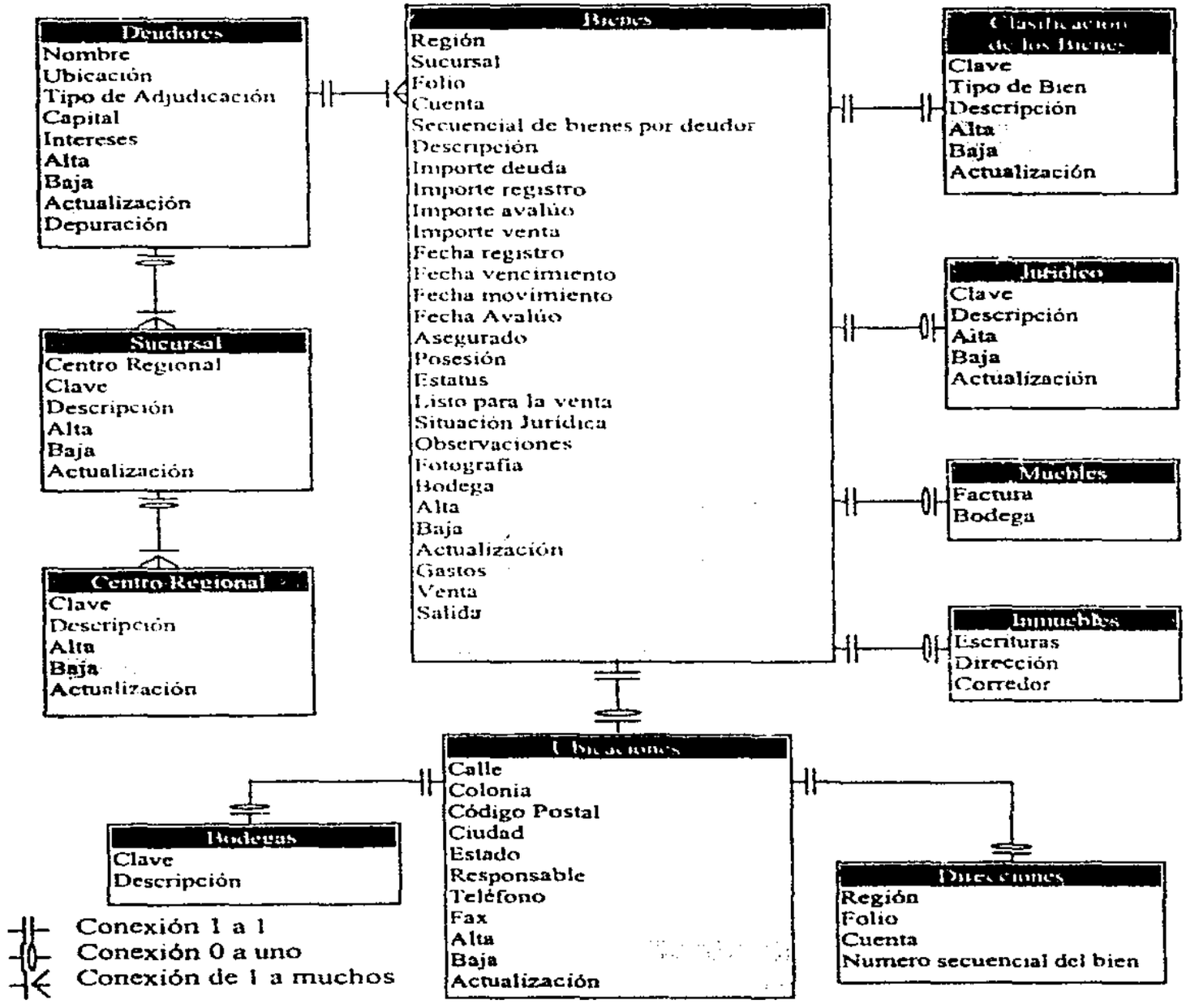
4.3 Diagrama Temático del Proceso.

Este tipo de diagrama permite identificar la dependencia entre los objetos, así como las derivaciones de los mismos.



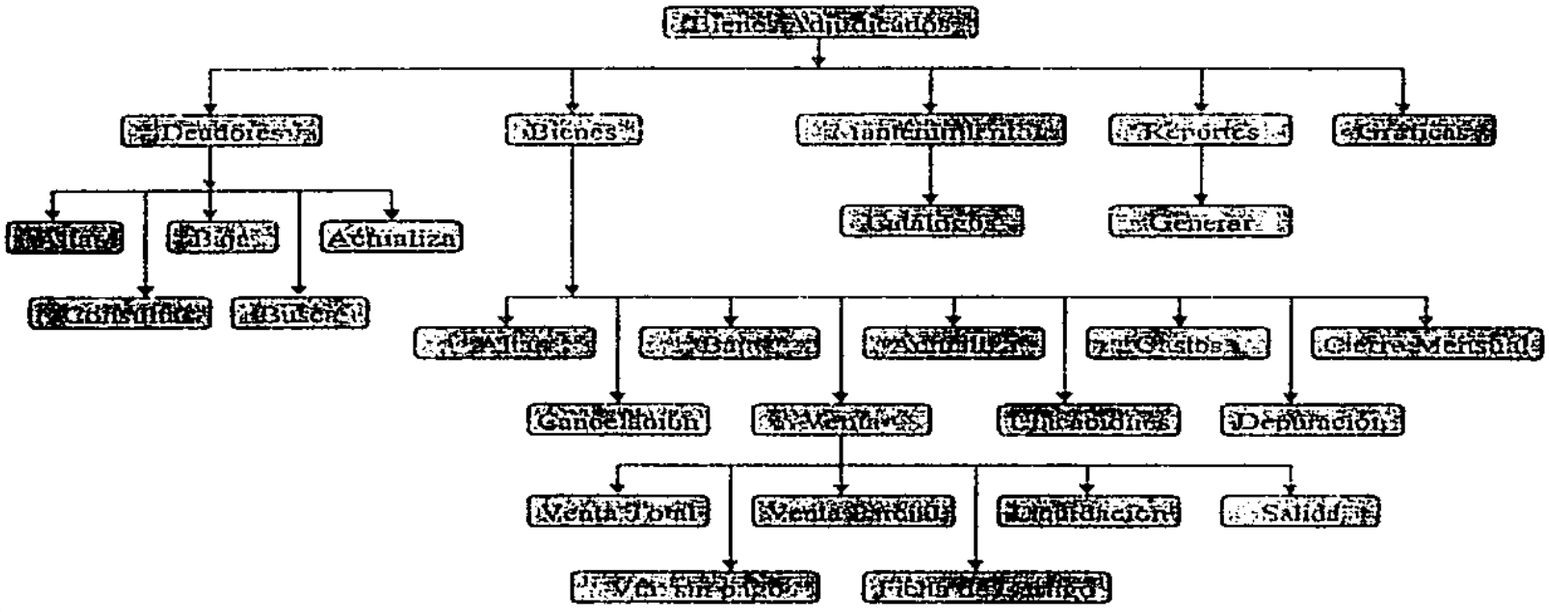
4.4 Diagrama Entidad-Relación.

El siguiente diagrama tiene como objetivo presentar de una manera general la interacción que tienen los objetos y la relación que guardan los entre si, es de suma importancia, dado que a través de éste es posible detectar algunos aspectos del análisis que no fueron considerados o bien que no pertenecen a la definición de la problemática dada.



4.5 Carta de Estructura.

Como una necesidad de resumir las funciones de los procesos que fueron identificadas en los objetos, se realiza este diagrama, el cual no es más que la descripción de la funcionalidad que conllevan cada uno de los procesos. La carta de estructura que se identificó en este análisis es la siguiente:



Deudores

- ***Alta.*** Permite el ingreso de un nuevo deudor al sistema identificando la región y sucursal a la que pertenece, incluyendo un número de folio que permite su identificación en ese conjunto.
- ***Baja.*** Permite dar de baja físicamente al deudor.
- ***Actualiza.*** Modifica los atributos de los objetos en la base de datos por la información proporcionada.
- ***Consulta.*** Permite la visualización de un folder con la información de todos los deudores que tiene registrada la base de datos, con opción a elegir uno de ellos para buscar o asociar un bien mueble / inmueble determinado.
- ***Busca.*** Realiza una búsqueda mucho más rápida con base en el número de bien asignado en el sistema.

- ***Bienes.*** Abre un portafolios con todos los bienes adjudicados asociados al deudor. En este punto es importante señalar que mientras el deudor no esté registrado en la base de datos, no podrá abrir ésta opción del módulo.

Bienes

- ***Alta.*** Permite el ingreso de un nuevo bien al sistema.
- ***Baja.*** Permite borrar físicamente la información de un bien asociado.
- ***Actualiza.*** Actualiza la información en la base de datos para cualquier modificación en la plantilla de ingreso de datos.
- ***Gastos.*** Abre un módulo que permite el ingreso de todos los gastos efectuados sobre dicho bien.
- ***Ubicaciones.*** Permite la asignación de dirección o bodega donde se encuentra el bien.
- ***Cancelación de movimientos de venta o de salida.*** Permite el regreso de un bien al que ya ha aplicado un tipo de venta a un estatus que tenía anteriormente.

Los diferentes tipos de ventas que se pueden efectuar sobre un bien son :

- ***Venta sin pago total.*** Permite la venta de un bien sin liquidarlo totalmente, es decir, mediante un anticipo. Es aplicable solamente a bienes inmuebles.
- ***Liquidación.*** Permite la liquidación del remanente, resultado de una venta sin pago total.
- ***Salida.*** Permite la salida de un bien sin venderlo. Aplicable a bienes muebles e inmuebles.
- ***Venta Parcial.*** Permite la venta parcial de un bien. Separa del bien, la parte proporcional que va a ser vendida.
- ***Ficha de castigo.*** Es necesario la aplicación de este proceso a los bienes cuyo importe de venta resulta menor a su valor de adjudicación.

Reportes

- ***Generar.*** Abre de la carpeta el reporte seleccionado.
  Dentro de los reportes considerados se encuentran los siguientes:
  - Relación de bienes de acuerdo a criterios.
  - Cuadernillo.
  - Movimientos contables del mes.

- Información para la CNB.
- Entradas y salidas del mes.
- Relación de bienes listos para su venta.
- Comparativo mensual (sumaria de las cuentas).
- Comparativo mensual (informe de tres meses)
- Resumen de bienes de acuerdo a su clasificación.
- Resumen trimestral de bienes.

En la construcción del diseño de los procesos se describen la información requerida para cada uno de estos reportes.

- ***Proceso mensual.*** Realiza un proceso que se efectúa al final de cada mes, en el cual se generan los concentrados que permiten la comparación de un mes con respecto al anterior.

Mantenimiento a Catálogos.

Realiza el mantenimiento de los siguientes catálogos:

- ***Región.*** Permite los movimientos de altas, bajas y cambios solicitando la clave y la descripción del mismo.

- ***Clasificación de bienes.*** Permite los movimientos de altas, bajas y cambios de un concepto de acuerdo al tipo de cuenta al que pertenece (1601 o 1602).

- ***Estatus jurídico.*** Permite los movimientos de altas, bajas y cambios del estatus que guarda un bien en específico.

- ***Sucursal.*** Permite los movimientos de altas, bajas y cambios de dicho concepto asociándolo a una región.

- ***Gastos.*** Permite el ingreso de altas, bajas y cambios de conceptos que se encuentran registrados en el catálogo y que son aplicables a los bienes muebles e inmuebles.

Depuración de registros.

Permite el borrado físico de aquella información que el usuario considere conveniente, seleccionando el bien y su deudor asociado filtrando la información por región y sucursal.

Gráficas.

Permite realizar un análisis para la toma de decisiones de acuerdo a una estadística, gráfica que se realiza con base en la información almacenada en la base de datos.

4.6 Diseño de la Base de Datos.

El almacenamiento de los datos constituye una parte fundamental en el diseño de cualquier sistema ya que a través de él giran todos los procesos que se efectúan para su recuperación y tratamiento en la parte del cliente, y que nos sirve como base para lograr que se convierta en información importante que ayude en la toma de decisiones. Es por ello que es necesario retomar algunos aspectos teóricos para comprender su importancia en la etapa del análisis y diseño.

Hay ciertos objetivos que deben alcanzarse para lograr un buen diseño:

- Los datos deben estar disponibles para cuando el usuario desee usarlos.
- Los datos deben ser precisos y consistentes, es decir, deben poseer una integridad.
- Los datos deben almacenarse de una manera eficiente.
- Los datos deben actualizarse y grabarse eficientemente.

Generalmente existen dos enfoques para almacenar los datos y que se usan de manera tradicional:

El primer enfoque consiste en almacenar los datos en archivos individuales, exclusivos para una aplicación en particular. Cuenta con la característica de que toda la información necesaria se concentra en archivos separados, lo cual implica que los mismos datos se almacenen en más de un lugar. Un ejemplo muy significativo de este caso consistiría en la manera como el área de Bienes Adjudicados realizaba sus operaciones de control de los bienes que tenía a su cargo, mediante el uso de archivos de hojas de cálculo sobre las cuales efectuaba operaciones contables para hacer que los resultados correspondieran entre sí.

El segundo enfoque consiste en la elaboración de un sistema de base de datos que permita un almacenamiento formalmente definido y controlado de una manera centralizada, de tal manera que permita atender a distintas aplicaciones. Se maneja como una fuente central de datos que es controlada por el Manejador de la Base de Datos (DBMS Database Management System) que permite la creación, modificación y actualización de toda la información involucrada, permitiendo la explotación de la información para propósitos de consulta y de reportes. De las características anteriores se desprenden a su vez objetivos que deben cumplir este enfoque para lograr ser más eficiente:

1. Asegurar que los datos puedan ser compartidos por los usuarios para una variedad de aplicaciones.
2. Que el almacenamiento de datos sea preciso y consistente.
3. Asegurar que todos los datos requeridos para las aplicaciones presentes y futuras se encuentren disponibles.
4. Permitir que la base de datos evolucione y se adapte a las necesidades crecientes de los usuarios.

Haciendo una comparación entre los dos enfoques expuestos, se observa que el segundo cuenta con una estructura ya definida en cuanto a la administración y manejo, permitiendo además que los datos se almacenen al menos una vez, apoyando así un objetivo implícito que

consiste en mantener la integridad de la información. A pesar de que cuenta con algunas desventajas como es el hecho de que los datos al estar almacenados en un solo lugar son más vulnerables a accidentes, requiriendo así un respaldo completo, es el esquema que se maneja con mayor frecuencia en las aplicaciones comerciales y el que se manejará a continuación.

Al abordar un esquema de gestión de base de datos es preciso puntualizar que éste como tal es una colección de archivos interrelacionados y un conjunto de programas que permiten a los usuarios acceder y modificar dichos archivos. Uno de sus objetivos es el mostrar a los usuarios una visión abstracta de los datos, lográndose mediante la definición de tres niveles de abstracción en los que se puede ver la base de datos:

- Nivel físico. En este nivel se describe la manera en que la información se guarda en los dispositivos físicos de almacenamiento representados por posiciones de memoria y que permiten extraer la información de una manera eficiente.
- Nivel Conceptual. Aquí se describen los datos que son realmente almacenados en la base de datos y las relaciones que existen entre ambos. Generalmente este tipo de actividades es efectuada por el Administrador de la Base de Datos (DBA) quien debe estipular la manera en que los datos van a ser guardados para un mejor uso de las aplicaciones que se encarguen de explotarlos.
- Nivel de visión. Aquí se define la parte de la base de datos que se le presentará y que puede manipular un determinado usuario y que se utiliza comúnmente para propósitos de control de acceso a la información.

Para la definición de la estructura interna de la base de datos se parte de un concepto llamado modelo de datos con el cual se establecen las herramientas conceptuales para escribir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia. Los diversos modelos de datos que se han propuesto se dividen en tres grupos:

1. Modelos físicos de datos.
2. Modelos lógicos basados en objetos.
3. Modelos lógicos basados en registros.

A continuación se dará una breve explicación de los tres diferentes modelos anteriores:

1. Modelos físicos de datos.

Los modelos físicos de datos se usan para describir datos en el nivel más bajo. A diferencia de los modelos lógicos de datos, hay muy pocos modelos físicos en uso. Dos modelos que se ubican en esta rama son:

- Modelo unificador
- Memoria de elementos.

2. **Modelos lógicos basados en objetos**. Se utilizan para describir datos en los niveles conceptual y de visión. Permiten definir restricciones de datos explícitamente. Existen algunos modelos conocidos como son:

- ***El modelo entidad-relación.***

El modelo entidad-relación (E-R) se basa en una precepción del mundo real que consiste en una colección de objetos básicos llamados entidades y relaciones entre estos objetos.

Una entidad es cualquier objecto o evento acerca del cual se recolectan datos. Una entidad puede ser una persona, un lugar o un objeto. Por ejemplo, aplicando este concepto para el diseño del sistema, encontraremos los objetos ***Bienes***, ***Gastos***, ***Deudores*** y ***Movimientos*** como las principales entidades que maneja el sistema y que contienen el volumen principal de la información.

Una relación es una asociación que existe entre varias entidades. Generalmente existen tres diferentes tipos de relaciones, sin embargo, para ejemplificarlas, se hace necesario mostrar los componentes que conforman un diagrama E-R, el cual es una representación gráfica de la estructura lógica de la base de datos mediante los siguientes componentes:

a) Rectangulos, que representan conjuntos de entidades.
b) Elipses, que representan atributos los cuales son una caracteristica de una entidad. Por ejemplo un bien puede tener muchos atributos, tales como estar escriturado, estar listo para la venta, tener un importe de adjudicación, etc. A esta característica también suele denominarse como datos, y de hecho son las unidades más pequeñas dentro de una base de datos.
c) Rombos, que representan relaciones entre conjuntos de entidades.
d) Líneas, que conectan atributos a conjuntos de entidades y conjuntos de entidades a relaciones.

- *Relación de uno a uno*. Designada por la nomenclatura **1:1**, nos muestra que para cada bien existe una ubicación específica, designada por una dirección.

- *Relación de uno a muchos*. Designado como **1:M**, nos muestra que un deudor puede tener muchos bienes adjudicados o dados en dación.

- *Relación de muchos a muchos*. Desigando como **M:M** describe la posiblidad de que las entidades puedan tener numerosas asociasiones en cualquier dirección, por ejemplo: un bien puede tener muchos gastos, mientras que al mismo tiempo, un gasto puede originarse del mantenimiento de diversos bienes.

- ***El modelo orientado a objetos.***

Se basa en una colección de objetos que contienen valores almacenados en variables de instancia, que representan objetos por sí mismos. Un objeto también contiene partes de código que operan sobre el objeto llamados métodos. Los objetos que contienen los mismos tipos de valores y los mismos métodos se agrupan en clases. Una clase puede ser vista como una definición de tipo para objetos. La única forma en la que un objeto puede acceder a los datos de otro objeto es invocando a un método de ese otro objeto. Esto se llama envío de un mensaje al objeto.

2. Modelos lógicos basados en registros.

Se utilizan para describir datos en los modelos conceptual y físico, permitiendo definir la estructura lógica global de la base de datos. Este modelo se llama así debido a que la base de datos está estructurada en registros de formato físico de varios tipos. Cada tipo de registro define un número fijo de campos o atributos, y cada campo normalmente es de longitud fija. Los tres modelos de datos más ampliamente aceptados son los modelos relacional, de red, y jerárquico. A continuación se dará una explicación de cada modelo:

- ***Modelo Relacional.*** Representa los datos y las relaciones entre los datos mediante una colección de tablas, cada una de las cuales tiene un número de columnas con nombres únicos.
- ***Modelo de Red.*** Aquí los datos se representan mediante una colección de registros y las relaciones entre los datos se representan mendiante enlaces, los cuales pueden verse como apuntadores. Los registros en la base de datos se organizan como colecciones de grafos arbitrarios.
- ***Modelo Jerárquico.*** Este modelo es similar al modelo de red en el sentido de que los datos y las relaciones entre los datos se representan mediante registros y enlaces respectivamente. Se diferencia del modelo de red en que los registros están organizados como colecciones de árboles en vez de grafos arbitrarios.

Hoy en día el modelo relacional es el que comúnmente utilizan los principales manejadores de datos debido a que presentan la ventaja sobre los dos modelos anteriores en que no utiliza apuntadores para definir las relaciones entre los registros, sino que las relaciones que se establecen son a través de los valores que contienen los mismos, lo cual implica una mayor libertad de consultas y la no repetición de la información.

Un concepto muy importante que se utiliza en el modelo relacional y que permite establecer las relaciones entre las distintas entidades es el de llave primaria, el cual identifica de manera exclusiva a un registro dentro de una tabla. Este concepto es uno de los elementos que hay que buscar durante el diseño de una base de datos relacional que nos permitan evitar la repetición de la información.

Existen algunas reglas para lograr un buen diseño de una base de datos relacional , que aunque no se tratarán de profundizar se mencionarán debido a su importancia y que consisten en la normalización de los datos, el cual es un proceso de transformación de las estructuras de datos a un esquema mucho más pequeño. Se manejan tres pasos para lograr una normalización suficiente:

1. (1FN) Comienza con un proceso de eliminación de grupos repetidos y la identificación de la llave que define al criterio primario. Aquí la relación principal se desglosa en dos o más relaciones.
2. (2FN) Asegura que todos los atributos no llave, o sin llave, sean dependientes de la llave del criterio primario.Todas las dependencias normales se eliminan y se colocan en otra relación.
3. (3FN) Elimina cualquier dependencia transitoria. Una dependencia transitoria es aquella en la cual sus atributos no-llave son dependientes de otros atributos no llaves.

Con los elementos anteriormente expuestos podemos aplicarlos en la definición del esquema de la base de datos del presente trabajo.

Aplicación.

A continuación se define la composición física de las tablas en cuestión, identificando las llaves que permiten la integridad de unicidad y los campos que componen al registro:

Lista de Tablas

Nombre	Descripción
Bcbienes	Contiene todos lo bienes adjudicados o en dación
Bcdeudas	Contiene información con respecto al deudor
Bcgastos	Contiene información de los gastos efectuados sobre determinado bien
Bcmovim	Representan todos los movimientos de venta efectuados sobre los bienes
Bodegas	Contiene referencias a ubicaciones de almacenaje para bienes muebles.
Comparativo	Tabla temporal para propósitos de reportes involucrando a dos meses
Comparativo2	Tabla temporal para propósitos de reportes, utilizando hasta tres meses.
Concbienes	Tabla que guarda un histórico con un resumen de todos los movimientos efectuados en un mes.
Direcciones	Tabla para almacenar la ubicación física de bienes inmuebles
Param	Tabla que contiene todos los conceptos asociados en el sistema

Tabla : Param

Lista de Columnas

Nombre	Descripción	Tipo	llave Primaria	Campo Nulo
Datospar	Clave asociada	char(58)	No	Yes
Iparam	identificación de la sección que invoca	char(19)	Yes	No
Itipopar	Parámetro que depende del tipo de aplicación	char(3)	Yes	No

Tabla: Bcdeudas

Lista de Columnas

Nombre	Descripción	Tipo	Llave primaria	Campo Nulo
Cimpadeudo	Importe total del adeudo	numeric(12,2)	No	Yes
Cimpcapital	Composición del adeudo por concepto de capital	numeric(12,2)	No	No
Cimpintmora	Composición del adeudo por concepto de intereses moratorios	numeric(12,2)	No	No
Cimpintnorm	Composición del adeudo por concepto de intereses normales	numeric(12,2)	No	No
Cimpotros	Otro tipo de adeudo	numeric(12,2)	No	No
Edeudor	Nombre del deudor	char(60)	No	No
Ifolio	Folio que identifica al deudor	numeric(5)	Yes	No
Iregion	Región en que se ha clasificado	char(1)	Yes	No
Isucursal	Sucursal en que se ha clasificado	char(3)	No	Yes
Sadjdac	Para determinar si fue por adjudicación o dación.	char(2)	No	Yes

Tabla: Bcbienes

Lista de Columnas

Nombre	Descripcion	Tipo	Llave Primaria	Campo Nulo
Cimpavaluo	Importe de avalúo	numeric(12,2)	No	No
Cimporiginal	Importe de avalúo original	numeric(12,2)	No	Yes
Cimpregistro	Importe de registro	numeric(12,2)	No	Yes
Cimpventa	Importe de venta	numeric(12,2)	No	No
Edescorta	Descripción corta	char(100)	No	Yes
Edeslarga	Descripción larga	char(230)	No	Yes
Enomcorr	Nombre del corredor asociado	char(60)	No	No
Eobserv	Observaciones	char(300)	No	No
Favaluo	Fecha de avalúo	date	No	No
Fmovim	Fecha de movimiento de venta	date	No	No
Fregistro	Fecha de registro	date	No	Yes
Fvencim	Fecha de vencimiento	date	No	No
Iclasific	Clasificación del bien	char(3)	No	Yes
Icuenta	Cuenta 1601, 1602 o 1603	char(4)	Yes	No
Ifolio	Num de folio asociado al bien	numeric(3)	Yes	No
Iregion	Región a la que pertenece el bien	char(1)	Yes	No
Isec	Identificador de registro	numeric(5)	Yes	No
Isucursal	Sucursal a la que pertenece	char(3)	No	Yes
sasegurado	Switch de asegurado	char(2)	No	No
sasigcorr	Switch de asignado a corredor	char(2)	No	No
sbodega	Switch de bodega asociada	char(3)	No	No
sescrituras	Switch de escriturado	char(2)	No	No
sestatus	Estatus de venta que guarda el bien	char(2)	No	Yes
sfactura	Switch de facturación	char(2)	No	No
sfotografia	Switch de fotografía	char(2)	No	No
sjuridic	Muestra el estatus juridico	char(2)	No	No
slistovta	Switch para indicar si se encuentra listo para la venta	char(2)	No	Yes
sposesion	Switch para indicar si se posee físicamente el bien	char(2)	No	No

Tabla: Bcmovim

Lista de Columnas

Nombre	Descripción	Tipo	Llave Primaria	Campo Nulo
cimporte	Importe de venta o de salida	numeric(12,2)	No	No
cimpreg	Importe de registro	numeric(12,2)	No	No
cvemov	Identificador del tipo de movimiento efectuado	char(1)	No	Yes
edescorta	Descripción corta	char(100)	No	No
edeudor	Nombre del deudor	char(60)	No	No
fregistro	Fecha de registro	date	No	Yes
icuenta	Cuenta 1601, 1602,1603	char(4)	No	Yes
ifolio	Folio del bien asociado a la venta	numeric(3)	No	Yes
iregion	Región en que se clasificó	char(1)	No	No
isec	Folio identificador	numeric(5)	No	Yes
isucursal	Sucursal en que está clasificado el bien	char(3)	No	No

Tabla: Bodegas

Lista de Columnas

Nombre	Descripción	Tipo	Llave Primaria	Campo Nulo
calle	Calle de ubicación	char(25)	No	No
ciudad	Ciudad de ubicación	char(25)	No	Yes
codpos	Código Postal	char(25)	No	No
colonia	Colonia de ubicación	char(25)	No	No
cvebod	Clave de la bodega	char(3)	Yes	No
descripcion	Descripción	char(25)	No	Yes
estado	Estado	char(25)	No	Yes
fax	Fax	char(10)	No	No
responsable	Nombre del responsable	char(40)	No	No
teléfono	Teléfono	char(10)	No	No

Tabla: Comparativo

Lista de Columnas

Nombre	Descripcion	Tipo	Llave Primaria	Campo Nulo
ci1601l_1	Importe de bienes en 1601 listos para la venta en el mes 1	double	No	Yes
ci1601l_2	Importe de bienes en 1601 listos para la venta en el mes 2	double	No	Yes
ci1601nl_1	Importe de bienes en 1601 no listos para la venta en el mes 1	double	No	Yes
ci1601nl_2	Importe de bienes en 1602 listos para la venta en el mes 2	double	No	Yes
ci1602l_1	Importe de bienes en 1602 listos para la venta en el mes 1	double	No	Yes
ci1602l_2	Importe de bienes en 1602 listos para la venta en el mes 2	double	No	Yes
ci1602nl_1	Importe de bienes en 1602 no listos para la venta en el mes 1	double	No	Yes
ci1602nl_2	Importe de bienes en 1602 no listos para la venta en el mes 2	double	No	Yes
ci1603_1	Importe de bienes 1603 en el mes 1	double	No	Yes
ci1603_2	Importe de bienes en 1602 en el mes 2	double	No	Yes
cn1601l_1	Cantidad de bienes en 1601 listos para la venta en el mes 1	integer	No	Yes
cn1601l_2	Cantidad de bienes en 1601 listos para la venta en el mes 2	double	No	Yes
cn1601nl_1	Cantidad de bienes en 1601 no listos para la venta en el mes 1	double	No	Yes
cn1601nl_2	Cantidad de bienes en 1601 no listos para la venta en el mes 2	double	No	Yes
cn1602l_1	Cantidad de bienes en 1602 listos para la venta en el mes 1	double	No	Yes
cn1602l_2	Cantidad de bienes en 1602 listos para la venta en el mes 2	double	No	Yes
cn1602nl_1	Cantidad de bienes en 1602 no listos para la venta en el mes 1	double	No	Yes
cn1602nl_2	Cantidad de bienes en 1602 no listos para la venta en el mes 2	double	No	Yes
cn1603_1	Cantidad de bienes en 1603 mes 1	double	No	Yes
cn1603_2	Cantidad de bienes en 1603 mes 2	date	No	No
dmes1	Fecha del mes 1	date	No	No
dmes2	Fecha del mes 2	char(1)	Yes	No
iregion	Identificador de Región	char(3)	Yes	No
isucursal	Sucursal en que se clasificó			

Tabla: Comparativo2

Lista de Columnas

Nombre	Descripción	Tipo	Llave Primaria	Campo Nulo
cibienm1	Importe de bienes vendidos en el mes 1	double	No	Yes
cibienm2	Importe de bienes vendidos en el mes 2	double	No	Yes
cibienm3	Importe de bienes vendidos en el mes 3	double	No	Yes
CNBienm1	Cantidad de bienes vendidos en el mes 1	integer	No	Yes
CNBienm2	Cantidad de bienes vendidos en el mes 2	integer	No	Yes
CNBienm3	Cantidad de bienes vendidos en el mes 3	integer	No	Yes
cpbienm1	Porcentajes del total en el mes 1	double	No	No
cpbienm2	Porcentajes del total en el mes 2	double	No	No
cpbienm3	Porcentajes del total en el mes 3	double	No	No
dmes1	Fecha del mes 1	date	No	Yes
dmes2	Fecha del mes 2	date	No	No
dmes3	Fecha del mes 3	date	No	No
idenest	idenest	integer	No	No
iidenreg	Identificador de región.	integer	Yes	No
iregion	Región	char(1)	Yes	No

Tabla: Conebienes

Lista de Columnas

Nombre	Descripción	Tipo	Llave Primaria	Campo Nulo
ci1601l	Cantidad de bienes en 1601 listos para la venta	double	No	Yes
ci1601nl	Importe de bienes en 1601 no listos para la venta	double	No	Yes
ci1602l	Importe de bienes en 1602 listos para la venta	double	No	Yes
ci1602nl	Cantidad de bienes en 1602 no listos para la venta	double	No	Yes
ci1603	Cantidad de bienes en 1603	double	No	Yes
cn1601l	Importe de bienes en 1603	integer	No	Yes
cn1601nl	Cantidad de bienes en 1601 no listos para la venta	integer	No	Yes
cn1602l	Cantidad de bienes en 1602 listos para la venta	integer	No	Yes
cn1602nl	Cantidad de bienes en 1602 no listos para la venta	integer	No	Yes
cn1603	Cantidad de bienes en 1603 listos para la venta	integer	No	Yes
imes	Mes en que efectúa un corte de operaciones.	date	Yes	No
iregion	Región en que se clasificó	char(1)	Yes	No
isucursal	Sucursal a la que pertenece	char(3)	Yes	No

Tabla: Direcciones

Lista de Columnas

Nombre	Descripcion	Tipo	Llave Primaria	Campo Nulo
calle	Calle de ubicación	char(25)	No	Yes
ciudad	Ciudad de ubicación	char(25)	No	Yes
codpos	Código Postal	char(5)	No	No
colonia	Colonia	char(25)	No	No
estado	Estado	char(25)	No	Yes
fax	Fax	char(12)	No	No
icuenta	Cuenta 1601, 1602, 1603	char(4)	No	Yes
ifolio	Folio identificador	numeric(3)	No	Yes
iregion	Región en que se clasificó	char(1)	No	Yes
isec	Folio identificador	char(5)	No	Yes
responsable	Nombre del responsable	char(40)	No	No
teléfono	Teléfono	char(12)	No	No

Tabla: Bcgastos

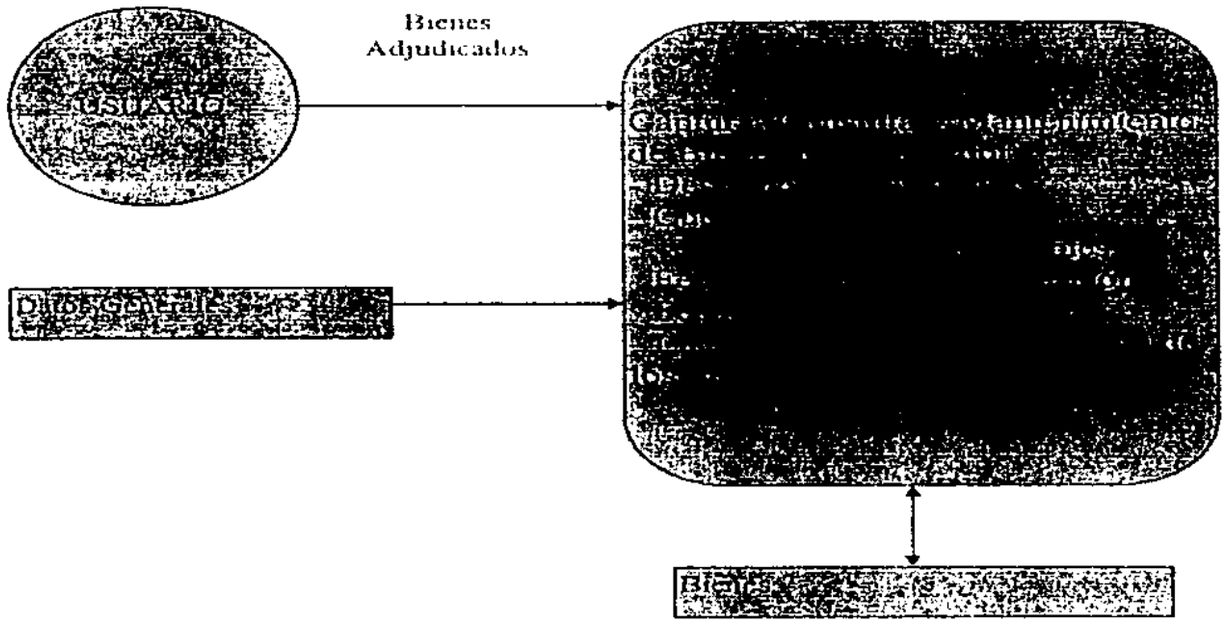
Lista de Columnas

Nombre	Descripción	Tipo	Llave primaria	Campo Nulo
cimporte	Importe del gasto efectuado	numeric(12,2)	No	Yes
econcepto	Concepto aplicado	char(2)	No	No
edescorta	Descripción corta	char(100)	No	No
edescripcion	Descripción larga	char(50)	No	No
edeudor	Nombre del deudor	char(60)	No	No
fecha	Fecha de aplicación del gasto	date	No	No
icuenta	Cuenta 1601, 1602 o 1603	char(4)	Yes	No
ifolio	Folio asociado al bien	numeric(5)	Yes	No
iregion	Región en que se clasificó	char(1)	Yes	No
isec_bien	Identificador del bien	numeric(5)	Yes	No
isec_gasto	Identificador del gasto	numeric(5)	Yes	No
isucursal	Sucursal a la que pertenece	char(3)	No	No

4.7 Diseño de Procesos.

Proceso: Consulta, Mantenimiento y Venta De Bienes

Bienes Adjudicados y en Dación en Pago



Objetivo: Proporcionar una función que permita la captura, consulta y actualización de Bienes para un Deudor seleccionado. Además del control y el proceso de venta de los mismos.

Especificaciones

Entradas:
Param: - Entidades de Parámetros y Conceptos (Cuentas, Clasificación de Bienes, etc.)

Entradas y salidas:
Bienes: - Tabla de Bienes

Eventos:

Las siguientes funciones están orientadas al ingreso, consulta y mantenimiento de los Bienes de un Deudor:

1.- Nuevo.
Permite ingresar un nuevo Bien al Sistema.

2.- Borrar
Permite borrar físicamente la información de un Bien.

3.- Salvar
Actualiza la información en la base de datos para Altas y Cambios

4.- Cerrar.
Permite salir del módulo de Bienes.

Venta de los bienes

Una vez que los Bienes de un Deudor han sido ingresados al Sistema, éstos pueden venderse mediante el **Menú de Acciones** del Sistema.

Para que un Bien pueda venderse, necesita estar **listo para la venta**. Un Bien está listo para la venta cuando:
- Se tiene posesión del mismo
- Se tiene escrituras (bienes inmuebles) o factura (bienes muebles)
- Sin algún tipo de trámite jurídico.

Existen los siguientes Tipos de Venta:

1.- Venta Total.
Permite vender totalmente un Bien. Se aplica a Bienes Muebles e Inmuebles.

2.- Venta sin Pago Total.
Permite vender un Bien sin liquidarlo totalmente, es decir, mediante un anticipo. Se aplica solamente a Bienes Inmuebles.

3.- Liquidación.
Permite la liquidación del remanente, resultado de una Venta sin Pago Total.

4.- Salida.
Permite la salida de un Bien sin venderlo. Se aplica a Bienes Muebles e Inmuebles.

5.- Prepara Venta Parcial.
Permite vender parcialmente un Bien. Separa del Bien, la parte proporcional que va a ser vendida, para que posteriormente pase al proceso de Venta Total.

6.- Ficha de Castigo.
Es necesario aplicar este proceso, a los Bienes cuyo importe de venta es menor a su valor de adjudicación.

Resumen:

Una vez que los Bienes de un Deudor han sido ingresados al Sistema, éstos pueden venderse mediante el **Menú de Acciones** del Sistema, siempre y cuando, **estén listos para la venta.**

Cuando un Bien ingresa al Sistema, se especifica la cuenta a la que pertenece (1601 para bienes muebles y 1602 para bienes inmuebles), el estatus ('A') y el importe de adjudicación.

Todo movimiento que se efectúe, actualiza el estatus del Bien, la fecha de movimiento, el importe del movimiento (importe de la venta o anticipo).

Las ventas totales para bienes muebles o inmuebles, concluyen con estatus igual a 'T'. En caso de que el importe de venta sea menor al importe de adjudicación, pasan por un estatus intermedio 'F'.

Las ventas parciales, sólo se permiten a bienes muebles. Esto provoca la generación de un nuevo registro (1601) para la parte proporcional del importe de adjudicación que se vende, y le resta dicho importe al importe de adjudicación del registro original. El nuevo registro 1601 tendrá un estatus 'D' y solo podrá venderse totalmente. El registro original podrá venderse totalmente o parcialmente de nuevo.

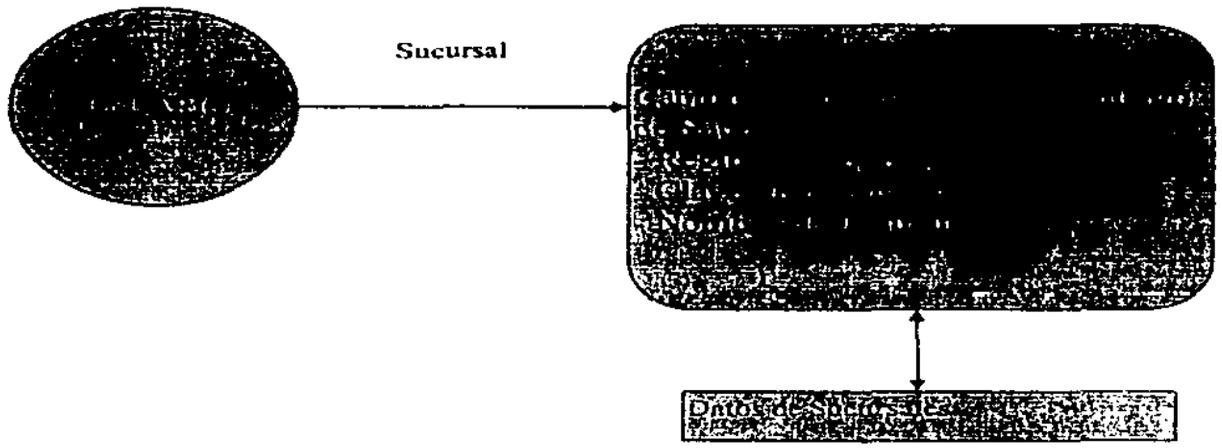
Las ventas parciales, concluyen con estatus igual a 'P'. En caso de que el importe de venta sea menor al importe de adjudicación, pasan por un estatus intermedio 'E'.

Las ventas sin pago total (anticipo), sólo se permiten a bienes inmuebles. Esto provoca la generación de un nuevo registro (1603) con importe de adjudicación igual al remanente pendiente de pago. El registro original cambia a estatus 'V'. El nuevo registro 1603 tendrá un estatus 'A' y solo podrá liquidarse totalmente.

Las liquidaciones, concluyen con estatus igual a 'L'. En caso de que el importe de venta sea menor al importe de adjudicación, pasan por un estatus intermedio 'F'. Por otro lado, la liquidación cambia el estatus del registro original (1602) a 'L' y actualiza su importe de venta sumándole el importe de liquidación.

En esta aplicación, sólo aparecen en consulta los registros de Bienes con estatus 'A', 'F' y 'E'.

Proceso: Consulta, Mantenimiento De Sucursales



Objetivo: Proporcionar una función que permita la captura, consulta y actualización de las sucursales del Banco.

Especificaciones

Entradas Y Salidas:
PARAM: - Tabla de Parámetros y Conceptos (Regiones, Sucursales)

Eventos:

Primero se selecciona la Región, y se despliegan en una ventana, todas las sucursales que la integran.

Una vez desplegadas las sucursales de la Región seleccionada, se tienen los siguientes eventos:

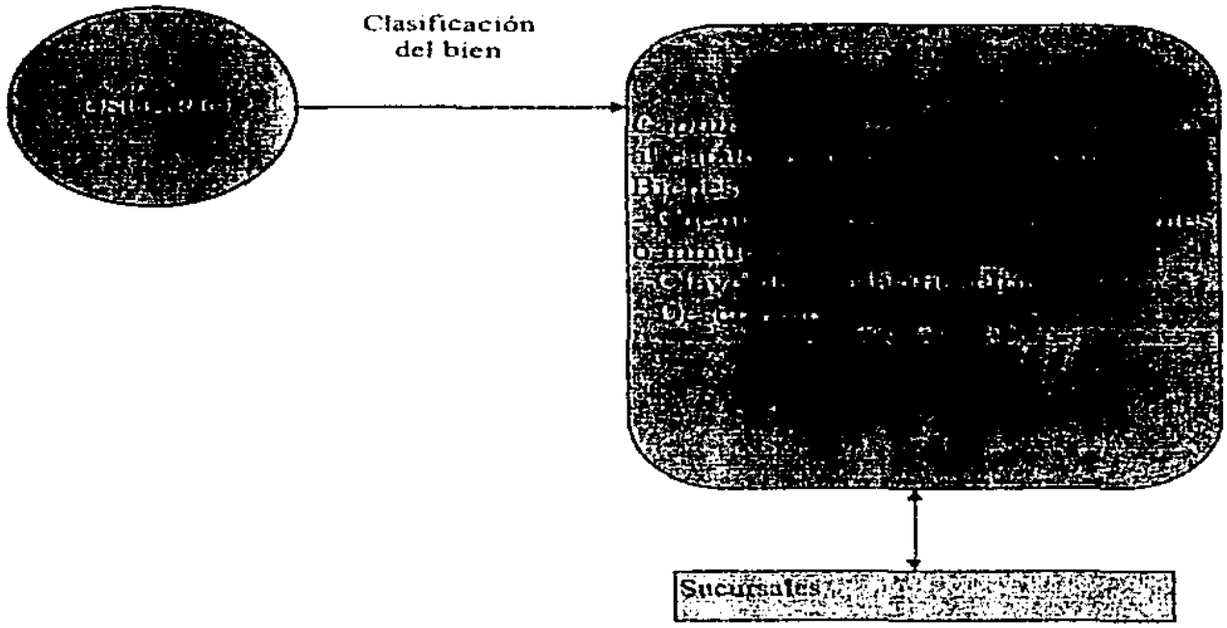
1.- Nuevo
Se abre una segunda ventana para ingresar los datos de una nueva Sucursal.

2.- Cambio
Presenta en una segunda ventana la información de la sucursal seleccionada, la cual puede ser modificada.

3.- Borrar
Borra fisicamente la información de una sucursal.

4.- Cerrar.
Permite salir de la ventana de sucursales.

Proceso: Consulta, Mantenimiento De Clasificación De Bienes



Objetivo: Proporcionar una función que permita la captura, consulta y actualización del catálogo de clasificación de Bienes, tanto para muebles como inmuebles.

Especificaciones

Entradas Y Salidas:
PARAM: - Tabla de Parámetros y Conceptos (Cuentas, Clasificación de Bienes)

Eventos:

Primero se selecciona la Cuenta (muebles o inmuebles), y se despliegan en una ventana, el catálogo de tipos de Bienes que la integran. Como ejemplo, se tiene que los tipos de Bienes Muebles son vehículos, equipo de cómputo, etc., y los tipos de Bienes Inmuebles son Casas, terrenos, etc.

Una vez desplegados los distintos tipos de Bienes de la Cuenta (Muebles o Inmuebles) seleccionada, se tienen los siguientes eventos:

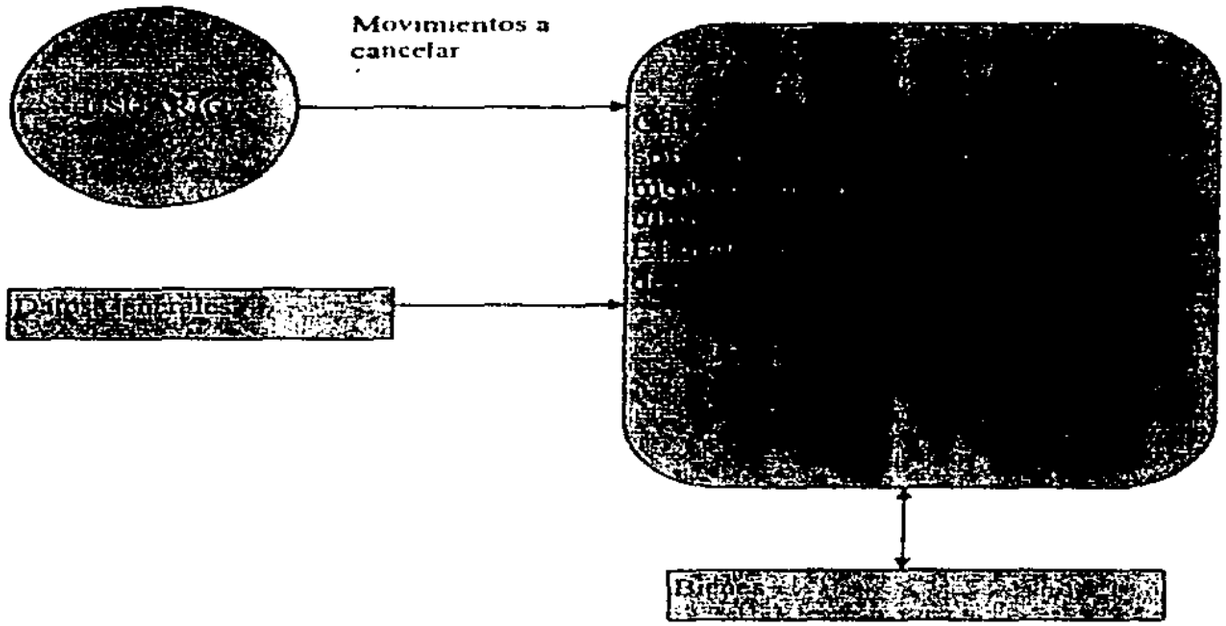
1.- Nuevo.
Se abre una segunda ventana para ingresar los datos de un nuevo tipo de Bienes.

2.- Cambio
Presenta en una segunda ventana, la información del Tipo de Bienes seleccionado, la cual puede ser modificada.

3.- Borrar
Borra fisicamente la información de un Tipo de Bienes.

4.- Cerrar.
Permite salir de la ventana de Bienes.

Proceso: Cancelación De Movimientos



Objetivo: Proporcionar una función que permita la cancelación de movimientos efectuados a los Bienes, con fecha de movimiento posterior a la fecha del último cierre mensual, es decir, sólo se podrán cancelar movimientos que no han sido considerados en algún cierre mensual. Por movimiento se entiende Venta Total, Venta Parcial, Venta sin Pago Total, Liquidación, Salida.

Especificaciones

Entradas:

PARAM: - Tabla de parámetros y conceptos.

Entradas Y Salidas:

BIENES: - Tabla de Bienes.

Eventos:

Primero se deben seleccionar la Región y la Sucursal. Esta función sólo presentará Deudores y Bienes de la Región y Sucursal seleccionadas.

Para esta función, el Sistema presenta dos ventanas, en la primera se despliegan todos los deudores con movimientos factibles de ser cancelados. En la segunda, se presentan los Bienes del Deudor seleccionado en la primera, factibles de cancelación

Una vez desplegada la información solicitada, se tienen los siguientes eventos:

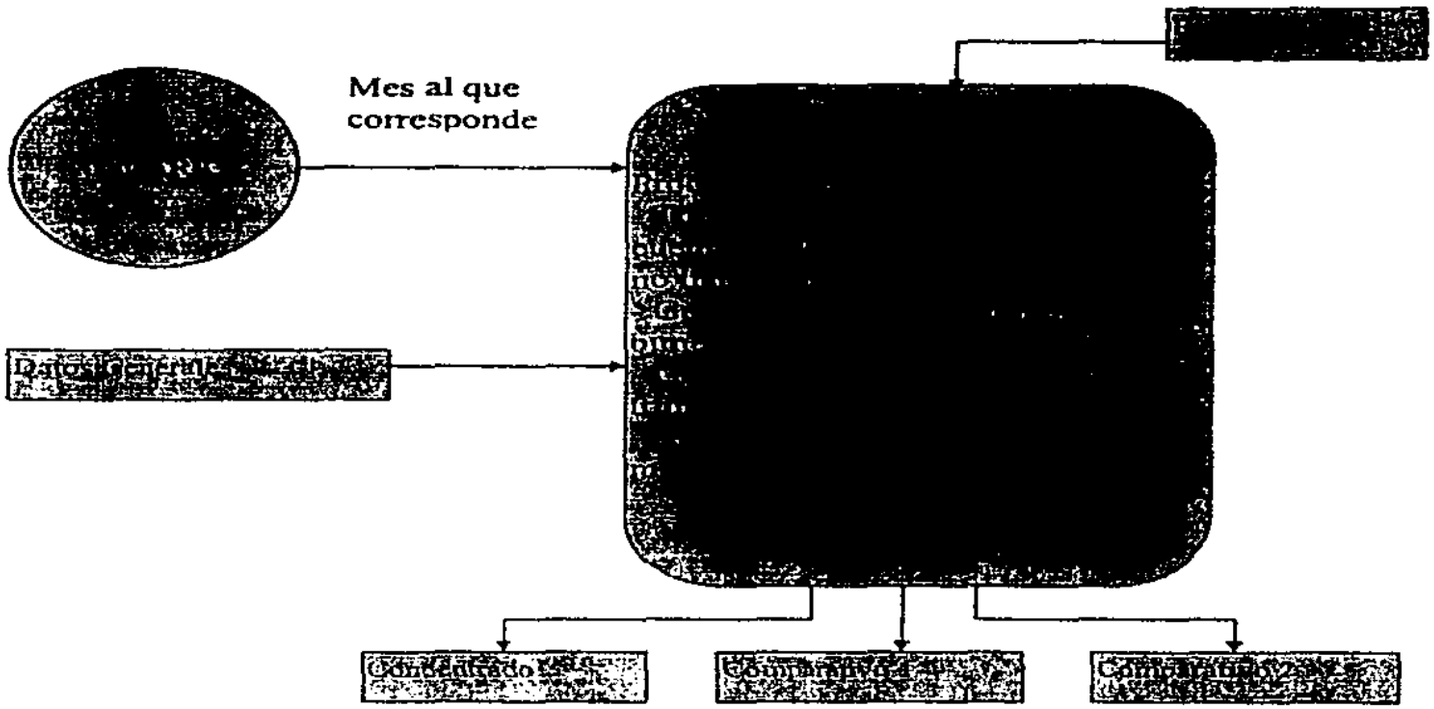
1.- Restaurar.
Cancela el movimiento para el Bien seleccionado.

2.- Cerrar.
Permite salir de esta función.

NOTAS:

Cancelar un movimiento, significa anular la venta, salida, etc. del bien, dejándolo como estaba originalmente. Significa borrar la información de los campos Fecha de Movimiento e Importe de Venta y dejar su Estatus en 'A' de Alta. En el caso de venta sin pago total, implica además eliminar el registro generado con el remanente de la operación (registro 1603).

Proceso: Cierre Mensual



Objetivo: Proporcionar una función que efectúe un cierre mes a mes. Este cierre consiste en registrar en un archivo de la base de datos la situación que guardan los bienes al fin de cada mes. Se genera un registro por sucursal y contempla los siguientes conceptos:

- Número de Bienes e Importe de adjudicación para Bienes Muebles (1601) listos y no listos para la venta.
- Número de Bienes e Importe de adjudicación para Bienes Inmuebles (1602) listos y no listos para la venta.
- Número de Bienes e Importe de adjudicación para Bienes Prometidos en Venta (1603).

Esto con la intención de contar con un historial y poder efectuar comparativos mensuales y estudios de comportamientos.

Especificaciones

Entradas:

BIENES: - Tabla de parámetros y conceptos.

Salidas:

CONCENTRADO:	- Concentrados Mensuales
COMPARATIVO1:	- Comparativo Bimestral.
COMPARATIVO2:	- Comparativo Trimestral.

Eventos:

Para efectuar el cierre se necesita indicar el mes al que corresponde. En caso de que el cierre de dicho mes ya se haya efectuado, el sistema lo indicará y solicitará confirmación para efectuar el proceso del cierre mensual.

El principal objetivo del cierre mensual es generar un concentrado a fin de mes de la situación que guardan los Bienes que no se han vendido (cuenta 1601 y 1602 con estatus = 'A') o que no se han liquidado totalmente (cuenta 1603 con estatus = 'A'), generándose un registro por sucursal y para cada sucursal los siguientes conceptos:

- Número de Bienes e Importe de adjudicación para Bienes Muebles (1601) listos y no listos para la venta.
- Número de Bienes e Importe de adjudicación para Bienes Inmuebles (1602) listos y no listos para la venta.
- Número de Bienes e Importe de adjudicación para Bienes Prometidos en Venta (1603).

Una vez generada la información anterior, se llevan a cabo dos procesos con la intención de generar la información en contenido y forma de dos reportes:

- Comparativo bimestral (mes actual y anterior) por región, sucursal y cuentas.
- Comparativo trimestral (mes actual y dos anteriores) por región, cuenta, listo y no listo para la venta.

Además registra la fecha a la que corresponde el proceso mensual. Esta se utiliza en los procesos de Depuración y Cancelación.

Proceso: Reporte Del Cuadernillo

Objetivo: Generar un reporte con la información más importante de cada uno de los Bienes que aún no han salido, clasificando la información por Región, Sucursal y Folio. Un folio (deudor) puede tener más de un Bien y son identificados por un consecutivo de Bienes por Deudor.

Especificaciones

Entradas:

DEUDORES:	- Tabla de Deudores.
BIENES:	- Tabla de Bienes.
PARAM:	- Tabla de Conceptos y Catálogos.

Salidas (columnas del reporte):

Para cada Deudor (en un renglón):

- Región
- Sucursal
- Nombre del Deudor
- Folio
- Dación o Adjudicación
- Importe del Adeudo

Para cada uno de los Bienes del Deudor (renglones siguientes):

- Cuenta e Importe de Adjudicación
- Descripción larga
- Fecha de Adjudicación
- Fecha de Avalúo
- Importe de Avalúo
- Listo o No listo para la venta
- Observaciones

Proceso (registros seleccionados):

- estatus in ('A', 'E', 'F') Todos los registros de las cuenta 1601, 1602, 1603 que no han salido.

Clasificación :

- Región, Sucursal, Folio, Consecutivo

Cortes (subtotales y totales):

- Sucursal

Proceso: Reporte Del Cuadernillo Por Sucursal

Objetivo: Generar un reporte con los totales por sucursal de cada una de las cuentas (1601, 1602, 1603), con lo Bienes que aún no han salido, clasificando la información por Región y Sucursal. Este reporte se puede interpretar como un resumen del cuadernillo y debe cuadrar con éste.

Especificaciones

Entradas:

BIENES:	- Tabla de Bienes.
PARAM:	- Tabla de Conceptos y Catálogos.

Salidas (columnas del reporte):

Un registro por cada Región-Sucursal-Cuenta

- Región
- Sucursal
- Cuenta
- Suma de importes de Adjudicación por cuenta para la Región-Sucursal

Proceso (registros seleccionados):

- estatus in ('A', 'E', 'F') que no han salido. — Todos los registros de las cuenta 1601, 1602, 1603

Clasificación :

- Región, Sucursal, Cuenta

Cortes (subtotales o totales):

- Cuenta

Proceso: Reporte De Relación De Entradas Del Mes

Objetivo: Generar un reporte con las entradas registradas durante el mes, es decir, todos los Bienes ingresados listos o no listos para la venta.

Especificaciones

Entradas:

BIENES:	- Tabla de Bienes.
PARAM:	- Tabla de Conceptos y Catálogos.

Salidas (columnas del reporte):

Un registro por cada Bien que haya entrado en el mes

- Región
- Sucursal
- Número de Bienes (1)
- Importe de Adjudicación
- Descripción corta del Bien.

Proceso (registros seleccionados):

- cuenta in ('1601', '1602')
- estatus in ('A', 'T', 'S','V','L','F') — No se incluyen registros derivados de ventas parciales
- fecregistro (en el mes de cierre)

Clasificación :

- Región, Sucursal

Cortes (subtotales o totales):

- Gran Total (Número de Bienes e Importe)

Proceso: Reporte De Relación De Salidas Del Mes

Objetivo: Generar un reporte con las salidas registradas durante el mes, es decir, todos los Bienes que se vendieron o salieron.

Especificaciones

Entradas:

BIENES: - Tabla de Bienes.
PARAM: - Tabla de Conceptos y Catálogos.

Salidas (columnas del reporte):

Un registro por cada Bien que haya salido en el mes

- Región
- Sucursal
- Número de Bienes (1: ventas totales, liquidaciones, salidas)
  (0: ventas parciales, venta sin pago total)
- Importe (Importe de Venta: venta sin pago total)
  (Valor Adjudicación: Demás casos)
- Descripción corta del Bien.

Proceso (registros seleccionados):

- cuenta in ('1601', '1602')
- estatus in ('T', 'P','S','L','V')
- Not (cuenta = 1602 y estatus = 'L') Liquidación del registro 1602
- fecmov (en el mes de cierre)

Clasificación :

- Región, Sucursal, Folio

Cortes (subtotales o totales):

- Gran Total. (Número de Bienes e Importe).

Proceso: Reporte De Relación De Entradas Del Mes Para La CNB

Objetivo: Generar un reporte con las entradas registradas durante el mes, es decir, todos los Bienes ingresados listos o no listos para la venta, de acuerdo a los requerimientos de la CNB.

Especificaciones

Entradas:

DEUDORES:	- Tabla de Deudores.
BIENES:	- Tabla de Bienes.
PARAM:	- Tabla de Conceptos y Catálogos.

Salidas (columnas del reporte):

Un registro por cada Bien que haya entrado en el mes

- Cuenta
- Nombre del Deudor
- Descripción corta del Bien
- Sucursal
- Valor de Adjudicación

Proceso (registros seleccionados):

- cuenta in ('1601', '1602')
- estatus in ('A', 'T', 'S','V','L','F') — No se incluyen registros derivados de ventas parciales
- fecregistro (en el mes de cierre)

Clasificación :

- Cuenta

Cortes (subtotales o totales):

- Cuenta

Proceso: Reporte De Relación De Salidas Del Mes Para La CNB

Objetivo: Generar un reporte con las salidas registradas durante el mes, es decir, todos los Bienes que se vendieron o salieron, de acuerdo a los requerimientos de información de la CNB.

Especificaciones

Entradas:

DEUDORES:	- Tabla de Deudores
BIENES:	- Tabla de Bienes.
PARAM:	- Tabla de Conceptos y Catálogos.

Salidas (columnas del reporte):

Un registro por cada Bien que haya salido en el mes

- Cuenta
- Nombre del Deudor
- Descripción corta del Bien
- Sucursal
- Importe de Adjudicación (Importe de Registro: ventas totales, salidas, liquidaciones) (0: ventas parciales, venta sin pago total)
- Importe de Venta
- Utilidad/Pérdida (Imp Venta - Imp Reg: ventas totales, salidas, liquidaciones) (0: ventas parciales, venta sin pago total)

Proceso (registros seleccionados):

- cuenta in ('1601', '1602')
- estatus in ('T', 'P','S','L','V')
- fecmov (en el mes de cierre)

Clasificación :

- Cuenta

Cortes (subtotales ototales):

- Cuenta

Proceso: Reporte De Relación De Bienes

Objetivo: Generar reportes de Bienes, de tal forma que al momento de solicitar el reporte, permita especificar ciertos criterios dependiendo de las necesidades de información. Es decir, que sea posible restringir la información por los siguientes parámetros:

- Región
- Sucursal
- Listo/No listo
- Cuenta (Muebles o Inmuebles)
- Clasificación por tipo de cuenta
- Importe de Adjudicación

Especificaciones

Entradas:

BIENES: - Tabla de Bienes.
PARAM: - Tabla de Conceptos y Catálogos.

Salidas (columnas del reporte):

Un registro por cada Bien seleccionado

- Región
- Sucursal
- Fecha de Adjudicación
- Descripción larga del Bien
- Asegurado/ No asegurado
- Valor de Registro (o importe de adjudicación)
- Valor de Avalúo
- Fecha de Avalúo
- Listo/No listo para la venta

Proceso (registros seleccionados):

- cuenta in ('1601', '1602')
- estatus = 'A'

Clasificación :

- Región, Sucursal, Fecha de Adjudicación

Cortes (subtotales o totales):

- Gran Total (Número de Bienes e Importe de Adjudicación)

Proceso: Resumen De Bienes Muebles Por Su Clasificación

Objetivo: Generar un resumen de Bienes Muebles de acuerdo a su clasificación (automóviles, camiones, joyas, obras de arte, etc.), separándolos en Listos y no Listos para su venta.

Especificaciones

Entradas:

BIENES: - Tabla de Bienes.
PARAM: - Tabla de Conceptos y Catálogos.

Salidas (columnas del reporte):

Un registro por cada clasificación de Bienes
- Descripción de la clasificación de Bienes
- Número de Bienes de la misma clasificación
- Suma de los importes de adjudicación

Proceso (registros seleccionados):

- cuenta = '1601'
- estatus = 'A'

Clasificación :

- Listo/No Listo para la venta, clasificación del Bien

Cortes (subtotales o totales):

- Listo/No Listo para la venta (Número de Bienes e Importe de Adjudicación)

Proceso: Resumen De Bienes Inmuebles Por Su Clasificación

Objetivo: Generar un resumen de Bienes Inmuebles de acuerdo a su clasificación (casas habitación, terrenos, edificios, oficinas, etc.), separándolos en Listos y no Listos para su venta.

Especificaciones

Entradas:

BIENES:	- Tabla de Bienes.
PARAM:	- Tabla de Conceptos y Catálogos.

Salidas (columnas del reporte):

Un registro por cada clasificación de Bienes

- Descripción de la clasificación de Bienes
- Número de Bienes de la misma clasificación
- Suma de los importes de adjudicación

Proceso (registros seleccionados):

- cuenta = '1602'
- estatus = 'A'

Clasificación :

- Listo/No Listo para la venta, clasificación del Bien

Cortes (subtotales o totales):

- Listo/No Listo para la venta (Número de Bienes e Importe de Adjudicación)

Proceso: Sumaria De Las Cuentas (Comparativo Bimestral)

Objetivo: Generar un comparativo entre el mes de cierre y el mes anterior de la Sumaria de las cuentas (1601, 1602 y 1603) por Región y Sucursal.

Especificaciones

Entradas:

COMPARATIVO: - Comparativo Bimestral.

Salidas (columnas del reporte):

Un registro por cada Sucursal.

- Región
- Sucursal
- Importe Cuenta 1601, mes anterior al mes de cierre
- Importe Cuenta 1602, mes anterior al mes de cierre
- Importe Cuenta 1603, mes anterior al mes de cierre
- Total de los tres conceptos anteriores
- Importe Cuenta 1601, mes de cierre
- Importe Cuenta 1602, mes de cierre
- Importe Cuenta 1603, mes de cierre
- Total de los tres conceptos anteriores

Proceso (registros seleccionados):

- Todos

Clasificación :

- Región, Sucursal

Proceso: Comparativo Mensual (Informe 3 Meses)

Objetivo: Generar un comparativo entre el mes de cierre y dos meses anteriores de la Sumaria de las cuentas (1601, 1602 y 1603) subdivididas en Listos y no listos para la venta. Este comparativo es sólo por Región. Para la cuenta 1601 listos para la venta se generará un registro por región, y así mismo para la 1602 listos para la venta, 1601 No listos para la venta, 1602 No listos para la venta, y 1603 (Inmuebles prometidos en venta).

Especificaciones

Entradas:

COMPARATIVO2: - Comparativo Trimestral.

SALIDAS (Columnas Del Reporte):

Un registro por cada Tipo de Registro y Región

Tipo de Registro (1: 1601 Listos para la venta, 2: 1602 Listos para la venta, 3: 1601 No listos para la venta, 4: 1602 No listos para la venta, 5: 1603)

- Región

- Num. de Bienes, dos meses anterior al mes de cierre
- Importe, dos meses anterior al mes de cierre
- Porcentaje del Total (Importe), dos meses anterior al mes de cierre

- Num. de Bienes, mes anterior al mes de cierre
- Importe, mes anterior al mes de cierre
- Porcentaje del Total (Importe), mes anterior al mes de cierre

- Num. de Bienes, mes de cierre
- Importe, mes de cierre
- Porcentaje del Total (Importe), mes de cierre

Proceso (registros seleccionados):

- Todos

Clasificación :

- Tipo de registro (1: 1601 Listos para la venta, 2: 1602 Listos para la venta, 3: 1601 No listos para la venta, 4: 1602 No listos para la venta, 5: 1603), Región.

NOTA: Para cada región existe cada uno de los tipos de registro.

5. CONSTRUCCIÓN DEL SISTEMA

5.1.Introducción.

Uno de los principales objetivos en la elaboración del prototipo consiste en la definición de la interfaz del usuario, ya que representa el medio por el cual se obtiene o se introduce información al sistema. Para lograr un buen diseño deben satisfacerse los siguientes puntos:

- ***Eficacia***, al lograr mediante el diseño de interfaces que el usuario tenga acceso al sistema, de tal forma que satisfaga sus necesidades particulares.
- ***Eficiencia***, demostrada a través de interfaces que mejoren la velocidad de captura de los datos y reduzcan los errores.
- ***Consideración del usuario***, al demostrar un diseño adecuado de la interfaz que favorezca la retroalimentación del sistema para los usuarios.

La interfaz cuenta con dos componentes principales: el lenguaje de presentación, que es parte de la relación computadora-hombre y el lenguaje de acción que caracteriza la parte hombre-computadora; en conjunto, ambos conceptos cubren la forma del término interfaz del usuario. Existen diferentes tipos de interfaces de usuario, las cuales se listan a continuación:

- ***Interfaz de preguntas y respuestas***. Aquí el usuario proporciona una respuesta y la computadora responderá con base a tal información de entrada de una manera preprogramada.
- ***Menús***. Permite que el usuario elija las posibles opciones de una lista en pantalla; al responder al menú, el usuario no necesita conocer el sistema pero sí necesita saber qué tareas pueden realizarse. Puede tener acceso a los menús a través del teclado, del ratón, o a través de cualquier otro dispositivo de entrada.
- ***Interfaz de entrada/salida***. Son formas en pantalla que despliegan campos que requieren información proporcionada por el usuario para procesar dicha instrucción y arrojar una serie de resultados en la misma pantalla.
- ***Interfaz de manejo directo***. Le confiere al usuario en forma gráfica una metáfora de la aplicación, permitiendo un manejo directo de la representación gráfica en la pantalla. Este concepto se apoya principalmente en la representación de imágenes (iconos) que permitan visualizar los elementos que interviene en el sistema , así como las relaciones que existen entre sí.

Al evaluar las interfaces deben tenerse en cuenta ciertas normas:

- La capacitación de los usuarios debe ser aceptablemente breve.
- Los usuarios deben ser capaces de hacer uso de comandos sin tener que pensar explícitamente en ellos o sin tener que hacer referencia a un menú de ayuda o manual.
- La interfaz debe ser consistente de tal forma que se reduzcan los errores y aquellos que se presenten no se deban a un diseño deficiente.
- El tiempo necesario de adaptación al sistema debe ser breve para que los usuarios superen los errores.
- Los usuarios esporádicos deben tener capacidad para comprender con rapidez el sistema

Un aspecto importante que deben incluir las interfaces de usuario ya mencionadas anteriormente, consiste en la retroalimentación al usuario con el fin de supervisar y modificar la conducta en el manejo del sistema. Cuando los usuarios interactúan con las computadoras necesitan saber la manera como progresa una petición hecha o qué información adicional necesita.

En términos generales, la retroalimentación sirve para que el usuario esté enterado de los siguientes puntos:

- La computadora aceptó la entrada.
- La entrada se encuentra en la forma correcta.
- La entrada no se encuentra en la forma correcta.
- Habrá un retraso en el procesamiento.
- La solicitud ha sido concluida.
- La computadora es incapaz de concluir la petición.
- Se dispone de mayor detalle en la retroalimentación y se indica la manera de obtenerla.

5.2.Entorno de desarrollo.

Los puntos anteriormente expuestos nos ayudan a reconocer la importancia y el cuidado que deben darse en esta etapa de la construcción del sistema.

Debido a que la arquitectura cliente-servidor se apoya mucho en una Interfaz Gráfica del Usuario (GUI) en la parte del cliente, la plataforma ideal para desarrollar aplicaciones comerciales y que se ajusta a todos los puntos anteriormente expuestos, consiste de *Microsoft Windows* en sus versiones de *Windows 3.x* o *Windows 95*. Por otro lado, la herramienta de desarrollo para construir dicha interfaz consiste de *PowerBuilder*. A continuación se expondrán algunos conceptos y puntos sobresalientes de dichos productos que nos servirán como soporte para la etapa de construcción de la aplicación.

5.2.1.Conceptos Generales acerca de Windows.

Interfaz Gráfica del Usuario (GUI). Usa gráficos y símbolos también llamados iconos basados en una metáfora de escritorio sobre el cual se tienen organizados todos los objetos que se utilizan comunmente en una oficina, representados en toda la pantalla y que pueden manipularse a través del uso del ratón o del teclado.

Acceso Común del Usuario (CUA). Es un conjunto de reglas publicadas, que reflejan un estándar común acerca de cómo construir aplicaciones GUI, sugiriendo una metodología para la creación de dichas interfaces. Siguiendo dichos estándares podremos construir una aplicación que sea muy similar al resto de aplicaciones *Windows*, reduciendo así la curva de aprendizaje por parte del usuario final.

Control. Se refiere a un objeto gráfico, el cual puede ser una ventana, un campo de edición, una caja de listado (list box) , una caja de opciones (check box), un icono, o un menú.

Ventanas y Controles.

La *Figura 2.1* nos muestra algunos de los componentes más importantes de una ventana de *Windows*.

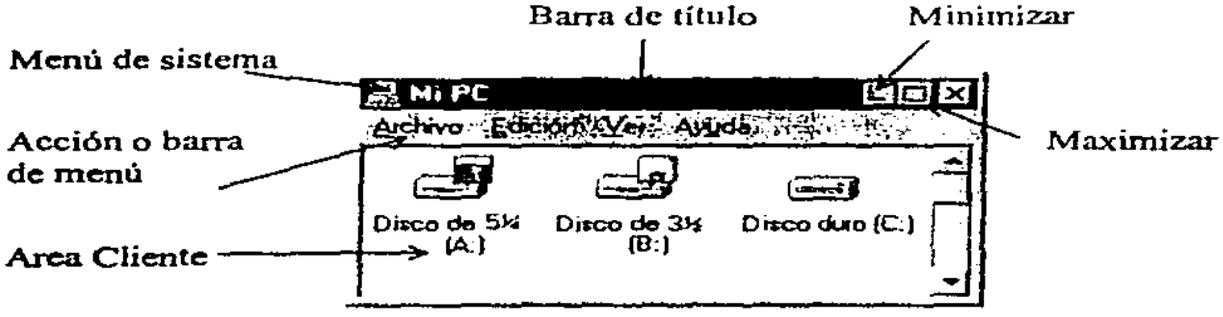


Fig 2.1 Componentes de una ventana

Ventana. El control primario de toda aplicación GUI. Los componentes principales son:

- Menú del sistema. Es una pequeña caja en la parte superior que permite acceder las funciones principales aplicables a la ventana. Funciones como restaurar, mover, maximizar y minimizar se encuentran aquí.
- Botones de maximizar/minimizar. Son dos botones en la parte superior izquierda que permite modificar el tamaño de la ventana.
- Barra de título. Es el área entre el menú del sistema y los botones para minimizar/ maximizar.
- Barra de menú. Está localizada bajo la barra de título y contiene accesos a ciertos procesos. Las opciones desplegadas son estándares de CUA (*Acceso Común del Usuario*), entre los cuales se encuentran Archivo, Edición, Ayuda.

Controles. Son objetos cuya misión es recolectar o desplegar datos, entre los cuales se pueden mencionar los siguientes:

Static Text
Single Line Edit
Check Box
Radio Button
Command Button
Group Box
MultiLine Edit
DropDown list
List Box
Horizontal Scroll Bar
Vertical Scroll Bar

5.2.2. Conceptos Generales acerca de PowerBuilder

La interfaz de *PowerBuilder (figura 2.2)* consiste en una serie de *painters* GUI para trabajar con objetos tales como bases de datos, aplicaciones, menús y ventanas. Este ambiente modular permite la explotación de las características de programación orientada a objetos *(OOP - Object Oriented Programming)* sobre las cuales descansa *PowerBuilder*.

El *painter* principal contiene una barra de herramientas que permite crear objetos comunes de *Windows*, como los ya mencionados en el apartado anterior. Cuenta además con un objeto propio de *PowerBuilder* como es el *DataWindow*, con el cual se crean gráficamente ventanas que reflejan la estructura de la base de datos. Éstas contienen funciones interconstruidas para permitir el acceso y actualización de las bases de datos, así como también para interactuar con el usuario. Esta es una característica que diferencia a *PowerBuilder* de otros productos de desarrollo, que no han sido diseñados específicamente para efectuar tareas de desarrollo de bases de datos, como es el caso de *Microsoft Visual Basic*.

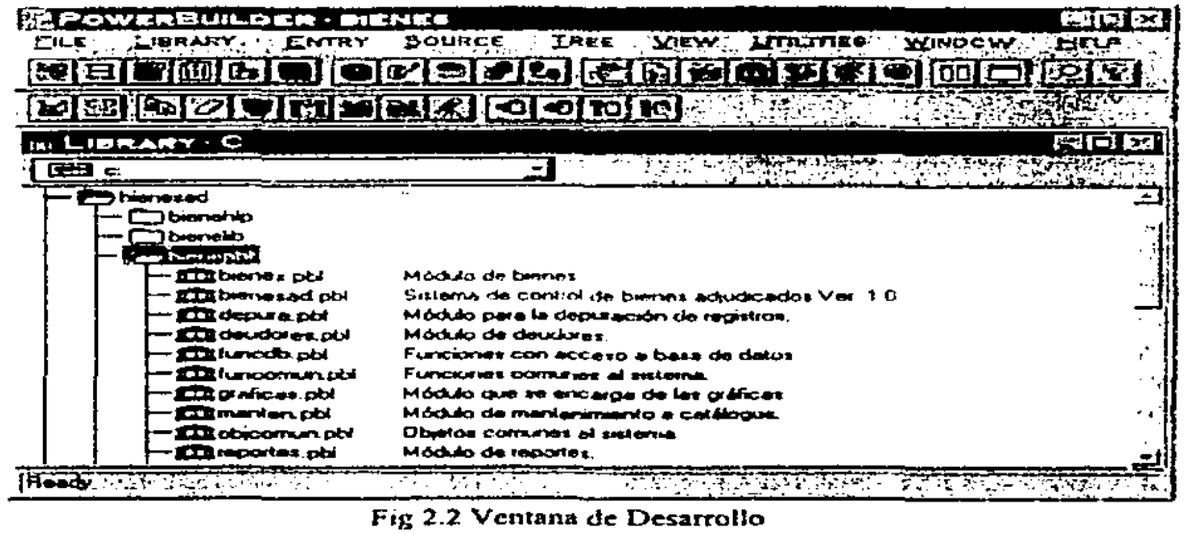


Fig 2.2 Ventana de Desarrollo

Debido a su característica de orientación a objetos, permite la creación de bibliotecas de objetos, que pueden ser reutilizados en aplicaciones futuras. En la actualidad hay empresas que han logrado desarrollar nuevas aplicaciones reutilizando objetos hasta en un 50% teniendo que escribir el resto de código para las nuevas aplicaciones.

Para poder lograr la reutilización de estos objetos, es necesario contar con controladores de versiones de los objetos que se utilizan y administradores de configuraciones. Estos ayudan a determinar las versiones de los objetos que se utilizan para configurar las versiones de las aplicaciones.

La reutilización de objetos se basa en características de herencia, polimorfismo, y encapsulación, cuyas descripciones se dan a continuación:

- **Herencia.** Significa que un objeto contiene todos los atributos, variables, funciones, eventos, estructuras y programas definidos previamente por su ancestro, además de poseer características que son específicas al descendiente.

- **Polimorfismo.** Significa que el objeto asume varias formas. Esto puede implementarse de dos maneras:
  a) en el modo que los objetos se comunican entre ellos
  b) en la habilidad de sobreponer sus funciones

- **Encapsulación.** Es el atributo por el cual el objeto puede ser de acceso múltiple, mientras conserva para sí y los descendientes todas sus especificaciones propias.

Debido a que es importante manejar un estándar en cuanto a la definición de las variables, objetos y funciones que intervienen en todo el proyecto de desarrollo, *PowerBuilder* propone algunos estándares que es conveniente seguir. Esto permite tener un mejor control sobre el código fuente, así como el subsecuente hecho de permitir el mantenimiento del sistema y su fácil integración a otros proyectos de software.

En el siguiente cuadro se enuncian algunas características del uso de variables y definición de nombres de objetos en el ambiente de desarrollo de *PowerBuilder:*

Objetos

Tipo	Prefijo	Alcance
Ventana	w_	Público
Función de ventana	wf_	Pública, Privada, Protegida
Estructura de ventana	ws_	Pública, Privada, Protegida
Menú	m_	Pública
Función de Menú	mf_	Pública, Privada, Protegida
Estructura de Menú	ms_	Pública, Privada, Protegida
Objeto de Usuario	u_	Pública
Función de objeto de Usuario	uf_	Pública, Privada, Protegida

Estructura de Objeto de Usuario	us_	Pública, Privada, Protegida
Objeto Datawindow	d_	Pública
Objeto Query	q_	Pública
Control Datawindow	dw_	Pública
Objeto Estructura	s_	Pública
Objeto función	f_	Pública

Alcance de variables

Tipos	Prefijo	Alcance
Variables Globales	gx_	Accesible en cualquier parte de la aplicación
Variables Compartidas	sx_	Asociada con un tipo de objeto
variables de Instancia	ix_	Asociada con una instancia de un objeto como una ventana
Variables Locales	lx_	Accesible solamente en una sección de código del objeto
Argumentos de funciones	ax_	Dentro de la misma función.

donde 'x' se refiere a uno de los siguientes tipos de datos:

Tipo de Dato	Prefijo	Tipo de Dato	Prefijo
Ventana	w	Long sin signo	ul
MenuItem	m	Boolean	b
DataWindow	dw	String	s
Estructura	str	double	db
Objeto de usuario	uo	real	r
Integer	i	decimal	c
Integer sin signo	ui	date	d
Long	l	time	t
datetime	dt		

La estructura final de una aplicación desarrollada en *PowerBuilder* puede ejemplificarse mediante el siguiente diagrama:

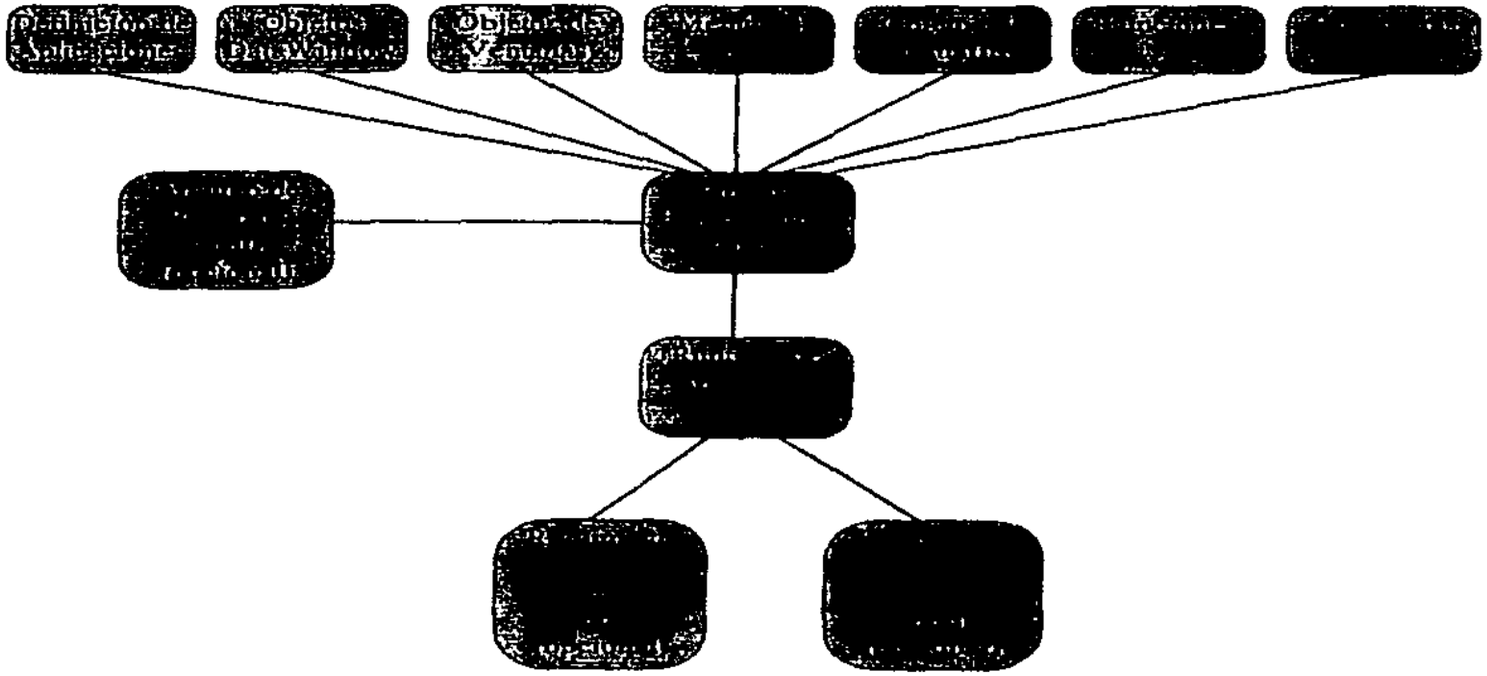


Fig 2.3 Creación de una aplicación con *PowerBuilder*

Con las características anteriormente establecidas en cuanto a los requerimientos y estándares que se deben utilizar, así como el esquema a seguir para la construcción de una aplicación en *PowerBuilder*, contamos ya con las bases y elementos suficientes para comenzar con la construcción del sistema, apoyados en el análisis y diseño que se realizó anteriormente sobre el sistema.

5.3.Planteamiento del esquema de desarrollo.

Para la construcción del sistema se planteó la idea de manejar módulos independientes y que paulatinamente se fueran agregando a un contenedor principal que los manejara, de tal manera que no se retrasara la construcción de un módulo nuevo.

Para lograr lo anterior, se manejaron elementos de programación orientada a objetos como son el polimorfismo, la herencia y el encapsulamiento, ya que al estar definidas las entradas y salidas que requieren cada uno de los objetos, se podían tratar como entes independientes, quedando por resolver solamente la manera en que se iban a comunicar entre sí. Esto trajo como ventajas que se construyera internamente toda la lógica de cada uno de los objetos mediante el uso de variables locales y de instancia a nivel de código escrito para hacer un uso más limitado de variables globales que pudieran cambiar en cualquier parte del programa y afectar la operación del mismo.

Para resolver la manera en que se iban a comunicar los objetos, se manejó un esquema (*figura 3.1*) del siguiente tipo:

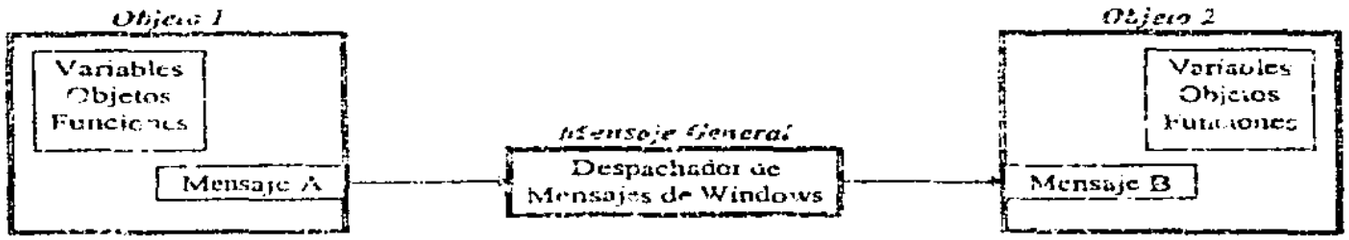


Fig. 3.1 Comunicación entre objetos.

Haciendo una similitud de la manera como se comunican los seres humanos en general, podemos identificar en la figura varios elementos que intervienen en éste proceso:

1. Una parte que envía un mensaje representado por el *Objeto 1*.
2. Otra parte que recibe el mensaje y lo procesa representado por el *Objeto 2*
3. Una lengua que contenga un conjunto de reglas y elementos comunes a ambos, representados por el objeto *Mensaje General*.

Se construyeron varios objetos del tipo estructura en un ámbito global al proyecto, representando mediante el objeto *Mensaje General* , y que contuvieran los elementos necesarios para la comunicación entre los objetos, como por ejemplo: el identificador del deudor mediante un número de folio.

A cada una de las ventanas que se deseaba comunicar, se declararon estructuras del tipo instanciado, referenciados por el ***tipo de Mensaje General***, de tal manera que ya se tenía definido el canal de comunicación y los parámetros que se iban a enviar y/o recibir.

Para ilustrar este proceso, se considerará un ejemplo: existen dos objetos muy importantes identificados por las ventanas ***W_Deudor*** y ***W_Bienes*** que internamente tienen declaradas estructuras instanciadas del tipo de estructura global ***S_Llave***, el cual está compuesto por elementos como ***región, sucursal y folio del deudor***; de tal forma que si deseamos enviar información de la ventana de deudores a la ventana de bienes, tan solo tenemos que llenar esa estructura referenciada, y como si fuera un paquete, lo enviamos al despachador de mensajes de Windows para que a su vez lo envie a su destino, siendo en este caso la ventana ***W_Bienes*** que lo recibe en su propia estructura instanciada, procesándola posteriormente como le sea conveniente.

Una vez definida la manera en la cual se comunicarían los objetos, quedó solamente por resolver la forma en la cual se iban a construir cada uno de los módulos. Para ésto, se aprovechó la característica de *PowerBuilder* de permitir la definición de una ventana y un menú asociado, de tal manera que pudiera cargarse en una ventana principal para su uso. Es decir, cada módulo estaría representado por una ventana y un menú que contuviera todas las acciones sobre dicho objeto.

Mas sin embargo, quedaba un punto por definir: la manera en la cual se iban a presentar cada uno de los módulos sin que se viera el efecto del cambio de menú y sobre todo, mantener un menú que tuviera la misma apariencia para cualquier módulo , no importando las funciones a realizar por cada uno de ellos. Esto se resolvió utilizando la característica de la herencia de *PowerBuilder* en la construcción de objetos, ya que se diseñó un menú principal que reuniera las principales caracteristicas de una aplicación bajo ambiente *Windows* y que además tuviera una opción dentro del menú que permitiera un cambio dinámico en cuanto a funcionalidad. El menú principal quedaría entonces así constituido por un objeto que heredaría el esquema de presentación de los elementos a cada uno de los menús generados a partir de él, ya que además de compartir código fuente utilizable en cualquier punto de la aplicación, serviría para asociarlos a cada uno de los módulos que conformarían el sistema, de tal manera que se mantuvieran homogéneos. La *figura 3.2* muestra la estructura jerárquica de cada uno de los menús:

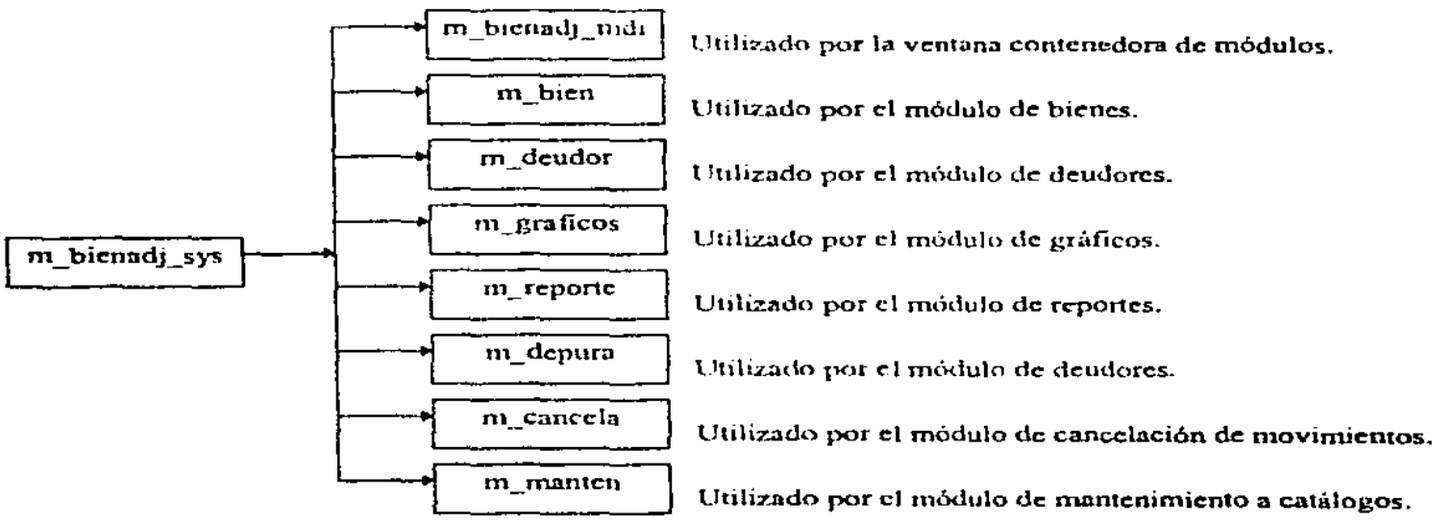


Fig 3.2 Estructura de Menús.

El menú principal (*figura 3.3*), del cual se iban a generar cada uno de los menús propios de cada módulo, se ajustó a una organización de elementos comunes de un sistema bajo ambiente Windows, bajo la siguiente forma:

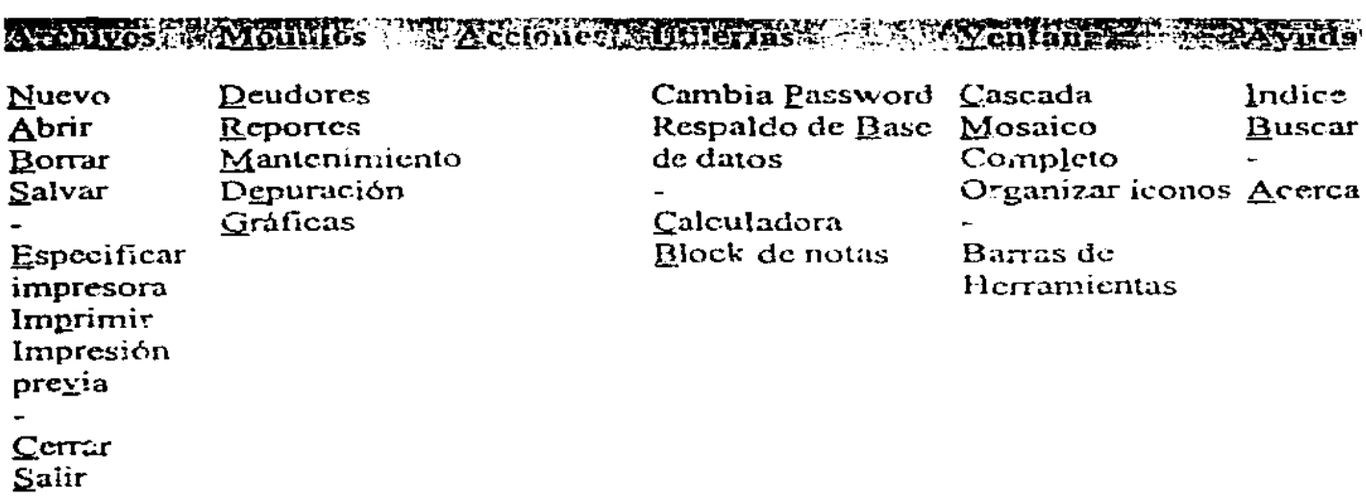


Fig 3.3 Organización del Menú principal

El código para el manejo de cada uno de los módulos se construyó a este nivel declarando llamadas a funciones con un mismo nombre. Se procedería después en cada módulo a construir dichas funciones con ese nombre, de tal manera que cada una de ellas tomarían diferentes acciones dependiendo del contexto en que se declararon. Solamente se dejó vacio un elemento del menú llamado ***Acciones*** el cual se utilizaría para la declaración de elementos de control propios de cada módulo, de esta manera , al heredar cada menú este esquema, solamente se declararían nuevos elementos bajo el nodo ***Acciones***, y se habilitarían elementos que tuvieran relación con el módulo asociado.

También resulta importante señalar que cada uno de los botones que presenta el sistema, están heredando características en cuanto al tamaño del objeto con el fin de mantener un tamaño estándar para cada uno de ellos. Por ejemplo, los botones estándares de ***Aceptar***, ***Cancelar***, ***Salir***, ***Imprimir***, etc, se aislaron por completo de cada uno de los módulos en construcción almacenándose en forma de objetos definidos por el usuario en una libreria de PowerBuilder, permitiendo heredar el estilo del objeto definido a cada uno de los objetos que se necesitaron durante la construcción de las ventanas. Esto tiene por ventajas que si quisieramos modificar alguna característica de algún botón para toda la aplicación, únicamente tendríamos que alterar las dimensiones y características del objeto padre de tal manera que este hecho afectara a cada uno de los objetos hijos.

Los puntos anteriormente expuestos se realizaron considerando la parte de la interfaz con el usuario y la forma de armar el esqueleto de la aplicación, de tal forma que solamente quedaba por definir la forma en que se iban a realizar los accesos a la base de datos. Para construir esta parte de la aplicación se tomó en cuenta el tipo de modelo cliente/servidor que se iba a utilizar. Recordando algunos puntos mencionados en el capítulo relacionado con los distintos tipos de

arquitectura cliente/servidor, observamos que existen varios modelos (figura 3.4) que permiten definir un esquema de operación.

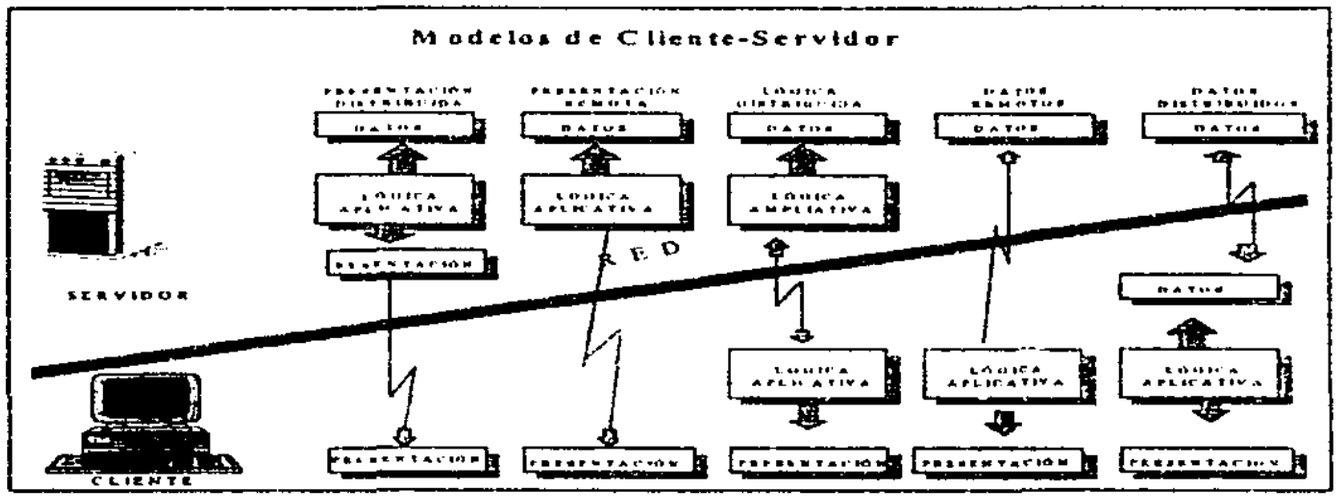


Fig 3.4 Modelos cliente/servidor

Al abordar un esquema del tipo Datos Remoto, es decir, mediante una arquitectura de aplicación de dos niveles, en el cual la aplicación se divide en partes que se ejecutan en una estación cliente y en un servidor, teníamos que plantear una forma de definir la manera de comunicarse con la base de datos para cumplir con este tipo de arquitectura. Debido a que la parte del servidor estaría a cargo de un manejador de base de datos que realizaría la gestión de la información, y que la herramienta de desarrollo de la parte del cliente (en este caso *PowerBuilder*) permitía la declaración de sentencias del tipo SQL resultó conveniente construir cada acceso a la base de datos mediante funciones de tipo global, que se almacenaron en una librería de PowerBuilder para su identificación posterior. Para ello se definió la construcción de funciones bajo el siguiente formato:

f_<acción>_<tabla>

donde **tabla** se refiere a la tabla de la Base de Datos utilizada y **acción** toma alguno de los siguientes valores

d - Delete	i - Insert
u - Update	s - Select

Se utilizó además una base de datos en modo stand-alone como es Watcom 4.0 para la etapa de pruebas del sistema, el cual utiliza sentencias del tipo SQL para realizar las consultas a las tablas, siendo su estructura y sus llamadas muy similares a las de cualquier otro manejador de bases de datos. Por otra parte, el construir las tablas en este ambiente y separar los accesos a la base de datos, facilitaría su migración a otros ambientes de producción, únicamente construyendo las tablas y direccionando los accesos a otro DBMS.

Los puntos anteriormente expuestos son de los más importantes en la construcción de sistemas bajo ambiente cliente/servidor ya que durante la etapa de construcción del prototipo y del sistema en sí, es posible construir en una base de datos de tipo local (como la mencionada) las tablas y las relaciones que se guardan, mientras los accesos a través de sentencias SQL se construyen para la parte del cliente. Esto nos permite realizar pequeñas pruebas a nivel local para definir el flujo de información, observando así su comportamiento, de tal manera que se facilite una migración posterior a cualquier otro manejador de base de datos de manera casi transparente.

Una vez planteada la manera en que se organizarían cada uno de los módulos, su integración en el contenedor principal, y la forma de acceder a la base de datos, quedaba por definir la forma en la cual se almacenarían cada uno de los objetos construidos en el ambiente de desarrollo de *PowerBuilder*. Se manejó el esquema de librerías que utiliza la herramienta, quedando distribuidos en archivos con extensión *.PBL:

Bienes	Módulo de bienes.
Bienesad	Sistema de control de bienes adjudicados Ver 1.0. Punto de Entrada
Depura	Módulo para la depuración de registros
Deudores	Módulo de deudores
Funcdb	Funciones con acceso a base de datos
Funcomun	Funciones comunes al sistema
Graficas	Módulo de gráficas
Mantea	Módulo de mantenimiento a catálogos
Objcomun	Objetos comunes al sistema
Reportes	Módulo de reportes

Dichos archivos, se compilan y enlazan mediante un esquema como el mostrado con anterioridad, dejando archivos con extensión *.PBD (librerías dinámicas), y uno solo con extensión *.EXE el cual representa en sí al programa y que se encarga de llamar a cada una de las librerías dinámicas en tiempo de ejecución del mismo.

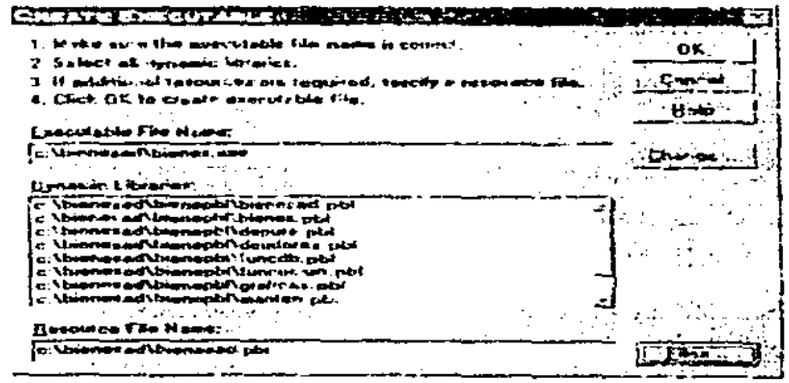


Fig 3.5 Creación del archivo ejecutable

A continuación se describirá la manera en que quedó construido el sistema y la explicación a cada uno de los elementos que conforman al mismo

5.4.Construcción de los módulos.

Basándonos en el diagrama de conexión entre objetos, podemos observar que la entidad más importante es la de ***Bienes*** ya que sobre ella giran todas las demás entidades identificadas. Para aprovechar el esquema que nos ofrece *Windows* como es el hecho de generar una aplicación que contenga una ventana principal que sea la contenedora de una serie de ventanas hijas, se diseñó una ventana similar (*figura 4.1*), con la finalidad que contuviera las principales entidades y que a la vez se pudieran descomponer en módulos accesibles en cualquier momento de la aplicación.

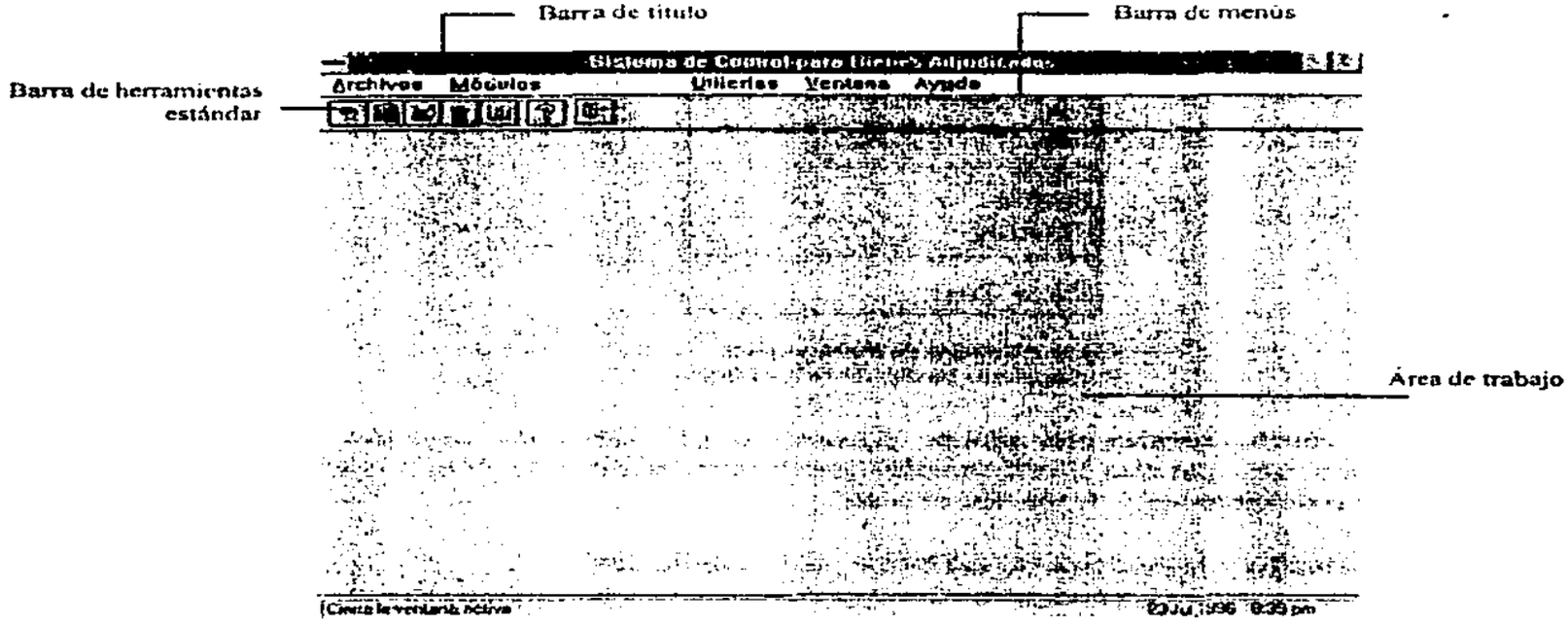


Fig 4.1 Pantalla principal

Componentes

Barra de herramientas

Las barras de herramientas proporcionan un acceso instantáneo a las herramientas y comandos de uso más frecuente, así mismo, contienen los módulos que constituyen el sistema. Cuando se inicia el sistema aparece una barra de herramientas estándar justo debajo del menú. Dicha barra de herramienta tiene la característica de permitir moverla en cualquier lado de la pantalla, ponerla como una ventana flotante, visualizar el texto asociado a cada icono, o simplemente trabajar con los comandos del menú si se desea desaparecerla.

Para tener una aplicación similar que se ajuste a los estándares de *Windows* , la ventana principal permite ajustar las dimensiones del mismo, así como guardar automáticamente la información acerca del tamaño y la ubicación de las barras de herramienta, con lo cual el usuario no deberá ajustarlas cada vez que utilice el programa.

En general, las acciones a tomar con respecto a la barra de herramientas son las siguientes:

- Para mostrar u ocultar una barra de herramienta.

Colocar el mouse sobre la barra de herramienta deseada, presionar el botón derecho del mouse, e inmediatamente aparecerá un submenú como el que se muestra en la figura 4.2.

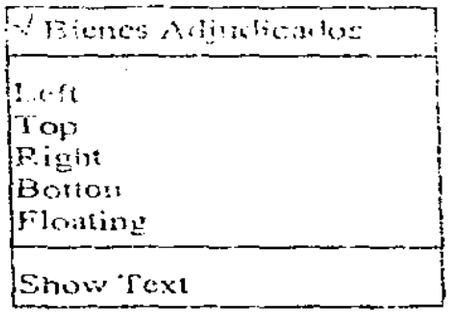


Fig 4.2 Cuadro de diálogo para personalizar barras de herramientas.

Con el botón derecho del mouse presionado se selecciona la barra de herramienta (indicada mediante una paloma si está visible) que desee mostrar o ocultar, se suelta el botón y la barra aparecerá o desaparecerá según sea el caso.

- Cambiar el tamaño y la posición de las barras de herramientas.

Las barras de herramientas no tienen una posición fija en la pantalla, se pueden mover mediante un click en un lugar de la barra de herramientas apareciendo de nuevo la ventana principal. Colocando el mouse en la palabra que describa el lugar a donde quiere mover la barra de herramientas y soltando el botón, inmediatamente después la barra cambiará al lugar elegido.

Para cambiar el tamaño de la barra de herramientas, hay que colocar el mouse sobre el texto marcado como "***Show Text***" y soltar el botón, inmediatamente después aparecerán los botones grandes y con una descripción de su funcionamiento.

Barra de menús

Las barras de menús (*figura 4.3*) proporcionan un acceso a las herramientas por medio de cuadros de diálogos en los que deberá seleccionar una opción entre varias. Los comandos que van seguido de puntos suspensivos siempre harán aparecer un cuadro de diálogo.

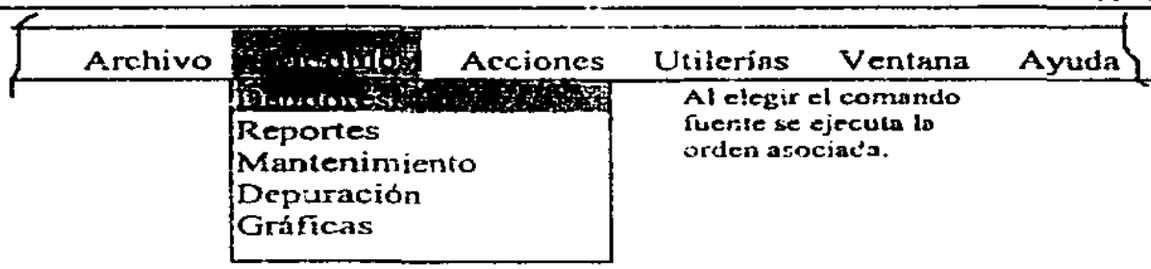


Fig 4.3 Menú de la aplicación.

Haciendo un acercamiento al menú general del sistema (*figura 4.4*) es posible la identificación de los distintos módulos que conforman el sistema, ubicándose la manera en que están distribuidos y la forma en que pueden ser accedidos dentro de algún elemento de la barra de herramientas.

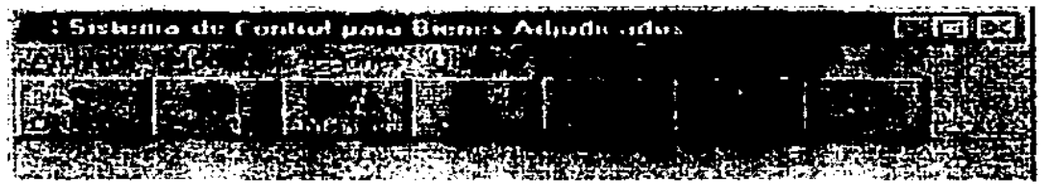


Fig 4.4 Menu principal del sistema.

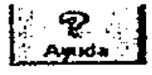
Deudores
Permite la introducción de nuevos deudores al sistema.

Reportes
Permite la visualización de los distintos tipos de reportes que el sistema maneja.

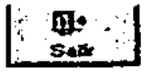
Mantenimiento
Permite el mantenimiento de los catálogos del sistema

Depuración
Permite la depuración de registros que el sietma ya no utiliza.

Gráficas
Presenta un conjunto de gráficas que muestran los diferentes tipos de bienes contra ciertos parámetros comparativos.



Ayuda
Permite visualizar la ayuda en línea del sistema.



Salir
Aunque no es propiamente un módulo, permite una salida más rápida del sistema.

En este momento comenzaremos a explicar cada uno de los diferentes módulos del sistema:

5.4.1.Módulo de Deudores

La función principal del sistema es la administración de los bienes adjudicados por Región/Sucursal. Recordando que el concepto "Región/Sucursal" significa que, al recibir un bien (ya sea por adjudicación o Dación en pago), el deudor está relacionado junto con su bien(es) mediante una Región y una Sucursal en la cual se encuentra dicho bien. Tener identificado al deudor nos permite poder llevar a cabo un control sobre los bienes al conocer su origen y la manera en que fue obtenido.

La siguiente ilustración (*figura 4.5*) nos muestra la pantalla del Módulo de Deudores.

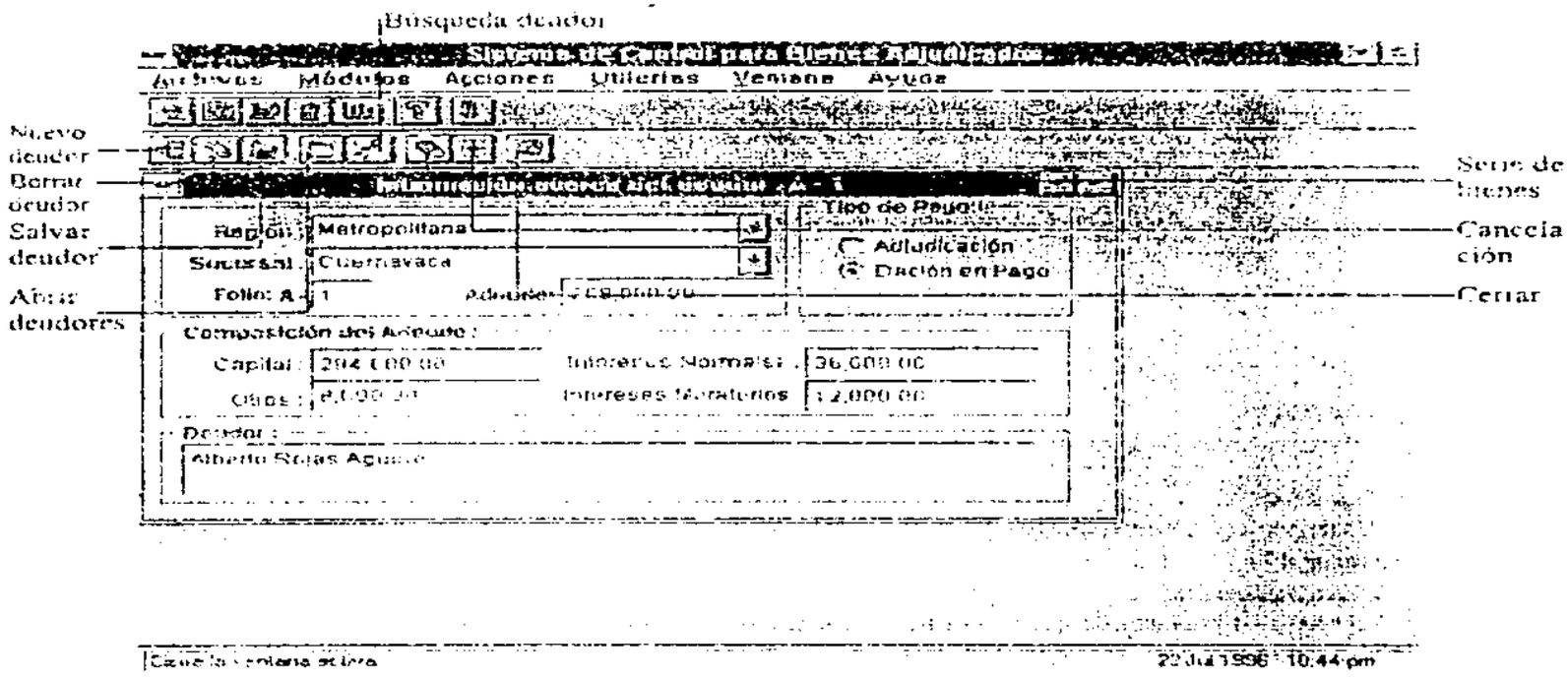


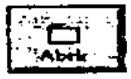
Fig 4.5 Pantalla de deudores

Nuevo
Permite la introducción de un nuevo deudor al sistema identificando la Región/sucursal a la que pertenece (datos obligatorios) así como la identificación de la manera como le fueron adjudicados sus bienes (dación o adjudicación)

Borrar
Permite la baja física en la base de datos del deudor presente en la pantalla, verificando la existencia de bienes asociados al mismo.

Salvar la información del deudor

Una vez introducida la información del deudor, el próximo paso consiste en salvar la información. Esto es posible seleccionando el tercer icono del la barra de herramientas; si faltara información necesaria para el tramite de alta, el sistema enviará un mensaje para que proporcione la información faltante, una vez que la información sea correcta, el deudor será dado de alta.



Abrir una relación de deudores por Región/Sucursal

Al elegir esta opción aparece una pantalla (*figura 4.6*) la cual muestra un catálogo de deudores, que se obtiene al filtrar la información por Región/Sucursal.

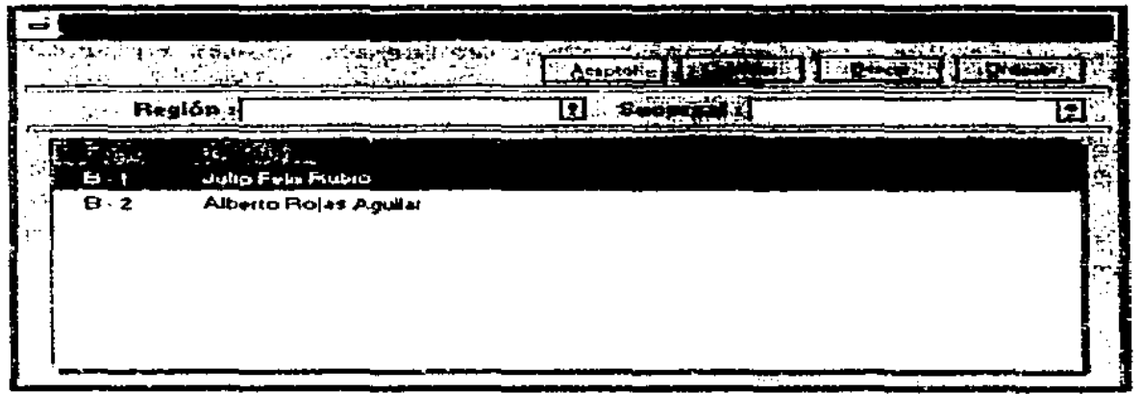


Fig 4.6 Catálogo de deudores

Una vez seleccionada la región y sucursal, aparecerán los deudores que pertenecen a la opción elegida. Es posible seleccionar a un deudor dando un "*doble click*" o presionando el botón de ***Aceptar***, lo cual origina que la ventana se cierre, apareciendo en la plantilla la información correspondiente al deudor elegido.

Buscar un deudor

La *figura 4.7* muestra la ventana que aparece una vez que se haya presionado el icono de "Buscar", esta ventana permite la carga de un deudor de una manera más específica.

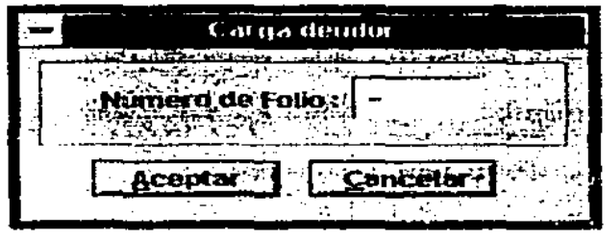


Fig 4.7 Búsqueda de deudor por número de folio

Si se conoce el número de folio de un deudor , la ventana ilustra una manera rápida de encontrar el deudor. Una vez proporcionado el número de folio del deudor a buscar, se presiona el botón de aceptar y si el deudor existe aparecerá en la ventana de deudores la información correspondiente al número de folio proporcionado.

Antes de continuar con los dos siguientes botones correspondientes a los Bienes del deudor y la Cancelación de Movimientos es necesario abrir un paréntesis para justificar el hecho de que existan propiamente en el módulo de deudores. Dichos botones representan en sí a dos módulos que se podrían manejar de manera independiente al módulo de Deudores, sin embargo, se optó por manejarlos aquí, debido a que resultaba para el usuario más fácil identificar el bien que está asociado al deudor, y aplicar así una cancelación de movimientos de venta, además de evitar una sobrecarga de información del módulo de Bienes. Por otra parte, resultó conveniente identificar primero al deudor para llegar al módulo de Bienes, debido a que obteníamos un subconjunto del universo de bienes, discriminando así solamente aquellos del interés del usuario.

Bienes del deudor

Una vez que se dio de alta un nuevo deudor, el paso siguiente consiste en dar de alta sus bienes. En el caso de que el deudor ya exista, se pueden consultar los bienes ya existentes o dar de alta nuevos. La *figura 4.8* muestra la ventana de bienes del deudor.

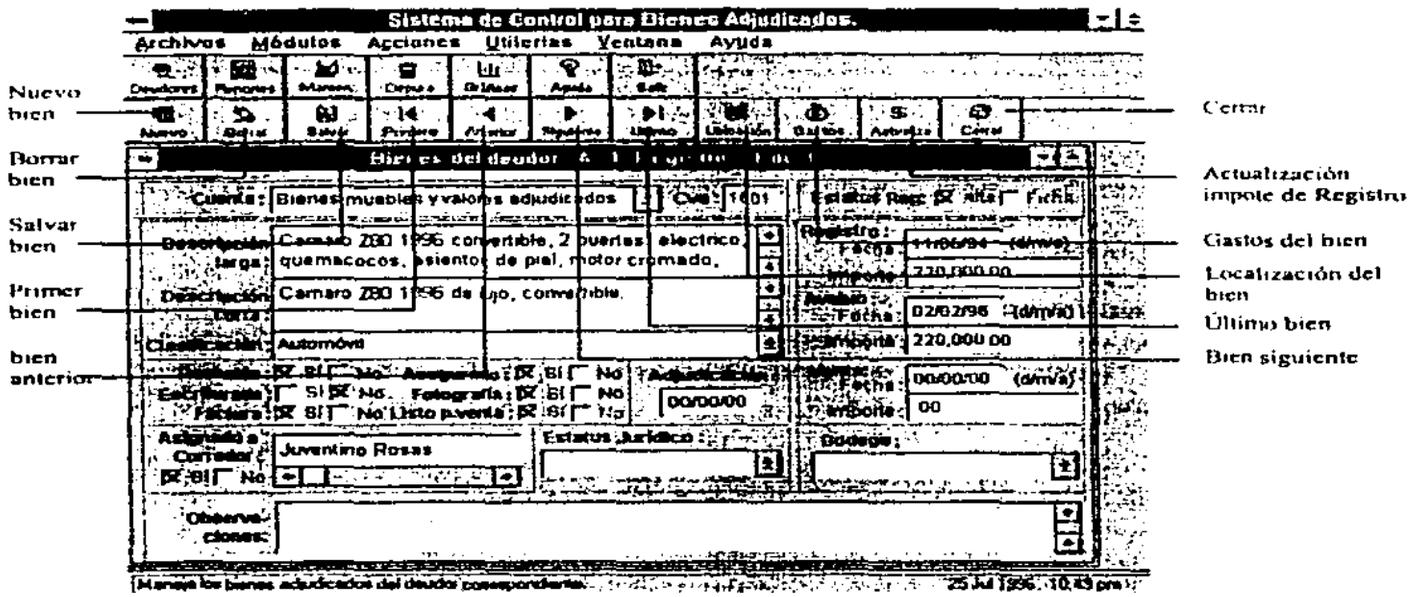


Fig 4.8 Ventana de los bienes

Nuevo bien
Esta opción permite dar de alta un nuevo bien para el deudor seleccionado previamente. El sistema tiene la capacidad de validar que, la información proporcionada sea la mínima requerida para realizar el alta del bien. Una vez que el bien fue dado de alta, en la barra de menús se habilita el menú "Acciones" con las acciones que se pueden realizar sobre el nuevo bien ingresado al sistema. Estas acciones las determina el sistema de forma automática, tomando en cuenta el tipo del bien y su estatus jurídico.

- ***Bienes muebles y valores adjudicados.*** Al realizar una alta de este tipo de bien, si el bien está listo para la venta, podemos ver en el menú "Acciones" las acciones que podemos realizar sobre este bien.

La *figura 4.9* muestra un Bien Mueble Listo para la Venta, y las acciones que podemos realizar sobre él.

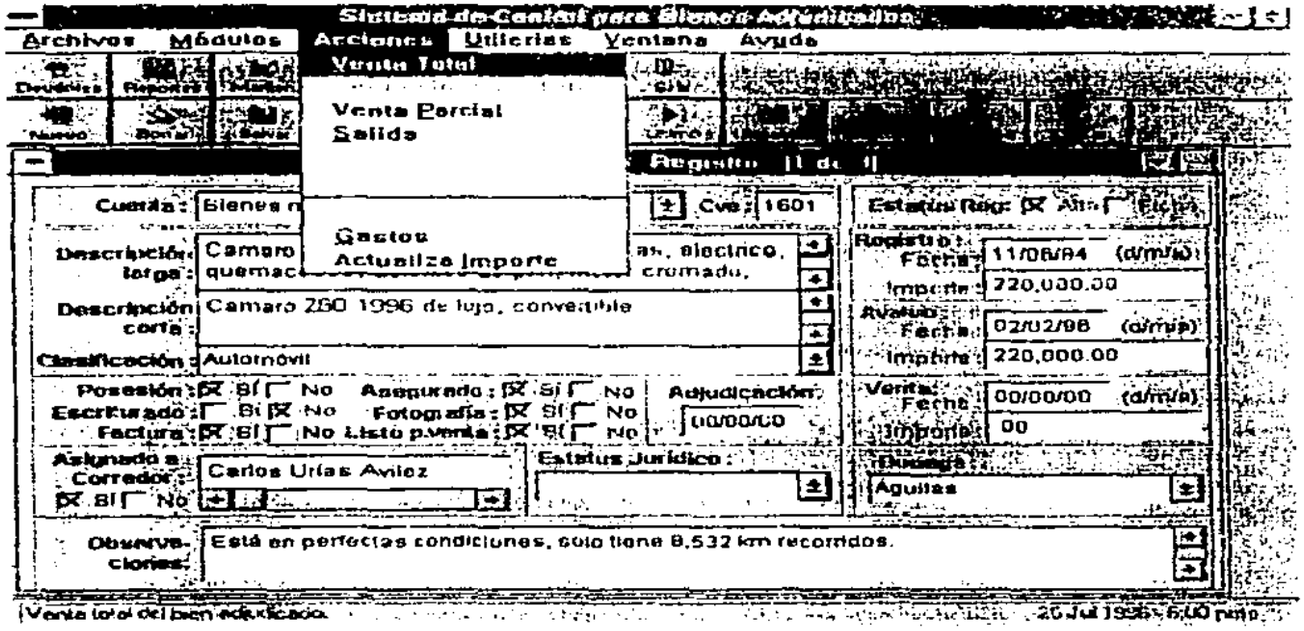


Fig 4.9 Acciones sobre los Bienes muebles y valores adjudicados.

En la ilustración anterior podemos apreciar, en la primera sección del menú "Acciones" los siguientes casos:

1. ***Venta Total,*** esta acción realiza un venta total del bien y el sistema la registra en las salidas del mes.
2. ***Venta Parcial,*** esto es sólo en el caso de que sea un lote de bienes.
3. ***Salida,*** se toma como salida un bien cuando el banco decide que el bien forme parte de sus activos.

En el caso de que el bien no estuviera listo para la venta, la única acción que podemos tomar es realizarle una salida.

En la segunda sección del menú "Acciones", tenemos lo siguiente:

1. ***Ubicación,*** la cual en este caso por ser un bien mueble se encuentra inhabilitada.
2. ***Gastos,*** esta acción permite llevar un control de los gastos que el bien ocasiona, como pueden ser tenencia, mantenimiento, verificación, etc.
3. ***Actualiza Importe,*** esta acción permite actualizar el importe de registro, ya que puede darse el caso de que la C.N.B. y V. decida que un bien tenga que ser revaluado, o también puede ser debido a que se haya cometido un error en la captura.

- ***Bienes inmuebles adjudicados.*** Al realizar una alta de este tipo de bien, si el bien está listo para la venta, podemos ver en el menú "Acciones" las acciones que podemos realizar sobre este bien.

La *figura 4.10* muestra un Bien Inmueble Listo para la Venta, y las acciones que podemos realizar sobre él.

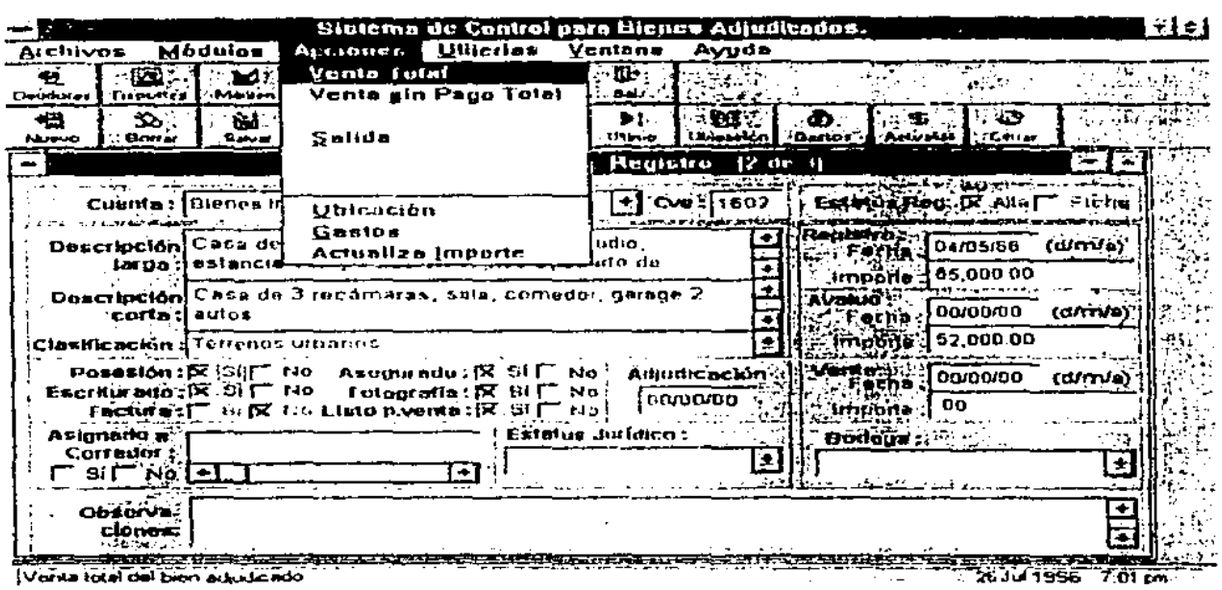


Fig 4.10 Acciones sobre los Bienes inmuebles adjudicados.

En la ilustración anterior podemos apreciar, en la primera sección del menú "Acciones" los siguientes casos:

1. ***Venta Total,*** esta acción realiza un venta total del bien y el sistema la registra en las salidas del mes.

2. ***Venta sin Pago Total,*** esta sólo puede realizarse en los bienes inmuebles, esto es, si se diera el caso de un anticipo, después de realizarlo, se habilita la acción "Liquidación" para realizar posteriormente la liquidación del bien.
3. ***Salida,*** se toma como salida un bien cuando el banco decide que el bien forme parte de sus activos.

En el caso de que el bien no estuviera listo para la venta, la única acción que podemos tomar es realizarle una salida.

En la segunda sección del menú "Acciones", tenemos lo siguiente:

1. ***Ubicación,*** aquí es donde se anota la dirección de donde está ubicado el bien.
2. ***Gastos,*** esta acción permite llevar un control de los gastos que el bien ocasiona, como pueden ser tenencia, mantenimiento, verificación, etc.
3. ***Actualiza Importe,*** esta acción permite actualizar el importe de registro, ya que puede darse el caso de que la C.N.B. y V. decida que un bien tenga que ser revaluado, o también puede ser debido a que se haya cometido un error en la captura.

Borrar un bien
Esta opción se ejecuta haciendo *click* en el segundo icono de la barra de herramientas de la pantalla de bienes o con la segunda opción del menú "Archivo" en la barra de menús.
Se debe tener especial cuidado con esta orden ya que al ejecutarla el bien se perderá físicamente de la Base de Datos. Es por ello que se muestra una señal de advertencia al usuario.

Salvar un bien
Al ingresar para un deudor un nuevo bien, el paso siguiente es salvar el bien, esto se logra haciendo *click* en el tercer icono de la barra de herramientas de la pantalla de bienes o con la tercera opción del menú "Archivo" en la barra de

Debido a que un deudor puede tener más de un bien asociado, existen cuatro iconos que permiten desplazarse del primero al último bien y viceversa. A continuación se describe cada una de ellos:

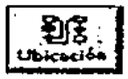
Primero
Este icono nos permite, estando en cualquier número de bien, colocarnos en el primer elemento del total de bienes.

Anterior
Este icono permite retroceder en los bienes de uno en uno hasta llegar al primero.

Siguiente
Este icono permite avanzar en los bienes de uno en uno hasta llegar al último.

Último

Este icono nos permite, estando en cualquier número de bien, colocarnos en el último elemento del total de bienes.



Ubicación del Bien

Cuando se ingresa un bien inmueble al sistema, es necesario saber dónde está ubicado el bien, esto se logra seleccionando el icono de "Ubicación del Bien", inmediatamente después de ser seleccionado, aparece una pantalla en la cual se captura la dirección del bien inmueble. La *figura 4.11* muestra cada uno de los datos de la pantalla.

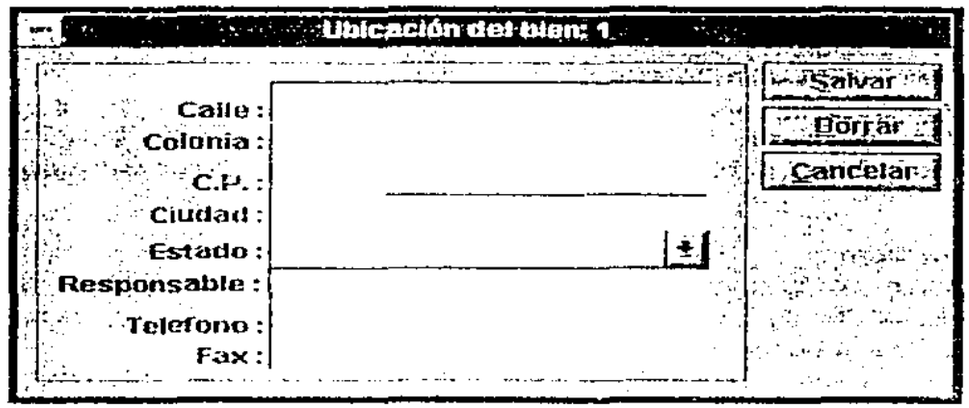
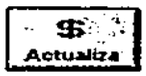


Fig 4.11 Ventana de ubicación del Bien.

Gastos del Bien

Una vez que la banca realizó la adjudicación de algún bien , si este no se vende, le ocasiona gastos; debido a esto, el sistema lleva un control de los gastos ocasionados por un bien, esto es, para poder llevar un control de gastos para efectos contables. La *figura 4.12* muestra la pantalla de gastos para un bien.



Actualización del Importe de Registro

Debido a que se puede dar el caso de que la CNByV ordene que se realice una revaluación sobre los bienes que en un tiempo determinado no han podido ser vendidos y tenga que ser modificado el Importe de Registro del bien o que se haya cometido un error de captura y como consecuencia el Importe de Registro tenga que modificarse. Debido a eso, el sistema tiene la opción de modificar el Importe de Registro del Bien. La *figura 4.14* muestra la pantalla de actualización (o modificación) al Importe de Registro.

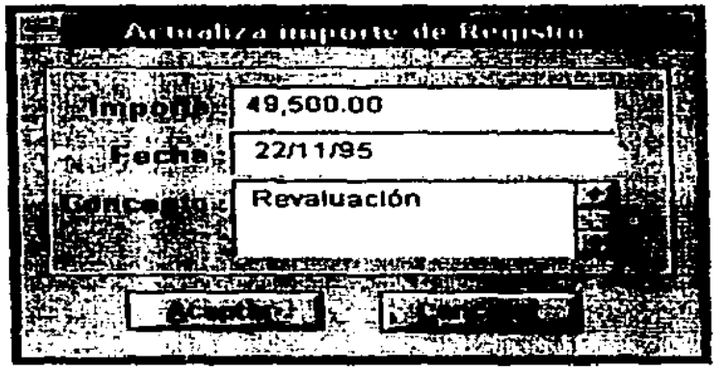


Fig 4.14 Actualización del importe de registro

Una vez que los datos de la pantalla son correctos, se presiona el botón de aceptar y el Importe de Registro se actualiza en la pantalla del bien. El sistema realiza los efectos contables que la actualización del Importe de Registro ocasiona.

Cerrar la pantalla de bienes
Este icono lo único que realiza es cerrar la pantalla de bienes y retornar a la pantalla del deudor.

En este punto termina el apartado de bienes del deudor; continuaremos con los dos últimos iconos del "Módulo de Deudores":

Cancelación de movimientos.
Permite la cancelación de cualquier movimiento de venta efectuado sobre un bien en particular. Cuenta así mismo con cuatro iconos como parte de un menú que permiten desplazarse del primero al último bien y viceversa, similares a los ya mostrados con anterioridad. La *figura 4.15* nos muestra una ventana típica con un tipo de movimiento de venta asociado.

Cerrar el "Módulo de Deudores"
Este icono lo único que realiza es cerrar el "Módulo de Deudores" y retornar a la pantalla principal del sistema.

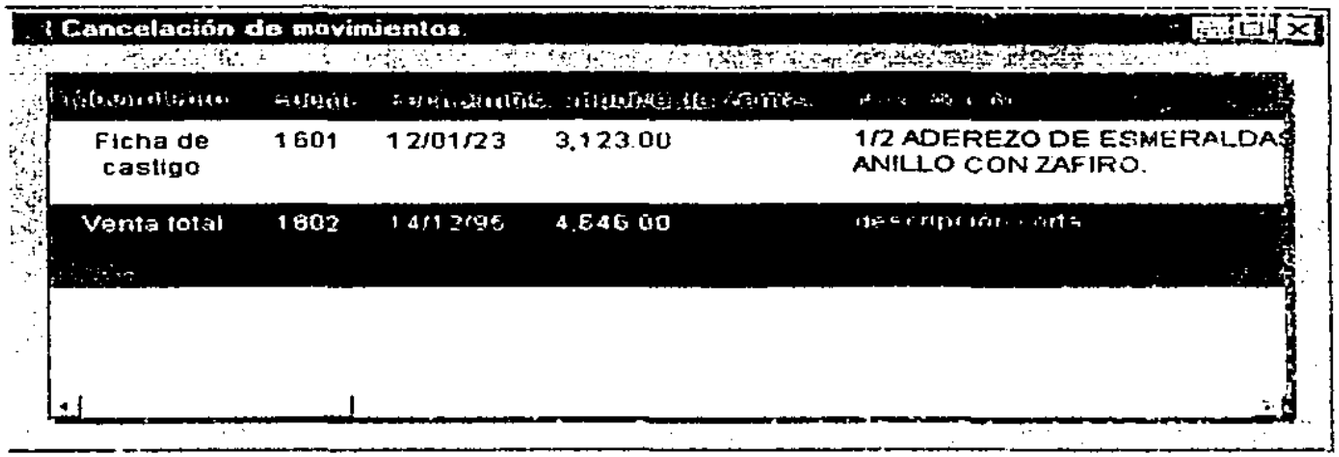


Fig 4.15 Cancelación de movimientos

Iconos asociados al módulo de cancelación de movimientos:

Primero.
Este icono permite desplazarse al primer registro del arreglo.

Anterior.
Este icono permite desplazarse al anterior registro con un movimiento hacia arriba.

Siguiente.
Este icono permite desplazarse al siguiente registro con un movimiento hacia abajo

Último.
Este icono permite desplazarse al último registro.

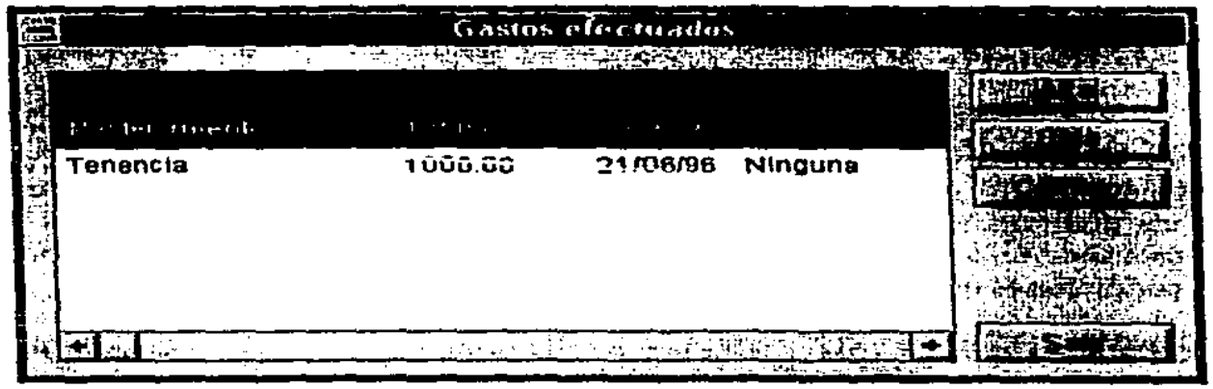


Fig 4.12 Gastos del bien

Para realizar el alta de un gasto, presionamos el botón de alta y luego aparecerá una pantalla para ingresar los datos de los encabezados de la figura anterior (Concepto, Importe, Fecha y Observaciones). La siguiente ilustración (*figura 4.13*) muestra la pantalla de alta para gastos del bien.

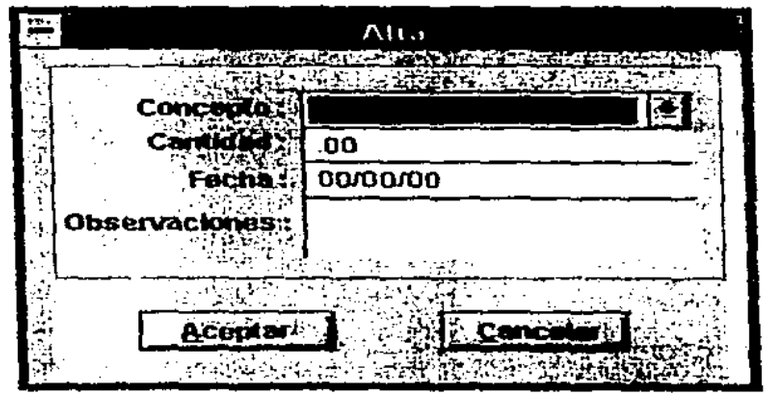


Fig 4.13 Alta de gastos de los bienes

Una vez que la información en la pantalla está completa, el siguiente paso es presionar el botón de "Aceptar" y la información que se ingresó a la pantalla de "Altas", pasará a la pantalla de "Gastos del Bien".

5.4.2.Módulo de Reportes.

Con el fin de ayudarle al usuario a presentar la información de manera clara y ordenada acerca de los deudores y/o sus bienes, el sistema contiene un módulo de reportes el cual permite al usuario ver, de distintas formas, la información que tiene el sistema incluyendo la opción de impresión. En esta sección se explicará cada uno de los reportes que contiene el sistema de forma detallada y las condiciones que cada uno incluye.

La siguiente ilustración (*figura 4.16*) muestra la pantalla principal del módulo de reportes.

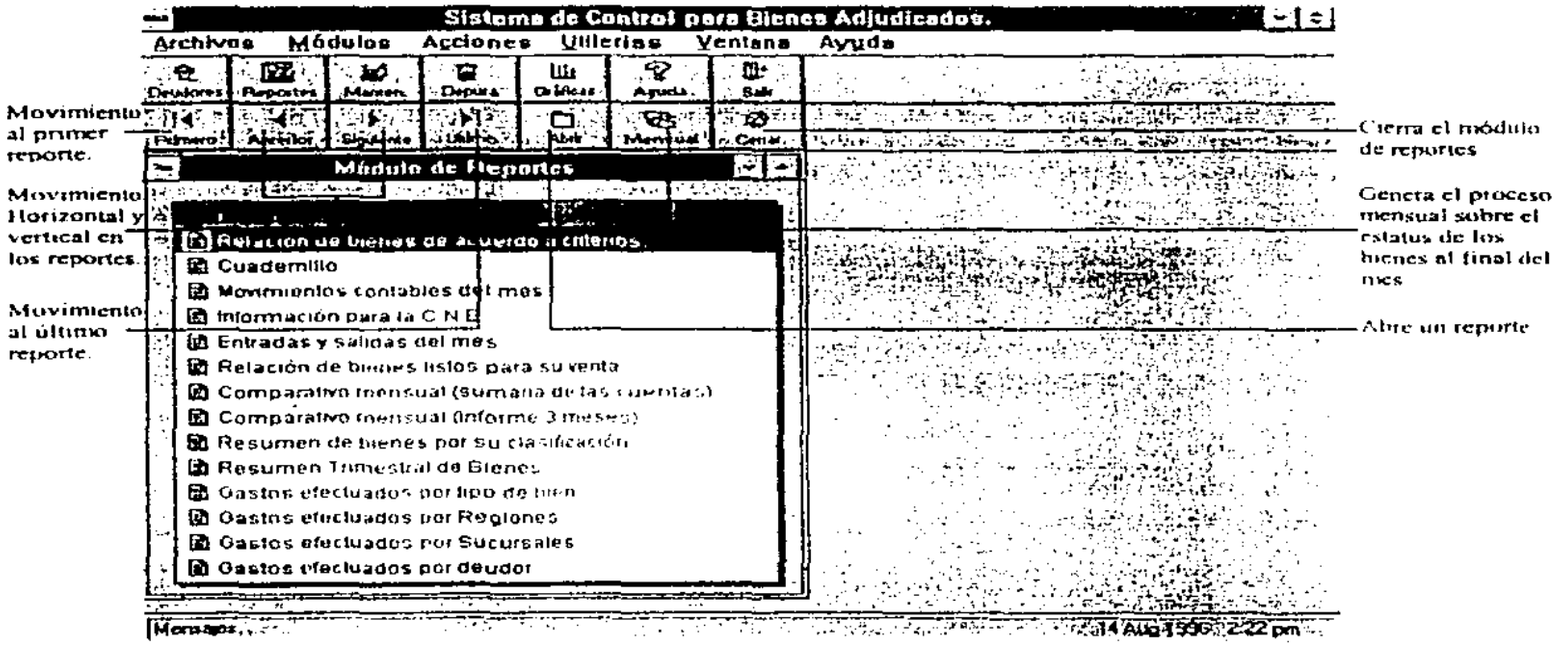


Fig 4.16 Modulo de reportes

A continuación se explicarán cada uno de los iconos de la barra de herramientas asociada a la ventana "Módulo de Reportes".

Primero.
Este icono permite desplazarse al primer reporte, estando colocado en cualquier reporte de la ventana "Módulo de Reportes".

Anterior.
Este icono permite desplazarse con un movimiento hacia arriba de reporte en reporte, estando en cualquier reporte de la ventana "Módulo de Reportes".

Siguiente.
Este icono permite desplazarse con un movimiento hacia abajo de reporte en reporte, estando en cualquier reporte de la ventana "Módulo de Reportes".

Último.
Este icono permite desplazarse al último reporte, estando en cualquier reporte de la ventana "Modulo de Reportes".

Abrir.
Este icono permite abrir el reporte seleccionado con la barra azúl. Cabe mencionar que un reporte puede abrirse haciendo doble clic sobre el texto del reporte deseado.

Mensual (Proceso mensual).
Este icono permite ejecutar un proceso mensual el cual realiza un concentrado sobre los bienes por región y sucursal, por tipo de cuenta, cuáles están listos o no listos para la venta. Este proceso sólo puede ejecutarse una vez por mes dependiendo del que se elija, ya que si se ejecuta dos veces se corre el riesgo de perder información de un mes anterior, esto se debe a que guarda información en dos tablas temporales. La *figura 4.17* nos muestra una ventana típica para realizar este proceso.

Cerrar.
Este icono permite cerrar el módulo de reportes.

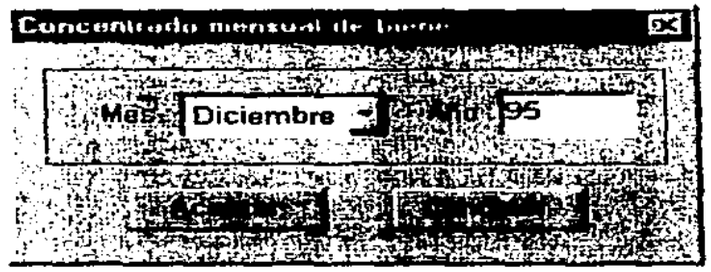


Fig 4.17 Concentrado mensual.

A continuación se explicará cada uno de los reportes que aparecen en la *figura 4.16*, pero antes explicaremos la barra de herramientas de los reportes, la cual es común para cada uno de ellos.

Condiciones.
Este icono establese las condiciones del reporte, es decir, si el reporte tiene más de una forma de presentar los datos. Es por medio de este icono que elegimos la forma de presentarlos.

Salvar... .
Este icono nos permite salvar el contenido del reporte en formato .xls, de esta manera podemos abrir el reporte desde *Microsoft Excel* y manipular la información del reporte.

Siguiente.
Debido a que el reporte puede tener más de una página, es por medio de este icono que podemos desplazarnos al resto de las páginas del reporte.

Anterior.
Este icono permite hacer lo contrario al icono anterior, esto significa, que estando en una página distinta a la primera, permite retroceder en una página cada vez que sea presionado.

Imprimir
Este icono permite imprimir el contenido del reporte.

Previo
Este icono, como su nombre lo dice, hace un previo, en distintos porcentajes, del reporte para revisarlo antes de mandarlo a impresión. A continuación se muestra la ventana (*figura 4.18*) que aparece al presionar este icono.

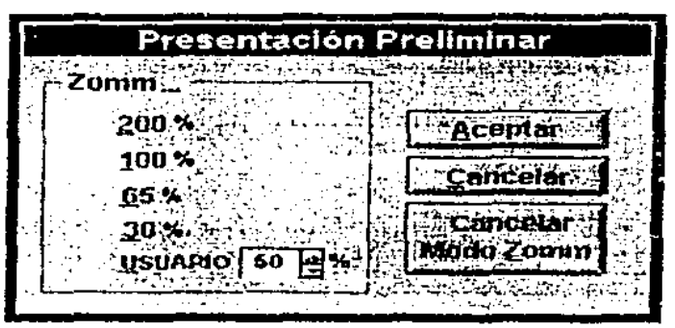


Fig 4.18 Zoom para reportes.

5.4.2.1.Relación de bienes de acuerdo a criterios.

Este reporte permite ver los bienes de acuerdo a una serie de criterios que se presentan en el icono de *"Condiciones"*, ya que cuenta con diferentes filtros como son los bienes por región/sucursal, listos o no listos para la venta, etc. La siguiente ilustración *(figura 4.19)* muestra el reporte Relación de bienes de acuerdo a criterios.

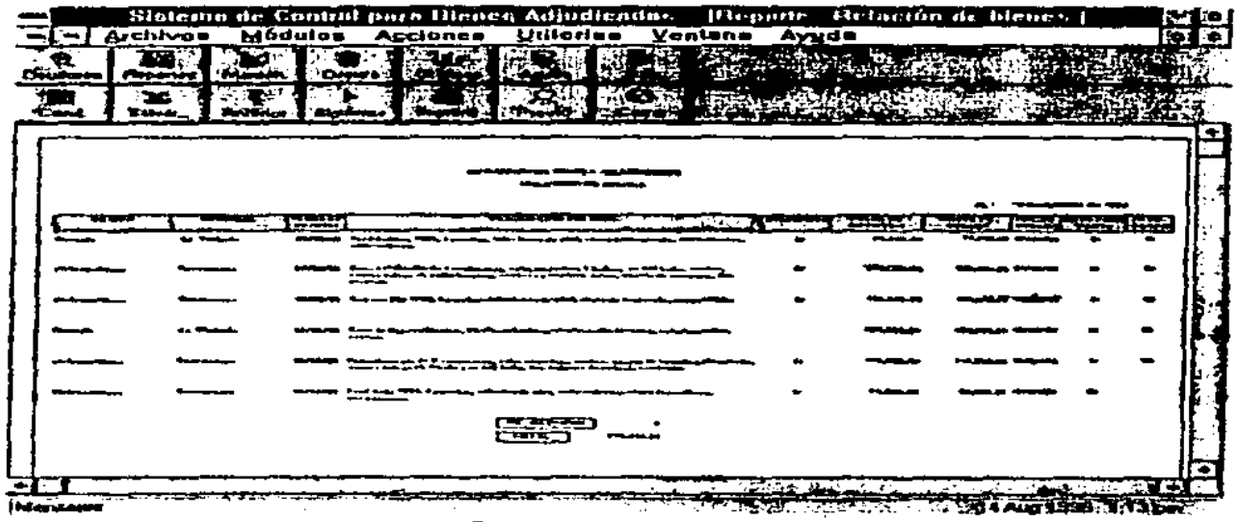


Fig 4.19 Reporte Relación de bienes de acuerdo a criterios

El reporte anterior muestra la siguiente información asociada al Bien:

- Región
- Sucursal
- Fecha de Adjudicación
- Descripción del bien
- Si está asegurado
- Valor de Registro
- Valor de Avalúo
- Fecha de Avalúo
- Listo para la Venta
- Si tiene Fotografía

La ventaja de este reporte es que nos permite seleccionar la información de acuerdo a una serie de condiciones que se muestran en la ventana que, como ya se mencionó, aparece al presionar el icono de condiciones de la barra de herramientas asociada al reporte. La siguiente ilustración *(figura 4.20)* muestra la "ventana de condiciones".

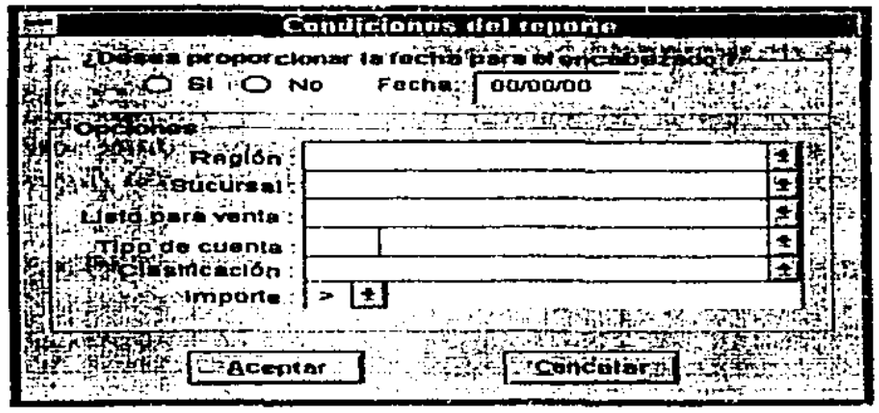


Fig 4.20 Ventana de condiciones para el reporte Relación de bienes de acuerdo a criterios

El objetivo de la ventana anterior *(figura 4.20)* es el de filtrar la información seleccionada, por ejemplo si se selecciona en el campo región la "Region Noreste" y en el campo sucursal la "Sucursal Cd. Victoria", en el reporte sólo aparecerán los bienes que existen en la "región Noreste" y "sucursal Cd. Victoria", el mismo concepto se aplica para los campos Listo para venta, Tipo de cuenta, Clasificación e Importe. Esto significa que, si se desean filtrar solo los bienes de la región Noreste y sucursal Cd. Victoria, que estén listos para la venta, dentro de la cuenta 1601 Bienes Muebles, clasificación automóviles e importe mayor a $30,000.00. El procedimiento a seguir se muestra en la *(figura 4.21)*.

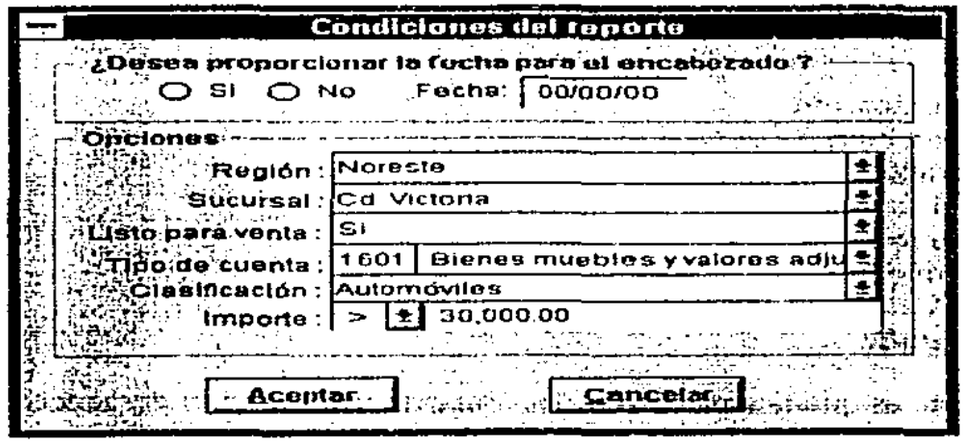


Fig 4.21 Ejemplo de filtrado de información para el reporte Relación de bienes de acuerdo a criterios

Una vez proporcionada la información, se presiona el botón de aceptar y aparece la información seleccionada en el reporte, como se muestra en la siguiente ilustración *(figura 4.22)*.

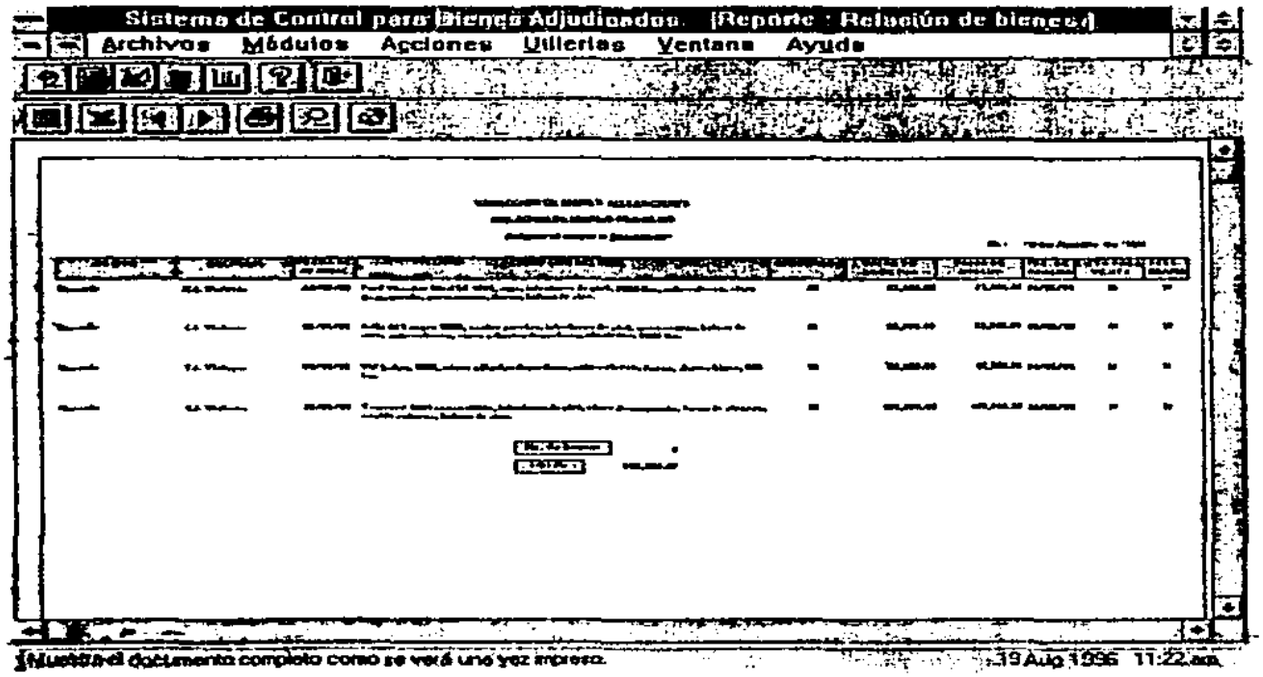


Fig 4.22 Reporte de Relación de Bienes de Acuerdo a Criterios despues del filtro seleccionado.

El reporte se presenta en modo zoom a un 65%, esto con el fin de poder ver el reporte completo, en caso de que el reporte no aparezca en modo zoom, apareceran barras espaciadoras horizontales y verticales para poder desplazarse horizontal y verticalmente a través del reporte.

5.4.2.2.Cuadernillo.

Esta opción de reportes es de gran ayuda, ya que presenta un amplia información acerca del deudor y sus bienes, clasificada por Región y Sucursal. El icono de condiciones permite elegir cuatro tipos de reportes: cuadernillo, balanza, cuadernillo sin observaciones, cuadernillo filiales que se pueden elegir desde la ventana de "condiciones del reporte", la cual se muestra a continuación *(figura 4.23)*.

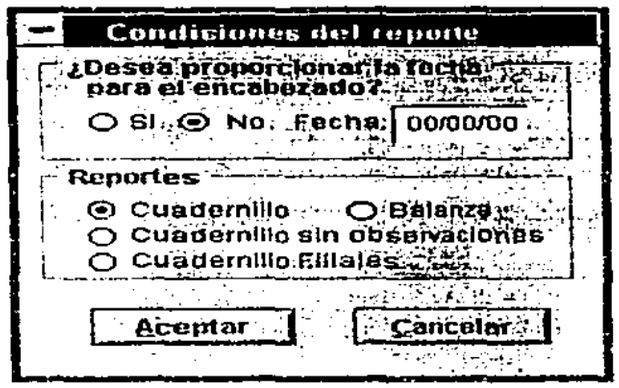


Fig 4.23 Condiciones para el reporte cuadernillo.

Cuadernillo.

Este reporte muestra información del deudor y sus bienes, clasificándola por Región/Sucursal. Muestra para un deudor, su folio, la composición del adeudo; cómo es que se descompone ese adeudo indicando el bien; su importe y a qué cuenta pertenece, así como otra información asociada al deudor y sus bienes.

El reporte cuadernillo muestra la siguiente información asociada al Bien:

- Región.
- Sucursal.
- Nombre del deudor.
- Folio.
- Dación o adjudicación.
- importe del adeudo.
- Importe del adeudo por tipo de cuenta.
- Descripción del bien.
- Fecha de adjudicación.
- Fecha de avalúo.
- Importe de avalúo.
- Listo para la venta.
- Fotografía.
- Observaciones.

La siguiente ilustración *(figura 4.24)* muestra el reporte Cuadernillo.

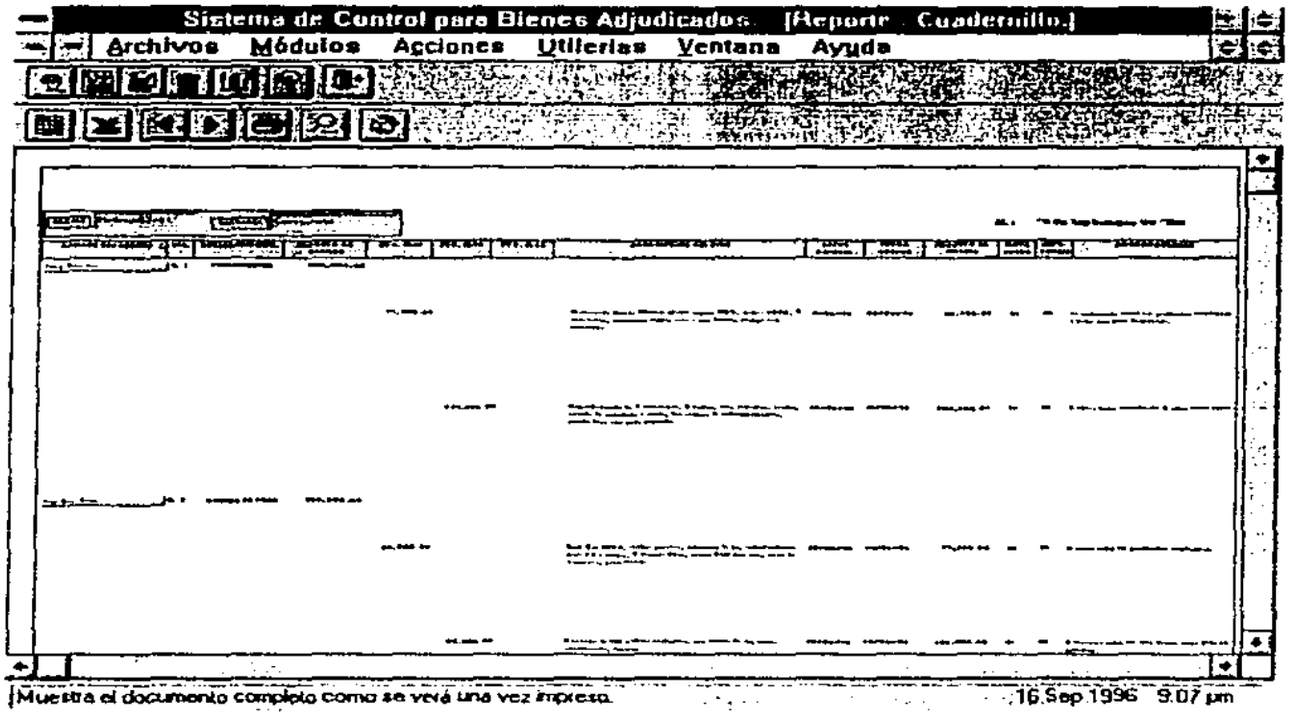


Fig 4.24 Reporte Cuadernillo.

Balanza.
El reporte balanza, muestra para una región, el importe total que existe en cada una de sus sucursales clasificándolo por tipo de cuenta (1601, 1602 y 1603) y obtiene un gran total realizando una suma que globaliza los importes de las cuentas.

El reporte balanza muestra la siguiente información:

- Región
- Sucursal
- Importe por tipo de cuenta
- Importe por el total de las cuentas

La siguiente ilustración *(figura 4.25)* muestra el reporte balanza.

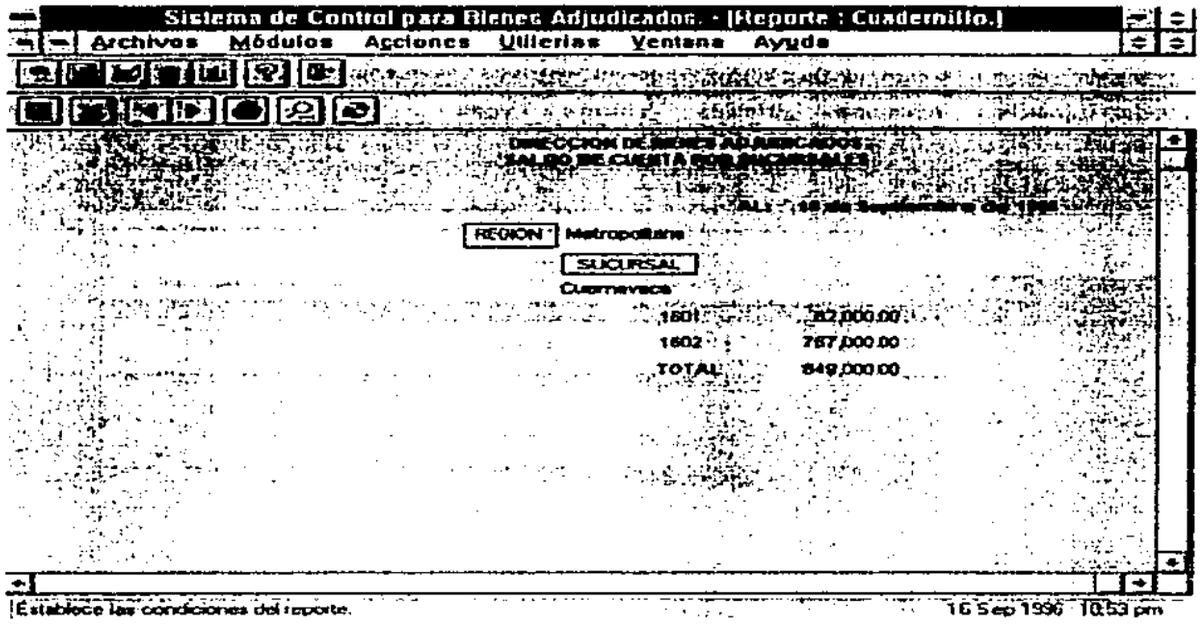


Figura 4.25 Reporte cuadernillo balanza.

Cuadernillo sin observaciones.
Este reporte, como su nombre lo dice, presenta la misma información que el reporte cuadernillo pero sin el campo de observaciones.

Cuadernillo filiales.
Este reporte, muestra la información que contiene el reporte cuadernillo sin observaciones, pero para las filiales.

5.4.2.3.Movimientos contables del mes.

Este reporte permite ver los movimientos contables de un mes determinado, estos pueden ser entradas de bienes (cargos) o ventas de bienes (abonos); clasificándolos por el tipo de bien: muebles (1601), inmuebles (1602) o prometidos en venta (1603). Para solicitar los movimientos realizados en un mes, seleccionamos el icono de condiciones, inmediatamente después aparecerá la ventana de "Condiciones del reporte", en la cual se elige el mes. A continuación se ilustra dicha ventana *(figura 4.26)*.

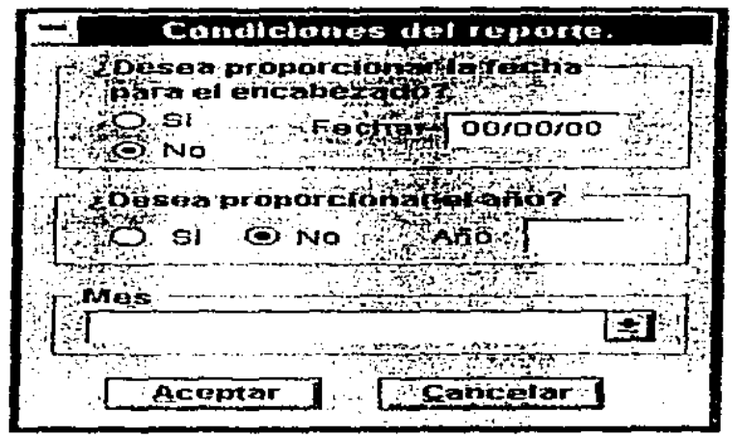


Fig 4.26. Ventana de "Condiciones del reporte" del reporte Movimientos contables del mes.

Una vez elegido el mes, se presiona el botón de aceptar e inmediatamente aparecerá en la pantalla la información con los movimientos que se hayan hecho durante ese mes. Todo tipo de entrada se muestra bajo la columna cargos y cualquier tipo de venta o salida se muestra bajo la columna abonos. La siguiente ilustración *(figura 4.27)* muestra el reporte.

El reporte movimientos contables del mes, muestra la siguiente información:

- Tipo de cuenta
- Nombre del deudor
- Sucursal
- Cargos y abonos.

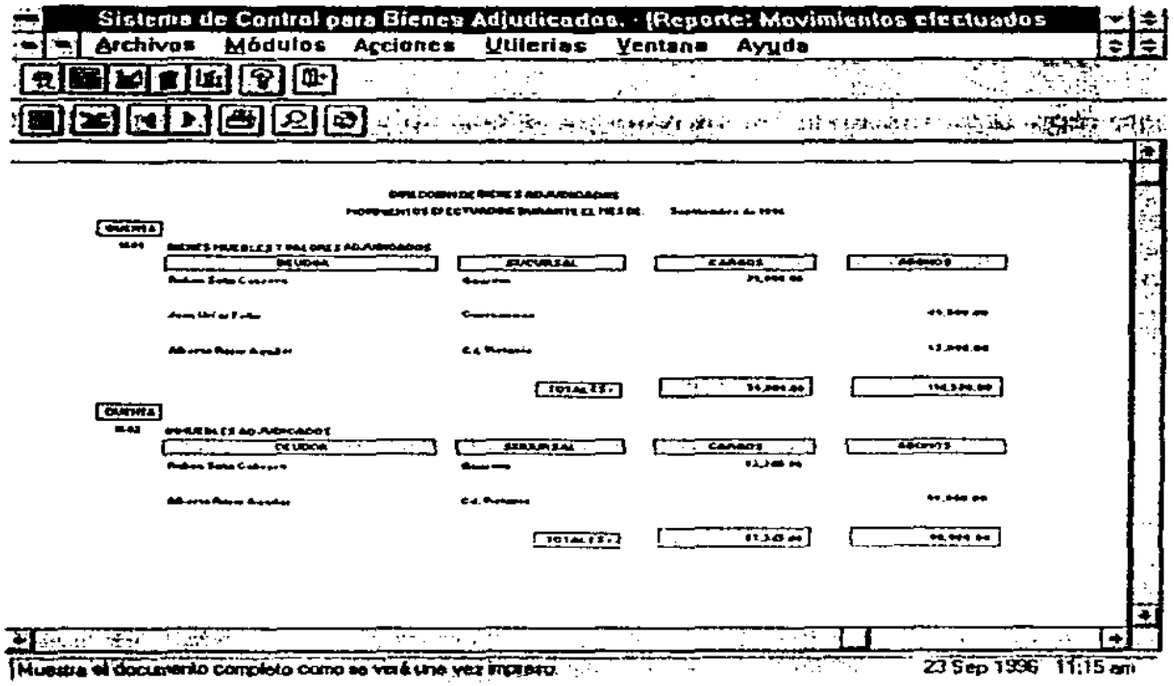


Fig 4.27 Reporte Movimientos Contables del mes.

5.4.3.Mantenimiento a Catálogos

Este módulo fue pensado para dar mantenimiento a todos los catálogos que el sistema utiliza con la finalidad de que el usuario pueda manipular los conceptos que giran en torno al sistema. Para ello se cuenta con una pantalla principal (*figura 4.28*) que es la contenedora de todos los conceptos relacionados

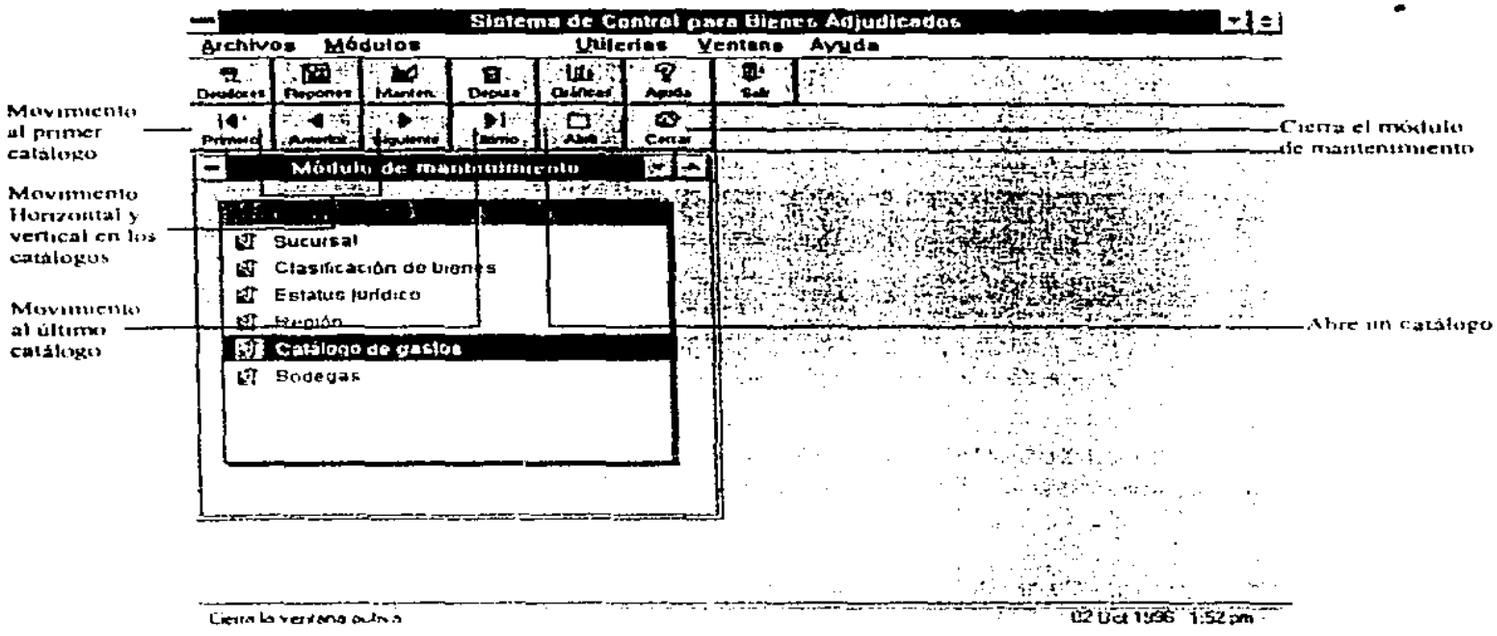


Fig 4.28 Ventana de Mantenimiento

La descripción de la barra de herramientas asociada es la misma que la descrita en dibujos anteriores, que consiste en un browser por cada uno de los elementos de la ventana activa.

Como se observa en la *figura 4.28*, existen 6 conceptos asociados con el sistema, los cuales se describen a continuación:

Catálogo de Regiones

Permite manipular los conceptos de regiones agregando o modificándolos, dando al usuario la facilidad de saber la clave que representa dicho concepto para llevar un control más detallado sobre el bien.

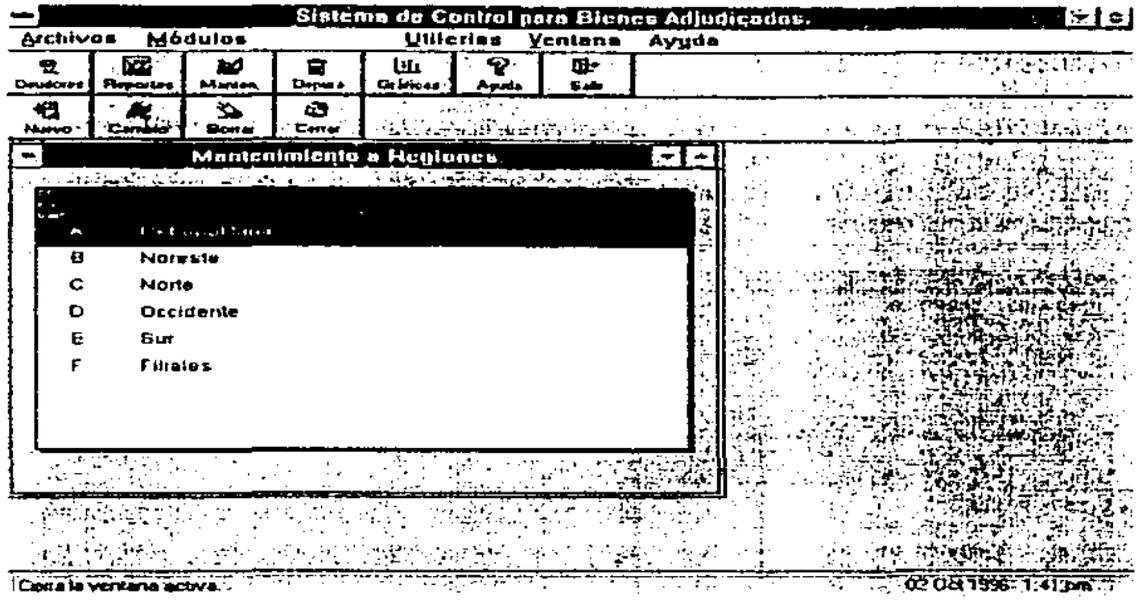


Fig 4.29 Catálogo de Regiones

A continuación se explicarán cada uno de los iconos de la barra de herramientas asociada al mantenimiento del catálogo de Regiones, el cual es extendible a cada uno de los catálogos que se describen posteriormente.

Nuevo.
Este icono permite ingresar un concepto nuevo al sistema

Cambio.
Este icono permite cambiar la descripción del concepto asociado a la clave

Borrar.
Este icono permite dar de baja físicamente el registro de la base de datos

Cerrar.
Este icono permite cerrar la ventana activa.

Al accionar los botones de Nuevo, Borrar y Cambiar se despliegan respectivamente las ventanas que aparecen a continuación:

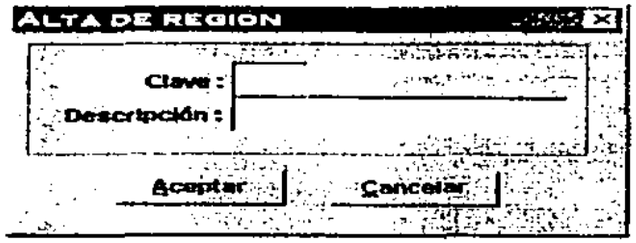


Fig.4.30 Altas

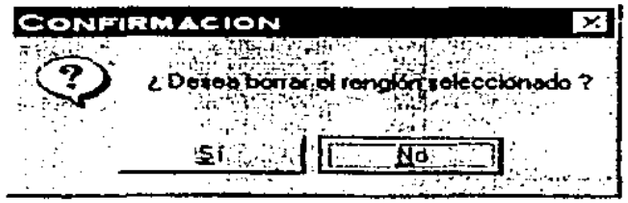


Fig 4.31 Bajas

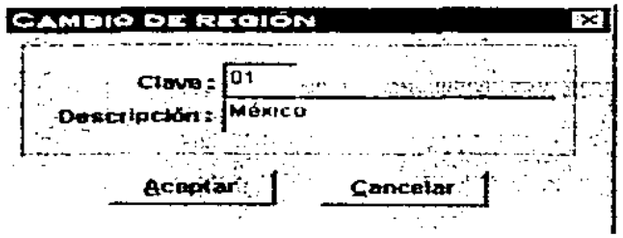


Figura 4.32 Cambios

Las ventanas mostradas nos permiten tener una mayor interactividad con el usuario al hacerle confirmar sobre las acciones que tomará el sistema al elegir una opción.

A continuación se presentan las pantallas para el resto de catálogos, con el objetivo de enseñar la forma en que muestra la información, ya que los conceptos anteriormente explicados, se aplican para estos casos.

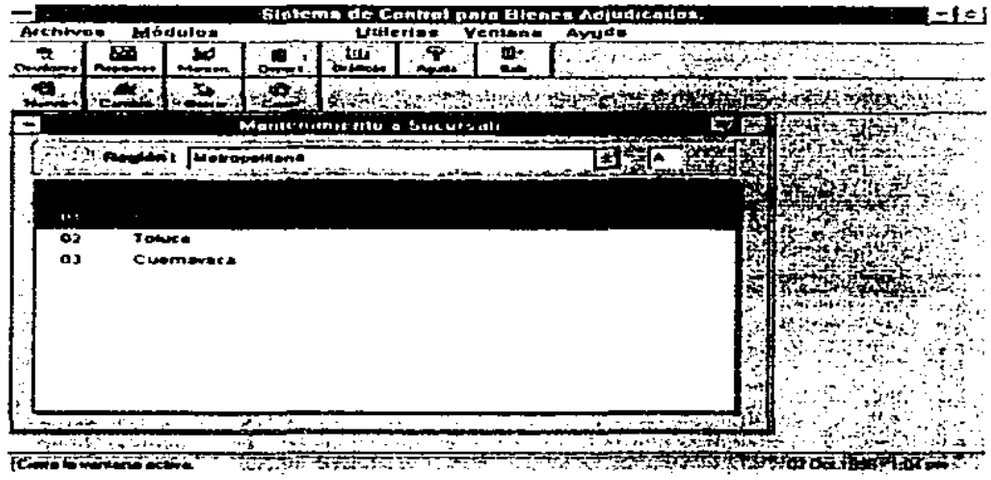


Fig 4.33 Catálogo de Sucursales

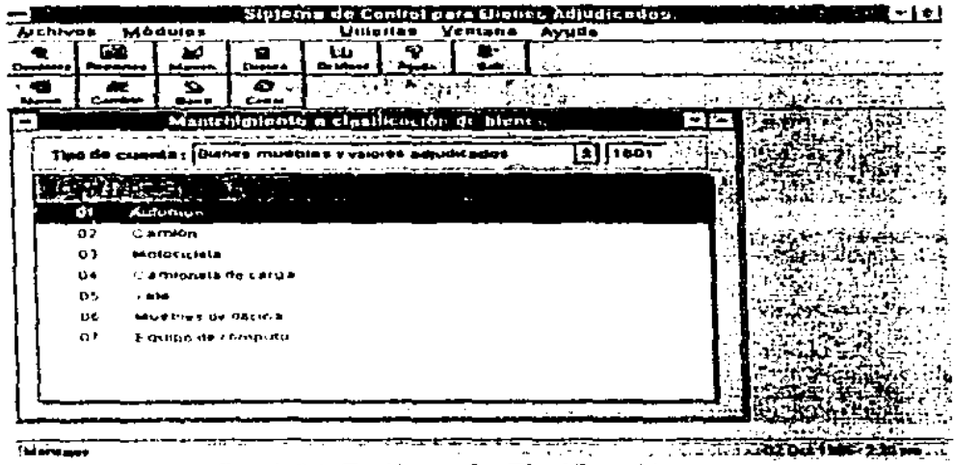


Fig 4 34 Catálogo de Clasificación

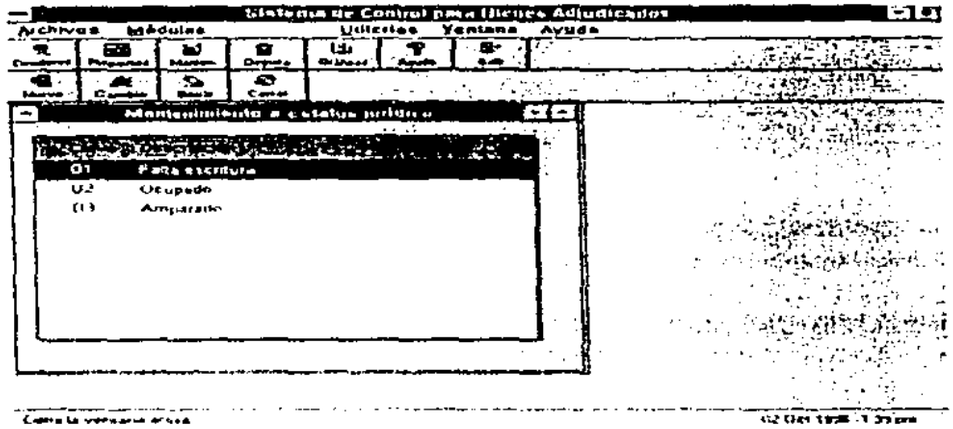


Fig 4.35 Catálogo de Estatus Jurídico

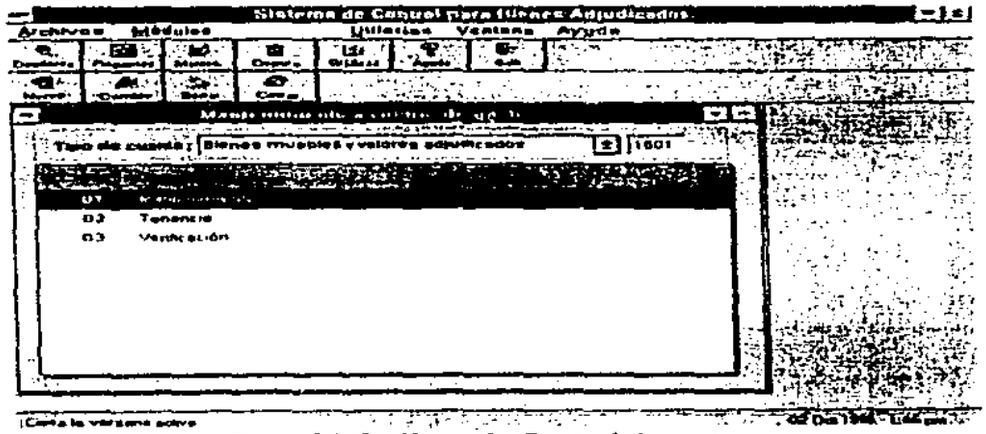


Fig 4.36 Catálogo de Control de gastos

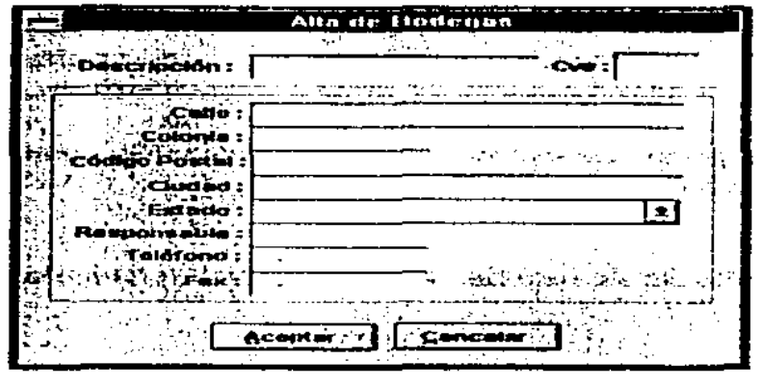


Fig 4.37 Catálogo de Bodegas

5.4.4.Módulo de Depuración de Registros.

Este módulo permite realizar una depuración de aquellos registros que se desean eliminar de la base de datos, no sin antes advertir al usuario sobre las acciones que lleve a cabo.

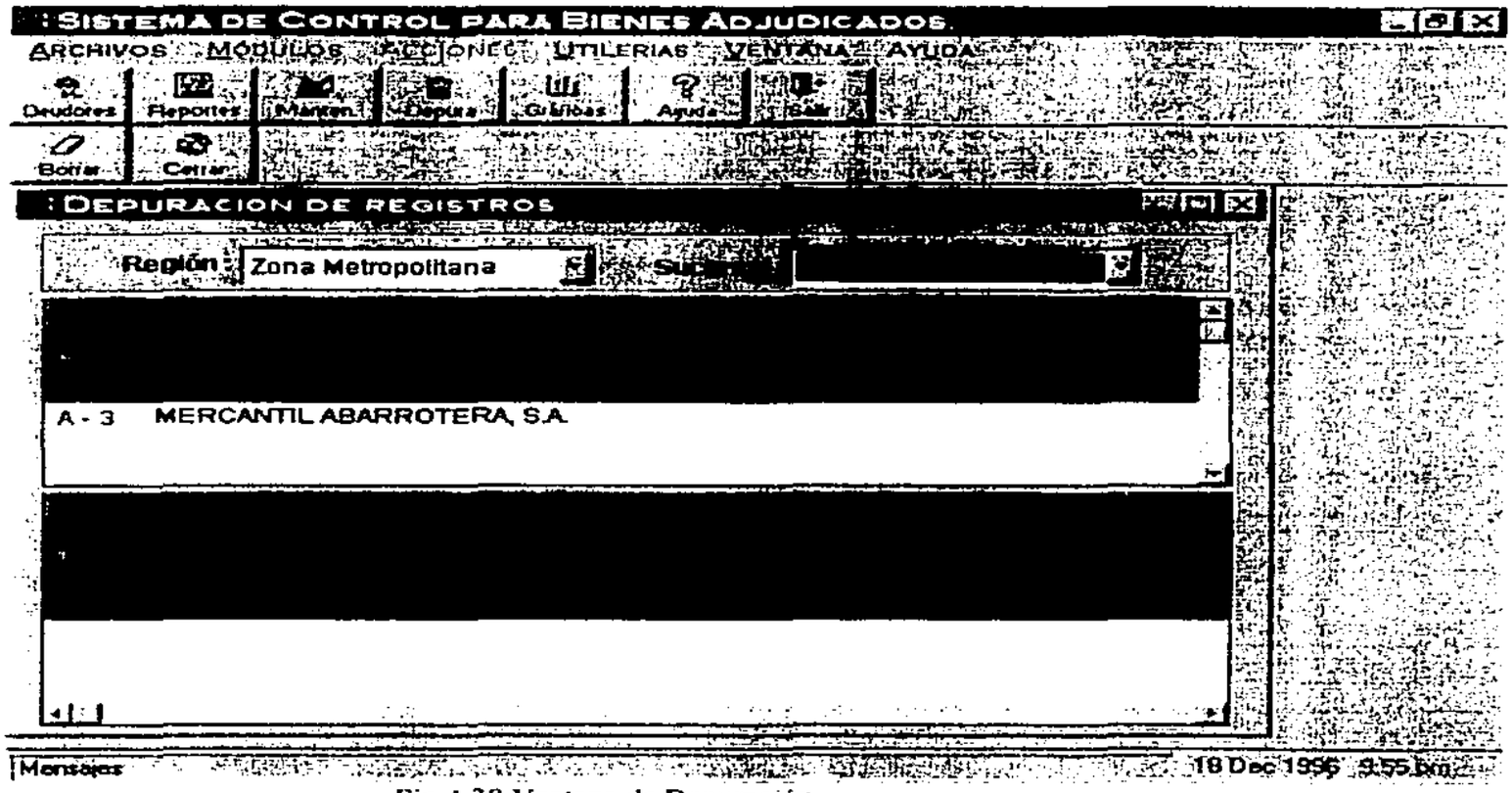


Fig 4.38 Ventana de Depuración.

Eimina el registro de la ventana inferior, que es el resultado de algún movimiento efectuado sobre dicho bien y que el usuario por alguna razón considera borrar de la base de datos. En primer lugar elimina el movimiento efectuado sobre el bien, y si ya no existe ningun renglón seleccionado, el usuario puede inclusive borrar al deudor del sistema. A continuación se describen los botones de la barra de herramientas asociada:

Borrar.
Este icono permite dar de baja fisicamente el registro de la base de datos

Cerrar.
Este icono permite cerrar la ventana activa.

5.4.5.Gráficas

Este módulo es una herramienta auxiliar que permite al usuario observar de una manera gráfica cuál es el comportamiento de las ventas de los bienes en las diferentes regiones y sucursales que posee, permitiéndole elegir los parámetros de consulta. La *figura 4.39* muestra la ventana principal con las opciones que puede realizar.

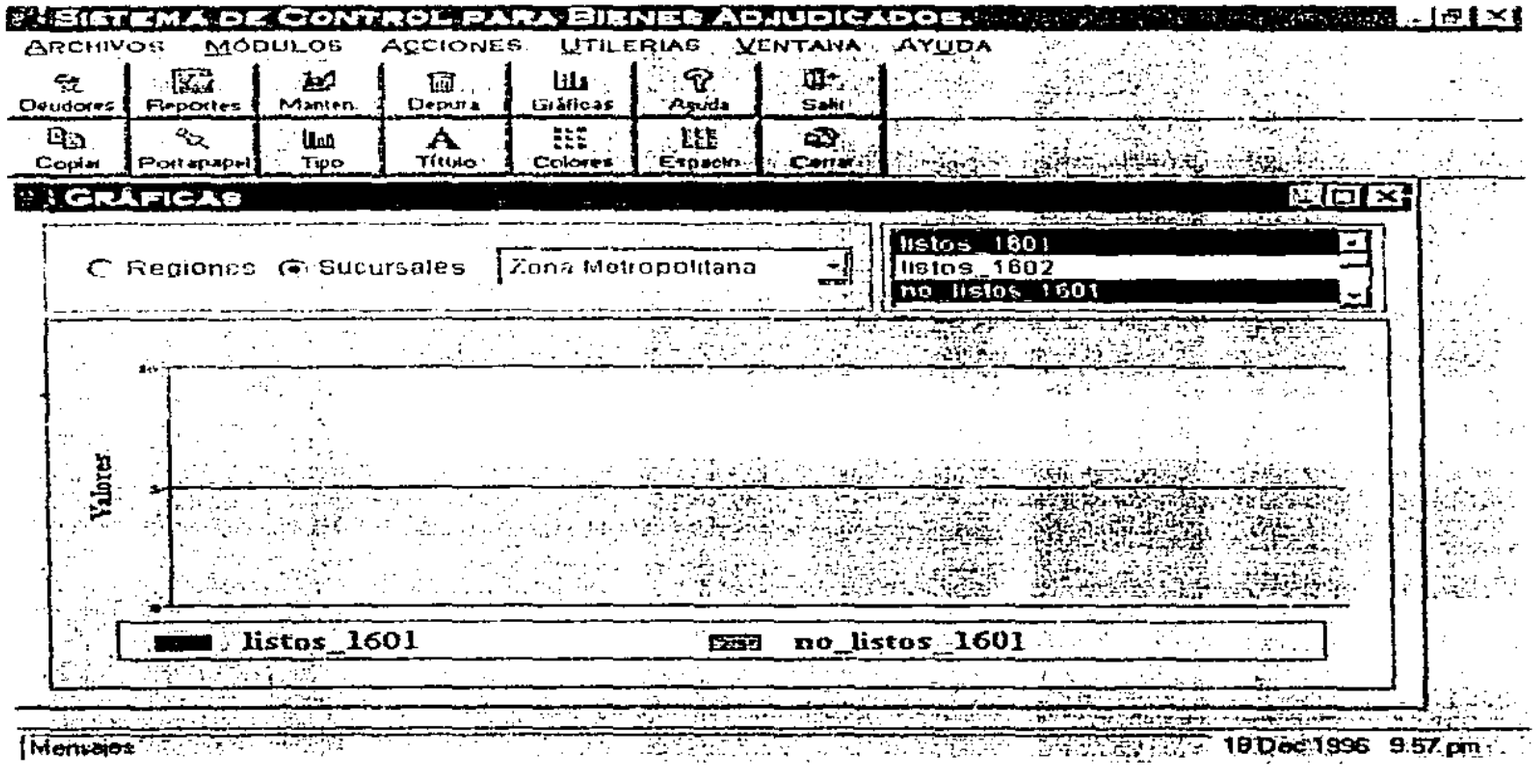
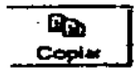


Fig 4.39 Ventana de Gráficas.



Borrar.
Este icono permite copiar el gráfico construido en el portapapeles de *Windows*.

Portapapeles.
Permite mostrar el portapapeles de *Windows*.

Tipo.
Este icono permite seleccionar un estilo de gráfico (fig 4.40)

Título.
Permite colocar un título al gráfico construido.

Colores.
Permite seleccionar una gama de colores en diversos elementos del gráfico (fig 4.41)

Espacio.
Permite seleccionar el espacio entre distintos gráficos

Cerrar.
Este icono permite cerrar la ventana activa.

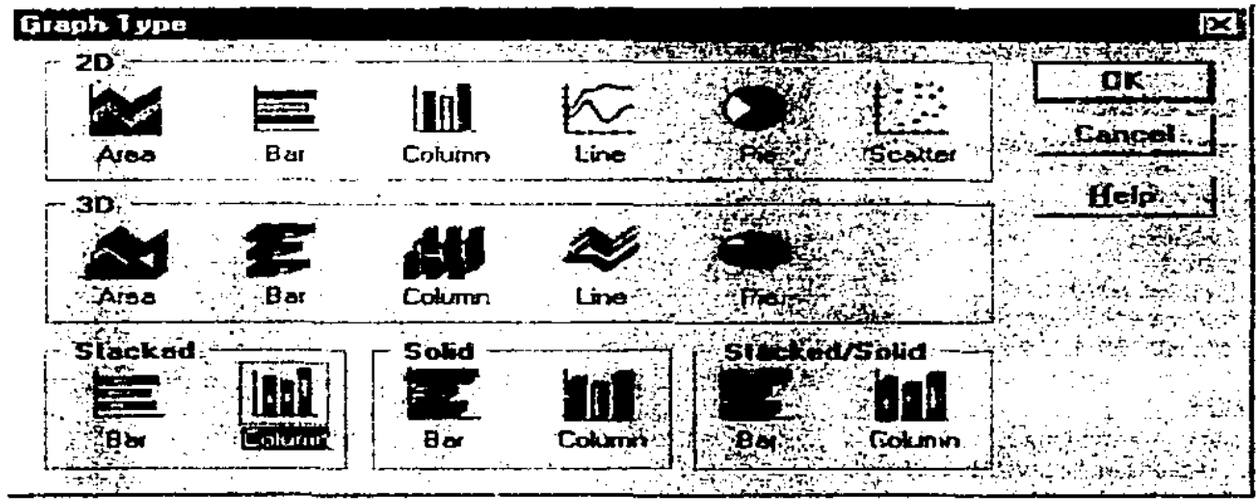


Fig 4.40 Tipos de Gráficas.

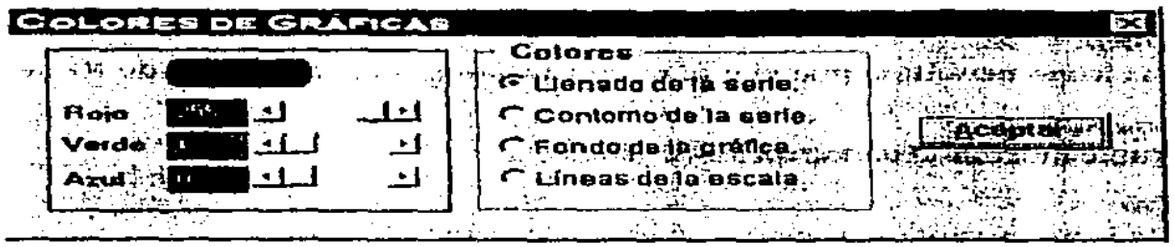


Fig 4.41 Colores de Gráficas.

5.4.6. Ayuda en Línea

A pesar de que no es en sí un módulo, representa una opción que tiene el usuario para saber de manera general la forma de utilizar el sistema, así como entender algunos conceptos que maneja el mismo.

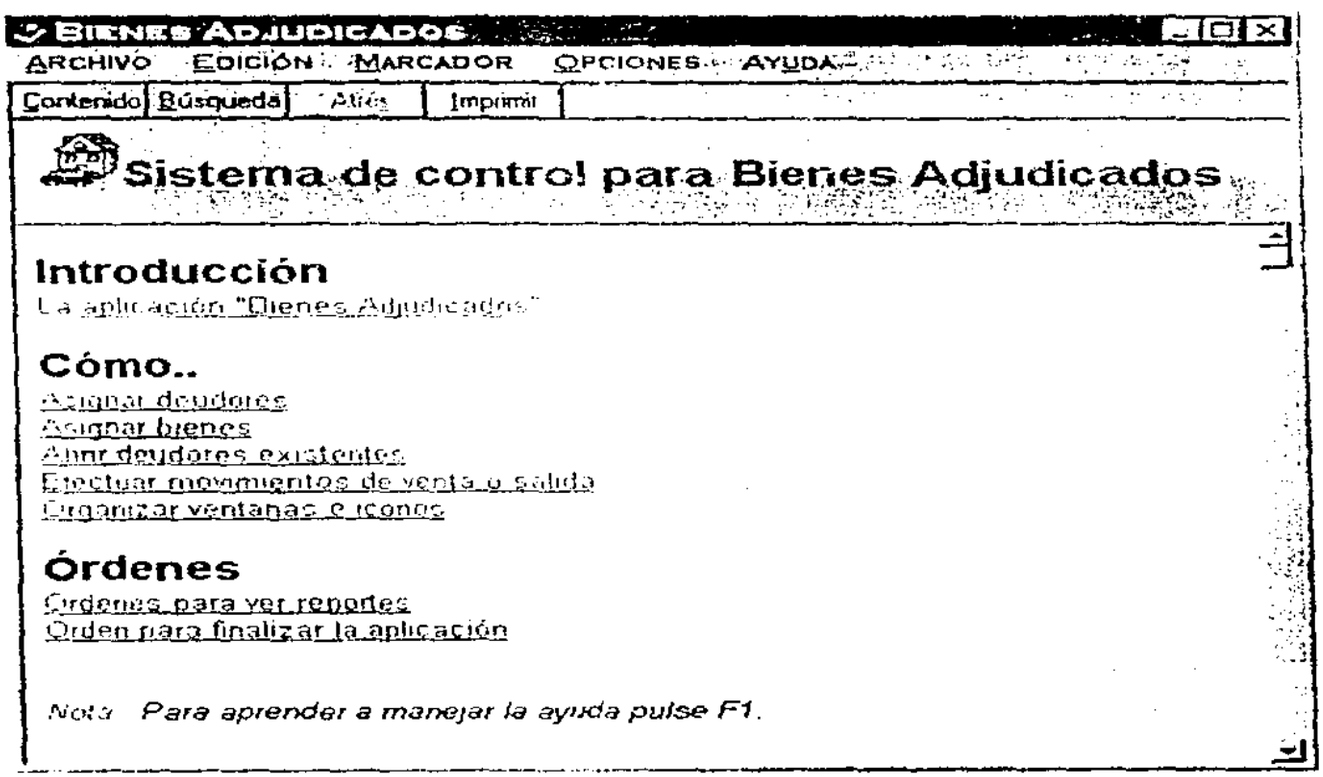


Fig 4.42 Ayuda en Línea

5.5 Pruebas del Sistema.

La prueba del software es un elemento crítico para la garantía de su calidad y consiste en una revisión final de las especificaciones, del diseño y de la codificación. Los errores pueden presentarse desde el primer momento del proceso, en el que los objetivos pueden estar especificados de forma errónea, así como en los posteriores pasos de diseño y desarrollo.

No es raro que una organización de desarrollo de software gaste el 40% del esfuerzo total de un proyecto en la prueba. En ciertos casos, la prueba del software para actividades críticas como el control de tráfico aéreo y control de reactores nucleares puede costar de 3 a 5 veces más que el resto de los pasos del desarrollo de software juntos.

Si la prueba se lleva a cabo con éxito, descubrirá errores en el software. Por otro lado, la prueba demuestra hasta qué punto los procesos y funciones del software parecen funcionales de acuerdo con las especificaciones. Los datos que se van obteniendo a medida que se llevan a cabo las pruebas proporcionan una buena indicación de la fiabilidad y calidad en general del software.

Debido a que la prueba normalmente se lleva hasta el 40% del esfuerzo total de un proyecto de desarrollo de software, las herramientas que reducen el tiempo de prueba son muy valiosas. Como consecuencia de esta situación los investigadores y profesionales han desarrollado una primera generación de herramientas automáticas de prueba.

Objetivos de la prueba.

Los siguientes puntos representan una serie de reglas que pueden interpretarse como objetivos de prueba:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir errores.
- Un buen caso de prueba es aquél que tiene una alta probabilidad de encontrar un error no descubierto hasta ese momento.
- Una prueba tiene éxito si descubre un error no detectado hasta ese momento.

El objetivo principal, es diseñar pruebas que saquen a la luz diferentes tipos de errores con la menor cantidad de tiempo y esfuerzo posible. La prueba no puede asegurar la ausencia de errores, sólo puede demostrar que existen errores en el software.

Una vez que se llevan a cabo las pruebas, se evalúan los resultados, es decir se comparan los resultados de la prueba con los esperados. Cuando se descubren errores inicia el proceso de depuración. El proceso de depuración es la parte más impredecible del proceso de prueba. El diagnóstico y corrección de un error puede llevar una hora, un día, una semana o un mes dependiendo de distintos factores.

Existen errores que son encontrados por los usuarios una vez puesto en producción un sistema. Estos errores son corregidos por profesionales durante la etapa de mantenimiento. El costo de

corrección de un error durante la etapa de mantenimiento puede ser mucho mayor al costo de corrección durante la etapa de desarrollo (entre 60 y 100 veces superior).

Técnicas de diseño de casos de prueba.

Normalmente se usan dos categorías de técnicas para el diseño de casos de prueba:

1. ***Pruebas de la caja blanca.*** Se centran en la estructura de control del programa. Se derivan casos de prueba que aseguren que se han ejecutado por lo menos una vez todas las instrucciones del programa y que se procesan todas las condiciones lógicas. La prueba del camino básico, una técnica de la caja blanca, deriva el conjunto de caminos linealmente independientes que aseguren todos los casos. La prueba de condiciones y del flujo de datos ejercitan adicionalmente la lógica del programa. La prueba de la caja blanca también se describe como "prueba a pequeña escala", ya que son comúnmente aplicadas a pequeños componentes de programas.

2. ***Pruebas de la caja negra.*** Son diseñadas para validar los requisitos funcionales sin tomar en cuenta el funcionamiento interno de un programa. Esta técnica de prueba se centra en el ámbito de información de un programa, de tal forma que se proporcione una cobertura completa de prueba. La prueba de la caja negra se puede definir también como "prueba a gran escala".

Se dice que la prueba nunca termina, simplemente se transfiere de los encargados del desarrollo a los usuarios del sistema. Cada vez que el usuario o cliente usa el programa se lleva a cabo una prueba. Aplicando el diseño de casos de prueba, el ingeniero de sistemas puede conseguir una prueba más completa y descubrir y corregir el mayor número de errores antes de que se ponga en producción y comiencen las "pruebas por parte de los usuarios o clientes".

Estrategia de prueba del software.

Una estrategia de prueba del software integra las técnicas de diseño de casos de prueba en una serie de pasos bien planificados que dan como resultado una correcta construcción del software. Una buena estrategia de prueba del software proporciona también una guía para el desarrollador del software, para el área de control de calidad y para el cliente. Cualquier estrategia de prueba debe incluir la planificación de la prueba, el diseño de casos de prueba, la ejecución de pruebas y la agrupación y evaluación de los datos resultantes.

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 de software.

La prueba es un conjunto de actividades que se pueden planificar por adelantado y llevar a cabo sistemáticamente. En general las estrategias de prueba del software tienen las siguientes características generales:

- La prueba comienza en el nivel de módulo y trabaja "hacia fuera", hacia la integración de todo el sistema.
- Diferentes técnicas de prueba son apropiadas en diferentes momentos.
- La prueba la lleva a cabo el que desarrolla el software y para grandes proyectos normalmente existe un grupo de prueba independiente.
- La prueba y la depuración son actividades diferentes, pero la depuración se puede incluir en cualquier estrategia de prueba.

Una estrategia para la prueba del software debe incluir pruebas de bajo nivel que verifiquen que cada pequeño segmento de código fuente se ha implementado correctamente, así como pruebas de alto nivel para validar las distintas funciones del sistema frente a los requerimientos del cliente.

La prueba, en el contexto de la ingeniería del software, es una serie de tres pasos que se realizan secuencialmente. Inicialmente la prueba se centra en cada módulo individual, asegurando que funcionan correctamente como una unidad, de ahí el nombre de prueba de unidad. A continuación se deben integrar los distintos módulos para formar el software completo, posteriormente se realizan un conjunto de pruebas de alto nivel. Se deben comprobar los criterios de validación definidos durante el análisis de requerimientos. El software una vez validado se debe combinar con otros elementos del sistema (hardware, base de datos, usuarios, etc.). La prueba del sistema verifica que cada elemento encaje de forma adecuada y que se alcanza la funcionalidad y rendimiento del sistema total.

Cada vez que se realiza una prueba de software surge una pregunta clásica: una vez que hemos realizado la prueba, ¿ como sabemos que hemos probado lo suficiente ? Una respuesta a esta pregunta es: la prueba nunca termina, ya que el desarrollador pasa el problema al cliente. Cada vez que el cliente/usuario ejecuta un programa, dicho programa se está probando con un nuevo conjunto de datos.

Prueba de Unidad.

La prueba de unidad, centra el proceso de verificación en la menor unidad del diseño del software, el módulo. Usando como guía la descripción del diseño detallado, se prueban los caminos de control importantes con el fin de descubrir errores dentro del ámbito del módulo.

La prueba de unidad comprende la interfaz del módulo para asegurar que la información fluye de manera correcta hacia y desde la unidad del programa que se está probando. Se examinan las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente conservan su integridad durante todos los pasos de la ejecución del proceso. Se prueban las condiciones límite para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento. Se prueban todos los caminos independientes de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por los menos una vez. Antes de iniciar cualquier otra prueba se debe probar el flujo de datos de la interfaz del módulo. Si los datos no ingresan correctamente, todas las demás pruebas no tienen sentido.

Además de las estructuras de datos locales, en las pruebas de unidad se debe contemplar el impacto de los datos globales sobre el módulo.

Se deben diseñar casos de prueba para detectar errores debidos a cálculos incorrectos, comparaciones incorrectas o flujos de control incorrectos. Los casos de prueba que ejerciten las estructuras de datos, el flujo de control y los valores de los datos por debajo, en y por encima de los máximos y los mínimos son muy apropiados para descubrir errores en las condiciones límites.

El diseño de casos de prueba de unidad comienza una vez que se ha desarrollado, revisado y verificado en su sintaxis el código fuente. Un repaso a la información del diseño proporciona una guía para el establecimiento de los casos de prueba que más probablemente descubrirán errores. Cada caso de prueba debe ir complementado de un conjunto de resultados esperados.

Prueba de Integración.

Podría pensarse que si todos los módulos de un sistema funcionan bien por separado, entonces, ¿por qué dudar de que funcionen bien cuando se integren ? En realidad pueden pasar muchos problemas: Los datos se pueden perder en una interfaz; un módulo puede tener un efecto erróneo sobre otro; las subrutinas cuando se combinan pueden no producir la rutina principal deseada; los márgenes de error aceptados individualmente pueden crecer hasta niveles inaceptables; las estructuras globales de datos pueden presentar problemas, etc.

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

La integración incremental se lleva a cabo de la siguiente manera: El programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir, es más probable que se puedan probar completamente las interfaces y se pueda aplicar un enfoque de prueba sistemática.

La integración descendente es un esquema incremental. En este se integran los módulos de arriba hacia abajo de acuerdo a la jerarquía de control, iniciando con el módulo de control principal. Los módulos subordinados al módulo de control principal se van incorporando a la estructura de acuerdo a su profundidad. La integración ascendente empieza la construcción y la prueba con los módulos de los niveles más bajos de la estructura del software.

Prueba del Sistema.

Finalmente, el software es incorporado a otros elementos del sistema y es necesario realizar una serie de pruebas de integración en relación con estos nuevos elementos (por ej. hardware, comunicaciones, información, etc.). Estas pruebas están fuera del alcance del proceso de ingeniería del software y no son conducidas únicamente por el encargado del desarrollo del software. De cualquier manera, los pasos dados durante el diseño del software y durante el

proceso de prueba pueden incrementar considerablemente la probabilidad de éxito en la integración del software en el sistema total. La prueba del sistema está constituida por una serie de pruebas diferentes cuyo propósito es ejercitar profundamente todo el sistema, todas trabajan para verificar que se han integrado correctamente todos los elementos del sistema y que realizan las funciones en forma adecuada.

El Proceso de Depuración.

La depuración aparece como consecuencia de una prueba efectiva. Cuando un caso de prueba descubre un error, la depuración es el proceso que consiste en la eliminación del error. El proceso de depuración comienza con la ejecución de un caso de prueba, se evalúan los resultados y aparece una falta de correspondencia entre los esperados y los reales. El proceso de depuración intenta hacer corresponder el error con una causa, llevando así, a su corrección.

Durante el proceso de depuración, pueden encontrarse errores que van desde lo ligeramente inesperado hasta lo catastrófico. A medida que las consecuencias de un error aumentan, crece la presión para encontrar su causa. Con frecuencia la presión provoca que un ingeniero de software, al intentar arreglar un error, genere dos más.

Es muy difícil asignar un tiempo al proceso de depuración de errores. Corregir un error depende de distintos factores. Se puede detectar en un módulo y haberse generado en otro, puede estar muy oculto, etc.

Pruebas al Sistema de Administración de Bienes Adjudicados.

Los distintos módulos del sistema de Administración de Bienes Adjudicados fueron probados de manera unitaria. Se diseñaron casos de prueba, datos de prueba y resultados esperados. Durante la realización de las pruebas se compararon los resultados arrojados con los resultados esperados. Cuando estos no coincidían significaba que se había detectado un error y posteriormente se corregía. Estas pruebas unitarias fueron realizadas por el mismo programador. Fueron basadas en gran parte en los diagramas de estado especificados en el Tema III, y en general en las especificaciones del diseño. Se asignó un tiempo para pruebas y depuración siendo responsabilidad del programador este primer nivel de calidad del software. En este primer nivel de pruebas el programador pretendió garantizar por sí mismo la calidad de cada uno de los módulos que construyó.

Este sistema fue desarrollado en relación muy estrecha con el área usuaria. Esto permitió que se involucrara en el proceso de prueba y se validara desde este momento las funciones del sistema con respecto a las especificaciones del diseño. Esto representó una gran ventaja para la calidad de la funcionalidad del sistema y pudo prever la ocurrencia de cierto número de errores al momento de la puesta en producción, aunque no evitarlos en su totalidad. En muchos desarrollos se comete el error de no asignar tiempo suficiente para la prueba de los módulos, esto normalmente se paga mucho más caro una vez puesto en producción el sistema.

Una vez que se aprobaron en forma independiente cada uno de los módulos, se iniciaron las pruebas de integración. Se tomaron como base las especificaciones del diseño y el diagrama

de flujo de datos entre los distintos módulo del sistema. Se probaron las distintas entradas y salidas de los módulos. Los datos ingresados o actualizados en un módulo debian leerse correctamente por otros módulos. Se probó el paso de parámetros o "mensajes" entre los módulos. En términos generales se validó la correcta interacción entre los distintos módulos del sistema.

Las pruebas de integración fueron realizadas por los distintos programadores de los módulos. Además en esta prueba se integró a un técnico de calidad para la revisión de estándares, interfaz de usuario, reportes, etc, de tal forma, de contar con un enfoque de prueba diferente al del programador. Una vez que los programadores probaron integralmente los distintos módulos del sistema y depuraron los errores encontrados, se procedió a involucrar a los futuros usuarios para que llevaran a cabo un nuevo conjunto de pruebas a todo el sistema, orientados a obtener su aprobación para cada uno de los módulos.

Este conjunto de pruebas efectuadas por los usuarios lograron su aprobación para la instalación y puesta en producción del sistema.

5.6 Instalación del Sistema.

El Sistema de Administración de Bienes Adjudicados se instaló bajo una arquitectura cliente/servidor de dos capas. La base de datos en el "servidor" y el software de la aplicación en cada uno de los "clientes". Se inició la producción del nuevo sistema mediante un "paralelo". Esto significa que la información se procesó tanto en el nuevo sistema como en el sistema o método tradicional que ya existía, con la intención de comparar e igualar los resultados.

Durante el uso en "paralelo" del nuevo sistema se encontraron errores que no se habian detectado durante las pruebas unitarias e integrales. Estos errores fueron atendidos por los programadores en forma inmediata, iniciándose un nuevo ciclo de depuración de los distintos módulos del sistema.

La ejecución del nuevo sistema en paralelo tuvo duración de un mes. Después de este período y como consecuencia de resultados satisfactorios, se decidió eliminar el uso del sistema tradicional, quedando en producción solamente el nuevo Sistema de Administración de Bienes Adjudicados.

A partir de este momento cualquier cambio que se efectúe sobre los distintos módulos del sistema, ya sean correcciones, adaptaciones, mejoras, etc., formarán parte de la etapa de mantenimiento del sistema.

6. MANTENIMIENTO DEL SISTEMA

6.1. Introducción.

Antecedentes.

Cuando nos referimos al término "mantenimiento", la mayoría de las personas lo asociamos con tareas tediosas y aburridas, aunque muchas veces esto es cierto, también ocurre en el caso de mantenimiento de software, que se hace muy necesario y a la vez importante, tomar en cuenta una serie de consideraciones al momento de desarrollar un proyecto de software.

Se dice que el mantenimiento del software existente puede llevarse hasta el 70% de todo el esfuerzo gastado por una organización de desarrollo. A mayor producción de software, mayor es el gasto en esfuerzo, y que por lo tanto, no será posible producir nuevo software mientras los recursos disponibles se gasten en el mantenimiento del antiguo software.

Por otro lado, existe una condición que prevalece como una característica interna a cualquier proyecto de ingeniería de software: "La naturaleza omnipresente del cambio". El cambio es inevitable en la construcción de sistemas, por ello, debemos desarrollar mecanismos de evaluación, control e implementación de modificaciones.

Definición de los distintos tipos de mantenimiento del software.

Cuando nos referimos al mantenimiento, y a las tareas que involucra, tal vez algún programador experimentado podría protestar diciendo "... pero yo no empleo el 60% de mi tiempo solucionando errores del programa de desarrollo!". Pero en realidad, el mantenimiento de software es mucho más que una corrección de errores.

Podemos distinguir que existen varios tipos de mantenimiento, dependiendo de la naturaleza de las actividades que se llevan a cabo tras distribuir una aplicación. A continuación presentamos una breve descripción de lo que se refiere cada uno de ellos.

1. ***Preventivo***: Ocurre aunque no muy frecuentemente, cuando se plantea alguna solución que desde un principio, se puede considerar que no es la única alternativa, o que conforme se va avanzando en el desarrollo de un proyecto, se aprenden cosas nuevas y se descubre que tampoco es necesariamente la mejor (aquí aplicaría la frase: "Todo es susceptible de mejorarse"). Sobre todo llega a suceder que por cuestiones de premura en los tiempos de entrega, desafortunadamente no se tiene oportunidad de implementar

esa nueva opción, pero se puede apuntar como una oportunidad de hacerlo en una fase posterior de mantenimiento.

2. ***Correctivo***: Este tipo de mantenimiento se debe a que no es razonable asumir que en la fase de prueba del software se hayan descubierto todos los errores latentes en un gran sistema de software. Hasta cierto punto es normal que surjan errores durante el uso de cualquier programa. Cuando se realiza un proceso de diagnóstico y corrección de este tipo de errores se le llama mantenimiento correctivo.

3. ***Adaptativo***: Hablábamos de la naturaleza tan cambiante que se da en un proyecto de desarrollo. Cuando se anuncian nuevas tecnologías de hardware en periodos relativamente cortos (24 meses aproximadamente), cuando aparecen en el mercado nuevos sistemas operativos o nuevas versiones de los anteriores, o incluso cuando se hacen necesarios cambios de ubicación de oficinas, bodegas, mostradores de atención a clientes, etc. Si estimamos un tiempo de vida útil de un software de aplicación de unos 10 años, entonces el sistema está obligado a adaptarse a los cambios de ambiente y sobrevivir al nuevo entorno de operación requerido. Es entonces cuando se requiere el mantenimiento de tipo adaptativo que normalmente es tan necesario como usual.

4. ***Perfectivo***: Cuando un producto de software es aceptado en el medio en forma generalizada, se dice entonces que ha pasado la "prueba de fuego", aquí es cuando sucede que los usuarios, normalmente ya expertos en el manejo del producto, comienzan a proponer recomendaciones sobre las mejoras que se le pueden aplicar al sistema, ya sea en la optimización de procesos o en la redefinición de funciones para un mejor desempeño tanto de las actividades del usuario, como del rendimiento y aportaciones del programa en general. Para satisfacer estas peticiones, se lleva a cabo el mantenimiento perfectivo, este tipo de actividad normalmente representa la mayor cantidad de esfuerzo empleado en el mantenimiento de software.

5. ***Aumentativo***: Este tipo de mantenimiento sucede cuando se hace posible incorporar nuevas funciones y/o procesos al sistema, los cuales no existían en la versión anterior del programa y lo que es más, muchas veces ni siquiera estaban contemplados durante la fase de análisis y diseño del proyecto, y sin embargo, cuando se plantean, es muy frecuente que se soliciten estos cambios en calidad de "indispensables".

Cabe señalar que en realidad las tareas que se dan como parte del mantenimiento adaptativo o perfectivo son las mismas que se aplican durante la fase de desarrollo del proceso de ingeniería de software. Es decir, se deben dimensionar los nuevos requisitos, rediseñar, generar código y efectuar pruebas, para poder adaptar o perfeccionar el software.

Características del mantenimiento.

Sobre el mantenimiento estructurado frente al no estructurado:

Cuando un usuario hace una petición de mantenimiento, y sólo se dispone del código fuente como único elemento de configuración, se hace muy difícil poder evaluar el cambio, y se complica todavía más si cuenta con una pobre documentación del proyecto. Las estructuras de datos, la mecánica de operación de las interfaces de usuario y el rendimiento del sistema son difíciles de descubrir y frecuentemente mal interpretadas, además de no poder calcular con exactitud el alcance de los cambios realizados finalmente sobre el código. Esto es lo que sucede cuando nos vemos obligados a darle mantenimiento del tipo no estructurado a un sistema. Con las consecuentes frustraciones que esta situación implica por no haber desarrollado el software mediante una metodología bien definida.

En cambio, cuando se cuenta con la documentación adecuada del diseño, y se pueden dimensionar las características estructurales y de rendimiento y de interfaz del software, y a su vez, esto permite hacer una evaluación del impacto de las correcciones y se logra trazar un plan de actuación, entonces podemos implementar un mantenimiento de tipo estructurado sobre el sistema, al cual es posible llegar mediante la anterior aplicación de una metodología de ingeniería de software. Aunque esto no garantiza un mantenimiento libre de problemas, sí reduce considerablemente la cantidad de esfuerzo requerido y mejora la calidad general del cambio.

Costos del mantenimiento.

Aunque al hablar de costos de mantenimiento, normalmente lo asociamos a un gasto en dinero, existen otros tipos de costos, menos tangibles, pero sí igual de perjudiciales. Como por ejemplo los retrasos en el desarrollo de nuevas aplicaciones, y el alargamiento en tiempo para estabilizar un sistema, son otra clase de situaciones relacionadas y que también implican un costo. Y existen más todavía:

- La insatisfacción del cliente, cuando hace alguna solicitud que no se pueda atender en un tiempo razonable.
- La disminución en la calidad global del software debido a los errores latentes que se introducen en los cambios al software mantenido
- La baja eficiencia en el esfuerzo entregado a las tareas de mantenimiento, etc.

Problemas asociados con el mantenimiento

La mayoría de los problemas asociados al mantenimiento se deben a las deficiencias de la forma en que el software ha sido definido y desarrollado, es decir, a la falta de control y disciplina en las actividades de desarrollo del proceso de ingeniería del software. En general se presentan los siguientes problemas:

- Dificultad para seguir la evolución del software a través de varias versiones. Cambios no documentados.
- Dificultad para seguir el proceso de construcción del software.
- Dificultad para la comprensión de programas "ajenos" y que muchas veces no se tiene disponible a la persona, para explicar lo que hizo.
- La mayoría del software no ha sido diseñado previendo el cambio.
- El mantenimiento no es visto como un trabajo atractivo, debido al alto nivel de frustración asociado con la tarea, demasiado esfuerzo y a veces muy pocos resultados.

Tareas de mantenimiento.

Aunque es muy importante para una área o empresa dedicada al desarrollo, disponer de elementos humanos y de control para llevar a cabo tareas de mantenimiento, este equipo no necesariamente debe de constituirse formalmente. Basta con que se establezca un plan adecuado de seguimiento a las tareas y que se traten de apegar lo más posible a dicho plan, el cual consistiría básicamente en lo siguiente:

- Generar un informe más o menos detallado cuando se realice alguna solicitud de mantenimiento.
- Determinar la acción de pasos a seguir y la prioridad de la solicitud.
- Hacer el registro de información de las actividades relacionadas con las modificaciones
- Evaluar si se lograron los resultados esperados y no haber provocado inestabilidad en la operación del sistema.

6.2. Mantenimiento del software.

De acuerdo a la metodología aplicada a nuestro proyecto, ha sido muy importante no descuidar este concepto como una característica intrínseca del mismo, y que en la medida en que se logren adherir lo más posible todas estas consideraciones de mantenimiento, a las diferentes fases del proyecto, lograremos llegar a la fase de estabilización del sistema a corto plazo y con menor esfuerzo.

Sin embargo, hemos considerado que para nuestro proyecto, la fase de mantenimiento se da con la liberación propia del producto, más que con los procesos constituidos en las etapas de desarrollo. Quizá la teoría de mantenimiento orientado al proceso de ingeniería aplica más bien al desarrollo por prototipos, por lo que en realidad extenderemos la sección 3, correspondiente al mantenimiento después de liberado el producto, en donde ya se toman en consideración todos los demás factores que intervienen en la implantación de un sistema de software: equipos, dispositivos de conectividad, configuraciones, cableado, etc.

Cabe destacar que para nuestro proyecto en particular, y para la ingeniería de software en general, consideramos que el concepto de sistema de software necesariamente debe involucrar a todos estos factores debido a que coexisten en un ambiente la mayoría de las veces heterogéneo, de forma tal que dependen mutuamente unos de otros.

6.3. Mantenimiento orientado a la tecnología Cliente-Servidor.

Condiciones de operación actuales.

- **Restricciones.** Una de las ventajas que presenta este tipo de tecnología, es el hecho de que el sistema puede iniciar operaciones incluso en modo monousuario y que para nuestro cliente ésta es la restricción.

  Hasta el momento, para nuestro cliente ha sido suficiente trabajar en este esquema, dado que es el de más fácil implantación y el más económico, además de permitir que la migración a otro tipo de arquitectura sea transparente para la aplicación. Por ello, se tiene el sistema funcionando en modo monousuario y además es controlado por un solo operador.

- **Coordinación con otras áreas.** Cuando nos referimos al mantenimiento de sistema, generalmente implica cambios, incluso de los procedimientos internos y hasta de la manera de organizar el trabajo de las personas, lo cual repercute en una cierta falta de control en cuanto a flujo de información que genera un nuevo sistema, por ello, es importante mencionar que parte de las tareas de mantenimiento también están orientadas a reubicar al usuario en términos de lo que ahora se puede hacer con la información que arroja el sistema y la manera de optimizar las actividades de intercambio de información con otras áreas y lo útil que ésta puede ser en un momento dado para esas áreas.

Capacidad de crecimiento y adaptación a diferentes esquemas de operación.

Con este tema, comenzamos a darle forma a algunos aspectos interesantes no sólo de nuestro programa, sino de aplicaciones cliente/servidor en general.

- **Ambiente centralizado.** Dado el contexto inicial de operaciones de nuestro cliente, dentro del ambiente centralizado, tenemos dos modos de operación a considerar, uno es el esquema actual y otro es al que se puede crecer, tal vez con ciertos cambios pero que son más que nada de adaptación y que depende de muchos factores el nivel de afectación al sistema.

  En el primer caso, hablamos de un sólo usuario que se hace cargo de la captura de información y gestión de la misma, para su interacción tanto con los agentes de venta de bienes, las áreas de contabilidad y de pagos, incluso de su contacto con el sistema maestro de clientes, sin olvidar la dependencia que todas las sucursales a nivel nacional tiene de este sistema. Bajo este esquema, toda la información que el operador recibe es vía fax y es el responsable directo del registro y actualización de la información.

  El otro esquema también opera bajo un ambiente centralizado, pero con la característica de que la aplicación, haciendo uso del concepto "portabilidad", la podemos reubicar en otra localidad física, para conectarse a través de la red amplia que se tenga disponible y seguir trabajando con el sistema centralizado pero con posibilidad de trabajar en modo multiusuario. Podemos hablar incluso de conexiones vía módem, con las cuales no es requisito contar con una infraestructura de red para su operación y que sin embargo es posible incrementar la explotación del sistema a nivel nacional, que a final de cuentas, el objetivo es otorgar un mayor nivel de productividad al usuario, a través del uso de sistemas informáticos.

  En este sentido, el mantenimiento al producto está orientado al uso de otro tipo de recursos de comunicación, para poder establecer su conexión con la base de datos, pero otro aspecto importante es que ahora se deberá procurar una fase de optimización y tal vez hasta de cambios a la mecánica de intercambio de información entre la computadora y el hombre, ya que muchas de las formas de acceder a los datos, se hacen totalmente diferentes cuando se acceden en red que cuando se acceden por módem.

- **Ambiente distribuido.** Para llegar a este nivel de operación, ya es necesario contar en la localidad remota con un equipo de características similares tanto en hardware y software al del esquema centralizado, para que en la sucursal nos permita por medio del mismo sistema llevar el control de bienes a nivel local o regional, y a su vez, se implementen como tareas de mantenimiento, ciertas funciones que permitan replicar información a nivel central. En este sentido la programación adicional relacionada a la replicación se realiza a nivel de la base de datos, no de la aplicación.

  En un momento dado, se podría considerar que con una replicación asíncrona bastaría para alimentar los movimientos en la base de datos centralizada, sin embargo, dada la

posibilidad de que ahora el usuario de la sucursal ya tendría oportunidad de realizar la venta de un bien en forma local, en vez de estar solicitando autorizaciones y trámites a nivel central, entonces se convierte en una condición crítica el hecho de poder actualizar movimientos en tiempo real, para mantener al día todas las operaciones por lo que la programación de replicación deberá ser síncrona y permitir lo siguiente:

a) Que la sucursal pueda realizar operaciones de venta de sus bienes locales y actualizar a la base de datos a nivel central sobre sus movimientos.
b) Que la base de datos central considere los movimientos en sucursales y tenga oportunidad de vender hacia cualesquiera de los bienes registrados en su sistema, no importando si la venta es local o foránea.

Podemos considerar como parte del mantenimiento, la conversión de la aplicación con soporte a conexión remota, esto es, que fuera de la red de operación del sistema, exista la posibilidad de que desde otras localidades, también sea posible trabajar con la aplicación, haciendo un enlace telefónico y explotando las ventajas que este tipo de tecnología representa.

- **Características del mantenimiento sobre este tipo de tecnología.** Cuando hablamos de mantenimiento a un sistema que funciona bajo la arquitectura cliente/servidor, existen ciertas consideraciones básicas , que son:
  1. Las modificaciones asociadas al propio código.
  2. La interfaz de usuario, la cual debe cumplir con las siguientes características:
  3. Ser lo suficientemente intuitiva y consistente como para que el usuario del sistema no tenga que depender de un complicado esquema de capacitación para poder dominarlo.
  4. Ser capaz de otorgar un rendimiento adecuado en términos de tiempos de respuesta, sin exagerar en la demanda de procesamiento ni por parte del cliente, ni por parte del servidor de base de datos. Esto debido a que ya en la práctica, si bien es cierto que aunque este tipo de tecnología ofrece muchas ventajas a futuro, también hay que reconocer que su implementación es cara. Por lo que se deberán de optimizar los recursos para poder acercarse, en la medida de lo posible a:

     a) Máquinas cliente lo suficientemente poderosas para soportar una interfaz gráfica y cierto nivel de procesamiento local
     b) Mediana capacidad en disco y memoria RAM
     c) Eficiencia en el tráfico de red, concentradores, switches, cableado estructural, etc.
     d) Ambiente controlado para soportar TCP/IP
     e) Servidor de Base de Datos de alto rendimiento

     Es en este sentido que debido al alto costo que representa incorporar esta tecnología, no podemos damos toda la libertad de contar con unos super equipos para cumplir con el compromiso de respuesta ideal del sistema, sobre todo en aplicaciones de misión crítica o con un alto nivel de contingencia.

- **El esquema de comunicaciones**. Aunque TCP/IP es uno de los más eficientes protocolos de comunicación existentes hoy en día, no deja de ser importante la eficiencia en el diseño físico de la red y las consideraciones de crecimiento futuro, las cuales sino se tienen debidamente contempladas, después será más difícil poder soportar ese crecimiento: los nodos disponibles en el edificio, puertos en los concentradores, adición de concentradores en cascada, direcciones IP, etc.

Es importante considerar la rapidez de crecimiento horizontal del cliente pues muchas veces detalles tan simples como la asignación de direcciones IP en caso de no ser automática a través de un servidor DHCP, por ejemplo un error en la asignación de IP a las máquinas cliente puede repercutir en una falla de considerables consecuencias, sobre todo si se duplican direcciones, y algunas de éstas llegan a ser las de los servidores de bases de datos.

Incorporación de objetos de consulta de imágenes.

Adicionalmente, debido al ambiente gráfico en el que corre la aplicación y a las características de la propia herramienta de desarrollo, es posible incorporar características más atractivas para el usuario, a modo de darle mayor fuerza al proceso de venta de los bienes en consignación, pues a final de cuentas una vez que se adjudica un bien, el objetivo primordial es venderlo y en ese sentido se hace importante administrar adecuadamente su mantenimiento en tanto se libera el bien.

Dentro de los planes de expansión orientados hacia este propósito, se debe analizar y estructurar de la mejor manera posible, la adición de información de tipo BLOB (Binary Large Object) a la base de datos. Es necesario contemplar los siguientes aspectos:

a) Qué tipo de imágenes es conveniente incorporar en el sistema, de tal manera que éstas les sean útiles tanto a las oficinas locales y foráneas, muchas de éstas manejadas quizá en forma confidencial.
b) De qué tipo serían las más solicitadas: planos de ubicación, estructurales, arquitectónicos, de vistas interiores o exteriores, de formas, colores, modelos, tamaños, etc.
c) Con base en lo anterior, determinar las características técnicas de estos archivos, el formato, el tamaño, en color o en tonos de gris, iluminación, resolución, etc.

Posteriormente, será necesario adecuar dentro de la aplicación algunas funciones que permitan visualizar el contenido de estas imágenes en forma estándar, dándole consistencia a estas opciones para que estas nuevas capacidades de consulta de información gráfica sean eficientes y para que no se altere la funcionalidad a la que ya estaba adaptado el usuario con el sistema.

6.4. Aspectos relacionados con el hardware y el software

Conceptos básicos

En esta sección trataremos de explicar básicamente a nivel de conceptos, aquellos términos más comúnmente utilizados, a fin de dar un panorama lo más completo posible de todo lo que involucra un proyecto de estas características, particularmente por el tipo de herramienta de desarrollo empleada y la tecnología de comunicación asociada (cliente-servidor).

Servidor:

Software instalado en un hardware de alta capacidad, que se encarga específicamente de administrar múltiples bases de datos y múltiples usuarios en forma concurrente, guardar la referencia de la localización actual de datos en el(los) dispositivo(s) de almacenamiento, mantener el mapeo entre la descripción lógica de datos y el almacenamiento físico de los mismos, y de gestionar los recursos de datos y objetos en la memoria del equipo.

La administración eficiente de todas estas operaciones las realiza a través de lo que se conoce como "engines" o motores del servidor de base de datos. La tecnología actual permite trabajar con más de un motor dentro del mismo servidor, orientados principalmente al aprovechamiento de sistemas multiprocesadores, aunque esto no significa que un "engine" represente a un CPU físico dentro de la máquina. Estos "engines" lo que hacen es comunicarse con otros procesos o servicios del servidor, utilizando la memoria como compartida. Corriendo en un equipo uniprocesador, siempre se tendrá en operación a un solo engine (engine 0). Para máquinas con varios procesadores, se pueden configurar varios engines y repartir las tareas en forma simétrica o bien asociar cierto tipo de operaciones a cada uno de ellos.

Base de datos:

Aunque generalmente este término se asocia comúnmente con el servidor de base de datos, estrictamente se refiere al conjunto de tablas de datos relacionadas y otros objetos que son organizados para un propósito específico. La base de datos queda almacenada en un dispositivo dedicado al almacenamiento de la información y los objetos que conforman la base de datos. Este dispositivo puede ser un arreglo de discos, un volumen de un disco o un archivo dentro de un sistema de archivos. A su vez los objetos de la base de datos a los que hacemos referencia son componentes del tipo: tablas, vistas, índices, procedimientos, disparadores (triggers), reglas referenciales (constraints), etc.

Consideraciones para la implantación C/S

El modelo cliente/servidor es muy sencillo en su conceptualización, pero no necesariamente sucede lo mismo en cuanto a su implantación. En el momento en que una compañía empieza a distribuir sus aplicaciones y servicios de cómputo, la administración de sistemas y la definición de dónde ubicar los datos deben ser seriamente considerados y evaluados.

Un solo sistema de LAN puede requerir poca administración, sin embargo, en un ambiente de cómputo cliente/servidor a nivel empresarial, las facilidades de administración de sistemas son requisito indispensable para una implantación exitosa.

Además de lo anterior existen otros puntos a considerar:

1. ***Reforzar la seguridad de acceso***: En los sistemas distribuidos, el acceso es frecuentemente limitado a proteger ciertos datos dentro de un servidor, sin embargo no hay ningún control físico, lógico ó administrativo para el control de datos usados por múltiples PC's, excepto para controlar el medio físico de almacenaje.

2. ***Mantener la integridad de la información***: Es necesario definir quién es el responsable de mantener la integridad de los datos y de las aplicaciones que se encuentran distribuidos en las redes que componen el sistema, estableciendo políticas y procedimientos de seguridad.

3. ***Soporte de proveedores***. Otro problema potencial es el soporte y servicio. Una red es generalmente integrada con productos de varios proveedores. Entre más proveedores involucrados en un sistema, más difícil es la determinación de los problemas. Esta situación a veces llega a empeorarse por la falta de experiencia en ambientes distribuidos y heterogéneos.

6.5. Migración a otras plataformas (Internet).

Adición de objetos de consulta en Internet.

Finalmente, dado que cliente/servidor e Internet trabajan bajo un mismo esquema de continuidad, se contempla la posibilidad de expandir el producto hacia esta plataforma. Está teniendo un gran impacto en la actualidad y seguirá creciendo de tal manera que para fines de siglo, podríamos considerar que si alguna empresa (bancos, sector privado, industrias, almacenes, etc, incluso gobierno) no cuenta al menos con su página en Internet, puede considerar nula su oportunidad de crecer por medio del comercio electrónico.

Es por ello que en nuestro caso, la etapa de migración hacia Internet deberá ser considerada como un proceso fundamental y estratégico para el crecimiento, no sólo del sistema, sino de la empresa misma. La idea que vale la pena resaltar aquí, es la posibilidad que en el diseño sea posible darle un estructura de operación similar al de la aplicación cliente/servidor.

Aunque definitivamente una página HTML tiene características de operación propias y muy diferentes a las de una herramienta de desarrollo como PowerBuilder, a lo que nos referimos es a tratar de presentar al menos una estructura similar en la presentación de la información por ejemplo, o hacer uso de los mismos objetos que se comunicarán con la base de datos para la recuperación de la información, etc., de tal manera que no se duplique el esfuerzo orientado hacia la explotación de los datos.

CONCLUSIONES.

Acerca de la Tecnología de Información:

La Tecnología de Información empieza a ser una necesidad para el crecimiento de las empresas, quienes se preocupan cada vez más por proporcionar nuevos servicios a sus clientes. La empresa moderna y con visión del futuro aprovecha las ventajas que ofrecen las nuevas tecnologías para mantener a sus clientes actuales y ganar nuevos, fortalecer su imagen y presencia en los mercados, llegar a un mayor número de prospectos y clientes minimizando los costos. El incremento de la productividad de los empleados en las empresas, así como la optimización de las distintas actividades y funciones mediante la automatización de los procesos, representa también una necesidad para permanecer en mercados tan dinámicos y competidos como los de hoy en día. Los clientes esperan y exigen cada vez más, mejores empresas y mejores servicios.

Por todo lo anterior, cada vez es mayor la participación de los altos niveles directivos en las decisiones sobre inversión en tecnología de información. Nuevas tecnologías como Cliente-Servidor e Internet son temas obligados al momento de definir estrategias. Afortunadamente la tecnología ha evolucionado de tal forma que los altos directivos tienden a conocerla, e incluso a utilizarla en sus actividades diarias, cada vez con mayor frecuencia. De este modo, empiezan a reconocer la importancia de estas nuevas tecnologías en el futuro de sus empresas. En la actualidad es más común ver a altos directivos de empresas en eventos relacionados con la tecnología, donde se les explican los beneficios que pueden obtener de su uso.

Acerca de las Metodologías:

Las Metodologías en el desarrollo de los Sistemas de Información representan un factor crítico de éxito. No utilizar una metodología puede representar el fracaso total de un proyecto, costos muy altos de mantenimiento, retrasos indefinidos en el logro de los objetivos, usuarios insatisfechos, etc. Tanto empresas especializadas en el desarrollo de sistemas como empresas que desarrollan sus propios sistemas de acuerdo a las reglas específicas de sus negocios, coinciden en los siguientes puntos:

- Siempre es necesario apegarse a una Metodología.
- Esta metodología puede ser el resultado de una combinación de diferentes metodologías, aplicando a éstas criterios propios en función de las necesidades del negocio.
- La rápida evolución de las tecnologías disponibles para el desarrollo de sistemas (Cliente-Servidor, Internet, Objetos, Eventos, etc.), en relación a la aparición de nuevas metodologías que involucren estas nuevas tecnologías, obliga a los desarrolladores a improvisar en este sentido mediante la adaptación de las ya existentes.
- El objetivo de desarrollar un sistema como solución a una necesidad, siempre debe estar como prioridad principal, por encima del uso de una metodología, cualquiera que ésta sea.

Acerca del Usuario:

Para el área de Bienes Adjudicados de nuestro cliente, este sistema ha representado un cambio en su manera de operar y administrar la información relativa a los Bienes Adjudicados. Ha permitido optimizar las actividades del personal del área incrementando la productividad, así como un fuerte apoyo a quienes toman las decisiones. Mientras que antes la información se tenía en diferentes medios, ahora se encuentra en una sola Base de Datos que garantiza la integridad de la misma, permitiendo la obtención de resultados confiables y oportunos, formas sencillas de obtener reportes y consultas, facilidad de uso, etc. Como ejemplo de ello, existen una serie de reportes que proporcionan información crítica: Los Bienes más costosos en su mantenimiento, los Bienes listos para la venta, los de mayor valor, los más antiguos, por distribución geográfica, etc. Cuenta también con informes requeridos por las autoridades bancarias que se generan de manera automática.

Por todo lo anterior, este sistema representa una alternativa de solución para el área de Bienes Adjudicados de todo el sector financiero, dado que se rigen por la misma normatividad y sus procesos son muy semejantes. Debido a que este sistema no es un paquete de software, sino más bien una solución integral, puede ser adaptado a las necesidades muy particulares de nuevos clientes, agregándole nuevos procesos, funciones y reportes, modificando los ya existentes, etc. Es decir, puede ser entregado justamente a la medida de sus necesidades.

Acerca del Equipo de Trabajo:

Para nosotros, este proyecto representa la oportunidad de aplicar nuevos conceptos en cuestión de desarrollo de sistemas. Nos ha permitido integrarnos en un equipo de trabajo con personas ya incorporadas al ambiente productivo y desempeñando actividades diversas. Esta mezcla de perfiles representó para nosotros una valiosa experiencia y enriqueció el desarrollo del proyecto.

A lo largo de nuestro trabajo tuvimos que cuidarnos de mantener un equilibrio entre nuestro sentido práctico, propio de profesionistas en ejercicio, y el apego a los cánones que marcan los estudiosos de los sistemas de información. Procuramos también mantener un sentido de calidad en cada actividad desarrollada. Creemos haber hecho una mezcla adecuada de estos importantes aspectos y esperamos que así se aprecie en nuestro trabajo.

Finalmente, nuestro proyecto implicó un esfuerzo extra, así como la satisfacción de haber concluido un compromiso con nosotros mismos, con nuestra universidad y con la sociedad.

BIBLIOGRAFIA

- Ingeniería del Software
  Roger S. Pressman
  3ª Edición.
  McGraw Hill

- Análisis y Diseño de Sistemas
  Kendall & Kendall
  1ª Edición.
  Prentice Hall 1991

- Fundamentos de Bases de Datos
  Henry F. Korth
  Abraham Siberschatz
  2ª Edición
  McGraw Hill 1993

- Introducción a los Sistemas de Bases de Datos
  C.J. date
  1ª Edición
  Addison-Wesley Iberoamericana.

- Application Development using Power Builder
  James J. Hobuss
  1ª edición
  John Wiley & Sons 1994

- Fundamentos Cliente/Servidor
  Centro Europeo de Soluciones Cliente/Servidor de IBM
  2ª Edición
  IDG Comunications S.A.

- A Practical Guide to Relational Database Design
  Peter Domanski & Philip Irvine
  Domanski-Irvine Book Company. 1995

- Principles of DataBase and Knowledge-Base Systems
  Joffrey D. Ullman
  1ª Edición.
  Computer Sciencie 1988

- DataBse Management and Design
  Gary W. Hansen & James V. Hansen
  2ª Edición.
  Prentice Hall 1996

- DataBase Analysis and Design
  I.T. Hawryszkiewycz
  2ª Edición
  Macmillan Publishing Company 1991

- Software Engineering Mathematics
  Jim Woodcok & Martin Loomes
  1ª edición.
  Addison Wesley 1989

- Software Engineering
  Shaki Laurence
  2ª Edición.
  Macmillan Publishing Company 1991

- Software Engineering
  Ian Sommerville
  3ª Edición
  Addison Wesley 1989