59

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

APLICACION DE TECNICAS DE REINGENIERIA A LOS PROCESOS ADMINISTRATIVOS PARA DAR DESTINO A LOS BIENES PROPIEDAD DEL FISCO FEDERAL

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A :

CESAR LOPEZ LEZAMA

DIRECTOR DE TESIS: ING. ERNESTO SUAREZ SPORT

CIUDAD UNIVERSITARIA MARZO DE 2000

A la memoria de mi hermano Luis Edmundo
y por los éxitos que esperan a mi hermano David.
Gracias a los dos.

Indice temático

Introducción

Un número considerable de las importaciones que se llevan a cabo en México no cumple con los requisitos en materia fiscal y/o aduanera para acreditar la estadía legal en el país de los bienes involucrados en dichas operaciones pasando estos a propiedad del Fisco Federal. Determinar las políticas, procedimientos y criterios para el control, administración y destino de las mercancías de comercio exterior que han pasado a propiedad del Fisco Federal o se encuentren en los casos previstos en el artículo 157 de la Ley Aduanera es la función principal de la Dirección General de Destino de Bienes de Comercio Exterior Propiedad del Fisco Federal de la Secretaría de Hacienda y Crédito Público.

El proyecto a desarrollar pretende llevar un control automatizado de las operaciones propias de esta dependencia:

- Recepción de solicitudes de asignaciones y donaciones por parte de las distintas dependencias de gobierno e instituciones de asistencia privada.
- Control de inventarios de mercancías y vehículos en los diferentes recintos fiscales del país.
- Hacer entrega de mercancías que cumplan con las características solicitadas por los beneficiarios de la SHCP.
- Agilizar el desalojo de recintos fiscales dando celeridad a los procesos administrativos asociados a la determinación del destino final de los bienes que ahí se encuentran.
- Generar reportes adecuados que apoyen la toma de decisiones a nivel gerencial.

Uno de los mayores problemas en un proyecto de reingeniería apoyado en un sistema de información es lo relativo de su evaluación a corto plazo. Un ejemplo claro de esto son los sistemas informáticos de la década de los 80s, rentables en su momento y cada vez más sencillos de administrar gracias al advenimiento de nuevas tecnologías, pero que, sin embargo, están causando pérdidas millonarias al tener que rediseñar la estructura de datos en la que se basan ya que en virtud a las limitaciones de espacio físico en memoria a las que entonces se enfrentaban, el registro de los datos de tipo fecha se limitaba a 6 caracteres, de los cuales solo dos corresponden al año, así, lo que en su momento vino a mejorar la

relación costo beneficio del proyecto, ahorrando grandes cantidades de espacio de almacenamiento para los datos del tipo fecha, ahora es la razón de la reestructuración mayor de estos sistemas de información. Este trabajo no es la excepción, y en unos cuantos meses el recién mencionado ejemplo será intrascendente, un cambio más como lo fue en su momento la migración de unidades de almacenamiento caseras con disquetes de 5 ¼" a las actuales de 3 ½". Es por esto que lejos de establecer una evaluación categórica de un proyecto particular, el objetivo del presente trabajo de tesis es el servir de referencia para el desarrollo de algún proyecto incluso de mayor proporción al aquí planteado.

El trabajo se organiza en tres partes: conceptos teóricos, metodología de implantación y conclusiones del proyecto; se describen a continuación las dos primeras:

Conceptos teóricos

La primera parte del trabajo comienza introduciendo la idea de la ingeniería de programación como una aplicación práctica de herramientas y técnicas propias de la ciencia de la computación a una serie de problemas reales dentro de una organización.

El capítulo dos trata de los conceptos generales de las bases de datos relacionales. Las actividades de una organización son dinámicas, sin embargo, se apoyan de una base de información que cuenta con una estructura estática. La mayoría de los requerimientos del sistema se expresan en lenguaje natural y habrán de constituirse de una manera clara y de modo que puedan validarse, a partir de ello se desarrolla una representación gráfica de la base de información de este proceso, la cual, en nuestro caso, es un diagrama entidad-relación.

El tercer capítulo introduce algunos de los conceptos del esquema cliente/servidor, el cual brinda los medios necesarios para integrar la productividad del personal individual con las necesidades específicas de procesamiento de datos para satisfacer así los requerimientos de la organización como un todo. La mayoría de las organizaciones de alto desempeño dependen de una red de área local y de redes de área amplia para sus necesidades de comunicación, un uso imaginativo de esta infraestructura puede elevar considerablemente la efectividad de una organización.

Para concluir con los conceptos teóricos en los que se basa el proyecto, el capítulo cuatro establece el papel de la ingeniería en computación dentro de la reingeniería de procesos de una organización. La reingeniería consiste en la transformación radical de una organización de una estructura rígida definida por departamentos a una estructura flexible definida por procesos, la meta de la ingeniería, por otro lado, es aprovechar al máximo la fuerza y los recursos de la naturaleza, los ingenieros en computación

no tratan con dichos fenómenos naturales, pero siguen las mismas premisas que los ingenieros convencionales, muchas de las cuales son relevantes para la reingeniería de sistemas.

Metodología de Implantación

La segunda parte del trabajo desarrollado, la metodología de implementación comienza con la definición en el capítulo cinco de la organización en términos de procesos y la especificación de estos. Los costos, riesgos y beneficios asociados con los diferentes procesos y las variantes de su implementación determinan las prioridades de la reingeniería. Los procesos de mayor interés son aquellos que transforman la investigación y el desarrollo en productos finales. Se concluye el capítulo con una descripción de los procesos detectados dentro de la organización y el flujo de información entre cada uno de ellos, identificando plenamente las interacciones de dichos procesos con las bases de información internas que culminaran en aplicaciones de explotación a estas bases de datos.

Una vez planteadas las expectativas de la organización ya definida por los procesos que la componen habrá de establecerse claramente en cuales de estos actuaran los sistemas de información a desarrollar, punto que se describe en el capítulo seis. El diseño de un sistema parte de dos planteamientos paralelos: el diseño conceptual para el cliente y el diseño técnico para los diseñadores del sistema y para apoyar al soporte físico que habrá de darse al mismo Las necesidades del cliente obligan a ver el sistema como un proceso y como un producto. El proceso creativo del diseño de un sistema es la transformación del problema en una solución. El producto resultante es una descripción de la solución: el diseño del sistema.

En el capítulo siete se definen las bases de la programación orientada a eventos utilizada por los sistemas de interfaz gráfica de amplia aceptación hoy en día. Con base en los lineamientos de operación definidos en el diseño conceptual y apoyados de la plataforma de programación, administrador de base de datos y estructura de la base de información establecidas en el diseño técnico se encaminarán esfuerzos a la obtención del sistema final, documentando debidamente la estructura de programación en la que se basa.

Como parte final y permanente de todo proceso de implementación de un nuevo sistema de información está el mantenimiento al mismo, pudiendo éste ser correctivo, de adaptación, perfectivo o preventivo. El mantenimiento se facilita si se anticipa la necesidad de futuros cambios en el momento de diseñar el proceso, el capítulo ocho discute este tema y el cómo habrá de efectuarse el mantenimiento al sistema generado.

En lo que refiere a la bibliografía utilizada, esta se concentra por mucho en la de los cursos de capacitación de la empresa Microsoft, en virtud de que la plataforma elegida por la Dirección General de Destino de Bienes para su manejador de base de datos fue SQL Server 6.5 y para la programación de aplicaciones cliente Visual Basic 5.0

La decisión, como es el caso de toda área de la administración pública, respondió a una licitación pública de la plataforma informática en la que habrán de apoyarse los sistemas informáticos de la Dirección General. La elección, Microsoft Corp., cubrió las expectativas de confiabilidad, soporte y escalabilidad planteadas. En el aspecto técnico y en mi particular opinión la decisión resulto adecuada, si bien es cierto que SQL Server sobre Windows NT no constituye un manejador de base de datos tan robusto como por ejemplo Informix sobre el sistema operativo Unix, esta última combinación requiere de un costoso mantenimiento, y sus características mejoradas de desempeño no se explotan en su totalidad con una base de datos de tamaño regular como la utilizada por parte de la Dirección General de Destino de Bienes. En lo que toca a la plataforma de programación de las aplicaciones cliente, la interfaz familiar de Windows generada con Visual Basic facilita la labor de capacitación a los usuarios finales, además el mantenimiento o ajustes de los módulos que componen al sistema es relativamente sencillo; a pesar de no poder efectuar en esta plataforma desarrollos de bajo nivel, aquellos que se refieren a acceso remoto a datos y compatibilidad con aplicaciones tales como Office están considerados de manera natural en la plataforma, lo que la hace adecuada a las necesidades de la Dirección General.

Conceptos teóricos

Introducción al desarrollo de sistemas de información

En este capítulo se introduce la idea de la ingeniería de programación como una aplicación práctica de herramientas y técnicas propias de la ciencia de la computación a una serie de problemas reales. La ingeniería en computación aporta soluciones prácticas y no teorías, buscando la producción de aplicaciones de calidad, pero no siempre se logra construir sistemas informáticos que cubran adecuadamente las necesidades del cliente.

1.1 Sistemas de información

Dentro de las organizaciones, el *análisis y diseño de sistemas* se refiere al proceso de examinar la situación de una empresa con el propósito de mejorarla con métodos y procedimientos más adecuados. Esta sección presenta un panorama del análisis y diseño de sistemas y describe el trabajo de los analistas de sistemas así como los diferentes tipos de usuarios que participan en el proceso de desarrollo.

En el sentido más amplio un sistema de información es un conjunto de componentes que interactúan entre sí para lograr un objetivo común. La finalidad de un sistema es la razón de su existencia. Existe un sistema legislativo por ejemplo, para estudiar los problemas que enfrentan los ciudadanos y aprobar la legislación que los resuelva. El sistema de encendido de un automóvil tiene el claro propósito de quemar combustible para crear la energía que emplean.

Para alcanzar sus objetivos, los sistemas interactúan con su medio ambiente, el cual esta formado por todos los objetos que se encuentran fuera de las fronteras de los sistemas. Los sistemas que interactúan con su medio ambiente (reciben entradas y producen salidas) se denominan *sistemas abiertos*. En contraste, aquellos que no interactúan con su medio ambiente se conocen como *sistemas cerrados*. Todos los sistemas actuales son abiertos.

El elemento de *control* está relacionado con la naturaleza de los sistemas, sean cerrados o abiertos. Los sistemas trabajan mejor cuando operan dentro de niveles de desempeño tolerables. Por ejemplo, las personas trabajan mejor cuando su temperatura es de 37 grados centígrados. Quizá una desviación de 37 a 37.5 grados no afecte mucho su desempeño aunque en algunos casos, la diferencia puede ser

notable. Una mayor desviación, sin embargo, tal como una fiebre de 39.5 grados, desencadena un cambio drástico en las funciones corporales. El sistema deja de funcionar y permanece inactivo hasta que se corrija su condición. Si esta condición se prolonga demasiado, los resultados pueden ser fatales para el sistema.

Este ejemplo muestra además la importancia del control en los sistemas de todo tipo. Todos los sistemas tienen niveles aceptables de desempeño, denominados *estándares* y contra los que se comparan los niveles de desempeño actuales. Siempre deben anotarse las actividades que se encuentran muy por encima o por debajo de los estándares para poder efectuar los ajustes necesarios. La información proporcionada al comparar los resultados con los estándares junto con el proceso de reportar las diferencias a los elementos de control recibe el nombre de retroalimentación (véase Fig. 1.1).

Para resumir, los sistemas emplean un modelo de control básico consistente en:

1. Un *estándar* para lograr un desempeño aceptable
2. Un método para *medir* el desempeño actual
3. Un método para *comparar* el desempeño actual contra el estándar
4. Un método de *retroalimentación*

Los sistemas que pueden ajustar sus actividades para mantener niveles aceptables continúan funcionando. Aquellos que no lo hacen, tarde o temprano dejan de trabajar.

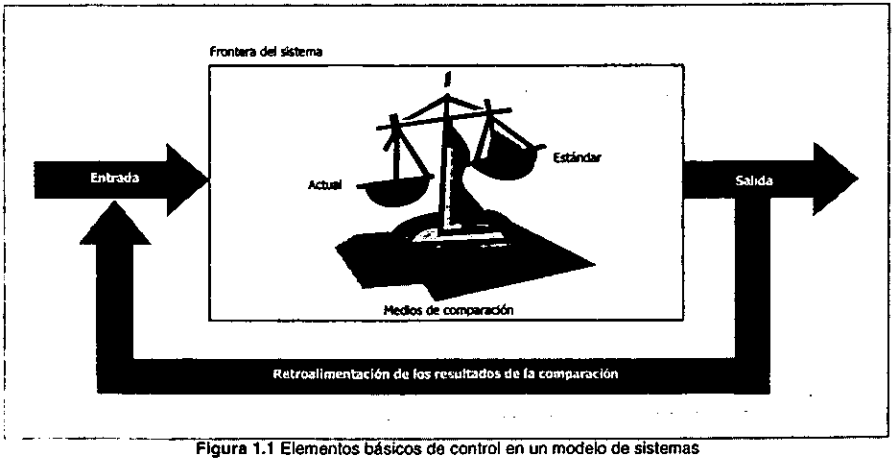


Figura 1.1 Elementos básicos de control en un modelo de sistemas

Sistemas organizacionales

Las organizaciones, como la del caso de estudio del presente documento, están formadas por muchos sistemas, cada uno con las características propias del sistema general. Por ejemplo, todos los sistemas de inventarios tienen similitudes. Su finalidad es tener conocimiento de los movimientos en un almacén, para alcanzar este objetivo, los sistemas interactúan con su medio ambiente llevando un libro de entradas-salidas donde se registra documentalmente los ingresos de mercancías al almacén y las entregas que de estas se realicen, y los datos de la persona a quien se entregan así como una adecuada clasificación de los mismos en función de la manejabilidad de los bienes. Si el control del almacén ha de mantenerse, no es posible prescindir de ninguno de los elementos de control. Los sistemas de control también generan salidas tales como actas de Entrega-Recepción, reportes de inventarios y calendarización de salidas.

Para mantener su funcionamiento, estos sistemas deben estar bajo control. Por ejemplo, necesitan satisfacer ciertos estándares normativos para el respaldo legal de las actas Entrega-Recepción. Los datos personales de los involucrados deben estar debidamente sustentados y la descripción de los bienes por salir ampliamente detallada.

Los encargados de almacén vigilan constantemente la entrada de bienes y las comparan contra la capacidad de sus almacenes. Si existe saturación en los últimos o bien desperdicio de espacios, entonces se efectúan los cambios necesarios. En este sentido los sistemas de inventario son autorregulables y auto ajustables ya que indican el movimiento a realizar de los bienes y el momento para hacerlo, el equipo de apoyo que debe comprarse o los procedimientos que deben modificarse. Si los ajustes internos no son satisfactorios entonces es probable que hagan su aparición las fuerzas regulatorias del medio ambiente.

Los sistemas de inventarios son subsistemas de organizaciones más grandes; éstas a su vez forman otros subsistemas para adquisición de bienes, contratación y capacitación de personal. Las características generales de todos los sistemas son las mismas. Cualquier sistema puede analizarse en este marco de referencia en mente, añadiendo los detalles que sean necesarios. Esta flexibilidad es la que hace útil los conceptos de sistemas en las organizaciones en general, y en el diseño de sistemas de información en particular.

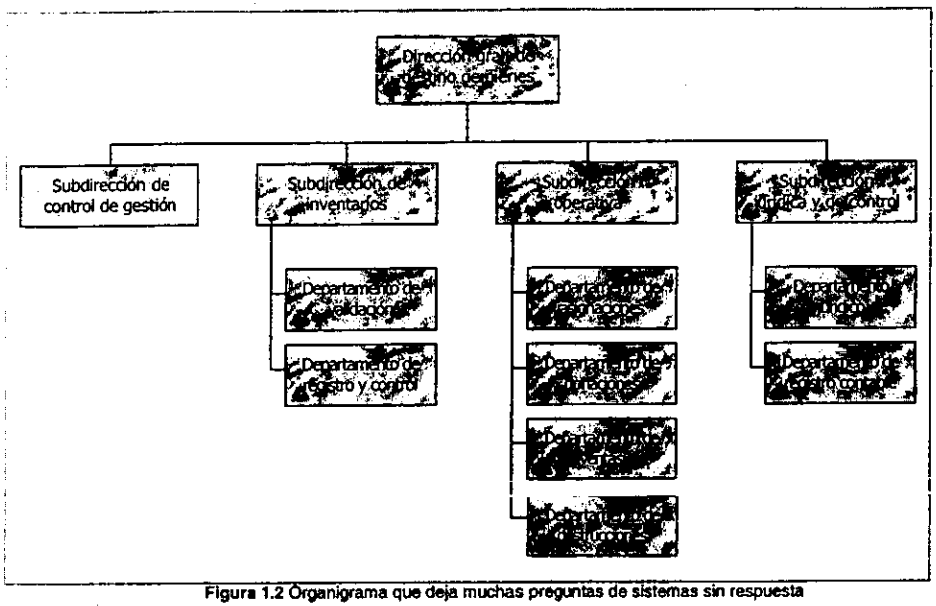


Figura 1.2 Organigrama que deja muchas preguntas de sistemas sin respuesta

Sistemas de información organizacionales

La finalidad de los sistemas de información, como las de cualquier otro sistema dentro de una organización, es la de procesar entradas, mantener archivos de datos relacionados con la organización y producir reportes, informes y otras salidas.

Los sistemas de información están formados por subsistemas que incluyen hardware, software, medios de almacenamiento de datos para archivos y bases de datos. El conjunto particular de subsistemas utilizados -equipo específico, programas, archivos y procedimientos-- es lo que se denomina una *aplicación* de sistemas de información. De esta forma, los sistemas de información pueden tener aplicaciones en ventas, contabilidad o compras.

Dado que los sistemas de información dan soporte a los demás sistemas de la organización, los analistas tienen primero que estudiar el sistema organizacional como un todo para entonces detallar sus sistemas de información. Los organigramas (véase Fig. 1.2) se emplean, con frecuencia, para describir la forma en que están relacionados los diferentes componentes de la organización, tales como direcciones, departamentos, oficinas y empleados. Aunque los organigramas indican con precisión las relaciones formales entre los diferentes componentes no dicen nada respecto a la forma en que opera el sistema

organizacional; ya que en este tipo de diagramas no es posible plasmar todos los detalles importantes. A continuación se dan varios ejemplos de detalles importantes para el analista de sistemas:

1. *Canales informales.* ¿Qué interacciones existen entre las personas y los departamentos que no aparecen en el organigrama o no están descritos en los procedimientos de la operación?
2. *Interdependencias.* ¿De qué otros departamentos y componentes de la organización depende un elemento en particular?
3. *Personas y funciones clave.* ¿Cuáles son las personas y elementos más importantes en el sistema para que este tenga éxito?
4. *Enlaces críticos de comunicación.* ¿Cómo es el flujo de información e instrucciones entre los distintos componentes de la organización? ¿Cómo se comunican las áreas entre sí?

La anterior no es una lista exhaustiva de preguntas pero recalca la importancia de investigar y analizar la manera en que operan las organizaciones.

En contraste, durante el diseño los analistas tienen la responsabilidad de identificar las características importantes y necesarias que deben tener los nuevos sistemas. El analista especifica la forma en que va a operar el sistema y sus subsistemas, las entradas requeridas, las salidas que se van a producir y los trabajos que se efectuarán tanto por las computadoras como en forma manual. Por otro lado, también participan en el control de los sistemas básicamente en dos formas: la primera cuando describen los elementos de control, tales como estándares y métodos de evaluación del desempeño en relación con los demás estándares para los sistemas de información que diseñan. Al mismo tiempo, los sistemas que especifican proporcionan información a los directivos y usuarios que permite a éstos determinar si los sistemas que administran operan correctamente. Incorporar mecanismos de retroalimentación es un paso esencial en el diseño ya que su inclusión permite sostener actividades de ambos sistemas. Ninguno de los sistemas perdurará si falta un control adecuado.

1.1.1 Categorías de sistemas de información

El analista de sistemas desarrolla diferentes tipo de sistemas de información para satisfacer las diversas necesidades de una empresa.

Sistemas para el procesamiento de transacciones

El sistema, basado en computadora, más importante dentro de una organización es el que esta relacionado con el procesamiento de las transacciones. Los *sistemas de procesamiento de transacciones* tienen

como finalidad mejorar las actividades rutinarias de una empresa y de las que depende toda la organización. Una transacción es cualquier suceso o actividad que afecta a toda la organización. Las transacciones más comunes incluyen: facturación, entrega de mercancías, pago a empleados y depósito de cheques. Los tipos de transacciones cambian en cada una de las diferentes organizaciones. Sin embargo, la mayor parte de las compañías procesan dichas transacciones como una mayor parte de sus actividades cotidianas. Las empresas con mayor éxito llevan a cabo este trabajo en una forma ordenada y eficiente.

El procesamiento de transacciones, que es el conjunto de procedimientos para el manejo de éstas, incluye entre otras, las siguientes actividades:

- Cálculos
- Clasificación
- Ordenamiento
- Almacenamiento y recuperación
- Generación de resúmenes

Todas estas actividades forman parte del nivel operacional de cualquier organización (Fig. 1.3). El estudio de un grupo de organizaciones también muestra la existencia de características similares entre ellas:

1. Gran volumen de transacciones.
2. Gran similitud entre las transacciones.
3. Los procedimientos para el procesamiento de transacciones están bien comprendidos y se pueden describir a detalle.
4. Existen muy pocas excepciones a los procedimientos normales.

Estas características permiten establecer rutinas para el manejo de transacciones. Las rutinas describen qué buscar en cada transacción, los pasos y procedimientos a seguir, y lo que debe hacerse en caso de que se presente una excepción. Los procedimientos para el proceso de transacciones se denominan *procedimientos de operación estándar*.

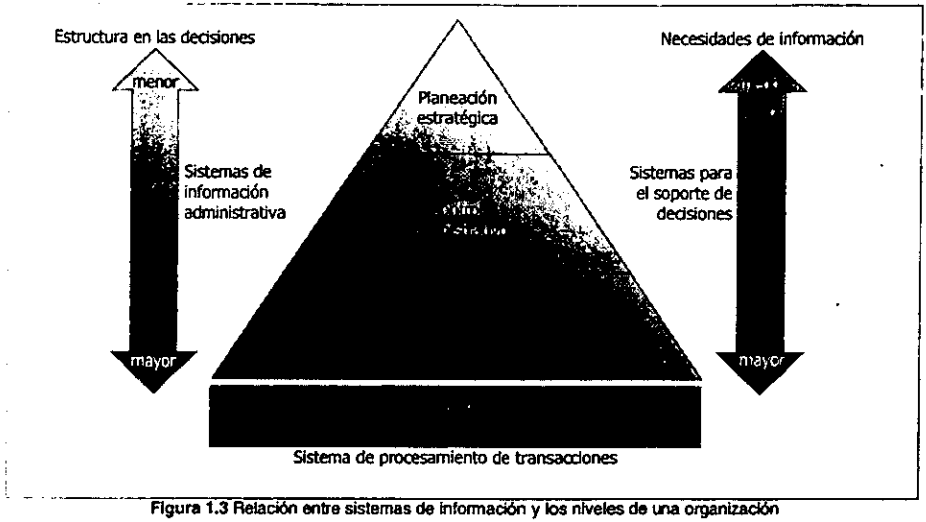


Figura 1.3 Relación entre sistemas de información y los niveles de una organización

Sistemas de información administrativa

Los sistemas de transacciones están orientados hacia operaciones. En contraste, los *sistemas de información administrativa* ayudan a los directivos a tomar decisiones y resolver problemas. Los directivos recurren a los datos almacenados como consecuencia del procesamiento de las transacciones, pero también emplean otra información.

En cualquier organización se deben tomar decisiones sobre muchos asuntos que se presentan con regularidad (diariamente, a la semana, al mes, al trimestre, etc.) y para hacerlo se requiere de cierta información. Dado que los procesos de decisión están claramente definidos, entonces se puede identificar la información necesaria para formular las decisiones. Se pueden desarrollar sistemas de información para que, en forma periódica preparen reportes para el soporte de decisiones. Cada vez que se necesita información esta se prepara y presenta en una forma y formato diseñados con anterioridad.

Con frecuencia los especialistas en sistemas de información describen estos sistemas como decisiones estructuradas (véase Tabla 1.1). El aspecto estructurado se refiere al hecho de que los administradores conozcan de antemano los factores que deben tenerse en cuenta para la toma de decisiones así como las variables con influencia más significativa sobre el resultado de una decisión (buena o mala). A su vez,

los analistas de sistemas desarrollan reportes bien estructurados que contienen la información necesaria para las decisiones o que indican el estado de las variables importantes.

Categoría de los sistemas de información	Características
Sistema para el procesamiento de transacciones	Sustituye los procedimientos manuales por otros basados en computadora. Trata con procesos de rutina bien estructurados. Incluye aplicaciones para el mantenimiento de registros.
Sistema de información administrativa	Proporciona la información que será empleada en los procesos de decisión administrativos. Trata con el soporte de situaciones de decisión bien estructuradas. Es posible anticipar los requerimientos de información más comunes.
Sistemas para el soporte de decisiones	Proporciona información a los directivos que deben tomar decisiones sobre situaciones particulares. Apoyan la toma de decisiones en circunstancias que no están bien estructuradas.

Tabla 1.1 Categorías de los sistemas de información

Sistemas para el soporte de decisiones

No todas las decisiones son de naturaleza recurrente. Algunas se presentan sólo una vez o escasamente. Los *sistemas para el soporte de decisiones* ayudan a los directivos que deben tomar decisiones no muy estructuradas, también denominadas *no estructuradas* o *decisiones semiestructuradas* (véase Tabla 1.1). Una decisión se considera no estructurada si no existen procedimientos claros para tomarla y tampoco es posible identificar, con anticipación, todos los factores que deben considerarse en la decisión.

Un factor clave en el uso de estos sistemas es determinar la información necesaria. En situaciones bien estructuradas es posible determinar esta información con anticipación, pero en un ambiente no estructurado resulta difícil hacerlo. Conforme se adquiere la información, puede ocurrir que el gerente se dé cuenta que necesita más información; es decir, tener información puede conducir a otros requerimientos.

En estos casos es imposible diseñar de antemano tanto el formato como el contenido de los reportes del sistema. En consecuencia, los sistemas para el soporte de decisiones deben tener una flexibilidad mayor que la de los demás sistemas de información. El usuario debe ser capaz de solicitar informes definiendo su contenido y especificando la forma para producir información. De manera similar, los datos necesarios para generar la información pueden encontrarse en diferentes archivos o bases de

datos más que en un solo archivo maestro, que es el caso más frecuente en los sistemas de transacciones y en muchos otros que generan reportes.

El criterio de los directivos tiene un papel importante en la toma de decisiones donde el problema no es estructurado. Los sistemas para el soporte de decisiones ayudan pero no reemplazan el criterio del directivo.

1.2 Análisis y determinación de requerimientos

Cuando un cliente solicita un nuevo sistema tiene en mente la idea de lo que hará el sistema, en ocasiones el sistema reemplaza a otro existente o a cierto modo de hacer las cosas, es decir, el sistema tiene un propósito. Un *requerimiento* es parte de la razón de ser del sistema, la descripción de algo que este será capaz de hacer para cumplir su propósito independientemente de la forma en que habrá de implantarse el mismo (administración de la base de datos, programación de interfaces, equipos de explotación, etc.).

Esta distinción se hace más clara si consideramos el propósito del análisis de requerimientos. Al principio del proceso de desarrollo de sistemas la meta es *determinar la naturaleza del problema del cliente*. La discusión de cualquier solución es prematura hasta que el problema este claramente definido en términos de la actividad del cliente, es decir, los requerimientos deben *enfocarse en el cliente y el problema*, no en la solución o la implementación.

Estrategias para el desarrollo de sistemas

- Un sistema es un conjunto de componentes que interactúan entre sí para lograr un objetivo común.
- Un sistema de información es el que proporciona datos relevantes de manera oportuna y veraz.
- Dado que los sistemas de información dan soporte a los demás sistemas de la organización, los analistas tienen primero que estudiar el sistema organizacional como un todo, para entonces detallar sus sistemas de información.
- Los analistas de sistemas son los encargados de proporcionar las ideas respecto a las mejores formas para usar eficientemente los recursos informáticos en apoyo a los procesos en comento.
- La información que reúnen los analistas sobre el sistema de la empresa, forma la base para el diseño de nuevos sistemas o para las modificaciones que ya existen.
- La participación de los **usuarios** es vital para el éxito de un proyecto ya que ambos, **directivos** y **operativos** apoyan la labor del analista mediante las siguientes técnicas:

  Entrevistas
  Observación del sistema actual
  Recopilación documental

Análisis preliminar

- Una organización es un sistema, el cual debe apoyarse en un sistema de información que lo complemente. No son entidades ajenas ya que son uno solo.
- Conocer la estructura jerárquica.
- Proceso de recopilación y revisión de datos para el nuevo sistema.
- Identificación de la operación actual.
- Identificar normas para operación.
- Identificar las entidades que intervienen en el proceso.
- Identificación de flujos de información: entradas, salidas y procesos que intervienen.
- Identificar errores que provoquen un desarrollo de trabajo lento, buscar los *cuellos de botella*:

  ¿Cómo evitarlos?

  ¿Qué interviene en ellos que impide avanzar con agilidad?

  ¿Qué se puede hacer?

  ¿Con qué elementos?

  ¿En que parte del proceso se opera incorrectamente?

  ¿Dónde no se concluye con las etapas de trabajo?

- Identificar quien toma decisiones y basándose en que.
- ¿Qué tipo de decisiones se toman?
- Comprender la organización actual, su objetivo, fin, alcances y limitaciones.

Funciones del analista de sistemas

- Debe ser consultor.
- Identificar el proceso básico y global de la organización.
- Es un especialista que está para apoyar la operación correcta y buscar optimizar procesos y provocar cambios.
- Es innovador.
- Analiza datos basados en hechos reales.
- Busca:

  Acelerar procesos

  Simplificación de procesos eliminando pasos innecesarios o duplicados

Combinar procesos

Reducir errores de captura mediante formas de acceso a la información y pantallas, así como su validación

Eliminar salidas de información redundante

- Debe comprender los procesos, no solo conocerlos.
- Seleccionar estrategias para satisfacer estos requerimientos.
- Preguntas que debe hacerse el analista de sistemas:

  ¿Qué datos utiliza en este proceso?

  ¿Cuáles son los limites impuestos por el tiempo y la carga de trabajo?

  ¿Qué controles de desempeño se utilizan?

  ¿Dónde se realizan estos pasos?

  ¿Quiénes los realizan?

  ¿Cuánto tiempo tardan en efectuarlos?

  ¿Con cuanta frecuencia lo hacen?

  ¿Quiénes emplean la información resultante?

- Identificar que pasos de los procedimientos actuales funcionan bien y cuales no.
- En donde se presentan la mayor parte de los problemas.
- Como se podrían agilizar.
- Como se pueden reducir errores.
- Que origina retrasos, porque y como se puede manejar esta situación.
- Tratar de llevar todos los procesos al sistema de tal forma que no se puedan omitir pasos ni realizar de más.
- Identificar problemas con español estructurado.
- Identificar flujos de datos.
- Elaboración de diagramas.
- Identificar factores críticos para el éxito.
- Elaboración de prototipo.
- Integración de otras herramientas como hojas de cálculo y administradores de bases de datos.

Lo que no es el análisis de sistemas

No es el estudio de una empresa para buscar procesos ya existentes con el propósito de determinar cuáles deberían ser llevados a cabo por una computadora y cuáles por métodos manuales. La finalidad del análisis está en comprender

los detalles de una situación y decidir si es deseable o factible una mejora. La selección del método, ya sea utilizando o no una computadora, es un aspecto secundario.

No es determinar los cambios que deberían efectuarse. La finalidad de la investigación de sistemas es estudiar un proceso y evaluarlo. En algunas ocasiones no sólo no se necesita un cambio sino que éste tampoco es posible. Los cambios deben ser un resultado, no un intento.

No es determinar la mejor forma de resolver un problema de sistemas de información. Sin importar cuál sea la organización, el analista trabaja en los problemas de ésta. Es un error hacer una distinción entre los problemas de la empresa y los de sistemas ya que estos últimos no existirían sin los primeros. Cualquier sugerencia debe primero considerarse a la luz de sí beneficiara o perjudicará a la organización. No se debe ir tras ideas técnicamente atractivas a menos que estas mejoren el sistema de la organización.

1.2.1 Herramientas para determinar requerimientos del sistema

Los analistas utilizan métodos específicos con el objeto de reunir datos relacionados con los requerimientos. Entre éstos se incluyen la entrevista, la revisión de registros (sistema actual) y la observación. En general, los analistas emplean más de una de estas técnicas para estar seguros de llevar a cabo una investigación amplia y exacta.

Elementos de una entrevista

Los analistas emplean la entrevista para reunir información proveniente de personas o grupos. Por lo general, los entrevistados son usuarios de los sistemas existentes o usuarios en potencia del sistema propuesto. En algunos casos, los entrevistados son gerentes o empleados que proporcionan datos para el sistema propuesto o que serán afectados por él.

ELEMENTOS CONTROLADOS POR EL ANALISTA:

- A quien entrevistar
- Fecha y lugar
- Tópicos
- Curso de ésta.
- Beneficios.

- Profundidad.

FINALIDAD DE LA ENTREVISTA:

- Conocer el estado actual del sistema en general.
- Metas personales del entrevistado.
- Metas de la organización.
- Procedimientos formales e informales.
- La opinión personal de la persona no una descripción de los procesos.

PREGUNTAS DE EJEMPLO:

- ¿Cuánto tiempo lleva trabajando aquí?
- ¿Qué piensa de los objetivos de este departamento?
- ¿Cuál es el problema más grave para usted en su área?
- ¿Cuál es su opinión del actual sistema de cómputo?
- ¿Cómo se relaciona esta forma de operar con el trabajo que usted desempeña en particular?
- ¿Cuáles son algunos de los problemas que percibe respecto a recibir oportunamente la información?
- ¿Cuáles son los errores más comunes en la captura de los datos de este departamento?
- Describa el sistema de cómputo más frustrante con el que haya trabajado.
- ¿Cuál sería el sistema de cómputo ideal para usted?
- ¿Cuántos informes genera usted en un mes?
- ¿Cuál es el problema más grave para usted del sistema actual?
- ¿Cómo se podría mejorar?
- ¿Cómo piensa usted que haría más efectiva la operación de su área?

OBSERVACIONES GENERALES:

Previo a la entrevista es necesario confirmar citas y una vez iniciada esta estrechar manos con el entrevistado presentarse e indicar el motivo de su presencia y la razón de haberlo elegido para entrevistarlo recordándole además que se registrarán los puntos más importantes de la charla recalcando la confidencialidad de los mismos.

Ya durante la entrevista es importante solicitar ejemplos y detalles de los puntos tratados, evitando preguntas dobles o bien preguntas que nos lleven a la respuesta que deseamos escuchar, haciendo sentir a cada entrevistado como parte fundamental del proyecto, una parte vital.

Una vez concluida la entrevista se deberán obtener las conclusiones correspondientes así como generar un informe de la misma.

Revisión de registros

Varios tipos de registros y reportes pueden proporcionar al analista información valiosa con respecto a las organizaciones y a sus operaciones. Al revisar los registros, el analista examina la información asentada en ellos relacionada con el sistema y los usuarios. La revisión de los registros puede efectuarse al comienzo del estudio, como introducción, o también después, y sirve de base para comparar las operaciones actuales, por lo tanto los registros pueden indicar qué está sucediendo.

Los registros incluyen manuales de políticas, reglamentos y procedimientos estándares de operación utilizados por la mayor parte de las organizaciones como guías para los gerentes y empleados. Estos registros no indican la forma en la que se desarrollan las actividades en la realidad, donde se encuentra todo el poder de la toma de decisiones, o cómo se realizan todas sus tareas. Sin embargo pueden ser de gran ayuda para el analista en su afán de comprender el sistema al familiarizarlo con aquellas operaciones que necesitan apoyo y con las relaciones formales dentro de la organización.

Observación

La observación permite al analista ganar información que no se puede obtener por otras técnicas. Por medio de la *observación* el analista obtiene información de primera mano sobre la forma en que se efectúan las actividades (Tabla 1.2). Este método es más útil cuando el analista necesita observar, por un lado, la forma en que se manejan los documentos y se llevan a cabo los procesos y, por otro, si se siguen todos los pasos especificados. Los observadores experimentados saben qué buscar y cómo evaluar la importancia de lo que observan.

La observación muestra al analista	
Lo que debería suceder...	**Lo que en realidad ocurre...**
• Procedimientos estándares de operación • Controles y comprobación de exactitud y grado de terminación • Documentos llenados en forma apropiada • Trabajo terminado con eficiencia y a tiempo	• Retrasos en el trabajo • Información que se recuerda de memoria (en forma incorrecta) • Pasos omitidos • Más fotocopias de las necesarias • Necesidad de nuevos controles • Falta de información en los archivos cuando se tiene la necesidad de hacer llamadas telefónicas • Los documentos no se llenan en la forma requerida • Los empleados desconocen los procedimientos críticos

Tabla 1.2 La observación proporciona información de primera mano sobre la forma en que se realiza un trabajo

1.2.2 Estrategia de desarrollo por análisis estructurado

El *análisis estructurado* se concentra en especificar lo que se requiere que haga el sistema o la aplicación. No establece cómo se cumplirán los requerimientos o la forma en que implantará la aplicación. Más bien permite que las personas observen los elementos lógicos (lo que hará el sistema) separados de componentes físicos (computadoras, terminales, sistemas de almacenamiento, etc.) Después de esto se puede desarrollar un diseño físico eficiente para la situación donde será utilizado.

Los elementos esenciales del análisis estructurado son el detalle de eventos, símbolos gráficos, diagramas, flujo de datos y el diccionario centralizado de datos.

Lista de eventos

Consiste de una lista narrativa de los estímulos que ocurren en el exterior a los cuales el sistema debe responder distinguiendo los acontecimientos desde el punto de vista del sistema. Por ejemplo, *El sistema recibe un pedido del cliente*, en lugar de *El cliente hace un pedido*. Los acontecimientos se identifican desde el punto de vista externo, es decir de afuera hacia adentro:

- *El usuario inserta su tarjeta en el cajero*
- *El cajero pide la contraseña al usuario*
- *El usuario proporciona su contraseña*
- *El cajero verifica su cuenta y contraseña*

Diagramas entidad-relación

Este modelo describe con un alto nivel de abstracción la distribución de los datos. Es una herramienta que nos apoya para unir el análisis y el diseño con los usuarios durante las fases de determinación de requerimientos y diseño conceptual debido a que es simple y fácil de entender.

Se basa en una percepción del mundo real consistente en un conjunto de objetos básicos llamados entidades y las relaciones que existen entre ellos.

Entidad: Es el objeto principal del cual se tiene que almacenar información, normalmente denotando una persona, lugar, cosa o evento. En un diagrama Entidad-Relación, las entidades se representan con un rectángulo y se nombran con un sustantivo.

Relación: Asocia la entidad de un conjunto con una o varias entidades de otro. En un diagrama entidad relación, estas últimas se representan con un rombo con líneas conectando las entidades en asociadas. Normalmente un verbo transitivo corresponde a la relación.

Diagramas Causa-Efecto

Permiten identificar de una manera clara los efectos de los problemas.

Causa detectada	Efecto
Controles internos de gestión de cada área sin interacción	Descontrol en la Gestión y falta de seguimiento a asuntos, falta de integración
Formas heterogéneas de formatos para solicitud de insumos	Complejidad en la captura lo que implica desperdicio de horas/hombre en su clasificación

Diagramas de contexto

En este diagrama una sola burbuja representa todo el sistema y las entidades externas que interactúan con él. El diagrama de contexto define el sistema en cuestión determinando las fronteras del mismo, todo aquello que no se encuentre dentro de las fronteras identificadas por este diagrama no forma parte de nuestro estudio. Las organizaciones o elementos externos están fuera de nuestro control, sin embargo afectan el proceso en estudio, ya que son fuentes o destinos y debemos tener una interfaz, o medio para interactuar con estos.

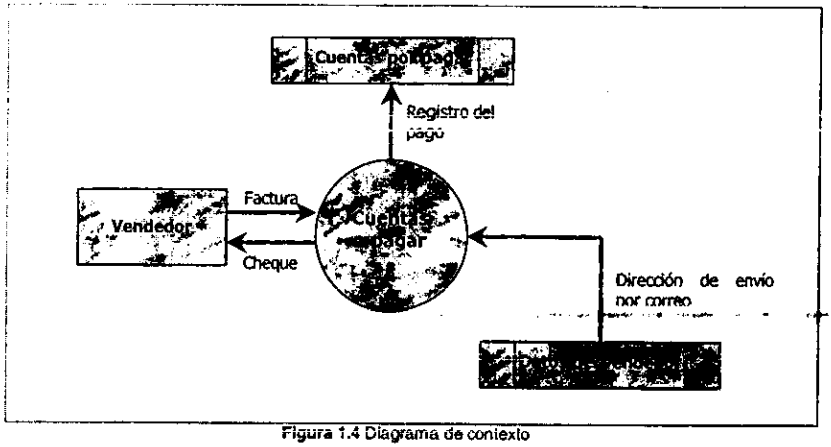


Figura 1.4 Diagrama de contexto

Diagramas de flujo de datos.

Los Diagramas de Flujo de Datos[15] son una herramienta gráfica que se emplea para describir y analizar el movimiento de información a través de un sistema, incluyendo procesos, entidades y bases de datos.

Durante el análisis del flujo de datos se evalúan todos los detalles en términos de los componentes lógicos de flujos de datos, procesos, bases de datos, orígenes y destinos.

En etapas posteriores a la determinación de requerimientos, estos se habrán de trasladar a detalles de diseño lógico. En las fases de construcción las especificaciones lógicas son trasladadas en características físicas y en un sistema de información completo y funcional.

[15] ss. DFD

Los niveles de diagramas de flujo de datos pueden compararse con los mapas de calles y carreteras que emplea una persona cuando viaja por un sitio desconocido. En primera instancia utiliza el mapa de todo el país que muestra las carreteras y ciudades. A medida que se acerca a la ciudad que va a visitar, necesita un mapa mas detallado que señale los diferentes sitios de la ciudad y las calles de acceso. Después, cuando ha llegado al sitio deseado, será de gran utilidad otro mapa que muestre calles y sitios de interés, como puentes y edificios. Tanto detalle es esencial cuando se busca una dirección determinada pero no tiene utilidad cuando se inicia el viaje y se presentan los problemas de orientación.

Los diagramas de flujo de datos se utilizan de la misma forma. Se desarrollan y emplean de forma progresiva, desde lo general hacia lo especifico para el sistema de interés.

PROCESO DE DESARROLLO

Los analistas de sistemas estudian primero el sistema en uso, esto es, las actividades y procesos que ocurren en el presente. En la terminología del análisis estructurado, éste es el estudio del *sistema físico*. Las técnicas para hallar hechos expuestas anteriormente forman la base para recopilar los detalles necesarios.

El sistema físico se traslada en una descripción lógica que se centra en datos y procesos (Fig. 1.5). Recalcar los datos y procesos para abordar las actividades que se llevan a cabo junto con los recursos necesarios para ello, más que sobre quienes realizan el trabajo, tiene sus ventajas.

Los siguientes detalles son ejemplos de un sistema físico:

Departamento	Sala de fotocopiado o ubicación de las instalaciones
Persona	Número de paso
Archivo	Procedimiento

Durante el análisis de flujo de datos se evalúan todos los detalles en términos de los componentes lógicos de flujos de datos, procesos, almacenes de datos, orígenes y destinos.

En todas las etapas de diseño que siguen, los requerimientos del sistema se trasladan en detalles de diseño lógico. En las fases de construcción como la programación del software para computadora, las

especificaciones lógicas son trasladadas en características físicas y en un sistema de información operable.

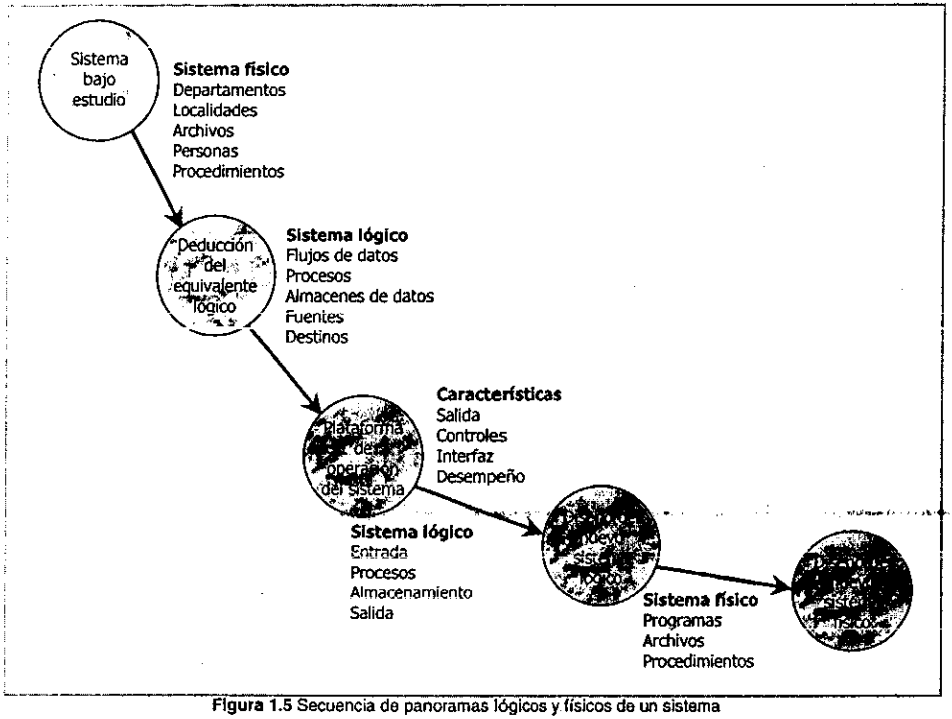


Figura 1.5 Secuencia de panoramas lógicos y físicos de un sistema

Este panorama de la secuencia de actividades para el análisis y diseño de sistemas de información es el escenario del estudio que sigue sobre el análisis de flujo de datos.

REGLAS PARA LA CONSTRUCCION DE DIAGRAMAS DE FLUJO

1. **Escoger nombres con significado para los procesos, flujos, almacenes y entidades.** Nombre que identifique claramente lo que se está representando. Un proceso representa una función que se está llevando a cabo, o indica como se lleva a cabo, identificando personas, grupos o mecanismos involucrados. Evitar nombres de personas: etiquetar los procesos para identificar las funciones que se están realizando. Una buena idea es utilizar un verbo activo (que tenga objeto) y un objeto apropiado para formar una frase descriptiva para el proceso. Evitar nombres ambiguos como

hacer, manejar y procesar, entre otros. El vocabulario utilizado para describir los procesos debe provenir de uno que tenga significado para los usuarios. Evitar la terminología orientada a al programación tales como rutina, procedimiento, subsistema, función entre otros.

2. **Numerar los procesos.** Es importante identificar cada burbuja de manera consistente a través de todo el diseño, teniendo en consideración que los números asociados con las burbujas no implican alguna secuencia. El modelo de DFD es una red de procesos asíncronos que se intercomunican entre sí. Los números se convierten en base para distinguir la jerarquía de los procesos cuando se introducen los diagramas de flujo por niveles.

3. **Evitar DFD complejos.** El propósito de un DFD es ser interpretado y comprendido no solo por el analista que construyó el modelo, sino también por los usuarios que conozcan la materia de nuestro estudio. Debe ser fácilmente entendible, asimilable y placentero a la vista. No es recomendable realizar un DFD con demasiados procesos, flujos, bases de datos y entidades. En promedio no debería de haber más de cinco procesos diferentes relacionados en un solo diagrama. El DFD debe acomodarse cómodamente en una hoja normal.

   Una excepción importante es el diagrama de contexto ya mencionado anteriormente, que representa al sistema completo como un solo proceso y destaca las interfaces entre el mismo y las entidades externas. Comúnmente los diagramas de contexto no se pueden simplificar, por que describen, con el más alto detalle una realidad que es intrínsecamente compleja.

4. **Redibujar el DFD tantas veces como sea estéticamente necesario.** Los DFD tendrán que diseñarse y volverse a diseñar, a menudo hasta 10 veces o más antes de que:
   - Sea técnicamente correcto.
   - Sea aceptable para el usuario.
   - Este lo suficientemente bien diseñado para que no sea embarazoso mostrarlo a los directivos.

5. **Asegurarse de que el DFD sea internamente consistente y que también lo sea con cualquier otro DFD relacionados con él mismo.** Las principales reglas de la consistencia son:
   - Evitar sumideros infinitos, es decir burbujas que cuenten con entradas pero sin salidas.
   - Evitar burbujas de generación espontánea, aquellas que tienen salidas sin contar con entradas o bien que teniéndolas son sospechosas y generalmente incorrectas.

- Evitar contar con flujos y procesos no etiquetados ya que esto indica falta de observación y esconde errores graves.
- Los datos a almacenar no serán de solo lectura o solo escritura, deberán tener tanto entradas como salidas. La única excepción a la regla es algún almacenamiento externo, aquel que sirva de interfaz entre el sistema y alguna entidad externa.

NIVELACION DE DIAGRAMAS DE FLUJO

1. Se debe organizar el DFD global en una serie de niveles de modo que cada uno proporcione sucesivamente más detalles sobre una porción del nivel anterior. Esto es análogo a la organización de mapas de un atlas.
2. El DFD de primer nivel consta sólo de una burbuja que representa al sistema completo; los flujos de datos muestran las interfaces entre el sistema y las entidades externas. Este DFD especial se conoce como diagrama de contexto.
3. El siguiente DFD se conoce como *figura* o *nivel 0*. Representa la vista de más alto nivel de las principales funciones del sistema, al igual que sus principales interfaces. Cada una de las burbujas deben numerarse para una conveniente referencia.
4. Los números también sirven como una manera adecuada de relacionar una burbuja con el siguiente nivel de DFD que la describe más a fondo.
5. El nombre de toda burbuja habrá de reutilizarse en la figura de nivel inmediato inferior.

CONSIDERACIONES GENERALES

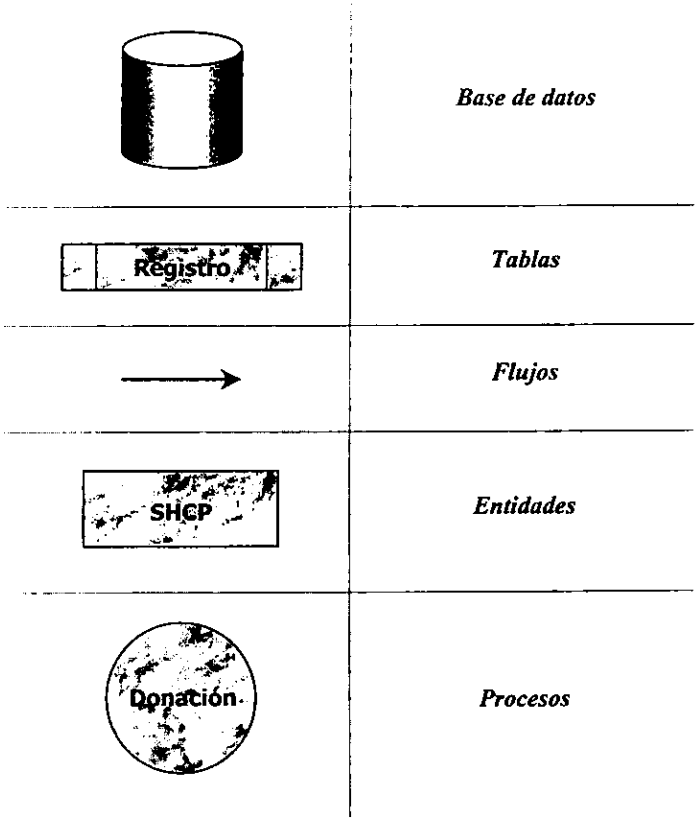
Un DFD valido deberá tener de 2 a 7 niveles, si una burbuja requiere tres niveles más de detalle, no significa que todas las burbujas del DFD tengan que cumplir con estos tres niveles, pueden ser menos o más dependiendo de la complejidad del proceso.

Para asegurarnos que cada figura sea consistente con aquella de mayor nivel a la que se refiere se deberá considerar lo siguiente: Los flujos de datos que salen y entran en una burbuja en un nivel deben corresponder con los que entran y salen de toda la figura en el nivel inmediato inferior que la describe.

Las bases de datos se deben mostrar en el nivel más alto donde por vez primera sirven de interfaz entre dos o más burbujas; luego se muestra de nuevo en cada diagrama de nivel inferior que describa más a fondo dichas burbujas de interfaces. Los almacenamientos locales, que utilizan sólo las burbujas en una

figura de menor nivel, no se mostrarán en niveles superiores, dado que se incluyen de manera implícita en un proceso del nivel inmediato superior.

SIMBOLOGIA



1.2.3 Estrategia de desarrollo por prototipos de aplicaciones

El desarrollo de prototipos es una técnica probada que mejora la efectividad total del esfuerzo de desarrollo -para beneficio del usuario, el analista y la organización en su conjunto.

Este método hace que el usuario participe de manera más directa en la experiencia del análisis y diseño que el propio análisis estructurado. La construcción de prototipos es muy eficaz bajo las circunstancias

correctas. Sin embargo, al igual que otros métodos es útil sólo si se emplea en el momento adecuado y en la forma apropiada.

Aunque el prototipo quizá no incluya todas las características de una aplicación completa, contiene las que son esenciales para profundizar en aquellas que son necesarias en el sistema. Un prototipo es, entonces, una aplicación que trabaja, creada en forma rápida y económica. Entre las razones que se encuentran detrás del desarrollo de prototipos están la aclaración de los requerimientos del usuario y la verificación de la factibilidad del diseño de un sistema. La información ganada con este proceso no solo mejora la eficiencia del desarrollo, también es una manera eficaz para garantizar el desarrollo del sistema correcto. El desarrollo de prototipos de aplicación ofrece ventajas mayores cuando no se conocen los requerimientos o es necesario evaluarlos, cuando el costo o los riesgos asociados con el sistema son grandes, o cuando se emplea una nueva tecnología.

¿Qué es un prototipo?

El *prototipo* es un sistema que funciona –no sólo una idea en el papel-, desarrollado con la finalidad de probar ideas y suposiciones relacionadas con el nuevo sistema. Al igual que cualquier sistema basado en computadora, está constituido por software que acepta entradas, realiza cálculos, produce información ya sea impresa o presentada en pantalla, o que lleva a cabo otras actividades significativas. Es la primera versión, o iteración, de un sistema de información; es el modelo original.

Los usuarios evalúan el diseño y la información generada por el sistema. Lo anterior sólo puede llegar a un buen fin si los datos utilizados, al igual que las situaciones, son reales. Por otra parte, deben esperarse cambios a medida que el sistema es utilizado.

Los usuarios pueden señalar las características que les agradaría o no tener, junto con los problemas que presenta un sistema que existe y funciona, con mayor facilidad que si se les pidiese que las describieran en forma teórica o por escrito. El uso y la experiencia producen comentarios más significativos que el análisis de diagramas y las propuestas por escrito, los usuarios son capaces de señalar características que no son de su agrado o de identificar aquellas que faltan y, a menudo, hacen esto con bastante entusiasmo, comparado con el trabajo hecho con las especificaciones por escrito, al que con frecuencia ellos temen, los usuarios están deseosos de ver un prototipo que funcione.

En general, los pasos a seguir en el proceso de desarrollo de prototipos son los siguientes:

1. Identificar los requerimientos de información que el usuario *conoce* junto con las características necesarias del sistema.
2. Desarrollar un prototipo que funcione.
3. Utilizar el prototipo anotando las necesidades de cambios y mejoras. Esto expande la lista de los requerimientos de sistemas conocidos.
4. Revisar el prototipo con base en la información obtenida a través de la experiencia del usuario.
5. Repetir los pasos anteriores las veces que sea necesario, hasta obtener un sistema satisfactorio.

Por medio de la iteración, el prototipo evoluciona hasta que llega el momento de tomar la decisión de implantar el prototipo, transformarlo en un sistema completo o abandonar el proyecto. En algunas ocasiones un prototipo de aplicación conduce hasta un nuevo prototipo como consecuencia de la información obtenida con el primer proceso.

Los prototipos de aplicación se pueden dirigir exclusivamente hacia las pantallas de visualización, los procedimientos para procesamiento o hacia las funciones básicas, dependiendo de las necesidades fundamentales de la situación particular.

El analista utiliza herramientas para desarrollar un prototipo de aplicación efectivo. Entre estas se incluyen diferentes tipos de *lenguajes de cuarta generación*, entre los que se incluyen *lenguajes orientados hacia procedimientos, lenguajes de consulta y recuperación* y *generadores de reportes*. Así mismo se pueden utilizar en este proceso *generadores de aplicaciones, generadores de pantallas, sistemas de diccionario de datos*, computadoras personales y *bibliotecas de código*. La combinación correcta de herramientas y técnicas esta determinada por las características de la aplicación y las normas utilizadas por el analista.

1.3 Diseño de sistemas

Especificación de los elementos lógicos del diseño

El diseño de sistemas tiene dos etapas: el diseño lógico y la construcción física del sistema. Cuando el analista formula el *diseño lógico*, escribe las especificaciones detalladas del nuevo sistema, es decir aquellas que describen sus características: salidas, entradas, archivos y bases de datos y los procedimientos, todo en una forma que satisfaga los requerimientos del proyecto. El conjunto formado por todas estas características recibe el nombre de *especificaciones de diseño* del sistema.

El diseño lógico de un sistema de información es similar al proyecto de ingeniería de un automóvil: muestra las características más sobresalientes (como el motor, la transmisión y el espacio para los pasajeros) y la relación que guardan entre sí (donde se conectan los componentes unos a otros o cual es la separación que existe entre las puertas). Los reportes y salidas generadas por el analista son similares a los componentes de diseño del ingeniero. Los procedimientos y datos se enlazan entre sí para producir un sistema que trabaja.

Al diseñar un sistema de inventarios, por ejemplo, las especificaciones del mismo incluyen definiciones de reportes y pantallas de presentación que describen las existencias disponibles, el abastecimiento y retiro de artículos, y el resumen de transacciones realizadas durante, por ejemplo un mes de operación. El diseño lógico también especifica los formatos de entrada y la distribución de la salida en pantalla para todas las transacciones y archivos que son necesarios para dar mantenimiento a los datos del inventario, a los detalles de las transacciones y a los datos de los proveedores. Las especificaciones de procedimientos describen los métodos utilizados para ingresar datos en el sistema, copiar archivos y detectar problemas, si estos se presentaran.

La *construcción física*, que es la siguiente actividad después del diseño lógico, produce el software, los archivos y un sistema que funciona. Las especificaciones de diseño indican a los programadores lo que el sistema doble debe hacer. A su vez, los programadores escriben programas que aceptan la entrada proporcionada por los usuarios, procesan los datos, producen los reportes y guardan los datos en los archivos.

El diseño físico para el sistema de inventarios ya mencionado, esta formado por instrucciones de programa, escritas en un lenguaje de programación. Estos pasos revisan los registros de mercancía en existencia utilizando para ello los datos asentados en la transacción, imprimen los reportes y guardan los

datos. El analista *especifica* los algoritmos necesarios para cambiar las cantidades de mercancía en existencia. Durante la construcción física, los programadores *escriben* las instrucciones necesarias del programa para calcular los datos y producir los resultados.

1.3.1 Transición del análisis al diseño

Durante el diseño, los requerimientos del usuario se trasladan en características del sistema. Se dice que un sistema de información satisface las necesidades de los usuarios sí:

- Realiza en forma apropiada los procedimientos correctos.
- Presenta información e instrucciones en una forma aceptable y efectiva.
- Produce resultados exactos.
- Proporciona una interfaz y métodos de interacción aceptables.
- Es percibido por los usuarios como un sistema confiable.

En discusiones anteriores sobre el análisis de sistemas, se menciono que el principal interés del analista es determinar el sistema correcto y proporcionar el sistema correcto. El objetivo del diseño de sistemas es alcanzar estos dos objetivos.

Elementos del diseño

Los componentes de un sistema de información descritos durante el análisis de requerimientos, son el punto focal del diseño de sistemas. Los analistas deben diseñar los siguientes elementos:

- *Flujos de datos*
  Movimiento de datos hacia, alrededor y desde el sistema.
- *Almacenes de datos*
  Conjuntos temporales o permanentes de datos.
- *Procesos*
  Actividades para aceptar, manejar y suministrar datos e información. Pueden ser manuales o basadas en computadora.
- *Procedimientos*
  Métodos y rutinas para utilizar el sistema de información y lograr con ello los resultados esperados.

- *Controles*
  Estándares y lineamientos para determinar si las actividades están ocurriendo en la forma anticipada o aceptada, es decir si se encuentran "bajo control". Asimismo, debe especificar las acciones que tienen que emprenderse cuando ocurren problemas o se presentan circunstancias inesperadas. Puede incluirse un reporte sobre las excepciones o procedimientos para la corrección de los problemas.
- *Funciones de personal*
  Las responsabilidades de todas las personas que tienen que ver con el nuevo sistema, incluyendo los usuarios, operadores de computadora y personal de apoyo. Abarca todo el espectro de componentes del sistema, incluso desde la entrada de datos hasta la distribución de salidas o resultados. A menudo, las funciones de personal se establecen en forma de procedimientos.

Como se vera mas adelante, estos elementos aparecen una y otra vez en muchas de las características de los sistemas de información. Por consiguiente, todos estos elementos tienen la misma importancia al estructurar el diseño.

Diseño de salidas

El termino *salida*, se refiere a los resultados e información generados por el sistema. Para muchos usuarios finales, la salida es la única razón para el desarrollo del sistema y la base sobre la que ellos evaluarán la utilidad de la aplicación. En la realidad muchos usuarios no operan el sistema de información y tampoco ingresan datos en él, pero utilizan la salida generada por el sistema.

Cuando diseñan la salida, los analistas deben realizar lo siguiente:

- Determinar que información presentar.
- Decidir si la información será presentada en forma visual, verbal o impresa y seleccionar el medio de salida.
- Disponer la presentación de la información en un formato aceptable.
- Decidir como distribuir la salida entre los posibles destinatarios.

La disposición de la información de una pantalla o documento impreso se denomina *distribución*.

Para llevar a cabo las actividades antes mencionadas, se requieren decisiones específicas tales como el empleo de formatos ya impresos cuando se preparan reportes, cuántas líneas planear sobre una pagina impresa o si se deben emplear gráficas y colores.

El diseño de la salida esta especificado en los formularios de distribución, que son hojas que describen la ubicación, características (como longitud y tipo) y formato de los encabezados de columnas y la paginación. Estos elementos son análogos al bosquejo donde el arquitecto indica la localización de cada componente.

Diseño de archivos

El diseño de archivos incluye decisiones con respecto a la naturaleza y contenido del propio archivo, como si se fuera a emplear para guardar detalles de transacciones, datos de tipo histórico o información de referencia. Entre las decisiones que se toman durante el diseño de archivos se encuentran las siguientes:

- Los datos que deben incluirse en el formato contenidos en un archivo.
- La longitud de cada registro, con base en las características de los datos que contiene.
- La secuencia y disposición de los registros dentro del archivo (la estructura de almacenamiento que puede ser secuencial, indexada o relativa).

No todos los nuevos sistemas de información requieren del diseño de todos los archivos utilizados por la aplicación. Por ejemplo, es probable que ya existieran archivos maestros porque estos son utilizados por otras aplicaciones existentes. Tal vez la nueva aplicación necesite hacer referencia solo al archivo maestro. En este caso, los detalles del archivo se incluyen en las especificaciones de diseño de la aplicación, pero el archivo no vuelve a diseñarse.

Diseño de interacciones con la base de datos

Muchos sistemas de información, ya sean implantados en sistemas de computo grandes o pequeños, interactúan con las bases de datos que abarcan varias aplicaciones. Dada la importancia que tienen las bases de datos en muchos sistemas, su diseño es establecido y vigilado por un administrador de bases de datos, que es una persona (o grupo de personas) con la responsabilidad de desarrollar y mantener la base de datos. En estos casos, el analista de sistemas no efectúa el diseño de la base de datos sino que

consulta al administrador de la base para determinar las *interacciones* mas apropiadas con la base de datos. El analista proporciona al administrador la descripción de 1) los datos que son necesarios de la base de datos, y 2) las acciones que tendrán efecto sobre la propia base de datos (por ejemplo la recuperación de datos, cambios en los valores de los datos o el ingreso de nuevos datos en la base).

A su vez, el papel del administrador de la base de datos incluye las siguientes responsabilidades:

- Evaluar la conveniencia de la solicitud del analista.
- Describir los métodos para interactuar con la base de datos
- Asegurar que la aplicación no pueda dañar la base de datos o que la afecte de manera adversa a las necesidades de otros sistemas de información

Diseño de entradas

Los analistas de sistemas deciden los siguientes detalles del diseño de las *entradas*:

1. Qué datos ingresan al sistema.
2. Qué medios utilizar.
3. La forma en que se deben disponer o codificar los datos.
4. El dialogo que servirá de guía a los usuarios para dar entrada a los datos.
5. Validación necesaria de datos y transacciones para detectar errores.
6. Métodos para llevar a cabo la validación de las entradas y los pasos a seguir cuando se presentan errores.

Las decisiones de diseño para el manejo de entradas, especifican la forma en que serán aceptados los datos para su procesamiento por computadora. Los analistas deciden si los datos serán proporcionados directamente, quizá a través de una estación de trabajo, o por el uso de documentos, como talones de ventas, cheques bancarios o facturas, donde los datos a su vez son transferidos hacia la computadora para su procesamiento.

El diseño de entrada también incluye la especificación de los medios por los que tanto los usuarios finales como los operadores darán instrucciones al sistema sobre las acciones que debe emprender. Por ejemplo, un usuario que interactúa con el sistema por medio de una estación de trabajo, tiene que ser capaz de indicarle al sistema ya sea que acepte una entrada, genere un reporte o termine el procesamiento.

Los sistemas en línea incluyen un *diálogo* o *conversación* entre el usuario y el sistema. Por medio del diálogo, el usuario solicita servicios al sistema y le indica cuándo realizar cierta función. A menudo la naturaleza de la conversación en línea hace la diferencia entre un diseño exitoso y otro inaceptable. Un diseño inapropiado, que deja la pantalla en blanco, confunde al usuario con respecto a qué acción debe emprender.

La disposición de mensajes y comentarios en las *conversaciones en línea*, así como la ubicación de los datos, encabezados y títulos sobre las pantallas o documentos fuentes, también forman parte del diseño de entradas. En general, se preparan bosquejos para comunicar la disposición a los usuarios, para que ellos la revisen, y a los programadores y otros miembros del equipo de diseño de sistemas.

Diseño de controles

Los analistas de sistemas también deben anticipar los errores que se cometerán al ingresar los datos en el sistema o al solicitar la ejecución de ciertas funciones. Algunos errores no tienen importancia ni consecuencias, pero otros pueden ser tan serios que ocasionarían el borrado de datos o el uso inapropiado del sistema. Aunque exista solo la más mínima probabilidad de cometer un error serio, un buen diseño de sistema de información ofrecerá los medios para detectar y manejar el error.

Los controles de entrada proporcionan medios para 1) asegurar que solo los usuarios autorizados tengan acceso al sistema, 2) garantizar que las transacciones sean aceptables, 3) validar los datos para comprobar su exactitud y 4) determinar si se han omitido datos que son necesarios.

Diseño de procedimientos

Los procedimientos especifican qué tareas deben efectuarse al utilizar el sistema y quienes son responsables de llevarlas a cabo. entre los procedimientos importantes se encuentran:

- *Procedimientos para entrada de datos*
  Métodos para la captura de datos de las transacciones y su ingreso en el sistema de información (*ejemplo*: secuencia para dar entrada a los datos registrados en los documentos fuente).

- *Procedimientos durante la ejecución*
  Pasos y acciones emprendidos por los operadores del sistema y, en ciertos casos, por los usuarios finales que interactúan con el sistema para alcanzar los resultados deseados (*ejemplo:* montar paquetes de discos o colocar en las impresoras formas preimpresas).
- *Procedimientos para el manejo de errores*
  Acciones a seguir cuando se presentan resultados inesperados (*ejemplo:* ocurre un error cuando el sistema intenta leer datos de un archivo, se atasca durante la impresión de una gran cantidad de hojas).
- *Procedimientos de seguridad y respaldo*
  Acciones para proteger al sistema y sus recursos contra posibles daños (*ejemplo:* ¿cuándo y cómo hacer copias de los archivos maestros o de partes de la base de datos?).

Estos procedimientos deben formularse por escrito y formar parte de la documentación del sistema.

Diseño de especificaciones para programas

Las especificaciones para programas son por sí mismas un diseño. Ellas describen cómo transformar las especificaciones de diseño del sistema –salidas, entradas, archivos, procesamiento y otras- en software de computadora.

El diseño de aplicaciones de computadoras es importante para asegurar que

- Los programas producidos lleven a cabo todas las tareas y lo hagan en forma establecida.
- La estructuración del software en módulos permita su prueba y validación para determinar si los procedimientos son correctos.
- Las modificaciones futuras se puedan realizar en forma eficiente y con un mínimo de interrupción en el diseño del sistema.

Un sistema de información en particular será diseñado sólo una vez, pero será usado repetidamente y es muy probable que evolucione en la medida que cambien las necesidades de los usuarios. Estas observaciones añaden mas importancia al diseño del software.

En algunas organizaciones, existe una separación entre las responsabilidades del programador y las que tiene el analista, en otras, tanto los programadores como analistas comparten las responsabilidades.

Aunque la combinación de responsabilidades facilita la comunicación del diseño a ciertos programadores que trabajan en el proyecto, ésta no elimina los aspectos mencionados hasta el momento.

Los métodos para desarrollar el diseño o para especificar los detalles varían de acuerdo con las prácticas establecidas en cada organización. También serán diferentes como consecuencia de los lenguajes utilizados para escribir el software. Los lenguajes de tercera generación requieren una atención mucho mayor en lo que respecta a los detalles de los procedimientos en comparación con la dedicada cuando se emplean lenguajes de cuarta generación.

1.4 Desarrollo de aplicaciones

Mientras más funciones críticas de los negocios, las organizaciones y las actividades cotidianas se estén automatizando, hay cada vez más confianza en los sistemas automatizados. Este hecho hace que haya una carga mayor en los analistas de sistemas para garantizar que los sistemas que desarrollan sean adecuados. La calidad de un sistema depende de su diseño, desarrollo, prueba e implantación. Una debilidad en cualquiera de estas áreas pondría seriamente en peligro la calidad y, por lo tanto, el valor del sistema para los usuarios.

1.4.1 Ingeniería de sistemas y aseguramiento de la calidad

Los dos objetivos operacionales del diseño de sistemas que siempre buscan las personas que los desarrollan son la confiabilidad y la facilidad de mantenimiento del sistema.

Diseño de sistemas confiables

Se dice que un sistema tiene *confiabilidad* si no produce fallas costosas o peligrosas al usarse de manera razonable, es decir, de tal forma que un usuario típico espera que sea normal. Esta definición reconoce que los sistemas no siempre se utilizan en la manera en que los diseñadores lo esperan. Existen cambios en las formas en que los usuarios usan el sistema y también en las operaciones de la empresa. Sin embargo, hay ciertos pasos que los analistas deben dar para garantizar que el sistema sea confiable cuando se instala y que la confiabilidad se puede mantener después de la implantación.

Diseño de sistemas fáciles de mantener

Las claves para reducir la necesidad de mantenimiento, al igual que para hacer posible que se realicen las tareas esenciales más eficientemente son:

1. Definir con mayor precisión los requerimientos del usuario durante el desarrollo del sistema.
2. Preparar lo mejor posible la documentación del sistema.
3. Usar métodos más efectivos para el diseño de la lógica del procedimiento y comunicárselos a los miembros del equipo del proyecto.
4. Hacer un mejor uso de las herramientas y técnicas existentes.

Como lo indican los comentarios anteriores el diseño es tanto un proceso como un producto. Las prácticas de diseño que se siguen para el software afectan en forma dramática la facilidad del mantenimiento de un sistema: las prácticas de diseño adecuadas dan como resultado un producto al cual puede dársele mantenimiento. La tabla 1.3 resume las clases de mantenimiento encontradas en los ambientes de sistemas de información.

Categoría	Actividad	Frecuencia relativa
Correctivo	Ajustes de emergencia, depuración rutinaria	20%
Adaptivo	Inclusión de cambios a los datos y archivos, así como al hardware y software del sistema	20%
Perfectivo	Mejoras solicitadas por los usuarios, mejoras en la documentación, recodificación para mejorar la eficiencia computacional	60%

Tabla 1.3 Tipos de mantenimiento de un sistema

Diseño de software

Los siguientes principios deben guiar el diseño de aplicaciones (Tabla 1.4)

- *Modularidad y fragmentación*

  Cada sistema debe estar formado por una jerarquía de módulos. Generalmente, los módulos de niveles inferiores son menores en alcance y tamaño comparados con los módulos de nivel superior y sirven para fragmentar procesos en funciones separadas.

- *Acoplamiento*

  Los módulos de un sistema deben tener poca dependencia entre sí.

- *Cohesión*

  Los módulos deben llevar a cabo sólo una función de procesamiento.

- *Extensión de control*

  Los módulos deben interactuar con y coordinar las funciones de un número limitado de módulos de nivel inferior.

- *Tamaño*

  El número de instrucciones contenidas en un módulo es generalmente pequeño.

- *Uso compartido*

  Las funciones no deben repetirse en módulos separados, sino establecerse en un único módulo que se pueda utilizar por cualquier otro cuando sea necesario.

Cada uno de estos principios se examinará con más detalle, para demostrar su aplicación en la práctica.

Principio	Descripción	Objetivos
Modularidad y fragmentación	Diseño de un sistema como una jerarquía de módulos	Diseñar la estructura en forma descendente con módulos que realicen funciones específicas
Acoplamiento	La fuerza de las relaciones *entre* módulos	Maximizar la independencia entre los módulos *minimizando el acoplamiento*
Cohesión (integración)	La fuerza de las relaciones *dentro* de un módulo	*Maximizar la cohesión*; los elementos altamente relacionados deben estar en el mismo módulo
Extensión del control	Número de módulos subordinados al módulo que hace la llamada	Limitar la extensión de control de 5 a 7 módulos
Tamaño	Número de instrucciones que componen al módulo	Limitar el tamaño de forma que la función de todo el módulo se centre en un sólo propósito
Uso compartido	Uso de un módulo por otros módulos	Evitar la duplicidad permitiendo que los módulos sean llamados por otros que necesiten la función de cada uno

Tabla 1.4 Principios del diseño de software

Manejo del proceso para garantizar la calidad

El *aseguramiento de la calidad* es la revisión de los productos y documentación relacionada con el software para verificar su cobertura, corrección, confiabilidad y facilidad de mantenimiento. Y por supuesto incluye la garantía de que un sistema cumple con las especificaciones y los requerimientos para su uso y desempeño deseados.

Niveles de seguridad

Los analistas usan cuatro niveles de aseguramiento de calidad: prueba, verificación, validación y certificación.

Prueba

La prueba de un sistema es un proceso caro pero crítico que puede llevarse hasta 50% del presupuesto destinado al desarrollo del programa. El punto de vista común respecto a las pruebas compartido por

los usuarios, es que se lleva a cabo para demostrar que no hay errores en un programa. Sin embargo esto es prácticamente imposible, puesto que los analistas no pueden demostrar que la aplicación esta libre de errores, en virtud de lo anterior, el enfoque más útil y práctico es en el entendimiento de que la prueba es el proceso de ejecutar el programa con la intención explícita de hallar errores, es decir, hacer que el programa falle. El equipo de pruebas, que puede estar formado por analistas, programadores, o especialistas entrenados en la prueba del software, está tratando en realidad de hacer que el programa falle. Así, una prueba exitosa es aquella que encuentra un error.

Los analistas saben que un plan de pruebas efectivo no garantiza la confiabilidad del sistema. La confiabilidad es asunto del diseño, por lo tanto, esta debe diseñarse en el sistema. Los analistas no pueden probar la confiabilidad del sistema.

Verificación y validación

Como la prueba, la *verificación* tiene la intención de hallar errores, esta se lleva a cabo ejecutando el programa en un ambiente simulado. Por otro lado, la *validación* se refiere al proceso de utilizar el mismo programa en un ambiente no simulado para hallar sus errores.

Cuando los sistemas comerciales se desarrollan con la intención explícita de distribuirlos a través de terceros para su venta, o comercializarlos por medio de oficinas de la propia compañía, primero deben pasar por un proceso de verificación, a veces denominado *prueba alfa*. La retroalimentación de la fase de validación generalmente produce cambios en el software para resolver los errores y fallas que se descubren. Se elige un conjunto de entre quienes serán los usuarios finales del sistema que ponen a trabajar al mismo en un ambiente real. Estas prácticas, ó *prueba beta* utilizan al sistema en las actividades cotidianas a las que se enfrentará; procesan transacciones en directo y producen salidas normales del sistema. El sistema está a prueba en toda la extensión de la palabra, excepto que los usuarios están advertidos de que están usando un sistema que puede fallar. Sin embargo, las transacciones que se están procesando y las personas que usan el sistema son reales.

COMPAÑÍA DE AUDITORES CERTIFICADOS
123 Wall Street
New York, NY 10015

CERTIFICACIÓN DE SOFTWARE

Hemos revisado el paquete de servicio médico de la Medical Office Associates Company y elaborado un informe que describe nuestra revisión. Las observaciones principales de nuestro informe son que, en nuestra opinión, el paquete tiene la documentación adecuada tanto en el nivel del sistema como el de la aplicación, lo cual permite a sus usuarios potenciales que entiendan el paquete. Proporciona controles internos de procesamiento, informes y archivos de datos que son adecuados para permitir el desarrollo de auditorías, de acuerdo con los principios y estándares de los Estados Unidos.

El sistema procesa la información de contabilidad utilizando procedimientos que permiten a los usuarios producir informes que se ajustan a los principios de contabilidad que generalmente son aceptados en los Estados Unidos.

Estos reportes y sus anexos son para el paquete de servicio médico proporcionado por Medical Office Associates Company.

Figura 1.6 Informe de certificación de software

Es posible continuar la validación durante varios meses, en el curso de esta, puede ocurrir una falla y se ha de modificar la programación. El uso continuo posiblemente produzca fallas adicionales y la necesidad de más cambios.

Certificación

La *certificación* de la aplicación es una garantía de que el programa cubre las expectativas que fueron planteadas por los que habrán de ser sus usuarios finales, su importancia va en aumento para las aplicaciones de sistemas de información comerciales. Existe una creciente dependencia de la compra o renta de aplicaciones de este tipo en vez del desarrollo propio dentro de la organización. Sin embargo, antes de que los analistas deseen aprobar la adquisición de un paquete a menudo requieren de la certificación de este por parte del fabricante o de un tercero sin prejuicios hacia el producto.

Por ejemplo, algunas empresas importantes de contabilidad están involucradas en la certificación de paquetes de software, para garantizar que realmente hace lo que el vendedor afirma que realiza y de una manera apropiada. Para certificar de esta forma el software, la empresa asigna un equipo de especialistas que cuidadosamente examinan la documentación del sistema para determinar lo que afirma el vendedor que hace el sistema y cómo lo lleva a cabo. Entonces ellos prueban el software contra estas

afirmaciones. Si no se encuentran serias discrepancias o fallas, certificarán que el software hace lo que la documentación afirma (Fig. 1.6). Sin embargo, no certifican que el software sea el paquete correcto para cierta organización y su equipo de analistas.

1.4.2 Implantación del sistema

La implantación de un sistema, ya sea nuevo o uno ya existente que ha sido modificado, se conforma por las actividades primarias de *capacitación, conversión* y *revisión después de la implantación*.

Capacitación

Aun los sistemas técnicamente elegantes y bien diseñados pueden tener éxito o fracasar debido a la forma en que se operan y usan. Por lo tanto, la capacitación recibida por el personal relacionado con el sistema ayuda, obstruye, o incluso puede llegar a impedir, la implantación exitosa de un sistema de información. Aquellos que estén asociados con el sistema o afectados por el mismo deben conocer con detalle cual es su papel en el mismo, cómo usarlo y qué hará o no hará el sistema. Tanto los operadores como los usuarios del sistema necesitan capacitación.

La capacitación de los operadores del equipo no sólo incluye el adecuado uso del mismo, sino también el cómo diagnosticar desperfectos y los pasos a seguir cuando estos ocurran. La capacitación también incluye lo relativo a los *procedimientos de ejecución* y las actividades de operación normal del sistema, tales como cargar archivos, cambiar formatos de impresión y preparar la comunicación de datos.

La mayor parte de la capacitación del usuario tiene que ver con la operación del sistema en sí, dando la mayor atención a los procesos de manejo de datos. Es imperativo que los usuarios sean capacitados adecuadamente en la captura de transacciones, edición de datos, formulación de consultas y eliminación de registros. Ninguna capacitación queda completa sin familiarizar al usuario con las actividades básicas de mantenimiento al sistema. El descuido en cualquier aspecto de la capacitación puede llevar a situaciones difíciles que produzcan frustración y errores del usuario.

Conversión

Aunque la capacitación de alta calidad es un paso esencial en la implantación de sistemas, no es suficiente por sí misma. La *conversión*, el proceso de cambiar un sistema anterior a uno nuevo, también debe ser cuidadosamente planeada y ejecutada. Existen cuatro métodos comunes: *sistemas paralelos, conversión directa, prueba piloto* y *sistemas por etapas* (Tabla 1.5). Los sistemas paralelos proporcionan la máxima seguridad en la instalación, mientras que la conversión directa ofrece el riesgo máximo. Cuando los sistemas tienen que ver con grandes organizaciones es común el método por etapas, en el que la conversión ocurre de manera gradual (un departamento a la vez por ejemplo). Cuando hay que probar nuevos métodos o ideas, a menudo se usa el enfoque de prueba piloto: un área de la organización usa el sistema y retroalimenta a los analistas. Cuando el sistema está listo para la implantación, se elige uno de los demás métodos de conversión para instalar el sistema.

Método	Descripcion	Ventajas	Desventajas
Sistemas paralelos	El sistema anterior se opera junto con el nuevo	Ofrece la máxima seguridad. Se puede recurrir al sistema anterior si se hallan errores en el nuevo o si ocurren problemas en su uso.	Duplica los costos de operación. El nuevo sistema puede no ser juzgado justamente.
Conversión directa	El sistema anterior se reemplaza por el nuevo. La organización confía completamente en el nuevo sistema.	Obliga a los usuarios a que hagan trabajar al nuevo sistema. Hay beneficios inmediatos de los nuevos métodos y controles.	No hay otro sistema al cual recurrir si surgen dificultades con el nuevo. Requiere de la más cuidadosa planeación.
Enfoque piloto	Se implanta una versión de trabajo del sistema en una parte de la organización. Con base en la retroalimentación se hacen cambios y el sistema se instala en el resto de la organización mediante uno de los demás métodos.	Proporciona experiencia y prueba directa antes de la implantación.	Puede dar la impresión de que el nuevo sistema no es confiable ni está libre de errores.
Por etapas	Se implanta el sistema de manera gradual a todos los usuarios.	Permite a los primeros usuarios aprovechar las ventajas del sistema. Permite la capacitación y la instalación sin uso de recursos innecesarios.	Un largo periodo de instalación provoca la duda en el usuario de sí el proyecto marcha bien (demasiado entusiasmo) o mal (resistencia y falta de un juicio justo).

Tabla 1.5 Métodos para la conversión de sistemas

Revisión después de la implantación

Después de implantar el sistema y completar la conversión, se hace una revisión del sistema conducida igualmente por los usuarios y los analistas. Esto no sólo es una práctica normal, sino que debe ser un proceso formal para determinar qué tan bien esta funcionando el sistema, cómo ha sido aceptado y cuáles ajustes son necesarios.

La revisión también es importante para recabar información para el mantenimiento del sistema. Puesto que ningún sistema es en realidad totalmente completo, el sistema permanecerá mientras no se requieran cambios debido a movimientos internos, como nuevos usuarios o actividades de la organización; o bien situaciones externas, por ejemplo, nuevos requisitos legales, estándares de la industria, o necesidades del mercado. La revisión después de la implantación es la primera fuente de información de los requisitos de mantenimiento.

La calidad de los sistemas, la confianza del usuario y las estadísticas de operación quedan fijadas por un detallado registro de eventos, evaluación del impacto del sistema en la organización y encuestas de actitud entre los usuarios. Los métodos de recopilación de datos utilizados durante el análisis de requerimientos son igualmente efectivos durante la revisión después de la implantación que puede aplicarse a un nuevo proyecto del sistema.

1.5 Administración del proceso de desarrollo de sistemas de información

Los procesos exitosos de sistemas de información son aquellos que han sido dirigidos adecuadamente. Recíprocamente la planeación del desarrollo de sistemas puede fallar si no se estiman los tiempos involucrados en cada una de las tareas que lo componen de manera adecuada, si no se toma en cuenta la naturaleza crítica de algunas tareas y si no se usa el personal efectiva o eficientemente.

El tiempo necesario para desarrollar un sistema se puede estimar usando registros históricos del esfuerzo que se requiere para desarrollar proyectos semejantes. A veces, la experiencia o la intuición son la base para las estimaciones tomando en cuenta las características del programa a desarrollar y el personal a cargo de esta tarea.

Los *requerimientos de tiempo del proyecto* se refieren al tiempo necesario para llevar a cabo una investigación del sistema, formular el diseño lógico, codificar los programas y ordenar e instalar el equipo. En otras palabras, cada actividad asociada con el desarrollo de un sistema de información requiere de cierta cantidad de tiempo que debe estimarse e incorporarse al calendario del proyecto.

Estimación de los tiempos de actividad del sistema

El tiempo de investigación del sistema se determina a partir del número de personas a entrevistar y la cantidad de tiempo necesaria para desarrollar, circular, recibir y analizar los cuestionarios, dirigir observaciones e inspeccionar registros. Aunque el analista depende de otras personas durante las investigaciones del sistema (aquellos que son entrevistados), la estimación del tiempo de investigación es en muchos sentidos menos complicada que la estimación de otras actividades del sistema.

Los diseños lógicos implican la creatividad del analista y también requieren el desarrollo de muchos detalles del sistema, tales como la definición de los reportes, la organización de los archivos, la validación de datos, los métodos de control de éstos y los procedimientos de ejecución. Así las actividades del diseño lógico son más difíciles de estimar que las actividades de análisis.

Sin embargo, en la mayoría de los sistemas, la mayor dificultad para formular los requerimientos del tiempo esta en la estimación del tiempo para codificar y probar los programas. El desarrollo de la lógica del programa requiere aproximadamente del 35% del tiempo de programación total, tiempo que también emplea el proceso de prueba y depuración (Figura 1.7). La codificación real y desarrollo de los

datos de prueba comprende 25% del tiempo. La documentación necesita aproximadamente del 5% de la asignación del tiempo del proyecto[16].

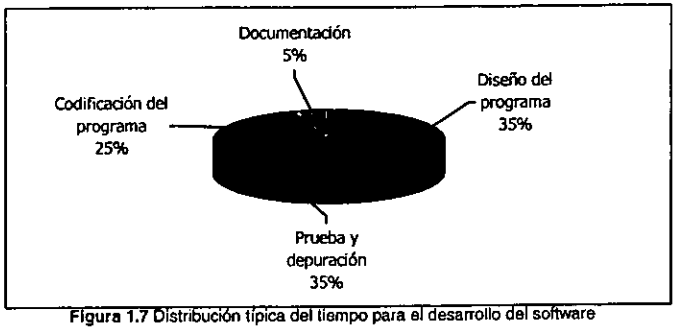


Figura 1.7 Distribución típica del tiempo para el desarrollo del software

La coordinación de un proyecto de sistemas es un aspecto importante del esfuerzo global del desarrollo. Si el sistema se entrega tarde o si los analistas y coordinadores no dan pasos para asegurarse de que el sistema es de alta calidad, seguramente los usuarios se decepcionaran. Incluso puede fallar el sistema. Las guías mencionadas en este capítulo dan la base para una sólida administración del proyecto.

[16] IBM Management Planning Guide for a Manual of Data Processing Standards

Bases de datos relacionales

Los procesos de una organización son dinámicos, sin embargo, se apoyan de una base de información que cuenta con una estructura estática. La mayoría de los requerimientos del sistema se expresan en lenguaje natural y habrán de constituirse de una manera clara y de modo que puedan validarse, a partir de ello se desarrolla una representación gráfica de la base de información de este proceso, la cual, en nuestro caso, es un diagrama entidad-relación.

2.1 Conceptos asociados a bases de datos

Las bases de datos permiten compartir los datos entre distintas aplicaciones. Cuando se diseña un sistema de información para el procesamiento de transacciones, a menudo el centro de atención es una entidad (por ejemplo, tomar una solicitud, aceptar un inventario o contratar un empleado).

Cuando los analistas y usuarios van adquiriendo experiencia con el sistema de información y surgen nuevos requerimientos de información, la atención cambia: de ser capaz de recuperar un registro específico a desarrollar la capacidad de relacionar los registros sobre distintas entidades.

1. Los detalles de una solicitud, tales como el artículo o artículos deseados y las cantidades de cada uno, el número de solicitud del beneficiario y la dirección de envío.
2. Información descriptiva del beneficiario para contar con un detalle histórico de las donaciones o asignaciones a que haya sido sujeto y así poder distribuir mercancías de la mejor manera posible.
3. Detalles de inventario de los artículos solicitados, tales como su descripción, información de empaque (unidades, cajas, galones, etc.) y existencias.

En este ejemplo, existen varias entidades relacionadas. La toma de solicitudes requiere relacionar tres entidades distintas: oficio de solicitud, beneficiario e inventario. De eso se trata el manejo de la base de datos: 1) marcar las relaciones naturales entre los datos y 2) compartir los datos entre entidades en todas las aplicaciones que necesiten los detalles.

Otros ejemplos de relaciones entre entidades pueden ser los departamentos, conformados por empleados; los productos que tienen partes y los proyectos que incluyen a los empleados. En cada uno de estos ejemplos, las entidades están relacionadas entre sí.

Es útil mostrar las entidades y relaciones en forma gráfica por medio de los diagramas de entidad-relación, analizados en el capítulo anterior, conviene hacer hincapié en los requerimientos de distribución de datos en una organización.

Redundancia

Se refiere a la repetición de los mismos datos o elementos a través de diferentes registros, aplicaciones o archivos. Esto sucede cuando en las bases de datos cada aplicación tiene sus propios archivos privados, esto provoca considerable redundancia en los datos almacenados, con el consecuente desperdicio del espacio de almacenamiento.

Puesto que los datos son requeridos por múltiples aplicaciones, con frecuencia se registran en múltiples archivos de datos. En la mayoría de los casos los datos se almacenan repetidamente. Esta situación, llamada redundancia de los datos, conduce a muchos problemas que tienen que ver con la integridad de los datos.

Consistencia

El concepto de consistencia nos lleva necesariamente al de integridad referencial. Desde luego, una base de datos que se halle en estado de inconsistencia puede suministrar información incorrecta o contradictoria.

Integridad

Cuando en una relación de información se modifica algún elemento que se encuentre en varias tablas sin afectar su contenido, ya que cada una de ellas puede poseer una información diferente.

Integridad implica asegurar que lo que se trata de hacer es correcto. El problema de la integridad radica en asegurar que la información de la base de datos sea correcta. La inconsistencia entre dos entradas que representan al mismo "hecho" es un ejemplo de falta de integridad (que, por supuesto, solo ocurre

sí existe redundancia en los datos almacenados). Es conveniente señalar que la integridad de los datos es más importante en un sistema de base de datos que en un sistema de archivos privados, precisamente por que el primero se comparte y porque sin procedimientos de validación adecuados es posible que un programa con errores genere datos incorrectos que afecten a otros programas que utilicen esa información.

Así, como se mencionó anteriormente, cuando en una relación de información se modifica algún elemento que se encuentra en varias tablas y no se afecta a las otras, teniendo información diferente en estas últimas, en este momento decimos que no hay integridad en la base de datos.

La falta de integridad de los datos también puede deberse a una mala verificación de la vigencia de los datos al hacer cambios. Antes del advenimiento de la tecnología de bases de datos, los intentos por integrar los datos eran más difíciles debido a:

- La insuficiente seguridad proporcionada a los datos almacenados.
- Los inadecuados procedimientos de recuperación en caso de falla.
- La dificultad en el manejo de registros largos.
- La inflexibilidad para hacer cambios.
- Los altos costos de programación y mantenimiento.
- La dificultad para manejar los procedimientos en las operaciones de computación (negligencia y errores humanos).

Seguridad

La seguridad implica garantizar que los usuarios están autorizados para llevar a cabo lo que tratan de hacer. El problema de la seguridad tiene muchos aspectos, entre ellos los siguientes:

- *Aspectos legales, sociales y éticos*, por ejemplo, ¿tiene la persona que solicita el crédito de un cliente, digamos, derecho legal a obtener la información solicitada?
- *Controles físicos*, por ejemplo, ¿deberá cerrar o resguardar de alguna otra manera el cuarto de computadoras?
- *Cuestiones de política interna*, por ejemplo, ¿cómo decide la empresa propietaria del sistema, quiénes pueden tener acceso a qué?
- *Problemas de operación*, por ejemplo, si se utiliza un sistema de contraseñas, ¿cómo se mantienen en secreto las contraseñas?, ¿Con qué frecuencia se cambian?

- *Controles de equipo*, por ejemplo, ¿posee el CPU características de seguridad tales como claves para la protección de las áreas de almacenamiento o un modo de operación privilegiado?
- *Seguridad del sistema operativo*, por ejemplo, ¿borra el sistema operativo subyacente el contenido de las áreas de almacenamiento y los archivos de datos cuando ya no se necesitan?
- *Seguridad de objetos*, se refiere a los permisos de los diferentes usuarios para poder hacer uso de tablas, procedimientos almacenados, *triggers*, etc.
- *Seguridad de operaciones*, aquí se manejan permisos para poder modificar (insertar, borrar, actualizar) la base de datos.

2.2 Bases de datos en el diseño de sistemas

Un *sistema de administración de base de datos*[1] proporciona flexibilidad en el almacenamiento de información, recuperación de datos y producción de reportes, sin embargo, el uso de un esquema DBMS no elimina la necesidad de los programas de computación. Como lo muestra la figura 2.1, el DBMS es un puente entre el programa de aplicación, el cual determina qué datos son necesarios y cómo se les procesará, además del sistema operativo de la computadora, que es el responsable de colocar los datos en los dispositivos de almacenamiento. Un *esquema* define a la base de datos y un *subesquema* a la *porción* de la base de datos que utilizara un programa en específico (por lo común, los programas sólo utilizan una sección de la base de datos). Para recuperar los datos de la base de datos:

1. El programa de aplicación determina qué datos se necesitan y comunica la necesidad al DBMS.
2. El DBMS determina que los datos solicitados realmente estén almacenados en la base de datos (aún cuando podrían estar almacenados bajo un nombre distinto, un alias).

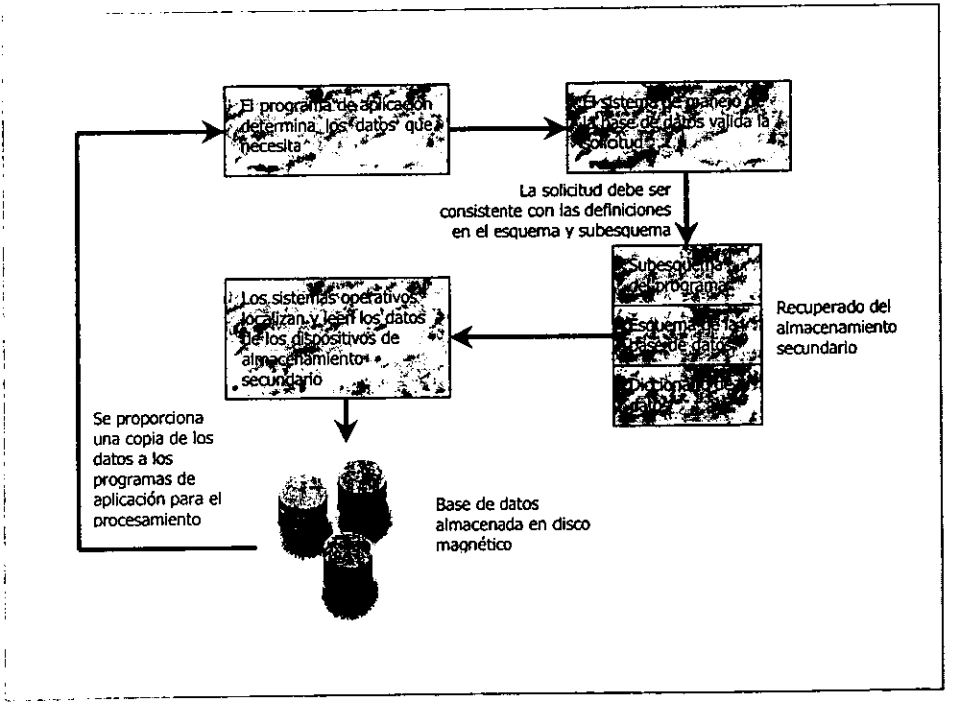


Figura 2.1 Interacción entre el programa de aplicación y la base de datos

[1] DBMS: Data Base Management System

3. El DBMS instruye al sistema operativo para localizar y recuperar los datos del lugar específico en el disco magnético (o cualquier dispositivo donde se almacenen).
4. Se da una copia de los datos al programa de aplicación para su mantenimiento.

El DBMS permite la *independencia de los datos*, lo cual significa que el programa de aplicación puede cambiar sin afectar los datos almacenados o recuperados, por otro lado, la base de información puede volverse a crear y reestructurar sin afectar a los programas de aplicación. Un diccionario de datos se introduce en el DBMS por medio del esquema y subesquema para asegurarse de que los datos están definidos y descritos de forma adecuada y que la duplicidad de los nombres (alias) no produce un almacenamiento redundante de los datos o la pérdida de integridad de los datos.

En general el uso de un DBMS tiene las siguientes ventajas:

- *Puede reducirse la redundancia.* En sistemas que no usan bases de datos, cada aplicación tiene sus propios archivos privados. Esto a menudo origina enorme redundancia en los datos almacenados, así como desperdicio resultante del espacio de almacenamiento.
- *Puede evitarse la inconsistencia*, al menos en cierta medida, en función de la robustez de la aplicación de explotación de base de datos.
- *Los datos pueden compartirse*, lo cual no sólo significa que las aplicaciones existentes pueden compartir la información de la base de datos, sino también que es factible desarrollar nuevas aplicaciones que operen con los mismos datos almacenados.
- *Pueden hacerse cumplir las normas establecidas*, con un control central de la información, el DBMS garantiza que se cumplan todas las formas aplicables a la representación de los datos. Las normas aplicables pueden comprender la totalidad o parte de lo siguiente: normas de la compañía, de instalación, departamentales, industriales, nacionales o internacionales. Es muy deseable unificar los formatos de los datos almacenados como ayuda para el intercambio o migración de los datos entre sistemas.
- *Pueden aplicarse restricciones de seguridad*, al tener jurisdicción completa sobre los datos de operación, el DBMS datos puede, 1) asegurar que el único medio de accesar la base de datos sea a través de los canales establecidos y, 2) definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles.
- *Pueden equilibrarse los requerimientos contradictorios*, cuando se conocen los requerimientos globales de la empresa, en contraste con los requerimientos de cualquier usuario individual, el DBMS puede

estructurar el sistema de bases de datos para brindar un servicio que sea el mejor para la empresa en términos globales.

- *Es compacto*, ya que no hacen falta archivos de papeles que pudieran ocupar mucho espacio.
- *Es rápido*, el DBMS puede obtener y modificar datos con mucha mayor velocidad que un ser humano, así es posible satisfacer con rapidez consultas de casos particulares, sin necesidad de búsquedas visuales o manuales que requieren mucho tiempo.
- *Es menos laborioso*, elimina gran parte del trabajo de mantener archivos a mano, las tareas mecánicas siempre serían mejor realizadas por las máquinas.
- *Es actual*, se dispone en cualquier momento de información precisa y al día.
- *Conserva la independencia física de los datos*, las aplicaciones de explotación de la base de datos permanecen inalteradas sin importar los cambios efectuados en el almacenamiento o en los métodos de acceso.

2.3 Modelo relacional

El *modelo relacional* es en la actualidad el más popular en los sistemas de manejo de una base de datos, puesto que es conceptualmente sencillo y comprensible por los profesionales de los sistemas de información y muchos otros usuarios finales; puede evolucionar, ya que las relaciones entre los datos no necesitan estar predefinidas, además utiliza valores de los datos para implicar las relaciones. El modelo relacional de datos, desarrollado en 1970 por E. F. Codd, se basa en una *relación*: una tabla bidimensional. Los renglones y las columnas muestran los atributos de la entidad (Fig. 2.2). Las *bases de datos relacionales* utilizan un modelo para mostrar cómo se relacionan lógicamente los datos de un registro.

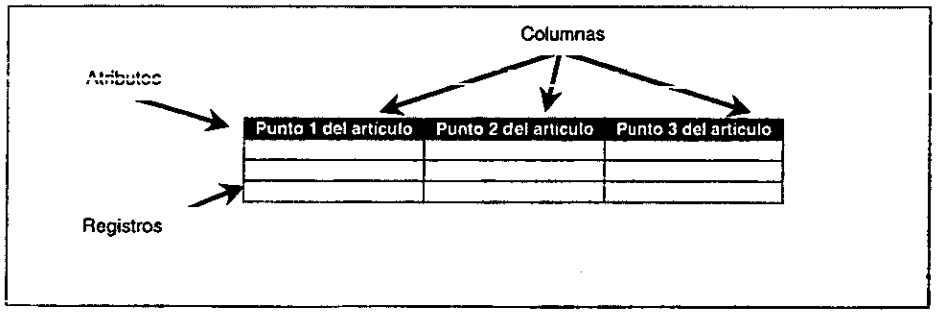


Figura 2.2 Componentes de relación

El orden de los datos en la tabla no es significativo y tampoco implica un orden cuando los registros están incluidos en la relación. Análogamente, los detalles físicos de almacenamiento (ya sea una organización aleatoria, indexada o secuencial) no son de interés para el analista. Las tablas relacionales muestran las relaciones lógicas, no físicas. Al hacer una solicitud de información, el sistema produce una tabla que contenga la información.

2.3.1 Estructuración de datos

Al planear la organización de los datos que van a almacenarse, el analista debe prever la necesidad de accesar los datos para cumplir con requerimientos inesperados, objetivo que se puede alcanzar mediante la normalización de los datos.

Normalización

Una dependencia funcional se representa cuando los valores de un conjunto de atributos de una tupla determinan de manera única los valores de otro conjunto de atributos.

El proceso de cristalización de las entidades y sus relaciones en formatos de tabla usando los conceptos relacionales se llama proceso de normalización. El proceso de reducción o normalización no es una función de los valores de los datos que aparecen en las relaciones en algún momento determinado, sino que es una función de las relaciones entre los atributos; es el proceso de agrupar a los campos de datos en tablas que representas a las entidades y sus relaciones. Por lo tanto el proceso de normalización es una disciplina que consiste en agrupar a los campos de datos en un conjunto de relaciones (tablas).

La teoría de normalización esta basada en la observación de que cierto conjunto de relaciones presenta mejores propiedades en un medio de actualización, inserción y supervisión, que las que presentan otros conjuntos de relaciones que contienen los mismos datos. Para segur el proceso de normalización, es absolutamente necesario que el diseñador de la base de datos entienda la base de la semántica de la información.

La razón de usar el procedimiento de normalización es asegurar que el modelo conceptual de la base de datos funcionara. Esto no significa que una estructura no normalizada no funcionará, sino que puede causar algunos problemas cuando los programadores de aplicaciones de explotación traten de modificar los datos.

Las formas normales son una serie de restricciones que se definen sobre las estructuras relacionadas para evitar, como ya se señaló, anomalías al efectuar adiciones, eliminaciones o actualizaciones de tuplas. Con el fin de conseguir que una relación cumpla con una forma normal se efectúa un proceso de descomposición. Esta implica dividir los atributos de una relación en dos subconjuntos (posiblemente con una intersección no vacía) sin que por ello se pierda alguna información contenida en la relación original.

Las formas de normalización fueron propuestas originalmente por Codd, en 1971 y 1972. Posteriormente varios investigadores continuaron trabajando en esta teoría y a lo largo del tiempo han surgido varias formas de normalización que complementan y refuerzan a las enunciadas por Codd.

La normalización se lleva a cabo por cuatro razones:

- Estructurar datos de forma que se puedan representar las relaciones pertinentes entre ellos.
- Permitir la recuperación sencilla de los datos en respuesta a las solicitudes de consulta y reportes.
- Simplificar el mantenimiento de los datos actualizándolos, insertándolos y borrándolos.
- Reducir la necesidad de reestructurar o reorganizar los datos cuando surjan nuevas aplicaciones.

Los analistas de sistemas deben familiarizarse con los pasos de la normalización, ya que este proceso puede mejorar la calidad del diseño de una aplicación:

1. Descomponer todos los grupos de datos en registros bidimensionales.
2. Eliminar todas las relaciones en las que los datos no dependan completamente de la llave primaria del registro.
3. Eliminar todas las relaciones que contengan dependencias transitivas.

Primera Forma Normal

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

Una entidad R está en primera forma normal[4] si los valores, para cada atributo $A \in R$, son atómicos. Esto implica que los valores en el dominio no deberán ser listas o conjuntos de valores.

Ejemplo:

La entidad GRUPO que se muestra enseguida no está en 1NF ya que contiene valores que son conjuntos de valores atómicos

nombre	materia
Guadalupe, José, Iván	Cálculo
María, Matilde	Física

Tabla 2.1 Entidad GRUPO

Para que estuviera en 1NF, tendría que cambiarse por la entidad mostrada a continuación:

nombre	materia
Guadalupe	Cálculo
José	Cálculo
Iván	Cálculo
María	Física
Matilde	Física

Tabla 2.2 Entidad GRUPO en 1NF

Las actualizaciones representan un problema potencial si la entidad no está en 1NF. Supongamos que se quiere cambiar la primera entidad GRUPO del ejemplo, modificando la materia de Guadalupe a Física. El resultado de esta actualización es ambiguo. ¿Qué deberíamos hacer? ¿Mover a Guadalupe de un conjunto a otro o cambiar el valor de Cálculo a Física? La misma situación aplicada a la segunda entidad GRUPO evita todo tipo de confusión.

[4] ss. 1NF

Segunda Forma Normal

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

Definiremos al atributo A de la entidad R como primario si forma parte de la llave R y como no primario en cualquier otro caso.

Una entidad R en 1NF, estará en segunda forma normal[5] si no existe un atributo no primario en R que dependa parcialmente de la llave de R.

Sea $x \rightarrow A$ una relación no trivial (por ejemplo, $A \notin x$). El atributo A es parcialmente dependiente de x si $\exists\, y \rightarrow A$ para alguna $y \rightarrow x$. En la relación $x \rightarrow A$, el atributo A es totalmente dependiente si $\exists\, y \subset x$ tal que $y \rightarrow A$.

La definición dependencia parcial implica que el atributo x sea compuesto; por ejemplo, que esté formado por más atributos. De aquí podemos concluir que una entidad en 1NF que tiene una llave primaria simple (formada por un solo atributo) también estará en 2NF.

Ejemplo:

Para visualizar las anomalías que se presentan en una entidad que no cumple la 2NF, observemos la entidad ALMACEN:

depto	artículo	supervisor	cantidad
10	pinza	Beltrán	100
10	tuerca	Beltrán	350
17	desarmador	Martínez	200
17	tuerca	Martínez	200
20	tornillo	Ramírez	500
23	broca	Arenas	50
23	pinza	Arenas	180
23	taladro	Arenas	30

Tabla 2.3 Entidad ALMACEN

[5] ss. 2NF

Esta entidad está restringida por las relaciones *depto* → *supervisor* (lo que muestra que cada departamento tiene un solo supervisor) y *depto-artículo* → *cantidad* (lo cual indica que cada pareja departamento-artículo tiene una sola cantidad en el inventario). Los atributos *supervisor* y *cantidad* no son primarios. En virtud de que la llave de la entidad *ALMACEN* está formada por los atributos *depto-artículo* y hay una dependencia parcial *depto-artículo* → *supervisor*, es claro que la entidad no está en 2NF.

Si quisiéramos abrir un nuevo departamento dentro del almacén para guardar un artículo (aún no conocido) incurriríamos en una violación de la integridad de las entidades (la llave primaria de una relación no puede ser nula). Esto es un caso de anomalía al insertar una tupla.

En caso de que el departamento 20 ya no manejara tornillos y eliminàramos la tupla correspondiente, perderíamos la información acerca de l supervisor de ese departamento, por ser la única tupla existente para esa área. Esto constituye un caso de anomalía al eliminar la tupla.

Si cambiara el departamento 23 de supervisor habría que hacer el cambio en 3 tuplas; de lo contrario la relación quedaría en estado inconsistente. Esto también representa un caso de anomalía al actualizar una tupla.

Con el fin de evitar estas anomalías tendríamos que descomponer la entidad ALMACEN para que cumpla con la 2NF. Para ello separaríamos el atributo no primario supervisor de llave depto artículo (debido a que supervisor depende parcialmente de ésta) creando dos entidades:

$almacen1 \subset D_{depto} \times D_{supervisor}$

el atributo no primario *supervisor* puede coexistir con el subconjunto propio *depto* de la llave *depto artículo*, y

$almacen2 \subset D_{depto} \times D_{articulo} \times D_{cantidad}$

no incluye al atributo no primario *supervisor*. Ambas entidades se describen a continuación:

depto	supervisor
10	Beltrán
17	Martínez
20	Ramírez
23	Arenas

Tabla 2.4 Entidad ALMACEN1 en 2NF

depto	artículo	cantidad
10	pinza	100
10	tuerca	350
17	desarmador	200
17	tuerca	200
20	tornillo	500
23	broca	50
23	pinza	180
23	taladro	30

Tabla 2.5 Entidad ALMACEN2 en 2NF

Tercera Forma Normal

El tercer paso consiste en separar los campos de las segundas relaciones normales, que aunque dependan solo de una clave, deben tener una existencia independiente en la base de datos. Esto se hace de forma tal que la información sobre estos campos pueda introducirse separadamente a partir de las relaciones en las que se encuentra implicada.

En cada modelo de datos, uno o más campos de datos se agrupan para representar entidades y sus relaciones. En los agrupamientos de los campos de datos pueden darse tres tipos generales de problemas, y la eliminación de cada uno de estos da pie a las tres formas normalizadas de relaciones (tablas).

Decimos que una entidad R esta en tercera forma normal[6] si esta en 1NF y no existe algún atributo no primario en R que depende transitivamente de la llave R.

Un atributo $A \in R$ depende transitivamente de x ($x \subset R$) si ($y \subset R$) tal que: $x \rightarrow y$, y no $\rightarrow x$, $y \rightarrow A$ y $A \notin x \cup y$.

Se puede demostrar que cualquier entidad que está en 3NF, también está en 2NF.

Ejemplo:

Consideramos la entidad *asignado* $\subset D_{vuelo} \times D_{día} \times D_{num\_piloto} \times D_{nombre} \times D_{hrs\_vuelo}$. Esta entidad satisface las relaciones *num_piloto* $\rightarrow$ *nombre* y *num_piloto* $\rightarrow$ *hrs_vuelo*.

[6] ss. 3NF

vuelo	día	num_piloto	nombre	hrs_vuelo
112	Junio 6	31174	Bonilla	10572
112	Junio 7	30046	Bravo	26803
203	Junio 8	31174	Bonilla	10572

Tabla 2.6 Entidad ASIGNADO

Podemos observar parejas redundantes en los atributos *hrs_vuelo* y *nombre*. Aquí la causa del problema no es la dependencia parcial de un atributo no primario. A pesar de ello la solución es la misma: dividimos la entidad *ASIGNADO* en dos subconjuntos como se muestra a continuación:

vuelo	día	num_piloto
112	Junio 6	31174
115	Junio 7	30046
203	Junio 8	31174

Tabla 2.7 Entidad ASIGNADO1 en 3NF

num_piloto	nombre	hrs_vuelo
31174	Bonilla	10572
30046	Bravo	26803

Tabla 2.8 Entidad IDENT en 3NF

La tabla 2.9 resume las tres formas de normalización analizadas hasta ahora. Si la base de datos se diseña de acuerdo con los principios de normalización, la manipulación de datos, la cual se describe posteriormente a través de SQL, será más fácil.

Forma	Pasos
Primera forma normal	Cambiar todas las estructuras que no sean bidimensionales (es decir, grupos de repetición) en estructuras de registro bidimensionales.
Segunda forma normal	Eliminar los datos que no dependan totalmente de las llaves del registro.
Tercera forma normal	Eliminar los datos que dependan transitivamente de las llaves primarias.

Tabla 2.9 Resumen de los pasos en la normalización de los datos

Tipos de llaves

- *Llave primaria*. El atributo que identifica de manera única a un registro. La llave de una relación r es un subconjunto $K = \{B_1, B_2, ..., B_n\}$ de dominios de r con la siguiente propiedad. Para cualesquiera parejas de tuplas t_1 y t_2 en r, hay un dominio $B_i \in K$ tal que $t_1(B_i) = t_2(B_i)$. Esto es, no

debe haber dos tuplas que tengan el mismo valor en todos los dominios de *K*. Por lo tanto, con sólo conocer el valor de *K* en una tupla será suficiente para identificarla de manera única.

- *Llave candidata.* Atributo o conjunto de atributos que podrían servir como llaves primarias.

- *Llave secundaria.* Todas aquellas llaves candidatas que no se eligieron como llave primaria. Son llaves que tienen todas las características para ser primarias, pero que por una razón o por otra no fueron tomadas como tales, ya que hubo otra (s) que cumplían mejor con ese objetivo.

- *Llave extranjera o foránea.* Llave secundaria que es la llave primaria en otra relación. Son la materialización de las asociaciones entre las entidades. Son llaves que son compartidas por dos tablas para lograr una relación entre ellas.

Manejo de valores nulos

En principio, cualquier atributo en cualquier relación puede aceptar valores nulos. La interpretación de un valor nulo es: *valor desconocido en este momento* o *valor no aplicable*. La necesidad de los valores nulos proviene del hecho de que frecuentemente la información sobre el mundo es incompleta y requerimos de algún medio para manejar esa falta de información en nuestro modelo de datos.

Los valores nulos dan la posibilidad de manejar en forma adecuada las siguientes situaciones:

1. Cuando se crea una nueva tupla y el usuario no conoce los valores de los atributos en ese momento.
2. Cuando se agrega un nuevo atributo a una relación ya existente.
3. Cuando se desea agregar los valores de algún atributo (por ejemplo, cuando queremos conocer el valor promedio de todos los valores en las tuplas para ese atributo). En esta situación es deseable reconocer la presencia de valores nulos e ignorar esos tuplas para fines de calculo.

Integridad de la entidad

Ningún componente de la llave primaria puede tener los valores nulos. La razón de lo anterior se debe a que por definición toda entidad debe distinguirse de otras, o sea que debe poseer alguna propiedad que la identifique de manera única. La llave primaria en una relación desempeña el papel de identificador

único para una entidad. La presencia de valores nulos en una llave primaria sería contrasentido; equivale a decir que una entidad no puede ser completamente identificada y por lo tanto no puede distinguirse de otras.

2.3.2 Definición de la Base de Datos.

Una base de datos es una colección de tablas que contienen información relacionada, donde múltiples bases son comunes, pueden expandirse a varios discos, y un diccionario de datos almacenado en cada base de datos.

Pasos para la definición de una tabla

1. Decidir qué tipo de entidades deben ser rastreadas. Éstas deben ponerse en tercera forma normal.
2. Dividir los datos en tablas. Desnormalizar según sea necesario para mejorar el desempeño.
3. Nombrar las columnas en cada tabla, y decidir qué tipos de datos (y longitud, si se aplica) se utilizaran.
4. Decidir cuales columnas deben permitir valores nulos y cuales no.
5. Decidir si se necesita un valor por omisión o regla para cada columna. Se debe tomar en cuenta la relación entre el estado *nulo|no_nulo* y el *por_omisión|regla*.
6. Crear tipos de datos definidos por el usuario, para aquellas columnas con características similares.
7. Crear las tablas.
8. Definir reglas y valores por omisión.
9. Atar o vincular las reglas y valores por omisión a las columnas apropiadas.
10. Cargar los datos.
11. Definir y crear los índices.

Tablas. Formada por campos o atributos (columnas) y registros o tuplas (filas)

Tipos de datos. Indican la forma en que serán almacenados los datos (número entero, punto flotante, carácter, binario, etc.)

Reglas. Actúa como una mascara de edición para columna o un tipo de dato definido por el usuario.

Indices. Los índices mejoran el desempeño de las consultas:

Non-Clustered (0-250, utilizan apuntadores)

Clustered (0-1, hacen una copia física de la tabla, utiliza la totalidad de la tabla)

Valor por omisión. Indica el valor a ser ingresado en una columna si el usuario no lo especifica en su entrada de datos.

Vistas. Tabla virtual que se crea a partir de dos o más tablas diferentes. Una vista puede ser tratada como una tabla para su manejo.

Triggers. Cuida la integridad referencial de los datos (si se quiere modificar un campo llave, primero lo guarda, luego modifica todos los datos relacionados con el campo y entonces lo modifica).

Procedimientos almacenados. Programas SQL que podemos llamar por nombre (son ejecutables).

2.4 Standard Query Language

El lenguaje estándar de consultas (SQL por sus siglas en inglés) es una herramienta para organizar, gestionar y recuperar información almacenada en una base de datos informática.

El SQL es un lenguaje para DBMS relacionales que puede ser usado como lenguaje de manipulación y definición de datos. SQL es parte integral de un sistema de gestión de base de datos, un lenguaje y una herramienta para comunicarse con el DBMS.

Existen cuatro postulados del SQL para realizar las funciones básicas de manipulación de bases de datos:

- SELECT para efectuar operaciones de lectura.
- DELETE que elimina tuplas de una relación.
- INSERT es útil para agregar nuevas tuplas a una relación.
- UPDATE modifica valores en uno o más dominios en una relación.

Estos operadores relacionales son la base de todas las operaciones en los datos. Por medio de éstos se pueden revisar las operaciones de adición, borrado y cambio:

- *Adición.* Se añade un registro a la base de datos sin afectar los demás registros.

```
INSERT INTO cliente VALUES ('1245', 'Marca Acme', 'Bolívar No. 123, Col. Centro, C.P. 06060')
```

- *Borrado.* Se borra la tupla de la base de datos que contiene el registro especificado

```
DELETE cliente WHERE cliente_id = '386'
```

- *Cambio.* El registro con la llave especificada se localiza y los valores de los datos en el atributo secundario se cambian de acuerdo a los nuevos datos.

```
UPDATE cliente
SET dirección = 'Donceles 23, Col. Centro, C.P. 06060'
WHERE cliente_id = '386'
```

El SQL también sirve como lenguaje de definición de datos, empleándose para crear las estructuras de datos; es decir, las relaciones en las que posteriormente se almacenaran los tuplas. Los postulados del SQL para ese propósito son:

- CREATE sirve para definir una estructura de datos.

- DROP para eliminar una estructura de datos.
- ALTER modifica alguna característica en una estructura de datos.

En realidad hay muchos otros objetos que pueden ser creados con el lenguaje de definición de datos, además de las relaciones. Estos objetos dependen del DBMS utilizado en particular, pero en general podemos mencionar que con estos postulados se pueden definir índices, características del almacenamiento físico, vistas lógicas, restricciones de integridad implícitas y explícitas, etc.

El SQL también es útil como lenguaje de control de acceso, dando autoridad de acceso a los datos almacenados en las relaciones. Hay dos postulados de SQL que se encargan de ello:

- GRANT que otorga autoridad.
- REVOKE que quita autoridad.

Conviene aclarar que, por lo general, los mecanismos de autorización en los DBMS son muy sofisticados y suelen ir mucho más allá de la simple autorización de acceso a datos. Por los requerimientos de seguridad, hoy en día prácticamente cualquier objeto bajo el control del DBMS esta protegido y el SQL ofrece la sintaxis necesaria, por medio de estos dos postulados, con el fin de controlar el acceso a esos objetos.

PAPEL DEL SQL:

- El SQL es un lenguaje de consultas interactivas
- Lenguaje de programación de bases de datos
- Lenguaje de administración de bases de datos
- Lenguaje cliente/servidor
- Lenguaje de bases de datos distribuidas
- SQL es un lenguaje de pasarela de base de datos

CARACTERÍSTICAS Y BENEFICIOS:

- Independencia de los servidores de datos con los clientes
- Portabilidad a través de los sistemas informáticos
- Estándares SQL
- Apoyo de IBM
- Fundamento relacional

- Estructura de alto nivel en ingles
- Consultas interactivas *adhoc*
- Vistas múltiples de datos
- Lenguaje completo de base de datos
- Definición dinámica de los datos
- Arquitectura cliente/servidor

Servidor SQL

El servidor SQL maneja el espacio físico, la memoria, múltiples bases de datos y el acceso de varios usuarios manteniendo el rastro de la ubicación física actual de la información en los dispositivos magnéticos por medio de un mapeo de la descripción lógica de los datos para un almacenamiento físico de los mismos, conservando además los datos y procedimientos mas utilizados en memoria.

El servidor SQL entiende SQL:

- Compila y ejecuta procesos en lotes de instrucciones SQL
- Devuelve los resultados a los programas clientes
- Optimizar inteligentemente las consultas en base a su costo
- Automáticamente determina la forma más eficiente de llevar a cabo las tareas.
- El servidor SQL maneja por sí mismo a múltiples usuarios.
- No depende del sistema operativo del equipo de cómputo cliente para llevar a cabo las tareas.
- Mejora el desempeño, al liberar al sistema operativo de varias tareas.
- Construido alrededor de redes
- No es un producto de una sola maquina adaptado para red.
- El diseño del servidor reduce el trafico de la red significativamente.

Esquema cliente/servidor

La mayoría de las organizaciones de alto desempeño dependen de una red de área local y de redes de área amplia para sus necesidades de comunicación, un uso imaginativo de esta infraestructura puede elevar considerablemente la efectividad de una organización. El esquema cliente/servidor brinda los medios necesarios para integrar la productividad del personal individual con las necesidades específicas de procesamiento de datos para satisfacer así los requerimientos de la organización entera.

3.1 Ventajas

En la actualidad las organizaciones buscan tomar ventaja del ambiente económico y de fácil operación que brinda una estación de trabajo, también existe la gran necesidad y deseo de mejorar las aplicaciones propias que corren en el servidor, por ende las redes corporativas tienen el objetivo de conectar las estaciones de trabajo de los usuarios con un servidor. Los beneficios inmediatos se dan al integrar estas tres tecnologías: estaciones de trabajo, conectividad y servidores. Los costos de capacitación y desarrollo de nuevas aplicaciones basadas en este esquema se reducen utilizando las aplicaciones existentes en un entorno compartido. El esquema cliente/servidor brinda la capacidad de usar interfaces costeables, almacenamiento masivo de datos, conectividad y servicios de aplicación.

Almacenamiento masivo de datos

Los datos registrados como una función natural de una organización en un servidor están inmediatamente disponibles para todos los usuarios autorizados. El uso de consultas SQL para definir y manipular los datos otorga el soporte necesario para un acceso abierto para todas las aplicaciones de los equipos cliente. El SQL brinda a todos los usuarios autorizados acceso a la información a través de una vista acorde a las necesidades de su labor. Un servicio de redes transparente asegura que los mismos datos están disponibles con la misma veracidad para todos los usuarios designados para tal efecto.

Servicios integrados

En el esquema cliente/servidor toda la información que el cliente (usuario final) necesita se pone a su disposición en su propio escritorio, no hay necesidad de ningún cambio a modo terminal o de acceder al esquema de seguridad de otro equipo. Toda la información y procesos autorizados están directamente disponibles desde la interfaz de escritorio. Las herramientas locales (correo electrónico, hojas de cálculo, gráficos de presentación y procesadores de texto) tienen acceso y pueden modificar la información recibida por la aplicación y los servidores de bases de datos residentes en la red.

La figura 3.1 muestra un ejemplo típico de integración. Puede crearse un documento en un procesador de textos que incluye una entrada de un editor de imágenes, una hoja de cálculo y una aplicación de acceso a bases de datos remotas. Gracias a las facilidades del Intercambio Dinámico de Datos de Microsoft pueden combinarse datos de gráficas, hojas de cálculo y pegarse en un procesador de textos junto con la ventana de información extraída de la base de datos corporativa.

Los desarrolladores de aplicaciones utilizan la misma manipulación de objetos dentro del entorno del sistema operativo para crear nuevas aplicaciones en una fracción del tiempo que les tomaría por medio del uso de las técnicas tradicionales de programación. Las técnicas de desarrollo orientadas a objetos aumentan considerablemente el poder disponible para desarrollar aplicaciones propias a no programadores y usuarios profesionales.

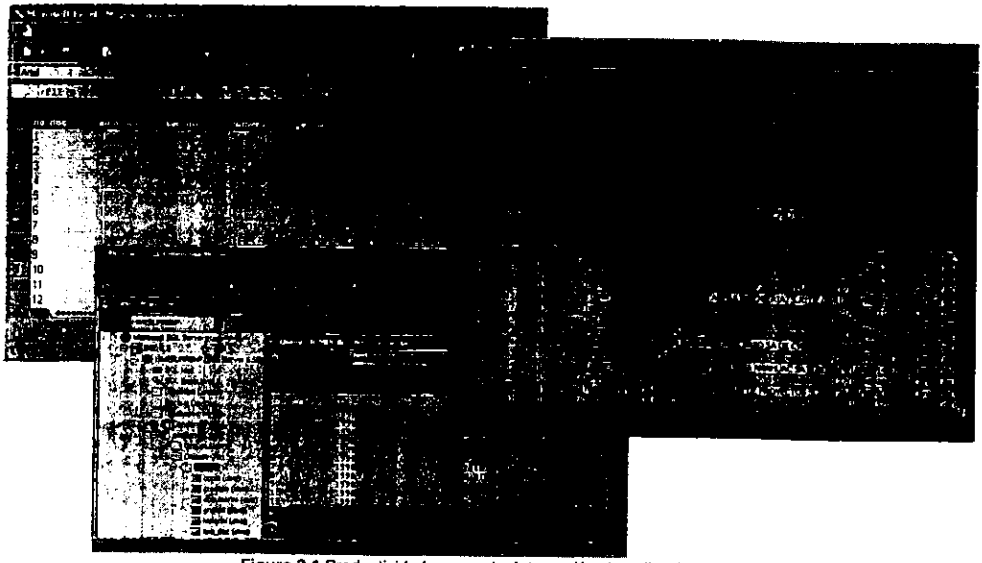


Figura 3.1 Productividad personal e integración de aplicaciones

Compartir recursos entre varias plataformas

El modelo de computo cliente/servidor otorga la oportunidad de un sistema de información verdaderamente abierto, bajo el cual pueden crearse e implementarse aplicaciones independientemente de las plataformas físicas o sistema operativo nativo, de este modo los usuarios pueden recibir servicios de clientes y acceso transparente a los servicios que otorgan los servidores de bases de datos, comunicaciones y aplicaciones. Los sistemas operativos y su soporte físico son independientes de la aplicación y se enmascaran por las herramientas de desarrollo utilizadas para construir la aplicación.

Bajo esta perspectiva, las aplicaciones en una organización se desarrollan para convivir con los procesos de la misma invocados por la existencia de un "evento" invocado por el usuario, un evento como la selección de una opción, el pulsado de un botón, la selección de un elemento de una lista o entrada en un campo de texto, mismos que ocurren sin que la lógica de la aplicación sea sensitiva de la plataforma física en la que opera.

Las aplicaciones cliente/servidor operan de dos modos, funcionan como una interfaz de usuario para una aplicación centralizada o brindan entrada de datos, almacenamiento y reportes utilizando un conjunto distribuido de clientes y servidores. En cualquiera de los casos, el uso –o aún la existencia– de un servidor es independiente de la estación de trabajo gracias al uso de interfaces como el SQL.

Intercambio e interoperabilidad de los datos

El SQL como definición de datos y lenguaje de acceso estándar ha permitido que diversas compañías desarrollen administradores de bases de datos para organizar la información en forma de tablas de este tipo, los servicios de red otorgan conectividad transparente entre el cliente y el servidor local o remoto.

La mayoría de los sistemas implementados a la fecha utilizan una plataforma única para el mantenimiento de sus datos, sin embargo, la habilidad de hacer un alto volumen de actualizaciones en múltiples locaciones y mantener la integridad de la base de datos sobre todos los tipos de error se esta haciendo disponible recientemente a través de productos de alto desempeño y recuperación. Los sistemas desarrollados hoy en día que utilizan SQL son por naturaleza transparentes a la ubicación física de los datos y a la tecnología de la plataforma de almacenamiento de los mismos, esta característica permite el movimiento de tablas a otras plataformas y ubicaciones sin afectar el código de la aplicación, lo cual resulta de gran utilidad al momento de migrar el administrador de la base de datos para adoptar alguna nueva tecnología.

Los servicios de base de datos pueden generarse en respuesta a una solicitud SQL, sin importar el motor de la base de datos del DBMS. Este motor de base de datos puede provenir de empresas como Oracle, Sybase o IBM, corriendo en plataformas como Windows NT, OS/2, UNIX o MVS. El proveedor del ambiente de desarrollo del sistema y sus herramientas deben implementar interfaces para el DBMS en cuestión y el sistema operativo de las estaciones de trabajo de los clientes. El desarrollador no necesita saber que DBMS o cual sistema operativo utiliza el servidor de base de datos, si el entorno de desarrollo no toma control del acceso directo al servidor de bases de datos, se evitara el deseo de invertir recursos en la plataforma de desarrollo en cuestión con características disponibles solamente por parte del propio proveedor. La transparencia de plataformas es esencial para que la aplicación permanezca portable, lo cual es necesario para tomar ventaja de las futuras innovaciones tecnológicas y los costos de otras plataformas, así como protección en caso de indisponibilidad del propio proveedor.

El esquema cliente/servidor brinda la capacidad de realizar solicitudes de información ad hoc, como resultado, la optimización de estructuras SQL dinámicas y el soporte de bases de datos distribuidas son cruciales para el éxito de la segunda generación de aplicaciones cliente/servidor. La primera generación introduce los aspectos operacionales de los procesos de la organización. La segunda generación es la introducción de consultas adecuadas generadas por el usuario que busca dar un valor agregado a la información disponible.

Acceso físico a datos

Cuando se utiliza SQL para accesar datos, los usuarios cuentan con información de las bases de datos en cualquier sitio de la red; la única diferencia palpable entre los distintos nodos desde los cuales se accede a la información puede ser la reducción de desempeño si el ancho de banda de la red no es el apropiado. Los datos pueden accesarse directamente de la memoria principal de un equipo, de un disco magnético o de un disco óptico con la misma instrucción SQL. Las tablas lógicas pueden accesarse -sin ningún conocimiento del orden de sus columnas, o noción de relaciones foráneas -seleccionando un subconjunto de las columnas en la tabla. Diversas tablas pueden agruparse en una vista que crea una nueva tabla lógica para su manipulación por parte de los programas de aplicación, independientemente de su formato de almacenamiento físico.

El uso de nuevos tipos de datos, como los objetos binarios de tamaño grande[7], permiten que imágenes, audio y video sean almacenados y accesados utilizando las mismas instrucciones SQL para acceso a datos convencionales.

Ubicación independiente de datos y procesos

Estamos pasando de la era de cómputo centralizado de los años 70s y 80s a una nueva era en la que usuarios familiarizados con el entorno de las computadoras personales demandan sistemas orientados a usuarios. Anteriormente, un usuario accesaba a un mainframe, miniaplicación o microaplicación. La sintaxis de acceso era diferente en cada plataforma, las teclas de función, mensajes de error, métodos de navegación, seguridad, desempeño y edición. Hoy en día los usuarios esperan un entorno estándar, accesando una aplicación desde su escritorio sin importarles la ubicación o la tecnología de los procesadores involucrados.

[7] BLOB, por sus siglas en inglés: binary large object

3.2 Componentes de aplicaciones Cliente/Servidor

Dentro del modelo cliente/servidor, el cliente es la estación de trabajo de escritorio sin importar cual fuere, cualquiera que se utilice por un único usuario es un cliente. La misma estación de trabajo, al compartir sus recursos simultáneamente con varios usuarios se convierte en un servidor. La estación de trabajo del cliente puede utilizar DOS, Windows, Windows NT, OS/2, MacOS o UNIX como sistema operativo. La estación de trabajo del cliente por lo regular brinda funciones de productividad personal tales como procesadores de texto, los cuales utilizan exclusivamente los recursos residentes en la propia estación de trabajo, cuando esta se conecta a una LAN, tiene acceso a los servicios que brinda el sistema operativo de red además de los propios. La estación de trabajo puede cargar programas o salvar documentos de un procesador de texto desde el servidor y por ende utilizar el sistema de archivos del sistema operativo de red, también puede imprimir en una impresora remota a través del mismo, incluso puede utilizarse como una terminal para accesar aplicaciones residentes en alguna minicomputadora o procesador mainframe.

En una aplicación cliente/servidor, las funciones vienen de una combinación de recursos tanto del procesador de la estación de trabajo cliente como del procesador en el servidor. Por ejemplo, un servidor de base de datos genera una colección de datos en respuesta a una solicitud SQL dada por una aplicación cliente, el procesamiento local del cliente puede calcular el valor promedio de los datos y dar formato a los mismos en la pantalla de la estación de trabajo.

El siguiente componente del entorno, el servidor, es una computadora multiusuario. No existe ninguna característica física en particular que convierta una computadora en un servidor, el equipo habrá de seleccionarse en base a las demandas de la aplicación y al costo. Los servidores para aplicaciones cliente/servidor trabajan mejor cuando están configurados con un sistema operativo que soporta memoria compartida, aislamiento de aplicaciones y multiprocesamiento, un sistema operativo con soporte multitareas permite que un proceso de mayor prioridad interrumpa o tome control del procesador cuando este efectúe alguna tarea de prioridad menor.

El servidor brinda y controla el acceso a los recursos compartidos, las aplicaciones en el servidor deben ser independientes una de otra de manera que un error en alguna no dañe a las otras. La administración de múltiples tareas a la vez asegura que ningún proceso acapare todos los recursos del servidor evitando que otorgue otros servicios para algún nuevo proceso, de ahí la necesidad de un medio que defina la prioridad relativa de los procesos en un servidor. Estos requerimientos son específicos a la implementación cliente/servidor y no a la implementación del servidor de archivos, dado que estos

últimos ejecutan la única labor de administrar archivos, pueden operar en un ambiente más limitado sin la necesidad de independencia de aplicaciones y soporte multiprocesamiento.

Los servidores convencionales (mainframe y minicomputadoras) han actuado como servidores completos para la red de terminales que soportan. Dado que la única funcionalidad disponible a la terminal de usuario es a través del servidor, la información personal así como la asociada a las aplicaciones de la organización residen en el servidor. Los servicios de red, de aplicación y de base de datos se reciben de manera centralizada por parte del servidor único.

Muchas organizaciones descargan información de sus servidores centralizados para su manipulación local en las estaciones de trabajo. Bajo el modelo cliente/servidor, la definición de servidor incluye estas mismas funciones, inclusive en las mismas plataformas u otras similares, lo que es más, el uso de servidores basados en plataformas abiertas esta facilitando el establecer los recursos de red en diversas plataformas según su naturaleza. Muchas organizaciones han integrado la funcionalidad de sus terminales tontas en sus estaciones de trabajo de escritorio con soporte de modo caracter y aplicaciones centralizadas del servidor en el mismo equipo cliente, lo anterior con la tendencia a aprovechar el poder de procesamiento de las estaciones de trabajo como parte de los sistemas de la organización.

"La red es la computadora" es la descripción más adecuada del esquema cliente/servidor. Los usuarios quieren sentir que en algún lugar de la red están disponibles los servicios que necesitan y que estos se encuentran disponibles en base a la necesidad y derecho de uso sobre los mismos, sin importar la tecnología involucrada en ello. El medio físico para lograr esta conexión es el cableado de una red de área local (LAN). Cada estación de trabajo se conecta a un cable que encamina la transmisión ya sea directamente a la siguiente estación de trabajo en la LAN o a un punto de bifurcación (concentrador), que dirige la transmisión al destino apropiado.

Existen dos topologías principales en una LAN, aquellas que usan Ethernet (bus) y las de tipo anillo, la diferencia básica funcional entre ambas es el modo en el que envían datos al cableado. Con el protocolo Ethernet el procesador intenta enviar datos al cable siempre que requiere de un servicio, las estaciones de trabajo contienden por el ancho de banda con estos intentos y el protocolo Ethernet incluye la lógica necesaria para resolver colisiones cuando estas ocurren. Por otro lado, con el protocolo anillo, el procesador solo intenta poner datos en el cableado cuando este es capaz de aceptar la información, las estaciones de trabajo se retransmiten los *paquetes* que secuencialmente dan a cada estación de trabajo el derecho de poner datos en la red.

3.2.1 El cliente

En el modelo cliente/servidor, el cliente es principalmente un consumidor de servicios dados por uno o más procesadores. El modelo brinda una separación clara de funciones basado en la idea de un servidor actuando como un proveedor de servicios respondiendo a solicitudes del cliente. Es importante entender que una estación de trabajo puede actuar como un cliente en algunas funciones o como un servidor en otras, por ejemplo, en un entorno LAN, una estación de trabajo puede actuar como cliente para un usuario mientras que simultáneamente actúa como un servidor de impresión para muchos usuarios.

Una estación de trabajo cliente utiliza un sistema operativo local para ofrecer servicios básicos e interfaces con el sistema operativo de red, este sistema operativo puede ser el mismo u otro distinto al utilizado en el servidor. La mayoría de las computadoras personales utilizan DOS o Windows 9x como su sistema operativo cliente, debido a que la mayoría de los usos locales se refieren a aplicaciones de productividad personal que no requieren del esquema cliente/servidor.

Servicios del cliente

La plataforma cliente/servidor ideal opera en un ambiente de sistemas abiertos utilizando una política de solicitudes al servidor basadas en estándares bien definidos, cuando se agregan estándares a esta política, los servidores pueden crecer y cambiar su sistema operativo y plataforma física sin alterar las aplicaciones cliente. Los clientes pueden ser estaciones de trabajo basadas en Pentium de Intel, o de avanzada arquitectura RISC, y correr las mismas aplicaciones haciendo las mismas solicitudes de servicio siempre que se sigan los mismos estándares para estos. Las aplicaciones remotas tradicionales que utilizan al cliente para servicios de presentación de operaciones únicamente envían y reciben una cadena de caracteres desde y hacia el servidor, toda la lógica de la aplicación residiendo en este último y es de este modo en que muchas organizaciones utilizan las estaciones de trabajo hoy. El CPU de mayor costo en el servidor se utiliza para manejar funciones que se podrían operar en la estación de trabajo a un costo mucho mayor.

La funcionalidad de los procesos del cliente puede expandirse al agregarle lógica no implementada en la aplicación del servidor. Edición local, entrada automática de datos, archivos de ayuda y otros procesos lógicos pueden adicionarse a los procesos existentes en el servidor. Si se hace posible detectar errores de captura por ejemplo en el cliente o funciones tales como la ayuda en línea se dejan este, la carga de trabajo del servidor disminuye.

El llenado de formularios múltiples involucra generalmente entradas de datos en diversos sistemas o aplicaciones de cómputo. La captura de dichos datos por su propia fuente o en una función común de entrada que distribuya la información a otras funciones de datos puede reducir costo y errores. Idealmente la información se captura por el individuo o proceso responsable por la creación de los datos, esto permite que el usuario con la capacidad de hacer correcciones las haga inmediatamente, dejando al servidor del grupo de trabajo LAN la captura los datos y su almacenamiento. Cuando se define un proceso de captura de datos provenientes de un formulario existente, los datos capturados se integran automáticamente al formulario, el cual a su vez es actualizado por el usuario con datos reales, de este modo, la información se captura solo una vez y cada proceso de la organización puede utilizar los mismos datos. La información se hace disponible inmediatamente después de capturada y puede distribuirse electrónicamente a todos los usuarios autorizados.

La tecnología de las estaciones de trabajo debe su efectividad al apoyo que otorga al nuevo paradigma de ampliar la capacidad de los empleados, brinda las facilidades de un entorno gráfico de trabajo al acceso y administración de toda la información necesaria para completar los procesos de la organización. La capacidad de una poderosa estación de trabajo para recomendar y tomar decisiones basadas en los antecedentes históricos de las operaciones dentro de una organización, pueden reducir considerablemente los costos y mejorar los servicios al reducir el ciclo de toma de decisiones.

Solicitud de servicios

Las estaciones de trabajo clientes requieren de recursos por parte del servidor al cual están conectados, independientemente si dicho servidor cuenta con el mismo tipo de procesador o un sistema operativo de red el formato de la solicitud por parte de la aplicación es el mismo. El sistema operativo de red traduce o agrega las características necesarias por parte del recurso solicitado a la solicitud de la aplicación.

Comunicación entre procesos[8] es el término general utilizado para describir la comunicación entre los procesos en ejecución. En el modelo cliente/servidor, estos procesos pueden residir en la misma computadora o en cualquier equipo dentro de la red.

[8] IPC, por sus siglas en inglés: Interprocess communication

El servicio más básico del sistema operativo de red es la *redirección*, Este servicio intercepta llamadas del sistema operativo del cliente y las redirecciona al sistema operativo del servidor, de este modo, las solicitudes de directorios de disco, archivos, impresoras, colas de impresión, dispositivos seriales, programas de aplicación y direcciones específicas de red se capturan por el programa de redireccionamiento y se canalizan a través de la red a la ubicación correcta del servidor. También es posible que algunos de estos servicios se lleven a cabo por la estación de trabajo cliente, por ejemplo, las unidades de disco locales pueden etiquetarse como A: y C: mientras que las unidades remotas como D:, E: y F:; en tal caso la dirección funciona para este caso como sigue:

- Cualquier solicitud para las unidades A: o C: se pasa directamente al sistema local de archivos por el programa de redirección. Las solicitudes de otros servicios se pasan al sistema operativo del servidor, las impresoras se accesan a través de puertos serie y paralelo virtuales definidos por el programa de redirección del sistema operativo de red.
- El programa de servicios del sistema operativo de red construye la llamada a procedimientos remotos para incluir la solicitud de la aplicación cliente recibida.
- El sistema operativo de red del servidor procesa la solicitud como si esta se ejecutara de modo local y envía la respuesta de vuelta a la aplicación.

Novell comercializo este concepto de redirección para las plataformas Intel y MS-DOS y ha sido adoptada por todos los sistemas operativos de red y sistemas de archivos UNIX. La simplicidad de ejecutar llamadas estándar en una red virtual de servicios es su ventaja principal.

Visión de la organización

Es importante destacar a los diseñadores de aplicaciones que la visión que del sistema tienen los usuarios es a través de la estación de trabajo del cliente. Cualquier innovación tecnológica en el servidor pasará por alto ante un diseño deficiente o mala implementación en el escritorio del cliente, resultando en una percepción desfavorable por parte del usuario final de la aplicación entera.

3.2.2 El servidor

Un servidor administra los recursos de aplicaciones, archivos, bases de datos, impresión, fax, imágenes, comunicaciones y seguridad. Es importante entender al servidor como un concepto de la arquitectura de red y no como una implementación física. Las funciones de cliente y servidor pueden llevarse a cabo por el mismo dispositivo físico, bajo la tendencia de sistemas abiertos cada dispositivo tiene el potencial de operar como un cliente o un servidor en virtud a las solicitudes de servicio.

Los servidores de aplicación dan a la organización la funcionalidad de aceptar la operación de la estación de trabajo cliente. En el modelo cliente/servidor estos servicios pueden darse para una función parcial o completa invocada a través de un servicio de comunicación entre procesos, ya sea una solicitud basada en mensajes o como una llamada remota a un proceso[9].

Procesamiento de solicitudes

Las solicitudes son administradas por el sistema operativo de red residente en el equipo cliente, este se encarga de dar formato a la solicitud convirtiéndola en la llamada RPC adecuada y la envía a la capa de aplicación de la pila de protocolos de comunicación del cliente para de ahí pasar a la pila de protocolos del servidor.

Servicios de archivo

Los servicios de archivo controlan el acceso a los directorios virtuales y archivos localizados en la estación de trabajo del cliente y hacia el dispositivo de almacenamiento del servidor. Estos servicios se otorgan a través de la programación de redireccionamiento implementado como parte del ambiente de operación de la estación de trabajo del cliente. Como se describe en el punto 3.2.1, todas las solicitudes se mapean en el conjunto virtual de recursos y se redirecciona según sea necesario al servidor local o remoto apropiado. Los servicios de archivos dan este soporte a través del procesador en el servidor remoto, en una implementación típica la programación, datos compartidos, bases de datos y respaldos se almacenan en discos, cintas y dispositivos ópticos administrados por el servidor de archivos.

Los respaldos en el servidor pueden calendarizarse y monitorearse por un técnico capacitado, los respaldos en la estación de trabajo del cliente pueden administrarse desde el servidor y almacenarse en

[9] RPC, por sus siglas en inglés: Remote Procedure Call

este para facilitar su recuperación. Para este tipo de operaciones por lo regular se utilizan cintas o dispositivos ópticos, con soporte para varios usuarios. La ubicación del servidor y los respaldos en un lugar seguro ayuda a prevenir el robo o la destrucción accidental de la información. A partir del uso de tecnología de multimedia e imágenes por parte de muchas organizaciones, los dispositivos de almacenamiento masivo ópticos se implementan más adecuadamente como servidores compartidos.

Servicios de impresión

Los servidores de impresión dan soporte a la recepción de documentos de los clientes, agregándolos a la fila de impresión, dándoles determinadas prioridades y ejecutando la lógica específica de los controladores de impresión para explotar las características únicas de cada impresora. Un soporte adecuado de impresión incluye recuperación de errores y notificación de estos al operador con instrucciones para el restablecimiento de las funciones de impresión.

Servicios de base de datos

Los servidores de bases de datos eran en un principio en realidad servidores de archivos con una interface diferente. Productos como dBASE, Clipper, FoxPro y Paradox ejecutaban el DBMS en el equipo cliente y utilizaban los servicios de archivo del servidor para el acceso a registros y administración del espacio libre en disco, esto los convertía en implementaciones más poderosas de los modelos originales de archivos planos con extracción de índices para acceso directo a los registros. El control de accesos se administra por parte del programa de aplicación, el cual recibe solicitudes de bloqueo de registros revisándolos, y por el servidor de base de datos, el cual bloquea una tabla para su modificación o consulta siempre que se da una aprobación del bloqueo por parte de la aplicación; dado que el acceso es a nivel registro, todos los registros que satisfagan la llave primaria deben devolverse a la estación de trabajo cliente para su filtrado. No existe la posibilidad de ejecutar código de procedimientos en el servidor, asociar tablas o discriminar filas antes de devolverlas a la estación de trabajo. Esta falta de capacidad imposibilita el bloqueo de registros cuando varios clientes accesan a las mismas bases de datos e incrementan el tráfico en la red cuando se devuelven al cliente muchas filas innecesarias solo para ser rechazadas.

La falta de una lógica de ejecución por parte del servidor evita que estos productos cuenten con algún esquema de recuperación en caso de contingencia, por lo que los sistemas que operan en este tipo de

ambientes requieren de programadores experimentados para llevar a cabo una recuperación después de una falla.

Los DBMS cliente/servidor como Sybase, SQL Server, Ingres, Oracle e Informix permiten al servidor ejecutar solicitudes SQL generadas en la estación de trabajo del cliente. Los servicios de archivo se utilizan exclusivamente para efectos de disponibilidad de espacio y servicios básicos de directorio, el resto de los servicios se generan directamente por parte del servidor de base de datos. Los sistemas de bases de datos relacionales son la tecnología actual para la administración de información.

Servicios de comunicación

Las aplicaciones cliente/servidor requieren de servicios de comunicación LAN y WAN. Los servicios básicos de una LAN esta integrados en el propio sistema operativo de red. Los servicios WAN se brindan por una serie de productos de comunicación independientes. El punto 3.2.3 trata a detalle lo relacionado al modelo cliente/servidor.

Servicios de seguridad

Las aplicaciones cliente/servidor requieren de servicios de seguridad similares a aquellos de un ambiente compartido de datos. Debe requerirse a cada usuario iniciar su sesión de trabajo bajo un nombre de usuario y una contraseña, si estas últimas se hacen visibles a usuarios no autorizados, el servidor de seguridad deberá insistir en que las contraseñas se cambien regularmente. Este entorno implica que una validación de usuario y contraseña se utiliza para ganar la autoridad de accesar a toda la información y procesos que requiera el usuario según su actividad. Dada la posibilidad de que los datos se almacenen en un área menos segura, debe existir la posibilidad de almacenar la información en forma encriptada, requiriéndose de una determinada combinación de usuario y contraseña para traducir la información asociada.

3.2.3 Conectividad

El modelo de referencia OSI mostrado en la figura 3.2 es el entorno de trabajo estándar para la operación interna de un sistema en red. La existencia de ambientes LAN heterogéneos en grandes organizaciones hacen a la operación entre equipos una realidad práctica. El modelo OSI define siete capas de protocolos y especifica que cada una de ellas debe estar aislada de las otras por una interfaz bien definida.

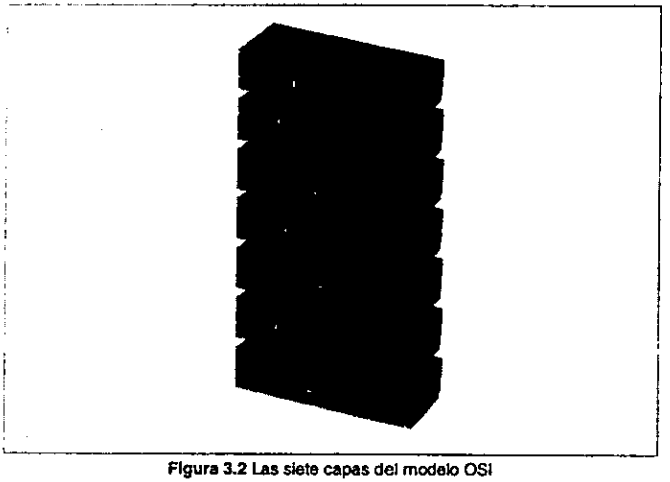


Figura 3.2 Las siete capas del modelo OSI

Capa física

La capa física es el mas bajo de los niveles del modelo OSI y define las características físicas y eléctricas de las conexiones que conforman a la red. Incluye especificaciones de interfaces así como especificaciones detalladas para el uso de cables par trenzado, fibra óptica y coaxial. Los estándares de interés en esta capa para las aplicaciones cliente/servidor son el IEE 802.3 (Ethernet) y IEE 802.5 (anillo) que definen los requerimientos para la tarjeta de interface de red y los requerimientos lógicos para la capa de control de acceso a medios. Otros estándares aquí incluyen las interfaces seriales EIA232 y X.21.

Capa de ligado de datos

La capa de ligado de datos define los paquetes básicos de información que se espera entren o salgan de la red física. Patrones de bits, métodos de codificación y tokens se definen en esta capa. La capa de ligado de datos detecta errores y los corrige solicitando la retransmisión de paquetes o mensajes dañados. Esta capa se divide en dos subniveles: el control de acceso a medios y el control de ligado lógico, el primero tiene la responsabilidad del acceso a la red en cuanto a reconocimiento de tokens, manejo de colisiones y control de red, por último la capa de ligado lógico opera sobre la de acceso a medios enviando y recibiendo paquetes de datos y mensajes.

Las diferentes topologías de red definen el formato de los paquetes que se comunican entre la capa de acceso a medios y la capa de red. Los formatos internos son diferentes y sin conversión alguna las estaciones de trabajo no pueden operar con ningún otra que opere con una definición diferente.

Capa de red

La capa de red es responsable de cambiar y dirigir mensajes al destinatario que corresponda. Coordina los medios necesarios para el direccionamiento y envío de mensajes. Para cada sistema otorga una única dirección de red, determina la ruta para transmitir datos a su destino, particiona grandes segmentos de información en menores paquetes de datos y lleva a cabo el control de flujo de información.

Capa de transporte

Cuando un mensaje contiene más de un paquete, la capa de transporte envía secuencias de paquetes de mensaje y regula el flujo de tráfico de salida. La capa de transporte es responsable de asegurar que la transmisión de datos este completa y libre de errores de extremo a extremo. La capa de transporte mantiene sus propias direcciones que se mapean en direcciones de red. Dado que la capa de transporte sirve a los procesos de los sistemas, diferentes direcciones de transporte (origen y destino) pueden compartir una misma dirección de red.

Capa de sesión

La capa de sesión brinda los servicios que permiten a las aplicaciones corriendo en dos procesadores coordinar su comunicación en una única sesión. Una sesión es un intercambio de mensajes –un dialogo entre dos procesadores. Esta capa ayuda a crear la sesión, informa a una estación de trabajo si la otra termina la sesión y termina la sesión actual si así se le requiere.

Capa de presentación

La capa de presentación es responsable de traducir los datos de la lógica interna de un procesador a la del otro.

Capa de aplicación

La capa de aplicación es aquella con la cual se comunica directamente la aplicación en el procesador. Los códigos de programación de una interface de usuario se definen en esta capa. Los mensajes entran a la pila del modelo OSI en este nivel, viajan a través de todas las capas hasta la capa física, a través de la red hasta la capa física del otro procesador, y de vuelta nuevamente a las capas de red hasta la capa de aplicación y el programa del procesador en el equipo de destino.

La reingeniería como parte del proceso del cambio

La reingeniería consiste en la transformación radical de una organización de una estructura rígida definida por departamentos a una estructura flexible definida por procesos, la meta de la ingeniería, por otro lado, es aprovechar al máximo la fuerza y los recursos de la naturaleza, los ingenieros en computación no tratan con dichos fenómenos naturales, pero siguen las mismas premisas que los ingenieros convencionales, muchas de las cuales son relevantes para la reingeniería de sistemas.

4.1 Lo establecido y la reingeniería

La reingeniería puede definirse como una operación o por el efecto que tiene en una organización. Operacionalmente, la reingeniería de sistemas es la identificación de procesos básicos en una organización y la implementación de dichos procesos. Por sus efectos, la reingeniería de sistemas es el rediseño estructural de una organización para permitirle llevar a cabo sus procesos básicos con una mejora dramática. Estas definiciones son complementarias y permiten que diferentes grupos se concentren en los aspectos de la reingeniería que más les interesen. Para el presente caso de estudio, la definición operacional es la más importante.

Una organización se define desde el punto de vista de la reingeniería como un conjunto de procesos llevados a cabo por la organización en su conjunto, generalmente de 10 a 20 procesos definen las actividades de ésta, aunque esto no es una regla.

En la especificación de procesos lo importante únicamente es la esencia de los mismos, se definen los pasos del proceso y el orden en que son llevados a cabo, indicando también cuales se llevarán a cabo simultáneamente, sin embargo, no es de interés alguno si cada paso del proceso se efectuará por alguien diferente o por la misma persona, o sí por parte del personal en su conjunto algunos y otros con la ayuda de sistemas expertos.

Sólo hasta comenzar el diseño de los nuevos sistemas de información se definen a detalle los datos en los que se apoya un proceso, creándose entonces la representación de la estructura del mismo por medio de un programa de aplicación que seguirá las especificaciones del proceso solicitando información al usuario cuando así lo requiera. En esta etapa, nuestra implementación es más bien esquemática, el analista determina las entradas necesarias pero sin considerar si estas proceden de personas o sistemas expertos, esta etapa se denomina *implementación esquemática*.

En la definición por efecto, los términos importantes son el *rediseño estructural* y la *mejora dramática*. Durante la implementación esquemática no interesa quien hace que cosa, pero para llevar a cabo una implementación eficiente es necesario involucrar a los participantes y esto está ligado al rediseño estructural de la organización.

Es común confundir a la reingeniería con la optimización gradual de los procesos de una organización. Por el contrario, la reingeniería es un esfuerzo único cuya meta es el mejorar dramáticamente el desempeño de una organización. Las metas a lograr con la reingeniería deben estar claramente estipuladas, si son reales puede llevarse a cabo un planteamiento que logrará su cumplimiento.

La reingeniería puede efectivamente ahorrar recursos, pero sólo si se hace adecuadamente. La reducción de insumos es positiva hasta cierto punto solamente, el mayor problema se presenta cuando se lleva a cabo de una manera irracional y esto se confunde con reingeniería, generalmente el resultado es que empleados que brindan valor agregado a los productos y servicios se eliminan y aquellos que no lo hacen permanecen; la pérdida de empleados experimentados y la baja moral que esto conlleva pueden hacer decaer los beneficios planteados. El recurso más valioso de una organización es el conocimiento colectivo de su personal.

Dado que la reingeniería debe considerar todos los procesos de la organización ésta debe preceder a todo esfuerzo de cambio en la misma, evitando llevar a cabo otra iniciativa importante paralelamente al proceso de reingeniería. Suponiendo que una organización considera aplicar reingeniería y por ejemplo Administración Total de Calidad[10], ambas debieran iniciar simultáneamente pero subordinando el esfuerzo de TQM al plan global de reingeniería, lo que es más, los equipos de trabajo, parte esencial de la reingeniería pueden funcionar como círculos de calidad de manera natural. La reingeniería no es una actividad estrictamente aislada, su objetivo es la definición de una serie de procesos que habrán de implementarse a la luz de toda técnica que favorezca la efectividad de sus resultados, entendiéndose por

[10] TQM, por sus siglas en inglés

esto una reducción de costos y tiempos de procesos que lleven a una mejora de la calidad. La eficiencia no se logra invirtiendo un gran esfuerzo en mejoras progresivas a procesos considerados por la reingeniería, si estas mejoras se hacen en aquellos procesos que ya no existirán en la propuesta de reingeniería se habrán desperdiciado esfuerzos.

La reingeniería busca reducir la especialización aunque también se aplica lo contrario. La razón de esta confusión es que el término *especialización* puede interpretarse de dos distintas maneras: como la ejecución de una simple tarea una y otra vez, o como la adquisición de grandes cantidades de conocimiento específico, como lo requiere por ejemplo una cirugía de transplante de órganos, la reingeniería tiende a eliminar la especialización en este primer sentido pero la especialización en el segundo sentido expuesto adquiere mayor importancia. En una organización de alto desempeño un equipo lleva a cabo un proceso, por ejemplo ensamblar automóviles, es claro que algunos miembros del equipo de ensamblado serán mejores en ciertas tareas que otros y que se desarrolla una especialización natural (bajo el primer sentido del término), sin embargo, si faltase uno de los miembros del equipo, el resto debiera ser capaz de cubrirlo, implicando una especialización en el segundo sentido expuesto; en otras palabras, los empleados de una organización de alto desempeño habrán de especializarse en más de una tarea. La necesidad de una especialización a nivel procesos es aún más marcada en el desarrollo de un sistema de información en el cual se requiere de conocimiento técnico especializado para lidiar con el desempeño y confiabilidad del sistema y por otro lado capacidad de diseño de la interfaz de usuario del mismo.

En una organización existen dos tipos de administración: aquella que agrega valor a los productos y servicios y aquella que lo quita, solo esta última habrá de eliminarse. A medida de que se identifican las funciones en una organización con distintos departamentos se adquiere la necesidad de que los administradores coordinen el trabajo de los departamentos separados, en adición la administración deberá evaluar situaciones específicas para evitar perdidas. Muchas veces el costo de la prevención supera por mucho cualquier perdida potencial. Los coordinadores o supervisores no brindan un valor agregado a los productos, sin embargo, un administrador que realiza un análisis costo-beneficio para un nuevo producto y en base a dicho análisis determina no producir, lleva a cabo una labor de valor agregado pues si el producto saliera al mercado y causara una perdida esta tendría que compensarse por el resto de los productos de la compañía. El administrador "agregó" la cantidad proyectada como pérdida al valor del resto de los productos, lo que es más, en una organización orientada a procesos en la cual estos corresponden a proyectos separados de duración limitada, la reubicación del personal en nuevos proyectos forma una parte muy importante en la administración de la organización.

El esfuerzo de reingeniería debe llevarse a cabo rápidamente, pero también hay que considerar que la reingeniería propone un cambio en la cultura entera de la organización, y este cambio no habrá de aceptarse de la noche a la mañana. En particular deben considerarse dos aspectos cuidadosamente, los empleados que se ven afectados por la reorganización impuesta por la reingeniería deben ser reubicados y los procesos diarios de la organización no habrán de detenerse por la reingeniería. Por lo tanto hay una necesidad de iniciar con un reacondicionamiento bien planeado de los empleados y el cambio debe llevarse a cabo con lujo de detalle. Es un mito que los empleados no puedan ajustarse, lo lograrán si se pone el cuidado suficiente en convencerlos de los beneficios del cambio y si este hace su labor más interesante.

La reingeniería no es una panacea, cualquier cambio externo puede tener un efecto determinado en la organización, sin embargo, una organización de alto desempeño tiene la flexibilidad necesaria para actuar rápida y efectivamente ante los cambios externos, por tanto, una organización con reingeniería será siempre mejor que otra que no la ha aplicado.

El primer paso hacia la reingeniería será el imponer un cambio cultural, el primer efecto de dicho cambio es la visión de la administración con una actitud científica. La ciencia busca establecer relaciones causa-efecto, pero la determinación de estas relaciones no siempre es fácil, es común en las ciencias naturales, donde a través de experimentos controlados se puede excluir todo aquello que es irrelevante; en las ciencias sociales aun los efectos no son del todo claros y cuando lo son pueden tener un gran número de causas que intervienen en ellos de manera aislada o colectiva. Esta visión de la reingeniería explica por que ha sido más exitosa en la industria de la transformación, aquí las ciencias naturales se ven más involucradas y podemos darnos una idea clara de lo que funciona y lo que no, es más fácil establecer el efecto que tendrá una mejora en el costo de producción, pero aún aquí los efectos sociales no pueden ser totalmente eliminados.

La tecnología propuesta por la ingeniería en computación se relaciona casi exclusivamente a procesos. La definición estricta de un proceso siempre ha sido importante para la ingeniería de sistemas, lo cual la convierte en la más apta para apoyar a los expertos de la organización en la determinación de los cambios a realizarse en la reingeniería de procesos.

4.2 El proceso de la reingeniería

Apoyo administrativo. La participación y apoyo pleno de la directiva de la organización es esencial para la reingeniería, debiendo ser esta completamente informada del progreso del esfuerzo de reingeniería. La reingeniería de una organización esta condenada al fracaso si los cambios se toman con recelo, si este es el caso la cultura institucional habrá de modificarse primero.

Identificación de procesos. Bajo el paradigma de una estructura definida por departamentos hay un cierto entendimiento de procesos, pero estos debieran verse como el movimiento de una operación de un sitio a otro, donde estos últimos están bajo el control de diferentes departamentos. Un proceso no solo se define en función de los de sitios por donde fluye dado que un sitio puede servir a más de un proceso, por lo tanto la identificación de procesos puede ser más difícil de lo que parece. Un proceso esencial para el buen funcionamiento de una organización de alto desempeño se basa en un flujo inteligente de información y en la evaluación de la misma, lo más probable es que este proceso ya exista en la vieja organización.

Especificación de procesos. En esta etapa un proceso se separa en tareas que lo componen, cada una de estas se examina para determinar si es realmente necesaria, eliminándose las que no lo son para después reestructurar las restantes. En la especificación reestructurada de procesos tiene que estar claramente identificado el orden en el cual se llevarán a cabo las tareas que lo componen y cuáles de estas pueden llevarse a cabo paralelamente. La especificación formal deberá complementarse con diagramas que ayuden a entender intuitivamente los procesos.

Alternativas de implementación. Los ingenieros civiles presentan a sus clientes distintas propuestas preliminares para que así decidan el diseño a efectuar, este mismo enfoque puede aplicarse a los procesos de las organizaciones, pero en esta etapa sólo habrán de mencionarse las alternativas de diseño sin determinar aún cual se implantara.

Análisis costo-beneficio. Los costos de cada una de las alternativas del paso anterior se establecen, lo cual es relativamente fácil, sin embargo, el establecer los correspondientes beneficios no lo es tanto pues depende de la dificultad de asignar un valor económico a un beneficio intangible como el buen servicio al cliente. Cuando se concluye un trámite en 7 horas en vez de 7 días, podemos medir la reducción de las cargas al inventario, y el efecto de finiquitar una operación 6 o 7 días antes, pero no podemos predecir si esto atraerá más al cliente a la organización, y si este es el caso, como expresar este beneficio cuantitativamente. Si una de las alternativas es una mejora sobre otra en todos los aspectos entonces es claro cual será la elegida, pero por lo regular se tiene que sopesar un mayor costo y mayores beneficios

intangibles de una alternativa contra un menor costo y menores beneficios de la otra, en estos casos la decisión habrá de basarse en algún tipo de análisis matemático.

Definición de la infraestructura. La infraestructura consiste de un sistema de comunicaciones y un sistema de información. El costo de instalación y mantenimiento de una red de área local (LAN) cuesta menos que una de área amplia (WAN), también un sistema distribuido presenta más problemas que uno centralizado. En esta etapa los costos de la infraestructura se determinan por el conjunto total de procesos involucrados por la alternativa elegida en el paso anterior. Los procesos, hasta ahora considerados aisladamente, se integran en este punto. Es posible que se cambie la elección de esta alternativa para algunos de los procesos que involucra en busca de reducir los costos de infraestructura.

Asignación de prioridades. Para decidir el orden en el cual habrán de implementarse los procesos rediseñados hay que considerar varios factores. Primero, un proceso para el cual su rediseño produce una mejora considerable en el índice costo-beneficio es buen candidato para implementación inmediata. Segundo, un proceso que no interactúa con otros es un buen candidato. Tercero, si dos procesos interactúan ampliamente entre ellos, ambos deberán implantarse al mismo tiempo. Cuarto, cuando la implementación de los procesos rediseñados impone un cambio en la estructura de un departamento, este deberá manejarse con especial cuidado teniendo un efecto mayor en el orden en que se implementarán el resto de los procesos. Quinto, las prioridades de implementación dependen del efecto que tengan en la infraestructura, por lo regular la reingeniería requiere combinar varias bases de datos independientes en un sistema de información compuesto, este proceso de implementación debe sincronizarse con la implementación de los procesos rediseñados. Similarmente un proceso que depende de una WAN no puede implementarse hasta que la WAN este activa.

Capacitación. Después de que un proceso pasa del equipo de reingeniería al equipo de producción, este deberá ser capaz de dar un buen seguimiento al proceso. No solo deberán llevar a cabo operaciones que antes se efectuaban en diferentes departamentos bajo estricta supervisión, sino que habrán de hacer cada una de las tareas del proceso y sin mayor supervisión, lo que implica que la responsabilidad del proceso rediseñado recae en el equipo de producción.

Implementación. La implementación puede introducirse de dos modos. El primero consiste en seleccionar un proceso y ejecutarlo de la manera tradicional y con el proceso rediseñado, este último con un subconjunto menor de tareas, de este modo se permite que el equipo de producción se acostumbre al proceso antes de que este tome la carga completa de trabajo; la desventaja es que se mantiene ambas plantillas de personal. El otro enfoque propone cambiar totalmente un proceso de la manera tradicional

a la propuesta de reingeniería, permitiendo que el equipo de producción se adapte gradualmente a la nueva estructura, llevando a cabo las mismas tareas que antes con un cambio gradual en el modo de operar según la propuesta del proceso rediseñado.

Evaluación de la reingeniería. La prueba piloto cumple con dos propósitos, el primero es demostrar al cuerpo administrativo que la reingeniería no es solo propuestas, pero resulta contraproducente elegir para la implantación piloto el proceso para el cual se espera una mejora más dramática, de este modo todo lo que continuase parecería de menor importancia. El segundo propósito es el ganar experiencia en el propio proceso de reingeniería y usarla para mejorarlo. El proceso o procesos seleccionados para implementación piloto deben ser moderadamente complicados, uno fácil no mostraría problemas y por otro lado, uno difícil desvalorizaría al equipo de reingeniería que aún esta ganando experiencia.

Una organización de alto rendimiento, estructurada en función a los procesos que la componen producida con reingeniería es necesaria para sobrevivir en un mundo postindustrial, pero la estructura por procesos y conocimientos no son suficientes, también debemos considerar el servicio a los clientes. Los equipos de producción idóneos buscan superar las necesidades y expectativas de sus clientes.

4.3 Principios de ingeniería en el desarrollo de software

Desde los primeros días del cómputo, el desarrollo de programas de aplicación se ha considerado en una crisis provocada por altos costos, retrasos y productos poco confiables, estas no son características de la ingeniería tradicional, pero el costo, la oportunidad y la confiabilidad son de amplia consideración para la ingeniería como en todo proyecto nuevo que ha de ser solucionado. Por lo tanto, la "crisis del software" no tiene mucho que ver con la naturaleza de la ingeniería de programación, pero si con la naturaleza de los problemas a los que habrá de enfrentarse y con el punto de vista de ingeniería que los analistas y programadores han de tomar para atacarlos; los resultados mejoran cuando se tiene una aproximación sistemática para el desarrollo de programas y esta aproximación es la ingeniería de programación.

Aunque el desarrollo de programas de aplicación aún se asocia con problemas, la situación mejora a medida que la ingeniería de programación madura como disciplina y un mayor número de analistas entienden su importancia. La ingeniería de programación tiene que ver con la adaptación de los principios generales de la ingeniería al desarrollo de programas de aplicación y la aplicación de las aportaciones de la ciencia de la computación en esta materia.

La ingeniería se define como la profesión en la cual el conocimiento de las matemáticas y las ciencias *naturales*, guiado por el estudio, la experiencia y la práctica, se aplican con juicio para desarrollar métodos para utilizar económicamente los *materiales y fuerzas de la naturaleza* para el beneficio de la humanidad[11].

En esta definición se destaca la diferencia principal entre la ingeniería convencional y la de programación; en otras palabras, los ingenieros convencionales producen objetos tangibles y la ingeniería en programación artefactos intangibles. La característica principal de los objetos tangibles es su falta de permanencia: los materiales naturales están sujetos a desgastes y los objetos construidos de tales materiales se deterioran con el tiempo y eventualmente fallan, por otro lado, las fallas en los programas de cómputo se deben exclusivamente a errores de diseño o de juicio, por lo cual las expectativas de la ingeniería de programación difieren de las de la ingeniería convencional, por otro lado, tienen algo en común: los ingenieros de sistemas tienen las mismas actividades guía que los otros ingenieros.

[11] Según el Consejo de Ingeniería para el Desarrollo Profesional (Reino Unido)

El campo de trabajo de la ingeniería es muy variado, desde la investigación, la planeación y diseño hasta la supervisión de los procesos de manufactura o construcción de proyectos, pero cada ingeniero debe llevar a cabo al menos alguna de las tareas mencionadas a continuación:

- Definición de requerimientos y desarrollo de planes de trabajo
- Resolución de conflictos en los requerimientos y establecimiento de prioridades
- Uso de estándares
- Anticipación a los cambios y posibilidad de ajustes
- Prevenir lo inesperado
- Puesta en práctica de la teoría
- Conversión de modelos en prototipos
- Control de calidad
- Selección de herramientas
- Cooperación con otros especialistas
- Administración de las tareas mencionadas
- Compromiso con los avances tecnológicos

Estas tareas definen uno de los dos soportes de la ingeniería de sistemas como se muestra en la figura 4.1.

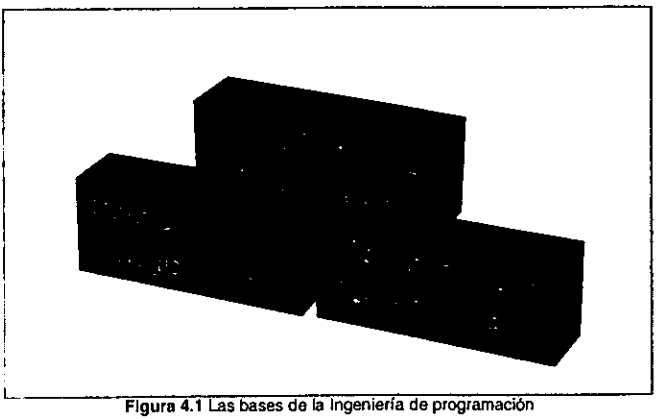


Figura 4.1 Las bases de la ingeniería de programación

1. *El desarrollo de la ingeniería sigue un plan de acuerdo a los requerimientos cuantitativos, y los ingenieros asumen la responsabilidad de seguir dicho plan.* Antes de construir un objeto debe haber un claro conocimiento por parte de todos los involucrados de lo que habrá de construirse y del proceso de construcción a seguir. El proceso de construcción se define por un plan y el objeto a construir por requerimientos, estos últimos deben poder verificarse, es decir deben expresarse de manera cuantitativa, como funciones cuando indican lo que habrá de hacerse y como variables (no funcionales) cuando dependen de una situación. Por ejemplo, un requerimiento funcional para una biblioteca es el otorgar el servicio de préstamo, un requerimiento no funcional puede ser que cada préstamo no tome más de un minuto, y que el 95% de casos los usuarios no deben esperar a ser atendidos más de dos minutos. En lenguaje matemático, un requerimiento funcional relaciona una entrada x con un resultado $f(x)$, donde f es el nombre de una función. Por ejemplo, el valor de $cos(x)$ es el coseno del ángulo x y el valor de *préstamo(Titulo, Usuario)* define si el libro *Titulo* es prestado a *Usuario*.

2. *Los requerimientos se jerarquizan de acuerdo al análisis costo-beneficio y el plan de desarrollo tiene una estructura ascendente.* En muchas ocasiones los requerimientos resultan contradictorios, por ejemplo, un requerimiento altamente necesario esta en conflicto con otro de bajo costo. Dado lo anterior es esencial determinar indicadores que permitan establecer la relación costo-beneficio de cada uno de los requerimientos para así compararlos como por ejemplo la confiabilidad, el costo, la versatilidad y la facilidad de capacitación para así asignar prioridades a los requerimientos. La confiabilidad es de especial cuidado aquí, indica hasta que grado los procesos pueden basarse en determinado insumo proveniente de otro proceso, cuando el objeto a producir esta compuesto de varias partes el orden en el que se construyen dichas partes puede producir un producto útil, aún cuando el proyecto global no pueda terminarse como se planeo originalmente.

3. *Los estándares se usan siempre que sea posible aplicarlos, cada discrepancia con respecto a algún estándar aplicable debe justificarse explícitamente.* Los estándares se fijan para permitir que los objetos o sus partes estén preparados para ser sustituidos, reduciendo así el costo de mantenimiento, el costo inicial de un objeto se reduce si este puede construirse con piezas estándar. La estandarización se logra cuando una empresa o toda la industria utilizan las mismas especificaciones para el objeto o parte, o para la interfase de usuario de un objeto, o las interfaces de una parte con otras.

4. *La anticipación a futuros cambios y el diseño de la ingeniería minimiza el costo de las modificaciones.* La ingeniería busca anticipar cambios y permitir que dichos cambios se den en sus diseños, por ejemplo, un inmueble puede iniciar con un solo piso, pero con la fuerza suficiente para soportar otro en caso de que se requiera construirlo por necesidades de espacio, este solo es un ejemplo de un cambio en el ambiente que provoca una modificación en el producto de la ingeniería. Otro tipo de cambio surge de los defectos de diseño.

5. *Las fallas y la tolerancia a errores se incluyen en un diseño de ingeniería.* No es posible anticipar todas las condiciones posibles, la ingeniería trata de minimizar los efectos de las condiciones inesperadas incluyendo una cierta tolerancia a errores en sus diseños. Una *falla* es un defecto en un objeto debido a un error de diseño, de producción o a alguna imperfección en el material que conforma al objeto; un *error* es una desviación visible de lo estipulado en los requerimientos. La tolerancia de errores significa que el ejercicio de una falla no lleva a un error, la tolerancia de errores minimiza el efecto de la falla de un objeto al ambiente en el cual reside el mismo.

6. *Los hallazgos de las ciencias y las matemáticas se aplican en la solución de problemas de ingeniería.* Una de las principales preocupaciones de las disciplinas maduras de la ingeniería es el adaptar las nuevas teorías a la resolución de problemas prácticos. El beneficio principal de la teoría es que permite que un diseño se evalúe por completo antes de ser implementado.

7. *Las técnicas eficientes se utilizan para escalar el tamaño o la producción.* Los proyectos de ingeniería comienzan por lo regular con un *modelo*, versión a escala del objetivo a alcanzar, la maqueta de un proyecto; una *muestra*, versión del resultado del proyecto que solo incluye algunas de las características finales que se buscan como una interfase de usuario; o un *prototipo*, consistente en una versión completa del producto terminado. Cada una de estas versiones experimentales resuelve su tipo de problema en particular pero plantea otros. Un modelo establece factibilidad, que es posible construir el proyecto; una muestra brinda satisfacción a los usuarios y un prototipo hace patentes las deficiencias de un producto antes de su liberación definitiva al mercado, pero aún cuando la construcción del prototipo halla tomado una semana, la producción deberá escalarse a 500 artículos por hora por citar un ejemplo.

8. *El uso de técnicas de control de calidad permiten mantener niveles predeterminados de calidad.* Uno de los ideales de la ingeniería es producir objetos sin defecto alguno, es decir que se adapten absolutamente a los

requerimientos tanto funcionales como no funcionales, pero esta meta puede no ser costeable, sin embargo, el control estadístico de calidad puede utilizarse para imponer un estricto control en los niveles de calidad.

9. *El uso de herramientas busca mejorar la productividad, pero sin hacer a un lado la creatividad humana.* La naturaleza de las herramientas de ingeniería ha cambiado dramáticamente durante los últimos 20 años con la introducción de herramientas informáticas. La infraestructura informática actual ha contribuido al éxito de la ingeniería concurrente, sin embargo, las herramientas electrónicas solo pueden producir representaciones de diseños conceptuales desarrollados por ingenieros y ayudarles a hacer ajustes a tales representaciones.

10. *Las contribuciones de diversas disciplinas participan en las tareas de ingeniería.* La práctica de la ingeniería no es individual, en el diseño de un edificio se requiere la participación de ingenieros civiles, mecánicos y eléctricos, así mismo los ingenieros de diseño deben interactuar con arquitectos y constructores.

11. *La administración efectiva de proyectos de ingeniería se basa en una serie de habilidades adquiridas por un amplio proceso de aprendizaje.* Dada la naturaleza de equipo en la ingeniería no todo ingeniero puede convertirse en administrador de un proyecto, pero cada ingeniero deberá tener la apreciación de la naturaleza de la administración de proyectos de ingeniería. Análisis costo-beneficio, empatar las tareas de diseño con el personal, monitoreo constante del avance de un proyecto e identificación oportuna de problemas requieren en buena medida de principios de administración de proyectos y de personal así como de un amplio conocimiento de aspectos técnicos de la ingeniería y mucha experiencia.

12. *La innovación tecnológica es positiva para la ingeniería, siempre que la meta sea enriquecer el conocimiento de la misma.* La innovación tecnológica tiene una amplia tradición en la ingeniería, cuando alguna rama de la tecnología no ha sido desarrollada aún, la ingeniería continua avanzando, desarrollando así la tecnología.

La ingeniería de programación tiene las mismas bases que el resto de las disciplinas más establecidas de la ingeniería: determinación de requerimientos, resolución de conflictos en estos, uso y reutilización de componentes estándar, planes de mantenimiento, previsión de lo inesperado por medio de la construcción de sistemas robustos, aplicación de las innovaciones de la ciencia de la computación en la

práctica, escalamiento, control de calidad, selección de herramientas, trabajo en equipo, administración de proyectos e ir más allá del ensayo y error.

De entrada los requerimientos se definen como un conjunto de variables, los requerimientos y la ubicación de recursos establecen un plan del sistema. La planeación suele omitirse en el desarrollo de sistemas con muy malas consecuencias, solo estableciendo un sólido plan de trabajo pueden evitarse futuros malentendidos. Por supuesto, un sistema como hubiese sido planteado por el cliente y el analista no necesita ser siquiera similar al producto terminado, sin embargo, las discusiones entre el cliente y el ingeniero en programación que conlleven al plan, y la necesidad de expresar los requerimientos en términos cuantitativos llevan a un mejor entendimiento del alcance del proyecto por ambas partes. Tampoco hay graves consecuencias en un cambio de políticas porque los cambios sólo afectaran al plan y no al propio sistema. El éxito de la reingeniería de sistemas depende en gran medida de que tan bien se ha planeado el proyecto de reingeniería, una de las preocupaciones principales del mismo es que la operación diaria no se vea seriamente afectada durante la reingeniería.

La búsqueda de equilibrio entre presupuesto y necesidades de usuario es bastante común. Otra situación frecuente es aquella en la que el cliente requiere, digamos, un sistema altamente flexible y que también sea fácil de aprender, desafortunadamente, estos requerimientos entran en conflicto, y el cliente debe decidir entre un sistema versátil que es difícil de aprender a operar, pero que después de un complejo periodo de aprendizaje cubre todas las necesidades de los usuarios, y un sistema que puede entenderse en una primera instancia pero que cuenta con una estructura inflexible. En términos de reingeniería de las organizaciones habrá que dar prioridad a los procesos que tendrán mayor impacto en los costos, satisfacción a usuarios o futuro crecimiento de la organización.

Otra forma de ahorrar es el uso de componentes estándar. La mayor aplicación de esta premisa esta en comprar un sistema listo para utilizarse en vez de generar uno a partir de los requerimientos establecidos. El uso de componentes existentes requiere que estos estén estandarizados, sin embargo, no debiera dependerse demasiado en la reutilización de programas existentes, la dificultad recae en que no hay mucho que estandarizar, y lo que puede serlo ya ha sido normado o lo será en breve, lo que es más, un sistema de aplicación es un modelo de un determinado entorno, y existen demasiados entornos los cuales presentan poco parecido entre ellos y el comportamiento de las entidades que los componen es difícil de normalizar.

La anticipación al cambio es un problema muy difícil de enfrentar, y el mantenimiento tiende a ser el componente más caro del desarrollo de sistemas. Las actividades de mantenimiento se llevan a cabo

principalmente por algún malentendido en las necesidades del cliente, por cambios en el entorno del sistema y por un deseo de agregar ciertas funciones nuevas al mismo. La sana labor de la ingeniería requiere que se eviten los malos entendidos, que los efectos de los cambios en la programación sean proporcionales a la magnitud de los cambios en su entorno y que la adición de nuevas funciones tenga el menor efecto posible en las partes existentes del sistema. Exactamente los mismos principios se aplican a la reingeniería de procesos, un proceso resultado de la reingeniería deberá cubrir ampliamente las expectativas que del proceso espera el cliente, y el proceso deberá ser fácil de modificar en caso de que las condiciones en las cuales se desenvuelve cambien.

Prevención a lo inesperado se refiere a que el sistema debe ser robusto, esto se refiere a tolerancia a errores y su enmascaramiento, evitando así que posibles fallas menores tengan consecuencias mayores. La tolerancia a errores en el aspecto físico de un sistema se logra duplicando componentes físicos. En lo que cabe a la programación, la tolerancia a errores es más difícil de establecer debido a los denominados fallos de causa común -dos desarrolladores pueden mal interpretar un requerimiento exactamente de la misma manera lo cual provocará que ambas versiones fallen de la misma manera. La tolerancia a errores disminuye los efectos de las fallas.

En lo que respecta al paso de la teoría a la práctica un aspecto importante es la secuencia de prioridades. En la ingeniería, la elasticidad aplicada o la fuerza de los materiales indica como se distribuyen las fuerzas en los cuerpos que han de soportarlas, ingenieros en materiales determinan si un cuerpo hecho de una sustancia en particular habrá de soportar las fuerzas a las que se verá sometido. La analogía aquí se refiere a los algoritmos y su implementación. Un ingeniero en computación necesita conocer y ser capaz de aplicar algoritmos fundamentales de almacenamiento y recuperación de archivos, y los algoritmos referentes a sistemas de cómputo, tales como la calendarización de procesos, sin embargo, el ingeniero en computación no necesita conocer todos los tipos de algoritmos de control de inventarios que puede utilizar el sistema en cuestión como lo requiere el personal experto, los cuales brindaran el conocimiento especializado en la materia. En términos secuenciales los algoritmos se desarrollan por expertos en el entorno, estos algoritmos se convierten en diseños de sistemas de cómputo por los ingenieros en computación y los sistemas se convierten en programas por parte de los programadores.

La escalabilidad esta relacionada directamente con la transferencia de las innovaciones de la tecnología de cómputo a la ingeniería de programación. Es labor natural para los programadores desarrollar pequeños programas en dependencias (que interactúen con otros programas o procesos), pero el desarrollo de sistemas a gran escala es labor de la ingeniería de programación. La transición de programar a nivel local a una mayor escala se optimiza dividiendo al producto de programación en

módulos de tamaño razonable, las técnicas de programación simples pueden aplicarse entonces a dichos módulos simplificando considerablemente el proceso. Los procesos de la organización identificados por la reingeniería son comparables a grandes módulos, y las tareas que los conforman a otros de menor tamaño. Lo que es más, la programación que habrá de apoyar los procesos rediseñados deberá tener una estructura modular que corresponda a las tareas de los procesos involucrados en la organización.

La capacitación que reciben los desarrolladores de sistemas tiende a relacionar a la programación principalmente con los aspectos funcionales del programa a realizar, resultando en un descuido a los atributos no funcionales del mismo. Es necesario que las pruebas a la programación se realicen con respecto a *todos los atributos del sistema a través del proceso de desarrollo del mismo* lo que asegura que la calidad se mantendrá en lo que concierne a todos los atributos. El índice costo-beneficio de tal actitud es positivo pues tanto el costo del desarrollo como el de mantenimiento se reducen considerablemente si se previenen posibles fallas o en caso de presentarse estas se eliminan tan pronto sea posible. Una de las principales premisas de la reingeniería de procesos es que una organización sólo puede subsistir si su producción o servicios son de alta calidad, y no lo conseguirá si los sistemas de información que apoyan a sus procesos son de baja calidad. Por tanto la conciencia de la calidad es una característica importante de la reingeniería.

Las herramientas automatizadas de diseño de sistemas facilitan por mucho el desarrollo de programas de aplicación, y mejores herramientas se encuentran disponibles en el mercado día con día. Pero en muchas ocasiones se espera demasiado de estas herramientas. Las herramientas no diseñan sistemas sino los analistas. Del mismo modo que un procesador de texto no escribe un libro, sino que exclusivamente manipula el texto creado por los autores, las herramientas de diseño solamente manipulan las representaciones de los procesos y las estructuras de información que definen los ingenieros en programación, de modo que, si un ingeniero produce un diseño deficiente, las herramientas resultan incluso contraproducentes, pues amplifican los efectos negativos del diseño.

Los ingenieros en computación deben colaborar con profesionales de otras disciplinas pues es ilógico pensar que han de conocerlo todo. Pareciera que una vez aplicada la reingeniería en una organización la carga de trabajo habría de disminuir, pero la necesidad de profesionales para ir a la par de los nuevos conocimientos en la organización continuará incrementándose, por otro lado la reducción de la plantilla de personal se equilibra en algunas organizaciones con una sobrecarga en el trabajo de los empleados que permanecen lo que conlleva a una perdida de calidad. Una forma de evitar esta anomalía es el incrementar la colaboración entre expertos, con mayor atención en los individuos dentro de un grupo de colaboración. Sin embargo, esto requiere que el personal hable un lenguaje común, y en el campo

técnico este tiende a ser el lenguaje de las matemáticas. La ciencia de la computación pone mucho énfasis en las matemáticas, y los ingenieros en programación necesitan ampliar su conocimiento matemático también, particularmente en estadística y teoría de control.

Por último consideramos la administración como un delicado balance entre productividad y espacio en el mercado, en corto y largo plazo. La productividad se ve dañada cuando los costos se incrementan desmesuradamente, hay incertidumbre en la industria o una atención inadecuada a las nuevas tecnologías. El espacio en el mercado en el contexto del ingeniero en computación consiste en la pregunta "¿Elegirá el cliente a nuestro equipo de trabajo para su próximo proyecto de desarrollo de sistemas?" lo cual se relaciona al análisis costo-beneficio. El riesgo no puede evitarse, el objetivo es sopesar los riesgos contra los beneficios, para efectos de un análisis costo-beneficio es necesario conocer los costos y la importancia de la estimación de costos ha sido reconocida por los ingenieros por miles de años.

El principio más importante que conlleva a todas las actividades de la ingeniería de programación es la partición de un proyecto grande en componentes manejables. Una aproximación a este problema es el separar el desarrollo de procesos en una secuencia de fases bien definidas. Una alternativa esta en dividir el sistema en módulos apropiados. Idealmente, se busca dividir el producto en módulos y a los procesos en fases, sin importar la estructura que se adapte debe existir una, esta depende en parte del tipo de programación a desarrollar, como y cuando se tomara la división por módulos y que atributos de calidad se consideran más importantes para los sistemas de apoyo.

4.4 Relación costo-beneficio aplicada al software

El costo de los sistemas se establece regularmente por los requerimientos de personal en términos de días-hombre, mes-hombre o año-hombre. La estimación de costos es una tarea compleja que requiere de mucha práctica. En el pasado las estimaciones tendían a ser muy optimistas, y la mayoría de los proyectos sobrepasaban los costos y tiempos establecidos. La mayor aceptación de los principios de ingeniería de programación, el uso de herramientas de desarrollo adecuadas, el énfasis en la reutilización de recursos, diseño y programación orientado a objetos y una mayor proporción de la fuerza de trabajo respecto a la capacitación formal en ciencias de computación o sistemas de información deberán eliminar las discrepancias del desempeño real contra el estimado. Desafortunadamente este no es el caso por varias razones. Primero, los proyectos cada vez son más ambiciosos tanto en tamaño como en complejidad. Segundo, algunos de los nuevos desarrollos en la materia, tales como las herramientas CASE[12] y la mayor atención dada a la reutilización de programas existentes no han cumplido con las expectativas que se buscaron cubrir inicialmente. Tercero, los efectos de los nuevos avances tecnológicos se han incluido en las formulas de estimación de costos de manera demasiado optimista. En particular, desde que los proyectos se han vuelto más ambiciosos se hace más valida ahora que nunca la observación de que la asignación de más personal a un proyecto en sus últimas etapas constituye un factor más de retraso al mismo.

Una de las técnicas más utilizadas para la estimación del costo de un sistema informático es el desarrollado por Boehm, COCOMO[13] (COnstructive COst MOdel) el cual utiliza expresiones empíricas para estimar el costo y duración de un proyecto. Las expresiones empíricas se derivan de la evaluación de proyectos concluidos con costos y duración conocidos.

La siguiente expresión indica el esfuerzo (o costo) medida en meses-hombre, necesarios para construir un sistema que tenga un estimado de S líneas de código fuente:

$$E = 3 \times S^{1.12} \times F$$

El parámetro de interés en esta fórmula es F, que se define como el factor de ajuste del costo y es el producto de las funciones de costo asociadas con 15 identificadores individuales del proyecto. Los identificadores se listan a continuación, agrupados en cuatro clases, cada uno de ellos con el rango de valores que pueden tomar:

[12] Ingeniería de Sistemas Auxiliada por Computadora, por sus siglas en inglés.

[13] Modelo de costo de construcción, ídem.

Atributos	Identificador	Rango
Producto	Confiabilidad de la programación	0.75-1.40
	Tamaño de la base de datos	0.94-1.16
	Complejidad del producto	0.70-1.65
Soporte físico	Variables en tiempo de ejecución	1.00-1.66
	Variables residentes en memoria	1.00-1.56
	Estabilidad del sistema operativo y DBMS	0.87-1.30
	Respuesta del procesador	0.87-1.15
Personal	Capacidad del analista	1.46-0.71
	Experiencia en aplicaciones reales	1.29-0.82
	Capacidad del programador	1.42-0.70
	Experiencia con el sistema operativo y DBMS	1.21-0.90
	Experiencia en el lenguaje de programación	1.14-0.95
Proyecto	Uso de prácticas modernas de programación	1.24-0.82
	Uso de herramientas de desarrollo	1.24-0.83
	Tiempo de desarrollo requerido	1.23-1.00-1.10

La presencia de F da el calificador al resultado del modelo COCOMO de "intermedio"; si todos los factores de costo asumen el valor nominal 1 el modelo COCOMO se considera en su nivel "básico". Multiplicando primero los valores menores de costo y luego los mayores posibles el valor de F varia de 0.0855 a 72.4, siendo el valor mayor más de 800 veces el menor y 72 veces el valor nominal. A pesar de los términos cuantitativos bien definidos del sistema, el COCOMO tiene sus contras, conceptos como la "capacidad del programador" son bastante inciertos, de modo que la asignación de valores es muy relativa además de que los avances tecnológicos debieran considerarse en los valores relativos de los diferentes identificadores.

El valor E obtenido por la formula previa puede utilizarse para calcular diferentes tiempos de desarrollo T (en meses) como sigue:

$$T = 2.5 \times E^{0.35}$$

Esta expresión demuestra que el tiempo de desarrollo no puede reducirse aumentando el tamaño del equipo de desarrollo (el exponente menos a uno que afecta al termino E lo explica matemáticamente). Por ejemplo, podría suponerse que un proyecto que se completaría por cinco personas en 10 meses

sería llevado a cabo por diez personas en 5 meses, sin embargo el proyecto tomara al menos 10 meses en cualquiera de ambos casos, es decir, que incrementar el equipo de trabajo a más de cinco no mejorara las cosas. La razón en recae en la dificultad de dividir un proyecto que requiere de 50 meses-hombre en más de cinco subproyectos, y un equipo de trabajo mayor requiere de una mayor comunicación entre los miembros del mismo.

El uso de las líneas de código para evaluar un programa ha presentado ciertas críticas, por ejemplo, hacen difícil comparar programas escritos en lenguajes diferentes; aún cuando se utilice el mismo lenguaje para la misma tarea puede haber mucha variabilidad, al igual que el volumen de un objeto no es factor determinante de su peso, así el largo de un programa no es un buen indicador de lo que realiza, es necesario establecer una medida del "peso" de un sistema de información, lo cual llevara a estimaciones de tiempo y costo más adecuadas.

Una de estas medidas son los puntos de función (FP), introducidos por Albrecht en 1979, y de amplia aceptación entre los desarrolladores de sistemas para negocios. Bajo la metodología de puntos de función, el "peso" de un sistema de información se determina por cinco parámetros: tipos de entradas externas, tipos de salidas externas, tipos de archivos internos, tipos de interfaces de archivos externas y tipos de consultas externas.

Los tipos de entradas externas comprenden los tipos de entradas que cambian la base de información del sistema o ejercen algún tipo de control sobre el sistema, tales como la entrada de un sensor. Las salidas externas pueden imprimirse, como los reportes, listados o cédulas de operación. Los tipos de archivos internos y externos miden el tamaño de la base de información de la aplicación, por ejemplo, cada tabla de una base de datos relacional que pertenece exclusivamente al sistema de aplicación cuenta como un tipo de archivo interno, sin embargo, si una tabla en una base de datos relacional es accesada por muchos sistemas de aplicación, entonces la tabla se cuenta como una interfase externa en todos los sistemas. Finalmente, cada combinación de entradas/salidas que no cambia la base de información se cuenta como un tipo de consulta externo. El conteo de los diferentes componentes del sistema es relativamente fácil, aunque existen opiniones divididas en cuanto a que tan diferentes deben ser los formatos de dos tipos de reportes que pertenecen a dos tipos de salidas diferentes.

A continuación debe asignarse un peso a las distintas interacciones del sistema, de acuerdo al tipo de complejidad de los mismos, por ejemplo, el peso de un tipo de entrada externa puede variar de 3 a 6, el peso de un tipo interno de archivo de 7 a 15 y el peso de un tipo de interfase externa de 5 a 10. La suma

de los pesos conforma el parámetro *UFP* (puntos de función sin ajustar). Este parámetro se adapta para tomar en cuenta la complejidad de la aplicación y el valor final esta dado por:

$$FP = UFP\ (0.65 + 0.01\ GI)$$

donde *GI*, el "grado de influencia", se obtiene asignando un valor entre 0 y 5 a cada una de las 14 características de la aplicación y sumando los valores. Las características típicas de una aplicación son la eficiencia con el usuario final, complejidad de procesamiento y facilidad de instalación.

En el pasado, la estructura interna de los sistemas de información tendía a ser muy simple, pero si un sistema de información ha de tomar decisiones complejas, es decir, si se convierte en un sistema de control de información, la estructura interna del mismo gana importancia. La característica de "complejidad de procesamiento" deja de ser un indicador adecuado para la complejidad estructural, ciertos autores incluso agregan un sexto parámetro: los algoritmos involucrados.

El *GI* es similar al factor *F* de ajuste de costo del COCOMO, pero con la principal diferencia de que para el cálculo de *GI* se suman sus factores, mientras que los de *F* se multiplican. Tomando por analogía a otros campos en los cuales el efecto conjunto de diferentes influencias debe considerarse como en la teoría de la probabilidad, un factor de ajuste multiplicativo es más apropiado que uno aditivo. Esta es una de las razones por las cuales el COCOMO, a pesar de ser obsoleto, continua siendo el más utilizado. Aún cuando no se aplique ninguna metodología para la estimación del costo y tiempo de un sistema, los calificadores de proyecto del COCOMO o las características involucradas en la metodología *FP* son de gran utilidad para determinar los factores de productividad de un proyecto de sistemas de información.

Los riesgos en la programación son de dos tipos: aquellos que se relacionan con el desarrollo de programas y los que se relacionan con los efectos que el sistema de información presentara una vez que sea liberado, *riesgos de desarrollo* y *riesgos operacionales* respectivamente.

Los riesgos de desarrollo tienen las siguientes causas principales:

- Una deficiente administración
- Interacción usuario-diseñador ineficiente
- Un proyecto técnicamente inoperable

Dado que la identificación de riesgos y el análisis son actividades de administración, es de esperarse que el mayor número de factores de riesgo importantes se relacionen con funciones estrictamente administrativas. Estos son (1) estimaciones de costo y tiempos equivocadas, basadas en suposiciones irreales en cuanto a la disponibilidad y capacidad del personal clave; (2) una revisión y monitoreo inadecuado de proveedores y contratistas; (3) supervisión inadecuada del desempeño del propio personal; (4) perdida del control administrativo al enfrentar un cambio en los requerimientos, como la aceptación de un flujo continuo de nuevos requerimientos de un cliente, o algún proveedor que no cumple las necesidades establecidas, o el agregar características no incluidas en los requerimientos; y (5) la no existencia de planes de contingencia para enfrentar algún tipo de desastre natural o acciones de la industria como huelgas, sabotajes o retrasos de los proveedores, contratistas o de los propios clientes al entregar sus requerimientos o no participar en las propuestas de sistemas y pruebas de aceptación de los mismos.

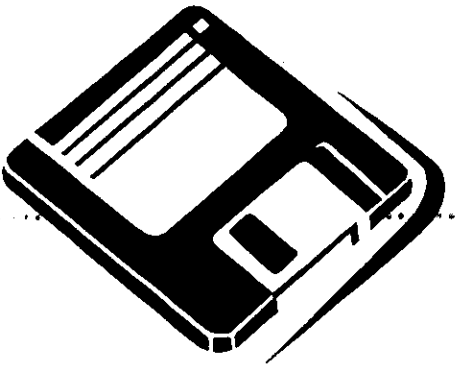
La administración del proyecto deberá asegurarse de que exista una comunicación sana entre los desarrolladores de sistemas y los usuarios finales del mismo, la falta de interacción con los usuarios lleva a (1) interfaces que causan desgaste físico o psicológico en los usuarios, o que no corresponden adecuadamente a las características de su entorno; (2) respuesta incorrecta al sistema en aplicaciones de control; y (3) bases de información que no contengan toda la información requerida para apoyar el proceso o que están cargadas de datos que no se necesitan en realidad. Una interacción deficiente con los usuarios puede provocar que estos no quieran tener nada que ver con el sistema, o un extenso rediseño después de la implementación inicial.

El riesgo debido a una mala administración de proyectos o a una deficiente interacción usuario-desarrollador puede controlarse mejorando la capacitación, lo que es más, se deben adecuar los antecedentes técnicos del personal para poder detectar una tercera clase de riesgo, cuando el sistema propuesto sobrepasa la capacidad física de la organización. La detección inmediata de este tipo de situaciones prevendrá el desperdicio de recursos en un proceso que no tendrá éxito.

Los riesgos relacionados al desarrollo no se establecen explícitamente en los requerimientos del proceso, pero pueden sugerir que un proyecto de reingeniería debe escalarse a un menor tamaño, lo cual conlleva a un nuevo conjunto de requerimientos o aún la posibilidad de abandonar el proyecto. Por otro lado, el control de los riesgos de operación debe incluirse en los requerimientos. En resumen, los riesgos operacionales se derivan de un deficiente control de calidad, donde calidad se refiere a lo adecuado de las funciones así como a la conjunción de las mismas con los pasos del proceso de la reingeniería previamente estudiado. Un control deficiente de riesgos operacionales puede llevar a que

los participantes en el proceso se nieguen a hacer uso del sistema de apoyo al proceso, depreciando así el esfuerzo de la reingeniería.

Así como los riesgos se relacionan al desarrollo de aplicaciones y a su uso, también lo hacen al costo de estas. La ingeniería de programación por lo regular se concentra en los costos de desarrollo haciendo a un lado el costo de la implementación y los beneficios. En el contexto de la reingeniería el énfasis es inverso, aquí, la pregunta relevante se refiere al uso: ¿Hasta que grado el proceso rediseñado reducirá costos y mejorará los beneficios?, sin embargo, el costo de poner en marcha un proceso rediseñado no ha de ser ignorado.



Metodología de Implementación

Determinación de requerimientos

La reingeniería de una organización comienza con la definición de la misma en términos de procesos y la especificación de estos. Los costos, riesgos y beneficios asociados con los diferentes procesos y las variantes de su implementación determinan las prioridades de la reingeniería. Los procesos de mayor interés son aquellos que transforman la investigación y el desarrollo en productos finales.

5.1 Diagnostico inicial

Un número considerable de las importaciones que se llevan a cabo en México no cumple con los requisitos en materia fiscal y/o aduanera para acreditar la estadía legal en el país de los bienes involucrados en dichas operaciones pasando estos a propiedad del Fisco Federal. Las funciones de la Dirección General de Destino de Bienes de Comercio Exterior Propiedad del Fisco Federal de la Secretaría de Hacienda y Crédito Público a este respecto son las siguientes:

- Determinar las políticas, procedimientos y criterios para el control, administración y destino de las mercancías de comercio exterior que han pasado a propiedad del Fisco Federal o se encuentren en los casos previstos en el artículo 157 de la Ley Aduanera.

- Planear e instrumentar los mecanismos de control, de la mercancía de comercio exterior que ha pasado a propiedad del Fisco Federal o ha sido puesta a disposición de la propia Dirección General conforme al artículo 157 de la Ley Aduanera, hasta su destino final, conservando la documentación correspondiente de cada operación realizada.

- Ordenar y practicar la identificación, recepción, traslado, maniobras, entrega, custodia, almacenaje y administración de la mercancía de comercio exterior puesta a disposición de la propia Dirección

General; así como de todas aquellas actividades o actos necesarios para lograr el destino final de esta mercancía.

- Determinar y realizar el destino de las mercancías de comercio exterior descritas por el artículo 157 de la Ley Aduanera mediante asignación, donación, venta o destrucción, asesorándose en todo caso del Consejo Asesor establecido en el artículo 145 de la citada ley, o de un comité de asignaciones para los destinos que no sean de la competencia del Consejo Asesor; así mismo, podrá destinar la mercancía de comercio exterior que ha pasado a propiedad del Fisco Federal para resarcimiento o indemnización.

- Controlar y supervisar las mercancías de comercio exterior, que por su naturaleza requieran de un destino inmediato, o su destrucción por ser nocivas para la salud, prohibidas, en estado de descomposición, u otra característica similar, conforme a los lineamientos establecidos por la propia Dirección General, informando posteriormente al Consejo Asesor o comité de asignaciones.

- Informar a las personas que presten los servicios señalados en el artículo 14 de la Ley Aduanera, de las mercancías en abandono que no serán objeto de destino por parte de esta Dirección General.

- Practicar la comercialización de mercancías, exclusivamente a través de un fideicomiso, o mediante un mandato otorgado a una institución de crédito y ordenar la práctica de avalúos de la mercancía de comercio exterior puesta a disposición, así como supervisar estas actividades.

- Resarcir o indemnizar por mercancías dispuestas por la propia Dirección General, en cumplimiento de resolución o sentencia que cause ejecutoria emitida por autoridad administrativa o judicial, mediante la devolución de la mercancía, y ante la imposibilidad práctica de esto, a través de mercancía de valor similar, o en su caso, mediante el pago de pecuniario, que determine la autoridad competente.

- Instruir que los ingresos obtenidos por la comercialización de las mercancías de comercio exterior se depositen en la Tesorería de la Federación de conformidad con la legislación aduanera y aplicar contra el fondo que se constituya, los pagos por resarcimiento o indemnización.

- Autorizar a las entidades federativas, la asignación definitiva de los vehículos que pasen a propiedad del Fisco Federal como pago de incentivos; así como la venta de vehículos inutilizados permanentemente para su circulación, conforme a lo establecido en la Ley de Coordinación Fiscal.

- Coordinarse en la esfera de su competencia, con las autoridades que directa o indirectamente estén involucradas con motivo del ejercicio de las facultades de la propia Dirección General.

- Instruir a la unidad administrativa que corresponda, se ponga a disposición de la propia Dirección General, la mercancía de comercio exterior que ha pasado a propiedad del Fisco Federal o se encuentre en los supuestos establecidos en el artículo 157 de la Ley Aduanera, así como requerir la documentación que sea necesaria para lo anterior y la integración de los expedientes.

- Informar a la unidad administrativa que corresponda de los hechos de que tenga conocimiento con motivo de sus actividades, que puedan constituir delitos fiscales o delitos de los servidores públicos de la Secretaría en el desempeño de sus funciones; así como coadyuvar en la esfera de su competencia, con la unidad administrativa que corresponda, en la investigación de hechos presumiblemente delictivos.

- Informar periódicamente y en forma anual al Oficial Mayor del resultado de las operaciones de destino de bienes; así como de los inventarios pendientes a disponerse.

Selección de procesos críticos

Una de las prioridades de la reingeniería es el orden en el cual los procesos rediseñados se integrarán a la operación de la organización, los procesos deben identificarse, especificarse y analizarse con respecto a los costos y riesgos. La definición de un proceso determina *qué* tareas pertenecen al proceso y el orden en el cual habrán de llevarse a cabo, pero no determina *cómo* realizarlas, es decir, quien es responsable de cada tarea.

Primero debe entenderse cual es el estado que guarda la entidad en estudio para definir donde conviene enfocar los esfuerzos, para realizar el diagnóstico y seleccionar los procesos críticos que se rediseñaran, es necesario definir el enfoque y alcance inicial del trabajo a realizar:

- Entrevistar al personal clave de la organización.

- Documentar el entendimiento de los principales procesos/subprocesos.
- Determinar los criterios de selección a utilizar para definir los procesos clave.
- Validar la estructura organizacional actual.
- Identificar la situación general de cada proceso y su principal problemática.
- Elaborar la matriz de evaluación.
- Seleccionar los procesos críticos.
- Determinar el alcance, beneficios y mejoras potenciales de dichos procesos.
- Definir el proyecto a realizar.

Es a través de las entrevistas con el personal clave de la organización como se identifican la principal problemática existente y las áreas de oportunidad. El cuestionario de entrevistas se enfoca a preguntas específicas sobre la situación actual del área y sus implicaciones:

- ¿Cuáles son las funciones del área?
- ¿Cuáles son las características generales de los procesos?
- ¿Cuáles son las características generales de la organización del área?
- ¿Cómo es el flujo natural de sus actividades?
- ¿Con qué sistemas se cuentan para controlar la operación?
- ¿Cuáles son las principales preocupaciones de su área de competencia?

De éstas preguntas surgen las principales implicaciones del área, las cuales se traducen en necesidades implícitas, y estas durante el rediseño, se convierten en oportunidades de mejora.

Para un entendimiento global de la organización, es necesario validar la estructura organizacional de las áreas y los principales procesos y subprocesos que se llevan a cabo:

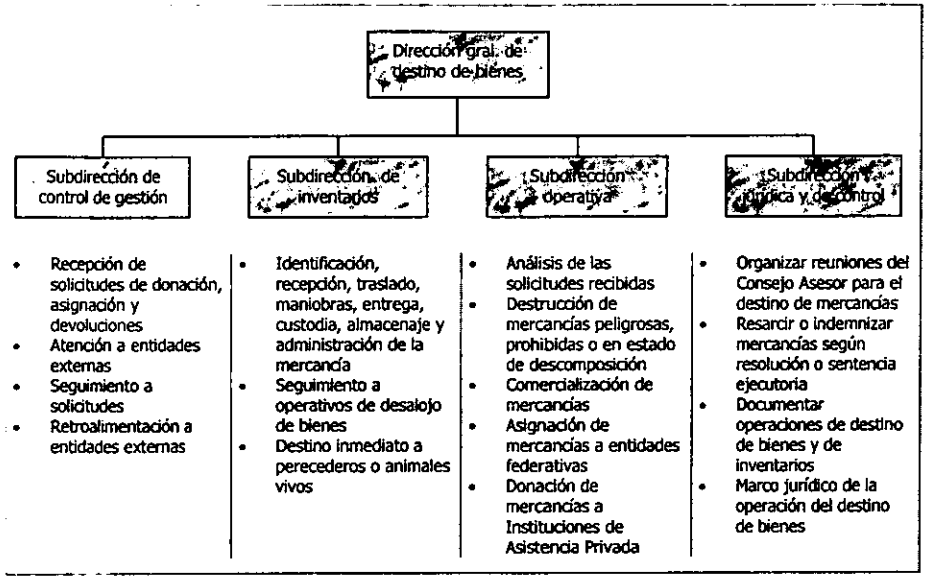


Figura 5.1 Validación de la organización

5.2 Métodos de especificación

Cuando un cliente solicita la elaboración de un sistema, debe tener alguna noción de lo que espera de este. Regularmente el nuevo sistema reemplaza a alguno existente o apoya algún procedimiento bien establecido, es decir, el sistema tiene un propósito. Un *requerimiento* es una característica del sistema o una descripción de algo que el sistema será capaz de hacer con el fin de alcanzar su último propósito. Por ejemplo, para un sistema de control de inventario un requerimiento puede ser que se haga un cierre de inventario cada dos semanas, y otro puede ser que los bienes que por su naturaleza requieran de algún cuidado especial se trasladen a los inmuebles que cuenten con las facilidades para su apropiado resguardo; el cliente puede solicitar una consulta a la base de datos del inventario desde diferentes ubicaciones físicas. Todas estas características son descripciones específicas de funciones o características que llevan al propósito general del sistema.

Nótese que ninguno de los requerimientos descritos especifica como se implementara el sistema, en otras palabras, no se menciona que administrador de base de datos se utilizará, cuanta memoria habrá de tener la computadora o que lenguaje de programación debe utilizarse para desarrollar el sistema. Este tipo de descripciones propias de la implementación no se consideran requerimientos a menos que así se indique por el cliente, en otras palabras, un requerimiento se refiere al *propósito* del sistema sin considerar como habrá de *implementarse*. Cuando se analizan los requerimientos desde este punto de vista, es fácil determinar aquellas características del sistema que resultan irrelevantes, en particular las que no tienen nada que ver con el propósito del sistema.

Esta discusión se hace más clara si se mantiene en mente el propósito del análisis de requerimientos y la determinación de prioridades. Se esta al principio de la etapa de desarrollo y la meta es *determinar la naturaleza del problema a resolver*. Cualquier discusión de una determinada solución es prematura hasta que el problema este claramente definido, lo que es más, el problema habrá de establecerse en términos del entorno de la organización, de este modo los requerimientos deben *enfocarse al cliente y al problema* y no a la solución o implementación.

Diagramas de flujo de datos

Para describir los requerimientos por el flujo de información en la organización se utilizaron diagramas de flujo de datos descritos a detalle en el capitulo 1. En este tipo de diagramación la jerarquía de los procesos se expresa por capas, de manera que diferentes niveles de detalle se muestran en capas diferentes. Se parte de la consideración del sistema como un *transformador* de datos, se examina la

información que fluye hacia el sistema, como se transforma y su estado al dejar el sistema. Como se muestra en la figura 5.2, la entrada es una flecha hacia la burbuja y la salida es otra desde la burbuja, es decir, la transformación se representa por la burbuja y las flechas son la ruta de la información.

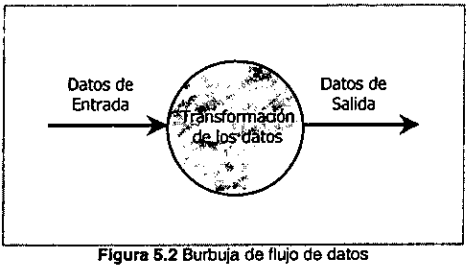


Figura 5.2 Burbuja de flujo de datos

Utilizando esta técnica es posible representar el flujo de datos para un sistema informático. La figura 5.3 representa el diagrama de contexto de la Dirección General de Destino de Bienes, caso especial de un DFD según se estudio en el capitulo 1 cuyo fin es representar las fronteras del sistema a desarrollar.

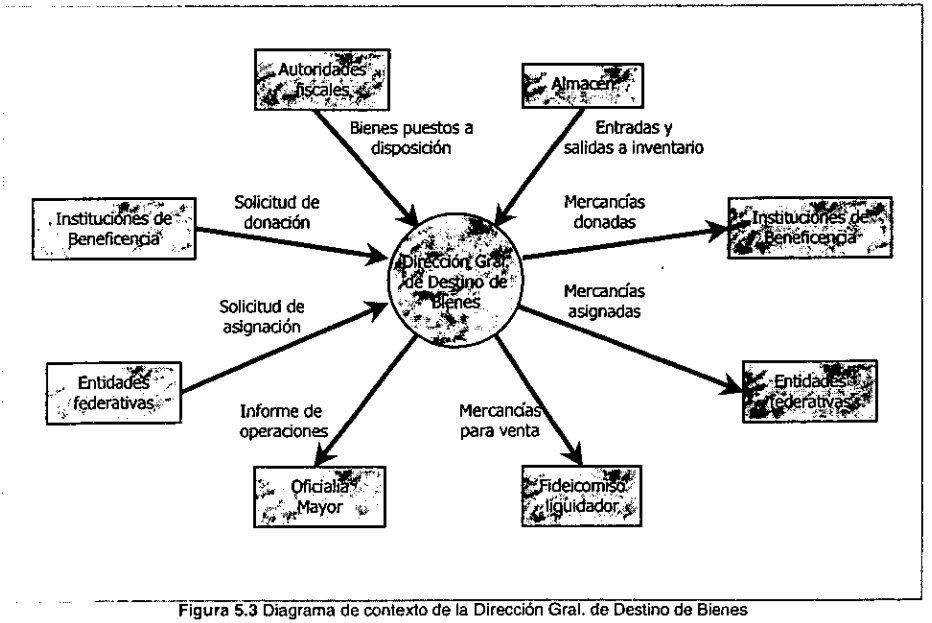


Figura 5.3 Diagrama de contexto de la Dirección Gral. de Destino de Bienes

A partir de aquí pueden dibujarse diagramas de flujo de datos para cada subsistema de la Dirección General: control de gestión, inventario, operación, jurídica y de control; diagramas de flujo de datos adicionales pueden describir como interactúan entre sí los subsistemas para completar las operaciones de la organización en general, este último se representa en la figura 5.4, los diagramas a detalle de cada uno de los subsistemas de la Dirección General se incluyen en la documentación del último punto del presente capítulo.

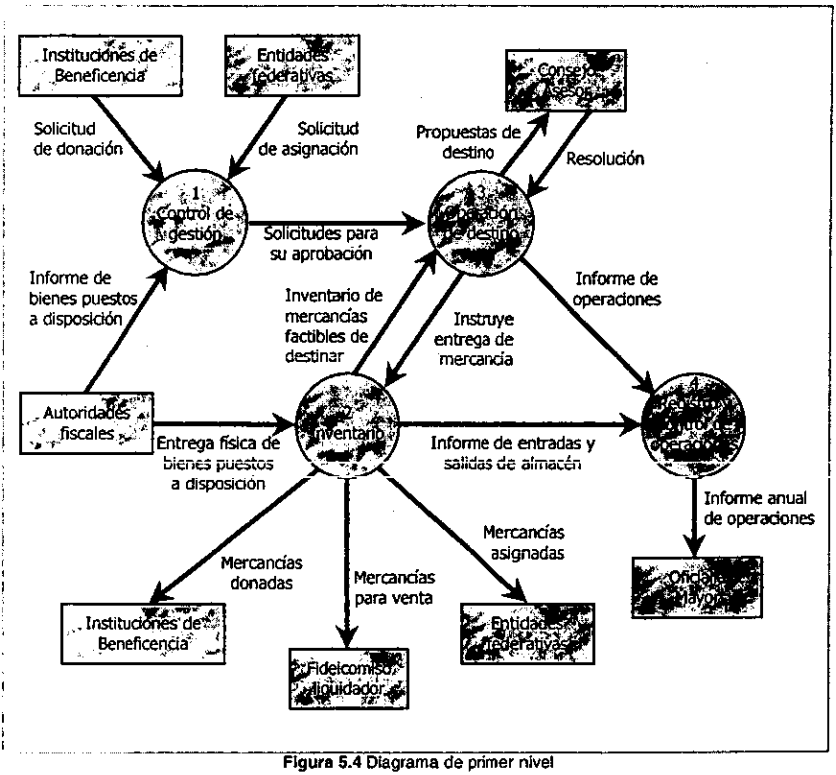


Figura 5.4 Diagrama de primer nivel

Para llevar a cabo la selección de procesos e identificar sus áreas de oportunidad, es necesario establecer los criterios de selección a utilizar:

Criterio	Area de oportunidad
Visibilidad	• Efecto al exterior percibido por los beneficiarios de la Dirección General o clientes en general.
Productividad	• Costo del proceso • Impacto presupuestal • Número de eventos relativos al total • Personal involucrado
Oportunidad	• Inversión requerida para realizar el cambio • Potencial de mejora • Rapidez para realizar el cambio
Factibilidad	• Problemática sindical • Intereses personales
Multiplicabilidad	• Capacidad de multiplicarse en otras entidades similares

Por otro lado, para la selección de los procesos que presentan mayores oportunidades con base en los criterios definidos, es necesario hacer una distinción:

Area de enfoque

Una vez entendido el marco general de la organización y conociendo los criterios de selección, aplicamos la matriz de evaluación:

Proceso	Subproceso	Evaluación de criterios					
		Vis	Prd	Opt	Fac	Mit	Total
Control de gestión	Registro de documentos de entrada	✓	✓	✓	✓		4
	Trámite y seguimiento		✓	✓	✓		3
Inventario	Control de entradas	✓	✓		✓	✓	4
	Validación física de bienes		✓	✓	✓		3
	Retiro de bienes	✓	✓		✓	✓	4
Operación de destino	Organización de la reunión del Consejo		✓	✓			2
	Seguimiento a las resoluciones	✓		✓	✓		3
Registro y control de operaciones	Control de inventario		✓	✓	✓		3
	Control de operaciones		✓	✓	✓		3
	Registro contable	✓	✓	✓	✓		4

De la calificación de los criterios obtenidos, se definen las principales áreas de oportunidad:

1. *Control de gestión:* Recepción de solicitudes y retroalimentación con beneficiarios.
2. *Inventario:* Administración de los bienes en almacén.
3. *Operación de destino:* Secuencias administrativas para asociar bienes a las solicitudes que satisfagan las necesidades del beneficiario.
4. *Registro y control de operaciones:* Servicios que se proporcionan dentro de la Dirección, en apoyo a los anteriores.

Las oportunidades identificadas reflejan las inquietudes captadas del personal entrevistado. Como resultado de este ejercicio podremos determinar los procesos considerados como críticos (proyectos a realizar):

1. Un sistema de captura y recuperación de solicitudes, que permita llevar seguimiento a las mismas, deberá incluir toda la información relacionada a la petición así como su actualización tras someterse a consideración del Consejo Asesor, señalando su resolución y en su caso los bienes que se entreguen al beneficiario, para efectos de control interno, deberá incluirse información referente a la reunión del Consejo en la cual se afecto.
2. Un control automatizado de entradas y salidas en almacén que permita detectar discrepancias entre los bienes puestos a disposición por las autoridades fiscales y las entradas al almacén, para tal efecto es necesario registrar la documentación asociada a las puestas a disposición de los bienes al Fisco Federal.
3. Un esquema de consultas a los sistemas antes mencionados que permita llevar seguimiento a las resoluciones del Consejo Asesor, el inventario en almacén, las solicitudes recibidas, las entregas realizadas e informes gerenciales que reflejen el sano desalojo de los recintos fiscales.

5.3 Programación existente

Uno de los principales problemas de la reingeniería tiene que ver con la programación que la organización ha acumulado por años pero que no cuenta con la orientación hacia procesos del rediseño propuesto para la organización. La programación existente no puede ser ignorada, brinda información esencial sobre la organización, representa una gran inversión y se ha convertido en imprescindible para la operación.

Debe considerarse como habrá de transformarse la programación existente de manera que se ajuste al soporte lógico de los procesos rediseñados de la organización. La transformación de esta programación puede verse como un proceso de tres etapas. Primero, se define un sistema de apoyo para cada uno de los procesos rediseñados y sus componentes se clasifican en base a su propósito, según este sea (a) apoyar la operación interna de la organización, o (b) comunicación con entidades externas. El resultado es un conjunto de eventos y otro de transacciones. Segundo, la programación existente se analiza bajo el mismo criterio. Tercero, la programación existente se mapea hacia la nueva estructura.

Actualmente la Dirección General de Destino de Bienes cuenta con un sistema nacido de la necesidad de establecer un control automatizado de los bienes de comercio exterior puestos a disposición del Fisco Federal.

Al no existir una infraestructura informática propia de la Dirección General, se encomendó a una empresa privada el desarrollo del sistema en operación, el cual fue liberado a finales de 1996, estableciendo con este una base de datos central para el control administrativo del proceso de integración y destino de los bienes de comercio exterior propiedad del Fisco Federal, sin embargo, es funcional solo parcialmente, ya que solo intervienen en su operación las áreas de inventarios y de asignaciones, faltando de incorporarse a su operación las áreas de ventas, donaciones y destrucciones, debido a que sus expectativas no son cubiertas por el sistema, ya que este fue realizado con necesidades detectadas en su momento y no ha evolucionado ni crecido de acuerdo a la operación real de la Dirección General, asimismo las áreas de inventarios y de registro y control de operaciones no obtienen la información procesada de acuerdo a sus requerimientos.

El presente estudio parte de las necesidades de información detectadas en el sistema en operación, incorporando además a las cuatro áreas de la Dirección General en base a las necesidades expresadas por los usuarios del mismo según se indica en la diagramación incluida en el último punto de este capítulo.

La protección de recursos de programación existentes no siempre es la mejor política. Ha habido muchas quejas respecto a la falta de resultados de la inversión en tecnología de información, donde no ha sido el caso la falla parte de la programación o, para ser más precisos, de los procesos implementados por esta. Ignorar el código anticuado escrito en un estilo anticuado que define procesos anticuados resulta ser en ciertas ocasiones la única manera de hacer redituable la inversión en tecnología de información. Uno de los mitos más concurrentes es que los costos de la programación son irrazonablemente altos, lo son solo cuando se implementa un proceso vagamente analizado a un ritmo acelerado, solo para ser modificado una y otra vez según se vaya adquiriendo un mayor conocimiento de la problemática de la organización. Si un proceso se define apropiadamente desde un principio, entonces la codificación es rápida y el código generalmente será confiable.

5.4 Documentación

Independientemente del método elegido para la definición de requerimientos, debe mantenerse bien documentado el resultado obtenido. Los analistas y clientes harán referencia a los requerimientos detectados a lo largo del desarrollo y mantenimiento del sistema final. Los requerimientos deben resultar significativos no solo a los clientes sino también a los diseñadores, lo que es más, los requerimientos deben organizarse de modo que puedan ubicarse fácilmente durante el desarrollo del sistema. Las ilustraciones y diagramas que acompañen a la documentación deberán ser consistentes con el texto. El numerar los requerimientos permite asociarlos fácilmente con el diccionario de datos y otros documentos de soporte al sistema, una convención numérica también se hace esencial para el equipo de desarrollo del sistema, si se hace algún cambio a los requerimientos durante el resto de las fases de desarrollo, los cambios pueden rastrearse a través del proceso de diseño y hasta los procedimientos de pruebas. Idealmente, cualquier función del sistema puede ligarse con el requerimiento que la genero o viceversa.

Todo proyecto comienza con el análisis y especificación de requerimientos. El tamaño del proyecto no elimina este paso del ciclo de desarrollo, sin embargo, puede afectar la decisión del método a utilizar para documentar, evaluar y llevar control de los requerimientos. Para el caso de la Dirección General de Destino de Bienes se pretende realizar un sistema de control de inventarios que lleve un control automatizado del destino final de las existencias en almacén en virtud a las solicitudes por parte de distintos beneficiarios y a las resoluciones del Consejo Asesor, así como a otros factores como la naturaleza de los propios bienes, todo lo anterior de manera transparente y con reportes adecuados. Dado que las funciones principales del sistema involucran manipulación de datos, se busca utilizar un método que capture la estructura de los datos y las relaciones entre los principales procesos de la organización, en la premisa anterior se basa la elección de los diagramas de flujo de datos para la especificación de necesidades del sistema propuesto para la Dirección General de Destino de Bienes.

En conclusión podemos ver al proceso de determinación de requerimientos como la culminación de una serie de sesiones de preguntas y respuestas. Es el momento de poner en claro los requerimientos de la organización y de verificar que todas las partes involucradas en el proyecto tengan la misma idea del problema a ser resuelto.

Control de gestión

Los cuatro procesos principales detectados en la figura 5.4 tienen una relación importante entre sí. El primero de ellos, control de gestión, es el encargado de recibir las solicitudes tramitadas por los beneficiarios de la Dirección General y dar seguimiento a las mismas, así como retroalimentar el avance de las solicitudes a los beneficiarios según se lo requieran.

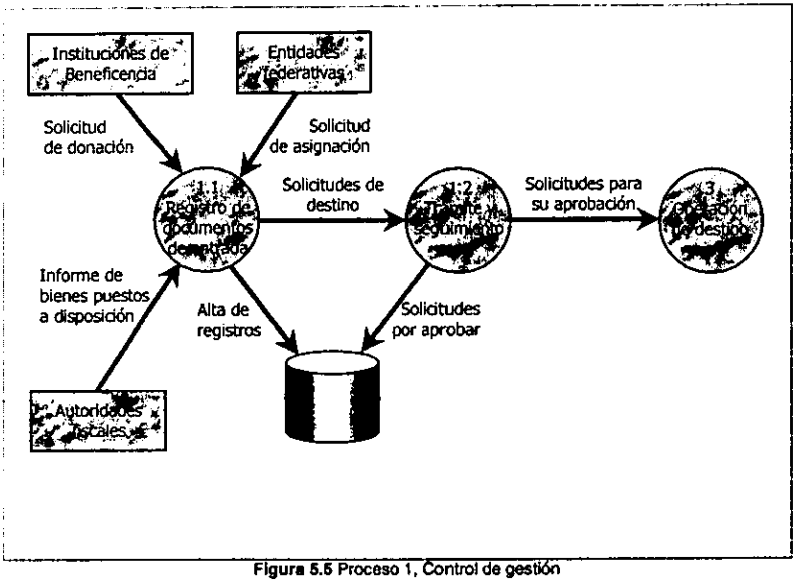


Figura 5.5 Proceso 1, Control de gestión

La figura 5.5 describe la interacción del proceso control de gestión con el resto de las actividades de la Dirección General, dentro del rediseño de procesos se propone además que la captura de las características de los bienes puestos a disposición se realice en este procedimiento, dado que el soporte documental de estos movimientos se recibe en las oficinas de la propia Dirección General, de este modo el área se especializa en la administración de toda documentación recibida, representando así la imagen de la Dirección General con las entidades administrativas externas que participan en su operación. La figura 5.6 describe a detalle los procesos de esta interacción.

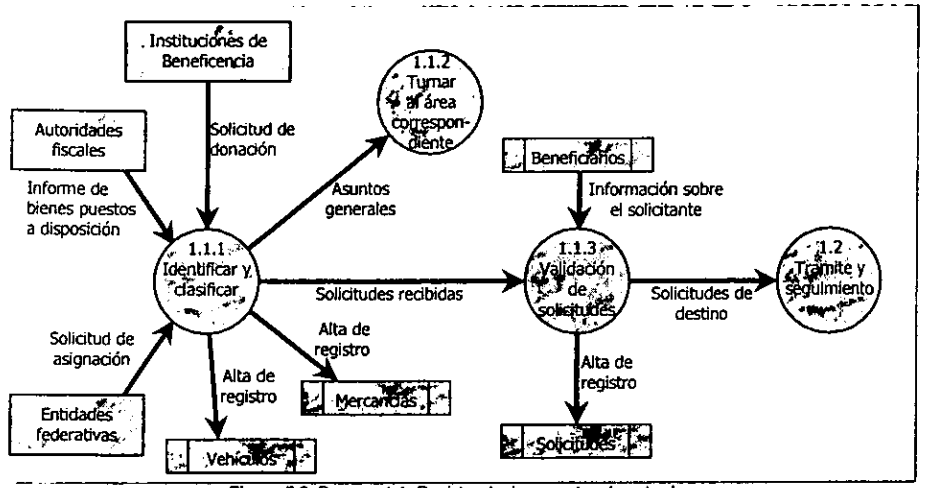


Figura 5.6 Proceso 1.1, Registro de documentos de entrada

Partiendo de la premisa anterior la validación de documentación recibida se lleva a cabo por esta área, básicamente se busca que la subdirección de operación de destino reciba únicamente solicitudes factibles de proceso, rechazando aquellas que se reciban de entidades no autorizadas a recibir donativos por parte de la Secretaría de Hacienda (aquellas que no estén registradas como Instituciones de Asistencia Privada). En lo que toca a los bienes de puesta a disposición, el área de control de gestión ha de registrar los datos tal como se encuentran asentados en el documento, dejando la revisión física de los bienes al área de inventario, lo anterior porque primero se recibe la documentación relacionada a los bienes puestos a disposición por parte de las autoridades fiscales vía fax y la recepción de los bienes descritos depende de la ubicación de los mismos al momento en que pasan a propiedad del Fisco Federal.

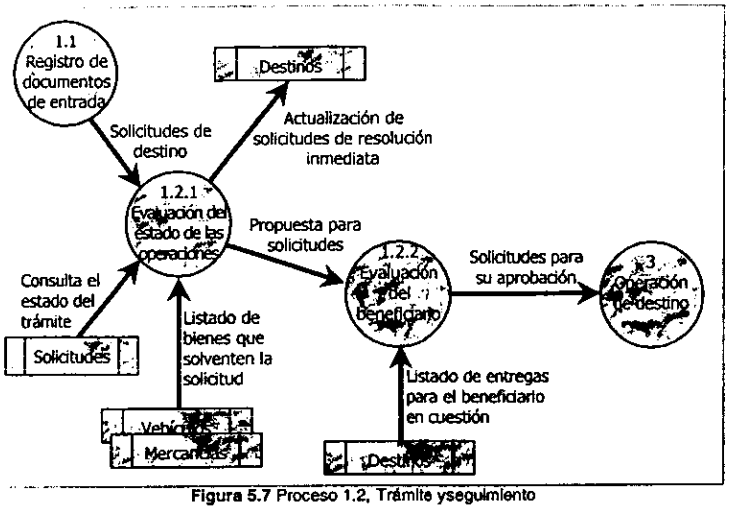


Figura 5.7 Proceso 1.2, Trámite yseguimiento

La evaluación del estado de las operaciones es un proceso constante que refleja el quehacer de la Dirección General. En este punto destaca el manejo de los bienes cuyo destino no se somete a consideración del Consejo Asesor, los perecederos y animales vivos que por su naturaleza requieren de un destino inmediato se entregan por parte de la Autoridad Aduanera a los Sistemas DIF, se da parte a la Dirección General por medio de un propio que incluye los oficios de puesta a disposición y las actas de entrega-recepción asociadas a estas operaciones, un caso similar ocurre con mercancías prohibidas como las armas que se entregan inmediatamente a la Secretaría de la Defensa Nacional y otras como el producto de la piratería o aquellas que ofendan a la moral, mismas que se destruyen por parte de la Autoridad Aduanera de manera inmediata, en ambos casos las operaciones se hacen del conocimiento de la Dirección General y es el área de control de gestión la que se encarga de registrar la documentación correspondiente, su validación se lleva a cabo por representaciones regionales de la Unidad de Contraloría Interna de la Secretaría de Hacienda.

Por último se complementa la información asociada a las solicitudes recibidas y factibles de aprobar con un histórico de las entregas realizadas al solicitante para así brindar un panorama más amplio al Consejo Asesor de la situación del beneficiario solicitante.

Inventario

El sistema de inventario habrá de satisfacer las necesidades de control de entradas y salidas a almacén que requiere toda infraestructura de este tipo. La propuesta se refiere a un control en almacén de las entradas y salidas con acceso a la base de datos de la Dirección General para la validación física de los bienes recibidos y detectar así de inmediato cualquier discrepancia entre los bienes relacionados en el oficio de puesta a disposición y la entrega física de mercancía.

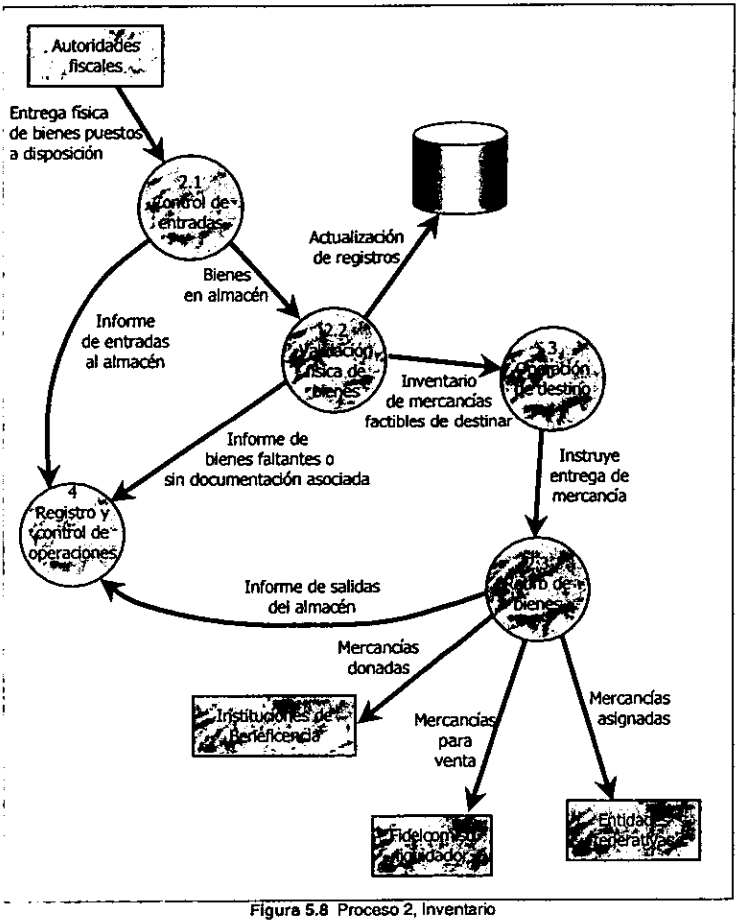


Figura 5.8 Proceso 2, Inventario

Es necesario hacer notar la comunicación que los procesos de inventario tienen con el área de registro y control de operaciones, lo anterior para dar seguimiento inmediato a las discrepancias detectadas.

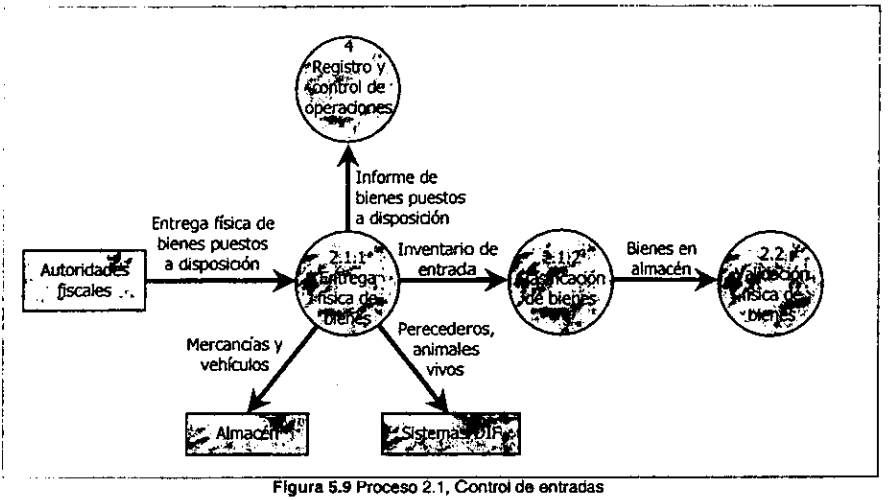


Figura 5.9 Proceso 2.1, Control de entradas

Se considera además la posibilidad de que el propio almacén tenga que efectuar la entrega de un bien de destino inmediato. Este caso se presenta sobre todo cuando la aduana cuenta con su propio almacén y encarga a este la entrega inmediata de los bienes a los Sistemas DIF, respetando el procedimiento normativo se hace participe a las áreas de Contraloría Interna de la Secretaria de Hacienda. Aunque este proceso es propio de la Autoridad Fiscal se hace del conocimiento de la Dirección General para su integración a su universo de operaciones.

Para el control de entradas al almacén es necesario hacer una clasificación de los bienes recibidos de acuerdo a algún estándar propio de la Secretaría. Para tal efecto se recurre a la fracción arancelaria, catálogo utilizado para las operaciones de importación y exportación que incluye todos los bienes de comercio exterior que pasan por las aduanas nacionales, lo anterior conlleva a un grado de detalle muy amplio lo que hace necesaria una capacitación constante del personal involucrado en las operaciones de recepción de bienes.

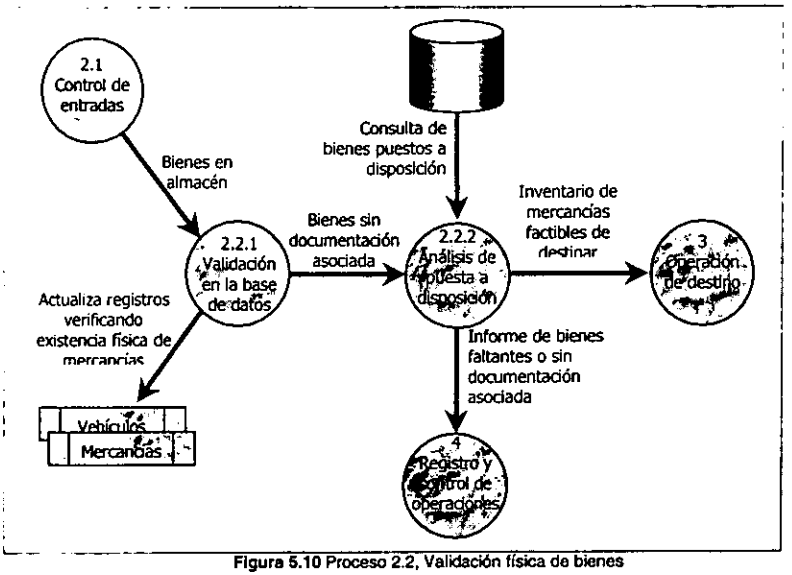


Figura 5.10 Proceso 2.2, Validación física de bienes

Una vez en claro la relación de los bienes recibidos en almacén se hace necesario cotejar la documentación asociada a su entrega, capturada en su momento por parte del área de control de gestión, en caso de no ser así se prepara un reporte al área de registro y control de operaciones, la cual se encargara de dar seguimiento a la falta de documentación, poniéndose en contacto con la entidad federativa que hizo la entrega y el área de control de gestión para aclarar la procedencia de los bienes. Para el caso de los bienes no relacionados en el oficio de puesta a disposición el procedimiento es el mismo, buscando aclarar la situación de los bienes. Si en cualquiera de ambos casos esto no fuera posible se turna el caso a la Unidad de Fondos y Valores de la Contraloría Interna de la SHCP para fincar responsabilidades respecto a la situación de los bienes faltantes o sin documentación asociada.

Una vez que se cuenta con un inventario solido, bien clasificado y con soporte documental que asegure la propiedad de los bienes por parte del Fisco Federal pueden considerarse estos para su destino entre alguno de los beneficiarios de la Dirección General. Lo anterior en base a un reporte de las mercancías en inventario que se encuentren en esta situación y que no hayan sido destinadas, dicho reporte se hace llegar a las áreas de operación de destino para que así se disponga a la brevedad de los bienes relacionados.

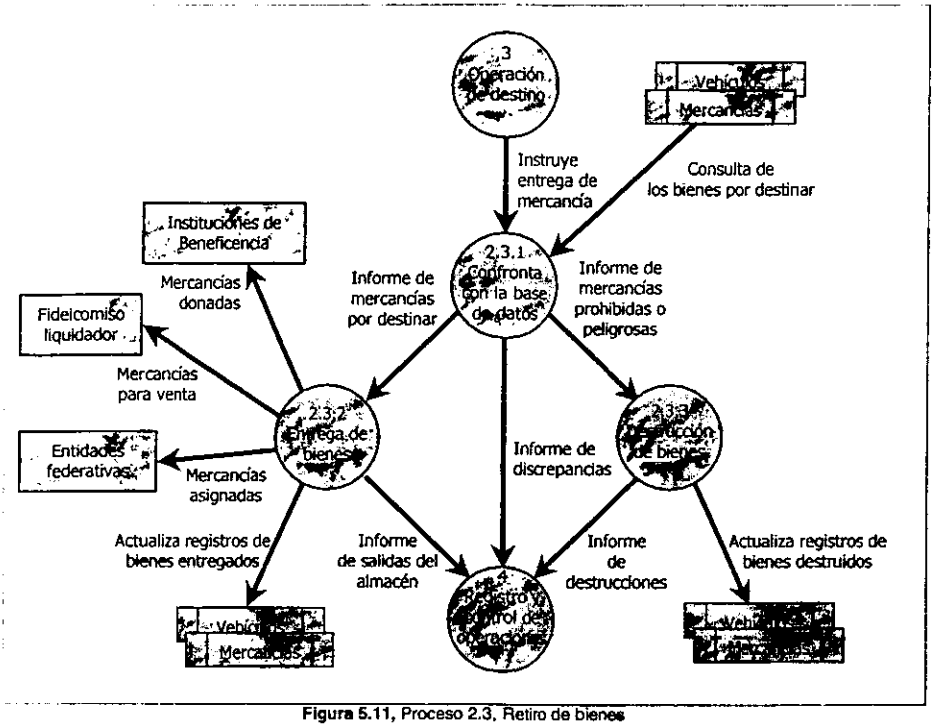


Figura 5.11, Proceso 2.3, Retiro de bienes

El desalojo de los bienes de almacén atiende exclusivamente a un oficio de instrucción turnado por el Consejo Asesor. Este se hace llegar al almacén a través del área de operación de destino e instruye a este a dar destino de las mercancías según uno de los siguientes casos: (1) entregar los bienes a las entidades federativas o instituciones de asistencia privada a las que se haya aprobado su solicitud, (2) las mercancías que no son factibles de donar, ya sea por tratarse de artículos de lujo o propios de alguna actividad industrial se ponen a la venta a través de un fideicomiso dependiente de la Dirección General encargado de llevar a cabo el proceso de licitación y comercialización de las mercancías en comento así como de integrar los recursos obtenidos al fondo de la Tesorería con el que se cuenta para tal efecto y que forma parte del Fisco Federal, y (3) los bienes que se consideren no aptos para su entrega o venta se destruyen levantandose una constancia con la participación de la Unidad de Fondos y Valores. En cualquera de los casos se hace una actualización de los registros por parte del área de destino que se estudiara más adelante.

Destino de bienes

El Consejo Asesor es el encargado de determinar el destino final que se dará a los bienes propiedad del Fisco Federal puestos a disposición de la Dirección General, esta constituido por representantes de las Instituciones de Asistencia Privada, de las Entidades Federativas, el Oficial Mayor de la Secretaria de Hacienda y el Director General de Destino de Bienes. El área de operación de destino es la encargada de convocar y organizar las reuniones del Consejo así como de vigilar el cumplimiento de las resoluciones del mismo.

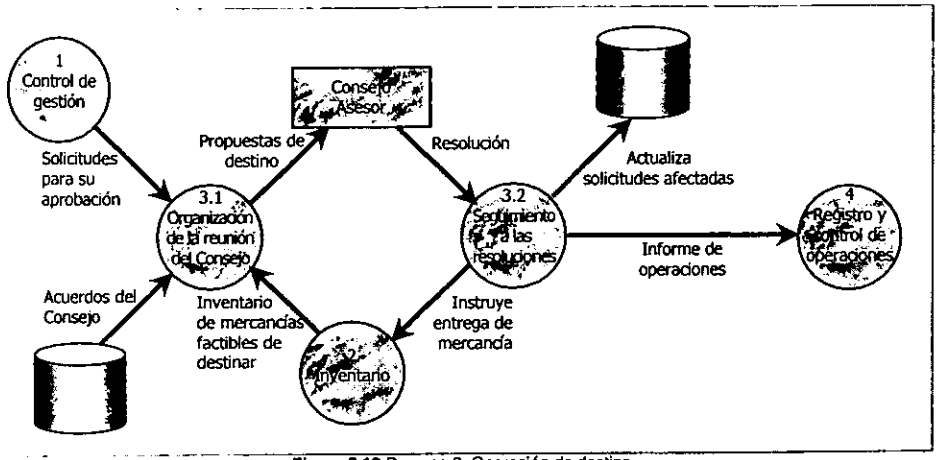


Figura 5.12 Proceso 3, Operación de destino

Para apoyar la toma de decisiones de los integrantes del Consejo Asesor el área de operación de destino se basa en el inventario de bienes propiedad del Fisco Federal, las solicitudes recibidas y los acuerdos tomados en anteriores reuniones del Consejo para así conformar la carpeta ejecutiva que contenga la información antes descrita de manera clara y bien organizada, lo anterior habrá de apoyarse de una serie de consultas al estado de las operaciones de la Dirección General y a una sana comunicación de las áreas que la conforman.

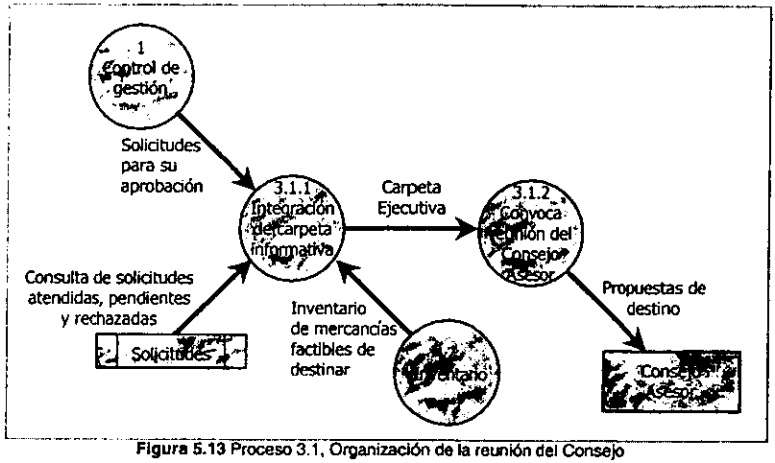


Figura 5.13 Proceso 3.1, Organización de la reunión del Consejo

Una vez efectuada la reunión del Consejo Asesor es necesario dar seguimiento a las resoluciones dictadas hasta su buen término. La Dirección General habrá de generar los oficios de resolución donde se asienten en expediente los resultados de la reunión del Consejo para su descarga en la base de datos por parte del área de operación de destino. Paralelamente esta última genera los oficios de instrucción, documento que se turnará a los responsables de almacén intruyendolos a entregar los bienes relacionados al beneficiario indicado según resolución del Consejo, mientras el área de control de gestión informa a los beneficiarios cuyas solicitudes se atendieron del resultado a favor o en negativa de su requerimiento. Todas las operaciones antes descritas han de actualizarse en la base de datos de solicitudes por parte del área de operación de destino.

Como parte de sus funciones de control el área de registro y control de operaciones habrá de ser informada del seguimiento de las resoluciones para que esta se encarge de llevar a buen término las operaciones, integrando toda la información asociada al trámite de destino para su resguardo definitivo, desde el oficio de puesta a disposición de los bienes hasta el acta de entrega-recepción de los mismos, incluyendo los oficios de resolución y de instrucción generados por el área de operación de destino.

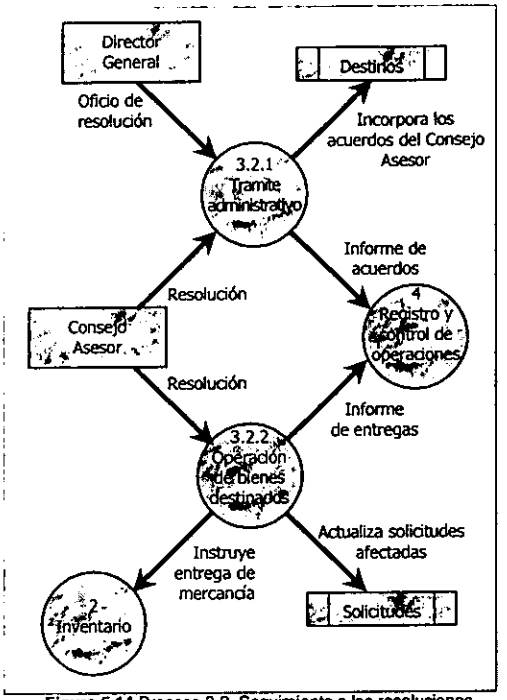


Figura 5.14 Proceso 3.2, Seguimiento a las resoluciones

Registro y Control de Operaciones

Por norma la Dirección General de Destino de Bienes tiene la obligación de presentar un reporte anual de sus actividades a la Oficialía Mayor de la Secretaría de Hacienda de la cual depende. Para tal efecto y a fin de llevar un control adecuado de las operaciónes de la Dirección General el área de registro y control de operaciones se encarga de vigilar las operaciones de puesta a disposición, movimientos al inventario, operaciones de destino y seguimiento a las resoluciones del Consejo Asesor. Así mismo en caso de haber alguna irregularidad en los oficios de puesta a disposición de mercancías con respecto a las existencias en almacén es el área responsable de informar a la Unidad de Fondos y Valores de lo anterior y dar seguimiento de los dictamenes al respecto. El área de registro y control de operaciones es, por lo tanto, usuario final de la información del sistema para la Dirección General, de apoyo a las funciones sustantivas de la operación de destino de bienes y en función a cuyas necesidades de información se generarán reportes adecuados en el sistema a desarrollar.

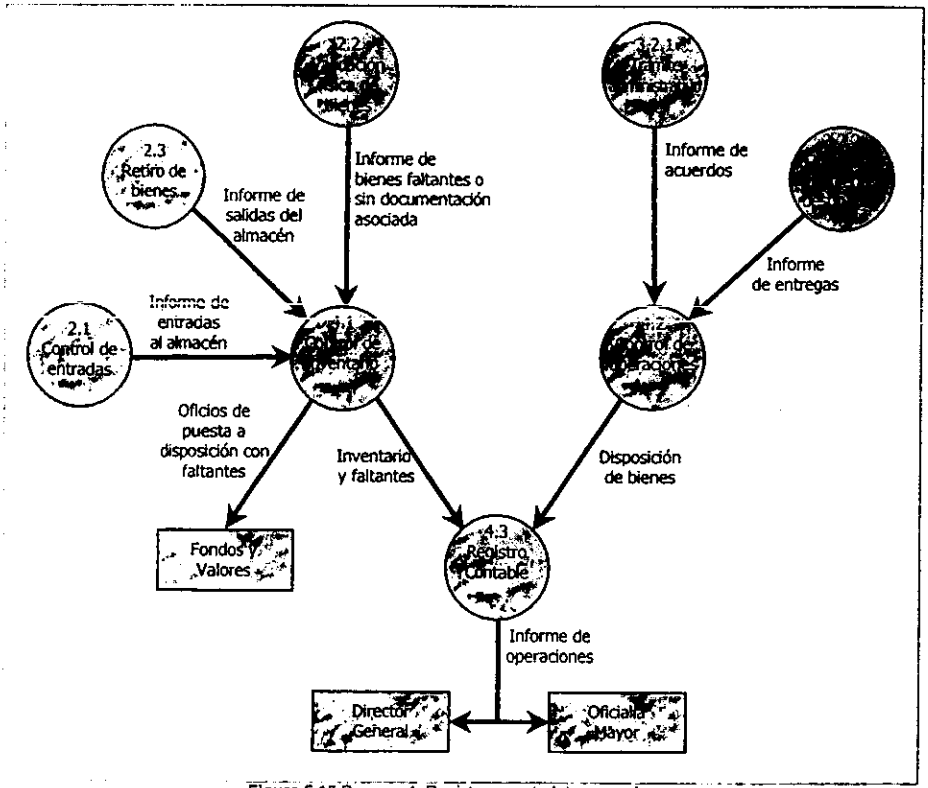


Figura 5.15 Proceso 4, Registro y control de operaciones

Diseño del sistema

El diseño de un sistema requiere de dos principios: el diseño conceptual para el cliente y el diseño técnico para los diseñadores del sistema y el soporte físico del mismo. Las necesidades del cliente obligan a ver el sistema como un proceso y como un producto. El proceso creativo del diseño de un sistema es la transformación del problema en una solución. El producto resultante es una descripción de la solución: el diseño del sistema.

6.1 Características de un buen diseño

En la determinación de requerimientos definimos claramente el problema a resolver, entonces, se supone que se tendrá una solución a la problemática si se satisfacen todas las necesidades especificadas. En muchos casos el número de soluciones posibles es infinito.

Por lo tanto, para transformar los requerimientos en un sistema funcional debe incluirse a los desarrolladores y a los usuarios finales en el diseño del sistema. El cliente entiende *lo que* el sistema ha de hacer, al mismo tiempo los desarrolladores del sistema entienden *como* ha de funcionar el sistema. Por esta razón el diseño del sistema es en realidad un proceso en dos partes. Primero se produce una *especificación de sistema* que dice al cliente exactamente lo que el sistema habrá de hacer, esta especificación suele denominarse *diseño conceptual del sistema*. Una vez que el cliente aprueba el diseño conceptual, se presenta al equipo de desarrollo del sistema el *diseño técnico* que les permitira construir el soporte físico y lógico para el sistema. Este diseño en dos partes refleja ambos participantes en la descripción de requerimientos: el cliente y el analista. El diseño conceptual se concentra en la *función* del sistema, mientras que el diseño técnico describe la *forma* que el sistema habrá de tomar.

Diseño conceptual

El diseño conceptual dice al cliente lo que el sistema habrá de realizar. Como se estudio en el capítulo 1, el sistema se describe en términos de su entorno, entidades, atributos y relaciones. ¿De dónde provienen los datos?, ¿Qué pasará con estos en el sistema?, ¿Cómo se vera el sistema para los usuarios finales? Las especificaciones dicen al cliente exactamente que funcionalidad esperar.

Lo que es más, el sistema se describe en un lenguaje que el cliente habrá de entender, sin involucrar lenguaje técnico. Por ejemplo, puede indicarse al cliente que un menú en pantalla dara acceso a los usuarios a las funciones del sistema. La descripción del sistema incluso puede listar las respuestas aceptables del usuario y las acciones que resultaran. Sin embargo, no se indica al cliente como se almacenan los datos en el sistema o que tipo de administrador de base de datos se utiliza. Similarmente puede indicarse al cliente en la especificación del sistema que un mensaje se transmitirá de una locación a otra, sin indicar el protocolo de red utilizado que define el *como* funciona el sistema en vez de lo que *hace* el sistema. El *como* pertenece al diseño técnico, por tanto, un buen diseño conceptual debe tener las siguientes características:

- Esta escrito en térmios del cliente
- No contiene jerga técnica
- Describe las funciones del sistema
- Es independiente de la implementación
- Se deriva del documento de requerimientos
- Incorpora todos los requerimientos a detalle adecuado

Diseño técnico

El diseño técnico explica el sistema a los desarrolladores encargados de su implementación. El diseño describe la configuración del soporte físico, las necesidades de programación, las interfaces de comunicación, la entrada y salida del sistema, la arquitectura de red y cualquier cosa que traslade los requerimientos en una solución al problema planteado en un principio. La descripción del diseño es un esbozo de la especificación del sistema.

El diseño conceptual permite al cliente conocer lo que hará el sistema al explicarle las características externas visibles del mismo. La descripción interna se deja al diseño técnico. Por tanto, este último deberá incluir lo siguiente:

1. La arquitectura del sistema: una descripción de los principales componentes físicos del sistema y sus funciones.
2. La estructura de programación del sistema: la jerarquía y función de los componentes de programación.
3. Los datos: las estructuras de información en el flujo de datos.

6.1.1 Modificaciones a los requerimientos

La facilidad con la que se adapten al sistema las modificaciones que los requerimientos del cliente pudieran tener es especialmente importante pues estos cambios o aquellos necesarios para la corrección de errores algunas veces resultan en un cambio al diseño del sistema. A continuación se mencionan las características de un buen diseño que permiten establecer un sistema robusto y confiable con accesibilidad al cambio:

Modularidad

La división por módulos es una característica de un buen diseño. Módulos de alto nivel permiten ver al problema como un todo y ocultar detalles que pueden distraer la atención del analista. La capacidad de estudiar niveles más a detalle cuando así se desee brinda la flexibilidad requerida para entender lo que habrá de hacer el sistema, seguir el flujo de datos a través del mismo y señalar los puntos de complejidad. Contrariamente a lo que pudiera pensarse, el dividir un problema en partes no convierte mágicamente a un problema complejo en un conjunto de problemas sencillos. Sin embargo, la modularidad permite aislar aquellas partes del problema más difíciles de manejar, a su vez, este aislamiento evita confusiones o complejidad adicional de funciones y datos no relacionados con el problema en estudio.

Niveles de abstracción

Debido a que los módulos en un nivel definen a los que se les derivan en niveles inferiores, el de nivel superior se considera como el más abstracto. Conforme se va entrando más a fondo se encuentran más detalles de cada módulo. De este modo, se dice que los módulos están dispuestos en *niveles de abstracción*. Los niveles de abstracción ayudan a entender el problema como parte del sistema. Examinando los niveles desde el mayor hacia abajo, los problemas más abstractos pueden manejarse primero y llevarse a

cabo su solución a medida que se genera la descripción general. En cierto modo, los niveles superiores más abstractos ocultan el detalle de las funciones o componentes de datos. Una ventaja de este "*ocultar la información*" es que cada módulo aísla una decisión importante de los otros. De este modo, si las decisiones de diseño han de cambiar, el diseño del sistema como un todo puede permanecer intacto mientras que solo el diseño de un módulo cambia.

La abstracción y la forma en que se muestra solo la información general de un módulo permite examinar las formas en las cuales se relacionan los módulos entre sí en el diseño general. El grado en el cual los módulos son independientes unos de otros es una medida de que tan bueno es el diseño del sistema. La independencia de los módulos es deseable por dos razones. Primero, es más fácil entender como funciona un módulo si su función no esta completamente ligada a la de demás. Segundo, es mucho más sencillo modificar un módulo si es independiente de los otros. Regularmente un cambio en los requerimientos o en una decisión del diseño implica que varios módulos han de modificarse. Cada cambio afecta datos, funciones o ambos. Si los módulos dependen demasiado unos de otros, un cambio en uno de ellos puede implicar cambios en otros más. Entre más independientes son los módulos, más fácil será aislar aquellos afectados por el cambio.

Acoplamiento

La dependencia entre módulos depende de varios aspectos:

1. Las *referencias* hechas de un módulo a otro. Por ejemplo, el módulo A puede invocar al módulo B, de modo que la función del módulo A depende de la del módulo B.
2. La *cantidad de información* que pasa de un módulo a otro. Por ejemplo, el módulo A puede pasar el contenido de un arreglo al módulo B, de modo que el módulo B depende del módulo A.
3. La *cantidad de control* que un módulo tiene sobre otro. Por ejemplo, el módulo A puede pasar al módulo B una bandera de control. La función realizada por B depende del valor de la bandera.
4. El *grado de complejidad* en la interfaz entre un módulo y otro. Por ejemplo, si un módulo A pasa una bandera al módulo B, pero los módulos C y D intercambian valores no antes de que D pueda completar cierta función, entonces la interfaz entre A y B es menos compleja que la que existe entre C y D.

Es preferible minimizar la dependencia entre módulos por diferentes razones. Primero, si un elemento se afecta por una acción del sistema siempre debe saberse que módulo causa el efecto en un momento dado. Este conocimiento permite cambiar una parte del diseño del sistema sin modificarlo todo. Por

ejemplo, si los requerimientos del cliente cambian y una función o tipo de datos se reemplaza por otro, un diseño modular con baja dependencia entre sus módulos permite reemplazar un módulo directamente por otro. Solo unos cuantos módulos más se afectaran por el cambio y pueden ser suceptibles a modificación.

Segundo, la modulabilidad ayuda a rastrear la causa de los errores del sistema. Si un error ocurre durante el desempeño de una función en particular, la independencia de los módulos permite aislar el módulo defectuoso más facilmente.

Un diseño orientado a objetos generalmente presenta poca dependencia entre sus módulos dado que cada definición de un módulo como un objeto contiene la definición de las acciones que este lleva a cabo y que se llevan a cabo sobre el. De este modo una baja dependencia es un beneficio automático de un diseño orientado a objetos.

Cohesión

En contraste con la medida de la interdependencia de los módulos, la cohesión se refiere a la forma interna en la que se construye un módulo. Entre más cohesión exista en un módulo, más se relacionan los componentes del módulo entre sí y con el desarrollo de la misma función. En otras palabras, un módulo presenta *cohesión* si todos los elementos del mismo se direccionan hacia el desarrollo esencial de la misma función.

El ideal de un buen diseño de sistemas es la *cohesión funcional*, donde cada elemento de un proceso es esencial para el desarrollo de una función única. Un módulo con cohesión funcional no solo lleva a cabo la función para la cual fue diseñado, sino que solo lleva a cabo esa función y nada más.

La noción de cohesión puede expandirse a diseños orientados a objetos teniendo en mente la meta general: colocar los objetos y las acciones juntas solo cuando tengan un sentido común y palpable. Se dice que el módulo que parte de un diseño orientado a objetos tiene cohesión si cada método o acción es esencial al objeto. Es difícil diseñar un sistema orientado a objetos que no tenga cohesión dado que el proceso de composición obliga a las acciones a ubicarse con los objetos que afecta.

Extensión del control

Un módulo controlando a muchos otros módulos generalmente indica que el módulo controlador esta haciendo demasiado, probablemente llevando a cabo más de una función. Por ejemplo, en muchos casos puede necesitarse buscar un caracter en particular en una cadena, si se diseña un módulo de propósito general para hacer esa tarea y entonces se invoca a ese módulo desde muchos otros, el diseño resultante es más eficiente y fácil de modificar que uno en el cual proliferan funciones de búsqueda de caracteres en una cadena. De este modo, para un diseño con un gran número de niveles se crea un conjunto de *módulos de utilerías:* herramientas para construir bloques utilizados por otros módulos para efectuar tareas de uso común.

Estrategias de diseño inicial

En general las características de un buen diseño son:

1. Baja dependencia entre módulos
2. Módulos con cohesión
3. Mínimo número de módulos con más elementos controlados que sus propios controles.
4. Efecto de un módulo limitado al alcance de sus controles

Estas características son metas por alcanzar en el diseño del sistema pues hacen a este fácil de construir, probar, corregir y mantener. Es útil conocer estas características antes de iniciar el diseño de modo que puedan incluirse en el sistema a desarrollar.

6.2 De la determinación de requerimientos al diseño conceptual

El diseño de un sistema puede depender de muchas cosas: los requerimientos, el uso de herramientas automatizadas, el tipo de sistema, el tipo de disciplina de diseño utilizada, etc.

Para el caso de estudio del presente trabajo, se comenzó intentando describir el sistema como un conjunto de módulos. Se desea que los módulos se diferencien por la función que realizan, por tanto, una primera aproximación al diseño de estos puede ser como la figura 6.1, donde la operación de la Dirección de Destino de Bienes se divide en módulos que apoyen la operación definida por el organigrama descrito en la figura 5.1.

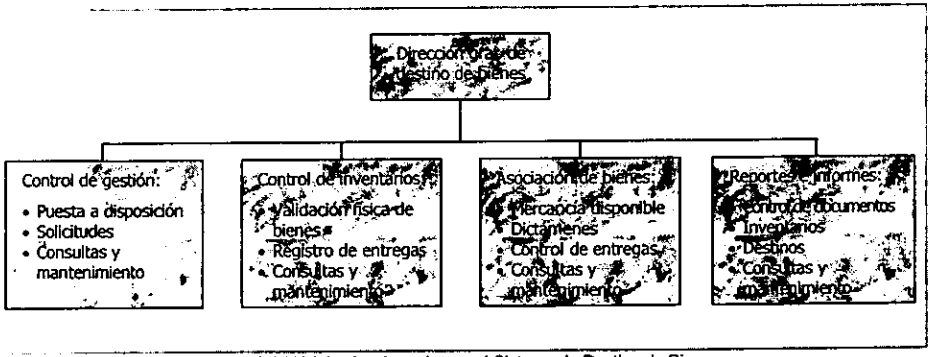


Figura 6.1 Módulos involucrados en el Sistema de Destino de Bienes

A continuación se considera cada proceso descrito en el diagrama de contexto de la figura 5.4 y en los diagramas de flujo de datos que se derivan de este para analizar que información adicional habrá de requerirse. Por ejemplo, el módulo asociado a Registro y control de operaciones puede determinar también el rango de fechas, los tipos de destino, y los beneficiarios de la Dirección de Destino de Bienes a incluirse.

El segundo nivel de descomposición describe cada uno de los bloques de esta figura a mayor detalle. Por ejemplo, la función de consulta puede además validar las operaciones que han llegado a buen término para considerar en los reportes oficiales de operaciones ó solicitar información general en la base de datos.

Se continua detallando los módulos en el diseño hasta que cada uno describe exactamente una función y existen descripciones de datos para todo el sistema. Durante el diseño se tiene especial cuidado de las características descritas en este capitulo: modulabilidad, cohesión, dependencia, etc.

En el siguiente punto del diseño y después de su análisis como un todo, se decide si el mismo es de una calidad aceptable. ¿Hay aspectos del diseño que debieran ser diferentes o pudieran mejorarse? ¿El diseño es lo suficientemente claro para entenderse por cualquiera? Este es el punto en el proceso de desarrollo cuando es más fácil reconsiderar y rediseñar. Una decisión de diseño pobre en este momento requerirá mucho tiempo y recursos para ser reparada o cambiarse después. Si es necesario, se rechaza el diseño previo, se rediseña entonces para construir un conjunto de módulos mejores que el diseño anterior pero que lleven a cabo la misma función.

Una vez decidida la estructura modular del sistema, se amplia el diseño describiendo su interfaz con los usuarios, lo anterior basado en los siguientes puntos:

1. Los requerimientos descritos por el usuario
2. El conocimiento del factor humano
3. La experiencia propia en el diseño de sistemas similares

Por ejemplo, puede decidirse colocar un cierto número de campos en un reporte en base en las solicitudes del usuario, dentro de un formato estético y que acomode en papel carta con un tamaño de 12 puntos para los caracteres.

Si se decide distribuir el procesamiento entre varios equipos, el diseño del sistema debe describir que nodos realizaran cada función. Si nuestros cuatro módulos principales son independientes, entonces los bloques funcionales pueden dividirse en subsistemas separados. Las dependencias entre las funciones se detallan y la comunicación entre los nodos del sistema deben describirse.

Una vez que el sistema este completo (es decir que incorpore todos los requerimientos del cliente), se presenta tanto como un diseño conceptual como técnico. El primero incluye la descripción de las pantallas de entrada de datos y los reportes de salida, pero no explica la división por módulos del sistema, esta se describe únicamente en el diseño técnico.

Similarmente, el diseño conceptual explica las opciones disponibles a los usuarios en cualquier punto del proceso del sistema, mientras que el diseño técnico describe que módulos controlan que etapas del procesamiento del sistema. Por ejemplo, el diseño conceptual contiene una sección que se lee:

Desde la ventana principal del módulo de Control de gestión se permitirá al operador:

1. Dar de alta mercancías y vehículos en la base de datos de inventario según se describan en el Oficio de puesta a disposición (1.1.1)
2. Dar de alta solicitudes de donación ó asignación, para aquellos beneficiarios autorizados de la Dirección de Destino de Bienes según un catálogo actualizado (1.1.3)
3. Efectuar consultas en la base de datos de inventario en función a los bienes referidos en las solicitudes de donación o asignación (1.2.1)
4. Efectuar consultas en la base de datos de destinos que reflejen el histórico de solicitudes para un mismo beneficiario (1.2.2)

Las referencias numéricas muestran cual de los requerimientos detectados en el capítulo anterior generan las funciones del diseño en particular.

Como se explico en el punto 6.1 del presente capítulo, el diseño técnico describe las mismas funciones señalando los módulos que las llevan a cabo. La descripción de los módulos se presenta de modo que sea consistente con la técnica de diseño utilizada.

6.3 Diseño técnico

El modelo entidad-relación introducido en el capítulo 1 representa al esquema de información de la organización, es una descripción a nivel conceptual independiente de cualquier DBMS. Una entidad es cualquier objeto tangible en el mundo real representado aquí por un rectángulo. Los atributos son representaciones de propiedades que las entidades tienen en el mundo real. El conjunto de valores permitidos para un atributo es su dominio, siendo en realidad un mapeo de las entidades hacia el dominio de los atributos.

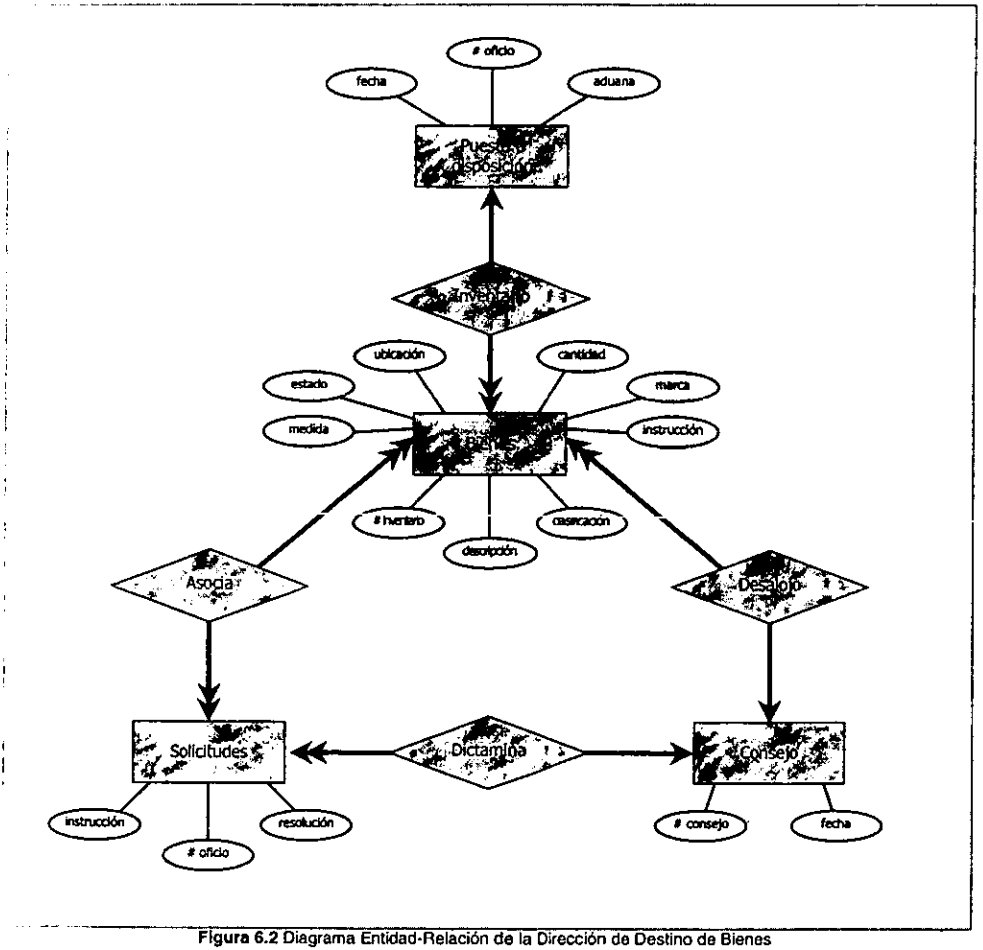


Figura 6.2 Diagrama Entidad-Relación de la Dirección de Destino de Bienes

La figura 6.2 muestra el diagrama entidad-relación que incorpora las entidades y relaciones involucradas en los procesos estudiados en el capítulo anterior y para el cual se identificaron las siguientes entidades y atributos:

- **Puesta a Disposición:** Número de oficio, fecha y aduana. Se asume que un mismo oficio de puesta a disposición puede involucrar varios bienes.
- **Bienes:** Número de inventario, descripción, clasificación, cantidad, unidad de medida, estado físico, marca, ubicación, oficio de instrucción de su destino final. Se asume que un bien puede considerarse en un solo oficio de puesta a disposición, puede asociarse a una única solicitud y ser desalojado de los recintos fiscales por instrucción del consejo en una única ocasión.
- **Solicitudes:** Número de oficio, beneficiario, resolución del Consejo Asesor. Se asume que una solicitud se vera solventada por uno o más bienes y que la solicitud se pondrá a consideración del Consejo Asesor en una única ocasión.
- **Consejo:** Número de consejo, fecha. Una misma reunión del consejo asesor y solo esa reunión en particular habrá de solventar varias solicitudes y ordenar el desalojo de varios bienes para su destino final.

Las relaciones establecidas son:

- **Puesta a disposición-Bienes (inventario),** que es una relación de uno a muchos, dada la consideración de que un bien solo es puesto a disposición una vez.
- **Bienes-Solicitudes (asocia),** que es una relación muchos a muchos pues una solicitud puede requerir de varios bienes, y un mismo bien, cuando se trata de un lote, puede asociarse a varias solicitudes.
- **Bienes-Consejo (desalojo),** relación uno a muchos según la cual un bien no puede ser destinado en más de una ocasión y por cada reunión del Consejo se da destino a varios bienes.
- **Solicitudes-Consejo (dictamen),** relación uno a muchos que establece que una solicitud se someterá a fallo del Consejo solamente una vez y que en una misma reunión del Consejo Asesor se resolverán varias solicitudes.

Del modelo E-R al modelo relacional

Un diagrama entidad-relación puede convertirse en un modelo relacional de modo relativamente fácil. Las entidades representadas por rectángulos se convierten en relaciones representadas por tablas. El nombre de la tabla es el mismo que el nombre de la entidad, que es el nombre escrito dentro del

rectángulo. Los atributos representados por óvalos se convierten en atributos de la relación, o encabezados de columna de la tabla. Aquellos atributos que puedan representarse por un determinado conjunto de valores se constituirán en un catálogo del sistema, esto es, una tabla en la cual se representan todos los valores posibles del atributo en cuestión, con un identificador único para hacer referencia a determinado valor del atributo que concuerde con aquel de las entidades en particular.

Las relaciones también pueden representarse por medio de tablas. Aunque no se representan explícitamente en el diagrama entidad-relación, se sobreentiende que una tabla de relación contiene las llaves primarias de las entidades asociadas como atributos propios. Dicha tabla contará con encabezados de columnas para cada una de las llaves primarias de las entidades asociadas, además de columnas para sus atributos descriptivos si existieren.

En algunos casos, puede elegirse no representar las relaciones por medio de una tabla explícita. Por ejemplo, si se tiene una dependencia entre dos entidades (como el número de oficio de puesta a disposición que comparten las entidades *bienes* y *puesta a disposición*) la llave primaria de la entidad principal ya forma parte de la tabla de la entidad asociada, así que no es necesario utilizar otra tabla para representar la conexión. Siempre que la relación sea de uno a uno o de uno a muchos, es posible utilizar las llaves foráneas para indicar la relación. Si la relación *A* hacia *B* es uno a uno, es posible incluir la llave de cualquier relación en la tabla de la otra para mostrar la conexión. Si *A* hacia *B* es uno a muchos, se puede colocar la llave de *A* (la entidad que aporta un elemento a la relación) en la tabla de *B* (la entidad que aporta "varios" elementos), donde se convierte en una llave foránea. El único caso para el cual es imposible romper la relación es el caso de una relación muchos a muchos. Aquí, el único modo de mostrar la relación es por medio de una tercera tabla.

Siempre que se tenga la opción de representar relaciones por medio de una tabla individual o mediante el uso de llave foráneas, la elección habrá de hacerse en función a la aplicación. El tener una tabla separada para la relación brinda la máxima flexibilidad, permitiendo efectuar cambios en las asociaciones fácilmente. Sin embargo, requiere de establecer una liga siempre que se utilice la relación, lo cual puede redituar en un pobre desempeño de la aplicación. El diseñador debe elegir entre flexibilidad y eficiencia, dependiendo en el criterio de la aplicación. Para el caso de la Dirección de Destino de Bienes, dada la irrevocabilidad de las resoluciones del Consejo Asesor la flexibilidad no es una prioridad por lo que las relaciones entre las entidades habrán de representarse por medio de llaves foráneas.

Para el caso que nos ocupa las entidades del diagrama entidad-relación de la figura 6.2 pueden representarse inmediatamente por las siguientes tablas:

PUDI(no_pudi, fec_pudi, aduana)

BIENES (no_inventario, no_pudi, no_solicitud, clasificación, descripción, cantidad, u_medida, marca, estado, ubicación, validado)

SOLICITUD (no_solicitud, beneficiario, no_resolucion)

CONSEJO (no_consejo, fecha)

La relación *desalojo* representa una relación uno a muchos, conectando las entidades *bienes* y *consejo*. Esta relación se basa en el número de oficio de instrucción por el cual se ordena el destino final de los bienes. Como es necesario conocer datos adicionales de este oficio tales como la fecha, el texto de la resolución, la aprobación y la fecha de entrega de los bienes, la relación se representará con otra tabla para no duplicar estos detalles en cada uno de los bienes de una misma resolución:

INSTRUCCION (no_inst, fec_inst, instrucción, aprobó, fec_inventario, fec_entrega)

El mismo caso aplica para el oficio de resolución por el cual se dictaminan cada una de las solicitudes sometidas a fallo del Consejo Asesor y que se refiere a la relación *dictamina*:

RESOLUCION (no_resolución, no_consejo, fec_resol, resolución, aprobó, fec_benef)

En el caso de la relación *inventario* que se refiere a las entidades *puesta a disposición* y *bienes*, esta se refleja a través de la inclusión de la llave primaria de la entidad *pudi* como llave foránea de la tabla *bienes* (campo *no_pudi*).

Por ultimo, la relación *asocia*, que involucra a las tablas *bienes* y *solicitudes*, es del tipo de muchos a muchos lo que hace imposible establecer la relación por medio de llaves foráneas. Es necesario aquí incluir la tabla *entregas*, que incluye como llaves foráneas las llaves primarias de las tablas *bienes* y *solicitudes*, así como la descripción adicional de la relación entre ambas entidades:

ENTREGAS (no_entrega, no_inventario, no_solicitud, no_instrucción, cantidad, valor, fec_entrega, observaciones, destino)

El detalle de las tablas que incluye sus relaciones y catálogos se muestra en la documentación que acompaña al último punto de este capítulo.

6.3.1 Elección de la plataforma de desarrollo

En un artículo de 1985, Codd quien propuso el modelo relacional de base de datos publicó los principios que un sistema administrador de base de datos debe utilizar para considerarse "completamente relacional":

1. *Representación de la información*: En un nivel lógico, toda la información debe representarse solamente como valores en tablas.
2. *Acceso garantizado*: Debe ser posible el acceso a cualquier registro en la base de datos dando el nombre de la tabla, nombre de la columna y valor de la llave primaria.
3. *Representación de valores nulos*: El sistema debe ser capaz de representar valores nulos de un modo sistemático, sin importar el tipo de dato del atributo. Los valores nulos deben ser distintos de cero, cualquier otro número o cadena vacía si es el caso.
4. *Catalogo relacional*: El catálogo de sistema que contiene las descripciones lógicas de la base de datos debe representarse del mismo modo que los datos ordinarios.
5. *Lenguaje de datos*: Sin importar el número de lenguajes que soporte, la base de datos debe incluir un lenguaje que permita instrucciones expresadas como cadenas de caractéres para la definición de datos, vistas manipulación, reglas de identidad, autorización de usuarios y un método de identificar unidades para recuperación.
6. *Vistas*: Cualquier vista que sea actualizable en teoría puede actualizarse en tiempo real por parte del sistema.
7. *Operaciones de inserción, actualización y diminución*: Cualquier relación que pueda manejarse por medio de un operando de ejecución también puede manejarse por medio de operaciones de inserción, actualización y eliminación de registros.
8. *Independencia física de datos*: Los programas de aplicación son inmunes a cambios hechos a los medios de almacenamiento o métodos de acceso.
9. *Independencia lógica de datos*: Los cambio hechos a nivel lógico que no afecten el contenido en este mismo nivel no requieren modificación alguna en las aplicaciones.
10. *Reglas de integridad*: Las constantes que dan integridad a los datos (entidades e integridad referencial) deben poder especificarse en el sublenguaje de datos y almacenarse en el catálogo. No debe utilizarse código de programación para expresar estas constantes.
11. *Independencia de distribución*: El sublenguaje de datos debe ser tal que si la base de datos es distribuida, los programas de aplicación y los comandos de los usuarios no necesitan cambiarse.
12. *No subversión*: Si el sistema permite un lenguaje que soporte acceso por registro individual, cualquier programa que utilice este tipo de acceso no puede ignorar las constantes de integridad expresadas en el lenguaje de mayor nivel.

Un DBMS se encarga por tanto de separar la administración de las aplicaciones de bases de datos (servidor) de las aplicaciones individuales (clientes) que despliegan, imprimen y actualizan la información en la base de datos. La mayoría de los sistemas cliente/servidor utilizados hoy en día operan bajo el sistema operativo Unix, pero Windows NT 4.0 esta ganando mercado a Unix como sistema operativo de red. La instalación y administración de Windows NT 4.0 y SQL Server 6.5 es más simple que la necesaria en Unix y los DBMS basados en este sistema operativo, por lo tanto, muchas organizaciones pequeñas, que tradicionalmente empleaban sistemas compartidos de archivos para bases de datos multiusario están migrando a sistemas cliente/servidor basados en SQL Server 6.5 con Windows NT 4.0 como sistema operativo de red.

Por su parte, las aplicaciones cliente son responsables de la creación de consultas y su envío al DBMS, así como del procesamiento de los registros resultado de la consulta. Una de las principales ventajas de utilizar un DBMS basado en SQL es que la integridad, seguridad y consistencia de la base de datos se efectúan por parte del DBMS, por lo tanto, el código que implementa dichas acciones no necesita incorporarse a las aplicaciones cliente. Las amplias facilidades de conexión con bases de datos remotas de Visual Basic 5.0 de Microsoft convierten a esta plataforma en ideal para el desarrollo de aplicaciones cliente. El término *aplicación cliente* se utiliza para describir una aplicación que puede seleccionar conjuntos de información contenida en una base de datos y desplegar aquellos que sean de utilidad al usuario final. Algunas aplicaciones cliente solo despliegan datos, otras permiten además actualizar la base de datos editando, agregando o eliminando registros. El sistema de administración de la base de datos formará parte exclusiva del *servidor*.

Uno de los principales objetivos de Microsoft para Visual Basic es el brindar un ambiente de programación en 32 bits para complementar el grupo de programas de red Microsoft BackOffice que corren sobre Windows NT. BackOffice incluye administración distribuida de bases de datos con SQL Server 6.5 y Visual Basic 5.0 la capacidad de desarrollo de aplicaciones gráficas con la conectividad necesaria para ejecutar de manera remota sentencias SQL sobre el DBMS.

La conexión entre el DBMS y las aplicaciones cliente se lleva a cabo a través de un controlador de origen de datos ODBC[14]. Para abrir una conexión a un origen de datos es necesario definir previamente el nombre del origen de datos (DSN), un identificador válido de usuario (UID) y la contraseña asociada (PWD).

[14] Conexión Abierta a Bases de Datos, por sus siglas en inglés

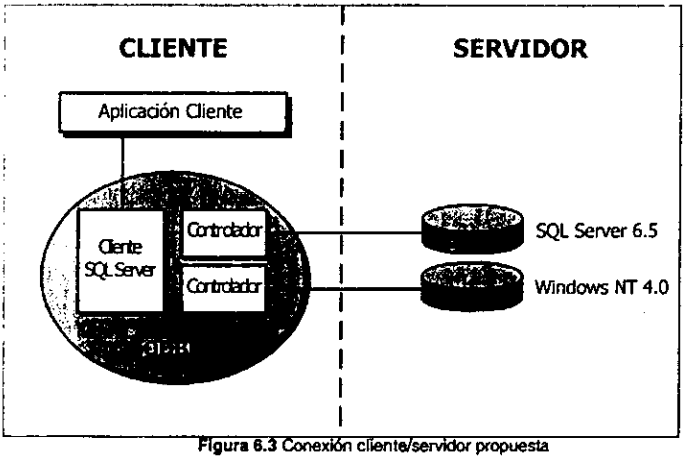


Figura 6.3 Conexión cliente/servidor propuesta

6.4 Documentación

Uno de los productos del proceso de diseño es un documento que describe el sistema a construir. Como se menciono anteriormente, la primera parte de la descripción dice al cliente lo que el sistema habrá de realizar; la segunda parte utiliza términos técnicos donde es necesario para decir a los diseñadores del programa lo que hará el sistema. Dado lo anterior, el contenido de ambas partes se repetirá, pero la manera de expresarlo no.

El documento del diseño del sistema contiene una sección señalando las *políticas* consideradas en la generación del sistema. Esta filosofía ayuda tanto al cliente como al analista a entender como y porque ciertas partes del diseño encajan unas con otras.

El diseño también incluye la *descripción de los componentes del sistema*. Un componente se refiere al como interactúa el usuario con el sistema, incluyendo lo siguiente:

- Menús y otros formatos de presentación en pantalla
- Interfaces con el usuario
- Formatos de reportes
- Entradas: de donde provienen los datos, como se les da formato, en que medio se almacenan
- Salidas: a donde se envían las salidas, como se les da formato, en que medio se almacenan
- Características funcionales generales
- Requerimientos de desempeño
- Procedimientos de archivos
- Requerimientos para el manejo de errores

Un diagrama jerárquico o algún otra figura muestra la *organización y estructura* general del sistema, incluyendo el flujo de datos así como la modulabilidad funcional. Se incluyen los varios niveles de interacción de los módulos.

Si el sistema es distribuido, la configuración en el diseño se detalla lo suficiente para mostrar la *topología de la red*, como se accesa a los diferentes nodos en la red y la ubicación de las funciones de los nodos. Si los requerimientos del sistema incluyen constantes dependientes del tiempo, o si los nodos de la red deben sincronizarse, el diseño describe la *sincronización* del sistema. El diseño también puede incluir directivas para garantizar la *integridad de los datos*: procedimientos que aseguren que los datos son precisos o que pueden recuperarse después de una falla.

Si el cliente lo requiere, los elementos del diseño pueden mencionar la *medición del desempeño del sistema*. Otros requerimientos pueden incluir *localización y aislamiento de errores, reconfiguración* del sistema o medidas de *seguridad* especiales. El diseño explica como se solventan estos requerimientos.

Finalmente el diseño es *asociado con los requerimientos* para demostrar como los componentes del diseño se derivan de ellos. Esto obliga a revisar que todos los requerimientos se cubran y la consistencia del proyecto. Adicionalmente, esta referencia hará más fácil de rastrear las mejoras o modificaciones posteriormente. Por ejemplo, si un requerimiento cambia, la referencia a estos apunta a los cambios necesarios en el diseño.

Diseño conceptual para el Sistema de Control Administrativo de Bienes

Política de operación del Sistema de Control Administrativo de Bienes

- Toda la documentación recibida en la Dirección de Destino de Bienes se registrará en el sistema a través del área de Control de Gestión: oficios de puesta a disposición, solicitudes de donación o asignación y asuntos generales. Para el caso de las solicitudes no se aceptaran aquellas que no provengan de un beneficiario propio de la dirección de Destino de Bienes (entidades federativas e Instituciones de Asistencia Privada) y en ningún caso se recibirá más de una vez una misma solicitud.
- El área de inventarios se encargará de validar físicamente lo asentado en los oficios de puesta a disposición, hacer las actualizaciones necesarias en la base de datos del sistema, así como asegurarse que todo bien solo sea referido en un único oficio de puesta a disposición. No se podrá disponer de los bienes a los que hacen referencia los oficios de puesta a disposición sino hasta que el área de inventarios coteje sus datos y estado físico.
- Para generar la carpeta informativa en la que se apoyará el Consejo Asesor para determinar el destino final de los bienes propiedad del Fisco Federal, el área de Operación de Destino se basará en los reportes de solicitudes recibidos por el área de Control de Gestión, un estadístico de las operaciones atendidas, pendientes y rechazadas así como de los reportes de inventario real generados a partir de lo asentado en los oficios de puesta a disposición y la propia área de Inventarios de la Dirección de Destino de Bienes.
- A partir de los dictámenes del Consejo Asesor, el área de Operación de Destino instruirá al área de Inventario para hacer entrega de los bienes que cubran las resoluciones generadas e informará a los solicitantes el resultado de su gestión.
- El área de Registro y Control de Operaciones se encargará de aclarar las discrepancias en la documentación referente a los bienes propiedad del Fisco, verificar que las resoluciones del Consejo Asesor se lleven a buen termino así como de generar el informe de operaciones de la Dirección de Destino de Bienes.

Módulo de Control de Gestión

Desde la ventana principal del módulo de Control de gestión se permitirá al operador:

1. Dar de alta mercancías y vehículos en la base de datos de inventario según se describan en el Oficio de puesta a disposición (1.1.1)
2. Dar de alta solicitudes de donación ó asignación, para aquellos beneficiarios autorizados de la Dirección de Destino de Bienes según un catálogo actualizado (1.1.3)
3. Llevar un registro de los asuntos generales turnados a la Dirección de Destino de Bienes y turnarlos al área de la misma a la cual correspondan (1.1.2)
4. Efectuar consultas en la base de datos de inventario en función a los bienes referidos en las solicitudes de donación o asignación (1.2.1)

5. Efectuar consultas en la base de datos de destinos que reflejen el histórico de solicitudes para un mismo beneficiario (1.2.2)

Reportes:

1. Solicitudes
2. Oficios de puesta a disposición
3. Beneficiarios

Módulo de Inventario

Dada la doble función del área de Inventario el módulo que apoyará su operación contará con una ventana principal para control de entradas y otra para el control de salidas del almacén. La primera de estas permitirá al operador:

1. Efectuar consultas a la base de datos de bienes para así cotejar los datos declarados de estos en su oficio de puesta a disposición y en su caso corregir la información que respecto a los bienes se tiene en función al estado de estos (2.2.1)
2. Generar un reporte detallado de las discrepancias detectadas en los oficios de puesta a disposición, así como un reporte de bienes que no se encuentren en la base de datos principal para el área de Registro y Control de Operaciones (2.2.2)
3. Generar un reporte detallado de los bienes con documentación en regla para el área de Operación de Destino (2.2.2)

Para el caso de las operaciones de entrega de bienes, el sistema permitirá al operador:

1. Efectuar consultas a la base de datos de oficios de instrucción y de los bienes referidos para su entrega (2.3.1)
2. Actualizar los registros de la base de datos de bienes y de destinos para reflejar así la entrega de los de bienes de los recintos fiscales (2.3.2)
3. Actualizar los registros de la base de datos de bienes y de destinos para reflejar así la destrucción de las mercancías prohibidas o peligrosas (2.3.2)

Reportes:

1. Bienes (mercancías y vehículos)
2. Entregas
3. Destrucciones
4. Validación física de bienes

Módulo de Destinos

La principal función de este módulo es el integrar la carpeta informativa para el Consejo Asesor y registrar las resoluciones del mismo en la base de datos, para ello el sistema permitirá:

1. Efectuar consultas del histórico de entregas para un determinado beneficiario, a las solicitudes de beneficiarios autorizados y al inventario de bienes factibles de destinar (3.1.1)
2. Dar de alta en la base de datos de destinos el oficio de instrucción para entrega de mercancías (3.2.1)
3. Actualizar el estado de las solicitudes para reflejar el dictamen del Consejo Asesor (3.2.2)

Reportes:

1. Carpeta informativa para el Consejo Asesor
2. Oficio de instrucción
3. Oficio de resolución

Módulo de Control de operaciones

La coordinación de funciones de las diferentes áreas de la Dirección de Destino de Bienes se apoyará de este módulo. Desde la ventana principal del módulo de Control de gestión se permitirá al operador:

1. Efectuar consultas a la base de datos de bienes y de oficios de puesta a disposición para conocer las existencias, entradas y salidas al almacén (4.1)
2. Efectuar consultas a la base de datos de solicitudes y destinos para conocer las resoluciones del Consejo Asesor y el estado de avance en la gestión de las mismas (4.2)
3. Generar reportes estadísticos del inventario de bienes y de la disposición de los mismos (4.3)

Reportes:

1. Inventario
2. Destinos
3. Entregas
4. Contabilidad

Diseño técnico para el Sistema de Control Administrativo de Bienes

Arquitectura del sistema

Servidor de base de datos:

Sistema Operativo de Red:	Windows NT 4.0
DBMS:	Microsoft SQL Server 6.5

Funciones del DBMS:

- Crear nuevas bases de datos y los archivos necesarios para contenerlas (muchas bases de datos pueden residir en un único archivo en disco).
- Implementar un esquema de seguridad para la base de datos para evitar que usuarios no autorizados accesen a la base de datos y a la información que contiene.
- Mantener un catálogo de objetos en la base de datos, incluyendo la información del propietario (creador) de la base de datos y de las tablas que contiene.
- Generar una bitácora de todas las modificaciones hechas a la base de datos de manera que esta pueda reconstruirse a partir de un respaldo previamente realizado en combinación con la información contenida en la bitácora (en caso de alguna falla en el equipo).
- Mantener la integridad referencial, la consistencia y llevar las reglas de integridad para evitar la corrupción de los datos contenidos en las tablas.
- Administrar el acceso a los datos de modo que diferentes usuarios puedan tener acceso a los datos sin encontrar retrasos considerables en el despliegue o actualización de los datos.
- Interpretar las consultas realizadas a la base de datos por aplicaciones de usuarios y devolver o actualizar los registros que correspondan a los criterios señalados en dicha consulta.

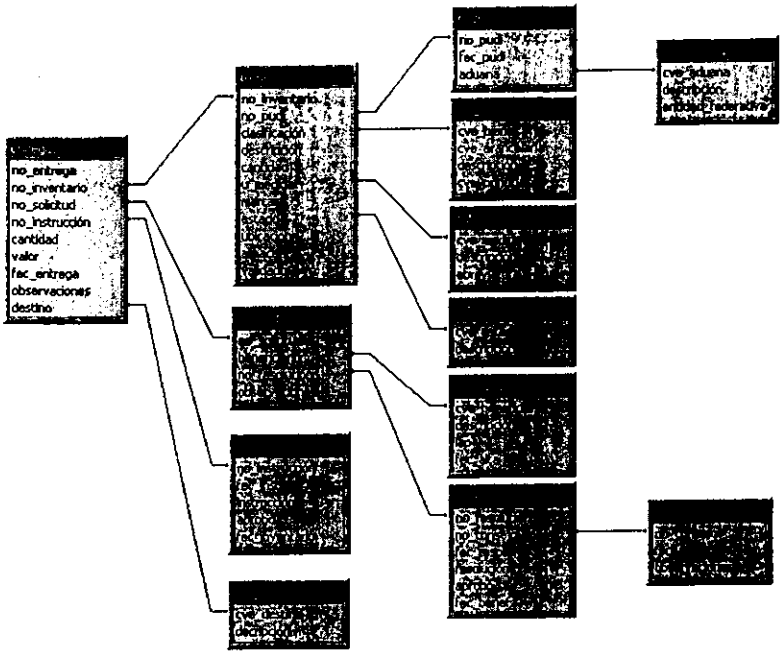
Aplicaciones Cliente:

Sistema Operativo de Red:	Windows 98
Entorno de programación:	Visual Basic 5.0

Funciones de la aplicación cliente:

- Proporcionar la interfaz visual para presentar información y reunir datos.
- Ofrecer los servicios de empresa necesarios para entregar las capacidades necesarias de la organización e integrar al usuario con la aplicación para que realice una tarea de empresa mediante la aplicación de procedimientos formales y reglas de la empresa a los datos correspondientes.
- Asegurar los servicios de datos necesarios para realizar la tarea o aplicar la regla de la empresa aislando al usuario de la interacción directa con la base de datos.

Estructura de datos



Programación

Una vez definidas las expectativas que del sistema a desarrollar se tienen se habrá de proceder a su implementación física. Con base a los lineamientos de operación definidos en el diseño conceptual y apoyados de la plataforma de programación, administrador de base de datos y estructura de la base de información establecidas en el diseño técnico se encaminarán esfuerzos a la obtención del sistema final, documentado debidamente la estructura de programación en la que se basa.

7.1 Programación en Visual Basic

¿Qué es Visual Basic? La palabra "Visual" hace referencia al método que se utiliza para crear la interfaz gráfica de usuario. En lugar de escribir numerosas líneas de código para describir la apariencia y la ubicación de los elementos de la interfaz, simplemente es necesario arrastrar y colocar objetos prediseñados en su lugar dentro de la pantalla.

La palabra "Basic" hace referencia al lenguaje BASIC[15], un lenguaje utilizado por más programadores que ningún otro lenguaje en la historia de la informática o computación. Visual Basic ha evolucionado a partir del lenguaje Basic original y ahora contiene centenares de instrucciones, funciones y palabras clave, muchas de las cuales están directamente relacionadas con la interfaz gráfica de Windows

Para entender el proceso de desarrollo de una aplicación, es útil comprender algunos de los conceptos clave alrededor de los cuales está construido Visual Basic. Puesto que Visual Basic es un lenguaje de desarrollo para Windows, es necesario familiarizarse con el entorno Windows.

[15] Beginners All-Purpose Symbolic Instruction Code

7.1.1 Funcionamiento de Windows: ventanas, eventos y mensajes

Un estudio profundo del funcionamiento interno de Windows necesitaría un libro completo. No es necesario tener un profundo conocimiento de todos los detalles técnicos. Una versión reducida del funcionamiento de Windows incluye tres conceptos clave: ventanas, eventos y mensajes.

Una ventana es simplemente una región rectangular con sus propios límites: una ventana de explorador en Windows 95, una ventana de documento dentro de algún procesador de textos o un cuadro de diálogo que emerge para dar un aviso emergente. Aunque éstos son los ejemplos más comunes, realmente hay otros muchos tipos de ventanas. Un botón de comando es una ventana. Los iconos, cuadros de texto, botones de opción y barras de menús son todos ventanas.

El sistema operativo Microsoft Windows administra todas estas ventanas asignando a cada una un único número identificador (controlador de ventana o **hWnd**). El sistema controla continuamente cada una de estas ventanas para ver si existen signos de actividad o eventos. Los eventos pueden producirse mediante acciones del usuario, como hacer clic con el *mouse* (ratón) o presionar una tecla, mediante programación o incluso como resultado de acciones de otras ventanas.

Cada vez que se produce un evento se envía un mensaje al sistema operativo. El sistema procesa el mensaje y lo transmite a las demás ventanas. Entonces, cada ventana puede realizar la acción apropiada basándose en sus propias instrucciones para tratar ese mensaje en particular (por ejemplo, volverse a dibujar cuando otra ventana la ha dejado al descubierto).

Tratar todas las combinaciones posibles de ventanas, eventos y mensajes podría ser interminable, Visual Basic evita tener que tratar con todos los controladores de mensajes de bajo nivel. Muchos de los mensajes los controla automáticamente Visual Basic, mientras que otros se tratan como procedimientos de evento, permitiendo así crear rápidamente eficaces aplicaciones sin tener que tratar detalles innecesarios.

7.1.2 Descripción del modelo controlado por eventos

En las aplicaciones tradicionales o "por procedimientos", la aplicación es la que controla qué partes de código y en qué secuencia se ejecutan. La ejecución comienza con la primera línea de código y continúa con una ruta predefinida a través de la aplicación, llamando a los procedimientos según se necesiten.

En una aplicación controlada por eventos, el código no sigue una ruta predeterminada; ejecuta distintas secciones de código como respuesta a los eventos. Los eventos pueden desencadenarse por acciones del usuario, por mensajes del sistema o de otras aplicaciones, o incluso por la propia aplicación. La secuencia de estos eventos determina la secuencia en la que se ejecuta el código, por lo que la ruta a través del código de la aplicación es diferente cada vez que se ejecuta el programa.

Puesto que no puede predecir la secuencia de los eventos, el código debe establecer ciertos supuestos acerca del "estado del mundo" cuando se ejecute. Al hacer suposiciones (por ejemplo, que un campo de entrada debe contener un valor antes de ejecutar un procedimiento para procesar ese valor), debe estructurarse la aplicación de forma que se asegure que esa suposición siempre será válida (por ejemplo, deshabilitando el botón de comando que inicia el procedimiento hasta que el campo de entrada contenga un valor).

El código también puede desencadenar eventos durante la ejecución. Por ejemplo, cambiar mediante programación el contenido de un cuadro de texto hace que se produzca el evento **Change** del cuadro de texto. Esto causaría la ejecución del código (sí lo hay) contenido en el evento **Change**. Suponiendo que este evento sólo se desencadenará mediante la interacción del usuario, podrían verse resultados inesperados. Por esta razón es importante comprender el modelo controlado por eventos y tenerlo en cuenta al diseñar una aplicación.

Desarrollo interactivo

El proceso de desarrollo de las aplicaciones tradicionales se puede dividir en tres etapas diferentes: escritura, compilación y comprobación del código. A diferencia de los lenguajes tradicionales, Visual Basic utiliza una aproximación interactiva para el desarrollo, difuminando la distinción entre los tres pasos.

En la mayoría de los lenguajes, si se comete un error al escribir el código, el compilador intercepta este error cuando comience a compilar la aplicación, se debe encontrar y corregir el error y comenzar de nuevo con el ciclo de compilación, repitiendo el proceso para cada error encontrado. Visual Basic interpreta el código a medida que lo escribe, interceptando y resaltando la mayoría de los errores de sintaxis en el momento.

Hay tres pasos principales para crear una aplicación en Visual Basic:

1. Crear la interfaz.
2. Establecer propiedades.
3. Escribir el código.

7.2 Creación de la interfaz

Los formularios son la base para crear la interfaz de una aplicación. Pueden usarse formularios para agregar ventanas y cuadros de diálogo a la aplicación. También como contenedores de elementos que no son parte visible de la interfaz de la aplicación. Por ejemplo, un formulario que sirva como contenedor para gráficos que quiera presentar en otros formularios.

El primer paso para generar una aplicación de Visual Basic consiste en crear los formularios que van a ser la base de la interfaz de la aplicación. Después se dibujan los objetos que van a componer dicha interfaz en los formularios creados.

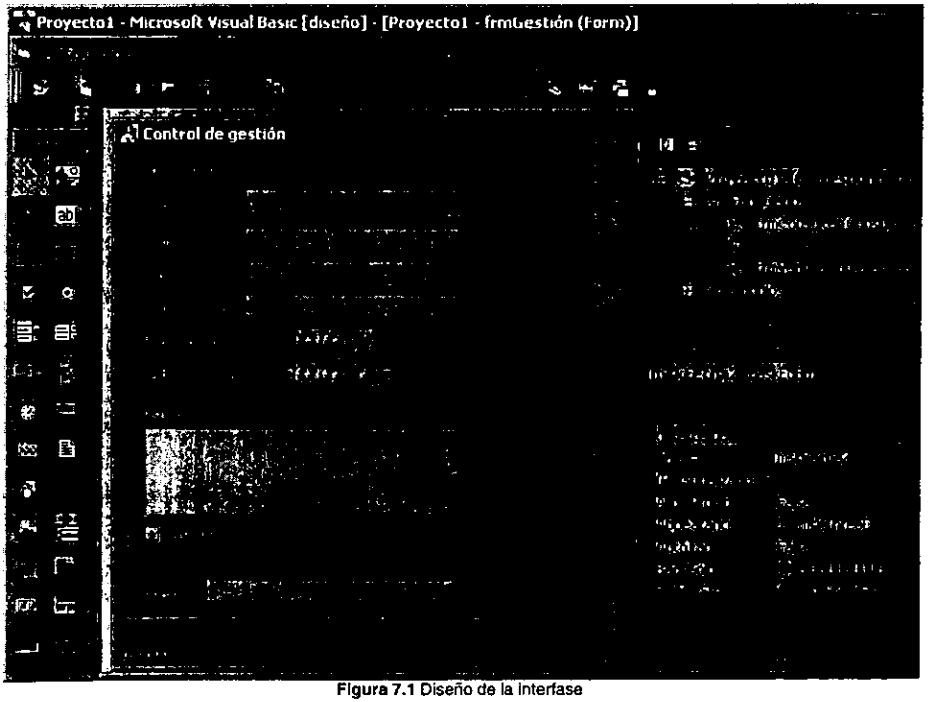


Figura 7.1 Diseño de la interfase

La interfaz de usuario es quizás la parte más importante de una aplicación; ciertamente, es la más visible. Para los usuarios, la interfaz es la aplicación; seguramente a ellos no les interesa el código que se

ejecuta detrás. Independientemente del tiempo y el esfuerzo que se haya empleado en la escritura y optimización del código, la facilidad de uso de la aplicación depende de la interfaz.

Cuando se diseña una aplicación se tienen que tomar muchas decisiones relacionadas con la interfaz. ¿Se debe utilizar el estilo de documento único o el de documentos múltiples? ¿Cuántos formularios diferentes se necesitarán? ¿Qué comandos se incluirán en los menús? ¿Se utilizarán barras de herramientas para duplicar funciones de los menús? ¿Cómo serán los cuadros de diálogo que interactúan con el usuario? ¿Qué nivel de asistencia se debe proporcionar?

Antes de empezar a diseñar la interfaz de usuario hay que considerar el propósito de la aplicación. El diseño de una aplicación principal de uso constante debe ser diferente del de una que sólo se utiliza ocasionalmente, durante breves periodos de tiempo. Una aplicación cuyo propósito fundamental sea de presentar información tiene unos requisitos distintos que otra que se utilice para obtener información. La audiencia prevista también debe influir en el diseño. Una aplicación destinada a usuarios principiantes requiere un diseño sencillo, mientras que una destinada a usuarios experimentados podría exigir uno más complejo.

7.2.1 Diseño pensando en el usuario

Aunque Visual Basic facilita la creación de una interfaz de usuario con sólo arrastrar controles dentro de un formulario, un poco de diseño previo puede marcar una gran diferencia en cuanto a la facilidad de uso de una aplicación.

Composición: la apariencia de una aplicación

La composición o distribución de un formulario no sólo afecta a su atractivo estético, sino que también tiene un tremendo impacto en la facilidad de uso de la aplicación. La composición incluye factores tales como la colocación de los controles, la coherencia de los elementos, su facilidad de uso, el uso del espacio en blanco y la sencillez del diseño.

Colocación de los controles

En la mayoría de los diseños de interfaces, no todos los elementos son de igual importancia. Es necesario un diseño cuidadoso para asegurar que los elementos más importantes sean inmediatamente accesibles para el usuario. Los elementos importantes o utilizados con más frecuencia deben tener una posición prominente; los elementos menos importantes deben estar relegados a posiciones menos prominentes.

En la mayoría de los idiomas, se enseña a leer de izquierda a derecha y de arriba a abajo en una página. Lo mismo se aplica a la pantalla de un equipo: los ojos de la mayoría de los usuarios irán primero a la parte superior izquierda de la pantalla, por lo que el elemento más importante debe estar allí. Por ejemplo, si la información de un formulario se refiere a un cliente, el campo con el nombre del cliente debe mostrarse allí donde se vea primero. Los botones, como **Aceptar** o **Siguiente**, deben estar situados en la parte inferior derecha de la pantalla; normalmente el usuario no los utilizará hasta que haya terminado de trabajar con el formulario.

La agrupación de elementos y controles también es importante. Como las funciones están relacionadas, los botones para desplazarse por una base de datos deben estar agrupados visualmente en lugar de estar esparcidos por el formulario. Esto mismo puede aplicarse a la información; los campos de nombre y

dirección se encuentran generalmente agrupados, ya que están estrechamente relacionados. En muchos casos, pueden utilizarse marcos para reforzar la relación entre controles.

Coherencia de los elementos de la interfaz

La coherencia es una virtud en el diseño de una interfaz de usuario. Una apariencia coherente aporta armonía a una aplicación: todo parece encajar perfectamente. La falta de coherencia de una interfaz puede provocar confusión y puede hacer parecer que la aplicación es caótica, desorganizada y barata, pudiendo llegar incluso a provocar al usuario dudas sobre la fiabilidad de la aplicación.

La coherencia entre los diferentes formularios de una aplicación es importante para su facilidad de uso. Si en un formulario utiliza un fondo gris y efectos en tres dimensiones y un fondo blanco en otro, parecerá que los formularios no están relacionados entre sí. Es necesario adoptar un estilo y mantenerlo en toda la aplicación, incluso si ello implica volver a diseñar algunas características.

Facilidades: la forma sigue a la función

Las *facilidades* son pistas visuales de la función de un objeto. Aunque el término no sea muy familiar hay muchos ejemplos de ellas. El manillar de una bicicleta tiene hendiduras para colocar los dedos, una facilidad que hace obvio que ése es el sitio donde poner las manos. Los botones, los mandos y los cambios luminosos en el tablero de un automóvil son todos facilidades; sólo con mirarlos puede deducirse su propósito.

La interfaz de usuario también hace uso de estas facilidades. Por ejemplo, los efectos tridimensionales utilizados en los botones de comando hacen pensar que están para ser presionados. Si diseñara un botón de comando con bordes planos, perdería esta facilidad y el usuario no tendría claro que es un botón de comando. Hay casos en que los botones planos pueden ser apropiados, como en juegos o en aplicaciones multimedia; está bien mientras se mantenga la coherencia en toda la aplicación.

Los cuadros de texto también aportan un grado de facilidad: los usuarios esperan que un cuadro con bordes y un fondo blanco contenga texto modificable. Aunque es posible presentar un cuadro de texto sin bordes, lo haría parecer una etiqueta y para el usuario no sería obvio que es modificable.

Uso del espacio en blanco

El uso del *espacio en blanco* dentro de la interfaz de usuario puede contribuir a resaltar elementos y aumentar su facilidad de uso. Aunque el espacio en blanco no tiene que ser necesariamente blanco, hace referencia al uso del espacio vacío entre y alrededor de los controles de un formulario. Si hay demasiados controles en un formulario puede parecer una interfaz congestionada, dificultando la localización de un campo o control concreto. Hay que incorporar espacio en blanco al diseño para hacer énfasis en los elementos del diseño.

El espaciado coherente entre los controles y la alineación vertical y horizontal de los elementos hacen que el diseño sea también más sencillo. De igual forma que el texto de un periódico está dispuesto en columnas con el mismo espacio entre las líneas, una interfaz ordenada la hace más legible.

Simplificar

Quizás el principio más importante en el diseño de la interfaz sea la sencillez. Cuando se trata de aplicaciones, si la interfaz parece difícil, probablemente lo sea. Pensando un poco se puede crear una interfaz que parezca (y sea) fácil de usar. Además, desde un punto de vista estético, siempre es preferible un diseño limpio y sencillo.

Un error frecuente en el diseño de la interfaz es el de modelarla a partir de objetos del mundo real. Por ejemplo, si se requiere de una aplicación para completar impresos de seguros la reacción natural sería diseñar una interfaz que duplicara de forma exacta el impreso en papel. Esto crea varios problemas: la forma y las dimensiones de un impreso en papel son diferentes a las de una pantalla, la duplicación de un impreso limita al uso de cuadros de texto y casillas de verificación, y el usuario no obtiene un beneficio real.

Es mucho mejor diseñar una nueva interfaz, proporcionando quizás un duplicado impreso (con presentación preliminar) del impreso original en papel. Agrupando los campos de forma lógica y utilizando una interfaz con fichas o varios formularios enlazados para presentar toda la información sin tener que hacer desplazamientos de pantalla. También es posible utilizar controles adicionales, como cuadros de lista con opciones predefinidas, lo que reduce la cantidad de datos que tiene que escribir el usuario.

La mejor prueba de la sencillez de uso de una aplicación es observar la aplicación en uso. Si un usuario típico no puede realizar inmediatamente una tarea deseada sin ayuda, quizás es necesario volver a diseñarla.

Imágenes e iconos

El uso de imágenes e iconos también puede agregar interés visual a la aplicación pero no deja de ser esencial un diseño cuidadoso. Las imágenes pueden transmitir información sin necesidad de incluir texto, pero las imágenes son percibidas de manera diferente por personas diferentes.

Las barras de herramientas con iconos que representen distintas funciones son un elemento útil de una interfaz, pero si el usuario no puede identificar la función representada por el icono, pueden ser contraproducentes. Al diseñar los iconos de una barra de herramientas, es de considerarse otras aplicaciones y las normas estándar ya establecidas. Por ejemplo, muchas aplicaciones utilizan una hoja de papel con una esquina doblada para representar el icono **Archivo Nuevo**. Puede que haya una mejor metáfora para esta función, pero si se representa de forma diferente, podría provocar confusión en el usuario.

7.3 Creación de procedimientos de evento

Los formularios son objetos que exponen las propiedades que definen su apariencia, los métodos que definen su comportamiento y los eventos que definen la forma en que interactúan con el usuario. Mediante el establecimiento de las propiedades del formulario y la escritura de código de Visual Basic para responder a determinados eventos se personaliza el objeto para cubrir las necesidades de la aplicación.

Los controles son objetos que están contenidos en los objetos de formularios. Cada tipo de control tiene su propio conjunto de propiedades, métodos y eventos, que lo hacen adecuado para una finalidad determinada. Algunos de los controles que se pueden usar en las aplicaciones son más adecuados para escribir o mostrar texto, mientras que otros controles permiten tener acceso a otras aplicaciones y procesan los datos como si la aplicación remota formara parte del código.

Este tema presenta los conceptos básicos del trabajo con formularios y controles, así como las propiedades, métodos y eventos que tienen asociados. Se explican muchos de los controles estándar y elementos específicos de formularios como menús y cuadros de diálogo.

7.3.1 Propiedades, métodos y eventos

Los formularios y controles de Visual Basic son objetos que exponen sus propios métodos, propiedades y eventos. Las propiedades se pueden considerar como atributos de un objeto, los métodos como sus acciones y los eventos como sus respuestas.

Un objeto cotidiano como un globo tiene también propiedades, métodos y eventos. Entre las propiedades de un globo se incluyen atributos visibles como el peso, el diámetro y el color. Otras propiedades describen su estado (inflado o desinflado) o atributos que no son visibles, como su edad. Por definición, todos los globos tienen estas propiedades; lo que varía de un globo a otros son los valores de estas propiedades.

Un globo tiene también métodos o acciones inherentes que puede efectuar. Tiene un método inflar (la acción de llenarlo de helio) o un método desinflar (expeler su contenido) y un método elevarse (sí se deja escapar). De nuevo, todos los globos pueden efectuar estos métodos.

Los globos tienen además respuestas predefinidas a ciertos eventos externos. Por ejemplo, un globo respondería al evento de pincharlo desinflándose o al evento de soltarlo elevándose en el aire.

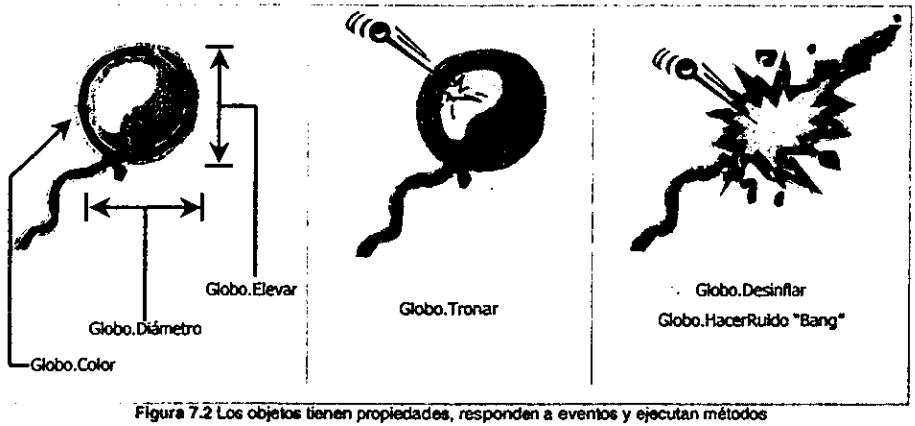


Figura 7.2 Los objetos tienen propiedades, responden a eventos y ejecutan métodos

Si se pudiera programar un globo, el código de Visual Basic podría ser como el siguiente. Para establecer las propiedades del globo:

```
Globo.Color = Rojo
Globo.Diámetro = 10
Globo.Inflado = True
```

La sintaxis del código: el objeto (Globo) seguido de la propiedad (Color) seguida de la asignación del valor (Rojo) permite modificar el color del globo desde el código si se repitiera esta instrucción sustituyendo el valor por otro diferente. Las propiedades también se pueden establecer en la ventana Propiedades de Visual Basic mientras se está diseñando la aplicación.

Los métodos de un globo se invocan de esta forma:

```
Globo.Inflar
Globo.Desinflar
Globo.Elevar 5
```

La sintaxis es similar a la sintaxis de las propiedades: el objeto (un nombre) seguido de un método (un verbo). En el tercer ejemplo hay un elemento adicional, llamado *argumento*, que indica la distancia que se eleva. Algunos métodos tendrán uno o más argumentos para describir más a fondo la acción que se va a ejecutar.

El globo puede responder a un evento como se muestra a continuación:

```
Sub Globo_Tronar()
        Globo.Desinflar
        Globo.HacerRuido "Bang"
        Globo.Inflado = False
        Globo.Diámetro = 0
End Sub
```

En este caso, el código describe el comportamiento del globo cuando se produce un evento **Pinchazo**: invoca el método **Desinflar** y luego invoca el método **HacerRuido** con un argumento **"Bang"** (el tipo de ruido que se va a hacer). Como el globo ya no está inflado, la propiedad **Inflado** tiene el valor **False** y la propiedad **Diámetro** adopta un nuevo valor.

Si bien no es posible programar un globo, sí lo es el programar un formulario o un control de Visual Basic. Como programador, se tiene el control de decidir qué propiedades se deben modificar, qué métodos se deben invocar o a qué eventos hay que responder para conseguir la apariencia y el comportamiento deseados.

7.3.2 Diseño de un formulario

Los objetos de un formulario son los elementos de desarrollo básicos de una aplicación de Visual Basic, las ventanas reales con las que interactúa el usuario cuando ejecuta la aplicación. Los formularios tienen sus propios eventos, propiedades y métodos con los que se puede controlar su apariencia y comportamiento.

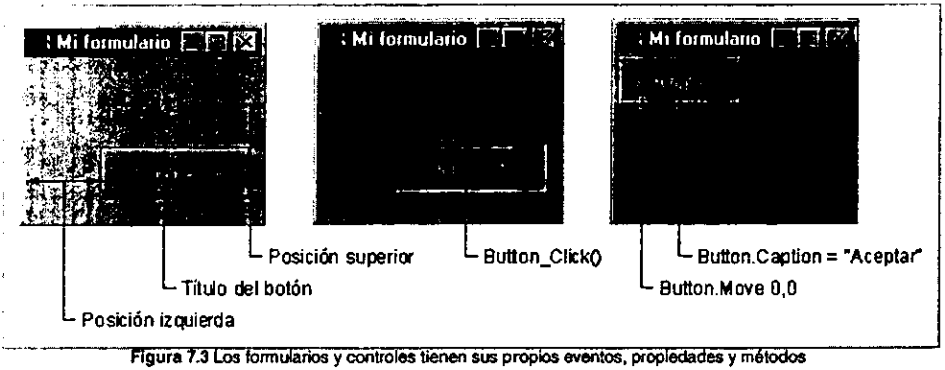


Figura 7.3 Los formularios y controles tienen sus propios eventos, propiedades y métodos

El primer paso para diseñar un formulario consiste en establecer sus propiedades. Estas pueden establecerse en tiempo de diseño en la ventana Propiedades o en tiempo de ejecución, escribiendo código. En *tiempo de diseño*, que es cualquier momento mientras se está desarrollando una aplicación en el entorno de Visual Basic, se trabaja con formularios y controles, se establecen propiedades y se escribe código para los eventos. *Tiempo de ejecución* es cualquier momento mientras se ejecuta realmente la aplicación y se interactúa con ella como lo haría un usuario.

Establecimiento de propiedades de un formulario

Muchas propiedades de un formulario afectan a su apariencia física. La propiedad **Caption** determina el texto que muestra la barra de título del formulario y la propiedad **Icon** establece el icono que aparece cuando se minimiza un formulario. Las propiedades **MaxButton** y **MinButton** determinan si el formulario se puede maximizar o minimizar. Cambiando la propiedad **BorderStyle** puede controlar el comportamiento de cambio de tamaño del formulario.

Las propiedades **Height** y **Width** determinan el tamaño inicial de un formulario, mientras que las propiedades **Left** y **Top** determinan la ubicación del formulario en relación con la esquina superior izquierda de la pantalla. Con la propiedad **WindowState** puede establecerse si el formulario se inicia en estado maximizado, minimizado o normal.

La propiedad **Name** establece el nombre con el que hará referencia al formulario en el código. De forma predeterminada, cuando se agrega un formulario por primera vez a un proyecto, su nombre es Form1, Form2, etc. Es conveniente establecer la propiedad **Name** a algo más significativo, como "frmEntrada" para un formulario de entrada de pedidos.

7.4 Estructura de programación en Visual Basic

Tras crear la interfaz de la aplicación mediante formularios y controles, se hace necesario escribir el código que defina el comportamiento de la aplicación. Como cualquier lenguaje moderno de programación, Visual Basic acepta ciertas construcciones de programación y elementos de lenguaje comunes.

Una aplicación no es más que un conjunto de instrucciones para que el equipo realice una o varias tareas. La estructura de una aplicación es la forma en que se organizan las instrucciones; es decir, dónde se almacenan las instrucciones y el orden en que se ejecutan.

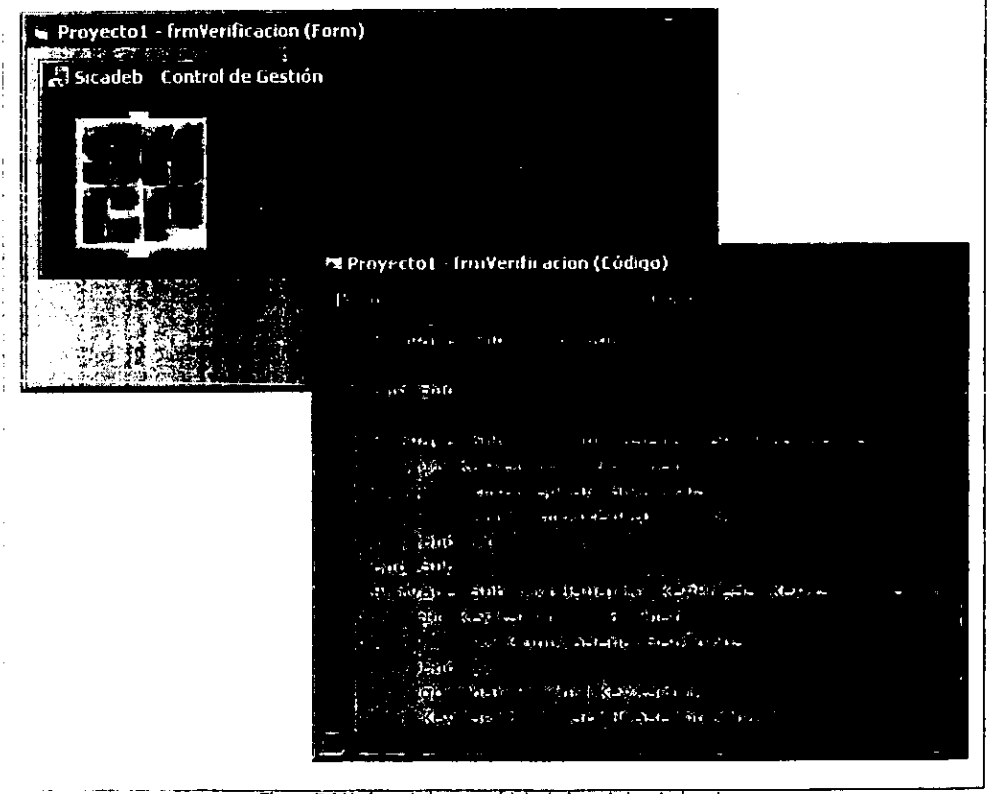


Figura 7.4 Un formulario y su módulo de formulario relacionado

Las aplicaciones sencillas tienen una estructura sencilla; la organización no es muy importante cuando sólo se tiene una línea de código. A medida que las aplicaciones se van haciendo más complejas, resulta obvia la necesidad de organizar o estructurar. Además de controlar la ejecución de una aplicación, la estructura es importante para el programador: ¿le resulta sencillo encontrar una instrucción determinada en la aplicación?

Puesto que una aplicación de Visual Basic se basa en objetos, la estructura de su código se aproxima mucho a su representación física en pantalla. Por definición, los objetos contienen datos y código. El formulario que ve en pantalla es una representación de las propiedades que definen su apariencia y su comportamiento intrínseco. Por cada formulario de una aplicación hay un *módulo de formulario* relacionado (con la extensión de nombre de archivo .frm) que contiene su código.

Cada módulo de formulario contiene *procedimientos de evento* (secciones de código donde se colocan las instrucciones que se ejecutarán como respuesta a eventos específicos). Los formularios pueden contener controles. Por cada control de un formulario, existe el correspondiente conjunto de procedimientos de evento en el módulo de formulario. Además de procedimientos de evento, los módulos de formulario pueden contener procedimientos generales que se ejecutan como respuesta a una llamada desde cualquier procedimiento de evento.

El código que no esté relacionado con un control o un formulario específico se puede colocar en un tipo diferente de módulo, un *módulo estándar* (.bas). Se deben colocar en un módulo estándar los procedimientos que se puedan utilizar como respuesta a eventos de diversos objetos, en lugar de duplicar el código en los procedimientos de evento de cada objeto.

Se utiliza un *módulo de clase* (.cls) para crear objetos a los que se puede llamar desde procedimientos de la aplicación. Mientras que un módulo estándar sólo contiene código, un módulo de clase contiene código y datos (puede considerarse como un control sin representación física).

Funcionamiento de una aplicación controlada por eventos

Un evento es una acción reconocida por un formulario o un control. Las aplicaciones controladas por eventos ejecutan código Basic como respuesta a un evento. Cada formulario y control de Visual Basic

tiene un conjunto de eventos predefinidos. Si se produce uno de dichos eventos y el procedimiento de evento asociado tiene código, Visual Basic llama a ese código.

Aunque los objetos de Visual Basic reconocen automáticamente un conjunto predefinido de eventos, el programador decide cuándo y cómo se responderá a un evento determinado. A cada evento le corresponde una sección de código (un procedimiento de evento). Cuando se desea que un control responda a un evento, se escribe código en el procedimiento de ese evento.

Los tipos de eventos reconocidos por un objeto varían, pero muchos tipos son comunes a la mayoría de los controles. Por ejemplo, la mayoría de los objetos reconocen el evento **Click**: si un usuario hace clic en un formulario, se ejecuta el código del procedimiento de evento **Click** del formulario; si un usuario hace clic en un botón de comando, se ejecuta el código del procedimiento de evento **Click** del botón. El código en cada caso será diferente.

He aquí una secuencia típica de eventos en una aplicación controlada por eventos:

1. Se inicia la aplicación y se carga y muestra un formulario.
2. El formulario (o un control del formulario) recibe un evento. El evento puede estar causado por el usuario (por ejemplo, por la pulsación de una tecla), por el sistema (por ejemplo, un evento de cronómetro) o, de forma indirecta, por el código (por ejemplo, un evento **Load** cuando el código carga un formulario).
3. Si hay código en el procedimiento de evento correspondiente, se ejecuta.
4. La aplicación espera al evento siguiente.

Muchos eventos se producen junto con otros eventos. Por ejemplo, cuando se produce el evento **DblClick**, se producen también los eventos **MouseDown**, **MouseUp** y **Click**.

7.4.1 Módulo de código

El código en Visual Basic se almacena en módulos. Hay tres tipos de módulos: de formulario, estándar y de clase.

Las aplicaciones sencillas pueden consistir de un único formulario donde todo el código de la aplicación reside en ese módulo de formulario. A medida que la aplicación vaya creciendo y siendo más sofisticada, se agregarán formularios adicionales. A veces se tendrá código común que habrá de ejecutarse en varios formularios, por lo que se creará un módulo independiente que contenga un procedimiento que ejecuta el código común. Este módulo independiente debe ser un módulo estándar.

Cada módulo estándar, de clase y de formulario puede contener lo siguiente:

- Declaraciones. Pueden colocarse declaraciones de constantes, tipos, variables y procedimientos de bibliotecas de vínculos dinámicos (DLL) al nivel de módulo de formulario, de clase o estándar.
- Procedimientos. Un procedimiento **Sub**, **Function** o **Property** contiene partes de código que se pueden ejecutar como una unidad. Se describen en el punto 7.4.2 de este mismo capítulo.

Módulos de formulario

Los módulos de formulario (extensión de nombre de archivo .frm) son la base de la mayoría de las aplicaciones de Visual Basic. Pueden contener procedimientos que controlen eventos, procedimientos generales y declaraciones a nivel de formulario de variables, constantes, tipos y procedimientos externos. El código que se escribe en un módulo de formulario es específico de la aplicación a la que pertenece el formulario y puede hacer referencia a otros formularios u objetos de la aplicación.

Módulos estándar

Los módulos estándar (extensión de nombre de archivo .bas) son contenedores de los procedimientos y declaraciones a los que tienen acceso otros módulos de la aplicación. Pueden contener declaraciones globales (disponibles para toda la aplicación) o a nivel de módulo de variables, constantes, tipos, procedimientos externos y procedimientos globales. El código que se escribe en un módulo estándar no está ligado necesariamente a una aplicación determinada; si se tiene cuidado de no hacer referencia a

controles o formularios por su nombre, es posible reutilizar un módulo estándar en distintas aplicaciones.

Módulos de clase

Los módulos de clase (extensión de nombre de archivo .cls) son la base de la programación orientada a objetos en Visual Basic. Es posible escribir código en módulos de clase para crear nuevos objetos. Estos objetos nuevos pueden incluir propiedades y métodos personalizados. En realidad, los formularios sólo son módulos de clase que pueden tener controles y que pueden mostrar ventanas de formulario.

7.4.2 Consideraciones de programación

Variables, constantes y tipos de datos

A menudo es necesario almacenar valores temporalmente cuando se estén realizando cálculos en Visual Basic. Por ejemplo, puede que se desee calcular diversos valores, compararlos y realizar distintas operaciones con ellos dependiendo de los resultados de la comparación. En tal caso será necesario conservar los valores para su comparación, pero no es necesario almacenarlos en una propiedad.

Visual Basic, como la mayoría de los lenguajes de programación, utiliza *variables* para almacenar valores. Las variables tienen un nombre (la palabra que utiliza para referirse al valor que contiene la variable) y un *tipo de dato* (que determina la clase de datos que puede almacenar la variable). Se pueden utilizar *matrices* para almacenar colecciones indexadas de variables relacionadas.

Las *constantes* también almacenan valores pero, como su nombre indica, estos valores permanecen constantes durante la ejecución de la aplicación. La utilización de constantes puede hacer más legible el código ya que proporciona nombres significativos en vez de números.

Los *tipos de datos* controlan el almacenamiento interno de datos en Visual Basic. De forma predeterminada, Visual Basic utiliza el tipo de dato **Variant**. Hay otros tipos de datos disponibles que permiten optimizar el código en cuanto a velocidad y tamaño cuando no sea necesaria la flexibilidad que proporciona el tipo **Variant**.

Procedimientos

Es posible simplificar las tareas de programación dividiendo los programas en componentes lógicos más pequeños. Estos componentes, llamados *procedimientos*, pueden convertirse en bloques básicos que permiten mejorar y ampliar Visual Basic.

Los procedimientos resultan muy útiles para condensar las tareas repetitivas o compartidas, como cálculos utilizados frecuentemente, manipulación de texto, controles y operaciones con bases de datos.

Hay dos ventajas principales cuando se programa con procedimientos:

- Los procedimientos permiten dividir los programas en unidades lógicas discretas, cada una de las cuales se puede depurar más fácilmente que un programa entero sin procedimientos.
- Los procedimientos que se utilizan en un programa pueden actuar como bloques de construcción de otros programas, normalmente con pocas o ninguna modificación.

En Visual Basic se utilizan varios tipos de procedimientos:

- Procedimientos **Sub** que no devuelven un valor.
- Procedimientos **Function** que devuelven un valor.
- Procedimientos **Property** que pueden devolver y asignar valores, así como establecer referencias a objetos.

Estructuras de control

Las estructuras de control permiten manipular el flujo de ejecución del programa. Si no se controla mediante instrucciones de control de flujo, la lógica del programa fluirá por las instrucciones de izquierda a derecha y de arriba a abajo. Aunque se pueden escribir algunos programas sencillos con un flujo unidireccional y aunque se puede controlar parte del flujo mediante operadores para regular la precedencia de las operaciones, la mayor parte del poder y utilidad de un lenguaje de programación deriva de su capacidad de cambiar el orden de las instrucciones mediante estructuras y bucles:

- *Estructuras de decisión:* Utilizadas para bifurcar, los procedimientos de Visual Basic pueden probar condiciones y, dependiendo de los resultados de la prueba, realizar diferentes operaciones. Entre las estructuras de decisión que acepta Visual Basic se incluyen las siguientes:

  If...Then
  If...Then...Else
  Select Case

- *Estructuras de bucle:* Utilizadas para repetir procesos, permiten ejecutar una o más líneas de código repetidamente. Las estructuras de bucle que acepta Visual Basic son:

  Do...Loop
  For...Next
  For Each...Next

7.4.3 Trabajo con objetos

Cuando se crea una aplicación en Visual Basic se trabaja con objetos. Es posible utilizar los objetos que proporciona Visual Basic como controles, formularios y objetos de acceso a datos, también controlar objetos de otras aplicaciones desde una aplicación de Visual Basic e incluso crear objetos propios y definir propiedades y métodos adicionales para ellos.

Un objeto es una combinación de código y datos que se puede tratar como una unidad. Un objeto puede ser parte de una aplicación, como un control o un formulario. También puede ser un objeto una aplicación entera. La tabla siguiente describe ejemplos de tipos de objetos que puede utilizar Visual Basic.

Ejemplo	Descripción
Botón de comando	Son objetos los controles de un formulario, como botones de comandos y marcos.
Formulario	Cada formulario de un proyecto de Visual Basic es un objeto distinto.
Base de datos	Las bases de datos son objetos y contienen otros objetos, como campos e índices.
Gráfico	Un gráfico de Microsoft Excel es un objeto.

Cada objeto de Visual Basic se define mediante una *clase*. La relación entre un objeto y su clase puede verse como el molde de las galletas y las galletas. El molde es la clase. Define las características de cada galleta, como por ejemplo el tamaño y la forma. Se utiliza la clase para crear objetos. Los objetos son las galletas.

Mediante dos ejemplos se puede aclarar la relación que existe entre las clases y los objetos en Visual Basic:

- Los controles del cuadro de herramientas de Visual Basic representan clases. El objeto conocido como control no existe hasta que se dibuja en un formulario. Cuando crea un control, se está creando una copia o *instancia* de la clase del control. Esta instancia de la clase es el objeto al que hará referencia en la aplicación.
- El formulario en el que se trabaja en tiempo de diseño es una clase. En tiempo de ejecución, Visual Basic crea una instancia de esa clase de formulario.

La ventana Propiedades muestra la clase y la propiedad **Name** de los objetos de una aplicación de Visual Basic, como se muestra en la figura 7.5.

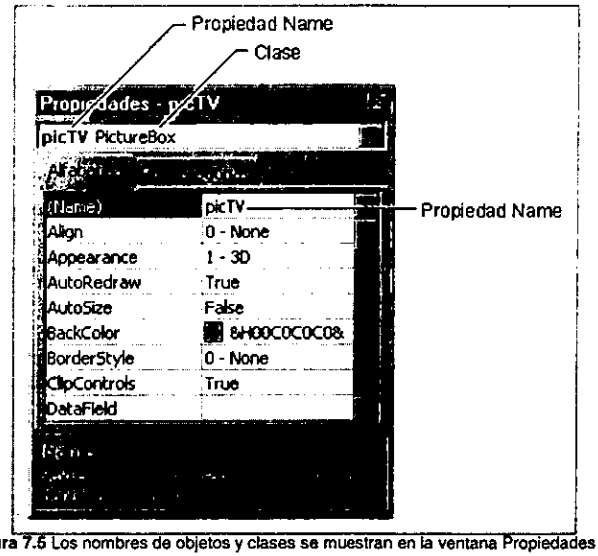


Figura 7.5 Los nombres de objetos y clases se muestran en la ventana Propiedades

Todos los objetos creados son copias idénticas de sus clases, una vez que existen como objetos individuales, es posible modificar sus propiedades. Por ejemplo, si se dibujasen tres botones de comando en un formulario, cada botón de comando es una instancia de la clase **CommandButton**. Cada objeto comparte un conjunto de características y capacidades comunes (propiedades, métodos y eventos), definidos por la clase. Sin embargo, cada uno tiene su propio nombre, se puede activar y desactivar por separado, colocarse en una ubicación distinta del formulario, etc.

Los objetos de Visual Basic aceptan propiedades, métodos y eventos. En Visual Basic, los datos de un objeto (configuración o atributos) se llaman propiedades, mientras que los diversos procedimientos que pueden operar sobre el objeto se conocen como sus métodos. Un evento es una acción reconocida por un objeto, como hacer clic con el *mouse* o presionar una tecla, y es posible escribir código que responda a ese evento.

Visual Basic permite cambiar las características de un objeto a través de sus propiedades. Para el caso de un radio por ejemplo, se tiene que una propiedad de la radio es su volumen. En Visual Basic, se diría que una radio tiene la propiedad "Volumen" que se puede ajustar modificando su valor. Suponiendo que establece el volumen de la radio de 0 a 10. Si se pudiera controlar una radio en Visual Basic, podría

escribirse el código en un procedimiento que modificara el valor de la propiedad "Volumen" de 3 a 5 para hacer que la radio suene más alto:

```
Radio.Volumen = 5
```

Además de propiedades, los objetos tienen métodos. Los métodos son parte de los objetos del mismo modo que las propiedades. Generalmente, los métodos son acciones que se desean realizar, mientras que las propiedades son los atributos que pueden establecerse o recuperarse. Por ejemplo, marcar un número de teléfono para hacer una llamada. Se podría decir que el teléfono tiene un método "Marcar" y podría utilizarse esta sintaxis para marcar el número de siete cifras 555 1111:

```
Teléfono.Marcar 5551111
```

Los objetos también tienen eventos. Los eventos se desencadenan cuando cambia algún aspecto del objeto. Por ejemplo, una radio podría tener el evento "CambiarVolumen" y el teléfono podría tener el evento "Sonar".

Las propiedades individuales varían ya que puede establecer u obtener sus valores. Se pueden establecer algunas propiedades en tiempo de diseño. Puede utilizarse la ventana Propiedades para establecer el valor de dichas propiedades sin tener que escribir código. Algunas propiedades no están disponibles en tiempo de diseño, por lo que se necesitará escribir código para establecer esas propiedades en tiempo de ejecución.

7.5 Acceso remoto a datos

El acceso a los datos es crucial en las aplicaciones cliente-servidor y para ello Visual Basic cuenta con opciones como lo son Objetos de Acceso a Datos y Objetos de Datos Remotos, el control RemoteData, la interfaz de programación de aplicaciones[16] ODBC y la interfaz VBSQL de Microsoft SQL Server.

La interfaz de programación de aplicaciones de ODBC utilizada en el presente desarrollo define un modelo de programación independiente de la base de datos que proporciona una única interfaz API que puede utilizarse para tener acceso a cualquier base de datos o servidor de bases de datos que disponga de un controlador de ODBC, SQL Server 6.5 de Microsoft para el presente caso de estudio. Mediante la API de ODBC se desarrollan aplicaciones con el rendimiento y la flexibilidad de los modelos de programación de código nativo.

Antes, la API de ODBC proporcionaba una interfaz para los programadores de aplicaciones clientes de bases de datos que necesitaban una plataforma rápida, pequeña y relativamente estable con la que conectarse a sistemas cliente-servidor. La API de ODBC está diseñada para ejecutarse en distintas plataformas, incluyendo Windows de 16 bits, Windows 98 de 32 bits, Windows NT y otras.

Parámetros de conexión

Además de los controladores de entorno y conexión proporcionados por Visual Basic, el código debe proporcionar los argumentos siguientes a la hora de establecer una conexión:

- Un nombre de origen de datos registrado correspondiente a una entrada DSN del archivo ODBC.ini o del registro del sistema.
- Un identificador de usuario (opcional), que es el Id. de inicio de sesión o el nombre de cuenta utilizado para tener acceso al origen de datos.
- Una contraseña válida (opcional) que corresponda al Id. de usuario.
- Cualquier parámetro opcional que proporcione información al controlador, por ejemplo si debe seleccionar una base de datos específica.

[16] API por sus siglas en inglés: Application Program interfase

Creación de un nombre de origen de datos ODBC

Cuando se abre una conexión con cualquier base de datos o servidor de bases de datos remoto mediante la API de ODBC, debe hacerse referencia al nombre de un origen de datos en particular. Aunque pueden crearse los DSN en el código mediante la función de configuración de ODBC ConfigDSN, la forma más fácil de crear o modificar un DSN es utilizar el programa Administrador de ODBC del Panel de control, que se instala al seleccionar la opción ODBC durante la instalación personalizada de Visual Basic.

La información del DSN se almacena en el archivo ODBC.ini para los sistemas operativos de 16 bits y en el registro del sistema para los sistemas operativos de 32 bits.

Envío de una instrucción SELECT de SQL

El código siguiente envía una instrucción SELECT de SQL asignando un controlador de instrucción y utilizando la función SQLExecDirect para enviar la instrucción SELECT al servidor. Si falla la consulta, se describe el error en un cuadro de mensajes.

```
Private Sub cmdBuscar_Click()
   Dim dbSHCP As Database
   Dim SQLQuery$

   Set dbSHCP = OpenDatabase("SQLServer", False, False, "ODBC; UID=" & sUsuario & "; PWD="
   _sContraseña)

   SQLQuery$ = "SELECT * FROM beneficiario "

'Efectúa consulta
   datConsulta.Options = dbSQLPassThrough
   datConsulta.RecordSource = SQLQuery$
   datConsulta.Refresh

End Sub
```

El ejemplo anterior abre un origen de datos previamente definido en el administrador ODBC de 32 bits del sistema operativo con nombre de origen de datos (DSN) **SQLServer** y al cual se accesa mediante el nombre de usuario contenido en la variable global **sUsuario** y la contraseña de la variable **sContraseña**. Una vez abierto el origen de datos la consulta SQL dentro de la variable **SQLQuery** se ejecuta sobre el objeto **datConsulta**, previamente ligado mediante sus propiedades en tiempo de diseño a la base de datos definida **dbSHCP**.

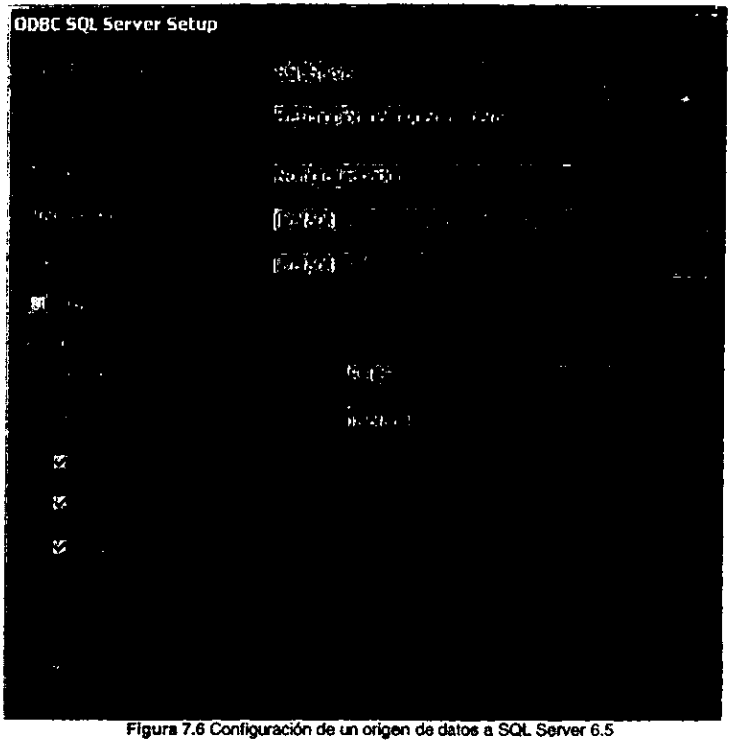


Figura 7.6 Configuración de un origen de datos a SQL Server 6.5

7.6 Documentación

Estructura de programación del Sistema de Control Administrativo de Destino de Bienes

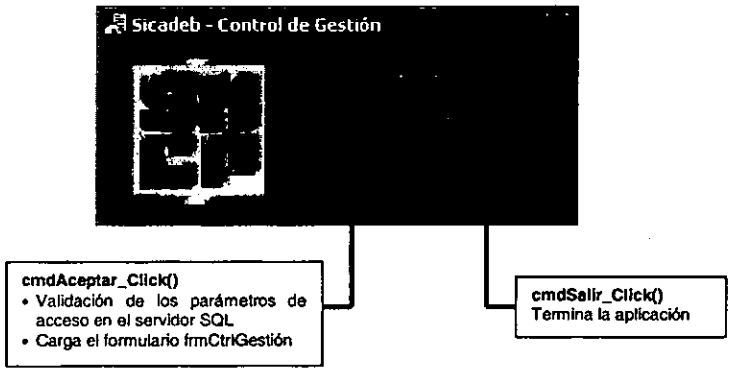
Módulo de Control de Gestión

frmVerificaciónCtrlGestión

En este primer formulario del módulo de control de gestión se dan de alta los parámetros de conexión con el servidor de base de datos, esta conexión se establece en el momento que el usuario hace clic sobre el botón *Aceptar*:

```
Set dbSHCP = OpenDatabase("SQLServer", False, False, "ODBC; UID=" & sUsuario & ";
PWD=" & sContraseña)
```

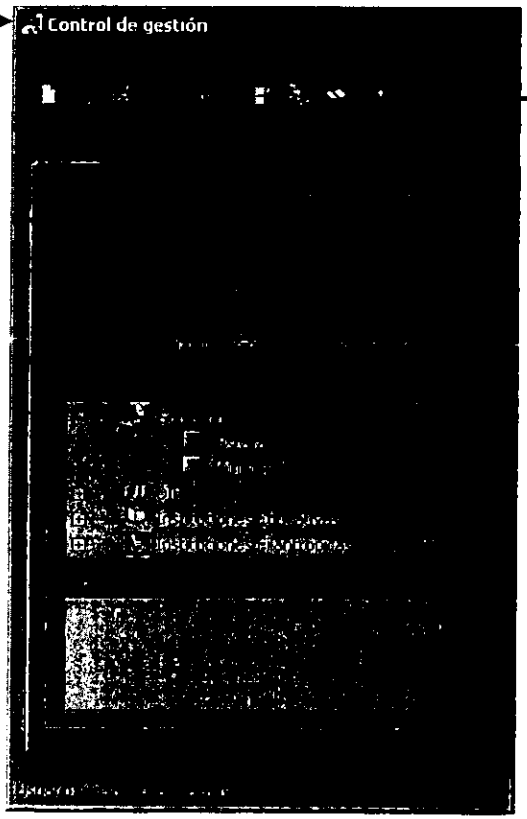
La instrucción anterior abre una base de datos a través de una conexión ODBC de nombre *SQLServer*, el nombre de usuario y la contraseña que se capturan en esta pantalla y se almacenan en las variables *sUsuario* y *sContraseña* indican al servidor de base de datos los privilegios de acceso de la presente sesión, una vez que el servidor de base de datos valida el usuario y contraseña del usuario se asocia la base de datos *SHCP* al objeto de Visual Basic *dbSHCP*, y es sobre este objeto que se llevaran a cabo las afectaciones a la base de datos por parte de esta aplicación. Por último se procede al formulario *frmCtrlGestión*.



frmCtrlGestión

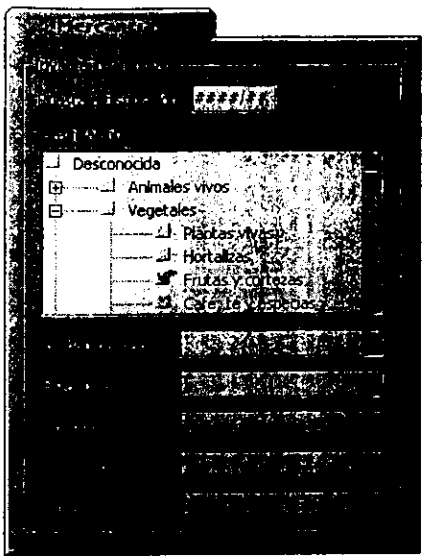
Este formulario cumple con la doble función de servir como plataforma de captura de nuevos registros y desplegar los resultados de una búsqueda para su edición. En primera instancia esta predispuesto para la captura de nuevos registros, clasificados en tres grupos: *Documentos* donde se capturan las solicitudes por parte de los beneficiarios de la Dirección de Destino de Bienes y *Mercancías*, que junto con *Vehículos* permite capturar los datos de los oficios de puesta a disposición y en particular las mercancías que en estos se describen.

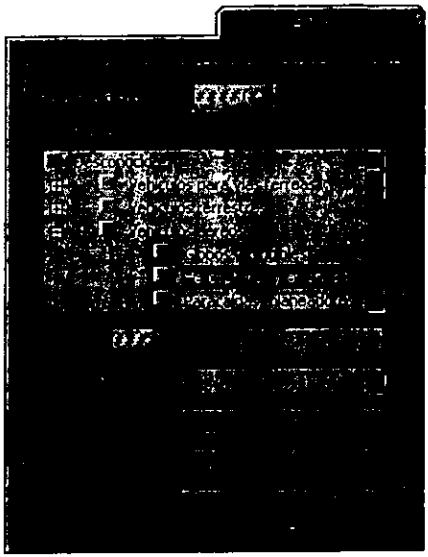
Form_Load()
- Carga los catálogos correspondientes a: *beneficiarios, aduanas, clasificación, unidad de medida* así como los datos particulares del usuario con el que se acceso al servidor para su despliegue en las opciones que así lo requieran
- Verifica los privilegios del usuario para habilitar o deshabilitar acceso al resto de los módulos



tlbCtrlGestión_ButtonClick(ByVal Button As ComctlLib.Button)

	Restaura los valores predeterminados en los campos de captura
	Valida los datos de la captura y en caso de ser correctos envía al servidor una instrucción SQL del tipo INSERT para nuevos registros ó UPDATE para actualizaciones
	Imprime un resumen de lo capturado
	Elimina el registro seleccionado enviando al servidor de base de datos una instrucción SQL del tipo DELETE
	Cambia al módulo de consultas (frmConsultasCtrlGestión) que permite además seleccionar un registro previamente capturado para su edición en esta pantalla
	Cambia al Módulo de Inventarios
	Cambia al Módulo de Destinos
	Cambia al Módulo de Control de Operaciones
	Sale de la aplicación





Para llevar a cabo las actualizaciones al objeto *dbSHCP*, asociado a la base de datos del sistema Visual Basic utiliza la siguiente instrucción:

```
dbSHCP.Execute SQLQuery$, dbSQLPassThrough
```

Donde *SQLQuery$* corresponde a una cadena que contiene la instrucción SQL a efectuar sobre la base de datos, el modificador *dbSQLPassThrough* implica que la validación de la sintaxis de la instrucción y la administración de recursos de memoria requeridos por la instrucción *SQLQuery$* correrán a cargo del servidor de base de datos. Todo acceso a la base de datos del sistema sigue esta premisa, que es en la que se basa la arquitectura cliente/servidor, la terminal solo se encarga de presentar la interfaz gráfica al usuario, validar los campos de captura, trasmitir la instrucción SQL al servidor de base de datos y recibir de este los resultados de la misma y en su caso desplegarlos.

Para el caso particular del formulario *frmCtrlGestión*, y cuando su función es de captura, la instrucción SQL asociada es del tipo INSERT, al tratarse de una actualización esta se lleva a cabo con una instrucción SQL del tipo UPDATE, en ambos casos las tablas afectadas son *bienes* y *solicitudes*.

frmConsultasCtrlGestión

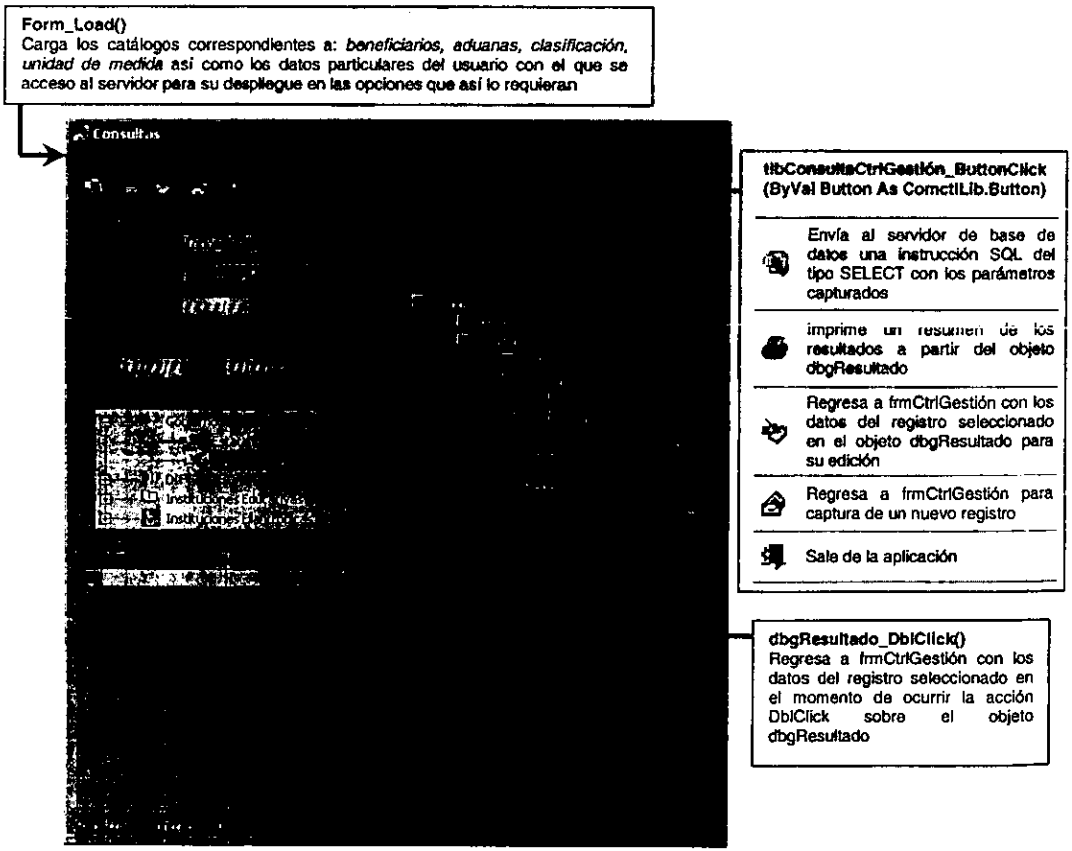
El propósito de este formulario es el de servir de interfaz gráfica para la generación de instrucciones SQL del tipo SELECT que permitan recuperar registros de la base de datos que cumplan ciertos criterios. La instrucción SQL enviada al servidor para el caso de consulta a solicitudes será de este tipo:

SELECT * FROM solicitudes s
WHERE *parámetros*

Para el caso de consulta de oficios de puesta a disposición:

SELECT * FROM bienes b, pudi p
WHERE b.no_pudi = p.no_pudi
AND *parámetros*

En ambos casos el estado de las opciones de búsqueda en el formulario determinaran las condiciones de la instrucción: *parámetros*.



Módulo de Inventario

frmVerificaciónInventario

En este primer formulario del módulo de inventario se dan de alta los parámetros de conexión con el servidor de base de datos, esta conexión se establece en el momento que el usuario hace clic sobre el botón *Aceptar*:

```
Set dbSHCP = OpenDatabase("SQLServer", False, False, "ODBC; UID=" & sUsuario & ";
PWD=" & sContraseña)
```

La instrucción anterior abre una base de datos a través de una conexión ODBC de nombre *SQLServer*, el nombre de usuario y la contraseña que se capturan en esta pantalla y se almacenan en las variables *sUsuario* y *sContraseña* indican al servidor de base de datos los privilegios de acceso de la presente sesión, una vez que el servidor de base de datos valida el usuario y contraseña del usuario se asocia la base de datos *SHCP* al objeto de Visual Basic *dbSHCP*, y es sobre este objeto que se llevaran a cabo las afectaciones a la base de datos por parte de esta aplicación. Por último se procede al formulario *frmConsultasInventario*.

cmdAceptar_Click()
- Validación de los parámetros de acceso en el servidor SQL.
- Carga el formulario frmConsultasInventario

cmdSalir_Click()
Termina la aplicación

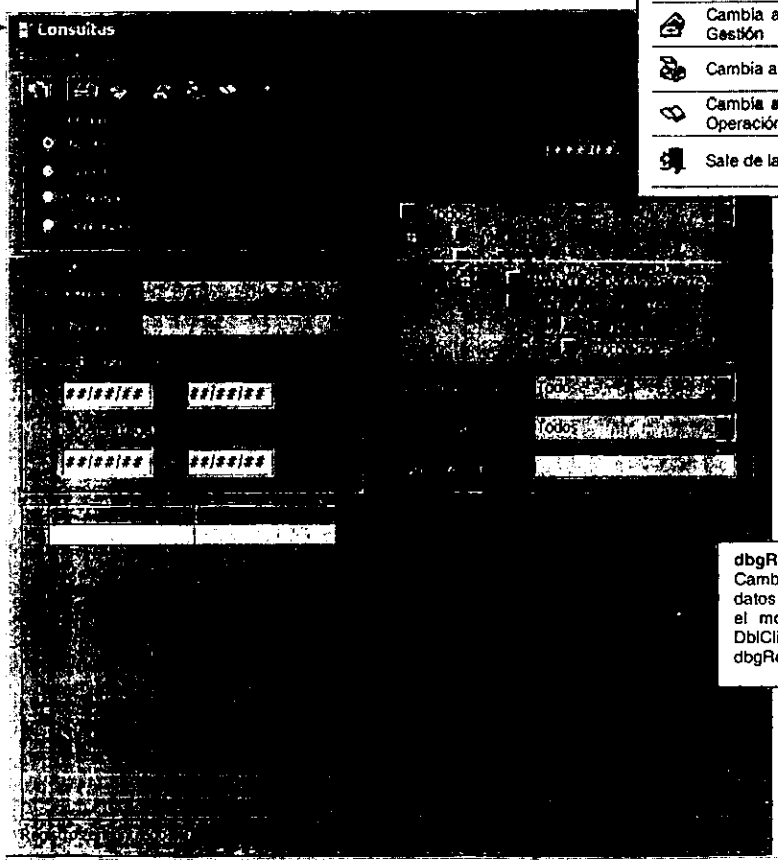
frmConsultasInventario

La validación física de los datos capturados por el área de Control de Gestión requiere de una busqueda avanzada a cargo de este formulario. La instrucción SQL enviada al servidor depende de la opción seleccionada en el cuadro *Inventario*, en el extremo superior izquierdo del formulario, los parámetros adicionales de la instrucción SQL varían en virtud al estado del resto de las opciones del formulario.

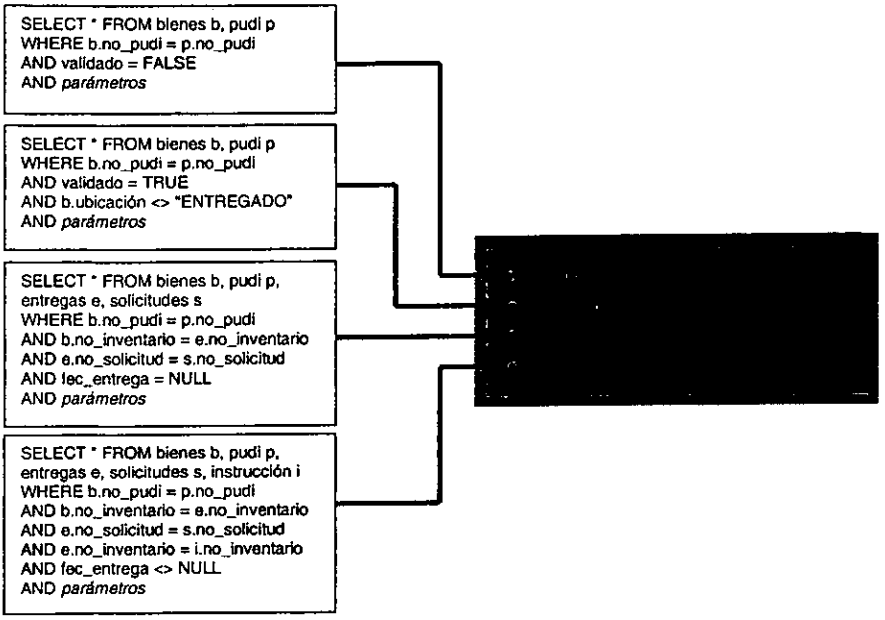
tlbConsultaInventario_ButtonClick (ByVal Button As ComctlLib.Button)

- Envía al servidor de base de datos una instrucción SQL del tipo SELECT con los parámetros capturados
- Imprime un resumen de los resultados a partir del objeto dbgResultado
- Cambia a frmInventario con los datos del registro seleccionado en el objeto dbgResultado para su edición
- Cambia al Módulo de Control de Gestión
- Cambia al Módulo de Destinos
- Cambia al Módulo de Control de Operación
- Sale de la aplicación

Form_Load()
Carga los catálogos correspondientes a: *clasificación, unidad de medida* y *estado físico* así como los datos particulares del usuario con el que se acceso al servidor para su despliegue en las opciones que así lo requieran



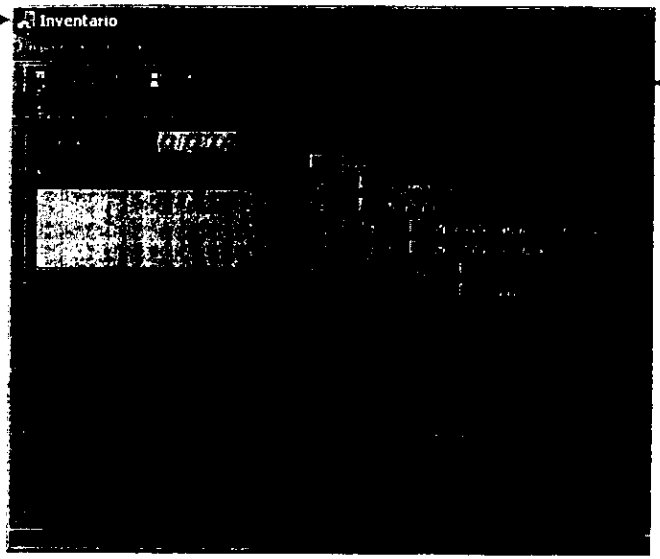
dbgResultado_DblClick()
Cambia a frmInventario con los datos del registro seleccionado en el momento de ocurrir la acción DblClick sobre el objeto dbgResultado


SELECT * FROM bienes b, pudi p
WHERE b.no_pudi = p.no_pudi
AND validado = FALSE
AND parámetros
SELECT * FROM bienes b, pudi p
WHERE b.no_pudi = p.no_pudi
AND validado = TRUE
AND b.ubicación <> "ENTREGADO"
AND parámetros
SELECT * FROM bienes b, pudi p,
entregas e, solicitudes s
WHERE b.no_pudi = p.no_pudi
AND b.no_inventario = e.no_inventario
AND e.no_solicitud = s.no_solicitud
AND fec_entrega = NULL
AND parámetros
SELECT * FROM bienes b, pudi p,
entregas e, solicitudes s, instrucción i
WHERE b.no_pudi = p.no_pudi
AND b.no_inventario = e.no_inventario
AND e.no_solicitud = s.no_solicitud
AND e.no_inventario = i.no_inventario
AND fec_entrega <> NULL
AND parámetros

frmInventario

Este formulario permite actualizar la información respecto a los bienes en inventario para adecuarla a su realidad física, por lo tanto, las afectaciones que hará a la base de datos serán del tipo UPDATE, sobre la tabla de *bienes* para el caso descrito y además sobre las tablas *entregas* e *instrucción* para reflejar el apartado ó desalojo de mercancías.

Form_Load()
Carga los catálogos correspondientes a: *clasificación, unidad de medida* y *estado físico* así como los datos particulares del usuario con el que se acceso al servidor para su despliegue en las opciones que así lo requieran



tlbInventario_ButtonClick
(ByVal Button As ComctlLib.Button)

- Envía al servidor de base de datos una instrucción SQL del tipo UPDATE con los parámetros capturados
- Imprime un resumen de los bienes descritos en pantalla
- Elimina el registro actual
- Regresa a frmConsultasInventario
- Sale de la aplicación

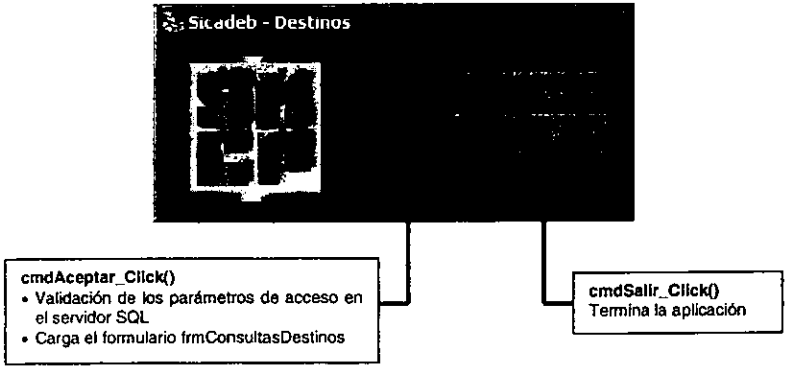
Módulo de Destinos

frmVerificaciónDestinos

En este primer formulario del módulo de destinos se dan de alta los parámetros de conexión con el servidor de base de datos, esta conexión se establece en el momento que el usuario hace clic sobre el botón *Aceptar:*

```
Set dbSHCP = OpenDatabase("SQLServer", False, False, "ODBC; UID=" & sUsuario & ";
PWD=" & sContraseña)
```

La instrucción anterior abre una base de datos a través de una conexión ODBC de nombre *SQLServer*, el nombre de usuario y la contraseña que se capturan en esta pantalla y se almacenan en las variables *sUsuario* y *sContraseña* indican al servidor de base de datos los privilegios de acceso de la presente sesión, una vez que el servidor de base de datos valida el usuario y contraseña del usuario se asocia la base de datos *SHCP* al objeto de Visual Basic *dbSHCP*, y es sobre este objeto que se llevaran a cabo las afectaciones a la base de datos por parte de esta aplicación. Por último se procede al formulario *frmConsultasDestinos.*



frmConsultasDestinos

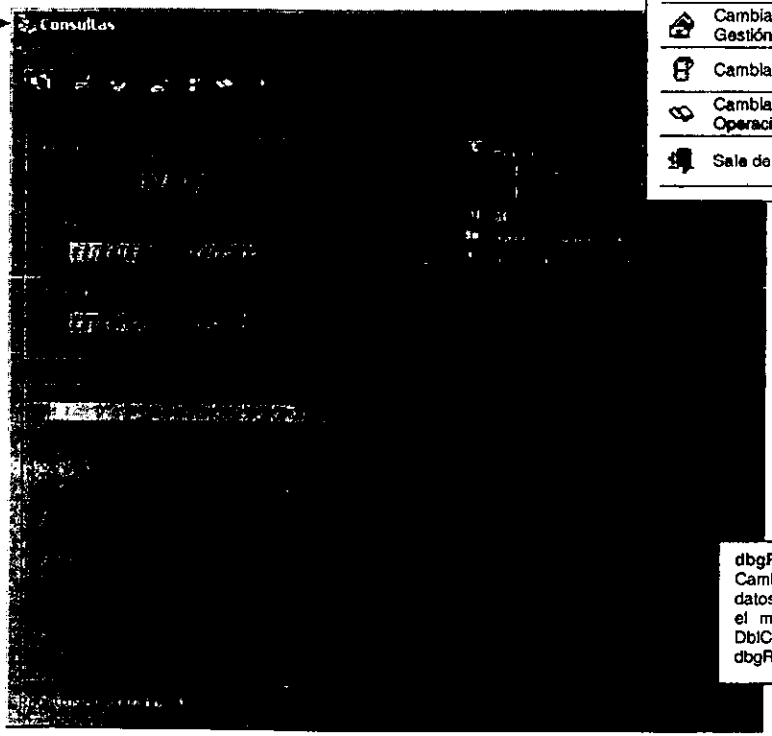
Caso similar al del módulo de inventario, para su operación el módulo de destinos requiere una consulta integral a la base de datos sobre la tabla *solicitudes*, operando bajo el mismo esquema, la instrucción SQL enviada al servidor de base de datos será del tipo:

SELECT * FROM solicitudes s
WHERE *parámetros*

tlbConsultaInventario_ButtonClick (ByVal Button As ComctlLib.Button)

- Envía al servidor de base de datos una instrucción SQL del tipo SELECT con los parámetros capturados
- Imprime un resumen de los resultados a partir del objeto dbgResultado
- Cambia a frmResolución con los datos del registro seleccionado en el objeto dbgResultado para su edición
- Cambia al Módulo de Control de Gestión
- Cambia al Módulo de Inventario
- Cambia al Módulo de Control de Operación
- Sale de la aplicación

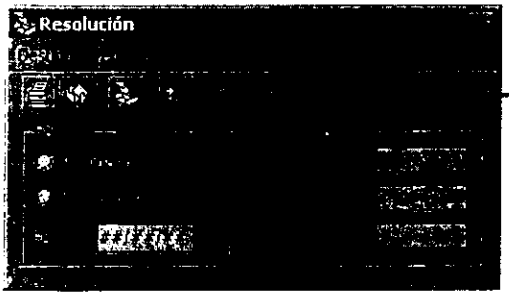
Form_Load()
Carga los catálogos correspondientes a: *beneficiario* y *clasificación* así como los datos particulares del usuario con el que se accesó al servidor para su despliegue en las opciones que así lo requieran



dbgResultado_DblClick()
Cambia a frmResolución con los datos del registro seleccionado en el momento de ocurrir la acción DblClick sobre el objeto dbgResultado

frmResolución

La función de esta pantalla es la de dar de alta los registros de las tablas *resolución*, *entregas* e *instrucción* que corresponden a la solicitud previamente seleccionada en el formulario *frmConsultasDestinos*, las instrucciones SQL que direccionará al servidor de base de datos serán del tipo INSERT. Como es de entenderse en caso de que la opción *Fallo* se encuentre en *Rechazado*, el proceso termina aquí con la captura del número de resolución únicamente, en caso contrario se procede al formulario *frmAsociaciar*.



tlbResolución_ButtonClick (ByVal Button As ComctlLib.Button)

Valida los datos registrados y en su caso envía al servidor una instrucción SQL del tipo INSERT con los datos capturados, imprime el oficio de resolución para la solicitud seleccionada con los datos registrados

Valida los datos registrados y en su caso envía al servidor una instrucción SQL del tipo INSERT con los datos capturados, finalmente cambia a frmAsociar

Regresa a frmConsultasDestinos

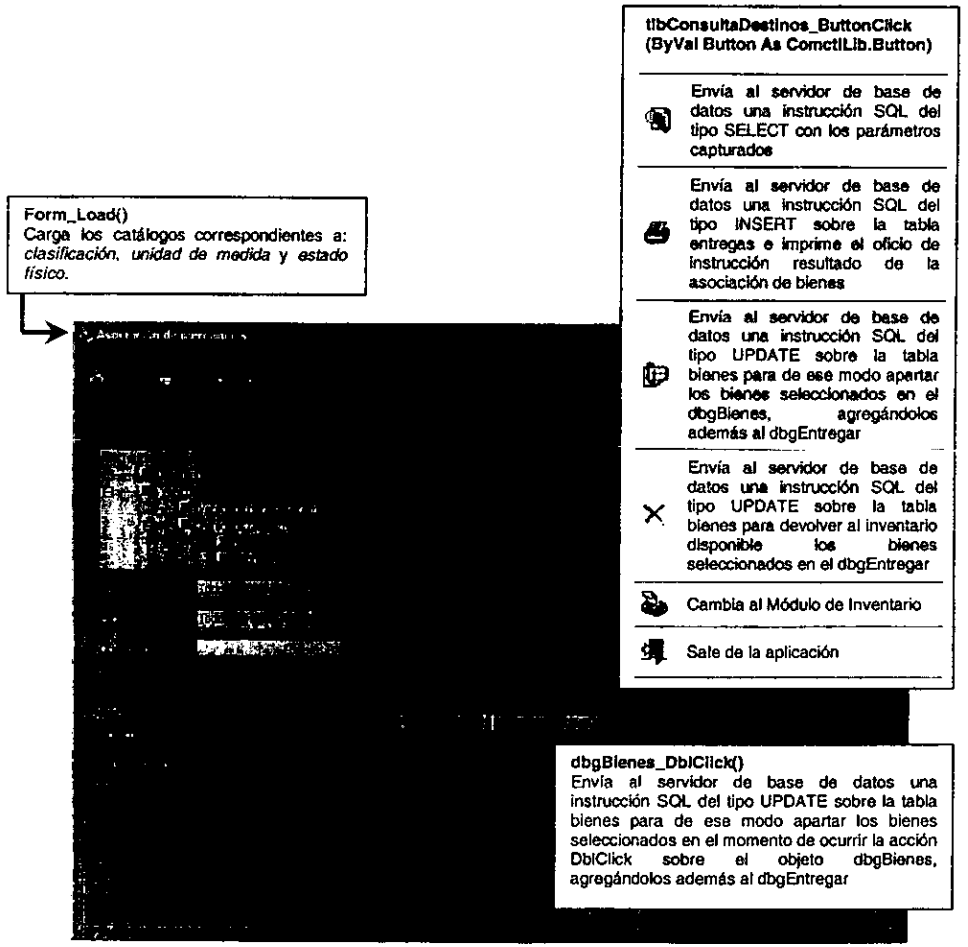
Sale de la aplicación

frmAsociar

Efectuar búsquedas en el inventario disponible de la tabla *bienes* en la base de datos y asociarla a la solicitud seleccionada es la doble tarea de este formulario. Para lograr su primer fin es necesaria una consulta del tipo SELECT como la siguiente:

```
SELECT * FROM bienes
WHERE validado = TRUE
AND parámetros
```

Por otro lado para asociar las mercancías seleccionadas con la solicitud activa es necesaria una instrucción SQL del tipo INSERT sobre la tabla *entregas* que haga referencia al campo *no_inventario* de la tabla *bienes* y al campo *no_solicitud* de la tabla *solicitudes*.



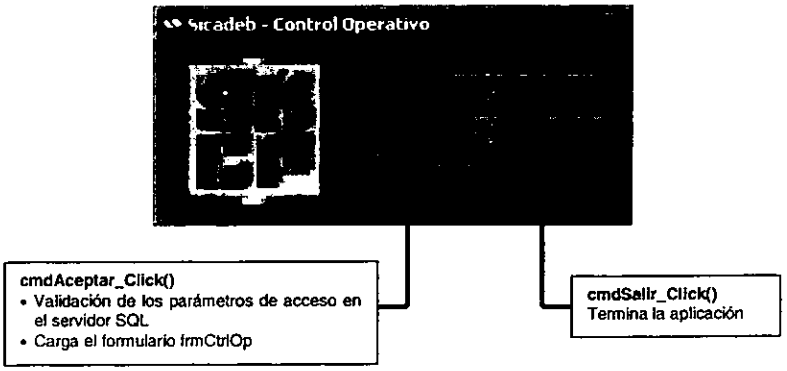
Módulo de Control Operativo

frmVerificaciónCtrlOp

En este primer formulario del módulo de control operativo se dan de alta los parámetros de conexión con el servidor de base de datos, esta conexión se establece en el momento que el usuario hace clic sobre el botón *Aceptar:*

```
Set dbSHCP = OpenDatabase("SQLServer", False, False, "ODBC; UID=" & sUsuario & ";
PWD=" & sContraseña)
```

La instrucción anterior abre una base de datos a través de una conexión ODBC de nombre *SQLServer*, el nombre de usuario y la contraseña que se capturan en esta pantalla y se almacenan en las variables *sUsuario* y *sContraseña* indican al servidor de base de datos los privilegios de acceso de la presente sesión, una vez que el servidor de base de datos valida el usuario y contraseña del usuario se asocia la base de datos *SHCP* al objeto de Visual Basic *dbSHCP*, y es sobre este objeto que se llevaran a cabo las afectaciones a la base de datos por parte de esta aplicación. Por último se procede al formulario *frmCtrlOp*.



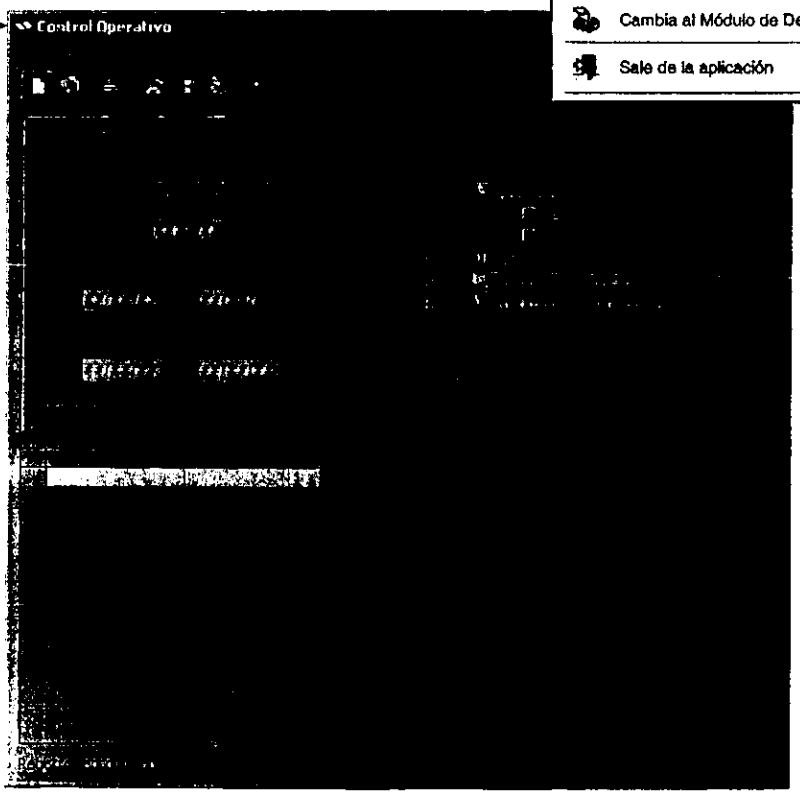
frmCtrlOp

La función del módulo de Control Operativo es la de generar reportes gerenciales que reflejen el estado de las operaciones de la Dirección de Destino de Bienes, para tal efecto se hace necesario un módulo de consulta general dispuesto en el formulario *frmCtrlOp* y que consta de tres opciones principales: *Solicitudes, Inventario* y *Entregas.*

tlbCtrlOp_ButtonClick (ByVal Button As ComctlLib.Button)

- Restaura los valores predeterminados en los campos de captura
- Envía al servidor de base de datos una instrucción SQL del tipo SELECT con los parámetros capturados
- Imprime un resumen de los resultados a partir del objeto dbgResultado
- Cambia al Módulo de Control de Gestión
- Cambia al Módulo de Inventario
- Cambia al Módulo de Destinos
- Sale de la aplicación

Form_Load()
Carga los catálogos correspondientes a: *clasificación, unidad de medida, estado físico, beneficiarios* y *aduanas*, así como los datos particulares del usuario con el que se acceso al servidor para habilitar acceso a los módulos que requiera.



Para el caso de consultas a solicitudes, la instrucción SQL será como sigue:

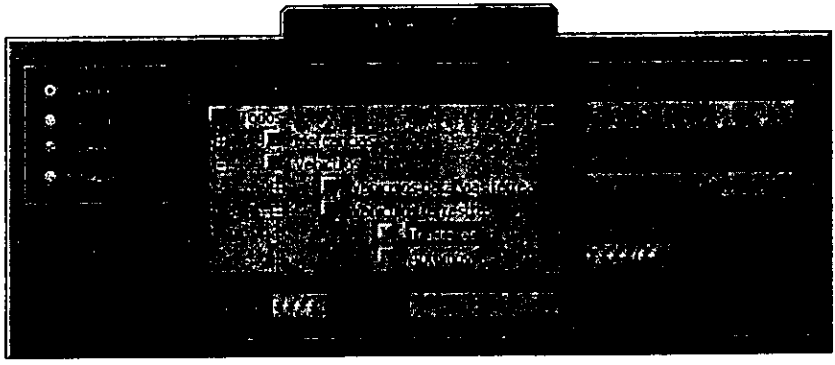
```
SELECT * FROM solicitudes s
WHERE parámetros
```

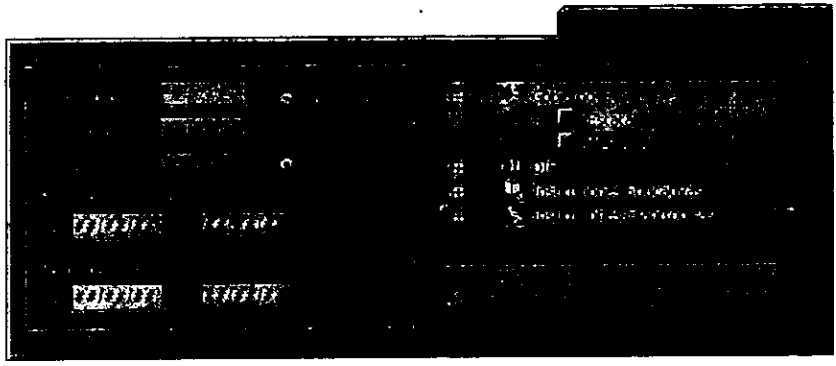
Para el caso de inventarios la consulta sigue el siguiente esquema:

```
SELECT * FROM bienes b, entregas e
WHERE b.no_inventario = e.no_inventario
AND parámetros
```

Y por último las consultas referentes a entregas son del siguiente tipo:

```
SELECT * FROM bienes b, entregas e, solicitudes s, instrucción i
WHERE b.no_inventario = e.no_inventario
AND e.no_solicitud = s.no_solicitud
AND e.no_instrucción = i.no_instrucción
AND parámetros
```





Pruebas y mantenimiento

El mantenimiento de un proceso es correctivo, de adaptación, perfectivo o preventivo. El mantenimiento se facilita si se anticipa la necesidad de un futuro mantenimiento en el momento de diseñar el proceso. El mantenimiento a un programa se considera por lo general casi como la reutilización del código, un caso particular de lo anterior es la adaptación de un producto a mercados internacionales.

8.1 El propósito de las pruebas

Muchos programadores ven a las pruebas de programación como una demostración de que sus programas funcionan correctamente. Sin embargo, la idea de demostrar lo correcto de los programas es en realidad lo contrario de lo que se busca en una etapa de pruebas. Un programa se pone a *prueba*, para demostrar la existencia de un error. Dado que la meta es el descubrir un error, solo podrá considerarse como exitosa aquella prueba que descubra errores. Una vez localizado, la *depuración* o corrección de errores es el proceso de determinar lo que causa el error y el llevar a cabo los cambios necesarios al sistema para eliminar el error.

La fase de pruebas del desarrollo se compone de un ciclo de vida propio, comenzando con pruebas individuales de cada módulo, seguidas de la integración de todos los módulos de programación para concluir con una demostración al cliente de la funcionalidad del sistema y el desempeño en su propósito final. Los pasos en el proceso de pruebas incluyen:

1. Establecer los objetivos de la prueba
2. Diseñar los casos de estudio
3. Probar los casos de estudio
4. Efectuar las pruebas
5. Evaluar los resultados de la prueba

El objetivo de las pruebas es esencial para decidir que tipos de casos de estudio se generaran. Lo que es más, el diseño de los casos de estudio es la clave para una prueba exitosa. Si los casos de estudio no son representativos y no ejercen en su totalidad las funciones que demuestran lo adecuado de la operación del sistema, entonces el resto del proceso es inútil.

Por lo tanto, el efectuar una prueba comienza con una revisión de los casos de estudio para verificar que:

1. Sean correctos
2. Sean factibles
3. Cubran adecuadamente el proceso
4. Demuestren la funcionabilidad deseada

Una vez validados los casos de estudio de acuerdo a estos criterios puede procederse a la prueba del sistema.

Las filosofías de pruebas por etapas se basan en ciertos objetivos predeterminados para cada una de ellas. Una vez que los objetivos se establecen, se selecciona un conjunto de datos de prueba que cubren tantos casos como sea posible. La prueba de cada módulo y su integración puede incluir un análisis estático (evaluar los programas antes de ejecutarlos) y un análisis dinámico (monitorear el desempeño mientras se ejecuta el programa). Muchas de las herramientas para probar programas han sido automatizadas, y existen herramientas para todas las etapas del ciclo de pruebas.

8.2 Mantenimiento

El principal objetivo del desarrollo es la producción de código que implemente los requerimientos y funcione adecuadamente. En cada etapa del desarrollo, el equipo de trabajo se refiere continuamente a los requerimientos detectados. El diseño de los módulos se hace con base a las especificaciones del mismo y las pruebas se basan en verificar que las funciones y constantes del código operen de acuerdo a los requerimientos y a su diseño. De este modo, el mantenimiento involucra a las etapas de diseño y determinación de requerimientos de un modo controlado.

El soporte y mantenimiento del sistema son diferentes. El mantenimiento se basa en todo el desarrollo: requerimientos, diseño, código y pruebas, en adición, el equipo de mantenimiento analiza el estado actual del sistema estableciendo una relación de trabajo con los usuarios y los operadores para establecer en que grado están satisfechas sus necesidades con la operación del sistema. Finalmente el soporte ve hacia delante para anticipar las fallas que pudiera tener el sistema y repararlas antes de que causen alguna dificultad o daño. Dado que el mantenimiento abarca más que el desarrollo, la labor es más difícil, hay más que rastrear y controlar.

Actividades de mantenimiento

Las actividades propias del mantenimiento son similares a las del desarrollo: analizar requerimientos, evaluar el sistema y el diseño del programa, escribir o reescribir código, probar cambios y actualizar documentación. Por tanto, el equipo de mantenimiento –analistas, programadores y diseñadores– tienen papeles similares, sin embargo, dado que los cambios por lo general requieren conocimiento inmediato de la estructura y contenido del código del sistema los programadores juegan un papel más importante en el mantenimiento que el que ocuparon en el desarrollo.

Generalmente los usuarios, operadores o clientes se acercan al equipo de mantenimiento con un comentario o un problema. Los analistas o programadores determinan que partes del código se ven afectadas por el comentario o problema, como se afecta al diseño del sistema ante el resultado de las modificaciones y cuanto costará (en tiempo o recursos) implementar el cambio. De este modo el equipo de mantenimiento trabaja con el sistema para lograr varios fines:

1. Entender el sistema
2. Localizar la información en la documentación del sistema
3. Mantener al día la documentación del sistema

4. Ampliar las funciones existentes para adaptarse a nuevos requerimientos o a cambios en los existentes
5. Agregar nuevas funciones al sistema
6. Encontrar los errores fuente en el sistema
7. Corregir los errores identificados en el sistema
8. Responder a las preguntas sobre la funcionalidad del sistema
9. Reestructurar el diseño y el código
10. Reescribir el diseño y el código
11. Eliminar diseño y código de módulos que ya no se utilicen
12. Administrar los cambios al sistema al momento de hacerse

Además, el equipo de mantenimiento trabaja con usuarios, operadores y clientes. Primero tratan de entender el problema tal como lo expresa el usuario. Después, el problema se transforma en una solicitud de modificaciones que incluye una descripción de como funciona actualmente el sistema, como desea el usuario que opere el sistema y que modificaciones se necesitan para producir los cambios. Una vez que el diseño ó código es modificado y probado, el equipo de mantenimiento capacita al usuario si es necesario. De este modo, el mantenimiento involucra interacción con la gente tal como con el código y equipos.

Flujo del mantenimiento

El mantenimiento se enfoca en cuatro problemas principales:

1. Mantener control sobre las funciones de operación diaria
2. Mantener control sobre las modificaciones hechas al sistema
3. Perfeccionar las funciones existentes aceptables
4. Evitar que el desempeño del sistema se vea degradado a niveles inaceptables

Mantenimiento correctivo. Para controlar las funciones de operación diaria del sistema, el equipo de mantenimiento responde a problemas derivados de errores en el sistema. Encaminarse a este tipo de problemas se conoce como mantenimiento correctivo. Al presentarse errores se efectúan reparaciones de emergencia. Cambios a largo plazo pueden implementarse para corregir problemas más generales en el diseño o en el código de los procedimientos.

Mantenimiento de adaptación. Algunas veces, un cambio efectuado en una parte del sistema requiere de cambios en otras partes del sistema. La implementación de estos cambios *secundarios* se conoce como mantenimiento de adaptación. Por ejemplo, suponiendo que el DBMS existente se actualiza, en el proceso, los programadores descubren que las rutinas de acceso a disco requieren de un parámetro adicional. Los cambios realizados para agregar el parámetro adicional son de adaptación, no corrigen errores, simplemente permiten al sistema adaptarse a un cambio externo a sí mismo.

Mantenimiento perfectivo. Las funciones de un sistema se basan en el ambiente real en el cual habrán de operar. Los cambios hechos al ambiente resultan en cambios al sistema. Tales modificaciones se conocen como mantenimiento perfectivo. Por ejemplo, el programa descrito en el presente trabajo puede requerir mantenimiento perfectivo si una nueva ley amplia el campo de operación de la Dirección de Destino de Bienes. Un sistema operativo puede requerir mantenimiento perfectivo si se le requieren cambios para así reconocer un nuevo tipo de dispositivo de almacenamiento.

El mantenimiento perfectivo también incluye cambios al diseño o código para *mejorar el desempeño* del sistema. Por ejemplo, el sistema puede estar funcionando satisfactoriamente, pero el equipo de mantenimiento puede rediseñar el acceso a datos de los módulos para acelerar el rendimiento. Similarmente, un programador de mantenimiento puede reescribir algún algoritmo del sistema para mejorar el tiempo de respuesta al usuario. En estos casos no hay nada malo con el modo en que opera el sistema; los cambios hacen que el mismo se desempeñe mejor.

Mantenimiento preventivo. El último tipo de mantenimiento es similar al que se lleva a cabo con los equipos físicos para intentar prevenir descomposturas. El mantenimiento preventivo es todo trabajo efectuado al sistema en un esfuerzo de prevenir un error o mal funcionamiento del mismo. Un ejemplo es la adición de una rutina de validación a un campo de entrada del sistema. El validar las entradas capturadas por el usuario permite detectar los errores en la captura antes de que estos pasen a otros módulos donde la información de la base de datos puede corromperse. El equipo de mantenimiento también puede agregar código para rastrear los cambios efectuados a un archivo durante una sesión de edición para que de este modo los cambios puedan recrearse si la sesión no termina correctamente.

Uso de los recursos de mantenimiento

La figura 8.1 muestra el resultado de una serie de encuestas para determinar cuanto tiempo requiere cada tipo de mantenimiento[17]. Los resultados indican que la mitad del esfuerzo de mantenimiento es perfectivo: mejorar la calidad del sistema.

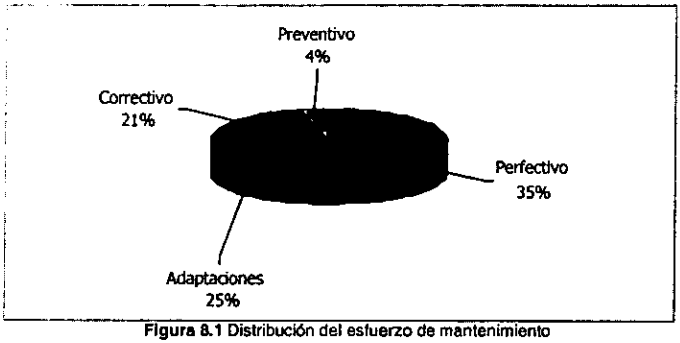


Figura 8.1 Distribución del esfuerzo de mantenimiento

[17] Lientz y Swanson, 1989

8.3 Documentación

Dirección General de Destino de Bienes
Solicitud de Servicio SICADEB

Usuario		Fecha:	
Solicitud			
Módulo(s) involucrado(s):			

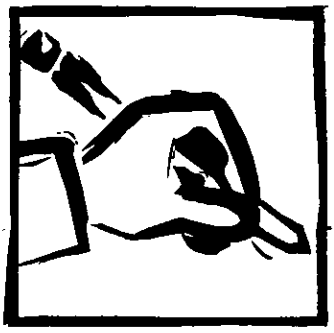
Para ser llenado por el personal de mantenimiento:

Diagnóstico	
Observaciones	

Conformidad de usuario

Técnico

Formato de solicitud de servicio al SICADEB



Conclusiones

Conclusiones

1. Si bien es cierto que la ingeniería en computación no trata como otras especialidades de la ingeniería con fenómenos naturales, sus premisas de operación si son las mismas. De este modo, los puntos a evaluar para un trabajo de esta naturaleza son:

   Actividades a cubrir en el proceso (alcance):

   - Alimentación al sistema después de captada la solicitud
   - Seguimiento a la solicitud a través de los diferentes procesos
   - Estadísticas de ejecución (tiempo, volumen, rechazos, etc.)
   - Medición del desempeño
   - Retroalimentación y publicación de la respuesta a los beneficiarios

   Que se reflejan en los siguientes beneficios:

   - Aumentar la velocidad de respuesta y mejorar la calidad del servicio
   - Saber el avance que tiene la petición del beneficiario y porderlo retroalimentar
   - Aumentar la transparencia del proceso
   - Mejorar la imagen que se tiene de la Dirección General

   En este sentido el sistema participó en la simplifación de los procesos propuesta por la reingeniería al detectar oportunidades de mejora, poder establecer parámetros de desempeño y al eliminar duplicidad de actividades o aquellas que no agregan valor, apoyando finalmente al control administrativo de la Dirección General al entender la naturaleza y evolución de las solicitudes recibidas mejorando el aprovechamiento de los recursos de la Dirección General.

   Agunas de las mejoras potenciales logradas son las siguientes:

   - Agilizar la captación de solicitudes
   - Evitar duplicidad de registros y seguimiento

- Crear parámetros de priorización de actividades
- Facilitar la interfase con las áreas
- Crear mecanismos de medición del desempeño
- Crear mecanismos de retroalimentación al usuario

2. Ajustarse a nuevas formas de operación toma tiempo. La tecnología puede llegar a ayudar a perpetuar viejas prácticas con sus vicios asociados dando una imagen de modernidad. El flujo de trabajo combinado con el uso mal encaminado de la tecnología lleva a que el mismo trámite administrativo se lleve a cabo bajo un esquema electrónico emulando viejas prácticas poco productivas. La tecnología por si sola no es reingeniería.

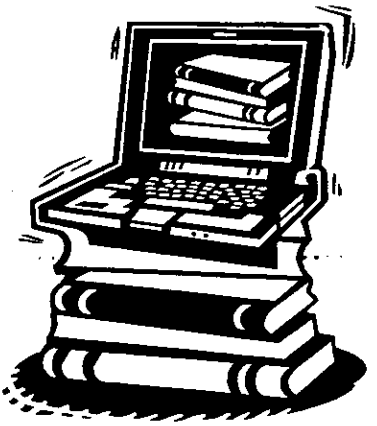
3. La reingeniería de organizaciones es ante todo el resultado del pensamiento creativo; pero en alguna etapa requiere soporte técnico, y este soporte se da por parte de los sistemas de información y de comunicación. Entre más integrados esten estos dos sistemas, daran un mejor soporte.

4. Un sistema es *útil* sí lleva a cabo la tarea que necesita llevarse a cabo, la cual no es efectiva o eficientemente realizada por humanos, les resulta tediosa o no tienen tiempo de hacer. El sistema es *práctico* sí esta disponible cuando requiere llevarse a cabo la tarea, es confiable en la ejecución de la misma y la lleva a cabo a satisfacción del usuario. El sistema es *asimilable* sí los usuarios tienen fácil acceso a él. Dado que la facilidad de acceso contribuye a la satisfacción del usuario, esta puede hacerlo aún más utilizable.

5. La utilidad de un sistema puede determinarse en un principio por medio de un análisis costo-beneficio, pero es difícil expresar algunos de sus beneficios en términos cuantitativos. Por ejemplo, dado un proceso al que se aplicó reingeniería para que cierta labor que el personal

encontraba tediosa se hiciese de manera automática, y que la actitud hacia el mencionado proceso ha cambiado desde que se implanto el sistema, hasta que grado fue gracias al nuevo sistema.

6. Para el caso de un proyecto de información como el aquí propuesto no esta en el desarrollador establecer el grado de funcionalidad del mismo, dado que la función final del sistema la establecen los lineamientos que dan razón de ser a la Dirección de Destino de Bienes solo ellos, los usuarios finales, tras evaluar los resultados obtenidos podrán juzgar el proyecto como exitoso o no.



Bibliografía

Análisis y Diseño de Sistemas de Información
James A. Senn
Ed. Mc Graw Hill, 1997

Técnicas de Bases de Datos: Estructuración en Diseño y Administración
Shakuntala Atre
Ed. Trillas, 1993

Introducción a los sistemas de bases de datos
Chris Date
Ed. Mc Graw Hill, 1997

Aplique SQL
James R. Groff y Paul N. Weinberg
Osborne Mc Graw Hill, 1993

Organización de las bases de datos
James Martín
Prentice-Hall Hispanoamericana, 1997

Ingeniería de Software un Enfoque
PRESSMAN
Mc Graw Hill, 1995

Software Engineering: The Production of Quality Software
Shari Lawrence Pfleeger
Macmillan, 1997

Software Methods for Business Reengineering
Alfs Berztiss
Springer, 1998

System Administration for Microsoft SQL Server 6.5
Microsoft Education and Certification, 1996

Implementing a Database Design on Microsoft SQL Server 6.5
Microsoft Education and Certification, 1996

Visual Basic for Windows
William J. Orvis
Sams Publishing, 1997

Mastering Microsoft Visual Basic 5
Microsoft Education and Certification, 1997

Internet

msdn.microsoft.com/vbasic
Microsoft Visual Basic Home Page
Octubre 1998

www.seagatesoftware.com/products/crystalreports/
Seagate Software: Crystal Reports
Febrero 1999

www.shcp.gob.mx/estruct/oficmay/dgdbce.html
Dirección General de Destino de Bienes de Comercio Exterior Propiedad del Fisco Federal
Julio 1999

www.y2k.gob.mx
Comisión Nacional para la Conversión Informática del año 2000
Agosto 1999