UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMA INTEGRAL DE ADMINISTRACION INMOBILIARIA

T E S I S

QUE PARA OBTENER EL TITULO DE

INGENIERO EN COMPUTACION

P R E S E N T A N :

ERNESTO GONZALEZ CARDENAS
JOSE UBALDO CHAVEZ CRUZ
LETICIA IRASEMA VIAU GOMEZ

DIRECTOR DE TESIS:
ING. ORLANDO ZALDIVAR ZAMORATEGUI

MEXICO, D.F. 1998

Esta tesis es dedicada a todas las personas que me apoyaron para lograr este trabajo.

A mis padres:
Por haberme dado la vida y una educación, por el esfuerzo realizado cuando era pequeño, a mi madre como una promesa cumplida.

A mis abuelos:
En memoria de mis abuelos que Dios los recogió antes de poder darles esta satisfacción.

A mis hermanos por la unión que han mostrado en la familia y como un estimulo para seguir superandonos.

A mis tíos por el apoyo recibido desde mi niñez.

A Alma por haberme apoyado y dado ánimos para lograr este trabajo, así como su comprensión en los momentos difíciles.

A la Universidad por ser la escuela que me formó como profesionista.

A nuestro director de tesis, el Ing. Zaldivar por el apoyo recibido durante la elaboración del trabajo.

Y a todas las personas que ayudaron y apoyaron en la elaboración de la tesis.

Ernesto González Cárdenas.

... A la Universidad.
... a la facultad.

GRACIAS.

José Ubaldo Chavez Cruz.

A mis padres por su
apoyo incondicional.

Leticia Irasema Viau Gómez

CONTENIDO

CAPITULO 1

Introducción

1. INTRODUCCION

Actualmente, en nuestro país las empresas enfrentan diversos obstáculos para hacer frente a las circunstancias del mercado, el cual es cada vez más competitivo, por lo que salir adelante, consolidarse y sobresalir en cualquier rama de la industria, requiere de grandes inversiones en recursos económicos, materiales y humanos.

Como parte de estos obstáculos, resalta uno que es fundamental: la falta de un sistema que permita la obtención oportuna y exacta de toda la información disponible dentro de la empresa y que ayude a simplificar y controlar de manera significativa gran parte de los procedimientos y procesos que se llevan a cabo dentro de los diferentes departamentos de la misma.

El presente trabajo contribuye de manera determinante a solucionar el problema de la organización, obtención y almacenamiento de la información dentro de la Empresa Inmobiliaria a través del análisis, diseño, desarrollo e implementación del Sistema Integral de Administración Inmobiliaria (SIAI), el cual, es un sistema de cómputo real, creado para solucionar un problema real.

Algo muy importante, es que ha sido elaborado tomando en cuenta las diferentes etapas que comprende la metodología del análisis y diseño estructurado.

El SIAI cuenta con los módulos y características necesarias para manejar la información referente a:

- Inventario de inmuebles.
- Programación de mantenimientos.
- Estimación de costos y tiempos de mantenimiento.
- Ejecución de mantenimientos y reparaciones.

Nosotros estamos seguros de que los resultados obtenidos por el SIAI, permitirán a la Empresa Inmobiliaria hacer de su información un recurso más para competir, además de que éstos ayudarán e influirán directamente en la toma de decisiones a nivel directivo y gerencial.

Nuestra tesis se compone de diez capítulos, cada uno de ellos va enfocado a cada una de las diferentes etapas del proyecto, también se incluye una parte de conclusiones, bibliografía y anexos.

A continuación se presenta una descripción breve del contenido de cada uno de los capítulos.

Capítulo 1. Aquí, se presenta una introducción general y una descripción breve del contenido de cada capítulo.

Capítulo 2. Este abarca toda la parte teórica que se investigó y en la que se apoyan las etapas de análisis, diseño y desarrollo, se incluyen temas importantes como análisis estructurado, diseño estructurado y bases de datos relacionales.

Capítulo 3. En este capítulo se lleva a cabo el estudio inicial del sistema, es donde se define el mismo, se hace un diagnóstico de la situación actual y se determina la factibilidad del nuevo sistema.

Capítulo 4. Consiste en determinar el verdadero alcance del sistema, se definen las metas y requerimientos del mismo, los recursos que se van a emplear y la estimación del costo del proyecto.

Capítulo 5. Es, sin lugar a dudas, un capítulo que provee de información fundamental para las siguientes etapas del sistema, pues abarca los diagramas de flujo de datos de la empresa y el diagrama entidad-relación, sin los cuales no es posible llegar a un diseño estructurado. También se tocan temas como el español estructurado aplicado a un flujo de datos real del sistema.

Capítulo 6. En este punto, se obtiene una estructura general del sistema mediante la carta de estructura, a la vez de aplicar los conceptos de modularidad, cohesión y acoplamiento a dicha estructura.

Capítulo 7. Es donde se decide, en base a una lista de criterios el software manejador de base de datos y el software de desarrollo para el sistema, habiendo tomado en cuenta diferentes opciones del mercado.

También se describen las características del lenguaje de programación elegido para la aplicación y los procesos realizados para obtener las imágenes y archivos de sonido empleados por el sistema.

Capítulo 8. La parte relativa a la documentación del sistema es abarcada en este capítulo. Aquí mencionamos lo referente a los manuales o en su caso, ayuda en línea de la que dispone el sistema.

Capítulo 9. Comprende la realización de las pruebas a las que se sometió el sistema, como por ejemplo: la prueba de validación, la prueba de integración, etc.

Capítulo 10. Trata todo lo relativo a la instalación del sistema, la carga de archivos, la capacitación y la aprobación final del sistema, esta es una de las etapas donde es primordial la participación del personal de la Empresa Inmobiliaria, el cual juega un papel muy importante: el usuario final.

Conclusiones. Aquí mencionamos lo que obtenemos como resultado final de la realización de este proyecto, cabe mencionar que para llegar a una conclusión final es muy importante valorar todo el esfuerzo y participación de todos los involucrados, pero principalmente se debe tomar en cuenta el cumplimiento de los objetivos iniciales.

Bibliografía. Es una lista de los libros o textos de diversos autores, que utilizamos como referencia y/o consulta para el desarrollo de este proyecto.

Anexos. Como parte de los anexos incluimos una serie de información que de alguna manera complementa y enriquece alguna etapa del proyecto.

CAPITULO 2

Ingeniería de Programación

2. INGENIERIA DE PROGRAMACION

2.1 ESTUDIO INICIAL DEL SISTEMA.

2.1.1 DEFINICION DEL SISTEMA.

La *definición del sistema* es una parte muy importante dentro del estudio inicial, ya que es aquí donde se tienen que determinar varios aspectos, que más tarde estarán directamente involucrados con el análisis y diseño del mismo como son: la definición del problema y el establecimiento de metas y requerimientos que se pretende cumpla el sistema, entre otros.

Dicha definición debe cumplir hasta donde sea posible con la idea general de que un sistema de información es utilizado por las empresas como un recurso, para poder competir. Esta definición debe ser pensada de tal manera que el sistema pueda ser diseñado para cumplir con los tres atributos fundamentales de todo sistema de información: oportunidad, exactitud y relevancia.

La definición del sistema se inicia desde la planeación del proyecto y ésta requiere de tres pasos:

- Definición del problema
- Desarrollo de una estrategia de solución
- Planeación del proceso de desarrollo

Definición del problema.

Generalmente cuando se piensa en un proyecto de desarrollo e implementación de un sistema de información y por consecuencia la definición del mismo, es necesario que se identifique el problema, además, la determinación de éste nos dará un gran avance sobre la definición del sistema. Para identificar el problema son necesarios los siguientes pasos:

"1. Desarrollar un enunciado definitivo del problema por resolver. Incluir una descripción de la situación actual, restricciones del problema y de las metas que se lograrán. El enunciado del problema debe realizarse empleando terminología del área a la que pertenezca la organización que requiere el sistema. Esto obedece a que no se puede emplear la misma terminología para todas las áreas profesionales. Por ejemplo: la terminología para un sistema de inventario puede diferir a la de uno de nómina o uno de control de tráfico aéreo, etc.
2. Justificar una estrategia de solución computarizada para el problema.
3. Identificar las funciones por realizar, las restricciones, el subsistema de equipo electrónico, el subsistema del producto de programación y el del personal.
4. Determinar los objetivos y requisitos en el nivel del sistema para el proceso de desarrollo y los productos finales.
5. Establecer criterios de alto nivel para la aceptación del sistema."[1]

La definición del problema requiere de un entendimiento, conocimiento y dominio total del problema y del entorno en el que éste se desarrolla. Las técnicas para obtener este conocimiento por parte del analista consisten en entrevistas con el cliente, la observación de las tareas problemáticas y el desarrollo de los procedimientos reales.

El analista debe ser muy hábil en la técnica de definición del problema ya que los distintos representantes del cliente tendrán diferentes puntos de vista y prejuicios acerca del área o áreas del problema. Además los representantes del cliente quizás no estén familiarizados con las posibilidades que una computadora pueda ofrecer en su situación, y rara vez son capaces de formular sus problemas de modo que sea factible su análisis lógico.

Desarrollo de una estrategia de solución.

Es el segundo paso en la planeación de un proyecto de programación. Esta estrategia además de ser eficaz en términos de costo, debe aceptarse social y políticamente. Para ser eficiente en costo, un nuevo producto de programación debe proporcionar los mismos servicios e información que el sistema antiguo, usando menos tiempo y personal, o proporcionar servicios e información que antes eran inaccesibles. Un sistema que desplace muchos trabajadores puede ser económica y técnicamente posible, pero inaceptable social o políticamente para el usuario.

Se debe poner mucha atención a la hora de elegir una solución al problema porque existe una tendencia a utilizar la primera que aparece. Una manera de evitarlo es desarrollar una estrategia de solución, la cual no es un plan detallado de solución, sino un enunciado general sobre la naturaleza de las posibles soluciones. Los factores estratégicos incluyen procesamiento por lote o por tiempo compartido; base de datos o sistema de archivos; gráficas o texto y procesamiento en tiempo real o en línea.

[1] Fairley Richard; Ingeniería de Software; McGraw-Hill, 1988, México, pág. 33.

Una estrategia de solución debe considerar todos los factores externos que son visibles para el usuario del producto, y debe redactarse de tal manera que permita caminos alternos para el diseño del producto.

Planeación del proceso de desarrollo.

Habiendo determinado, por lo menos en una forma preliminar que es apropiada una solución computarizada para el problema, la atención se centra en las funciones de los principales subsistemas del sistema computacional. Un sistema computacional está formado por los subsistemas de personal, equipo y de productos de programación, más las interconexiones entre ellos. El primer subsistema incluye operadores, personal de mantenimiento y usuarios finales. El segundo comprende el equipo de cómputo y los dispositivos periféricos. El tercer subsistema contiene programas que deben desarrollarse, más programas que ya existen y pueden emplearse como están o modificándolos. Deben identificarse y definirse las funciones de cada subsistema y la relación entre ellos para poder planear el proceso de desarrollo que demanda el sistema.

La planeación del proceso de desarrollo de un sistema implica considerar la definición del ciclo de vida del mismo, éste incluye todas las actividades requeridas para definirlo, desarrollarlo, probarlo, entregarlo, operarlo y mantenerlo. Es esencial definir un modelo de ciclo de vida para cada proyecto, ya que permite clasificar y controlar las diferentes actividades necesarias para el desarrollo y mantenimiento del producto.

Un modelo de ciclo de vida que es aceptado y entendido por las partes interesadas en el proyecto mejora la comunicación, lo que permite una mejor administración, asignación de recursos, control de costos y calidad del producto.

Un modelo de ciclo de vida tiene las siguientes fases: ingeniería de sistemas, análisis, diseño, codificación, prueba del sistema y mantenimiento.

Determinación de metas para el sistema.

Una vez que se tiene identificado con precisión el problema y se han indicado las restricciones para su solución se pueden determinar metas y requisitos. Las metas son logros por alcanzar; y nos sirven para establecer el marco de referencia para el proyecto de desarrollo del producto de programación. Estas se aplican tanto para el proceso de desarrollo como para los productos finales y pueden ser cualitativas o cuantitativas, por ejemplo:

Para el proceso:

- Meta cualitativa. El proceso de desarrollo debe mejorar las habilidades profesionales del personal de control de calidad.
- Meta cuantitativa. El sistema se debe entregar en un plazo máximo de doce meses.

Para el producto:

- Meta cualitativa. El sistema debe hacer más interesante el trabajo de los usuarios.
- Meta cuantitativa. El sistema debe reducir el costo de una transacción en un 25%.

Algunas metas se aplican a todos los proyectos y productos. Por ejemplo: todo producto de programación debe ser útil, confiable, comprensible y eficiente en costos.

Todo proceso de desarrollo debe generar productos finales a tiempo, dentro de los estimados de costo y debe permitir que el personal del proyecto aprenda nuevas habilidades. *"Otras metas, como transportabilidad, entrega anticipada de subsistemas y facilidad de uso para los no programadores, dependerán de la situación particular."*[2]

El establecimiento de metas implica la revisión del alcance de las operaciones de la organización, las políticas de sistemas y el plan de la empresa. El objetivo es definir las metas de la organización y encadenarlas con las metas de los sistemas de información.

A partir de este proceso, empiezan a surgir ideas de proyectos en sistemas para el soporte de estas metas.

Para esta etapa la recopilación de información es útil, también la revisión de documentación adicional como planes previos de sistemas, revisiones posteriores a la implementación y reportes periódicos de la evaluación de los sistemas, organigramas y descripciones de puestos, reportes de auditorías internas y externas y documentación de sistemas y manuales de procedimientos.

A partir de este proceso de investigación, se formulan metas generales de sistemas de información. Estas metas pueden plantearse como:

1. Diseñar e implementar un sistema que de apoyo a las metas organizacionales.
2. Explotar las oportunidades de negocios proporcionadas por las nuevas tecnologías informáticas.
3. Seguir una metodología para desarrollo de sistemas que interactúen con los usuarios y proporcione el estado y el progreso de los nuevos proyectos de sistemas.

[2] Fairley Richard; Ingeniería de Software; McGraw-Hill, 1988, México, pág. 35.

Determinación de requerimientos para el sistema.

Los requisitos especifican las capacidades que debe tener un sistema para la solución de un problema. Estos se establecen para la funcionalidad, el rendimiento, el equipo, la programación en el equipo y las interfaces con el usuario. Los requisitos pueden establecer también estándares de desarrollo y de control de calidad tanto para el desarrollo como para el producto; deben ser cuantificados siempre que sea posible.

Es difícil cuantificar requisitos en la fase de planeación porque por lo regular, no está claro qué se necesita para resolver el problema o qué se puede lograr dentro de las restricciones. Sin embargo, se debe realizar un esfuerzo para formular requisitos con significado y los métodos que se utilizarán para verificar cada requisito.

Otros requerimientos de sistemas son los operacionales, inherentes al sistema de información mismo y surgen especialmente por la producción de información de calidad y por el objetivo claro de mejorar u optimizar el o los sistemas que operan actualmente, estos sistemas actuales deben ser superados por el nuevo proyecto en cuanto a funcionalidad, eficiencia y cobertura. Los requerimientos son:

1. Confiabilidad.

La confiabilidad se refiere al grado de seguridad con que un recurso realiza su función, produciendo los mismos resultados en procesos sucesivos.

2. Disponibilidad.

La disponibilidad significa que el sistema es accesible a los usuarios. Un sistema puede ser confiable, pero no estar disponible cuando está siendo probado o cuando se le está agregando un componente. Por otra parte un sistema puede estar disponible pero no ser confiable.

3. Flexibilidad.

El requerimiento de flexibilidad se refiere a la habilidad del sistema para cambiar o adaptarse para satisfacer los requerimientos cambiantes de los usuarios.

4. Programa de instalación.

El programa de instalación comprende el espacio de tiempo existente entre el momento en que una organización reconoce una necesidad y el momento en que implementa la solución.

5. Expectativa de vida y potencial de crecimiento.

Algunos sistemas no cuentan con una expectativa de vida debido a que ya son obsoletos en el momento en que se implementan. O bien, un sistema puede instalarse

y trabajar muy bien durante cierto tiempo pero debido a que es un sistema con una sola salida sin la capacidad de crecer, queda obsoleto cuando aumentan las necesidades.

Por lo tanto los sistemas deben diseñarse para satisfacer requerimientos durante un tiempo razonable y ser también capaces de crecer si las necesidades cambian de manera significativa.

6. Capacidad para recibir mantenimiento.

Una vez que un sistema se implementa, debe recibir mantenimiento, debido a que se deben corregir fallas, a que se deben satisfacer solicitudes especiales y a que deben efectuarse mejoras generales a los sistemas, siempre y cuando estén contempladas desde la fase de análisis. La meta, por lo tanto, deberá ser la de diseñar sistemas que sean capaces de recibir mantenimiento mediante el empleo de estándares, programación estructurada y modular, configuraciones estándares y documentación.

Existen además, los requerimientos de procesos de datos, los cuales se refieren al trabajo de detalle del sistema y se dividen en cuatro categorías:

1. Volumen.

El volumen se refiere a la cantidad de datos que deben procesarse en un periodo dado, para lograr una meta de la información. Una forma de cuantificar el volumen podría ser la de hacer referencia a las transacciones organizacionales.

2. Complejidad.

La complejidad se refiere al número de operaciones de datos interrelacionados que se deben realizar para lograr una meta de la información.

3. Restricciones de tiempo.

Estas se definen como la cantidad de tiempo permitido o aceptable entre el momento en que los datos están disponibles y el momento en que la información se requiere.

4. Demandas computacionales.

Las demandas computacionales son una combinación única de volumen, complejidad y restricciones de tiempo para un requerimiento específico de información. Estas demandas computacionales pueden ser considerables si se debe procesar un modelo grande de programación lineal o si se debe dar mantenimiento en línea a una base de datos grande.

En general para determinar y/o identificar cualquier tipo de requerimiento de información dentro de la empresa, pueden utilizarse diversos instrumentos, los cuales

incluyen: el muestreo, el estudio de los datos y formas usadas por la organización, la entrevista, los cuestionarios, la observación de la conducta de quien toma las decisiones, así como su ambiente y también el desarrollo de prototipos.

En esta etapa el analista hace todo lo posible por identificar qué información requiere el usuario para desempeñar sus tareas. Puede ver cómo varios de los métodos para establecer las necesidades de información, lo obligan a relacionarse directamente con los usuarios. Esta etapa sirve para elaborar la imagen que el analista tiene de la organización y de sus objetivos.

A continuación se presentan algunos factores que deben considerarse al establecer metas y requerimientos del sistema.

- Nivel de complejidad del usuario.
- Requisitos de eficiencia.
- Requisitos de confiabilidad.
- Modificaciones factibles.
- Requisitos de portabilidad.
- Asuntos de seguridad.

Finalmente se ha considerado a la definición del problema y a la determinación de metas y requerimientos, como las partes más sobresalientes de la definición del sistema, pero cabe señalar que la definición del sistema abarca algunos otros parámetros que también son importantes para una buena definición del sistema tales como justificación y restricciones del sistema.

Un formato completo de la definición del sistema con sus respectivas secciones es el siguiente:

1. Definición del problema.
2. Justificación del sistema.
3. Metas y requerimientos del sistema.
4. Restricciones del sistema.
5. Características del usuario.
6. Ambientes de desarrollo/operación/mantenimiento.
7. Criterios de aceptación del sistema.
8. Fuentes de información.
9. Glosario de términos.

De este formato se desprende la necesidad de contar con equipos de trabajo que resuelvan las diferentes secciones que contiene, tanto por la parte solicitante del sistema como por la parte que se va a encargar de planear, organizar, dirigir, coordinar y controlar el proyecto del sistema.

Por lo tanto, debe quedar bien claro que para llegar a una definición del sistema es esencial la participación de diversos sectores de la organización o empresa (desde la

dirección hasta los usuarios), cada uno con funciones específicas y trabajar en conjunto con el analista de sistemas para llegar a definir el sistema que pueda resolver las necesidades de los usuarios, las metas y objetivos planteados por la empresa, todo esto respetando las restricciones impuestas y cumpliendo con los requerimientos establecidos.

Las secciones: criterios de aceptación del sistema, fuentes de información y glosario de términos del formato de definición del sistema se consideran necesarias pensando en que este documento pueda estar al alcance de los grupos (dentro de la empresa) que no estén directamente involucrados con la toma de decisiones, las fuentes de información o la terminología empleada en el proyecto. Así, el formato de definición del sistema no queda como un documento exclusivo de entender por el analista o las personas más involucradas en el proyecto.

2.1.2 DIAGNOSTICO DE LA SITUACION ACTUAL.

El diagnóstico de la situación actual es una fase dentro del proyecto del sistema que es necesario abarcar ya que éste obliga a narrar un planteamiento claro y más perceptible del problema que enfrenta actualmente la organización, es decir, describe la situación actual que se utilizará como punto de partida para considerar nuevas metas a las cuales aspira la empresa con el proyecto de análisis, diseño e implementación del nuevo sistema de información.

El diagnóstico de la situación actual definitivamente depende del estudio y evaluación real y en el momento presente del sistema actual con que trabaja la organización, sea o no un sistema automatizado.

Estudio del sistema actual.

Resulta poco común que un analista de sistemas tenga la oportunidad de desarrollar un sistema de información en donde anteriormente no haya existido ninguno. En la mayoría de los casos existe un sistema o subsistema que da servicio a la organización.

Como resultado, el analista se enfrenta con decisiones del tipo ¿qué papel desempeña el sistema actual con respecto al nuevo sistema?, ¿se debe analizar el sistema actual? y si es así ¿qué módulos se deben analizar del sistema actual?. Con frecuencia se dedica gran cantidad de dinero y tiempo para analizar el sistema actual y en muchos casos esta inversión es sólo para determinar que realmente se tiene razón en solicitar uno nuevo. Sin embargo, muchos afirman que el primer paso en todos los análisis de sistemas es el análisis del sistema actual y éste como parte del diagnóstico de la situación actual.

Para determinar qué tan profundamente debe analizarse el sistema actual, es recomendable un examen de las ventajas y desventajas del diagnóstico de éste.

Ventajas de analizar el sistema actual.

1. Eficacia del sistema actual.

El estudio y diagnóstico del sistema actual proporciona una oportunidad para determinar si dicho sistema es satisfactorio, requiere de reparaciones menores, requiere de mantenimientos considerables o debe ser reemplazado.

2. Ideas de diseño.

El analista del sistema actual puede proporcionar una fuente inmediata de ideas para el diseño. Estas ideas incluyen lo que se está haciendo actualmente y en qué forma, así como las necesidades o capacidades adicionales que han sido solicitadas con el paso de los años. El analista puede obtener una mayor comprensión de la forma en que el

sistema de información actual sirve a la función de toma de decisiones, así como para determinar relaciones clave.

3. Reconocimiento de recursos.

El examen del sistema actual le permite al analista identificar los recursos disponibles para el nuevo sistema o subsistema. Estos recursos podrían incluir el talento de la gerencia, el talento del personal de oficinas y el equipo que se posee actualmente y está en operación.

4. Conocimiento de conversión.

Cuando se implemente el nuevo sistema, el analista de sistemas tiene la responsabilidad de haber determinado previamente las tareas y actividades que serán necesarias para ir abandonando gradualmente el sistema actual y empezar a operar el nuevo sistema. Para identificar estos requerimientos de conversión, el analista debe conocer no solamente qué actividades se realizarán sino también qué actividades se realizaban. El estudio del sistema actual le da al analista la respuesta al "qué había".

5. Punto de partida común.

Cuando se comunica con la gerencia, el analista de sistemas es un agente de cambio. Como tal, el analista con frecuencia se enfrentará a la resistencia a las nuevas técnicas, ideas y métodos, a la falta de comprensión de los nuevos conceptos, retrasos en la obtención de las decisiones, falta de compromiso para hacer que funcione el nuevo sistema y otros aspectos similares de las personas a las que se les solicita que cambien las actividades con las que están familiarizados.

Para minimizar estas reacciones, el analista puede comparar y contrastar el nuevo sistema con el sistema anterior y demostrar que éste no es totalmente nuevo, sino que en determinado momento tiene las funciones y procesos rescatables del sistema actual. Sin embargo, debe ser muy claro en el sentido de que, con el nuevo sistema se mejorará sustancialmente el funcionamiento de la empresa, ya que en caso contrario no hay razón para cambiar.

Desventajas de analizar el sistema actual.

1. Gastos.

El estudio del sistema actual requiere tiempo, y en todas las organizaciones el tiempo puede convertirse en dinero. En este sentido surge la pregunta ¿Por qué pasarse meses analizando y modelando el sistema que va a desaparecer? Por ello se considera una desventaja.

2. Barreras innecesarias.

Un análisis extenso de un sistema existente puede dar por resultado que se incluyan barreras innecesarias o restricciones artificiales en el diseño del nuevo sistema. Entre más se familiarice el analista con un sistema dado, es más probable que se pierda cierta perspectiva u objetividad con relación a él.

Además del estudio de la situación actual existen dos fuentes de información que nos pueden ser de gran utilidad en el diagnóstico de la situación actual: fuentes internas y fuentes externas.

Fuentes internas de información.

La fuente más importante de hechos de estudio para el diagnóstico de la situación actual es la gente. Esta incluye no sólo la gerencia formal sino también a los empleados de oficina y de producción. Los requerimientos de información pueden ser planteados mejor por los usuarios de la información. Sin embargo el analista puede ayudar a los usuarios a definir sus requerimientos explicándose lo que puede proporcionarse. La función del analista es la de eliminar actitudes desfavorables de los usuarios, a fin de obtener los verdaderos requerimientos de información.

Una fuente secundaria de hechos de estudio para el analista proviene de los documentos existentes dentro de la organización. Estos documentos pueden clasificarse como el que describe la forma en que la organización está estructurada, el que describe lo que la organización está haciendo y el que describe lo que la organización desea hacer.

A continuación se presenta una lista del contenido que debe tener cada documento.

Documento que describe cómo está organizada la empresa.

Contenido: Declaraciones políticas.
Manuales de métodos y procedimientos.
Organigramas.
Descripción de puestos.
Delegación de desempeño.
Delegación de autoridad.
Catálogo de cuentas.

Documento que describe lo que hace la empresa.

Contenido: Estados financieros.
Reportes de desempeño.
Estudios por personal asesor.
Reportes históricos.
Archivos de transacciones.

Documentos legales.
Archivos de referencia maestros.

Documento que describe lo que planea hacer la empresa.

Contenido: Declaración de metas y objetivos.
Presupuestos.
Programas.
Pronósticos.
Planes (a largo y corto plazo).
Minutas corporativas.

Una tercera fuente de estudio importante para el analista dentro de las fuentes internas son las relaciones entre las personas, los departamentos y las funciones. Estas relaciones pueden proporcionar al analista información e ideas profundas que no se conocían anteriormente o que no se encuentran documentadas en ninguna parte dentro de la organización.

Fuentes externas de información.

El trabajo del analista de sistemas lo puede llevar fuera de los límites de la organización. Es también significativa la revisión de otros sistemas de información similares en otras organizaciones. Esto no solamente puede ser una fuente de nuevas ideas, sino que también puede proporcionar al analista una oportunidad de ver realmente sistemas, subsistemas, conceptos, técnicas y mecanismos en operación.

Otras fuentes externas de información que pueden ayudar al analista a hacer un diagnóstico de la situación actual son: libros de texto y revistas profesionales, seminarios, talleres, conferencias, folletos de ventas de los proveedores de equipo, etc.

Técnicas de recopilación de información en el diagnóstico de la situación actual.

1. Entrevistas.

Es una forma de obtener información crítica por parte del analista y permiten a quienes responden, contribuir al análisis. Es importante que el analista se asegure que cada persona que responde entienda que el objetivo final del trabajo es hacer que el nuevo sistema sea más útil. La entrevista puede usarse para obtener apoyo o comprensión por parte del usuario acerca de una nueva idea o método y debe realizarse a todos los niveles de la organización.

2. Método de análisis en grupos.

Esta técnica al abarcar más personas (en grupos de trabajo) puede hacer que el analista se fortalezca en sus ideas o bien perciba la inconformidad o desacuerdo de los diferentes grupos, ya que cada idea será aprobada o rechazada ya no por una persona,

1. Factibilidad técnica.
2. Factibilidad económica.
3. Factibilidad legal.
4. Factibilidad operacional.
5. Factibilidad de programa.

Estos componentes ayudan a determinar de manera clara y precisa si una estrategia de solución es factible o no. A continuación se explica cada uno de ellos.

1. Factibilidad técnica.

Para decidir la factibilidad técnica, el analista determina si se puede desarrollar e implementar la estrategia de solución propuesta empleando la tecnología existente. Esta determinación generalmente incluye la experiencia tecnológica que existe actualmente dentro de la organización, pero también puede incluir la experiencia de lo más avanzado en tecnología fuera de la organización. Como afirmación general, la tecnología está mucho más adelante de la capacidad de la gente para aplicarla con efectividad. Por esta razón, en la mayoría de los casos, la factibilidad técnica desde el punto de vista de disponibilidad es un aspecto no relevante; pero si cabe resaltar que el problema es el de la habilidad para adquirirla, aplicarla y utilizarla.

De hecho, debido a que la tecnología aglutina y soporta a los otros componentes estructurales, el nivel de acceso a ella tendrá claramente un impacto significativo en la forma en que finalmente se diseñe el sistema.

2. Factibilidad económica.

Esta área de factibilidad origina una pregunta básica que se deben hacer todas las organizaciones o empresas cuando intentan emprender un proyecto de esta naturaleza:

¿Cuenta la organización con los fondos necesarios para desarrollar e implementar un sistema de información, dados los requerimientos de otros proyectos de capital dentro de la organización?

Si la respuesta es sí, se origina una segunda pregunta:

¿Cuál es el nivel de compromiso financiero?

Todo esto se origina porque en muchos casos de manera simultánea al proyecto del sistema surgen otros proyectos también importantes para la organización, entonces se evalúa la inversión económica que representa el proyecto del sistema y se considera si estos recursos debieran canalizarse en otras necesidades de la empresa u organización.

Por lo tanto, es obvio que el nivel de análisis y diseño y el alcance del proyecto del sistema de información están relacionados directamente con el apoyo económico.

3. Factibilidad legal.

Este factor ordena que no exista ningún conflicto entre el sistema que se está considerando y la capacidad de la organización para descargar sus obligaciones legales. El analista debe considerar en este aspecto las implicaciones legales que surjan de los estatutos aplicables federales y estatales, las reglas de la ley común, las agencias administrativas etc. y las cláusulas contractuales. Por ejemplo, al considerar los requerimientos de retención de registros, el analista debe conocer qué registros se deben retener, quién debe guardarlos, cuánto tiempo deben guardarse y el nivel de seguridad requerido. Actualmente, la legalidad también tiene que ver con el uso y licencias del software que se emplea para desarrollar los sistemas, es decir si no se tiene la autorización del fabricante para usar el software de desarrollo en un número determinado de equipos, se viola la legalidad. Por lo tanto si el software empleado en el desarrollo del sistema es ilegal, el sistema desarrollado será ilegal también.

4. Factibilidad operacional.

Esta factibilidad consiste en determinar si el análisis y el diseño están basados en el ambiente organizacional, los procedimientos existentes y el personal con que opera la organización. En caso de no ser así se debe investigar y definir si se pueden adquirir las habilidades suficientes, adiestrar y capacitar al personal y efectuar otros cambios para que el sistema sea operacional. En caso de que así sea, el sistema cumplirá con esta factibilidad. En caso contrario es necesario que haya cambios en el análisis y diseño del sistema para que éste sea operacional dentro de las condiciones existentes.

5. Factibilidad de programa.

Esto significa que el diseño del sistema debe ser capaz de ser operativo dentro de algún plazo de tiempo. Si no es así el diseño o el plazo de tiempo tendrán que cambiar porque la implantación de un nuevo sistema debe empezar a operar y dar resultados en un período establecido de tiempo ya sea de manera parcial o total, para así empezar a justificar su existencia.

Dado que se tienen diversos componentes de los requerimientos de factibilidad es muy recomendable decidir sobre en qué momento una estrategia de solución es factible de realizarse o no, es decir si debe cumplir con todos los tipos de factibilidad explicados anteriormente o con la mayoría de ellos. Es también de gran utilidad darle un cierto porcentaje o un cierto peso a cada uno de los componentes de factibilidad anteriores basándose principalmente en el nivel de importancia que cada uno tenga dentro de la organización y en las consecuencias inmediatas y futuras que representa el que una alternativa no cumpla con algún componente de la factibilidad. En muchos casos la factibilidad económica es la de mayor peso cuando la implementación del proyecto del sistema implica la adquisición de nueva tecnología, contratación de personal,

capacitación, etc., pero sólo se puede determinar qué tanto porcentaje del 100% de la factibilidad se le da a la factibilidad económica realizando un análisis del costo/beneficio.

Todo lo anterior nos ayuda para poder visualizar en base a una comparación, cuál de las diferentes estrategias de solución es factible de realizarse y en caso de que varias sean factibles, cuál de ellas lo es más, es decir cuál se acerca más a ser 100% factible de realizarse.

Una vez realizado el análisis de factibilidad, la aprobación de éste y la aceptación de la propuesta que haya cumplido con los componentes de factibilidad se da paso, ahora sí, al análisis ya detallado del sistema, al diseño, a la determinación específica de los requerimientos y a la planeación en todos los niveles del desarrollo e implementación del proyecto, pero tomando como base todos los documentos que anteriormente se hayan elaborado de manera general o no tan detallada de los diversos aspectos del proyecto.

2.1.4 ANALISIS DEL SISTEMA.

El desarrollo de un sistema de información, independientemente de su tamaño y complejidad, requiere muchas actividades coordinadas y el empleo de una diversidad de herramientas y modelos. La metodología del desarrollo de sistemas es una forma estándar de organizar y coordinar estas actividades.

El análisis del sistema llega a la raíz del problema o la necesidad y define los requerimientos de los usuarios. Con frecuencia, lo que los usuarios creen que necesitan o lo que parece ser el problema al principio, resulta ser algo totalmente diferente después de realizar un análisis profundo. Cuando el analista se reúne con los usuarios surgen nuevos y en ocasiones diferentes requerimientos que al principio no eran evidentes necesariamente.

Después de iniciar un proyecto de sistemas, se debe definir su alcance y desarrollar un enfoque profundo en los requerimientos de los usuarios. El documento que resulta de este análisis preliminar es el reporte de la propuesta para realizar el análisis del sistema. Este documento demuestra que se ha llegado a un acuerdo entre las ideas del analista de sistemas y las de los usuarios.

Razones para iniciar el análisis del sistema.

1. Mejora de los sistemas de información estratégica.

Idealmente, un nuevo trabajo en sistemas se inicia para desarrollar un sistema de información estratégica que no sólo realice el procesamiento de transacciones y otras operaciones, sino que también se ajuste a la estrategia de la organización y de apoyo a las metas de la misma. La mejora de sistemas de información estratégica trae consigo una mayor productividad, una mejor diferenciación de productos y servicios, una mejora en el desempeño gerencial, reducción de costos y reportes más rápidos y más completos. El indicador de la mejora de sistemas de información estratégica es el plan de sistemas. El principal componente del plan de sistemas es una cartera de proyectos de sistemas prioritarios que hay que desarrollar.

2. Nuevo requerimiento.

Un nuevo requerimiento o un nuevo reglamento significa que debe hacerse un trabajo en sistemas para satisfacer dicho requerimiento. Como ejemplo podemos citar nuevas leyes de impuestos, nuevos procedimientos contables y nuevas cláusulas en la nómina.

3. Aplicación de una nueva idea o tecnología.

El analista puede realizar un trabajo en sistemas para determinar si una nueva forma de hacer las cosas o un nuevo componente tecnológico pueden mejorar el rendimiento

o reducir los costos. Por ejemplo, el analista podría investigar el empleo de códigos de barras para el control de inventarios.

4. Solución y mantenimiento de problemas no planeados.

En las empresas que no hacen uso de una estrategia de planeación del sistema, el trabajo en sistemas comienza con una necesidad por cubrir, una falla que requiere corrección o un problema que necesita solución. Por lo general, con estas tareas está relacionado un alto grado de urgencia debido a que no se desarrollan planes de sistemas de información que pudieran ayudar a anticipar y coordinar los proyectos de sistemas. En dichos ambientes, el mantenimiento consume una gran parte del presupuesto de sistemas a medida que se retrasan más y más los nuevos proyectos.

Definición del alcance del análisis del sistema.

Las actividades y eventos que comprenden el análisis del sistema se dirigen en su mayor parte a responder a la pregunta ¿qué va a incluir el sistema? Al responder esta pregunta general el analista debe plantear muchas preguntas específicas como:

¿Qué información se necesita?, ¿quién la necesita?, ¿cuándo?, ¿dónde?, ¿en qué forma?, ¿en donde se origina?, y ¿cómo puede obtenerse?

Además, el alcance del análisis del sistema puede variar ampliamente en términos de duración, complejidad y gastos. En consecuencia, en ocasiones el alcance se debe definir en forma un poco arbitraria para satisfacer restricciones como tiempo y costo.

En la práctica, con frecuencia un analista de sistemas fracasa en el logro de los objetivos o los alcanza con una gran pérdida de tiempo y dinero. Sin embargo se debe entender que la presencia de objetivos limitantes (o restricciones) sobre el alcance del análisis, limita las soluciones o recomendaciones potenciales que resultan del análisis. Como regla, la definición inicial del propósito y el alcance, así como cualquier objetivo y restricción, están sujetos a una redefinición en una fecha posterior, basados en los hallazgos del análisis.

Enfoque del análisis del sistema.

Uno de los errores más perjudiciales que se presentan en el análisis del sistema es pensar en términos computacionales y hacer énfasis primeramente en el componente estructural de la tecnología.

El enfoque primario del análisis del sistema debe estar en las operaciones de la empresa, los requerimientos de los usuarios y los componentes de la entrada, la salida, la base de datos y los controles y no en las computadoras, ni en los discos, ni en las telecomunicaciones, ni en el software.

El objetivo tanto de los usuarios como de los analistas de sistemas durante el análisis del sistema es llegar a un acuerdo de ideas para establecer lo que realmente se necesita para realizar el trabajo y lo que el sistema les puede proporcionar.

Conclusión sobre el análisis del sistema.

Una vez que el analista de sistemas completa las entrevistas iniciales y determina que deberá realizarse el análisis del sistema, se deben comunicar formalmente al solicitante y a la propia gerencia del analista de sistemas el entendimiento de lo que debe realizarse y el enfoque general hacia esa meta. Esta comunicación se denomina reporte de la propuesta para realizar el análisis del sistema. Proporciona un punto de verificación en el que el solicitante puede evaluar si el analista entiende o no claramente lo que se desea, y le da a la gerencia del analista la oportunidad de evaluar el enfoque y la cantidad de recursos que se van a emplear durante el análisis.

El reporte deberá facilitar una comprensión inicial profunda, así como proporcionar puntos de referencia a los que se pueda tener acceso para reportar periódicamente el desempeño real del análisis. Deberá incluir lo siguiente:

1. Una definición clara y concisa de las razones para realizar el análisis.
2. Un planteamiento específico referente a los requerimientos del desempeño del sistema propuesto.
3. Una definición del alcance del análisis.
4. Una identificación de los hechos que probablemente necesiten recopilarse.
5. Una identificación de las fuentes potenciales donde pueden obtenerse los hechos.
6. Un programa que indique los eventos principales del análisis.

2.2 PLANEACION DEL SISTEMA.

La ingeniería de software a diferencia de otras ingenierías, carece de leyes físicas para la programación, además de ser un producto intangible y tener una parte oculta en las interfaces de sus módulos de programación. En el campo de la ingeniería de software los programas son intangibles (físicamente hablando), puesto que no tienen masa, volumen o color, esto es, carecen de propiedades físicas. Su intangibilidad y falta de propiedades físicas limita los lineamientos para realizar el diseño y la instrumentación de un producto de programación. Es por esto que la ingeniería de software se auxilia de técnicas de estimación para manejar el diseño de algún programa.

"La ingeniería de software se define como la disciplina tecnológica preocupada de la producción sistemática y mantenimiento de los productos de software que son desarrollados y modificados en tiempo y dentro de un presupuesto definido".[3] La diferencia entre esta ingeniería y la programación tradicional, radica en las técnicas que se utilizan para el diseño, la instrumentación, la validación y el mantenimiento de los productos de programación; estás técnicas ayudan a mantener dentro del rango de tiempo y presupuesto establecidos al proyecto. El personal encargado del proyecto, se ocupa en aspectos de análisis, diseño, verificación y prueba de los programas, además de la documentación, el mantenimiento y la administración del proyecto.

Por lo antes mencionado, la planeación es una de las etapas más importantes en la ingeniería de software, en la cual, se combinan dos actividades: la investigación y la estimación. La primera permite definir el alcance del sistema. Aquí se utilizan las especificaciones del sistema, como condiciones que se requieren para el desarrollo del software, donde cada función del software puede ser descrita y determinada; la segunda actividad es la estimación, la cual es necesaria para poder aproximar costos y tiempos del proyecto.

2.2.1 EL ALCANCE DEL SISTEMA.

Dentro de la planeación, la primera actividad a realizar es determinar los alcances del sistema. Aquí se definen las necesidades y limitaciones del usuario, éstas deben estar bien definidas y explícitamente establecidas; estas definiciones no pueden quedar de forma ambigua, ya que esto causaría una mala interpretación por parte de los desarrolladores, es por esto, que dentro de la planeación del proyecto, se debe diseñar tomando en consideración, tanto a los desarrolladores como a los usuarios.

[3] Fairley Richard; Ingeniería de Software; McGraw Hill, 1988, México, pág. 5.

Es importante identificar a los clientes externos y sus necesidades, además se tienen que llevar a cabo estudios de factibilidad, así como la supervisión de cómo se desarrolla el producto desde su inicio hasta que termina.

Se considera que un factor importante en la calidad de un producto es su utilidad, esto es, que el producto resultante cumpla con las necesidades del usuario, por esta razón se requiere de una buena planeación para alcanzar un producto de mejor calidad en menos tiempo y bajo costo. El contar con una buena planeación lleva a un producto más confiable, esto se observa en la capacidad del programa para el desempeño de una función específica bajo ciertas condiciones durante un tiempo específico.

Un aspecto importante en el desarrollo de un sistema, es que todas las actividades que se establecieron, deben generar productos finales dentro del tiempo programado y el costo estimado.

Es importante que los criterios para la aceptación del sistema se definan durante la fase de planeación. La falta de claridad en estos criterios puede producir graves malentendidos entre el cliente y el encargado del desarrollo.

Para planear la forma de alcanzar cada logro dentro del tiempo programado, es necesario responder a las siguientes preguntas:

- ¿Cuántas metas son apropiadas?.
- ¿Cuántos recursos son necesarios para alcanzar cada logro?.
- ¿Quién será el responsable de alcanzarlos?.
- ¿Qué se debe cumplir para obtener cada logro?.
- ¿Qué técnicas de estimación se utilizarán?.
- ¿Qué precisión es aceptable?.
- ¿Qué modelo de ciclo de vida se utilizará?.
- ¿Con qué funciones de control se implementará?.
- ¿Qué estructura organizacional se seguirá?.
- ¿Quiénes son los responsables de cada proceso?.
- ¿Qué herramientas se emplearán para el desarrollo?.

El producto se entiende mejor conforme se realizan las etapas de análisis, diseño e implementación; sin embargo, el desarrollo del proyecto no debe depender de la disponibilidad de suficiente información para iniciar una planeación preliminar. La decisión de llevar a cabo el desarrollo del proyecto se basa generalmente en el resultado de un estudio de factibilidad. Durante la etapa de la planeación es adecuado:

1. Definir un modelo de ciclo de vida, así como una estructura organizacional para el desarrollo del proyecto.
2. Planear las actividades para el control de calidad y la validación.
3. Determinar en cada etapa las herramientas, las técnicas y la notación que se van a utilizar.
4. Establecer estimaciones preliminares de costos.

5. Establecer un programa preliminar para el desarrollo.
6. Establecer un estimado preliminar de la plantilla de personal.
7. Realizar un estimado preliminar de recursos de cómputo que son necesarios para operar y mantener el sistema.
8. Preparar un glosario de términos utilizados.
9. Identificar fuentes de información y referirse a ellas durante el desarrollo del proyecto.

2.2.2 RECURSOS.

Entre las técnicas necesarias para administrar un proyecto se cuentan las siguientes:

- Las que sirven para organizar y dar seguimiento al proyecto.
- Para estimar costos y establecer políticas de asignación de recursos.
- Para establecer el control del presupuesto.
- Determinar el avance del proyecto.
- Reasignar recursos y ajustar el calendario de trabajo.
- Para establecer procedimientos de control de calidad.
- Para proporcionar el mantenimiento.

Estas técnicas propician la comunicación entre los integrantes del equipo de desarrollo, ayudan a establecer comunicación con los clientes y aseguran el cumplimiento de los términos legales del proyecto.

El objetivo de la planeación del proyecto, es el proporcionar una área de trabajo al director del proyecto, para realizar estimaciones de recursos, costos y metodología de trabajo.

Al inicio de un proyecto, el director realiza estimaciones de tiempo sin tener un límite definido, estas estimaciones se actualizan conforme avanza el proyecto.

La falta de planeación en los proyectos, es la causa principal de retraso en la programación de las actividades, en que el costo se incremente, en la falta de calidad y altos costos de mantenimiento. En la práctica es muy difícil realizar una planeación, esto se debe a que no se conoce la información precisa sobre las metas del proyecto, las necesidades del cliente y las restricciones del proyecto. Por esta razón, uno de los principales propósitos de la planeación es definir los objetivos, las necesidades y las restricciones del problema.

Para realizar las estimaciones es indispensable primero, definir los recursos con los que se cuenta, tanto humanos como de hardware. Algunas de las características que se toman en cuenta son: la descripción del recurso, informe de disponibilidad, fecha cronológica en la que esté disponible el recurso y tiempos aplicados al recurso.

Recursos humanos.

Uno de los recursos indispensables en el desarrollo del sistema, es el contar con personal capacitado. Además, precisar que la gente es el elemento primordial para el desarrollo de un proyecto.

Por esto el encargado de la planeación necesita evaluar los siguientes puntos: el alcance del sistema, las técnicas necesarias para el desarrollo, la designación de

puestos del personal participante en el proyecto, en este caso se asigna cada participante en su especialidad.

Recursos de hardware.

Dentro de los recursos de hardware se consideran tres categorías: *"El sistema de desarrollo, la máquina objetivo y otros elementos de hardware del nuevo sistema"* [4].

El sistema de desarrollo es la computadora que se utilizará para el desarrollo e implementación del sistema; la máquina objetivo es la computadora donde se ejecuta el programa eventualmente, ésta puede ser la computadora donde se ejecutará el sistema permanentemente. Entre los elementos de hardware se consideran los periféricos, las redes de computadoras, así como los equipos de comunicación. Estos recursos son seleccionados por el encargado de realizar la planeación del proyecto.

Recursos de software.

En el desarrollo de un software es necesario utilizar otro software, éstos son los programas que nos ayudan a desarrollar el sistema. En la actualidad se cuenta con una variedad de herramientas para el desarrollo de sistemas. A continuación se mencionan algunos:

- Editores de texto.
- Comparadores.
- Manejadores de bases de datos.
- Lenguajes de consulta de bases de datos.
- Generadores de aplicaciones.
- Intérpretes.
- Compiladores.
- Depuradores.
- Editores de enlace.
- Lenguajes de alto nivel.
- Lenguajes de cuarta generación.
- Lenguajes de programación orientados a objetos.

Además, en el desarrollo del proyecto se debe considerar la posible reutilización de librerías existentes. En este caso se toman en cuenta los siguientes puntos:

- Si el software existente satisface las necesidades del usuario.
- Verificar si el software existente requiere de alguna modificación antes de ser integrado, en este caso hay que tener cuidado, ya que el costo de modificarlo puede ser mayor que el costo para desarrollar un software totalmente nuevo.

[4] Pressman Roger; Ingeniería de software, Un enfoque práctico; McGraw Hill, 1993, México, pág. 76.

Una vez definidos los recursos necesarios para el desarrollo, se procede a la planeación del proyecto. Una primera etapa es la definición de un modelo de ciclo de vida del producto. Este modelo incluye todas las actividades requeridas para la definición del proyecto, además del desarrollo, las pruebas, la operación y el mantenimiento del mismo. Es importante definir un modelo de ciclo de vida para el desarrollo del proyecto, esto permite clasificar y controlar las diferentes actividades que se necesitan para el desarrollo y mantenimiento del producto. Un modelo del ciclo de vida definido y aceptado por las partes involucradas en el proyecto, mejora la comunicación, permitiendo así una mejor administración y asignación de recursos, además de un mejor control de costos y mayor calidad del producto.

Modelo del ciclo de vida.

Es el conjunto de actividades sucesivas que los analistas, diseñadores y usuarios realizan; esta sucesión de actividades es en forma de cascada, cada una de las actividades requieren información de entrada, procesos y resultados, los cuales deben de estar bien definidos. Se considera el modelo de fases compuesto por las siguientes etapas: ingeniería de sistemas, análisis, diseño, codificación, pruebas y mantenimiento. En algunas ocasiones a este modelo se le denomina modelo en cascada, este se puede observar en la FIG. 2.1.

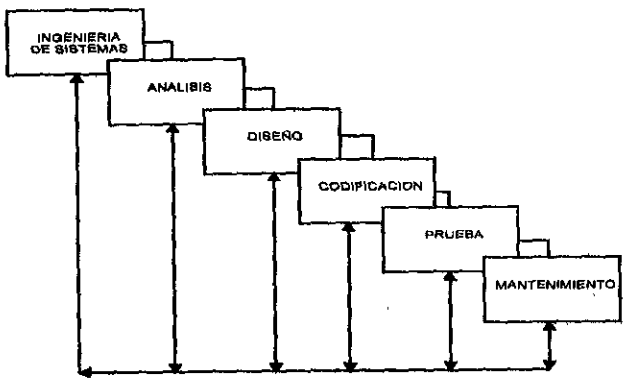


FIG. 2.1. Modelo de Ciclo de vida.

A continuación se describe brevemente cada una de las etapas del modelo de ciclo de vida.

Ingeniería de sistemas. En esta etapa se abarcan los requerimientos globales del sistema, además de establecer los requerimientos del mismo, aquí se realiza un análisis a grandes rasgos y se genera un diseño en el ámbito global.

Análisis. Es el proceso de recolectar los requerimientos de los usuarios, así como el entendimiento del dominio del problema y el entorno en que se encuentra éste. Las técnicas utilizadas para obtener este conocimiento son: Entrevistas con el cliente, la observación, el análisis de las tareas problemáticas y el desarrollo de reglas. Una característica importante por parte del encargado de la planeación es que debe tener habilidad para definir el problema, ya que tendrá que reunirse con diferentes representantes de las áreas usuarias, esto es, enfrentarse a diferentes puntos de vista, sesgos y prejuicios que influyen en la percepción del problema.

Diseño. Esta etapa se enfoca sobre tres atributos del sistema: La estructura de datos, la arquitectura del sistema y los detalles de los procedimientos. En esta etapa se detalla el análisis de los requerimientos para posteriormente traducirlos a un lenguaje de programación.

Codificación. En esta etapa el diseño se traduce en líneas de código de algún lenguaje de programación de computadora.

Pruebas. En esta etapa se realizan las pruebas al sistema, éstas se realizan cada vez que se va generando el código de algún módulo, aquí se verifica la lógica del programa para asegurar que todas las sentencias del programa se ejecuten correctamente, así como verificar que las funciones externas manejen los datos correctos y entreguen los datos necesarios.

Mantenimiento. La etapa de mantenimiento se realiza posteriormente a la entrega del sistema; en ésta se realizan adecuaciones a nuevos requerimientos de los usuarios o modificaciones solicitadas posteriormente a la entrega. En algunos casos se tendrán que corregir errores menores, que puedan resultar de cambios solicitados por el usuario.

Este modelo de ciclo de vida se retroalimenta dependiendo de la necesidad de regresar o avanzar a la siguiente etapa, en este modelo se indica que en cualquier etapa del desarrollo del sistema se puede regresar a la etapa requerida dependiendo de cómo se desarrolle el proyecto.

2.2.3 ESTIMACION DE COSTOS.

Actualmente en muchos sistemas de programación, el mayor porcentaje del costo de un sistema corresponde al software, una mala estimación del costo puede producir pérdidas para el equipo de desarrollo, lo difícil es que la estimación del costo del proyecto no es una labor fácil, ya que existen muchos elementos variables como son: el personal, el tiempo de desarrollo, las técnicas utilizadas o alguna política por parte de los usuarios.

Por esto dentro de la planeación, la estimación de costos es una parte importante en la que se debe tomar un cierto riesgo; existen algunos procedimientos sistematizados que proporcionan estimaciones con un grado de riesgo aceptable, esta estimación del costo debe ser aceptado por el usuario, además de que el producto nuevo de programación debe proporcionar mejores servicios y mayor información que el sistema anterior, usando menos tiempo y personal o proporcionar servicios e información que antes no se tenían o eran inaccesibles.

El realizar la estimación del costo de un producto de programación, se torna en una actividad muy difícil y errática para la ingeniería de software; la dificultad radica, en que es difícil hacer estimaciones exactas durante la fase de planeación de un proyecto, debido a la gran cantidad de factores desconocidos que se tienen en ese momento. Es por esto que durante la planeación se prepara un estudio preliminar del costo y se presenta una revisión de los requisitos del sistema. La estimación final se presenta durante la revisión preliminar del diseño del proyecto.

A continuación se enlistan algunos aspectos que influyen en el costo del software:

- Capacidad del programador. La capacidad del programador influye en el costo, ya que dependiendo de su habilidad la exigencia de sueldo es mayor.
- Complejidad del producto. Dependiendo de lo complejo que sea el problema a resolver será necesario invertir más recursos.
- Tamaño del programa. Un proyecto grande de programación es obviamente más caro en su desarrollo que uno pequeño.
- Tiempo disponible. El esfuerzo del proyecto se relaciona con el calendario de trabajo asignado para la terminación del proyecto, la mayoría de los desarrolladores están de acuerdo en que los proyectos de programación requieren más esfuerzo, si el tiempo de desarrollo se reduce o se incrementa más de su valor óptimo.
- Confiabilidad requerida. La confiabilidad del producto puede definirse como la probabilidad de que un programa desempeñe una función requerida bajo ciertas condiciones específicas, durante cierto período de tiempo. El nivel de confiabilidad deseado debe establecerse durante la fase de planeación, además se considera el costo de las posibles fallas del programa; en algunos casos, las fallas pueden causar al usuario pequeñas inconveniencias en el desempeño de sus labores.
- Nivel tecnológico. El nivel de tecnología utilizado en un proyecto se refleja en los elementos que son utilizados como son: el lenguaje de programación utilizado, el

hardware empleado, las prácticas y las herramientas de programación utilizadas Por ejemplo, los lenguajes modernos de programación brindan características adicionales para mejorar la productividad y confiabilidad de la programación, lo cual influye directamente en los costos del proyecto; un ejemplo de las características que proporcionan este tipo de lenguajes es la verificación de los tipos de datos, las estructuras definidas para la abstracción de datos y la compilación por separado de cada elemento. La facilidad de acceso a la programación influyen en la productividad del programador y esto en el costo del proyecto. Como parte del desarrollo tecnológico es necesario tener gente preparada para el desarrollo del proyecto con conocimiento de dicha tecnología, ya que la productividad se afecta si los programadores tienen que aprender a usar un nuevo ambiente de programación como parte del proceso para el desarrollo del producto. Dentro de la tecnología existen tres categorías para los productos de programación: Programas de aplicación, en los que se incluyen procesamientos de datos y programas científicos; programas de apoyo, como compiladores, ligadores y sistemas de inventarios y programas de sistema, como sistemas de base de datos, sistemas operativos y sistemas en tiempo real.

Técnicas de estimación de costos del software.

En la mayoría de los desarrollos, la estimación de costos se basa en las experiencias pasadas. Los datos históricos se usan para identificar factores de costo y determinar la importancia de los diversos aspectos que intervienen en el desarrollo del proyecto.

El juicio experto.

Es la técnica más utilizada para la estimación de costos, además es una técnica de tipo jerárquico hacia abajo. Se basa en la experiencia, en el conocimiento anterior y en el sentido comercial de uno o más individuos involucrados en el proyecto. La mayor ventaja del juicio experto es la experiencia, la cual puede llegar a ser su debilidad; el experto puede confiarse de que el proyecto sea similar al anterior; pero puede suceder que haya olvidado algunos factores que ocasionan que el sistema nuevo sea significativamente diferente, para compensar tales factores, los grupos de expertos tratan de llegar a un consenso en el costo estimado del proyecto.

Estructuras de división de trabajo.

La estructura de división de trabajo, es un método de tipo jerárquico donde se establecen las diferentes partes de un sistema. Un organigrama de este tipo refleja una jerarquía de productos o de procesos. Las ventajas de esta técnica son la identificación y el conteo de los diversos procesos y factores que intervienen en el sistema.

Estimación del nivel de contratación.

La cantidad de personal requerido a través del desarrollo de un proyecto no es constante; por lo regular, la planeación y el análisis lo realiza un grupo de individuos; el diseño del proyecto un grupo con más integrantes y el diseño detallado lo realiza un grupo grande de personas. Es por esto, que varía la cantidad de personas que intervienen en el desarrollo del proyecto.

Estimación de costos de mantenimiento.

El mantenimiento al software necesita del 40 al 60% y en algunos hasta 90% del esfuerzo total durante el ciclo de vida del producto; entre estas actividades se encuentran, el agregar mejoras al producto, adaptar el producto a nuevos ambientes y corregir problemas de programación, los cuales deben ser menores. La estimación de los costos es una de las tareas más difíciles y erráticas al desarrollar un sistema. Las técnicas de estimación de costos se basan en los datos históricos de cada organización de acuerdo al desempeño de proyectos anteriores, así, una estimación de costos es tan buena como nuestra capacidad de extrapolar al futuro las experiencias pasadas.

La estimación del costo de programación se basa en datos históricos, por lo que estos datos deben recolectarse de los proyectos pasados para poder estimar el esfuerzo y el calendario de proyectos futuros. La estimación o determinación previa de costos, específicamente de los costos de realización o manufactura, es un aspecto esencial de un estudio económico.

Durante el desarrollo del sistema el líder de proyecto, debe tener los elementos suficientes para tomar una decisión relacionada al proyecto. En el caso del estudio de viabilidad, dependerá del resultado de la valuación económica, que se lleve a cabo una etapa preliminar del proyecto y, posteriormente, la etapa del proyecto detallado. Por lo tanto, es importante poder calcular con la mayor exactitud posible el costo que representaría la realización del proyecto.

Se considera que los datos contables de experiencias pasadas son un elemento útil para estimar los costos futuros. Los datos contables deben ser complementados, por datos estadísticos, económicos y de ingeniería, además de un análisis profundo para obtener una información básica que ayude a calcular los costos futuros.

2.2.4 HERRAMIENTAS DE CONTROL DE AVANCE.

En toda labor es necesario no sólo mantener un control sobre lo que se está haciendo, sino establecer desde antes lo que se va a hacer. En otras palabras existe la necesidad de definir los siguientes factores:

¿Qué meta se quiere alcanzar?.

¿Con qué elementos se cuenta?.

¿Qué limitaciones existen para poder alcanzar la meta?.

Estas preguntas son importantes y requieren de atención para ordenarlas de una forma efectiva. La organización de estos factores y su control es lo que se llama la administración de un proyecto.

Para la administración de un proyecto existen diferentes herramientas para su control. Existen técnicas que se le conocen como: Método de Ruta Crítica (MRC) y PERT (siglas en inglés para Evaluación de Programas y Técnicas de Revisión). Estos dos métodos se emplean con frecuencia en proyectos medianos y en la gran mayoría de proyectos grandes. A continuación se muestra en qué consisten estos métodos de administración y cuáles son las diferencias, además se explicarán algunos de los elementos de dichas técnicas.

El método de Ruta Crítica y el PERT son herramientas poderosas para la administración de un proyecto, además son útiles para su planeación, ya que al no existir antecedentes, es necesario organizar el trabajo de tal forma que no tome ni más tiempo, ni más dinero de lo estrictamente necesario.

Ruta crítica y PERT.

Estos dos métodos de planeación consisten en un diagrama o red de actividades que muestra la dependencia de cada actividad, en la que se tiene que desarrollar en función del tiempo, del costo, o de los recursos utilizados o una combinación de estos elementos.

Por lo general se emplea el MRC en los proyectos en donde se ha tenido más experiencia y los tiempos de cada actividad ya están definidos. El método PERT se emplea en aquellos proyectos en donde no se conoce con certeza la duración de cada actividad, y por lo tanto, hay que hacer una estimación con cierto nivel de seguridad.

Las ventajas de los dos métodos de planeación son múltiples. A continuación se enumeran algunas de estas ventajas:

- Se logra una planeación más lógica de las actividades a desarrollar.
- La planeación se lleva a cabo con un plazo de tiempo mayor.

- Se simplifica la coordinación de un proyecto entre los distintos elementos que lo integran.
- Se ahorra dinero debido a que no existen actividades inesperadas.
- Se pueden comunicar las ideas de una manera gráfica más concisa.

Método de Ruta Crítica.

Para entender este método es necesario definir algunos términos antes de desarrollar una red y utilizar ruta crítica.

Actividad. Es la parte individual de trabajo que hay que efectuar en un proyecto. Es un trabajo único con una duración determinada. Por ejemplo: instalar un transformador, dos días. Todas las actividades deben estar conectadas entre sí y dependen de una o más actividades. La actividad se representa por medio de una flecha.

Evento. Es el punto de partida de una actividad y sucede sólo cuando todas las actividades que le preceden han llegado a su término.

Red. Es el conjunto de actividades y eventos que reflejan, de una manera fiel el proyecto.

Actividad virtual. Es una actividad que dura un tiempo igual a cero. Como cada actividad, debe estar precedida por un evento y debe concluir con otro evento. Cuando es necesario usar una actividad virtual, se indica mediante una flecha punteada.

Tiempo libre u holgura. Es el tiempo que existe entre el final de una actividad y el principio de la siguiente. Así, una actividad que dura cinco días y que tiene un tiempo libre de tres días, puede prolongarse más del tiempo establecido de antemano, sin alterar la fecha de terminación del proyecto.

La técnica de ruta crítica es la secuencia de actividades y de eventos en donde el tiempo libre es mínimo. La duración de la Ruta Crítica es el tiempo mínimo requerido para terminar un proyecto. Este tiempo es valioso, pues permite determinar con certeza la fecha de terminación del proyecto y por lo tanto es la base para programar las erogaciones para llevarlo a cabo.

Esta es la ventaja de una Ruta Crítica, el poder controlar las actividades sabiendo cuáles deben terminarse en fechas específicas y cuáles pueden retrasarse sin afectar el tiempo de entrega del proyecto. Para ver el avance de estos métodos se puede utilizar un diagrama visual que permita valuar el progreso de un proyecto, a este diagrama se le conoce como diagrama de Gantt, FIG. 2.2.

	1a	2a	3a	4a	5a	6a
ANALISIS						
DEFINICION DE OBJETIVOS						
ELABORACION DIAGRAMAS						
CARTAS DE ESTRUCTURA						
DICCIONARIO DE DATOS						
CODIFICACION						
.						
.						
.						
LIBERACION						

FIG. 2.2. Diagrama de Gantt.

Método del PERT.

La diferencia de PERT con MRC, es que el tiempo de cada actividad es variable y se determina de una forma probabilística. Para estimar el tiempo de cada actividad atribuye tres estimaciones de tiempo diferentes:

Tiempo pesimista: p
Tiempo más probable: m
Tiempo optimista: o

El tiempo esperado (e), será el que considere el encargado de la planificación y se obtiene realizando el siguiente cálculo:

$$e = \frac{o + 4m + p}{6}$$

Al igual que MRC utiliza diagramas de red para su representación gráfica.

2.3 ANALISIS Y ESPECIFICACION ESTRUCTURADA.

2.3.1 EL DIAGRAMA DE FLUJO DE DATOS (DFD).

Definiciones y utilidad del diagrama de flujo de datos (DFD).

El diagrama de flujo de datos es un modelo lógico que se utiliza en el análisis y diseño de sistemas de información, es además una herramienta gráfica de modelado, éste, describe los flujos de información y los procesos o actividades que cambian o transforman los datos en un sistema. El diagrama de flujo de datos puede mostrar los procedimientos que debe seguir el personal de una organización para llevar a cabo ciertas tareas y así convertir los datos en información.

El diagrama de flujo de datos es una herramienta que permite visualizar un sistema como una red de procesos funcionales, conectados entre sí por líneas y círculos o rectángulos; se le conoce también como carta de burbujas, DFD, diagrama de burbujas, modelo de proceso, diagrama de flujo de trabajo o modelo de función. Los diagramas de flujo de datos son gráficas dirigidas en donde los nodos especifican las actividades de proceso y los arcos la transferencia de datos entre nodos de proceso.

El diagrama de flujo de datos es una de las herramientas más comúnmente usadas, sobre todo por sistemas operacionales en los cuales las funciones del sistema son de gran importancia y son más complejas que los datos que éste maneja. También, *"los DFD no sólo se pueden utilizar para modelar sistemas de proceso de información, sino también como manera de modelar organizaciones enteras; es decir, como una herramienta para la planeación estratégica y de negocios."* [5]

Como cualquier otro diagrama de flujo, éste puede ser utilizado a cualquier nivel de abstracción. Un diagrama de flujo de datos puede representar el flujo de datos entre actividades individuales o entre bloques de actividades dentro de un proceso; flujos de datos secuenciales, flujo de datos entre procesos concurrentes o flujo de datos entre sistemas de cómputo distribuidos, donde cada nodo representa una unidad de proceso geográficamente separada. Distinto a otros diagramas de flujo, las burbujas no indican la lógica de la decisión o las condiciones bajo las cuales varios nodos de proceso se activen.

Los diagramas de datos pueden expresarse utilizando una notación informal o por medio de símbolos especiales para denotar a los nodos de proceso, a los nodos de entrada, a los nodos de almacenamiento y a los nodos de salida. La anterior descripción se muestra en la FIG. 2.3.

[5] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 158.

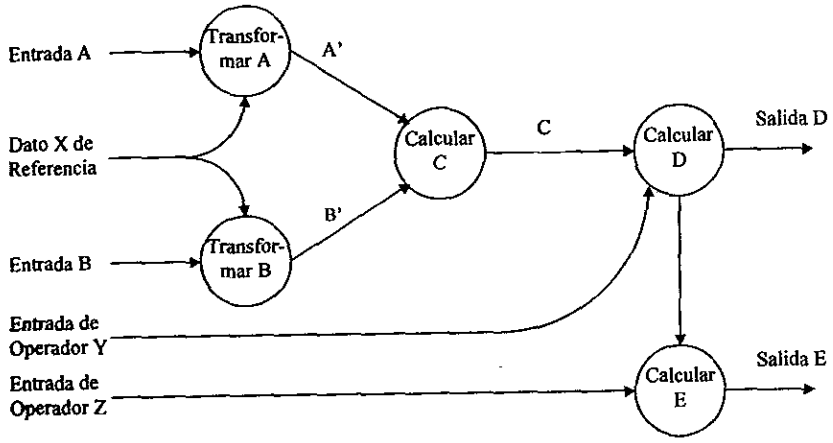


FIG. 2.3. Diagrama de flujo de datos.

Los DFD son un mecanismo excelente para la comunicación con el cliente durante el análisis de requerimientos; así mismo, son empleados para representar las especificaciones del diseño externo y del diseño interno en un alto nivel; en este último caso, las burbujas son valiosas para el establecimiento de las diversas convenciones de nombres para el sistema, los subsistemas, los archivos y las ligas de datos.

Dada una técnica de diagramación correcta es mucho más fácil describir actividades y procedimientos complejos mediante diagramas que mediante texto. Los dos tipos de diagramas de flujo de datos más populares son los de burbuja, que utilizan círculos para representar procesos y líneas curvas para el flujo de datos y los diagramas que emplean rectángulos para los procesos y líneas rectas para el flujo de datos.

Independientemente de la simbología usada en los diagramas de flujo de datos, el aspecto relevante es su aplicación (sin importar los símbolos empleados) para comunicar diferentes modelos a los usuarios a fin de lograr un mejor entendimiento entre ellos y el analista de sistemas. En la FIG. 2.4 podemos observar los diferentes símbolos que se emplean para los diagramas de flujo de datos, los cuales son sugeridos en su mayoría por Yourdon.

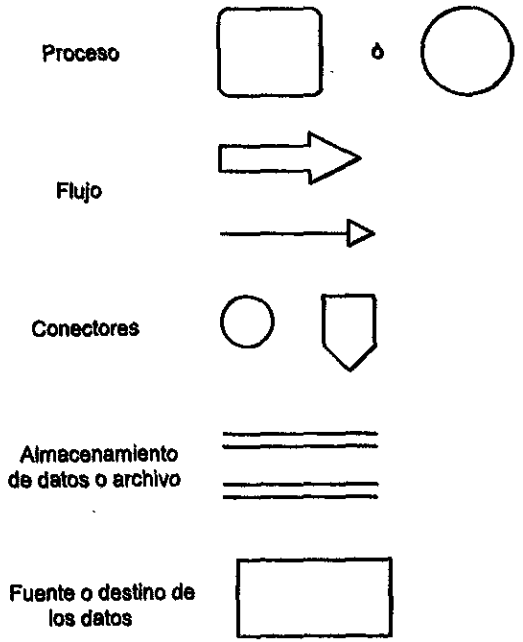


FIG. 2.4. Paleta de símbolos para los diagramas de flujo de datos.

Los símbolos de proceso se emplean para indicar aquellos lugares dentro del sistema en donde los flujos de datos que entran se procesan o transforman en flujos de datos que salen. El nombre del proceso generalmente se escribe dentro del componente.

También se pueden utilizar números para referencia o secuenciación. Los flujos de materiales se muestran mediante una flecha ancha. Los flujos de datos se muestran mediante flechas que marcan el movimiento de los datos a través del sistema. Un flujo de datos puede imaginarse como una tubería que transporta paquetes de datos desde su origen hasta su destino. Un rectángulo con extremos abiertos representa un archivo lógico en donde se agregan o donde se extraen datos.

El origen de los datos se denomina fuente, y el recipiente de los datos se denomina receptor o depósito.

Las fuentes y los receptores pueden ser una persona, una organización o incluso otro sistema. Dichas entidades externas se representan mediante rectángulos. En consecuencia, un DFD representa en esencia los límites del sistema, las interacciones externas, los procesos y el flujo de datos.

El DFD es un modelo lógico, y por lo tanto, no identifica discos, cintas, impresoras, computadoras o algún otro dispositivo físico. Los DFD se construyen en forma

descendente. Los detalles de más bajo nivel de los DFD son utilizados por los programadores para desarrollar el software que soporta las aplicaciones.

También como técnicas suplementarias, los usuarios y programadores utilizan para información adicional: diccionario de datos, lenguaje natural estructurado, árboles y tablas de decisión.

2.3.2 CARACTERISTICAS DEL DFD.

En el tema anterior se menciona la utilidad de los diagramas de flujo de datos; ésta, así como su simbología pueden considerarse como parte de las características de los mismos. Sin embargo la característica más importante de los diagramas de flujo de datos es que poseen cuatro componentes con los cuales nos es posible minimizar las posibilidades de construir DFD's confusos, incorrectos e inconsistentes.

Los componentes de los DFD's son:

1. El proceso
2. El flujo
3. El almacén
4. El terminador

1. El proceso.

"El primer componente del DFD se conoce como proceso. Los sinónimos comunes son: burbuja, función o transformación. El proceso muestra una parte del sistema que transforma entradas en salidas, es decir muestra cómo es que una o más entradas se transforman en salidas." [6]

El proceso se representa gráficamente como un círculo, algunos analistas prefieren usar un óvalo o un rectángulo con esquinas redondeadas y algunas veces se usa simplemente un rectángulo. Las diferencias entre estas tres formas son puramente estéticas, aunque es importante usar la misma forma de manera consistente para representar las funciones de un sistema.

La FIG. 2.5 nos muestra las diferentes representaciones de un proceso.

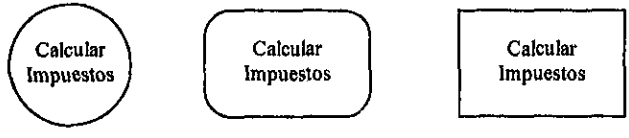


FIG. 2.5. Representación de un proceso. En círculo, rectángulo con esquinas redondeadas y rectángulo.

El proceso se nombra o describe con una sola palabra, frase u oración sencilla. Un buen nombre generalmente consiste en una frase verbo-objeto por ejemplo: CALCULAR IMPUESTOS.

[6] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 160.

En algunos casos el proceso *contendrá el nombre* de una persona o un grupo (por ejemplo un departamento o una división de una organización) o de una computadora o algún aparato mecánico. El proceso a veces describe *quién o qué* lo está efectuando, *más* que describir el proceso mismo.

2. El flujo.

Un flujo se representa *gráficamente* por medio de líneas que terminan en flecha que entra o sale de un proceso tal y como se muestra en la FIG. 2.6.

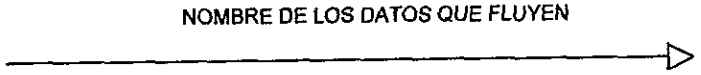


FIG. 2.6. Representación gráfica de un flujo.

El flujo se usa para describir el movimiento de bloques o paquetes de *información de una parte del sistema a otra.* Por ello los flujos representan datos en movimiento, mientras que los almacenes representan datos en reposo. En la mayoría de los sistemas que el analista *modele, "los flujos realmente representarán datos, es decir bits, caracteres, mensajes, números y los diversos tipos de información con los que las computadoras pueden tratar."*[7]

Los flujos tienen un nombre, dicho nombre representa el *significado del paquete de* datos que se mueve a lo largo del flujo. A veces es útil considerar varios flujos elementales en uno solo. El mismo contenido de datos en un flujo, puede tener distintos *significados en distintas partes del sistema.* La FIG. 2.7 muestra un ejemplo que describe esta situación.

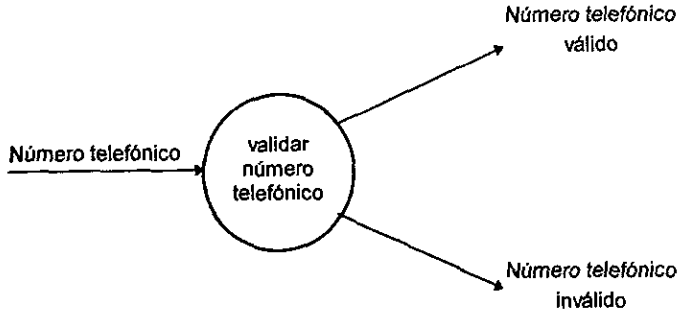


FIG. 2.7. Distintos significados del contenido de datos.

[7] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 162.

Aquí, el mismo fragmento de datos (Número telefónico) tiene distinto significado cuando viaja a lo largo de un flujo llamado Número telefónico que cuando viaja a lo largo del flujo Número telefónico válido. *"En el primer caso significa un número telefónico que pudiera ser o no válido; en el segundo caso, significa un número telefónico que, dentro del contexto de este sistema, se sabe que es válido. Otra forma de verlo es que el flujo número telefónico es como un ducto, lo suficientemente poco discriminador como para permitir el paso de números no válidos al igual que válidos; el flujo denominado Número telefónico válido es más estrecho y permite pasar datos más definidos."*[8]

Una cabeza de flecha en cualquier extremo indica la dirección del flujo; es decir; si los datos (o materiales) se están moviendo hacia adentro o hacia afuera de un proceso. Los datos que se mueven en dicho flujo viajarán ya sea a otro proceso (como entrada) o a un almacén o a un terminador. El flujo puede ser en ambas direcciones, como se muestra en el ejemplo de la FIG. 2.8.

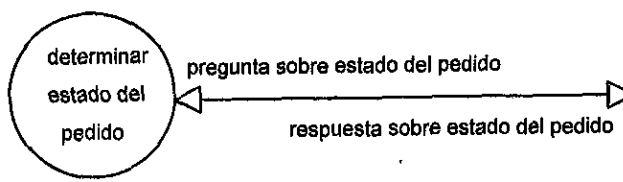


FIG. 2.8 Flujo en ambas direcciones.

Este flujo de dos direcciones es un diálogo, es decir un flujo con dos paquetes de datos (una pregunta y una respuesta) en el mismo flujo.

Los flujos de datos pueden divergir o converger en un DFD; conceptualmente esto es algo así como un río principal que se divide en varios más pequeños, o varios pequeños que se unen. Esto quiere decir que los flujos de datos pueden presentarse en diversas formas. En un DFD típico en el cual hay paquetes de datos que se mueven a través del sistema, en el caso de un flujo divergente, esto significa que se están enviando copias de un paquete de datos a diferentes partes del sistema, o bien en un paquete complejo de datos se está dividiendo en varios paquetes individuales más, cada uno de los cuales se está enviando a diferentes partes del sistema.

De manera inversa, en el caso de un flujo convergente, significa que varios paquetes elementales de datos se están uniendo para formar agregados más complejos de paquetes de datos.

La FIG. 2.9 y la FIG.2.10, nos muestran ejemplos de flujo divergente y flujo convergente respectivamente.

[8] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 163.

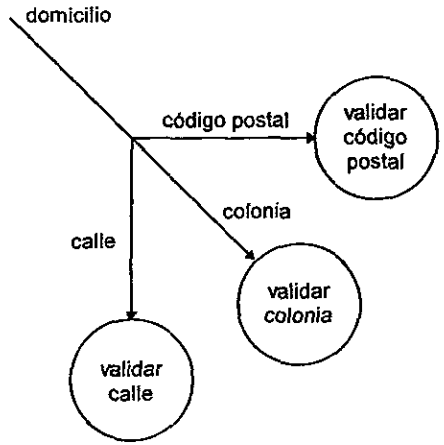


FIG. 2.9. Flujo divergente.

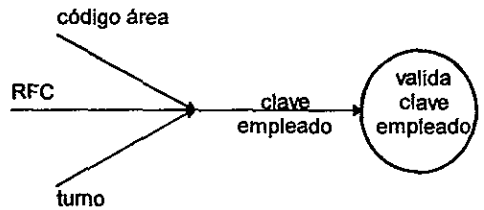


FIG. 2.10. Flujo convergente.

El flujo no responde a muchas dudas de procedimiento que se pudiera tener cuando se está viendo un DFD como pueden ser dudas acerca de la petición de entradas o de flujo de salidas.

Finalmente, hay que considerar una situación común donde se presentan múltiples flujos de entrada y de salida como se muestra en la FIG. 2.11.

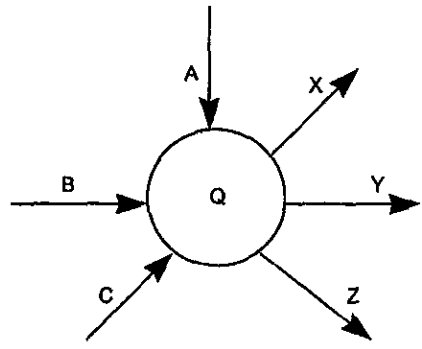


FIG. 2.11. Múltiples flujos de entrada y salida.

En este tipo de flujo, no se sabe en qué secuencia llegan los paquetes de datos ni en qué secuencia se generan los paquetes de salida. Es decir, no está determinado si el proceso Q requiere un paquete de los flujos A, B y C para producir exactamente un paquete de salida para los flujos X, Y y Z.

Casos como el anterior no son abordados por los DFD's ya que su uso principal no es determinar las condiciones para las cuales se genera un flujo, sino más bien ilustrar acerca del flujo de información existente en la realización de una tarea o procedimiento.

3. El almacén.

"El almacén se utiliza para modelar una colección de paquetes de datos en reposo." [9] Si nos apegamos a términos de procesos para computadoras, un almacén viene siendo un archivo o una base de datos. De modo característico el nombre que se utiliza para identificar al almacén es el plural del que se utiliza para los paquetes de datos que entran y salen del almacén por medio de flujos. Por ejemplo: si el paquete de datos es el pedido, el almacén se llamará pedidos. En la FIG. 2.12 se muestra como se representa un almacén.

PEDIDOS

FIG. 2.12. Representación gráfica de un almacén.

Los almacenes o bases de datos se conectan por medio de flujos a los procesos. Así, el contexto en el que se muestra un almacén en un DFD es uno de los siguientes o ambos:

[9] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 168.

- Un flujo desde un almacén.
- Un flujo hacia un almacén.

Normalmente se interpreta un flujo que procede de un sistema como una lectura o un acceso a la información del almacén. Esto significa que:

- Se recupera del almacén un solo paquete de datos.
- Se ha recuperado más de un paquete del almacén.
- Se tiene una porción de un paquete del almacén.
- Se tienen porciones de más de un paquete del almacén.

Como se notó cuando se examinaron los flujos que entraban y salían de un proceso, se tienen varias dudas de tipo procedimiento cuando se examinan los flujos que entran y salen de un almacén, una de ellas es si el flujo representa un solo paquete, muchos, porciones de uno o porciones de diversos paquetes. En algunos casos, esto se puede detectar simplemente viendo la etiqueta del flujo; si el flujo no está etiquetado, significa que todo el paquete de información se está recuperando; si la etiqueta del flujo es la misma que la del almacén significa que se recupera todo un paquete; si la etiqueta del flujo es diferente del nombre del almacén, entonces se está recuperando uno o más componentes de uno o más paquetes. De hecho para conocer todo lo deseado acerca del flujo que emana del almacén se tienen que examinar los detalles, como la especificación del proceso, al cual se conecta el flujo.

El almacén es pasivo, y los datos no viajarán a lo largo del flujo a menos que el proceso lo solicite explícitamente. Un programador puede referirse a esto como una lectura no destructiva o, en otras palabras, del almacén se recupera una copia del paquete y el almacén mantiene su condición original. Un flujo hacia un almacén habitualmente se describe como una escritura, una actualización o posiblemente una eliminación.

4. El terminador.

El terminador se representa gráficamente con un rectángulo, como se muestra en la FIG. 2.13.

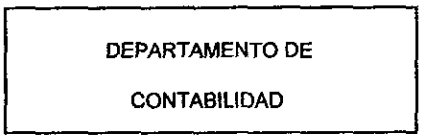


FIG. 2.13. Representación gráfica de un terminador.

Los terminadores representan entidades externas con las cuales el sistema se comunica. Comúnmente un terminador es una persona o un grupo, una organización externa o un departamento que esté dentro de la misma compañía, pero fuera de control del sistema que se está modelando. En algunos casos un terminador puede ser otro sistema, como algún otro sistema computacional con el cual se comunique.

Puede ser fácil identificar los terminadores en el sistema que se está modelando. A veces el terminador es el usuario. En otros casos, el usuario se considera parte del sistema y ayudará a identificar los terminadores relevantes. Existen tres cosas que se deben tomar en cuenta acerca de los terminadores:

1. Son externos al sistema que se está modelando; los flujos que conectan los terminadores a diversos procesos en el sistema, representan la interfaz entre él y el mundo externo.
2. Como consecuencia, es evidente que ni el analista, ni el diseñador del sistema están en posibilidades de cambiar los contenidos de un terminador o la manera en que trabaja, es decir el terminador está fuera del dominio del cambio. Lo que esto significa es que el analista está modelando un sistema con la intención de permitir una considerable flexibilidad y libertad al diseñador para elegir la mejor implantación posible.
3. Las relaciones que existan entre los terminadores no se muestran en el modelo del diagrama de flujo de datos. Pudieran existir de hecho diversas relaciones, pero, por definición, no son parte del sistema que se analiza.

De manera inversa, si existen relaciones entre los terminadores y si es esencial para el analista modelarlos para poder documentar los requerimientos del sistema, entonces, por definición, los terminadores son en realidad parte del sistema y deberían modelarse como procesos.

Todo buen diagrama de flujo de datos debe reflejar estos cuatro componentes que lo forman si es que existen. Con los cuatro componentes de los DFD's pueden construirse modelos de sistemas, sin embargo existen algunas reglas adicionales que se requieren para utilizar los diagramas de flujo de datos.

Estas reglas ayudan a evitar DFD's erróneos, ya sea incompletos o lógicamente inconsistentes. Algunas de estas reglas tienen la finalidad de ayudar a hacer DFD's más gratos a la vista, y que, por tanto, existan más probabilidades de que lo lea con cuidado el usuario.

"Las reglas son las siguientes:

1. *Escoger nombres con significado para los procesos, flujos, almacenes y terminadores.*
2. *Numerar los procesos.*
3. *Redibujar el DFD tantas veces como sea necesario estéticamente.*
4. *Evitar DFD's excesivamente complejos.*
5. *Asegurarse de que el DFD sea internamente consistente y que también lo sea con cualquier otro diagrama de flujo de datos relacionado con él."* [10]

[10] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 177.

Los nombres de procesos, *flujos*, *almacenes* y *terminadores* deben provenir de un vocabulario que tenga algún significado para el usuario. Esto sucederá de manera muy natural si el DFD se dibuja como resultado de una serie de entrevistas con los usuarios y si el analista tiene algún entendimiento mínimo de la materia de aplicación.

2.3.3 EL DICCIONARIO DE DATOS.

El diccionario de datos es también una herramienta de datos importante, aunque no tenga la presentación de los DFD o los diagramas entidad-relación. Sin el diccionario de datos, el modelo de los requerimientos del usuario, no puede considerarse completo.

"El diccionario de datos es un listado organizado de todos los datos pertinentes del sistema, con definiciones precisas y rigurosas para que tanto el usuario como el analista tengan un entendimiento común de todas las entradas, salidas, componentes de bases de datos y cálculos intermedios."[111]

El diccionario de datos define los datos haciendo lo siguiente:

- Describe el significado de los flujos y almacenes que se muestran en los DFD.
- Describe la composición de agregados de paquetes de información que se mueven a lo largo de los flujos, es decir paquetes complejos (como el domicilio de un cliente) que puedan descomponerse en unidades más elementales (como ciudad, estado, etc.).
- Describen la composición de los paquetes de datos en los almacenes.
- Especifica los valores y unidades relevantes de piezas elementales de información en los flujos de datos y en los almacenes de datos.
- Describe los detalles de las relaciones entre almacenes que se enfatizan en un diagrama de entidad-relación.

En la mayoría de los sistemas reales con los que se trabaja, los paquetes o elementos de datos, serán los suficientemente complejos como para que se necesite describir en términos de otras cosas. Los elementos complejos de datos se definen en términos de elementos más sencillos y los sencillos en términos de los valores y unidades legítimos que puedan asumir.

[111] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 212.

Notación del diccionario de datos.

Existen muchos esquemas de notación comunes usados por el analista de sistemas.

"El que a continuación se muestra es de los más comunes y usa símbolos sencillos:

=	*está compuesto de.*
+	*y.*
()	*optativo (puede estar presente o ausente).*
{}	*iteración.*
[]	*seleccionar una de varias alternativas.*
**	*comentario.*
@	*identificador (campo clave) para un almacén.*
|	*separa opciones alternativas en la construcción."* [12]

Como puede apreciarse, los símbolos parecen algo matemáticos y pudiera pensarse que son demasiado complicados de entender. Sin embargo la notación es fácil de leer.

Definiciones.

La definición de un dato empieza con el signo "=". En este contexto el "=" se lee: "se define como", o "se compone de", o simplemente "significa".

Para definir por completo un dato, la definición debe incluír lo siguiente:

- El significado del dato dentro del contexto dentro de la aplicación de este usuario. Por lo común se ofrece como comentario utilizando la notación "**".
- La composición del dato, si se compone de partes elementales con significado.
- Los valores que puede tomar el dato, si es un dato elemental que no puede descomponerse más.

Además de las unidades y la escala, podría requerirse la precisión de la medición del dato. En muchas aplicaciones científicas y de ingeniería es importante indicar el número de dígitos significativos en el valor de los datos.

Elementos de datos básicos.

Las partes elementales de los datos son aquellas para las cuales ya no existe una descomposición con significado dentro del contexto del ambiente del usuario. Esto usualmente es una cuestión de aplicación y es algo que se debe explorar cuidadosamente con el usuario.

[12] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 214.

Cuando se han identificado los datos elementales, deben introducirse al diccionario de datos. Como se indicó anteriormente, el diccionario de datos debe proporcionar una breve narrativa, encerrada entre caracteres "**" que describa el significado del término en el contexto del usuario. Desde luego, habrá términos que se definan solos, es decir cuyo significado es universal para todos los sistemas de información, o donde el analista pudiera estar de acuerdo en que no se necesita aclarar más.

Datos opcionales.

Un dato opcional, como su nombre lo indica es aquel que puede estar o no presente en un dato compuesto, por ejemplo:

- El nombre de un cliente pudiera no incluir un segundo nombre
- El domicilio de un cliente pudiera incluir o no información secundaria como el número de departamento.
- El pedido de un cliente pudiera contener el domicilio al que se tiene que mandar la cuenta, el domicilio al que hay que hacer envío, o ambos.

Todas las situaciones de este tipo que pudieran presentarse deben verificarse con cuidado con el usuario y deben documentarse en el diccionario de datos.

Iteración.

La notación de iteración se usa para indicar la ocurrencia repetida de un componente de un dato. Se lee como "cero o más ocurrencias de". Así, la notación:

solicitud=nombre del cliente+domicilio de envio+{artículo}

significa que la solicitud siempre debe contener un nombre de cliente, un domicilio de envío y también cero o más ocurrencias de un artículo. Así, pudiéramos estar tratando con un cliente que pide un artículo, o dos, o algún comprador compulsivo que decide ordenar 397 artículos diferentes. En muchas situaciones reales, el usuario querrá especificar los límites inferior y superior de la iteración

Selección.

La notación de selección indica que un dato consiste en exactamente un elemento de entre un conjunto de opciones alternativas. Las opciones se encierran en corchetes y se separan por una barra vertical. Ejemplos:

sexo = [femenino | masculino]
tipo de cliente = [gobierno| industria | universidad | otro]

Es sumamente importante revisar las opciones de selección con el usuario para asegurarse de cubrir todas las posibilidades.

Alias.

Un alias, como el término lo indica, es una alternativa de nombre para un dato. Esto es una ocurrencia común cuando se trata con diversos grupos de usuarios en diferentes departamentos o ubicaciones geográficas, que insisten en utilizar distintos nombres para decir lo mismo.

El alias se incluye en el diccionario de datos para que esté completo, y se relaciona con el nombre primario u oficial del dato. Por ejemplo:

comprador = "alias de cliente"

Nótese que la definición de comprador no muestra su composición (nombre, domicilio, etc.). Todos estos detalles deben darse sólo para el nombre primario del dato, para minimizar la redundancia en el modelo.

Aún cuando el diccionario de datos relaciona correctamente los alias con el nombre primario de los datos, debe evitarse el uso de alias hasta donde sea posible. Esto se debe a que los nombres de datos se suelen ver primero y son más visibles para todos los usuarios en los DFD's, en donde pudiera no ser tan obvio que comprador y cliente sean alias. Es mejor, de ser posible, lograr que todos los usuarios se pongan de acuerdo en un solo nombre para cada dato.

Cómo mostrar el diccionario de datos al usuario.

El diccionario de datos lo crea el analista durante el desarrollo del modelo del sistema, pero el usuario debe ser capaz de leerlo y entenderlo para poder verificar el modelo. Esto plantea las siguientes preguntas:

¿Podrán los usuarios entender la notación del diccionario de datos?.
¿Cómo podrán los usuarios verificar que el diccionario está completo y correcto?
¿Cómo se crea el diccionario?.

La cuestión de la aceptación por el usuario de la notación del diccionario, puede en un momento dado confundir en la mayoría de los casos. Es cierto, la notación del diccionario se ve algo matemática; pero como se ha visto, el número de símbolos que el usuario debe aprender es muy pequeño. Los usuarios están acostumbrados a la variedad de notaciones formales en su trabajo y vida personal, por lo tanto puede no ser difícil para el usuario aprender la notación del diccionario.

En el aspecto de verificación del diccionario de datos por el usuario se llega generalmente a la pregunta ¿debe el usuario leer a detalle todo el diccionario para asegurarse de que está correcto?, es difícil que un usuario esté dispuesto a hacer esto. Lo correcto sería que el usuario verificara el diccionario en conjunto con el DFD, el diagrama entidad-relación y las especificaciones de los procesos.

Hay varios detalles acerca de la corrección del sistema que el analista puede hacer por su cuenta sin ayuda del usuario: puede asegurarse que el diccionario esté completo y sea consistente y no contradictorio.

Implantación del diccionario de datos.

En un sistema mediano o grande, el diccionario de datos puede representar una cantidad grande de trabajo. Puede haber diccionarios con miles de entradas e incluso un sistema sencillo tendrá cientos de entradas. Así que debe pensarse en cómo desarrollar el diccionario de datos, porque muchas veces es una tarea muy grande para el analista.

En la actualidad se dispone de herramientas de programación y paquetes de computadora que permiten construír el diccionario de datos y desde luego que es totalmente recomendable emplear estos recursos como una gran ayuda para que el analista pueda desarrollar el diccionario y posteriormente implementarlo.

Para implantar el diccionario de datos es necesaria la aprobación y previa verificación del usuario para evitar que se tenga que regresar a etapas anteriores del análisis.

Construír un diccionario de datos es un proceso laborioso y extenso del análisis del sistema. Pero también es una de las etapas más importantes: sin un diccionario formal que defina el significado de los términos, no se puede esperar precisión por parte del sistema.

2.3.4 EL DIAGRAMA DE ENTIDAD-RELACION.

El diagrama de entidad-relación es un modelo que describe la distribución de los datos almacenados en el sistema. También se le conoce como DER. El principal propósito del DER es representar los objetos de datos y sus relaciones.

Es importante modelar los datos de un sistema ya que las estructuras de datos y sus relaciones son complejas y resulta necesario examinarlas independientemente del proceso del sistema.

Los componentes de un diagrama entidad-relación.

"Hay cuatro componentes principales en un diagrama de entidad-relación:

1. *Tipos de objetos.*
2. *Relaciones.*
3. *Indicadores asociativos de tipo objeto.*
4. *Indicadores de subtipo/supertipo."* [13]

Tipos de objetos.

El tipo de objeto se representa con una caja rectangular, en la FIG. 2.14 se muestra un ejemplo.

FIG. 2.14. Representación del tipo de objeto en el DER.

Representa una colección o conjunto de objetos del mundo real cuyos miembros individuales (llamados instancias) tienen las siguientes características:

- Cada uno puede identificarse de manera única por algún medio. Existe alguna forma de diferenciar entre instancias individuales del tipo objeto. Por ejemplo en la FIG. 2.14 el objeto cliente debe distinguirse tal vez por un número de cuenta, por sus apellidos, etc.
- Cada uno juega un papel necesario en el sistema que se construye. Es decir, para que el tipo de objeto sea real, debe poder decirse que el sistema no puede operar sin el acceso a ese objeto.
- Cada uno puede describirse por uno o más datos. Es decir, un objeto tiene sus atributos, como por ejemplo: el objeto CLIENTE puede describirse por medio de datos tales como nombre, domicilio, número telefónico, etc.

[13] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 262.

El objeto es algo material del mundo real y el tipo de objeto es su representación en el sistema. Sin embargo, un objeto también puede ser algo no material, por ejemplo: horarios, planes, etc. Es importante aclarar que la personas son tipos de objeto en un sistema, pero una persona puede tener diversos tipos de objeto, por ejemplo: una persona puede ser empleado en un modelo y cliente en otro, aún dentro del mismo modelo. Por convención, los nombres de los tipos de objeto se utilizan en singular.

Relaciones.

"Los objetos se conectan entre sí mediante relaciones. Una relación representa un conjunto de conexiones entre objetos, y se representa por medio de un rombo. La FIG. 2.15 muestra una relación sencilla que pudiera existir entre dos o más objetos."[14]

FIG.2.15. Representación gráfica de una relación en un DER.

La relación representa un conjunto de conexiones. Cada instancia de la relación representa una asociación entre cero o más ocurrencias de un objeto y cero o más ocurrencias del otro. Así en la figura anterior, la relación etiquetada como COMPRAS puede contener las siguientes instancias individuales:

- Instancia 1: el cliente 1 compra el artículo 1.
- Instancia 2: el cliente 2 compra los artículos 2 y 3.
- Instancia 3: el cliente 3 compra el artículo 4.
- Instancia 4: el cliente 4 compra los artículos 5, 6 y 7.
- etc..

Esto significa que, una relación puede conectar dos o más instancias del mismo objeto.

Además, la relación representa algo que debe ser recordado por el sistema: algo que no pudo haberse calculado ni derivado mecánicamente. Esto es, la relación representa la memoria del sistema.

Puede existir más de una relación entre dos objetos, aunque una situación más común es ver múltiples relaciones entre múltiples objetos. La relación se puede describir desde la perspectiva de cualquiera de los tipos de objetos participantes, y todas esas perspectivas son válidas. De hecho, el conjunto de todos estos puntos de vista es el

[14] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 264.

que describe completamente la relación. Por ejemplo, en la FIG. 2.16 puede verse la relación de negociación de precios en cualesquiera de las tres formas:

1. El agente de bienes raíces negocia el precio entre el cliente y el vendedor.
2. El cliente negocia el precio con el vendedor, mediante el agente de bienes raíces.
3. El vendedor negocia el precio con el cliente, mediante el agente de bienes raíces.

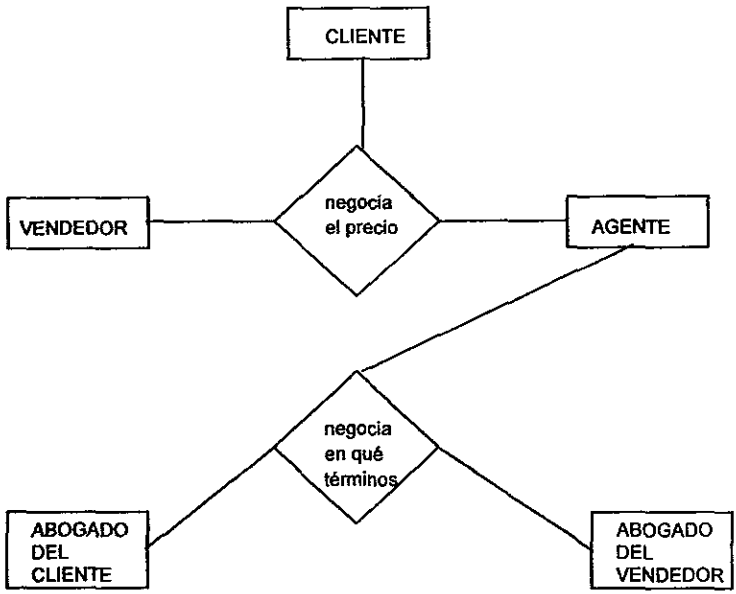


FIG. 2.16 Diferentes formas de las relaciones entre los tipos de objetos.

En algunos casos, puede haber relaciones entre diferentes instancias de un mismo tipo de objeto, aunque la mayoría de las relaciones de interés serán entre instancias de diferentes tipos de objeto.

Notación alternativa para relaciones.

Entre las notaciones utilizadas, encontramos una que muestra tanto la cardinalidad como la ordinalidad. Por ejemplo, la FIG. 2.17 muestra una relación entre CLIENTE y ARTICULO en la cual la notación indica que:

- El CLIENTE es el objeto principal.
- La *relación consiste en un cliente* conectado con N artículos, sólo una a la vez, es decir, sólo puede haber un cliente involucrado en cada instancia de la relación como se muestra en la FIG. 2.17a.

Otra notación común es la flecha de dos puntas seguidas, esto indica una relación de uno a muchos entre objetos como se muestra en la FIG. 2.17b.

FIG. 2.17a.

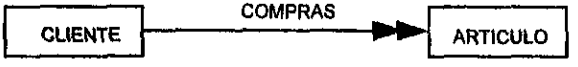


FIG. 2.17b.

FIG. 2.17. Notación para la relación entre objetos.

Indicadores asociativos de tipo objeto.

Un indicador asociativo de tipo objeto representa algo que funciona como objeto y como relación. Es algo acerca de lo cual se desea mantener alguna información.

Por ejemplo, supongamos que un cliente adquiere un artículo. La relación asocia un cliente con uno o más artículos, pero suponiendo que se desean recordar algunos datos acerca de la instancia de una compra, como la hora en que ésta se realizó, este dato, hora del día no es un atributo de CLIENTE, ni de ARTICULO. Más bien, se asocia con la compra misma. Como se ilustra en la FIG. 2.18.

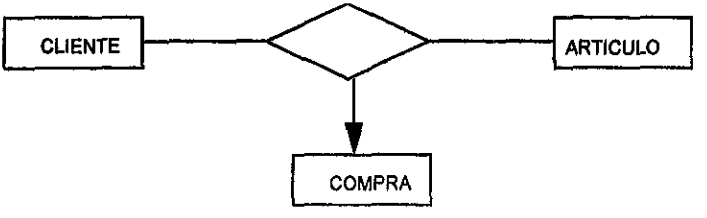


FIG. 2.18. Creación de un tipo de objeto como resultado de una relación entre dos tipos de objetos.

Nótese que COMPRA ahora se escribe dentro de una caja rectangular conectada a un rombo de relación sin nombre. Esto pretende indicar que COMPRA funciona como: un tipo de objeto y como una relación que conecta los dos tipos de objeto CLIENTE y

ARTICULO, lo importante en este caso es que cliente y artículo se mantienen solos, de hecho existirían con o sin la compra. Por otro lado una COMPRA aparece sólo como resultado de una relación entre los otros objetos con los cuales está conectada.

Además nótese que la relación de la FIG. 2.18 no tiene nombre, esto se debe a que el *indicador asociativo* de objeto (COMPRA) también es el nombre de la relación.

Indicadores de tipo subtipo/supertipo.

"Los tipos de objeto de subtipo/supertipo consisten en tipos de objeto de una o más categorías, conectados por una relación."[15] La FIG. 2.19 muestra un subtipo/supertipo, la categoría general es EMPLEADO y las subcategorías son EMPLEADO ASALARIADO y EMPLEADO POR HORAS.

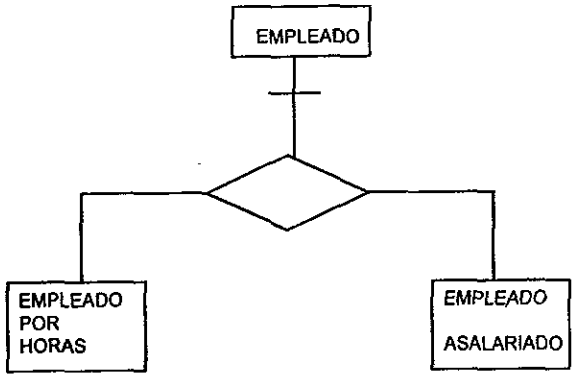


FIG. 2.19. Tipo de objeto subtipo/supertipo.

Podemos observar que los subtipos se conectan por medio de una relación sin nombre y el supertipo se conecta a la relación con una línea que contiene una barra. El supertipo denota los datos que se aplican a todos los subtipos. Por ejemplo, puede ser que todos los empleados se describan por datos tales como: nombre, años de servicio, domicilio particular, sueldo, etc.

Sin embargo, cada subtipo se describe por medio de datos diferentes, por ejemplo un empleado asalariado puede tener los siguientes datos: salario mensual, porcentaje anual adicional, etc, y el empleado por hora por medio de datos como: paga por hora, cantidad por tiempo extra, hora de comienzo, etc.

También se dispone de herramientas de programación y paquetes de computadora para construir el diagrama entidad-relación y desde luego que es totalmente válido y recomendable emplear estos recursos como una gran ayuda para el analista.

[15] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 269.

2.3.5 MINIESPECIFICACIONES.

"Una miniespecificación define lo que debe hacerse para transfomar entradas en salidas. Es una descripción detallada de la política de negocios del usuario que cada burbuja lleva a cabo." [16] Una miniespecificación se conoce también como especificación del proceso.

Existe una variedad de herramientas que podemos utilizar para producir una miniespecificación como son: tabla de decisiones, lenguaje estructurado (español, inglés, etc.), pre/post condiciones, diagramas de flujo, etc.

Es válido utilizar cualquier método, siempre y cuando se satisfagan tres requerimientos:

- La miniespecificación debe expresarse de una manera que puedan verificarse tanto por el usuario como por el analista. Precisamente por esta razón se evita el lenguaje narrativo como herramienta de especificación, ya que es ambiguo, sobre todo si describe acciones alternativas (decisiones) y acciones repetitivas (ciclos).
- La miniespecificación debe expresarse en una forma que pueda ser comunicada efectivamente al público amplio que esté involucrado. A pesar de que el analista es quien la escribe, generalmente será un público bastante diverso de usuarios, administradores, auditores y otros, el que leerá la miniespecificación.
- La especificación del proceso no debe imponer decisiones de diseño e implantación arbitrarias. A menudo esto es muy difícil, pues el usuario, de quien depende la política que realizará cada burbuja en el DFD, suele escribirla en los términos en los que la lleva en la actualidad. Por ello, el trabajo del analista consiste en destilar de esto la esencia de lo que dicha política es y no cómo se lleva a cabo hoy en día.

Un punto que debe quedar claro es que las miniespecificaciones sólo se desarrollan para los procesos de más bajo nivel en un conjunto de diagramas por niveles en un DFD. Esto es, la miniespecificación para una burbuja de nivel superior es el DFD de nivel inferior. Esto se ilustra en la FIG. 2.20, aquí los procesos de mayor nivel se definen por medio de la red de procesos del nivel inmediato inferior.

Las herramientas principales para definir las miniespecificaciones o especificación de proceso son tres:

- Lenguaje estructurado (español, inglés, etc).
- Pre/post condiciones.
- Tablas de decisión.

Cada uno de ellos seran objeto de análisis a detalle más adelante.

[16] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 227.

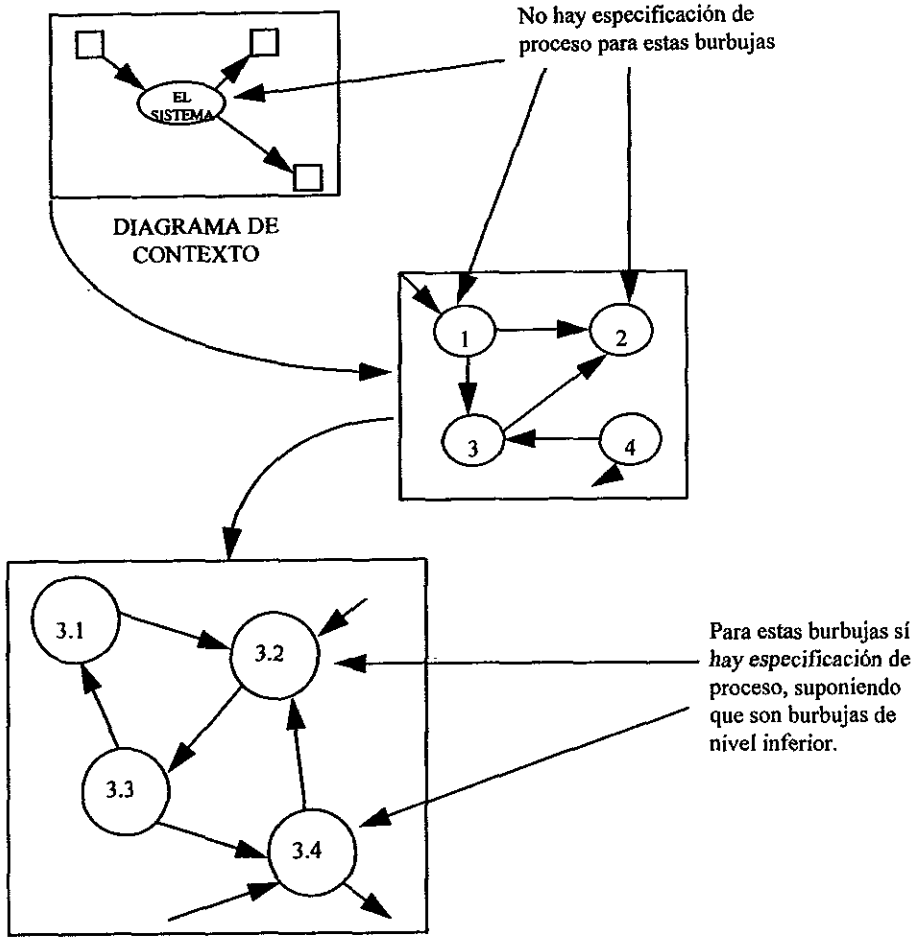


FIG. 2.20. Especificaciones de proceso para burbujas de bajo nivel.

2.3.6 ARBOLES Y TABLAS DE DECISION.

Los árboles y tablas de decisión son modelos cuyas características los hacen aplicables en el análisis y diseño de sistemas de información.

Arboles de decisión.

Un árbol de decisiones es una secuencia de condiciones y decisiones en forma horizontal utilizando líneas y nodos como lo describe la FIG. 2.21.

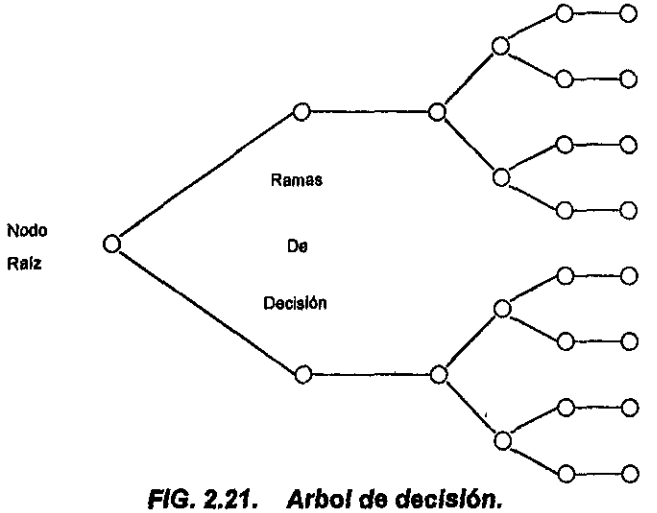


FIG. 2.21. Arbol de decisión.

Cada decisión que se toma depende del valor actual de la variable que se está probando y de todas las decisiones previas que se hayan tomado. El árbol aparece en forma horizontal con la raíz del lado izquierdo. La raíz indica la primera decisión y corresponde a la primera variable de decisión. Los resultados de las decisiones se muestran como ramas de la raíz. Cada nodo del árbol representa un punto de decisión, en donde cada una de las ramas que salen del nodo corresponde a un valor posible de la variable de decisión asociada.

El árbol proporciona una visión gráfica de las variables que se prueban, las decisiones que se toman y el orden en que se realiza esta toma de decisiones. El árbol de decisiones hace que los analistas de sistemas identifiquen las decisiones reales que deben tomarse, y de esta manera reduce la probabilidad de que se pasen por alto decisiones críticas. Asimismo, el árbol de decisiones obliga a los analistas de sistemas a considerar la secuencia de decisiones.

Tablas de decisión.

Una tabla de decisiones es una matriz de filas y columnas que muestra las condiciones del proceso a evaluar y las acciones a tomar respecto a dichas condiciones. La TABLA 2.1 muestra el esqueleto de una tabla de decisiones en su forma general:

	Reglas				
Titulo de la tabla de decisiones	**1**	**2**	**3**	**4...**	**N**
Lista			**entrada**		
de			**de**		
condiciones			**condic.**		
Lista			**entrada**		
de			**de**		
acciones			**acciones**		

TABLA 2.1. Forma general de una tabla de decisión.

La mitad superior de la tabla contiene las condiciones de las decisiones que se expresan en las áreas denominadas lista y entrada. La lista de condiciones son aquellos criterios que el tomador de decisiones desea aplicar a su modelo lógico. Para incorporar estos criterios en una tabla de decisiones, éstos deben expresarse de manera que sigan la palabra SI (Si condicional).

La mitad inferior de la tabla contiene las acciones que se van a tomar cuando se cumplen las condiciones especificadas. Para incorporar acciones en la tabla, éstas deben estructurarse de manera que sigan la palabra ENTONCES.

La combinación de las condiciones con las acciones da por resultado una oración de la forma SI existen estas condiciones ENTONCES realiza estas acciones.

Una tabla de decisiones se crea listando todas las variables relevantes (conocidas como condiciones o entradas) y todas las acciones relevantes en su lado izquierdo. En muchas aplicaciones es fácil (y preferible) expresar las variables como binarias, pero también se pueden construir las tablas de decisión a partir de variables multivaluadas.

"Debe discutirse cada regla con el usuario para asegurarse de que se ha identificado la acción o acciones correctas para cada combinación de variables. Es bastante común, al hacer esto, encontrar que el usuario jamás ha pensado en ciertas combinaciones de variables, o que nunca hayan ocurrido en su experiencia. La ventaja del enfoque de la tabla de decisiones es que el analista se puede concentrar en una regla a la vez." [17]

Otra ventaja del enfoque de la tabla de decisiones es que no implica ninguna forma particular de implantación. Es decir, cuando el analista entrega la tabla (junto con los DFD's) al diseñador, hay mucha libertad de elección en términos de la estrategia de implantación. Por ello, a menudo se conocen las tablas de decisiones como una herramienta de modelado de sistemas que no es de tipo procedimiento, pues no especifican ningún algoritmo de procedimiento concreto para realizar las acciones requeridas.

"En resumen, deben seguirse los siguientes pasos para crear una tabla decisiones para una especificación de proceso:

1. *Identificar todas las condiciones o variables de la especificación. Identificar todos los valores que cada variable puede tomar.*
2. *Calcular el número de combinaciones de las condiciones. Si todas las condiciones son binarias, entonces existen 2^n combinaciones de N variables.*
3. *Identificar cada posible acción que se pide en la especificación.*
4. *Crear una tabla de decisión vacía, listando todas las condiciones y acciones en el lado izquierdo y numerando las combinaciones de las condiciones en la parte superior de la tabla.*
5. *Listar todas las combinaciones de condiciones, una para cada columna de la tabla.*
6. *Examinar cada columna (conocida como regla) e identificar las acciones apropiadas que se deben tomar.*
7. *Identificar con el usuario las omisiones, contradicciones o ambigüedades."* [18]

[17] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág. 245.
[18] Ibidem, pág. 246.

2.3.7 ESPAÑOL ESTRUCTURADO.

"El español estructurado, es lenguaje español (o inglés u otro) con estructura. Es un subconjunto de todo el idioma con importantes restricciones sobre el tipo de frases que pueden utilizarse y la manera en que pueden juntarse dichas frases." [19] Su propósito es hacer un balance razonable entre la precisión del lenguaje formal de programación y la informalidad y legibilidad del lenguaje cotidiano. Una frase en lenguaje estructurado puede consistir en una ecuación algebraica, por ejemplo:

X = (Y * Z)/(Q+14)

o en una frase declarativa. Además las frases que describen los cálculos pueden usarse con expresiones de los verbos CALCULAR, AÑADIR, FIJAR, etc., por lo que pudo haberse escrito el ejemplo anterior asi:

CALCULAR X = (Y*Z)/(Q+14)

y también se tienen cálculos expresados en lenguaje, como los siguientes:

FIJAR IMPUESTO A 13.
SUMAR 3 A X.
MULTIPLICAR PRECIO UNITARIO POR CANTIDAD.
DIVIDIR GANANCIAS ACTUALES ENTRE PERDIDAS ACTUALES.

Los verbos deben escogerse de entre un pequeño grupo de verbos orientados a la acción tales como:

CONSEGUIR (o ACEPTAR o LEER).
PONER (o MOSTRAR o ESCRIBIR).
ENCONTRAR (o BUSCAR o LOCALIZAR).
SUMAR.
RESTAR.
MULTIPLICAR.
DIVIDIR.
CALCULAR.
BORRAR.
ENCONTRAR.
VALIDAR.
MOVER.
REEMPLAZAR.
FIJAR.
ORDENAR.

[19] Yourdon Edward; Análisis estructurado moderno; Prentice Hall, 1993, México, pág 231.

En muchas organizaciones se llega a la conclusión de que entre 40 y 50 verbos son suficientes para describir cualquier política dentro de una miniespecificación.

Los objetos (el tema de las frases imperativas sencillas) deben consistir sólo en datos que se han definido en el diccionario o ser términos locales. Los términos locales son aquellos que se definen explícitamente en una especificación de proceso individual. Esto significa que los términos locales se definen dentro de la especificación de proceso donde ocurren, y no se definen en el diccionario de datos. Además por definición, los términos locales sólo se conocen en un contexto local, es decir dentro de una burbuja, en un DFD.

Finalmente, el lenguaje estructurado permite que se combinen frases en unas cuantas formas limitadas que se toman de las construcciones acostumbradas de la programación estructurada.

La construcción SI-ENTONCES-OTRO se utiliza para describir frases alternativas que se deben realizar según el resultado de la decisión binaria. Esta puede tomar cualquiera de las formas siguientes:

```
SI condición-1
  frase-1
FIN SI
```

o bien

```
SI condición-1
  frase-1
OTRO
  frase-2
FIN SI
```

La construcción CASO se utiliza para describir frases alternativas que se efectuarán basándose en los resultados de una decisión multivaluada (en contraste con la decisión binaria que tiene lugar con la construcción SI-ENTONCES). La construcción CASO toma la forma general:

```
HACER CASO
CASO variable = valor-1
  frase-1
.
.
CASO variable = valor-n
  frase-n
OTRO
  frase n+1
FIN CASO
```

La cláusula OTRO suele usarse para abarcar situaciones que el usuario se olvida de especificar y por las que el analista se olvida de preguntar, esto a menudo llevará a discusiones entre el usuario y el analista que de otra manera no sucederían hasta después de puesto en operación el sistema.

La construcción HACER-MIENTRAS se usa para describir una frase que deberá llevarse a cabo repetitivamente hasta que alguna condición boolena se haga verdadera. Toma la forma general:

```
HACER-MIENTRAS condición-1
  frase-1
FIN HACER
```

También se incluye otra estructura que ejecuta una frase especificada por lo menos una vez antes de hacer una prueba para ver si debe repetirse. Esta variante, usualmente conocida como la construcción REPITE-HASTA, tiene la siguiente forma:

```
REPITE
  frase-1
HASTA condición-1
```

Se pueden construir frases compuestas a partir de combinaciones de frases sencillas y las estructuras sencillas, de acuerdo con las siguientes reglas:

1. Una secuencia lineal de frases sencillas equivale (estructuralmente) a una frase sencilla. Así que la secuencia:

```
frase-1
frase-2
.
.
frase-n
```

es estructuralmente equivalente a una frase sencilla única y puede ser sustituida dondequiera que se espera una frase sencilla. Esto permite construir estructuras como la siguiente:

```
SI condición-1
        frase-1
        frase-2
OTRO
        frase-3
        frase-4
        frase-5
FIN SI
```

o bien,

```
HACER MIENTRAS condición-1
        frase-1
        frase-2
        frase-3
FIN HACER
```

2. Una construcción SI-ENTONCES-OTRO sencilla se considera estructuralmente equivalente a una frase única sencilla. Esto permite que las estructuras SI-ENTONCES-OTRO se aniden dentro de otras estructuras iguales, o dentro de estructuras HACER-MIENTRAS, o dentro de estructuras CASO. Como la siguiente:

```
SI condición-1
        frase-1
        SI condición-2
                frase-2
                frase-3
        OTRO
                frase-4
                frase-5
        FIN SI
        frase-6
OTRO
        frase-7
        SI condición-3
                frase-8
        FIN SI
        frase-9
FIN SI
```

3. Una estructura HACER-MIENTRAS sencilla se considera estructuralmente equivalente a una frase única sencilla. Esto permite que las estructuras HACER-MIENTRAS se aniden dentro de otras estructuras iguales, o dentro de estructuras SI-ENTONCES-OTRO, o dentro de estructuras CASO. Así, se puede tener una especificación en lenguaje estructurado.

4. Una estructura sencilla CASO se considera estructuralmente equivalente a una frase única sencilla. Esto permite que las estructuras CASO se aniden dentro de otras iguales, dentro de las estructuras SI-ENTONCES-OTRO o dentro de estructuras HACER-MIENTRAS.

Como se observa, la construcción de miniespecificaciones a través del método de Lenguaje Estructurado es arbitrario y en ocasiones complejo, para evitar esto existen tres reglas:

1. Restringir la miniespecificación en lenguaje estructurado a una sola página de texto. Si la especificación ocupa más de una página, entonces el analista (con la ayuda del usuario) debe pensar en una forma totalmente distinta de formular la política. Si no se puede, entonces es posible que el proceso mismo (esto es, la burbuja dentro del DFD) sea demasiado complejo, y debe partirse en una red de procesos más simples de nivel inferior.
2. No permitir más de tres niveles de anidamiento. En particular, en el caso de estructuras SI-ENTONCES-OTRO, incluso más de dos niveles de anidamiento es un indicio de que sería preferible una especificación mediante tabla de decisiones.
3. Deben utlizarse sangrías, para los niveles de anidamiento, para evitar confusiones.

Pre/post condiciones.

Las pre/post condiciones son una manera de describir la función que debe realizar el proceso, sin decir mucho acerca del algoritmo o procedimiento que se utilizará. Resulta ser un enfoque particularmente útil cuando:

1. El usuario tiene tendencia a expresar la política llevada a cabo por la burbuja en términos de un algoritmo particular que ha estado utilizando durante décadas.
2. El analista está seguro de que existen muchos algoritmos distintos que podrían usarse.
3. El analista desea que el programador explore varios de estos algoritmos, pero no quiere involucrarse en esos detalles, y sobre todo, no quiere entrar en discusiones con el usuario.

Existen dos partes principales en este proceso: precondiciones y postcondiciones. Además tales especificaciones pueden contener términos locales.

Las precondiciones describen todas las cosas que deben darse antes de que el proceso pueda comenzar a ejecutarse. Regularmente éstas describirán lo siguiente:

- Qué entradas se encuentran disponibles. Estas entradas llegan mediante un flujo conectado con un proceso. Puede haber casos en los que diversos flujos entran a un proceso, pero sólo uno de ellos es precondición necesaria para que se active el proceso.
- Qué relación debe existir entre las entradas. A menudo, una precondición especificará que deben llegar dos entradas con campos que corresponden. O bien, la precondición puede especificar que un componente de un dato de entrada debe estar dentro de cierto intervalo.
- Qué relaciones deben existir entre entradas y almacenes de datos. Una precondición pudiera estipular que exista un registro dentro de un almacén que corresponda con algún aspecto de un dato de entrada.
- Qué relaciones deben existir entre diferentes almacenes o dentro de un almacén dado.

De manera similar, las postcondiciones describen lo que debe darse cuando el proceso ha concluido y describen lo siguiente:

- Las salidas que generará o producirá el proceso. Esta es la forma más común de postcondición.
- Las relaciones que existirán entre los valores de salida y los valores originales de entrada.
- Las relaciones que existirán entre valores de salida y los valores en uno o varios de los almacenes.
- Los cambios que se hayan dado en los almacenes.

Cuando se construyen miniespecificaciones de pre/post condiciones se debe comenzar por describir las situaciones normales de proceso. Pudieran existir diversas situaciones normales diferentes, cada una de las cuales se expresa como precondición distinguible e individual. Para cada una de estas precondiciones se debe describir la condición de la burbuja del proceso cuando se han producido las salidas y se han modificado los almacenes. Después de haber descrito las situaciones normales de proceso, deben incluirse precondiciones y postcondiciones apropiadas para los casos de error y casos anormales.

El método de pre/post condiciones es bastante útil, pero hay ocasiones en que puede ser no tan apropiado. La falta de pasos intermedios entre entradas (precondiciones) y salidas (postcondiciones) y si además existen relaciones complejas entre entradas y salidas pueden hacer más fáctible de realizar las miniespecificaciones utilizando un lenguaje estructurado.

2.4 BASES DE DATOS RELACIONALES.

Las bases de datos surgieron como resultado de la evolución de los métodos de almacenamiento de información. Estos métodos, en sus primeras etapas incluían archivos secuenciales y posteriormente archivos de acceso directo pero de tamaño fijo los cuales difícilmente cumplían con los requerimientos que exigía un sistema de información, además los programas de aplicación requerían de muchas líneas de código para la organización de los datos contenidos en los archivos y el mantenimiento de estos archivos implicaba una modificación y recompilación importante en los programas fuente.

Más adelante fueron surgiendo las bases de datos no relacionales, las cuales tuvieron un importante avance en el almacenamiento de información porque no solo cumplían con las funciones básicas de entrada y salida sino que su representación era ya en forma tabular, sin embargo, todavía existía cierta dependencia de datos y al no ser relacionales se generaba mucha redundancia innecesaria. Todo lo anterior dio como resultado la necesidad de contar con una técnica más avanzada para resolver las situaciones mencionadas.

Así, surgen las bases de datos relacionales como un método para realmente diseñar, administrar, organizar y actualizar todo lo relativo a los datos en un sistema de información de manera segura, independiente, en tiempo real y con una estructura que le permita crecer en cualquier momento sin afectar a la aplicación.

2.4.1 BASE DE DATOS RELACIONAL, TERMINOLOGIA Y CARACTERISTICAS.

Base de datos relacional es un conjunto de tablas bajo una misma identificación que trabajan en base a relaciones entre las mismas, las relaciones pueden ser entre dos o más tablas y puede generarse una nueva a partir de los registros que cumplen con el criterio de correspondencia, las relaciones se llevan a cabo a través de campos llaves.

Los términos más comunmente empleados en las bases de datos relacionales son:

Campo. Es el grupo de datos más pequeño y puede estar formado por cualquier número de bytes. Se le conoce también como ítem de datos o columna.

Registro. Es un conjunto de campos, los cuales están relacionados de tal forma que componen un registro de alguna entidad. Un registro en una tabla, no necesariamente debe ser leído completo, sino que pueden leerse agregados de datos de él. Se le llama también tuple o renglón.

Agregado de datos. Es una colección de campos dentro de un registro. Varios campos que forman parte de un registro al leerse juntos pueden tener un significado especial o más completo al formar un agregado de datos.

Tabla. Una tabla es un conjunto de registros. En las bases de datos relacionales se emplean tablas bidimensionales las cuales son una de las maneras más naturales de representar y visualizar datos. Las tablas deben organizarse y definirse de tal forma que no se pierdan las relaciones entre datos. Una tabla es una matriz bidimensional por lo que un renglón es un registro y una columna es un campo. Las tablas tienen las siguientes características:

- Cada columna de la tabla representa un campo.
- Son homogéneas por columna, es decir, todos los datos de una columna son del mismo tipo o clase de datos.
- Cada columna tiene nombre propio.
- Todos los renglones son diferentes, deben variar en al menos una columna.
- Tanto los renglones como las columnas pueden ser considerados en cualquier secuencia y en cualquier momento sin afectar la información ni la semántica de cualquier función que utilice la tabla.

Relación. Es la correspondencia existente entre los registros de las diferentes tablas que componen la base de datos relacional. Esta correspondencia se lleva a cabo por medio de campos llaves, los cuales, al hacer la relación agrupan a un conjunto de registros que cumplen con los criterios establecidos.

Entidad. Las entidades son objetos tangibles o intangibles del mundo real sobre las cuales se almacena información (un artículo, un empleado, un lugar, un suceso, una cuenta, etc.) y que tiene propiedades o atributos que eventualmente se deben registrar. A cada registro debe referirse por medio de un identificador de entidad para ser relacionado con una entidad. Uno de los atributos de la entidad es el que toma el carácter de identificador de entidad.

Características de las bases de datos relacionales.

- Servir a más de una aplicación.
- Representación en forma tabular.
- Independencia de datos.
- Redundancia de datos controlada o mínima.
- Protección de seguridad.
- Accesibilidad en tiempo real.
- Operación basada en relaciones.
- Existe un software para administrarla (administrador de base de datos) y un lenguaje de consulta y actualización para realizar las transacciones.
- Pueden crearse índices y esquemas para las tablas.
- Facilidad para importar y exportar datos.

2.4.2 OBJETIVOS DE LA ORGANIZACION DE LA BASE DE DATOS.

Todas estas características nos permiten una buena organización de la base de datos. Los objetivos que se persiguen con la organización de la base de datos según propone James Martin se dividen en primarios y secundarios.

Objetivos primarios de la organización de la base de datos.

Los datos podrán usarse de múltiples maneras.

Esto quiere decir que la base de datos puede estar disponible para múltiples usuarios y aplicaciones y aunque sean los mismos datos se pueden percibir de diferente manera.

Proteger la inversión intelectual.

Cada vez que se modifique la base de datos no será estrictamente necesario rehacer los programas fuente, los cuales representan una gran inversión en horas-hombre.

Bajo costo.

El almacenamiento y uso de los datos debe tener un bajo costo además de minimizar el costo del mantenimiento a la base de datos.

Menor proliferación de datos.

Muchas de las necesidades de datos de las nuevas aplicaciones quedarán satisfechas con los datos existentes (con relación a un buen diseño), evitándose así la excesiva proliferación de los datos.

Desempeño.

La petición de datos se atenderá con la rapidez adecuada de acuerdo al uso que de ellos habrá de hacerse.

Claridad.

Debido a su representación los usuarios pueden saber más fácilmente que datos se encuentran a su disposición.

Facilidad de uso.

Se puede tener un fácil acceso a los datos gracias a la existencia de un sistema administrador de la base de datos, quedando ajenas al usuario todas las complejidades internas.

Flexibilidad.

Los datos pueden ser accesados o explorados de manera flexible, es decir, por diferentes caminos gracias a las relaciones existentes entre tablas.

Facilidad para el cambio.

La base de datos puede crecer y variar sin interferir con la manera de utilizar los datos.

Precisión y coherencia.

El sistema puede evitar múltiples versiones o versiones erróneas de datos gracias al control de transacciones.

Reserva.

Puede evitarse el acceso no autorizado de los datos. Los mismos datos pueden estar sujetos a diferentes restricciones para diferentes usuarios.

Disponibilidad.

Los datos están disponibles en cualquier momento y pueden accesarse aún sin los programas de aplicación debido a un lenguaje estructurado de consulta.

Objetivos secundarios de la organización de la base de datos.

Estos sirven para facilitar el logro de los objetivos primarios.

Independencia física de los datos.

El hardware y las técnicas de almacenamiento pueden ser modificados sin obligar a la modificación de los programas de aplicación.

Independencia lógica de los datos.

Lo mismo sucede con la estructura lógica de los datos, éstos pueden variar sin que afecte a los programas de aplicación.

Redundancia controlada.

Los ítems de datos se almacenarán sólo una vez excepto cuando existan razones técnicas o económicas que obliguen al almacenamiento redundante.

Normalización de los datos dentro de un organismo.

Es importante lograr un acuerdo interdepartamental sobre el formato y definición de los datos. Si no existe una normalización se crearán datos incompatibles.

Diccionario de datos.

Es muy necesario para tener un documento que defina todos los ítems de datos que intervienen en la base de datos.

Lenguaje del usuario final.

Se cuenta con un lenguaje estructurado de consulta para la generación de reportes con lo cual los usuarios finales se ven libres de tener que escribir un programa de aplicación convencional.

Controles de integridad.

Siempre que sea posible se pueden hacer chequeos sobre el estado de los controles (tamaño de tablas, valores de los campos, tamaño del registro, integridad de índices, etc.) para asegurar la exactitud de los datos.

Fácil recuperación en caso de fallo.

Es posible una recuperación automática de datos en caso de fallas, siempre y cuando se lleve una bitácora de todas las transacciones que se llevan a cabo en la base de datos.

Afinación.

La base de datos puede afinarse aún después de terminada la aplicación, pueden hacerse ajustes. Por ejemplo, sobre el tipo de datos de cada columna o eliminación, modificación o creación de índices de las tablas.

2.4.3 TIPOS DE RELACIONES.

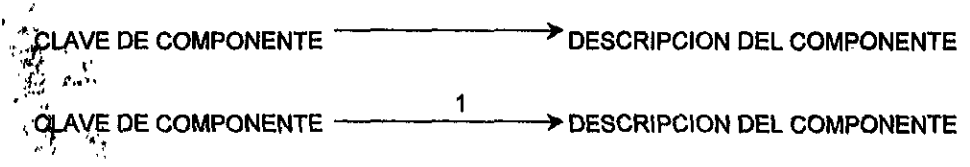
Toda relación existente entre dos ítems de datos puede ser de dos formas diferentes: simple o compleja.

Relación simple.

Se entiende por relación simple aquella en la que hay una correspondencia biunívoca (de uno a uno) entre los ítems. Por ejemplo:

En una empresa inmobiliaria, donde el producto es un inmueble, el cual a su vez se divide en componentes, los ítems CLAVE DE COMPONENTE y DESCRIPCION DEL COMPONENTE tendrán siempre una relación simple, es decir, una correspondencia biunívoca porque a cada CLAVE DE COMPONENTE corresponde una DESCRIPCION DEL COMPONENTE y viceversa.

La relación simple se representa por medio de una flecha, o por medio de una flecha con el dígito 1 escrito sobre ella:



Relación compleja.

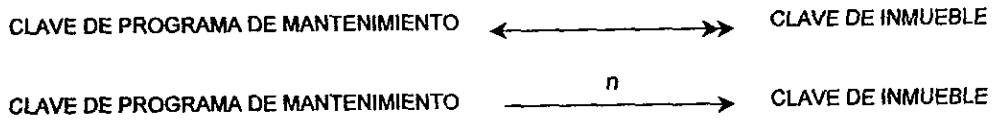
Se entiende por relación compleja aquella en la que hay una correspondencia de uno a muchos (1 a n) entre los ítems. Por ejemplo:

Los ítems CLAVE DE INMUEBLE y CLAVE DE PROGRAMA DE MANTENIMIENTO tienen las siguientes relaciones:

La relación entre CLAVE DE INMUEBLE y CLAVE DE PROGRAMA DE MANTENIMIENTO es simple, porque a cada CLAVE DE INMUEBLE corresponde sólo una CLAVE DE PROGRAMA DE MANTENIMIENTO.

En cambio la relación entre CLAVE DE PROGRAMA DE MANTENIMIENTO y CLAVE DE INMUEBLE es compleja (1 a n) porque en un programa de mantenimiento pueden estar contenidos muchos inmuebles.

La relación compleja se representa con una flecha doble o con una flecha con la letra n escrita sobre ella:



Cuando se tienen dos conjuntos de *ítems* por ejemplo: A y B se pueden presentar cuatro tipos de relación:

- La relación de A a B puede ser simple y la relación recíproca (de B a A) puede ser compleja.
- La relación de A a B puede ser compleja y la relación recíproca (de B a A) puede ser simple.
- Ambas relaciones pueden ser simples.
- Ambas relaciones pueden ser complejas.

2.4.4 CLAVES O LLAVES.

Las claves o llaves son un atributo o conjunto de atributos que se utilizan para identificar un registro. Existen dos tipos de claves: clave primaria y clave secundaria.

Clave primaria.

Es aquella que se utiliza para definir unívocamente un registro o tuple, ésta puede ser el identificador de entidad, el cual puede formarse por uno o más atributos. Esta clave es sumamente importante, ya que es utilizada para localizar el registro o tuple por medio de un índice. Los índices generalmente se definen en base a las claves primarias.

Clave secundaria o foránea.

Es aquella que se utiliza para identificar registros que no son únicos, sino todos aquellos que tienen cierta propiedad. Por lo general las tablas tienen varias claves secundarias, las cuales sirven para explorar en busca de entidades que cuentan con determinadas propiedades.

En siguiente ejemplo podemos identificar una clave primaria y dos claves secundarias:

Tabla Inmuebles.

Clave_de_Inmueble	Clave_de_Estado	Tipo_de_inmueble
111001	1	01
111002	1	01
111003	2	01
111004	2	02
111005	5	03
111006	32	03

La tabla anterior cuenta con una clave primaria y dos claves secundarias. La clave primaria es Clave_de_Inmueble porque identifica un registro de manera única, es decir, al inmueble, el estado donde se encuentra y el tipo de inmueble al que corresponde.

Asimismo las claves secundarias son Clave_de_Estado y Tipo_de_Inmueble porque identifican registros que cuentan con cierta propiedad: los inmuebles que se encuentran en un determinado estado y los inmuebles que pertenecen a un determinado tipo de inmueble.

2.5 DISEÑO ESTRUCTURADO.

El diseño estructurado proporciona una representación gráfica del software estableciendo las especificaciones del sistema y definiendo características para lograr que el producto cumpla con la calidad deseada. El objetivo del diseño estructurado es la creación de un sistema, el cual esté formado por la interacción de módulos independientes entre sí, esto hace que se facilite el mantenimiento a dichos módulos. Además, el diseño estructurado muestra gráficamente todos los módulos a desarrollar.

Sin un buen diseño, existe el riesgo de construir un sistema inestable, el cual puede fallar en el momento de realizar cambios a los módulos que lo integran.

Teniendo en cuenta que el diseño estructurado es un proceso por el cual se traducen los requerimientos del usuario en una representación gráfica del sistema, es importante mencionar que esta representación no muestra nada relacionado con la definición de archivos, la conceptualización de las bases de datos, la presentación de los datos de entrada y salida, la secuencia de instrucciones del lenguaje de programación; más bien nos conduce a una especificación de módulos independientes, los cuales interactúan entre sí para obtener las funciones requeridas del sistema.

Algunas de las características que se consideran en la conceptualización del diseño son las siguientes:

> *"1. Un diseño debe exhibir una organización jerárquica que haga un uso inteligente del control entre los componentes del software.*
> *2. Un diseño debe ser modular; esto es, el software debe estar dividido de forma lógica en elementos que realicen funciones y subfunciones específicas.*
> *3. Un diseño debe contener representaciones distintas y separadas de los datos y de los procedimientos.*
> *4. Un diseño debe llevar a módulos que exhiban características funcionales independientes.*
> *5. Un diseño debe llevar a interfaces que reduzcan la complejidad de las conexiones entre los módulos y el entorno exterior.*
> *6. Un diseño debe obtenerse mediante un método que sea reproducible y que esté conducido por la información obtenida durante el análisis de los requisitos del software".* [20]

Las características antes mencionadas no se consiguen por casualidad. Es necesario utilizar técnicas de ingeniería del software para producir un buen diseño, esto se logra a

[20] Pressman Roger; Ingeniería de software, Un enfoque práctico; McGraw Hill, 1993, México, pág. 332.

través de aplicar una metodología sistemática y una revisión a fondo en cada proceso en el desarrollo del sistema.

Esta representación hace referencia a dos características importantes de un programa para computadora. La primera es la estructura jerárquica que guardan los componentes por medio de procedimientos (módulos) y la segunda la estructura de datos utilizada.

Esta representación gráfica se realiza mediante la partición del problema en partes más pequeñas, esta partición relaciona los elementos de la solución propuesta con los elementos definidos durante el análisis de requerimientos del sistema.

Para realizar la representación de la estructura del programa se utilizan diferentes notaciones. La más común es un diagrama de árbol, la cual con frecuencia se le conoce como carta de estructura.

2.5.1 LA CARTA DE ESTRUCTURA (CDE).

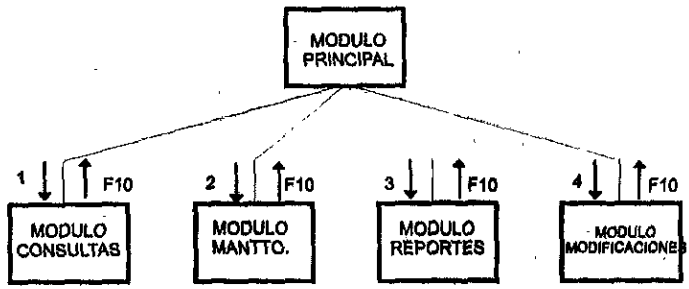
La carta de estructura es la representación gráfica de un sistema, ésta muestra el refinamiento conceptual del problema de forma general, identificando funciones internas de los procesos, la descomposición de funciones de alto nivel en subfunciones, la definición de los tipos de datos locales y el tipo de almacenamiento utilizado. Además establece las relaciones e interconexiones entre las funciones y los datos.

La carta de estructura generalmente presenta una organización de tipo jerárquico, la cual permite que el sistema se defina en unidades más pequeñas y manejables, además de proporcionar una definición más clara de la relación entre los diferentes elementos del sistema.

Dentro de la organización jerárquica que guarda la carta de estructura, se cuenta con procesos concurrentes que se pueden ejecutar en forma paralela y que se comunican a través de variables comunes entre ellos. Además, esta carta proporciona las interfaces de comunicación entre los módulos.

La organización jerárquica que guarda la carta de estructura puede ser representada por una gráfica dirigida acíclica, con un nodo principal que representa la raíz; éste a su vez tiene otras entidades subordinadas, en este caso la entidad que controla a otra se dice que es superior.

En la gráfica de la carta de estructura, se debe mostrar la relación de subrutinas y el paso de parámetros, éstos se indican con líneas que conecten las rutinas o módulos del sistema, como se muestra en la FIG. 2.22.



1,2,3 Y 4 Indica la selección del módulo

F10 Indica el regreso al módulo anterior.

FIG. 2.22. Ejemplo de una carta de estructura.

Esta representación jerárquica aisla los componentes de programación y facilita el entendimiento, la codificación, la depuración, las pruebas y la integración, así como las modificaciones al sistema.

2.5.2 CARACTERISTICAS DE LA CDE.

Dentro de los conceptos fundamentales del diseño estructurado se incluye la *abstracción*, el guardado de información, la modularidad, la verificación y los aspectos estéticos del diseño.

La abstracción es un concepto intelectual que permite aislar mentalmente o por separado las *cualidades de un problema* en particular. Durante el desarrollo del diseño y la definición de los requerimientos, la abstracción permite separar los aspectos conceptuales del sistema, los cuales más tarde serán instrumentados por medio de herramientas del diseño estructurado.

Se consideran tres mecanismos que se utilizan para la abstracción en el diseño de sistemas. A continuación se mencionan cuales son:

Abstracción procedimental.

Esta abstracción utiliza una secuencia de enunciados o instrucciones con una función *bien definida y limitada*, esta función se establece en términos del entorno del problema.

Abstracción en los datos.

En la abstracción de datos se especifica los tipos de datos u objetos, además, de especificar las operaciones permitidas sobre ellos; en esta forma de abstracción no se toman en cuenta los detalles de los objetos.

Abstracción en el control.

Esta forma de abstracción se utiliza para mostrar el efecto deseado en algún módulo sin necesidad de definir el mecanismo exacto del control. Cada módulo independiente del sistema oculta los detalles internos de sus procesos, comunicándose los módulos solamente a través de interfaces bien definidas.

Estos mecanismos de abstracción permiten controlar sistemáticamente un proceso muy complejo por medio de procedimientos más particulares, llevando de lo abstracto a lo concreto.

2.5.3 ESTRUCTURAS TIPICAS.

El concepto de estructura de datos proporciona una relación lógica entre cada elemento individual de datos. La estructura de datos define la organización de cómo se almacenan los datos, los métodos de acceso a los datos, el grado de asociatividad entre los datos y la forma de procesar la información. La organización y complejidad de una estructura de datos está sólo limitada a la creatividad e ingenio del diseñador.

La estructura de datos más sencilla es el elemento escalar, éste representa un elemento simple de información, el cual puede ser direccionado mediante un solo identificador, como se muestra en la FIG. 2.23.

FIG. 2.23. Elemento escalar.

Cuando varios elementos escalares se organizan en una lista o grupo ordenado de forma contigua, se forma un elemento denominado vector secuencial. Este tipo de elemento es el más común dentro de las estructuras de datos, abre las puertas a la indexación de variables de información, ver la FIG. 2.24.

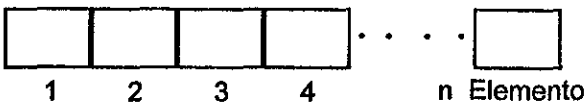


FIG. 2.24. Vector secuencial.

En el momento en que un vector secuencial se extiende a dos, tres y finalmente a un número arbitrario de dimensiones, se crea un espacio n-dimensional, como se observa en la FIG. 2.25.

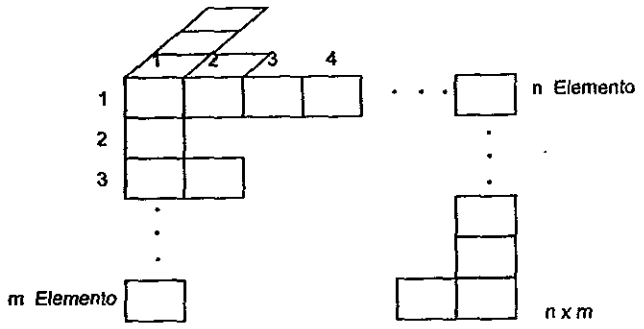


FIG. 2.25. Espacio n-dimensional.

Los vectores y espacios n-dimensionales se pueden organizar de diferentes maneras, por ejemplo, una lista enlazada, es la estructura de datos que está formada por elementos escalares, vectores o espacios no contiguos, a cada elemento enlazado se le denomina nodo, los cuales pueden ser procesados como una lista. Cada nodo por separado está formado por una estructura de datos apropiada y uno o más apuntadores indican la dirección de memoria del siguiente nodo de la lista. Una estructura de datos jerárquica se implementa usando listas multienlazadas que contengan elementos escalares, vectores y posiblemente espacios n-dimensionales.

Una estructura jerárquica se encuentra frecuentemente en aplicaciones que requieren ordenar por categorías la información, en este caso se busca la forma de asociar los datos para obtener los resultados necesarios. Es importante observar que las estructuras de datos y las estructuras de los programas, pueden representarse a diferentes niveles de abstracción.

Una estructura de datos jerárquica se implementa usando listas multienlazadas que contienen elementos, vectores, y posiblemente, espacios n-dimensionales. Una estructura jerárquica se encuentra frecuentemente en las aplicaciones que requieren categorización y asociatividad de la información.

Con la estructura del programa se define la jerarquía de control, sin tomar en cuenta las decisiones y secuencias de procesamiento interno de los procedimientos. En esta estructura se enfocan los detalles del procesamiento y el flujo de datos de cada módulo individual. Además debe tener una especificación precisa del flujo de la información, la cual debe incluir la secuencia de sucesos, los puntos de decisión, las operaciones repetitivas e incluso la organización de la estructura de los datos, es por esto, que existe una relación entre estructura y procedimiento.

Dentro de cada módulo de la estructura del programa se debe incluir una referencia a los módulos subordinados. Esto es, la representación por medio de procedimientos de programación que se realiza por capas como ilustra la FIG. 2.26.

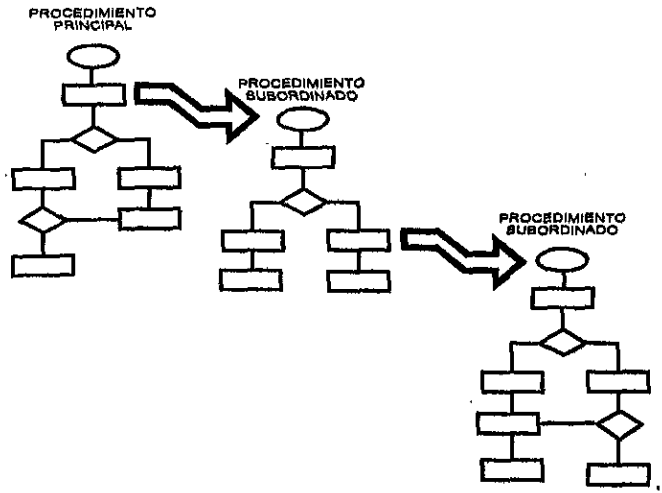


FIG. 2.26. Procedimientos en capas.

2.5.4 MODULARIDAD.

La modularidad de un sistema consiste en dividir al sistema en partes más pequeñas llamadas módulos, los cuales deben de ser independientes entre sí para trabajarse por separado. Por ejemplo, los módulos que pueden ser compilados por separado y su división no depende de un determinado número de líneas. Esta división en partes más pequeñas ayuda a integrar a los módulos sin tener que afectar el funcionamiento de los otros módulos integrantes del sistema.

Uno de los objetivos de la modularidad es la posible reducción del costo de elaboración de programas, ya que puede dividirse el trabajo entre varios programadores para reducir el tiempo de desarrollo.

En el diseño de la carta de estructura está implícito el concepto de modularidad. Esto es, el software se divide en elementos con nombres y direcciones separadas, llamándose módulos, los cuales se integran para realizar los requerimientos del sistema.

Dentro del ambiente del diseño estructurado se maneja que *"la modularidad es el atributo más sencillo del software que permite a un programa ser manejable intelectualmente"* [21] . Esto es que los programas modulares se pueden manejar más fácilmente por los desarrolladores. A diferencia de un programa que esté integrado por un bloque de instrucciones, éste no sería fácil de entender por una sola persona. Ya que seguir el flujo de control de instrucciones o el número de variables se dificulta debido a la complejidad global del programa, esto puede hacer imposible el entendimiento total de dicho programa.

Al considerar una solución modular para algún problema, ésta se puede descomponer en varios niveles de abstracción. El primer nivel de abstracción es el procedimental, aquí se establece la solución al problema con enunciados amplios, en donde se utiliza un lenguaje relacionado con el entorno del mismo. El siguiente nivel de abstracción es el de datos, esta abstracción se orienta hacia los procedimientos, en este caso se utiliza una terminología de implementación más técnica de algún lenguaje de programación. Finalmente, al alcanzar el nivel más bajo de abstracción se genera el código fuente del sistema, el cual es escrito en código de algún lenguaje de programación. Existe otra forma de abstracción a la que se le denomina de control, esta abstracción implica métodos de control del programa, en donde no se explica nada de los detalles internos de los procedimientos y subrutinas del programa.

Para lograr que el sistema sea modular, se requiere utilizar una serie de refinamientos sucesivos de los procedimientos, éstos van desde una descripción a grandes rasgos de la solución del problema, hasta refinarlo en una representación más abstracta de algún lenguaje de programación.

[21] Pressman Rogers; Ingeniería de software. Un enfoque práctico; McGraw Hill, 1993, México, pág. 332.

Conforme se avanza en el proyecto, cada nivel de abstracción del programa representa un refinamiento.

Ocultamiento de la información.

Este concepto hace referencia a una característica importante que deben cumplir los módulos, esto es, que se deben diseñar de forma que los procedimientos y datos que lo integran sean inaccesibles por otros módulos que no necesiten tal información. El ocultamiento implica que una modularidad efectiva se logra definiendo un conjunto de módulos independientes, que se comuniquen con otros sólo por la información necesaria para la ejecución de las funciones del software. El uso del ocultamiento de la información para el diseño de sistemas modulares, presenta algunas ventajas, por ejemplo, cuando es necesario realizar modificaciones durante la etapa de pruebas y posteriormente en el programa de mantenimiento.

Un diseño modular reduce la complejidad del problema en pequeños bloques, esto facilita los cambios a los programas, por lo cual a futuro el mantenimiento es más fácil y da como resultado una implementación más rápida y sencilla, abriendo la posibilidad para el desarrollo en paralelo de diferentes partes del sistema. Dentro del diseño estructurado existen varios tipos de módulos, éstos pueden ser caracterizados como sigue:

- Los módulos secuenciales se ejecutan sin interrupción aparente.
- Los módulos incrementales que pueden ser interrumpidos antes de terminar de ejecutar la aplicación y después se restablecen en el punto donde se interrumpió.
- Los módulos paralelos que ejecutan procesos simultáneamente con otros módulos, éstos se manejan en entornos de multiprocesadores concurrentes.
- Los módulos independientes los cuales se ejecutan de forma separada de los otros módulos sin importar el orden o el tiempo de ejecución.

Independencia funcional.

Este concepto es el resultado directo de aplicar la modularidad y la abstracción en el desarrollo del programa, implícitamente se maneja otro concepto importante en el diseño estructurado, éste es el ocultamiento de la información. La independencia funcional se consigue en el momento en que se desarrollan módulos con una función bien definida, para esto es necesario diseñar el software de forma que cada módulo realice una subfunción específica, la cual tenga una interface sencilla con los otros módulos del sistema. Algunos autores consideran que "*la independencia funcional es la clave de un buen diseño, y el diseño es la clave de la calidad del software.*" [22]

Algunos de los criterios con los que debe cumplir un sistema modular para tener una independencia funcional son los siguientes:

[22] Pressman Rogers; Ingeniería de software, Un enfoque práctico; McGraw Hill, 1993, México, pág. 349.

- Cada abstracción de un proceso, debe ser un subsistema bien definido y con el potencial de ser útil para otras aplicaciones.
- Cada función de cada abstracción tiene un propósito claramente definido.
- Cada función maneja no más de una estructura de datos principal del sistema.
- Las funciones que compartan datos globales lo realicen de forma selectiva.

Es importante mencionar que la independencia funcional entre módulos se determina usando dos criterios cualitativos: cohesión y acoplamiento, los cuales serán explicados posteriormente. Para considerar que los módulos son *independientes* se deben manejar como una entidad bien definida. A continuación se describen algunas de las características que deben cumplir:

"- Los módulos contienen instrucciones, lógica de proceso y estructuras de datos.
- Los módulos pueden ser compilados aparte y almacenados en una biblioteca.
- Los módulos pueden quedar incluidos dentro de un programa.
- Los segmentos de un módulo pueden ser utilizados por medio de invocar un nombre con algunos parámetros.
- Los módulos pueden usar a otros módulos."[23]

El desarrollo de un sistema modular permite al diseñador ordenar de manera jerárquica el uso de las funciones implementadas. Además, permite que se generen abstracciones de datos y el desarrollo de subsistemas independientes.

[23] Fairley Richard; Ingeniería de Software; McGraw Hill, 1988, México, pág.155.

2.5.5 COHESION.

El concepto de cohesión se puede entender como la medición de la fuerza de asociación de los elementos internos de los módulos, esto se toma como una extensión del ocultamiento de información. Idealmente un módulo coherente ejecuta una sola tarea y la realiza por medio de un procedimiento de programación, el cual necesita poca interacción con otros procedimientos que se ejecuten en otras partes del programa. En otras palabras, en situaciones ideales el módulo coherente sólo debe realizar una única tarea.

La cohesión es una propiedad que se utiliza para medir la fuerza funcional de un módulo individual. Esto es, medir en qué grado, el módulo realiza una sola función dentro del sistema y al terminar de ejecutarla, pasa el resultado a otro módulo. La cohesión interna de un módulo se mide en términos de la fuerza de unión de los elementos dentro del módulo; esta cohesión ocurre de una escala de la más débil (la menor deseada) a la más fuerte (la más deseada) en el siguiente orden:

1. Cohesión coincidental.
2. Cohesión lógica.
3. Cohesión temporal.
4. Cohesión en la comunicación.
5. Cohesión secuencial.
6. Cohesión funcional.
7. Cohesión informacional.

Cohesión coincidental.

Se presenta cuando los elementos internos de un módulo no tienen relación aparente entre ellos; por ejemplo, cuando un programa monolítico de gran tamaño es modularizado, esto es, separarlo en varias partes de forma arbitraria y considerarlas como módulos de menor tamaño. También es cuando se crea un módulo con un conjunto de instrucciones que no tengan una relación aparente. Es por esto que un módulo que ejecute un conjunto de tareas que estén relacionadas con otras débilmente, tiene una cohesión coincidental.

Cohesión lógica.

Este tipo de cohesión implica que la existencia de alguna actividad similar entre los elementos del módulo, por ejemplo, un módulo que desempeñe todas las funciones de entrada/salida y otro que se dedique a la edición general de los datos. Un módulo con cohesión lógica comúnmente combina varias funciones interrelacionadas en otros módulos.

Cohesión temporal.

En este tipo de cohesión todos los elementos son ejecutados en un momento dado sin requerir de ningún parámetro o lógica alguna para determinar qué elementos deben ejecutarse.

Cohesión en la comunicación.

Esta cohesión se presenta cuando algunos elementos de un módulo desarrollan diferentes funciones, pero cada función es alimentada por la misma entrada de datos. En otras palabras, esto es cuando varios elementos de un módulo se concentran sobre un área de una estructura de datos, en estos casos se presenta una cohesión de comunicación.

Cohesión secuencial.

Esta cohesión ocurre cuando la salida de un elemento es la entrada para el siguiente. Cuando los elementos de procesamiento de un módulo están relacionados y deben ejecutarse en un orden específico, existe cohesión secuencial.

Cohesión funcional.

Esta cohesión representa un tipo fuerte, por lo tanto es deseable que exista, aquí todos los elementos de un módulo deben de estar relacionados al desempeño de una sola función. Si el módulo realiza una sola función se dice que tiene una cohesión funcional.

Cohesión informacional.

Esta cohesión ocurre cuando el módulo contiene una estructura de datos compleja.Esta cohesión es la realización total de la abstracción de los datos.

2.5.6 ACOPLAMIENTO.

El acoplamiento es una medida de la interconexión entre módulos en una estructura de programa. El acoplamiento depende de la complejidad de las interfaces entre los módulos, además del punto donde se hace referencia a una entrada o algún módulo. También se considera a los datos que pasan a través de las interfaces. Al desarrollar un producto de software se busca que cumpla con el acoplamiento más bajo posible. La conectividad sencilla entre módulos da como resultado un software que sea más fácil de comprender y menos propenso a efecto onda, causado cuando los errores ocurren en una parte del programa y se propagan a lo largo del sistema.

La fuerza del acoplamiento entre dos módulos está influida por la complejidad de la interfaz y el tipo de comunicación existente entre ellos; a medida que es menor la complejidad de los elementos, se obtienen relaciones más sencillas de comprender, a diferencia de elementos más grandes y complejos que se hacen más difíciles en su compresión. Por ejemplo, la modificación de un bloque común de datos o de control puede requerir de modificaciones en todas las rutinas que se encuentran acopladas a ese bloque; por otro lado, si los módulos se comunican, solamente por los parámetros y si las interfaces entre módulos permanecen constantes, los detalles internos de los módulos pueden ser modificados sin tener que modificar las rutinas que usan los módulos modificados.

El acoplamiento entre módulos puede ser considerado en una escala del más fuerte (el menos deseable) al más débil (el más deseable) de la siguiente forma:

1. Acoplamiento por contenido.
2. Acoplamiento por zonas compartidas.
3. Acoplamiento por control.
4. Acoplamiento por zona de datos.
5. Acoplamiento por datos.

Acoplamiento por contenido.

Este se presenta cuando un módulo modifica los valores locales o las instrucciones de otro módulo, este acoplamiento se puede presentar en programas en lenguaje ensamblador.

Acoplamiento por zonas compartidas.

Los módulos son amarrados en forma conjunta por medio de zonas globales para las estructuras de datos, en este caso la misma zona está disponible para ser utilizada por otros módulos.

Acoplamiento por control.

Este acoplamiento incluye el paso de banderas de control y de parámetros en forma global entre los módulos, de tal forma que un módulo controla la secuencia de un proceso a otro.

Acoplamiento por zonas de datos.

Es similar al de zonas compartidas, excepto que los elementos globales son compartidos en forma selectiva entre las diversas rutinas que requieren los datos.

Acoplamiento por datos.

Incluye el uso de listas de parámetros para pasar a los elementos entre rutinas. La forma más deseada de acoplamiento es ciertamente una combinación de zonas de datos y de acoplamiento de datos.

2.6 REDES DE COMPUTADORAS Y ARQUITECTURA CLIENTE-SERVIDOR.

2.6.1 REDES DE COMPUTADORAS Y ELEMENTOS BASICOS QUE INTEGRAN UNA RED.

Las redes locales surgen ante la necesidad de compartir recursos entre varios usuarios de microcomputadoras, entre los recursos para compartir se cuentan los servicios de impresión, almacenamiento masivo de información, intercambio de archivos etc. Ante esta necesidad se tuvo que crear una infraestructura que permita la utilización de dispositivos periféricos entre varios usuarios, los cuales se puedan compartir. Entre los recursos a compartir se cuentan los siguientes:

Dispositivos:	Impresora, fax, módem.
Información:	Archivos para uso compartido, bases de datos.
Aplicaciones:	Software de aplicación (cliente-servidor), software de comunicación como correo electrónico.

Para poder analizar a las redes de computadoras se tienen que entender algunos conceptos. A continuación definiremos el concepto de redes de computadoras:

Red de computadoras.

Una red de computadoras es un sistema de comunicaciones e intercambio de datos, creado para conectar físicamente a dos o más computadoras mediante tarjetas de interfaz para red y cables, además de ejecutar un software denominado sistema operativo para red (NOS).

Una red de computadoras está compuesta por hardware y software, entre el hardware se tienen las tarjetas de red, el medio de comunicación que une al servidor con las estaciones de trabajo (cable, fibra óptica, etc.), concentradores, etc. y el software abarca el sistema operativo de la red (NOS), programas controladores de periféricos y programas de aplicación que aprovechen los recursos de la red.

Elementos básicos que integran una red de computadoras.

NOS (Network Operating System).

Es el sistema operativo para la red, es el software de sistema de una red de área local (LAN) que integra los componentes de hardware de la red. Es común que incluya características como interfaz de administración controlada por menús, respaldo en cinta de la información del servidor de archivos, restricciones de seguridad, medios para compartir impresoras, almacenamiento central de programas de aplicación y bases de

datos, registro remoto a través de módem y soporte para las estaciones de trabajo sin disco.

Este sistema operativo establece y mantiene la conexión entre las estaciones de trabajo y el servidor de archivos; para soportar el trabajo en red, no basta con las conexiones físicas. Está compuesto por dos partes: el software del servidor de archivos y el de la estación de trabajo.

Servidor.

Es el equipo de cómputo central de la red, en este equipo se ejecuta el sistema operativo de red (NOS) el cual ofrece los servicios de red a las estaciones de trabajo, entre los servicios ofrecidos se encuentran, el almacenamiento de archivos, la gestión de usuarios, la seguridad, etc.

Nodos de la red.

Son las computadoras que están conectadas como estaciones de trabajo a la red.

Tarjetas de interface de red.

Este tipo de tarjeta permite la conexión física al medio de comunicación utilizado en la red, además de soportar una topología de red para su conexión. La tarjeta incluye circuitos electrónicos de codificación y decodificación y un dispositivo de recepción para la conexión de un cable para red. Como la información se transmite con más rapidez dentro del bus interno de la computadora, la tarjeta de interfaz para red permite que la red opere a velocidades mayores que las que se alcanzarían si las demorara el puerto serial.

Medio de transmisión.

Es el medio físico por el cual las estaciones de trabajo pueden comunicarse con el servidor, este puede ser cable coaxial, fibra óptica o cable telefónico, entre otros.

Concentradores.

Son los equipos de comunicación necesarios para la conexión remota de las estaciones de trabajo, este dispositivo se encarga de concentrar todas las señales del servidor y de las estaciones de trabajo.

Recursos y periféricos compartidos.

Estos son los dispositivos que van a estar disponibles para ser compartidos entre los usuarios de la red, esto incluye dispositivos de almacenamiento conectados al servidor, las unidades de disco óptico, impresoras etc.

2.6.2 CLASIFICACION DE LAS REDES DE COMPUTADORAS Y TOPOLOGIAS DE REDES.

Una clasificación de redes de computadoras en cuanto a sus alcances, es como a continuación se describe:

Redes de área local (LAN).

Son las redes más pequeñas, éstas pueden conectar desde 2 hasta 75 computadoras dependiendo del equipo con que se cuente, mientras que otras pueden conectar 75 o más computadoras. Estas se pueden encontrar dentro de un mismo edificio o en una área pequeña.

Redes de cobertura amplia (WAN).

Estas redes son más grandes que las LAN, emplean líneas telefónicas u otros medios de comunicación a larga distancia para enlazar computadoras separadas por decenas o cientos de kilómetros.

Topología de redes.

El concepto de topología se refiere a la organización geométrica que presenta el cableado de una red. Se puede considerar a la topología como un mapa del cableado, ésta define cómo se lleva el cable a cada estación de trabajo.

Las topologías de red se dividen en dos categorías: la centralizada y la descentralizada. En la primera, por ejemplo una red de estrella, una computadora central controla el acceso a la red. Este diseño garantiza la seguridad de los datos y un control administrativo central sobre el contenido y las actividades de la red, esta topología de estrella será mostrada posteriormente.

En una topología descentralizada, como una red de bus o una de anillo, cada estación de trabajo puede acceder a la red de forma independiente y establecer sus propias conexiones con otras estaciones de trabajo. Esta topología, al igual que la anterior, se explicarán posteriormente.

Topología de bus lineal.

En las redes de Area local, la topología de bus define una disposición de las computadoras, en la cual están conectadas cada una a un segmento común de cable de red, como se muestra en la FIG. 2.27. El segmento de red se coloca como un bus lineal, es decir un cable largo que va de un extremo a otro de la red, el cual conecta cada nodo de red, en este caso en los extremos del cable lleva dispositivos terminadores del segmento. El cable puede ser una combinación de varios segmentos, siempre y cuando sea un segmento continuo.

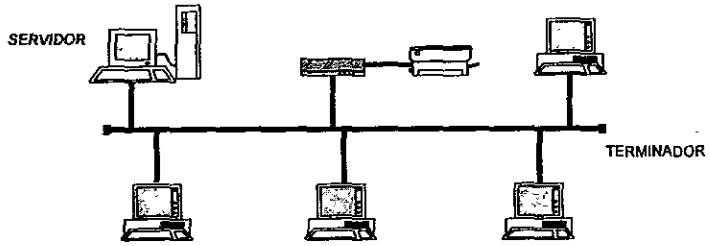


FIG. 2.27. Topología de bus lineal.

Topología de estrella.

En las redes de área local (LAN), es la topología de red centralizada cuya disposición física asemeja una estrella. En el centro de la red se encuentra un concentrador de cableado o HUB, FIG. 2.28; los nodos se conectan directamente alrededor del punto central. En este caso el costo del cableado es mucho más elevado que el costo de otra topología de red, debido a que cada estación de trabajo requiere un cable que conecte directamente al concentrador central.

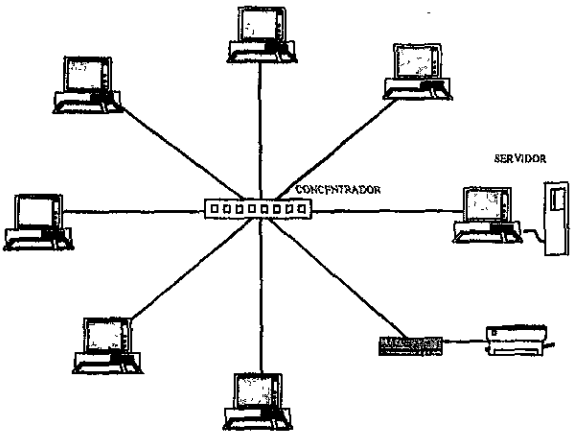


FIG. 2.28. Topología de Estrella.

Topología de Anillo.

En este caso cada computadora se conecta en forma de anillo a la red, como se muestra en la FIG. 2.29. En algunas redes la topología de anillo puede realizarse de manera lógica pero teniendo de manera física la topología de estrella, como en la FIG. 2.30.

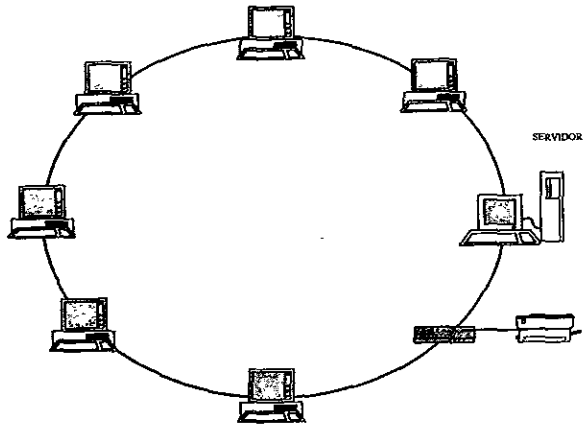


FIG. 2.29. Topología en Anillo.

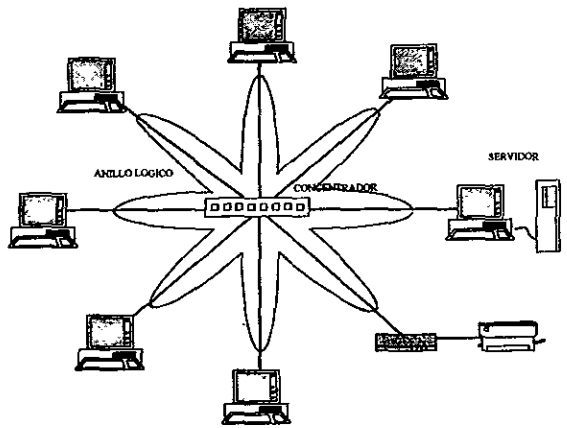


FIG. 2.30. Anillo lógico con estrella física.

Topología doble anillo.

Esta topología es igual a la topología anterior, a diferencia que utiliza dos anillos para su conexión, un anillo es redundante para manejo de la seguridad en los datos, este tipo de topología se utiliza con estándares de fibra óptica (FDDI). Se muestra en la FIG. 2.31.

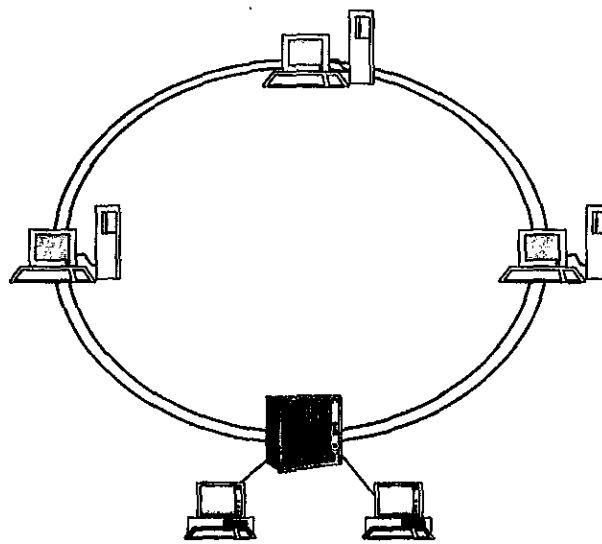


FIG. 2.31. Topología de Doble Anillo.

2.6.3 ESTANDARES DE LAS REDES DE COMPUTADORAS.

Las redes de computadoras se integran de diferentes elementos que deben interactuar en conjunto para poder hacer funcionar la red. Entre estos componentes podemos contar tarjetas de red, cables, conectores, concentradores y el mismo servidor.

Ante la existencia de una gran variedad de fabricantes de estos componentes, surgió la necesidad de un entendimiento y comunicación entre los fabricantes, además de una forma en que cada componente trabaja e interactúa con los demás componentes de la red, a pesar de ser de diferentes fabricantes. Para esto surgieron organizaciones que definen estándares en la industria, estos estándares definen la forma de conectar componentes de hardware en las redes y los protocolos de uso cuando se establece comunicación por red.

En estos casos, toda la información que fluye por el medio de comunicación (cable) de la red debe distinguirse y seguir una secuencia, esto para que las estaciones de trabajo puedan asegurarse de que los datos correctos lleguen al lugar que le corresponda.

Entre estas organizaciones se encuentra IEEE (Institute of Electrical and Electronics Engineers Computer Society), esta organización está integrada por ingenieros, científicos y estudiantes. El IEEE ha declarado estándares para las computadoras y comunicaciones, un estándar de los principales es el IEEE 802 para las redes de área local.

Otra organización el Instituto Nacional Americano de Estándares (ANSI), esta organización es con fines no lucrativos dedicado al desarrollo de normas voluntarias, diseñadas para mejorar la productividad y competitividad internacional de las empresas industriales de Estados Unidos, además es representante de Estados Unidos la Organización Internacional de Estándares (ISO). Esta organización es no lucrativa, teniendo como sede la ciudad de Génova, Suiza, que busca el avance tecnológico y científico estableciendo estándares no exclusivos. Además, es una organización que agrupa estándares de más de 90 países en el mundo, esta agrupación es la responsable del desarrollo del modelo de referencia de interconexión de sistemas abiertos (OSI), este modelo es el medio de conceptualización de las redes de cómputo que ha demostrado tener gran influencia en el desarrollo de tecnología para redes.

Modelo OSI de ISO.

El modelo Open Systems Interconectivity (OSI) es el estándar propuesto por la Organización Internacional de Estándares (International Standard Organization), el cual define los protocolos de comunicación para redes, este modelo consta de siete niveles, los cuales se muestran en la FIG. 2.32.

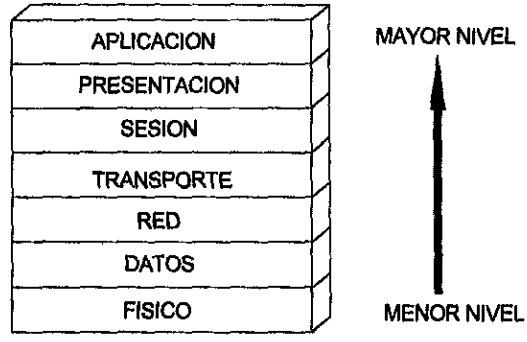


FIG. 2.32. Modelo OSI.

A continuación se describen las funciones de cada nivel del modelo antes mencionado:

Nivel Físico.

Este nivel es el que se encarga de la transmisión de las cadenas de bits por el medio físico, sin ningún formato o estructura de organización de los bits, este nivel es el que interactúa directamente con el medio físico de la red (cable), se muestra en la FIG. 2.33.

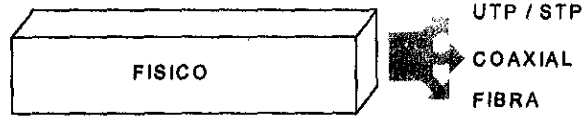


Fig. 2.33. Nivel Físico.

Algunas de las funciones de este nivel son:

- Proporcionar los voltajes y pulsos de cada bit.
- Proporcionar el medio de interface.
- Define el comportamiento de la línea (Full-Duplex, Half-Duplex).
- Asigna los pines correspondientes a los conectores.

Nivel de Datos.

Este nivel es el que provee la transmisión de la información "sin errores" sobre el medio físico de la red (cable), se muestra en la FIG. 2.34.

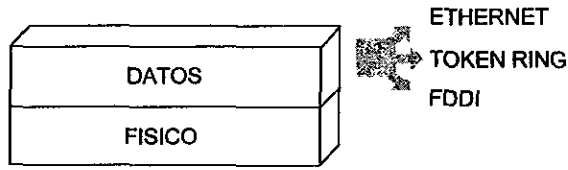


FIG. 2.34. Nivel de Datos.

Este nivel realiza lo siguiente:

- Crea y reconoce el tamaño de los paquetes (Frame).
- Checa la integridad de los mensajes.
- Administra el acceso y flujo del canal.
- Asegura que el orden de los datos transmitidos sea correcto.
- Detecta y corrige errores sin utilizar los niveles superiores.
- Controla el flujo de datos para no exceder la capacidad del canal.

Nivel de Red.

Este nivel es el que decide qué camino deben de tomar los paquetes de datos dependiendo del estado en que se encuentra la red, además de asignar prioridades de servicio, como se muestra en la FIG. 2.35.

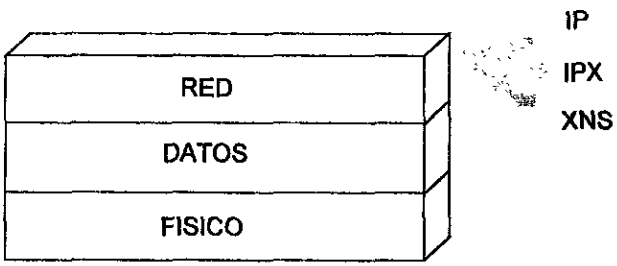


FIG. 2.35. Nivel de Red.

Las funciones que realiza este nivel son:

- Checa los mensajes de direccionamiento.
- Elige el camino de datos entre nodos de diferentes redes interconectadas entre si.
- Controla el congestionamiento si hay muchos paquetes en una subred.
- Convierte direcciones lógicas o nombres en direcciones físicas.

Nivel de Transporte.

Este nivel se asegura que las unidades de datos sean enviadas sin errores, en secuencia, no duplicados y sin pérdidas entre los nodos que se van a comunicar, este nivel provee la calidad e integridad de los datos y se muestra en la FIG. 2.36.

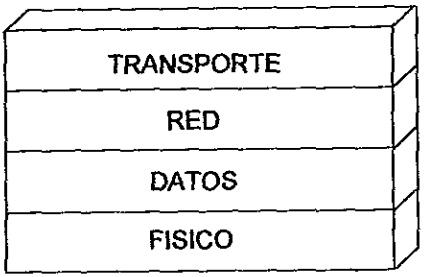


FIG. 2.36. Nivel de Transporte.

Los niveles sesión, presentación y aplicación son los últimos niveles del modelo, interactúan con los gateways y sus funciones son las siguientes:

Sesión	Coordina la interacción entre los procesos finales.
Presentación	Provee conversión de códigos y formatos de datos.
Aplicación	Elige el servicio apropiado para el proceso (interface de usuario).

En la FIG. 2.37 se pueden observar los diferentes dispositivos de red y la capa donde trabajan.

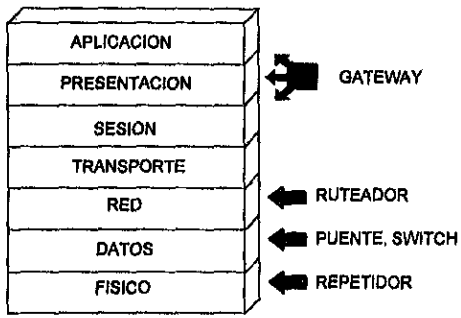


FIG. 2.37. Dispositivos de red.

Protocolo.

Es el conjunto de reglas que define la forma en que deben efectuarse las comunicaciones de las redes. incluyendo la temporización, la secuencia, la revisión y la corrección de errores.

Estándar.

Es la especificación de red que se adopta, ésta incluye las guías y reglas que deben tener los componentes, así como los protocolos de comunicación que se deben utilizar.

Ethernet.

A este protocolo se le conoce como IEEE 802.3, es el estándar más popular en las redes LAN.

Este estándar utiliza una topología lógica de bus y una topología física de estrella. Este protocolo tiene una velocidad de transmisión de información a través de la red de 10 Mbps. También, utiliza un método de transmisión de datos conocido como Acceso múltiple con detección de portadora y detección de colisiones (CSMA/CD).

Thinnet (10Base2).

En ocasiones se le denomina 10Base2, éste se instala por medio de una topología física de bus, éste consiste en varios segmentos de cable de red que en los extremos tiene sus terminadores, por lo regular en este cableado se utiliza el cable coaxial, que es de menor costo que otros cables.

Par trenzado (10 base-T).

A este estándar se le conoce como UTP(Par trenzado sin blindaje), se instala con una topología de estrella. Cada nodo se conecta a un concentrador (HUB). Este estándar es de mayor costo que el anterior, además de requerir un concentrador adicional para su conexión.

Token ring.

Se le conoce como IEEE 802.5, fue diseñado por IBM, con una velocidad de transmisión de 4 Mbps o 16 Mbps, este estándar utiliza una topología lógica de anillo y una topología física de estrella. Se basa en un esquema de paso de señales (Token passing), es decir que pasa una señal a todas las computadoras de la red, y la computadora que esté en pocesión de la señal tiene autorización para transmitir su información a otra computadora.

FDDI.

Es el estándar de interfaz de distribución de datos por fibra óptica (FDDI), es un estándar para transferencia de información por medio de cable de fibra óptica, se le conoce como ANSI X3T9.5 para FDDI, maneja una velocidad de 100 Mbps, este cable no es susceptible a interferencias eléctricas, su costo es mucho mayor que los estándares antes mencionados.

2.6.4 ARQUITECTURA CLIENTE-SERVIDOR.

Definición.

Cliente-servidor es una arquitectura que permite que un gran número de computadoras personales, estaciones de trabajo, impresoras, servidores de archivos y otros equipos estén interconectados a través de una red, en la cual los clientes y los servidores (de archivos o de impresión) tengan funciones específicas que permitan el mejor rendimiento de ambos y de los programas de aplicación, a la vez de aprovechar todos los recursos disponibles desde cualquier punto de la red.

Los nuevos entornos de cómputo tienden a trabajar bajo este modelo, el cual al utilizar un protocolo totalmente abierto hace que las mini y grandes computadoras no queden aisladas sino que tengan un papel distinto dentro de las necesidades informáticas de las empresas.

Los sistemas de información que trabajan bajo este modelo son generalmente sistemas de procesamiento distribuido, los cuales, al operar con una base de datos distribuida (cuya característica principal es la de tener tablas en diferentes puntos de la red o en diferentes servidores) realizan transacciones tanto en el cliente como en el servidor, quedando los procedimientos de consulta y actualización totalmente transparentes al usuario.

Front-end y Back-end.

En los sistemas desarrollados bajo la arquitectura cliente-servidor el software se divide en dos niveles: al software con el que se construye la interfaz que ve el usuario final, es decir los programas de aplicación que opera el usuario es el front-end y al software con el que se crea y administra la base de datos se le conoce como back-end. Tanto el front-end como el back-end pueden realizar en mayor o menor medida transacciones en la base de datos.

Una verdadera aplicación cliente-servidor delega parte del procesamiento en el cliente, reduciendo así, la carga de trabajo al servidor.

Aunque todavía no se ha establecido una forma precisa de dividir la funcionalidad entre el cliente y el servidor debido a la existencia de diferentes enfoques del modelo, se pueden listar algunas funciones que generalmente le corresponden a cada uno.

2.6.5 FUNCIONES DEL SERVIDOR Y FUNCIONES DEL CLIENTE.

Funciones del servidor.

- Contener el sistema operativo de red que soporte el modelo cliente-servidor.
- Hacer funciones de servidor de archivos y/o servidor de impresión.
- Contener el servidor de base de datos.
- Ejecutar el software del back-end.
- Realizar las transacciones más complejas y que requieran de mayores recursos que tengan que ver con procesos e información que compartan diferentes usuarios.
- Contener los programas de aplicación que serán utilizados por la mayoría de los usuarios.
- Los grandes respaldos de las bases de datos pueden seguirse haciendo desde el servidor.

Funciones del cliente.

- Contener el sistema operativo del cliente.
- Contener la aplicación que utilice únicamente cada cliente.
- Ejecutar la interfaz del usuario front-end.
- Realizar las transacciones que únicamente tienen que ver con procesos e información que no se comparte con otros usuarios.
- En cada cliente se debe replicar o respaldar la parte de la base de datos contenida en cada uno de ellos.

2.6.6 VENTAJAS DE LA ARQUITECTURA CLIENTE-SERVIDOR.

Las ventajas de la arquitectura cliente-servidor son:

- Mejora el desempeño de la red porque al utilizar la capacidad de almacenamiento y procesamiento del cliente disminuye la carga de trabajo del servidor, el cual mejora su rendimiento.

- Permite un mejor aprovechamiento y disponibilidad de todos los recursos de la red, ya que se pueden compartir datos y dispositivos periféricos desde cualquier punto de la red.

- Genera la utilización de software especializado ya que las aplicaciones cliente-servidor dividen el software en dos niveles front-end y back-end, en consecuencia se utilizará un software como servidor de base de datos y otro software para crear la aplicación.

- Se puede hacer una mejor elección del hardware, porque al utilizar los recursos del cliente, en muchos casos la máquina que se va a utilizar como servidor ya no requiere ser tan robusta.

- Al utilizar un protocolo abierto, pueden interactuar una o mas redes con diferentes equipos e inclusive con diferentes sistemas operativos, siempre y cuando sean capaces de manejar el mismo protocolo de comunicaciones.

- El software que opera con el modelo cliente-servidor permite crear bases de datos distribuidas, con lo cual ya no se tiene la información en forma centralizada, de manera que si se daña una máquina o por alguna razón no se puede accesar a ésta, solo se detendrán los procesos para la parte de la base de datos que ahí se encuentre.

2.6.7 ACERCA DEL SOFTWARE PARA ARQUITECTURA CLIENTE-SERVIDOR.

Características que debe tener un software manejador de base de datos para arquitectura cliente-servidor.

Todo servidor de bases de datos para esta arquitectura debe contar con ciertas características que lo hagan seguro, confiable y capaz de poder administrar bases de datos que se encuentren en un solo punto o distribuídas para poder asegurar la integridad de la información:

- La capacidad de tener acceso a *sitios remotos* y *transmitir consultas* y datos entre los diversos nodos de una red.
- La capacidad de seguir la pista a todos los procesos y transacciones en el catálogo del sistema administrador de la base de datos.
- La capacidad de elaborar las estrategias de ejecución para consultas y transacciones que tienen acceso a datos de más de un sitio.
- La capacidad de decidir a cuál copia de un elemento de información replicado se tendrá acceso.
- La capacidad de mantener la consistencia de la copias de un elemento de información replicado.
- La capacidad de recuperarse de caídas de nodos individuales sin afectar a toda la *red*.

Software para arquitectura cliente-servidor.

En la actualidad existe una gran cantidad de software disponible en el mercado tanto de sistemas operativos como de lenguajes de programación y servidores de bases de datos que *pueden trabajar y aprovechar las ventajas de la arquitectura cliente-servidor.* Todo el software que se menciona a continuación en sus últimas versiones cuenta con las características apropiadas para la arquitectura cliente-servidor.

Software para el servidor.

Sistemas Operativos:	WINDOWS NT, NOVELL, OS/2, UNIX, etc.
Servidor de base de datos: (Back-end)	SQL-SERVER, ACCESS, ORACLE, INFORMIX, SYBASE, etc..

Software para el cliente.

Sistemas Operativos:	DOS, WINDOWS 3.x o 95, NOVELL,UNIX, OS/2, etc.
Software para aplicaciones: (Front-end)	VISUAL BASIC, VISUAL FOXPRO, DELPHI, C++, INFORMIX, ORACLE, SYBASE, etc.

2.7 CODIFICACION Y LENGUAJES DE PROGRAMACION.

2.7.1 LA PROGRAMACION SISTEMATICA.

La programación sistemática es una técnica basada en la teoría de crear una estructura de programa a partir de estructuras de control que son métodos para especificar el orden en que las instrucciones de un algoritmo se ejecutarán. Las estructuras de control básico son:

- Secuencia.
- Selección.
- Repetición.

La programación sistemática se auxilia de los recursos abstractos de los que disponen los lenguajes de programación.

"Descomponer un programa en términos de recursos abstractos consiste en descomponer una determinada acción compleja en términos de un número de acciones más simples capaces de ejecutarlas o que constituyan instrucciones de computadoras disponibles".[24]

Se destaca la importancia de las abstracciones de control para crear programas legibles, mantenibles y revisables. En especial de las construcciones para el control de ciclos, las construcciones de decisión y el manejo de excepciones.

Construcciones para el control de ciclos.

Las construcciones para el control de ciclos son aquellas construcciones de control del lenguaje que especifican que una proposición, o grupo de proposiciones, de un programa se deben ejecutar varias veces. El número de veces se puede especificar de manera explícita o puede depender de que una condición se haga verdadera. Las construcciones de ciclos más comunes son las proposiciones **FOR**, **WHILE-DO** y **REPEAT-UNTIL**.

La proposición FOR numérica se usa cuando se quiere ejecutar un ciclo un número determinado de veces. Tiene la forma general:

FOR <contador> := <valor inicial> STEP <incremento> TO <valor final>

Los lenguajes difieren ligeramente en la naturaleza exacta de su proposición FOR. La sentencia FOR requiere que conozcamos por anticipado el número de veces que se ejecutan las sentencias del interior del ciclo. Cuando se ejecuta la sentencia FOR la

[24] Pressman S. Roger; Ingeniería de Software; McGraw-Hill, 1993, México, pág. 358.

primera vez, el valor inicial se asigna a una variable de control, y a continuación se ejecuta la sentencia del interior del ciclo. Otra representación puede ser:

```
FOR <variable> := <expresión> TO <expresión> DO
        <sentencia 1>
        <sentencia 2>
        .
        .
        .
        <sentencia n>
```

Al ejecutar la sentencia FOR, a la variable de control se le asigna el valor inicial; al llegar a la sentencia n se verifica si el valor final es mayor que el valor inicial, en caso negativo se incrementa el valor de la variable de control en uno y se vuelven a ejecutar las sentencias del interior del ciclo hasta que la variable de control sea mayor que el valor final, en cuyo momento se termina el ciclo.

En la estructura FOR es posibe que en ciertas ocasiones se requiera que el contador de la sentencia se decremente en lugar de que se incremente. Esto es posible en algunos lenguajes. El formato es el siguiente:

```
FOR <variable> := <expresión> DOWNTO <expresión> DO <sentencia >
```

En esta variante la sentencia FOR se ejecuta un número determinado de veces (valor inicial - valor final + 1). El funcionamiento del FOR-TO-DOWNTO está sujeto a varias reglas que son:

1. Las variables de control, valor inicial y valor final deben ser del mismo tipo. Los valores iniciales y finales pueden ser tanto expresiones como constantes.
2. Antes de la primera ejecución del ciclo, a la variable de control de asigna el valor inicial.
3. La última ejecución del ciclo normalmente ocurre cuando la variable de control es igual al valor final.
4. Cuando se utiliza TO, la variable de control se incrementa, si se utiliza DOWNTO, la variable se decrementa.
5. No debe modificarse el valor de la variable de control, el valor inicial y el valor final dentro del ciclo.
6. El valor de la variable de control queda indefinido cuando se termina el ciclo.

La proposición FOR suele ser una construcción muy segura, porque, en la mayoría de los lenguajes, se garantiza la terminación del ciclo. Sin embargo, en algunos casos se pueden modificar los parámetros de la proposición por medio de proposiciones en el ciclo, de modo que es posible construir ciclos FOR infinitos.

En otros este problema se evita al considerar el contador, el valor inicial y el valor de terminación del ciclo como parámetros del ciclo FOR, ya que los parámetros se evalúan

por valor, lo cual significa que los cambios que se hagan a éstos son sólo locales al ciclo.

Un diseño más satisfactorio de la proposición FOR hace que el contador del ciclo sea redeclarado implícitamente como una constante después de cada ejecución del ciclo.

Las proposiciones WHILE y REPEAT permiten al usuario especificar que el ciclo se debe ejecutar hasta que alguna condición sea verdadera, o mientras lo sea.

WHILE <condición> DO <proposición>
REPEAT <proposición> UNTIL <condición>

Estos tipos de ciclo no tienen los mismos problemas que algunos diseños de la proposición FOR, porque la variable de control es manejada por el programador. La proposición WHILE permite al programador poner la condición de terminación del ciclo al principio de éste y la proposición REPEAT le permite hacerlo al final.

Ninguna de las dos construcciones permite poner la condición de terminación en la mitad del ciclo.

La proposición WHILE es aquella en la que el número de iteraciones no se conoce por anticipado y el cuerpo del ciclo se repite mientras se cumple una determinada condición.

Cuando la sentencia WHILE se ejecuta, la primera cosa que sucede es la evaluación de la expresión lógica. Si se evalúa falso, ninguna acción se realiza y el programa prosigue en la siguiente sentencia después del ciclo. Si la expresión lógica se evalúa verdadero, entonces se ejecuta el cuerpo del ciclo y se evalúa de nuevo la expresión. Este proceso se repite mientras que la expresión lógica permanezca verdadera. Sus reglas de funcionamiento son:

1. La condición se evalúa antes y después de cada ejecución del ciclo. Si la condición es verdadera, se ejecuta el ciclo, y si es falsa, el control pasa a la sentencia siguiente del ciclo.
2. Si la condición se evalúa falso cuando se ejecuta el ciclo por primera vez, el cuerpo del bucle no se ejecutará nunca.
3. Mientras la condición sea verdadera el ciclo de ejecutará. Esto significa que el ciclo se ejecutará indefinidamente a no ser que alguna condición se modifique dentro del mismo haciendo que su valor pase a falso.

La sentencia REPEAT especifica un ciclo condicional que se repite hasta que la condición se haga verdadera. Después de cada iteración el ciclo evalúa la condición. Si la condición es verdadera, el ciclo se termina y se sale de él, ejecutándose la sentencia siguiente. Si la condición es falsa, el cuerpo del ciclo se repite. Sus reglas de funcionamiento son:

1. La condición se evalúa al final del ciclo, después de ejecutarse todas las sentencias.
2. Si la expresión lógica es falsa, se vuelve a repetir el ciclo y se ejecutan todas las sentencias.
3. Si la expresión lógica es verdadera, se sale del ciclo y se ejecuta la siguiente sentencia a ejecutar.

Existen algunas situaciones de programación donde es necesario poner la prueba de terminación dentro del ciclo. Existe también la alternativa de utilizar variables boolenas o terminadores.

Existen algunas otras opciones para terminar ciclos como son las instrucciones: LOOP y EXIT.

La proposición EXIT hace que el control se transfiera a la proposición que sigue al ciclo, a menos que vaya seguida de un identificador. En este caso, el control se transfiere a la proposición que sigue al ciclo, etiquetada con ese identificador.

Construcciones de decisión.

Las construcciones de decisión permiten seleccionar una proposición o un grupo de ellas para su ejecución partiendo de la base de que alguna condición sea verdadera. Estas abarcan el condicional unilateral (**IF THEN**), el condicional bilateral (**IF THEN ELSE**) y el condicional multilateral (**CASE**). La forma de estos condicionales es:

IF <condición> THEN <proposición>
IF <condición> THEN <proposición> ELSE <proposición>

La estructura IF-THEN se utiliza cuando en algunos casos se desea que se ejecute una determinada acción, y no realizar nada si la condición es falsa.

La estructura IF-THEN-ELSE funciona de la siguiente forma:

1. Se evalúa la expresión lógica.
2. Si la expresión toma el valor verdadero, se ejecutará la sentencia A y el control pasará a la sentencia inmediata siguiente a la IF-THEN-ELSE.
3. Si la expresión toma el valor falso, entonces sólo se ejecutará la sentencia B y el control pasa inmediatamente a la siguiente sentencia del programa.

La sentencia CASE se utiliza para elegir entre diferentes alternativas. Una sentencia CASE se compone de varias sentencias simples. Cuando CASE se ejecuta, una (y sólo una) de las sentencias simples se selecciona y ejecuta.

Existen algunas reglas y son las siguientes:

1. La expresión selector se evalúa y se compara con las listas de constantes. Sólo se ejecuta una sentencia.

2. La cláusula ELSE es opcional como en la sentencia IF.
3. Si el valor del selector no está comprendido en ninguna lista de constantes y no existe la cláusula ELSE, no sucede nada y sigue el flujo del programa; si el valor del selector no coincide con alguna constante, se ejecuta la sentencia a continuación de la cláusula ELSE.
4. En algunos lenguajes el selector debe ser un tipo ordinal, no pueden utilizarse valores reales.
5. Todas las constantes CASE deben ser únicas y de un tipo ordinal compatible con el tipo del selector.

La proposición CASE tiene diversas formas. Esta incluye varias proposiciones ejecutables por separado. Si el selector de CASE tiene el valor N, para la ejecución se escoge la N-ésima proposición.

```
CASE <expresión entera> OF
        <S1>
        <S2>
        ...
        ...
        <Sn>
```

Después de haberse ejecutado la expresión seleccionada, la ejecución continúa con la proposición que sigue a la del CASE. Esta forma presenta dos problemas:

1. No se especifica la acción a tomar si el valor de la expresión del CASE es menor que cero o mayor que N.
2. El orden correcto de las proposiciones a seleccionar por la expresión CASE es crucial. Ya que un error cometido en este orden no se puede detectar ni al tiempo de compilación ni al de ejecución.

En otros casos, la expresión selectora CASE puede evaluar cualquier escalar a excepción de los reales. Cada proposición ejecutable del CASE recibe una o más etiquetas constantes especiales. La ejecución de la proposición implica evaluar la expresión seleccionada, comparar ese valor con las etiquetas de dicha proposición y seleccionar para su ejecución la que coincida con el valor del selector CASE.

En algunos casos es posible etiquetar una expresión para validar el caso de que la expresión a evaluar no coincida con las propuestas en las ordenes CASE.

"La proposición CASE es, en esencia, una abstracción sobre una condición y permite expresar de la misma manera todas las proposiciones condicionales"[25]

[25] Pressman S. Roger; Ingeniería de Software; McGraw-Hill, 1993, México, pág. 359.

2.7.2 LAS HERRAMIENTAS DE PROGRAMACION.

Las herramientas de programación son aquellas que permiten a los desarrolladores de software editar, interpretar, compilar, probar y depurar, todos los programas fuentes de una aplicación o un sistema además de detectar errores y hacer comparaciones, todo esto con la finalidad de que cada vez sea más fácil, rápido y agradable el desarrollo de los programas.

También cabe mencionar que existen diversos tipos de herramientas, desde editores y compiladores hasta generadores de código automático. En la actualidad están disponibles herramientas de programación orientadas a los objetos y herramientas y lenguajes de programación de cuarta generación, ambos son de las tecnologías más actuales en la ingeniería de software y se componen de una gran variedad de productos, los cuales hacen que cada vez menos código sea escrito manualmente.

Además, el objetivo final del CASE (Ingeniería de Software Asistida por Computadora) es la generación del código automático, esto es, la representación de sistemas a un nivel de abstracción más alto que el de los lenguajes de programación convencionales. Idealmente, estas herramientas de generación de código no sólo traducirán la descripción de un sistema a un programa operativo, sino que también ayudarán a verificar la corrección de la especificación del sistema, de tal forma que la salida resultante satisfaga los requisitos del usuario.

Entre las diferentes herramientas de programación tenemos:

- Editores de texto.
- Comparadores.
- Manejadores de bases de datos.
- Lenguajes de consulta de bases de datos.
- Generadores de aplicaciones.
- Intérpretes.
- Compiladores.
- Depuradores.
- Editores de enlace.
- Lenguajes de alto nivel.
- Lenguajes de cuarta generación.
- Lenguajes de programación orientados a objetos.

Editores de texto.

Un editor de texto es un programa empleado para crear y manipular archivos de texto tales como archivos de programas fuente, a diferencia de los procesadores texto, no disponen de características elaboradas para formateado e impresión. Los editores de texto diseñados para programación, poseen características especiales tales como sangrado automático, numeración de líneas, identificación de palabras reservadas, etc.

Comparadores.

Son programas cuya función primordial es la de comparar dos o más archivos e informar las diferencias existentes entre ellos. Pueden comparar en base a un patrón específico o en un determinado bloque de texto, o bien, simplemente indicar si los archivos tienen por lo menos una diferencia sin importar cuál es.

Manejadores de bases de datos.

Son sistemas que administran una o más bases de datos. Estos sistemas controlan la organización, recuperación, seguridad e integridad de los datos. Además, en muchos casos, permiten a programas de aplicación accesar a los datos.

Lenguajes de consulta de bases de datos.

Son lenguajes que contienen instrucciones que permiten manipular los datos de la base de datos, a fin de convertir dichos datos en información. Por lo general, estas instrucciones clasifican, ordenan y muestran la información en diferentes formatos, además de generar reportes y totalizar registros y campos dentro de la base de datos.

Generadores de aplicaciones.

Software que genera programas de aplicación a partir de descripciones del problema en lugar de una programación detallada. Es uno o varios niveles más altos que un lenguaje de programación de alto nivel, sin embargo, requiere que el usuario introduzca expresiones para describir funciones complejas. Son entornos de programación que generan aplicaciones completas y no necesariamente generan el código correspondiente. Estos entornos pueden, a partir de una base de datos, generar vistas, pantallas de consulta y actualización, reportes, etc. a partir de fórmulas y además ofrecen múltiples alternativas para administrar la base de datos.

Intérpretes.

Son programas traductores de algún lenguaje de programación. Un interprete traduce y ejecuta las instrucciones de un programa fuente una por una hasta concluir el programa. Un intérprete no genera programas ejecutables, por lo que un programa desarrollado con un intérprete siempre dependerá de éste para poder ser ejecutado. Un programa que se interpreta siempre será menos rápido que su correspondiente programa compilado.

Compiladores.

Software que basado en uno o más programas fuente escritos en un lenguaje de programación de alto nivel genera un programa ejecutable en lenguaje de máquina.

Depuradores.

Son programas que ayudan a encontrar errores en la lógica del programa. Un depurador proporciona formas para detener un programa, ejecutarlo línea por línea o capturar datos en momentos determinados. El depurador puede saltar directamente a las líneas del programa que producen errores.

Editores de enlace.

Son programas que ligan módulos objeto del código compilado para producir un programa ejecutable. También sirve para ligar de modo selectivo los módulos objeto con el fin de incluir uno, varios, o todos los programas objeto dentro de un programa ejecutable.

Lenguajes de programación de alto nivel.

Son lenguajes de programación cuyas instrucciones de alguna manera poseen cierta similitud o se acercan a un lenguaje natural (inglés). Los lenguajes de alto nivel permiten al programador concentrarse en la lógica del problema a resolver en lugar de la arquitectura de la máquina, como se requiere en los lenguajes de bajo nivel o ensambladores. Los lenguajes de alto nivel, son de propósito general.

Lenguajes de programación de cuarta generación.

Los lenguajes de cuarta generación son lenguajes que contienen instrucciones de control y para la representación de estructuras de datos a un nivel de abstracción más alto que en los lenguajes de alto nivel, es decir, estas instrucciones, eliminan la necesidad de especificar los detalles de los algoritmos.

Los lenguajes de cuarta generación son de propósito específico, orientados a la resolución de problemas de áreas determinadas.

Lenguajes de programación orientados a objetos.

Son lenguajes que utilizan la tecnología de programación orientada a objetos, con características como: *encapsulamiento*, *herencia* y *polimorfismo*. Gracias a los lenguajes de programación orientados a objetos pueden aterrizarse en programas el análisis y el diseño orientados a objetos.

Algunos lenguajes orientados a objetos están unidos a lenguajes específicos de alto nivel, como el C++.

2.7.3 CLASES Y CARACTERISTICAS DE LOS LENGUAJES DE PROGRAMACION.

Características de los lenguajes de programación.

Los lenguajes de programación son un vehículo de comunicación entre los humanos y las computadoras. Existen varios aspectos que se toman en cuenta para caracterizar a los lenguajes de programación, como son: el psicológico, el sintáctico/semántico y el de ingeniería, todos ellos en gran medida contribuyen a complementar un lenguaje de programación, con el cual, se aterriza y hace realidad el sistema de información después de haber pasado por las etapas de análisis y diseño.

También cabe mencionar que las características técnicas de un lenguaje pueden influir en la calidad del diseño. Por lo tanto, las características técnicas pueden influir tanto en los aspectos humanos como en los de la ingeniería de software.

Características psicológicas.

El proceso de codificación -codificación mediante un lenguaje de programación- es una actividad humana. Es por ello que las características psicológicas de un lenguaje afectan directamente a la calidad de la comunicación.

El aspecto psicológico de un lenguaje de programación se centra en los aspectos humanos, tales como la facilidad de uso, la simplicidad de aprendizaje, la mejora en fiabilidad, la reducción de la frecuencia de error y el aumento de satisfacción del usuario, mientras se mantiene una garantía de eficiencia de la máquina, de la capacidad del software y de las restricciones del hardware. La ingeniería del software es una actividad intensamente humana, aún considerando que las herramientas automáticas CASE proporcionan una importante ayuda.

Los diseñadores de lenguajes de programación a menudo hacen que tengamos que ajustar nuestro enfoque a un problema, de manera que tenga en cuenta las limitaciones impuestas por ese determinado lenguaje de programación.

Debido a que los factores humanos son de importancia crítica en el diseño de lenguajes de programación, las características psicológicas de un lenguaje tienen una influencia en el éxito del diseño y en la traducción al código e implementación.

Aunque las características psicológicas no se pueden medir de forma cuantitativa, se reconoce su manifestación en todos los lenguajes de programación.

Las características psicológicas son:

- Uniformidad.
- Ambigüedad.
- Lo compacto.
- Localización.
- Linealidad.

Uniformidad.

La uniformidad indica el grado en que un lenguaje usa una notación consistente, aplica restricciones aparentemente arbitrarias o incluye excepciones a reglas sintácticas o semánticas. Por ejemplo, FORTRAN usa el paréntesis como delimitador para índices de arreglos, como modificador de la precedencia aritmética y como delimitador para las listas de argumentos de subprogramas.

Ambigüedad.

La ambigüedad de un lenguaje de programación es percibida por el programador. Un compilador siempre interpreta una sentencia de una única forma, pero el lector humano puede interpretar la sentencia de formas diferentes. Es aquí donde reside la ambigüedad psicológica. Por ejemplo, la ambigüedad psicológica aparece cuando la precedencia aritmética no es obvia:

$$X = X1 / X2 * X3$$

Un lector del código fuente puede interpretar lo anterior como X=(X1/X2)*X3, mientras que otro puede interpretarlo como X=X1/(X2*X3). Otra fuente potencial de ambigüedad es el uso no estándar de identificadores que tienen tipos de datos implícitos. Por ejemplo, en FORTRAN, un identificador KDELTA se asumirá (por defecto) como de características enteras. Sin embargo, una declaración explícita, REAL KDELTA, puede producir confusión debido a la ambigüedad psicológica.

La falta de uniformidad y la presencia de ambigüedad psicológica, normalmente van juntas. Si un lenguaje de programación muestra los aspectos negativos de estas características, el código fuente será menos legible y la traducción desde el diseño más problemática.

Lo compacto.

Lo compacto que sea un lenguaje de programación es un indicativo de la cantidad de información orientada al código que se debe retener en la memoria humana. Entre los atributos del lenguaje que miden lo compacto que es, se encuentran:

- El grado en que un lenguaje soporte las construcciones estructuradas.
- Los tipos de palabras clave y de abreviaturas que se pueden usar.
- La variedad de tipos de datos y las características implícitas.
- El número de operadores aritméticos y lógicos.

- El número de funciones, comandos y procedimientos incorporados.

Localización.

La localización se presenta en los lenguajes de programación cuando las sentencias se pueden combinar en bloques, cuando las construcciones estructuradas se pueden implementar directamente y cuando el diseño y el código resultante son altamente modulares y cohesivos. Una característica de los lenguajes de programación que viola la localización es aquella que induce al manejo de excepciones como por ejemplo la orden GOTO o las instrucciones ON KEY u ON ERROR de algunos lenguajes ya que éstas por lo general realizan llamadas a funciones o ejecutan instrucciones que no se encuentran ubicadas conforme al flujo de control del programa y esto dificulta la localización y el seguimiento del código.

Linealidad.

La linealidad es una característica psicológica que se asocia con el concepto de mantenimiento de un ámbito funcional, es decir, la percepción humana se facilita cuando se encuentra una secuencia lineal de operadores lógicos. Las grandes ramificaciones violan la linealidad del procesamiento, porque al existir diferentes caminos para ciertas condiciones, el código puede no cumplir con la linealidad, por ello es mejor hacer código que sirva para diferentes situaciones durante la corrida o puesta en marcha de un programa. Cabe señalar que para que un código sea o no lineal, influye mucho la forma de programar y las costumbres del programador al hacer estructuras de control. De nuevo, la implementación directa de las construcciones estructuradas ayudan a la linealidad de un lenguaje de programación.

Las características psicológicas de los lenguajes de programación afectan de forma importante a nuestra capacidad de aprenderlos, aplicarlos y mantenerlos. El lenguaje de programación tiene que ver directamente con nuestra forma de pensar sobre los programas y puede además poner el límite en la forma en que nos comunicamos con la computadora.

Características sintáctico/semánticas.

Las características semánticas se refieren a la interpretación y validación del significado de la estructura gramatical de los lenguajes de programación. Los programadores aprovechan el conocimiento semántico al aplicar los métodos de ingeniería de software.

El conocimiento semántico es el más difícil de adquirir y el que intelectualmente es el más importante de aplicar. Todos los pasos de la ingeniería de software que preceden a la codificación hacen un fuerte uso del conocimiento semántico. Los ingenieros de software, los diseñadores, los analistas y los programadores de sistemas deben

aprender de los lenguajes de programación actuales y de los que surjan su información semántica para replantear o mejorar sus ideas acerca del diseño y la implementación de sistemas.

Además es importante señalar que un problema de crisis de software reside en el alcance del conocimiento semántico y la capacidad para aplicarlo. Es por ello que el objetivo de la ingeniería de software es ampliar el conocimiento de la semántica en el desarrollo de software.

Las características sintácticas dependen de cada lenguaje de programación, cada uno de éstos, tiene características específicas en este aspecto.

En el proceso de codificación de programas, se aplica el conocimiento sintáctico, el cual es totalmente instructivo y se aprende de manera rutinaria. Esto se demuestra, al aprender un nuevo lenguaje, ya que al hacerlo, seguramente como parte de ese aprendizaje agregamos nueva información sintáctica a nuestra memoria y terminamos aprendiendo después de cubrir un período determinado de tiempo en el que practicamos el lenguaje con sus nuevas características sintácticas. Es ahí donde nos damos cuenta que los lenguajes de programación pueden tener características sintácticas similares pero no equivalentes.

Características de ingeniería.

Desde el punto de vista de la ingeniería de software las características de los lenguajes de programación se centran en las necesidades que pueda tener un proyecto específico de desarrollo de software. De entre una gran diversidad de requisitos que deben tener los sistemas, se puede establecer un conjunto general de características de ingeniería de los lenguajes de programación:

1. Facilidad de traducción del diseño al código.
2. Eficiencia del compilador.
3. Portabilidad del código fuente.
4. Disponibilidad de herramientas de desarrollo.
5. Facilidad de mantenimiento.

Estas cinco características de los lenguajes deben tomarse en cuenta al llegar al paso de codificación, el cual comienza tras haber definido, revisado y modificado, en caso de ser necesario, el diseño detallado.

1. Facilidad de traducción del diseño al código.

El grado de facilidad de esta característica proporciona una indicación de cómo se aproxima un lenguaje de programación a la representación del diseño. Un lenguaje que implementa directamente las construcciones estructuradas e incluya estructuras de

datos sofisticadas, una entrada/salida especializada, posibilidades de manipulación de bits y construcciones orientadas a los objetos, hará que la traducción del diseño al código fuente sea mucho más fácil.

2. Eficiencia del compilador.

Un compilador de un lenguaje de programación, debe generar programas ejecutables rápidos y que consuman la menor cantidad de memoria posible. Aunque los grandes avances en cuanto a velocidad del procesador y capacidades de memoria han hecho que el código no tan eficiente parezca serlo, es muy importante seguir tomando en cuenta esta característica de los lenguajes porque es algo que todavía no solucionan en su totalidad algunos compiladores de lenguajes de alto nivel.

Además, el objetivo es lograr el mejor rendimiento del software y para resolver este requisito crítico debe elegirse un lenguaje con un compilador optimizado.

3. Portabilidad del código fuente.

Es una característica de los lenguajes que puede interpretarse de tres formas:

- El código fuente puede ser transportado de un procesador a otro y de un compilador a otro sin ninguna o muy pocas modificaciones.
- El código fuente permanece inalterado cuando cambia su entorno de funcionamiento, por ejemplo: al instalar una nueva versión del sistema operativo.
- El código fuente puede ser integrado en diferentes paquetes de software sin que prácticamente se requieran modificaciones debidas a las características propias del lenguaje de programación.

De las tres interpretaciones de portabilidad, la primera es la más frecuente. La estandarización (ya sea por la ISO Organización Internacional de Estándares o por la ANSI Instituto Nacional Americano de Estándares) continúa siendo el principal esfuerzo para la mejora de la portabilidad de los lenguajes de programación.

4. Disponibilidad de herramientas de desarrollo.

La disponibilidad de herramientas de desarrollo puede acortar el tiempo requerido para la generación del código fuente y puede mejorar la calidad del código. Muchos lenguajes de programación pueden ser adquiridos con un conjunto de herramientas que incluyen: compiladores con depuradores, ayudas de formato para el código fuente, facilidades de edición incorporadas, herramientas para el control de código fuente, bibliotecas de subprogramas, correctores, etc. Un lenguaje con herramientas de desarrollo proporciona cierta comodidad y ventaja al programador y representa la posibilidad de ofrecer un producto de mayor calidad.

5. Facilidad de mantenimiento.

La facilidad de mantenimiento del código fuente es críticamente importante para cualquier desarrollo de software. El mantenimiento no se puede llevar a cabo hasta que no se entiende el software. La documentación del diseño proporciona un fundamento para la facilidad de comprensión, ya que el código fuente final debe ser leído y modificado de acuerdo con los cambios en el diseño. La facilidad de traducción del diseño al código es un elemento importante en la facilidad de mantenimiento del código fuente. También se sabe, que las propias características de documentación de un lenguaje tienen una fuerte influencia sobre el mantenimiento.

Elección de un lenguaje.

La elección de un lenguaje para un proyecto específico debe tener en cuenta tanto las características de ingeniería como las psicológicas. Los criterios que se aplican durante la evaluación de los lenguajes son:

- Area de aplicación general.
- Complejidad algorítmica y computacional.
- Entorno en el que se ejecutará el software.
- Consideraciones de rendimiento.
- Complejidad de las estructuras de datos.
- Conocimiento del personal de desarrollo de software.
- Disponibilidad de un buen compilador.

Existen algunos lenguajes que son muy atractivos, pero debe elegirse el que tenga una sólida documentación y un buen soporte para el software y haya sido aplicado anteriormente con éxito. El lenguaje de programación tendrá impacto en la planificación, el análisis, el diseño, la codificación, la prueba y el mantenimiento de todo el proyecto. El papel del lenguaje de programación se debe tener presente en todo momento. Los lenguajes de programación proporcionan los medios de la traducción hombre-máquina; sin embargo, la calidad del resultado final se encuentra fuertemente unida a las actividades de ingeniería del software que preceden y siguen a la codificación.

Clases de los lenguajes de programación.

En la actualidad existen muchos lenguajes de programación que han sido aplicados en el desarrollo de software. Existen diversas clasificaciones de lenguajes, en las cuales, un lenguaje puede aparecer en más de una categoría. Para efecto de separar los lenguajes en clases, los clasificaremos por generaciones, lo que además corresponde a la evolución histórica de los mismos. En la FIG. 2.38 se ilustra esta evolución y clasificación, la cual es sugerida por Pressman.

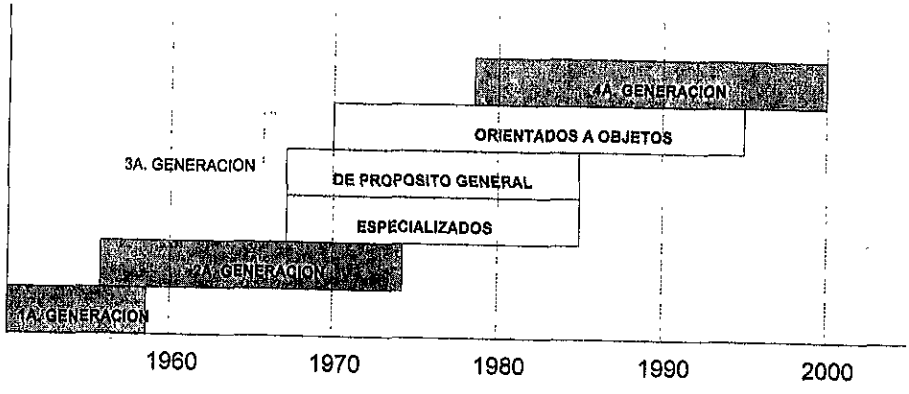


FIG. 2.38. Generaciones de los lenguajes de programación.

Primera generación de lenguajes.

La primera generación de lenguajes de programación es la que abarca los lenguajes de máquina y los ensambladores, lenguajes orientados a las computadoras.

Lenguaje de máquina.

El lenguaje de máquina es un código binario que la computadora interpreta directamente. Para los programadores resulta muy tedioso escribir en este lenguaje ya que requiere de seguimiento a las localidades de almacenamiento de los datos y las instrucciones bajo la forma de cadenas de 0s y 1s. El lenguaje de máquina es eficiente para la computadora, pero muy complicado para el programador y el usuario final. Este lenguaje muy raras veces se utiliza hoy en día para el desarrollo de software de aplicaciones.

Lenguajes ensambladores.

Estos lenguajes se desarrollaron a principios de la década de 1950 para reducir la complejidad y acelerar el proceso lento de desarrollo ocasionado por tener que escribir programas en lenguaje de máquina. Los lenguajes ensambladores están compuestos de códigos de operación llamados mnemónicos y direcciones simbólicas que son más fáciles de manejar que los códigos y direcciones de máquina.

Sin embargo, estas instrucciones deben traducirse a un lenguaje de máquina que la computadora entienda. Esta traducción la realiza un programa ensamblador que convierte el programa fuente en una forma que pueda ser procesada por la máquina, es decir, el programa objeto. Los programas desarrollados a partir de lenguajes

ensambladores son altamente eficientes en términos del uso del espacio de almacenamiento y tiempo de procesamiento.

Existen tantos lenguajes ensambladores como arquitectura de procesadores con sus correspondientes conjuntos de instrucciones. Desde el punto de vista de la ingeniería de software, los lenguajes ensambladores sólo se deben usar cuando un lenguaje de alto nivel no satisfaga los requisitos o no esté disponible.

Segunda generación de lenguajes.

La segunda generación de lenguajes fue desarrollada a finales de los años 50 y principios de los 60 y ha servido como base para todos los lenguajes de programación modernos (tercera generación). Esta generación está caracterizada por su amplio uso, la enorme cantidad de bibliotecas de software que proporcionan sus lenguajes y su gran familiaridad y aceptación.

Los lenguajes que caracterizan a esta generación, y que además gozan de gran popularidad y aceptación son: FORTRAN, COBOL, ALGOL y BASIC.

FORTRAN ha subsistido durante más de treinta años y sigue siendo el primer lenguaje de programación en el ambiente científico y de ingeniería. FORTRAN es una herramienta potente para la resolución de problemas computacionales y a veces ha sido forzado para ajustarse a áreas de aplicación para las que no fue diseñado. Sin embargo, sigue siendo por muchos el lenguaje elegido para aplicaciones de cálculo numérico, aunque otros lenguajes como C, y PASCAL están reemplazando a FORTRAN por tener ventajas significativas.

COBOL al igual que FORTRAN ha alcanzado una gran madurez y durante mucho tiempo ha sido el lenguaje aceptado como estándar para aplicaciones comerciales. Asimismo, tiene gran popularidad y aceptación debido a que es muy auto-documentado, proporciona excelentes posibilidades de definición de datos y proporciona soporte para un gran rango de técnicas algorítmicas relativas al procesamiento de datos en los negocios.

El lenguaje COBOL empieza a ser sustituido por lenguajes de cuarta generación, pero todavía posee un número considerable de aplicaciones que operan bajo este lenguaje, sobre todo en organizaciones y empresas grandes.

ALGOL es el predecesor de muchos de los lenguajes de tercera generación y ofrece un repertorio extremadamente rico de instrucciones procedimentales y de tipificación de datos, además soporta el concepto de estructuración en bloques, asignación dinámica de memoria, recursividad y otras características que han tenido gran influencia en los lenguajes que más tarde aparecerían en la tercera generación.

BASIC es un lenguaje que originalmente fue diseñado para la enseñanza de la programación en modo de tiempo compartido. Este lenguaje aunque parecía quedar obsoleto, resurgió con la aparición de las computadoras personales y en la actualidad existen muchas versiones de BASIC, algunas con grandes ventajas, otras con deficiencias.

Tercera generación de lenguajes.

Los lenguajes de tercera generación (también llamados de programación estructurada) están caracterizados por sus potentes posibilidades procedimentales y de estructuración de datos. Pueden dividirse en: lenguajes de alto nivel de propósito general, lenguajes de alto nivel orientados a los objetos y lenguajes especializados.

Lenguajes de alto nivel de propósito general.

Entre este tipo de lenguajes aparecen PL/1, PASCAL, Modula-2, C y Ada.

PL/1 fue desarrollado con un rango amplio de alternativas para ser usado en muchas áreas de aplicación diferentes. PL/1, da soporte a aplicaciones convencionales de ciencia e ingeniería y de negocios, permitiendo la especificación de sofisticadas estructuras de datos, la multitarea, una compleja E/S, el procesamiento de listas y muchas otras posibilidades. Se han desarrollado subconjuntos del lenguaje para enseñar programación (PL/C), para su uso en aplicaciones de microprocesador (PL/M) y para programación de sistemas (PL/S).

PASCAL es un moderno lenguaje de programación desarrollado a principios de los años 70 para enseñar técnicas modernas de desarrollo de software (programación estructurada). Desde su introducción, PASCAL ha encontrado mucho apoyo en organizaciones, instituciones de educación y en profesionales dedicados al desarrollo de sistemas. PASCAL contiene: estructuración en bloques, fuerte tipificación de datos, soporte directo de la recursividad y muchas otras características suplementarias. Ha sido implementado en computadoras de todo tipo.

Modula-2 es un descendiente evolucionado de PASCAL, junta las posibilidades de implementación directa del diseño, como el ocultamiento de información, la abstracción y la fuerte tipificación de datos, con las estructuras de control que soportan la recursividad y la concurrencia. Hasta ahora el uso de Modula-2 en aplicaciones industriales ha estado limitado.

C, es un lenguaje de programación que fue originalmente desarrollado para implementaciones de sistemas operativos. El sistema operativo UNIX está implementado en C. Sin embargo, actualmente se han construido una gran cantidad de productos de software en todas las áreas profesionales.

C, posee una gran flexibilidad, y como otros lenguajes de esta categoría soporta estructuras de datos sofisticadas, con características muy sobresalientes de tipificación, hace uso intensivo de los punteros y tiene un rico conjunto de operadores para el cálculo y la manipulación de datos.

Ada es un lenguaje similar al PASCAL en notación y estructura, soporta un conjunto de alternativas que incluyen la multitarea, el manejo de interrupciones y la sincronización y comunicación entre tareas. Ada tiene una gran estructura de lenguaje y un buen enfoque del entorno operativo. Por otra parte es complejo y tiene una larga curva de aprendizaje.

Lenguajes orientados a los objetos.

Los lenguajes de programación orientados a los objetos permiten al ingeniero de software implementar los modelos de análisis y diseño orientados a objetos. Algunos ejemplos de lenguajes orientados a objetos son: C++, y Smalltalk.

El C++ es un lenguaje muy difundido, sobre todo porque posee la gran fortaleza del propio C y esto permite una suave transición a partir de este lenguaje de alto nivel de propósito general y ampliamente usado. En general, los lenguajes de programación orientados a objetos comienzan a adquirir más adeptos porque los propios diseñadores de sistemas toman en cuenta esta técnica para el diseño de los mismos y esto trae como consecuencia el empleo de un lenguaje de programación orientado a los objetos por parte de los programadores para el desarrollo e implementación.

Lenguajes especializados.

Los lenguajes especializados se caracterizan por su inusual formulación sintáctica que ha sido especialmente diseñada para una aplicación particular. Actualmente se usan muchos lenguajes especializados, pero éstos, tienen una base de usuarios menor que los de propósito general.

Desde un punto de vista de ingeniería del software, los lenguajes especializados tienen tantas ventajas como desventajas. Debido a que cada lenguaje especializado se ha diseñado para una aplicación específica, se puede facilitar la traducción de los requisitos del diseño a la implementación en código. Por otro lado, la mayoría de los lenguajes especializados son poco portables y en ocasiones puede resultar complicado su mantenimiento en relación a los lenguajes de propósito general.

Lenguajes de cuarta generación.

En el desarrollo de software, siempre se ha intentado generar programas de computadora con cada vez mayores niveles de abstracción. Los lenguajes de cuarta generación (L4G) han elevado aún más el nivel de abstracción que los lenguajes de tercera generación.

Los lenguajes de cuarta generación, al igual que los lenguajes de inteligencia artificial, contienen una sintaxis distinta para la representación de control y para la representación de estructuras de datos. Sin embargo un L4G representa estas estructuras en un mayor nivel de abstracción, eliminando la necesidad de especificar los detalles algorítmicos. Los lenguajes de cuarta generación, combinan características procedimentales y no procedimentales. Es decir, el lenguaje permite al usuario especificar condiciones con sus correspondientes acciones (componente procedimental), mientras que, al mismo tiempo, se pide al usuario que indique el resultado deseado (componente no procedimental), encontrando los detalles procedimentales mediante la aplicación de su conocimiento del dominio específico.

Lenguajes de petición.

La gran mayoría de los L4G se han desarrollado para ser usados conjuntamente con aplicaciones de bases de datos. Tales lenguajes de petición permiten al usuario manipular de forma sofisticada la información contenida en una base de datos previamente creada.

Algunos lenguajes de petición, tienen una sintaxis compleja que no es más sencilla que la de los lenguajes de tercera generación. Sin embargo, otros lenguajes de petición actualmente disponibles ofrecen una interfaz de lenguaje natural que permite al usuario expresar claramente el objetivo de su petición.

Generadores de programas.

Son otra clase de L4G, aunque más sofisticados. Los generadores de programas, más que apoyarse en una base de datos predefinida, permiten al usuario crear programas en un lenguaje de tercera generación usando notablemente menos sentencias. Estos lenguajes de programación de muy alto nivel hacen gran uso de la abstracción de datos y de procedimientos.

La nueva generación de herramientas CASE permiten al ingeniero de software modelizar gráficamente una aplicación de ingeniería y después generar el código fuente en C o Ada a partir del modelo gráfico.

Otros L4G.

Aunque los lenguajes de petición y los generadores de programas son lo L4G más comunes, existen otras categorías.

Los lenguajes de soporte a la toma de decisiones permiten que los no programadores lleven a cabo una gran variedad de análisis, que van desde los simples modelos de hojas de cálculo bidimensionales hasta los sofisticados sistemas de modelos estadísticos y de investigación operativa.

Los lenguajes de prototipos se han desarrollado para asistir en la creación de prototipos facilitando la creación de interfaces de usuario y diálogos, además de proporcionar medios para la modelización de datos.

2.7.4 HERRAMIENTAS DE PUESTA A PUNTO.

Las herramientas de puesta a punto son los instrumentos de los que disponemos para poner en funcionamiento un sistema. Estas se describen a continuación.

Compiladores de los lenguajes de programación de alto nivel.

Estos van desde COBOL y PASCAL hasta los lenguajes actuales de cuarta generación que permiten al programador usar instrucciones de alto nivel, que se traducen a instrucciones primitivas de bajo nivel que entiende la computadora.

Herramientas de prueba, de corrección de errores, simuladores, etc.

"Estos proporcionan al programador información acerca del comportamiento dinámico de su programa mientras se ejecuta. Las herramientas de modelado permiten al programador crear una gran variedad de casos de prueba para asegurar que el programa se pruebe efectivamente. Las herramientas de corrección permiten rastrear errores cuando se sabe que algo anda mal. Los simuladores proporcionan una representación más visual y gráfica del programa, por ejemplo, mostrándolo en forma de diagrama de flujo y simulando su comportamiento a la vez que se ejecuta, mostrando el flujo de control a través del diagrama de flujo".[26]

Terminales de tiempo compartido.

Estas reemplazan a los ambientes de desarrollo por lote, porque son terminales de tiempo compartido, en las que cada programador comparte los recursos. Es por ello que es importante considerar como herramienta a una terminal.

Computadoras personales para el desarrollo de programas fuera de línea.

Es una alternativa atractiva ya que el programador puede usarla para crear su programa y hacerle correcciones y revisiones apropiadas utilizando un programa estándar de procesamiento de palabras, para compilarlo y ver si existen errores de sintaxis y para realizar algunas pruebas fuera de línea.

Paquetes de control de programas fuente.

Estos evitan que el programador haga cambios no autorizados a las versiones oficiales de un programa. En proyectos de programación grandes, una problemática es el control de la configuración, ya que debe existir un control firme sobre las diversas partes del sistema final. *"Un paquete de control de código fuente es como un bibliotecario automatizado: evita el acceso no autorizado a documentos oficiales".*[27]

[26] Pressman S. Roger; Ingeniería de Software; McGraw-Hill, 1993, México, pág. 513-514.
[27] Ibidem, pág. 515.

Revisión inicial para evitar complejidad excesiva.

"Una de la métricas más útiles a la larga es la complejidad. Existen modelos matemáticos de complejidad de programas que pueden usarse para predecir la dificultad de probar y mantener un programa de computadora. Si los modelos matemáticos se aplican automáticamente a cada módulo del programa en el sistema que se está desarrollando, entonces los desarrolladores y el administrador del proyecto tendrán una rápida advertencia sobre las porciones potencialmente peligrosas del sistema y así podrán explorar otros diseños". [28]

Simulación y pruebas auxiliadas por computadora.

Existen paquetes de prueba y animadores auxiliados por computadora que ofrecen al programador una representación gráfica de la ejecución de su programa.

Con las herramientas de apoyo mencionadas, gradualmente podemos darnos cuenta de que se aleja cada vez más la revisión informal para acercarse más a pruebas completas y formales de corrección. Las herramientas descritas anteriormente se ocupan principalmente de la labor de escribir programas y ponerlos en operación.

Existen otras herramientas de puesta a punto de sistemas y son las siguientes: técnicas de programación, errores típicos de programación, estilo de programación y herramientas de depuración.

Técnicas de programación.

1. Los programas del sistema no pueden considerarse correctos hasta que han sido validados utilizando un rango amplio de datos de prueba.
2. Los programas deben ser legibles y comprensibles.
3. Utilizar comentarios que describan el propósito de un programa o segmentos de un programa, así como los elementos importantes de programas, funciones, rutinas, etc.
4. Etiquetar las salidas producidas por un programa.
5. Los programas deben ser eficientes, se deben evitar redundancias innecesarias.
6. Programas generales y flexibles. Deben ser fáciles de modificar para solucionar un problema sin cambiar mucho el programa.
7. En programación interactiva, se debe incluir una línea de mensaje de aviso al usuario cuando desee introducir datos.

[28] Pressman S. Roger; Ingeniería de Software; McGraw-Hill, 1993, México, pág. 528.

Errores típicos de programación.

Estos errores están en función del lenguaje utilizado para el desarrollo de sistemas, así que podemos establecer un estándar. Pero generalmente se refieren a los errores de sintaxis, errores como división por cero, raíces cuadradas de números negativos, etc.

Esencialmente hay dos tipos de errores: gramaticales y lógicos. Los lógicos constituyen una gama que comprende todos los errores de un mala compresión de lo que tenía que hacer el programa. Esto incluye a los procedimientos que no se comunican correctamente y a los errores lógicos internos del código.

Estilo de programación.

El estilo de programación está en función principalmente de la experiencia del programador, quien debe tomar en cuenta los siguientes puntos:

- Los errores de los programas pueden ser: sintaxis, en tiempo de ejecución y lógicos.
- La planificación de los programas comienza con el análisis del problema y sigue con el algoritmo. Especialmente en rutinas grandes, seguir el diseño descendente y refinamiento sucesivo. A continuación debe realizarse la documentación externa, la escritura del programa con toda la documentación interna (comentarios) necesaria.

Herramientas de depuración.

Estas consisten en hacer la depuración de los programas mediante el debug, normalmente, las herramientas de depuración se usan cuando el programa está suspendido temporalmente. Las herramientas de depuración que se pueden usar son:

- Iniciar/Reiniciar.
- Parar.
- Terminar.
- Punto de ruptura.
- Vistazo.
- Llamar.
- Paso a paso.
- Procedimiento en un paso.

Las tres primeras proporcionan la forma más común de cambiar desde el diseño del proyecto o modo diseño, al modo ruptura, donde se hará la mayor parte de la depuración.

La prueba de los programas es el primer paso de la depuración. En estos casos, la elección de los datos de prueba es muy importante. Puesto que los fallas generalmente son originadas con frecuencia por errores, las trampas para errores pueden ayudar a aislar la parte del programa en la que se producen.

2.7.5 RECURSOS DE MULTIMEDIA.

Multimedia es una combinación entrelazada de texto, imágenes, sonido, animación y video. Una aplicación de multimedia puede abarcar todos o algunos de los componentes anteriores.

De acuerdo a la definición anterior los componentes de multimedia son:

- Texto.
- Imágenes.
- Sonido.
- Animación.
- Video.

Para cada uno de dichos componentes se han desarrollado con el paso del tiempo diversos formatos y técnicas para su obtención y aplicación en la multimedia. También existen muchas aplicaciones en el mercado de autoedición, CAD (Diseño Asistido por Computadora), edición de sonido, etc. para crear multimedia.

Componentes de multimedia.

Texto.

En la actualidad, cuando se habla de texto en un ambiente gráfico y en multimedia debe tomarse en cuenta que el texto por sí solo ya no existe, ahora involucra tipo de letra, fuente y su tamaño en puntos, además de otros atributos como color, orientación, *inclinación, grosor, subrayado y delineado.*

Tipo de letra.

Se conoce como tipo de letra a una familia de caracteres gráficos que incluyen varios tamaños y estilos de letra.

Fuente.

Una fuente es una colección de caracteres con un solo tamaño y estilo, que pertenecen a un tipo de letra.

Los tamaños de letra se expresan en puntos; un punto es .0138 pulgadas o cerca de $^1/_{72}$ de pulgada. El tamaño de la fuente es la distancia desde la parte de arriba de las letras mayúsculas hasta la parte de abajo de las letras minúsculas como la *g* y la *y*. Helvética, Times y Courier son tipos de letra y por ejemplo Times de 12 puntos itálica es una fuente. Existen tres conjuntos de fuentes importantes, como son: PostScript,

TrueType y mapas de bits. Cada uno de estos conjuntos utiliza una tecnología diferente para dibujar las formas de los caracteres en forma digitalizada.

Las fuentes PostScript describen cada caracter en términos matemáticos (curvas de Béizer) y se pueden escalar. Esto hace que los caracteres se vean bien si están dibujados a 10 o a 100 puntos. Además los caracteres pueden dibujarse mucho mas rápido que en la forma tradicional.

Las fuentes TrueType trabajan con una metodología de fuentes de contorno con curvas cuadráticas "mejor y más rápido". Además de imprimir caracteres suavizados, TrueType dibuja caracteres en un monitor de baja resolución (72 dpi).

Las fuentes que trabajan a base de mapas de bits, utilizan tablas de formas en una tabla de mapas de bits que contienen la representación de cada caracter en cada tamaño.

La selección del tipo y fuente a utilizar puede resultar algo complicado por la gran variedad que existe. El texto puede combinarse o complementarse dentro de la aplicación, con menús de navegación, botones de interacción, campos de lectura, símbolos e iconos o incluso animando el texto.

Sonido.

Para aplicaciones de multimedia el sonido es un componente muy importante que le da vida a la aplicación, pero para poder tener sonido de alta calidad en una aplicación en una PC se debe contar con una tarjeta de sonido de alta calidad.

Existen dos formas para producir y reproducir sonido en multimedia: MIDI y audio digital.

MIDI.

La Interface Digital para Instrumentos Musicales (MIDI) es un estándar de comunicaciones desarrollado para instrumentos musicales electrónicos y computadoras. Permite que la música y los sintetizadores de sonido de diferentes fabricantes puedan comunicarse entre sí enviando mensajes a través de cables conectados a los dispositivos.

MIDI proporciona un protocolo para pasar descripciones detalladas de una partitura musical, como notas y secuencias de notas y qué instrumento las tocará.

Los datos MIDI no son digitalizados, son una representación taquigráfica de la música almacenada en forma numérica.

Ventajas de utilizar MIDI sobre audio digital:

- Los archivo MIDI son mucho más compactos que los archivos de audio digital, y su tamaño es por completo independiente de la calidad de reproducción. En general, los archivos MIDI serán entre 200 a 1000 veces más pequeños que los archivos de audio digital con calidad CD. Debido a que son pequeños, no emplean mucha RAM, espacio en disco o recursos del CPU.
- En algunos casos, los archivos MIDI pueden sonar mejor que los de audio digital si la fuente de sonido de MIDI que utiliza es de alta calidad.
- Se puede cambiar el tamaño de un archivo MIDI (variando su ritmo) sin cambiar el tono de la música ni degradando la calidad de audio. Los datos MIDI son completamente editables hasta el nivel de una nota individual. Se pueden manipular los detalles más pequeños de una composición MIDI de diferentes maneras, algo que es imposible con el audio digital.

Desventajas de MIDI respecto al audio digital:

- Debido a que los datos MIDI no son sonido, la reproducción será precisa sólo si el dispositivo de reproducción MIDI es idéntico al que se utilizó para la producción.
- Aún con el estándar MIDI el sonido de un instrumento MIDI varía de acuerdo a la electrónica del dispositivo de reproducción y al método de generación de sonido que utiliza.
- MIDI no puede utilizarse con facilidad para volver a reproducir un diálogo hablado, aunque existen dispositivos de muestreo, técnicamente complicados y caros.

Audio digital.

Puede digitalizar sonido desde un micrófono, un sintetizador, grabaciones en cinta, emisiones en vivo de radio y televisión y CDs. De hecho puede digitalizar sonidos desde cualquier fuente, natural o pregrabada.

Los sonidos digitalizados son muestras de sonido. De cada enésima fracción de segundo se toma una muestra de sonido y se guarda como información digital en bits y bytes. La velocidad de muestreo es la frecuencia con que se toman las muestras y el tamaño de la muestra es la cantidad de información almacenada de cada muestra.

Mientras más seguido se utilice una muestra y almacene más datos acerca de ella, mejor será la resolución y la calidad de reproducción del sonido capturado.

La ventaja principal de trabajar con audio digital consiste en la calidad de reproducción, además de que está disponible una selección más amplia de programas de aplicación y soporte para audio digital que para MIDI. También la preparación y programación requerida para crear audio digital no demanda conocimientos de teoría musical.

Las operaciones de edición básica de sonidos que se necesitan para producir multimedia son:

- Recortes.
- Empalmar y montar.
- Ajustes de volumen.
- Conversión de formato.
- Remuestreo y muestreo a baja velocidad.
- Disolvencia y desvanecimiento.
- Ecualización.
- Ajustes de tiempos.
- Procesamiento digital de las señales.

En Windows los sonidos digitalizados se almacenan como archivos de onda (.WAV), el formato por omisión y el más común.

Imágenes.

Las imágenes son quizá el elemento más importante de un proyecto de multimedia. Los mapas de bits y los vectores son dos técnicas diferentes que se emplean para crear y mostrar imágenes.

Los mapas de bits se utilizan para obtener imágenes fotorrealistas y dibujos complejos que requieren detalles finos. Los objetos dibujados con vectores se emplean para hacer líneas, cajas, círculos, polígonos y otras figuras gráficas que se pueden expresar matemáticamente en términos de ángulos, coordenadas y distancias. Un objeto dibujado puede llenarse con colores y patrones. La apariencia de ambos tipos de gráficos depende de la resolución del monitor y de las capacidades gráficas del sistema.

Mapas de bits.

Un mapa de bits es una matriz de información que describe los puntos individuales que son el elemento de resolución más pequeño en la pantalla o la impresora. Se requiere una matriz de una dimensión para datos monocromáticos; pero se necesita una mayor profundidad (más bits de información) para describir los más de dieciséis millones de elementos de colores que puede tener una imagen a colores.

Los pixeles pueden estar encendidos o apagados (1 bit para el caso de los monocromáticos blanco y negro); o pueden representar varios tonos de color (4 bits para 16 colores; 8 bits para 256 colores; 16 bits para 32768 colores y 24 bits para millones de colores).

Existen tres formas para crear un mapa de bits:

- Crearlo desde cero con un programa de pintura.
- Capturar un mapa de bits de la pantalla activa y luego pegarlo con un programa de pintura en la aplicación.
- Capturar un mapa de bits de una fotografía, arte gráfico o imagen de televisión utilizando un digitalizador o dispositivo de captura de vídeo.

Dibujo de vectores.

La mayoría de los sistemas de desarrollo de multimedia proporcionan líneas, rectángulos, óvalos, polígonos y texto dibujado con vectores.

- Los programas de diseño asistido por computadora (CAD) han utilizado sistema de objetos de vectores para crear las figuras geométricas altamente complejas que requieren los arquitectos e ingenieros.
- Los artistas gráficos que diseñan medios impresos también utilizan esta técnica.
- Los programas para animación en tercera dimensión (3-D) también utilizan gráficos de vectores. Por ejemplo, los diferentes cambios de posición, rotación y sombra de luces que se requieren para girar un logotipo deben calcularse matemáticamente.

Los objetos de vectores se describen y dibujan en la pantalla de computadora empleando una fracción de espacio de la memoria requerido para describir y almacenar el mismo objeto en un mapa de bits. Un vector es una línea que se describe con la localización de los puntos de sus extremos. Un rectángulo por ejemplo puede definirse así: RECT 0,0,200,200

Empleando coordenadas cartesianas, el programa dibujaría el rectángulo comenzando *en el extremo superior izquierdo* de la pantalla, va horizontalmente 200 pixeles a la derecha y verticalmente 200 pixeles para abajo.

La mayoría de los programas de dibujo ofrecen muchos formatos de archivos para grabar la imagen, puede convertirse un dibujo que consista en muchos objetos de vectores en un mapa de bits cuando se grabe.

La utilización de *un solo mapa de bits para una imagen complicada* puede dar un mejor desempeño de refresco a la pantalla que el empleo de un gran número de objetos de vectores para crear la misma pantalla. La TABLA 2.2 contiene formatos de archivos de imágenes que se pueden utilizar en ambiente Windows.

Formato	Extensión
DIB de Windows de Microsoft	BMP, DIB y RLE
RLE DIB de Microsoft	DIB
Paletta de Microsoft	PAL
RIFF DIB de Microsoft	RDI
Metaarchivo gráfico de computadora	CGM
Designer/Draw de Micrografx	DRW
Formato 2D de AutoCAD	DXF
Especificación inicial para el intercambio de gráficos	IGS
PostScript encapsulado	EPS
GIF de compuserve	GIF
Lenguaje gráfico HP	HGL
PC Paintbrush	PCX
PICT de Macintosh de Apple	PIC
Gráficas de lotus 1-2-3	PIC
Importación de AutoCAD	PLT
TGA de Truevision	TGA
TIFF	TIF
Metaarchivo de windows	WMF
DrawPerfect	WPG

TABLA 2.2. Formatos de Imágenes para Windows.

Componentes mínimos de hardware para una PC multimedia.

La multimedia reúne una serie de técnicas diferentes, por lo que el usuario que quiera entrar a este mundo debe involucrarse en toda una serie de términos nuevos.

Actualmente existen sistemas completos de multimedia o bien por otro lado están las soluciones económicas de los llamados "conjuntos de ampliación de multimedia", a través de los cuales es posible ampliar la computadora existente para convertirla en una PC Multimedia.

En principio para que una PC pueda realmente convertirse en PC Multimedia, debe trabajar bajo Windows, al que se le agrega una tarjeta de sonido y una unidad de CD-ROM. Como la plataforma en la que se trabaja es Windows la PC debe tener suficiente potencia para que las aplicaciones puedan ejecutarse a una velocidad adecuada.

La estructura de hardware que se escoja depende de las aplicaciones multimedia con la que se vaya a trabajar, si se quiere digitalizar imágenes de video, se necesitará una tarjeta de video (overlay). También se puede adecuar un escáner para digitalizar imágenes.

Se puede optar bien por adquirir un equipo multimedia completo o bien adquirir una PC con procesador pentium y un conjunto de ampliación multimedia. O también pueden adquirirse por separado cada uno de los componentes.

Veamos ahora los recursos que nos proporciona una PC con Multimedia:

Tarjeta de sonido.

Las tarjetas de sonido reúnen diversas funciones, en primer lugar, la conversión de señales analógicas en datos digitales y viceversa constituyen la tarea más importante de una tarjeta de sonido. La calidad de la grabación de las señales analógicas que puedan captarse a través de las diversas conexiones, define la calidad de la tarjeta.

El procedimiento de convertir señales analógicas en datos digitales se conoce como muestreo ("sampling"). Una tasa de muestreo de 44.1 KHz en estéreo corresponde a una tarjeta de sonido de alta calidad. El estándar está en 11.025 KHz en estéreo y 22.05 KHz en mono.

Al igual que la calidad de grabación, también la calidad de reproducción es un criterio que define la calidad de la tarjeta de sonido. Es aquí donde se transforman los datos digitales en señales analógicas y debe alcanzarse la calidad de 44.1 KHz de los discos compactos. Adicionalmente, en este proceso no deben consumirse más del 15% de la capacidad total del CPU.

La reproducción se realiza a través de las bocinas que puedan conectarse a la tarjeta de sonido. Los amplificadores integrados permiten una salida de entre 1 y 6 volts por canal a 4 ohms, que resulta suficiente para la reproducción en una habitación.

El volumen es generalmente regulado a través del software correspondiente. Con un programa que mezcla, que acompaña a casi todas las tarjetas de sonido, no sólo es posible regular el volumen de salida, sino también el volumen para el micrófono y para todas las otras entradas de forma independiente.

También es posible una regulación diferente para el canal estéreo derecho y para el izquierdo.

Según las posibilidades de conexión, será posible grabar desde diferentes fuentes a través de la tarjeta de sonido. Se requiere una conexión para el micrófono, una conexión AUX para el equipo Hi-Fi, una conexión con la unidad de CD-ROM para la reproducción de los discos compactos de audio y una conexión de salida a través de bocinas o audífonos. De esta forma es posible grabar todo tipo de música, ruidos y voces. En ocasiones se encuentran conexiones adicionales para incorporar una unidad de CD-ROM.

Otro componente de la tarjeta de sonido, es el sintetizador integrado, a través de éste se pueden crear y reproducir artificialmente tonalidades de diferentes instrumentos

musicales. La cantidad de instrumentos que pueden ser reproducidos simultáneamente, así como el tipo de creación de sonido definen la capacidad del sintetizador. La creación de sonidos se basa normalmente en la frecuencia modulada (FM). A través de la modulación de frecuencia se pueden crear sonidos verdaderamente reales.

El sintetizador interno es controlado a través de datos MIDI (Interface Digital para Instrumentos Musicales). MIDI denomina en primer lugar a un formato de archivo estandarizado para el almacenamiento de archivos de sonidos, aquí se guardan valores tales como altura del tono, duración, volumen, etc., también se trasladan los nombres de los instrumentos que deben ser utilizados en la reproducción de tonos. Entonces, es el chip sintetizador el que construye los sonidos.

Las tarjetas de sonido disponen de un puerto MIDI, a través de éste pueden conectarse dispositivos MIDI externos como sintetizadores, dispositivos de efectos o teclados. Mediante programas "secuenciadores" pueden guardarse piezas musicales completas en forma de archivos MIDI y reproducirlos a través del sintetizador de la tarjeta de sonido.

Los puertos MIDI suelen asumir una doble función ya que también es posible operar un joystick a través de los mismos.

Bocina y micrófono.

Al instalar una tarjeta de sonido, las bocinas son imprescindibles. Aquí hay dos opciones bocinas activas y pasivas. La diferencia radica en que las bocinas activas disponen de un amplificador propio y normalmente el volumen se regula directamente en la bocina. Lo que significa que la señal es ampliada de nuevo. Las bocinas activas, debido a su amplificador propio, tienen que, recibir el suministro eléctrico de forma independiente. Como norma, esto se realiza a través de una conexión eléctrica o mediante baterías.

Las bocinas pasivas reproducen la señal sin más modificaciones. La incorporación de estas bocinas está en función de la capacidad de la tarjeta gráfica. Una capacidad de 4 volts, con 4 ohms por canal estéreo es suficiente. El volumen se regulará sólo a través de la tarjeta de sonido.

Para la grabación de voces y sonidos es posible conectar un micrófono a la mayoría de las tarjetas de sonido. A través de éste es posible, en conjunto con los programas de "Microsoft Windows Multimedia Extensión", crear, modificar y reproducir archivos de sonido. Un buen micrófono debe reunir las siguientes características:
Impedancia = 600 ohms y Sensibilidad = 74 dB o mayor.

En el campo de software para juegos, existen muchos programas que reproducen sonidos muy naturales a través de una tarjeta de sonido. El control de estos programas es a menudo a través de un joystick, mediante el movimiento de la palanca de control

se controla el cursor en la pantalla y la ejecución de determinadas acciones se realiza a través del llamado botón de disparar.

Los joysticks analógicos en los PC, normalmente tienen que ser calibrados (alineados). Esto se hace generalmente dentro de una rutina de configuración de los distintos juegos. Para que pueda usarse correctamente dentro de Windows se cuenta con un programa para calibrarlo dentro del panel de control de la Mutimedia Windows Extensión y de Windows 3.1.

Unidad de CD-ROM.

Las unidades de CD-ROM deben cumplir con determinados requisitos en los referente a la velocidad de transferencia de datos. Si la unidad cumple con las exigencias del nivel 1, con una tasa de transferencia de datos de 150 KB/s, resulta ser muy baja la transferencia de datos, además no deben requerirse más del 40% de la capacidad de el CPU.

Para el uso profesional, es imprescindible la adquisición de una unidad que utilice la tecnología Double Spin para la transferencia de datos y permita con ello alcanzar tasas por encima de los 300 KB/s. El rendimiento puede incrementarse aún más si la unidad dispone de por lo menos un buffer de datos de 64 KB. Los bloques de datos que se lean no deben estar por debajo de lo 16 KB. El tiempo de ejecución no debe superar el tiempo que se necesita para leer un bloque de datos en el buffer de la unidad.

El tiempo promedio de acceso debe estar por debajo de un segundo. En realidad resulta conveniente contar con tiempos de acceso que estén en el orden de los 400 milisegundos.

2.8 DOCUMENTACION.

2.8.1 MANUAL DEL USUARIO.

El manual de usuario es una guía que muestra la forma de utilizar el sistema. Esta guía es una interface gráfica para el usuario, que por medio de un menú selecciona el tema en el cual existan dudas sobre el manejo del programa. Este tipo de manual son los llamados en línea, en el que se realizan consultas en cualquier momento durante la ejecución del programa, sobre las dudas que surjan sobre la operación del sistema.

Esta guía está integrada por información descriptiva del sistema, la cual explica su operación, instalación, configuración y funciones que realiza. Es importante que describa las funciones que ofrece el programa y la manera de cómo realizar consultas. Dentro de la información proporcionada por la guía también se debe incluir información orientada hacia el administrador, en la cual encuentre información acerca de la posible acción a seguir ante ciertos problemas que lleguen a presentarse durante la ejecución del sistema.

La información que contenga la guía debe proporcionar una visión precisa del sistema. Dicha guía debe estructurarse de tal forma que el usuario pueda comprender con un grado de detalle sus necesidades de consulta. Es importante tomar en cuenta la variedad y diferencia de usuarios que utilizan el sistema, esto debido a que no todos los usuarios tienen el mismo grado de conocimientos. Por esta razón, es necesario que la persona encargada de la elaboración de la guía deba estructurarla de tal manera que le sea útil a diferentes tipos de usuarios. Por otra parte, es importante que se distinga entre usuarios finales y administradores del sistema.

A continuación se muestran algunos de los puntos que se pueden incluir en la guía del usuario:

- Contenido temático.
- Búsqueda de palabras específicas.
- Indice.
- Descripción de comandos y funcionamiento de los módulos.

La guía de usuario necesita ser construida de tal forma que los temas sean presentados con una estructura más fácil de entender al usuario. En los siguientes incisos se describen algunas características que puede presentar el documento:

- Una interface gráfica que habilite al usuario para seleccionar una lista de temas, en la cual se pueda ir al detalle más a fondo.
- Implementación de las definiciones por medio de contexto sensitivo.

- *Utilización de saltos a definiciones.*
- Implementación de botones para el control de saltos entre temas.

Es importante que dentro de esta guía no se incluya demasiada información técnica, esto es porque a los usuarios no les interesan las cuestiones técnicas de la computadora, ni los detalles internos del diseño. Sino que deben ser textos lo más concisos y fáciles de entender.

Para elaborar esta guía en línea se utiliza la documentación que se genera en cada una de las etapas del desarrollo del sistema. Es importante considerar a esta guía como una revisión externa del producto que se está desarrollando. Ya que al ir elaborándola, va existiendo una mayor comunicación entre los analistas y los usuarios. Además, una vinculación más estrecha con las fuentes de información utilizadas en el desarrollo del sistema. Como resultado de esto se detectan errores durante el proceso de análisis.

Es importante mencionar que la elaboración de una versión preliminar de la guía sirve para empezar a estructurar la información que se presentará al usuario. La elaboración del manual durante el desarrollo del sistema ayuda a realizar las pruebas al mismo, esta actividad detecta errores que puedan existir en el desarrollo del sistema. Además, de proporcionar una mayor comunicación entre las personas involucradas en el proyecto.

En la FIG. 2.39 se muestra el contenido que se incluye en el manual del usuario y los servicios para los diferentes usuarios.

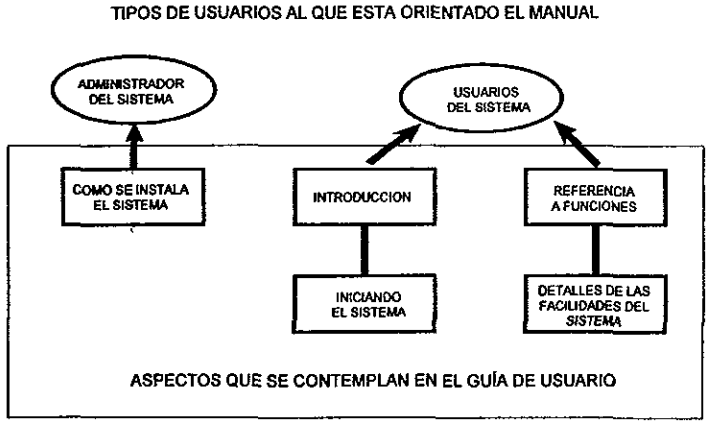


FIG. 2.39. Documentación para los diferentes usuarios.

Como se observa en la FIG. 2.39, los usuarios pueden utilizar esta guía como un manual de introducción. Para este caso, los usuarios deciden si el sistema cubre sus necesidades o requieren de otro tipo de funciones, además provee información básica de las funciones del sistema. En caso de que no satisfagan las necesidades de los usuarios se puede referir al manual de operación.

El manual del usuario puede presentar una introducción formal para el uso del sistema describiendo su uso normal, además de describir la forma de inicializarlo y cómo los usuarios finales pueden utilizar las facilidades que ofrece. También se muestra el alcance de las funciones. Debe ser sencillo de entender y con la facilidad de poder descubrir información sobre el sistema y la forma de reestablecerlo al encontrar errores, además de explicar la forma de reiniciar el sistema para seguir con sus actividades de trabajo.

Como se mencionó, este manual debe ser de fácil uso, con referencia rápida, además de tener disponibles las facilidades del sistema, algunos temas avanzados para usuarios experimentados en el sistema y sistema de ayuda en línea para los usuarios. Es importante que la información esté bien estructurada, que proporcione un resumen y una descripción detallada de cada aspecto del sistema.

En la elaboración del manual es necesario cuidar la calidad del producto, algunas recomendaciones para una mejor calidad son:

- Ortografía.
- Guía temática.
- Facilidad para entender.

La calidad del manual es importante, porque sin una información de cómo utilizar el programa la utilidad del sistema se degrada hasta llegar a no ser útil al usuario. Para cuidar este aspecto, se elaboran manuales con mejor calidad, manejando estándares y una buena calidad en los procesos.

Calidad de la documentación.

El manejo de estándares para la elaboración del manual, permite lograr una mejor calidad, los documentos producidos de acuerdo a los estándares una apariencia consistente y una mejor calidad. Existen dos tipos de estándares que se manejan:

- Estándar del proceso. Estos definen el proceso a seguir para obtener una mayor calidad en la elaboración del manual.
- Estándar del producto. Estos son los que gobiernan al manual.

Estándar del proceso.

Este estándar se utiliza en la elaboración de los documentos, establece la aproximación que debe utilizarse en la producción del manual, generalmente define el tipo de herramientas de software que se utilizan para la producción, además define la calidad para la elaboración del producto, esto ayuda a generar un documento con mejor calidad. En la FIG. 2.40 se muestran los pasos que se siguen para lograr una mejor calidad en la elaboración del manual de usuario.

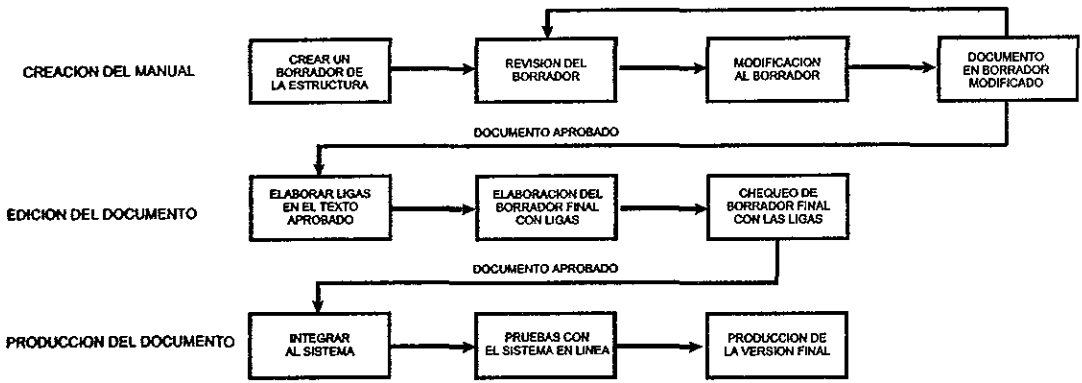


FIG. 2.40. Pasos para la producción de un documento.

En esta serie de pasos se realizan algunas de las actividades siguientes:

- Creación de manual. La creación se realiza con un procesador de texto.
- Depuración del documento. Se utilizan verificadores ortográficos, y correctores de estilo.
- Producción del documento. Compiladores de ayuda y software para manejo de hipertexto.

Para elaborar un manual se tiene que modificarlo, checarlo, revisarlo y volver a revisar. Este es un proceso interactivo que se sigue hasta lograr un producto con calidad y que sea aceptado por el cliente.

Un aspecto importante en la elaboración de los documentos es la depuración, esto ayuda a hacer más accesible el documento al usuario. Dentro de la depuración es importante considerar los aspectos gramaticales que se utilizan para elaborar el documento. Entre éstos se tienen los siguientes:

- Uso correcto de construcciones gramaticales y ortografía.
- No usar enunciados muy largos cuando se presenten varios factores.
- Utilizar enunciados cortos que sean más fáciles de asimilar por el lector.
- Párrafos no muy largos.

- Ser preciso en la definición de términos utilizados.
- Si se utiliza una definición muy compleja repetirla con otras palabras.
- Hacer uso de encabezados y subtítulos, esto se usa para romper el esquema de un capítulo en varias partes, las cuales se pueden leer separadamente, siempre y cuando sea consistente la numeración y el orden de la convención utilizada.
- Resaltar frases o palabras para mejor entendimiento.
- No hacer referencia a la información sólo con el número, sino escribir los conceptos a mostrar, y poner una liga a otro texto.

Estándar de producto.

El estándar del producto se aplica a todos los documentos que se producen durante el desarrollo del sistema, estos documentos pueden tener consistencia en apariencia y los documentos de la misma clase deben tener una estructura consistente.

Algunos de los estándares de producto que se pueden desarrollar son:

- Estándar en la estructura del documento. Debe definir la organización que tendrán los documentos, ésta puede especificar las convenciones utilizadas para el índice temático, para la distribución de textos, el subrayado y resaltado de palabras y la distribución esquemática.
- Estándar en la presentación del documento. El estándar en la presentación del documento incluye la definición del tipo de letra y el estilo usado en el documento, el uso de logos y nombre de las compañías, el uso de color para que resalte el documento.

El uso de estándares debe aplicarse durante el proyecto a todos los documentos, desde el borrador hasta la versión final del manual. Otro aspecto importante que permite obtener una mejor calidad de un producto, es la habilidad de redacción de las personas encargadas de redactar los documentos, para esto el escritor tiene que escribirlo, leerlo criticarlo y reescribirlo hasta que se llegue a producir un documento satisfactorio.

Preparación del documento.

La preparación del documento es casi de forma automática, las herramientas de software, se utilizan en todas las etapas de su elaboración, desde el inicio, en borradores y correcciones, hasta la producción de la versión final del manual.

Las herramientas existentes como los procesadores de textos, editores de gráficos y herramientas de hipertexto, ayudan en la producción de la documentación, éstas facilitan las modificaciones a los documentos desarrollados.

2.8.2 MANUAL DE OPERACION.

El manual de operación está orientado más a gente con preparación técnica. Este manual tiene un enfoque más técnico y está orientado a la operación del sistema, éste incluye aspectos de mantenimiento así como de configuración del sistema. La elaboración de este manual sirve como un medio de comunicación entre los integrantes del equipo que desarrolla el sistema. Otra característica importante es que es utilizado como un depósito de información del sistema, el cual puede ser usado por los ingenieros para dar el mantenimiento necesario. Este manual provee información acerca del diseño del sistema, tal como los diagramas de flujo de datos, entidad-relación, diccionario de datos, etc., además de toda la información utilizada en el desarrollo del sistema

Esta documentación describe las características internas del sistema, desde un punto de vista del desarrollo de la ingeniería de software, a diferencia del manual de usuario que provee una descripción orientada al uso del sistema por el usuario.

Este manual puede ser una descripción completa de las técnicas que se utilizaron, además el estilo utilizado en la redacción puede no ser muy didáctico, pero sí debe ser entendible para gente con un perfil técnico, esto por el uso de términos más técnicos, además que requiere de mayor conocimiento por parte de la persona que lo consulte. A continuación se listan algunos documentos asociados a este manual:

- Documentos que son requeridos en el desarrollo del sistema.
- Documentos que describan la arquitectura del sistema.
- Para cada componente, una especificación y una descripción.
- Listados del código fuente de los programas, éstos deben de estar comentados apropiadamente, donde se explique la función de cada línea de código.
- Documentos de validación en donde se describa cómo la información se valida.

Para elaborar este manual se requiere un buen manejo de los procesos que integran al sistema, esto no es fácil debido a que no los desarrolló una sola persona, esto hace que sea difícil entender la forma de pensar de los desarrolladores.

La documentación generada puede separarse en las siguientes categorías:

- Planes, estimados y cédulas. Estos los producen los líderes del proyecto, los cuales son utilizados para estimar y controlar los procesos del sistema.
- Reporte. Estos reportan el uso de los recursos y dónde son utilizados durante el proceso de desarrollo.
- Estándares. Especifican la forma de implementar los procesos, éstos definen el tipo de organización que se manejará en el desarrollo del sistema.

- Papeles de trabajo.

Esta documentación proporciona la comunicación técnica durante el desarrollo del sistema y sirven como registro de ideas para los ingenieros que están involucrados en el proyecto. Debido a que incluye toda la descripción de los procesos de implementación, desde la especificación de los requerimientos hasta la definición de los objetivos.

Este tipo de documento describe el diseño, implementación y pruebas del sistema, este documento es indispensable para el mantenimiento requerido del sistema.

2.9 PRUEBAS Y CONFIABILIDAD DE LOS SISTEMAS.

2.9.1 CARACTERISTICAS DE LA PRUEBA.

"La prueba es un grupo de actividades que se pueden planear por adelantado y llevar a cabo sistemáticamente."[29] Por ello es necesario definir una serie de pasos en los que podamos definir las técnicas específicas de diseño de casos de prueba y los métodos de prueba.

Se han definido diversas estrategias para las pruebas de software y todas tienen las siguientes características generales:

- La prueba comienza a nivel de módulo y trabaja "hacia afuera", es decir hacia la integración del sistema.
- Diferentes tipos de prueba son apropiadas en cada caso para diferentes momentos.
- La prueba la lleva a cabo el desarrollador del software y en el caso de grandes proyectos la realiza un grupo de prueba independiente.
- Se realiza la prueba y la depuración, aunque son actividades diferentes, es importante incluir la depuración en cualquier estrategia de prueba.

Una estrategia para la prueba del software debe incluir pruebas de bajo nivel que verifiquen que cada módulo de código fuente se ha implementado correctamente, así como pruebas de alto nivel que muestren la validez de las principales funciones del sistema.

[29] Pressman S. Roger; Ingeniería de software; Mc Graw-Hill, 1993, México, pág. 662.

2.9.2 PASOS EN LA PRUEBA DE LOS SISTEMAS DE PROGRAMACION.

La prueba consta de una serie de tres pasos que se describen a continuación y quedan ilustrados en la FIG.2.41.

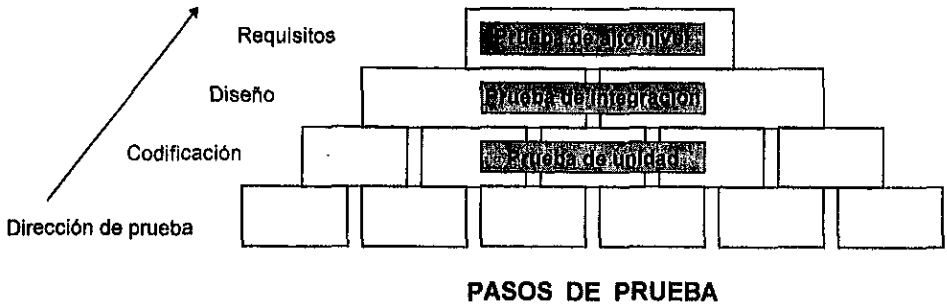


FIG. 2.41. Pasos en la prueba de los sistemas de programación.

1. La prueba se centra en cada módulo individual, asegurándose que cada uno de ellos funciona correctamente como una unidad. Se presenta aquí la llamada prueba de unidad, en ésta se utilizan caminos específicos de la estructura de control del módulo para asegurar un alcance completo y una detección máxima de errores.
2. Luego se deben integrar los módulos para formar el sistema completo. Se lleva a cabo la prueba de integración, la cual se dirige a todos los aspectos asociados con el problema de verificar y construir el programa. Después de que el sistema se ha integrado se realizan un conjunto de pruebas de alto nivel. Se deben comprobar los criterios de validación. La prueba de validación debe proporcionar la seguridad de que el sistema satisface todos los requisitos funcionales, de comportamiento y de rendimiento.
3. En el último paso de la prueba, el software una vez validado, se debe combinar con otros elementos del sistema, como pueden ser de hardware, usuarios, bases de datos. La prueba del sistema verifica que cada elemento encaja de forma adecuada y que se alcanza la funcionalidad y el rendimiento del sistema en su totalidad.

2.9.3 GENERADORES DE DATOS PRUEBA.

"Los generadores de datos de prueba son programas que generan automáticamente una gran cantidad de entradas de prueba para un sistema." [30] Además de considerarse un instrumento para pruebas del sistema.

También, son útiles para situaciones en las que se debe probar el funcionamiento de un sistema en un ambiente práctico. Por ejemplo, la prueba de un sistema de administración de bases de datos puede empezar utilizando bases de datos muy pequeñas; esto es, las pruebas se diseñan inicialmente para detectar errores de los programas que originen una salida incorrecta. Estas pruebas, a pequeña escala, en realidad no reflejan el verdadero ambiente en el que se va a usar el sistema, ya que, en general, se operará con bases de datos muy grandes. Dada la especificación de una base de datos, un generador de datos de prueba puede generar grandes cantidades de datos, así se podrá probar el rendimiento del sistema en un ambiente realista.

Otro ejemplo, en donde los generadores de datos de prueba pueden ser útiles es en aquellas situaciones en que la sintaxis de la salida de un sistema se puede especificar de un manera formal. Para este caso, puede escribirse un programa que revise esa sintaxis. Dada una especificación de entrada, el generador de datos de prueba puede producir un gran volumen de datos de entrada para presentar el sistema que se prueba. La salida se presenta al revisor de la sintaxis y las discrepancias que se encuentran pueden deberse a errores en el sistema que se prueba.

Un ejemplo de tal situación es la prueba de la fase de análisis sintáctico de un compilador. La salida de esa fase para un programa correcto puede ser cualquier programa de entrada, pero si se presenta un programa incorrecto, la salida también incluirá indicadores de error.

"Un generador de datos de prueba puede aceptar una especificación de la sintaxis del lenguaje que se compila y a partir de ella generar programas correctos e incorrectos. La salida del compilador se puede revisar automáticamente, para asegurar que no se generen mensajes de error para programas correctos y, a la inversa, que se generen mensajes de error para programas incorrectos." [31]

Es importante la existencia de los generadores de datos de prueba ya que generalmente, el programador prueba el sistema basándose en sus conocimientos acerca del mismo, sabiendo de antemano cuáles pueden ser las posibles fallas, por ello si se prueba el sistema con un generador de datos de prueba, es posible detectar errores. Además, puede considerarse la opción de pruebas al sistema directamente por los usuarios, ya que son ellos los que presentan los casos de prueba menos esperados.

[30] Sommerville Ian; Ingeniería de software; Addison Wesley, 1991, USA, pág. 212.
[31] Ibidem.

2.9.4 PRUEBA DE UNIDADES Y PRUEBA DE INTEGRACION.

Pueba de unidades.

Esta prueba es el nivel básico en donde se prueban las funciones que componen un módulo para garantizar que operan de manera correcta.

La prueba de unidad se aplica a la menor unidad del diseño del software, el módulo, se prueban los caminos de control importantes, con el fin de descubrir errores dentro del mismo. Esta prueba se puede llevar a cabo en paralelo para múltiples módulos.

"La prueba de unidad consiste básicamente en probar la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad del programa que está siendo probada. Se examinan las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente, conserven su integridad durante los pasos de ejecución del algoritmo. Se prueban las condiciones límite, para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento. Se ejercitan todos los caminos independientes de la estructura de control, con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez. Y, finalmente, se prueban todos los caminos de manejo de errores."[32]

Antes de iniciar otra prueba, es preciso probar que el flujo de datos de la interfaz de los módulos fluye en forma correcta. Para ello se propone una lista de preguntas a considerar a fin de comprobar la prueba de interfaz:

1. ¿Es igual el número de parámetros de entrada al número de argumentos que llegan?
2. ¿Coinciden los atributos y las unidades de los parámetros con los atributos y unidades de los argumentos?.
3. ¿Es igual el número de argumentos transmitidos a los módulos de llamada que el número de argumentos enviados?.
4. ¿Son correctos el número, los atributos y el orden de los argumentos de las funciones incorporadas?.
5. ¿Existen referencias de parámetros que no estén asociados con el punto de entrada actual?.
6. ¿Entran sólo argumentos alterados?.
7. ¿Son consistentes las definiciones de variables globales entre los módulos?.
8. ¿Se pasan las restricciones como argumentos?.

Cuando un módulo realice E/S externas se deben llevar a cabo, pruebas de interfaz adicionales y para ello existe otro bloque de preguntas:

[32] Pressman S. Roger; Ingeniería de software; Mc Graw-Hill, 1993, México, pág. 668-669.

1. ¿Son correctos los atributos de los archivos?.
2. ¿Son correctas las sentencias de apertura de los archivos?.
3. ¿Cuadra el tamaño del buffer con el tamaño del registro?.
4. ¿Se abren los archivos antes de usarlos?.
5. ¿Se tienen en cuenta las condiciones de fin-de-archivo?.
6. ¿Se manejan los errores de E/S?.
7. ¿Hay algún error textual en la información de salida?.

Las estructuras de datos locales en cada módulo es una de las causas de error más común, por lo tanto se deben diseñar casos de prueba para descubrir errores de las siguientes categorías:

1. Tipificación impropia o inconsistente.
2. Inicialización o valores implícitos erróneos.
3. Nombres de variables incorrectos (mal escritos o truncados).
4. Tipos de datos inconsistentes.

Además, debe comprobarse el impacto de los datos globales sobre cada módulo.

"Durante la prueba de unidad, la comprobación selectiva de los caminos de ejecución es una tarea esencial. Se deben diseñar casos de prueba para detectar errores debidos a cálculos incorrectos, comparaciones incorrectas o flujos de control inapropiados. Las pruebas del camino básico y de bucles son técnicas muy efectivas para descubrir una gran cantidad de errores en los caminos." [33]

Los casos de prueba deben descubrir errores tales como: comparaciones entre tipos de datos distintos, operadores lógicos o de precedencia incorrectos, igualdad esperada cuando los errores de precisión la hacen poco probable, variables o comparadores incorrectos, terminación de bucles inapropiada o inexistente, variables de bucles modificadas de forma inapropiada.

La prueba de límites es el último de los pasos de la prueba de unidad. El programa falla en sus condiciones límites. O sea, a menudo aparece un error cuando se procesa el elemento *n-ésimo de un arrego n-dimensional, cuando se hace la i-ésima repetición* de un bucle de *i* pasos o cuando se encuentran los valores máximo o mínimo permitidos. Los casos que prueban las estructuras de datos, el flujo de control y los valores de los datos por encima de los máximos y los mínimos son apropiados para detectar errores.

Procedimientos de prueba de unidad.

Generalmente la prueba de unidad se considera como algo adyacente a la codificación. El diseño de casos de prueba comienza una vez que se ha desarrollado, revisado y verificado la sintaxis del código fuente. Es conveniente dar un repaso al diseño para

[33] Pressman S. Roger; Ingeniería de software; Mc Graw-Hill, 1993, México, pág. 670.

establecer los casos de prueba que probablemente descubrirán errores como los ya mencionados. Cada caso de prueba debe ir acompañado de un conjunto de resultados esperados.

Debido a que los módulos no son programas independientes, debe desarrollarse para cada prueba de unidad cierto software que conduzca y/o resguarde. Un conductor es un programa que acepta los datos del caso de prueba, pasa estos datos al módulo que se va a probar e imprime los resultados que sean relevantes. Los resguardos sirven para reemplazar módulos que están subordinados al módulo a ser probado, lleva a cabo una mínima manipulación de datos e imprime una verificación de la entrada y vuelve.

Los conductores y los resguardos son adicionales al software que debe ser escrito y generalmente no se adjunta al producto de software final.

Prueba de integración.

"La prueba de integración es una técnica para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción." [34] En esta prueba se lleva a cabo la integración de los módulos agrupados para conformar el sistema completo. Está relacionada con la detección de errores en el diseño y la codificación. El objetivo es tomar los módulos probados en unidad y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño.

En la prueba de integración existen dos tendencias: una es la integración no incremental y la integración incremental. La primera consiste en combinar todos los módulos por anticipado y se prueba todo el programa en conjunto, normalmente está integración no funciona ya que surgen un gran conjunto de errores, la corrección se hace difícil, puesto que es complicado aislar las causas al tener delante el programa entero.

En la integración incremental el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y de corregir. También se pueden probar completamente las interfaces.

Integración descendente.

La integración descendente es un planteamiento de integración incremental. Se integrán los módulos moviéndose hacia abajo por la jerarquía de control, comenzando con el módulo de control principal. Los módulos subordinados se van incorporando al módulo de control principal en la estructura, bien primero en profundidad, o primero en anchura.

[34] Pressman S. Roger; Ingeniería de software; Mc Graw-Hill, 1993, México, pág. 673.

La integración primero en profundidad integra todos los módulos de un camino de control principal de la estructura. La selección del camino principal es arbitraria y dependerá de las características de la aplicación. Basándonos en la FIG. 2.42, por ejemplo, si se elige el camino a mano izquierda, se integrarán primero los módulos M1, M2 y M5. A continuación se integrará M8 o M6. Luego se construyen los caminos de control central y derecho. En tanto que la integración primero en anchura incorpora todos los módulos directamente subordinados a cada nivel, moviéndose por la estructura en forma horizontal. Según la FIG. 2.42, los primeros módulos que se integran son M2, M3 y M4. Sigue el siguiente nivel de control M5 y M6 y así sucesivamente.

El proceso de integración se lleva a cabo en una serie de cinco pasos:

"1. Se usa el módulo de control principal como conductor de la prueba, disponiendo resguardos para todos los módulos directamente subordinados al módulo de control principal.
2. Dependiendo del enfoque de integración elegido se van sustituyendo los resguardos subordinados uno a uno por lo módulos reales.
3. Se llevan a cabo pruebas cada vez que se integra un nuevo módulo.
4. Tras terminar cada conjunto de pruebas, se reemplaza otro resguardo con el módulo real.
5. Se hace la prueba de regresión para asegurar que no se han introducido nuevos errores." [35]

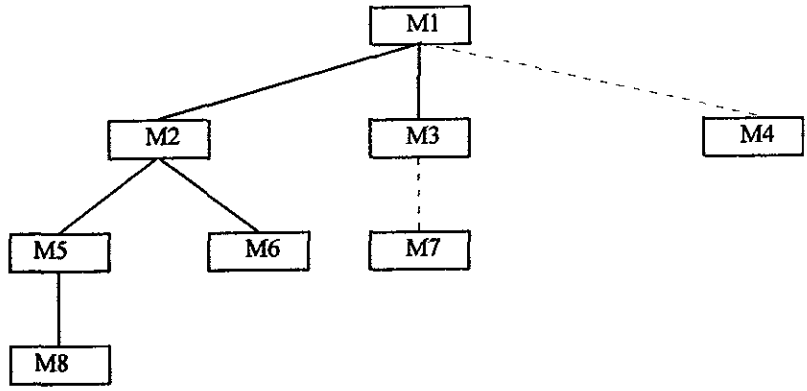


FIG. 2.42. Integración descendente.

[35] Pressman S. Roger; Ingeniería de software; Mc Graw-Hill, 1993, México, pág. 674.

El proceso continúa desde el paso 2 hasta que se haya construido la estructura del programa entero. La estrategia de integración descendente verifica los puntos de decisión o de control principales que aparecen más pronto en el proceso de prueba.

Integración ascendente.

La prueba de integración ascendente, empieza la construcción y la prueba con los módulos de los niveles más bajos de la estructura del programa. Dado que los módulos se integran de abajo hacia arriba, cuando se requiere que algún módulo subordinado esté disponible ya no es necesario crear resguardos.

Para que una estrategia de integración ascendente pueda ser implementada se listan los siguientes pasos:

"1. Se combinan los módulos de bajo nivel en grupos que realicen una subfunción específica de software.
2. Se escribe un conductor (un programa de control de la prueba) para coordinar la entrada y la salida de los casos de prueba.
3. Se prueba el grupo.
4. Se eliminan los conductores y se combinan los grupos moviéndose hacia arriba por la estructura del programa." [36]

La integración ascendente sigue el esquema ilustrado en la FIG. 2.43.

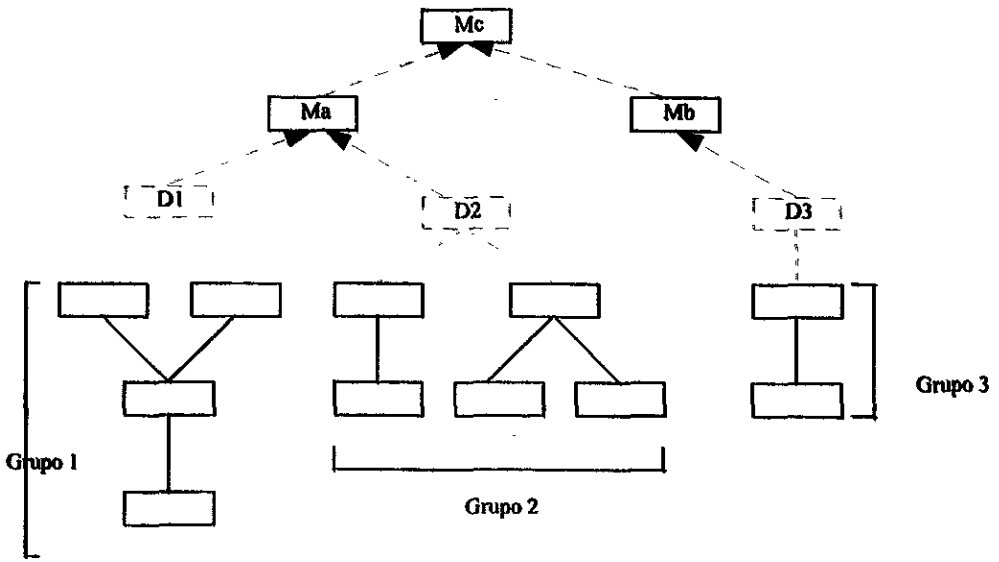


FIG. 2.43. Integración ascendente.

[36] Pressman S. Roger; Ingeniería de software; Mc Graw-Hill, 1993, México, pág. 676.

Se combinan los módulos para formar los grupos 1, 2 y 3. Cada uno de los grupos se somete a prueba mediante un conductor (ilustrado como un *bloque punteado*). Los módulos de los grupos 1 y 2 son subordinados de Ma. Se eliminan los conductores D1 y D2 y se conectan los grupos directamente a Ma. De forma similar, se eliminan el conductor D3 del grupo 3 antes de integrarlo con el módulo Mb. Finalmente, tanto Ma como Mb se integran en el módulo Mc y así sucesivamente.

Después de esta descripción de la prueba de integración es importante comentar que, la integración ascendente es lo contrario de la integración descendente. La principal desventaja del enfoque descendente es la necesidad de crear resguardos y la dificultad que asocia, esto puede quedar desplazado por la ventaja de poder probar las principales funciones de control. La principal desventaja de la integración ascendente es que el programa como entidad no existe hasta que sea añadido el último módulo, este inconveniente puede desaparecer si vemos que se tiene la facilidad de diseño de casos de prueba y que no se tienen que crear resguardos.

La selección de la estrategia de integración depende de las características del software. En general, la mejor opción puede ser un planteamiento combinado, que use la descendente para los niveles superiores de la estructura del programa, junto con la ascendente para los niveles subordinados.

2.9.5 PRUEBA DE VALIDACION.

La prueba de validación es una de las pruebas finales del software. La validación se logra cuando el software funciona de acuerdo con las expectativas del cliente y estás se definen a través de la especificación de requisitos.

La validación del software se consigue mediante una serie de pruebas que demuestran la conformidad con los requisitos. Se determina un plan de prueba, el cual consiste en trazar las pruebas que se han de llevar a cabo y un procedimiento de pruebas define los casos de prueba que se usarán para demostrar la conformidad con los requisitos. Tanto el plan como el procedimiento estarán diseñados para asegurar que se satisfacen los requisitos funcionales, que se alcanzan todos los requisitos de rendimiento, que la documentación es correcta.

Una vez que se lleva a cabo cada prueba de validación, pueden ocurrir dos condiciones: las características de funcionamiento o de rendimiento están de acuerdo con las especificaciones y son aceptables, otra que se dan cuenta que las especificaciones no se cubren y se encuentran deficiencias.

Un elemento importante en el proceso de validación es el repaso de la configuración del sistema, en donde se intenta asegurar que los elementos que componen el software se han desarrollado de forma correcta, y que están bien detallados para facilitar la fase de mantenimiento que más adelante será necesaria.

A lo largo del proceso de desarrollo de un software se llevan a cabo una serie de pruebas que van encaminadas a detectar errores, malas interpretaciones por parte, tanto del analista como del cliente. Dado que es virtualmente imposible desarrollar el sistema previendo totalmente el uso que el usuario pueda darle, se debe tomar en cuenta que se pueden interpretar mal las instrucciones de uso, se pueden usar extrañas combinaciones de datos, etc.

Por ello se realizan pruebas de aceptación para permitir que el usuario valide todos los requisitos. Estas son llevadas a cabo por el usuario final en lugar del equipo de desarrollo, una prueba de aceptación puede ir desde un informal paso de prueba hasta la ejecución sistemática de una serie de pruebas bien planificadas.

Cuando el software va a ser empleado por un grupo de usuarios, no es práctico realizar pruebas para cada uno de ellos. Es por ello que existe un proceso denominado *prueba alfa y beta,* cuyo objetivo es detectar errores que sólo el usuario final puede determinar.

La prueba alfa es realizada por un cliente en el lugar de desarrollo. Se usa el software de forma natural, con el encargado del desarrollo observando al usuario y registrando errores y problemas de uso. Esto es, las pruebas alfa se llevan a cabo en un entorno controlado.

La prueba beta se lleva a cabo en uno o más lugares en donde se encuentren los usuarios finales, normalmente el encargado del desarrollo no está presente. Así la prueba beta es una aplicación el software en un entorno que no puede ser controlado por el equipo de desarrollo. El usuario registra todos los problemas (reales o imaginarios) que encuentre durante la prueba e informa en intervalos regulares de tiempo al equipo de desarrollo. Como resultado de los problemas anotados por el usuario durante la prueba beta, el equipo de desarrollo lleva a cabo modificaciones.

2.9.6 PRUEBA DE VOLUMEN.

El propósito de este tipo de prueba es asegurar que el sistema pueda manejar el volumen de datos y transacciones de entrada especificados en el modelo de implantación del usuario, además de asegurar que tenga el tiempo de respuesta requerido. Esto puede requerir que el equipo que realiza el proyecto simule una gran red de terminales en línea, de manera que se puede engañar al sistema para que crea que está operando con una gran carga de datos.

De esta forma es posible generar grandes cantidades de datos a través de alguno de los instrumentos de confirmacion del sistema como pueden ser: generadores de datos de prueba, generadores de resúmenes de flujo, comparadores de archivos y simuladores. Y de esta forma, podemos hacer pruebas donde se manejen grandes volúmenes de datos y determinar los posibles errores en el sistema.

2.10 INSTALACION, MANTENIMIENTO Y ASEGURAMIENTO DE LA CALIDAD DEL SISTEMA.

2.10.1 PLAN DE INSTALACION.

La instalación requiere de construir un proceso en donde se combinen los elementos que integren al sistema para ejecutarse en una configuración en particular, esto involucra la compilación de elementos o el ligado de programas, donde se instalará el código objeto.

Para planear un programa de instalación se consideran algunos factores que a continuación se describen:

- Se deben tener todos los componentes que integran el sistema, además de un procedimiento de instalación y todas las rutas o archivos que se van a instalar.
- Verificar que se tenga la versión apropiada de cada componente requerido para la instalación del sistema.
- Disposición de todos los datos requeridos para la instalación.
- Los archivos de datos que hagan referencia con algún componente del sistema, deben tener el mismo nombre en la máquina donde se ejecutará el sistema.
- Revisar las versiones apropiadas de las herramientas necesarias. Las versiones existentes pueden ser incompatibles con la versión anterior que se utilizó para el desarrollo del sistema.
- Verificar qué componentes del código fuente existen en las diferentes versiones del sistema.

2.10.2 LA CAPACITACION.

Cuando se libera una versión siempre es necesario capacitar y enseñar a los usuarios sobre el uso del nuevo sistema. El programa de capacitación se elabora en las fases de diseño y codificación. Esto permite ir estructurando el nivel de conocimientos que se aplica al programa, pero depende del sistema el grado de complejidad de la capacitación. Algo similar pasa con la instalación, ésta puede ser muy simple o muy difícil, dependiendo del sistema.

Como se mencionó, la capacitación varía dependiendo del sistema, esto se conjuga con las diferencias que existen entre los usuarios que lo utilizarán, por ejemplo, el manejo y el uso varía entre las diferentes funciones que desempeñan cada uno, además del grado de conocimientos o de capacidad técnica que tengan los usuarios.

Para estructurar el programa de capacitación, el encargado debe considerar cada nivel y función de los usuarios, así como las tareas que realiza cada uno de ellos, además tiene que ir diferenciando los niveles de experiencia y jerarquía que exista entre el personal. En este punto es importante distinguir entre usuarios finales y administrador del sistema. Los usuarios finales necesitan la capacitación para las actividades que ayuden a desempeñar sus labores y el administrador necesita que se oriente más en cuestiones técnicas como son: el manejo del sistema, configuración y soporte técnico a usuarios finales.

2.10.3 LA CARGA DE ARCHIVOS.

Antes de iniciar el proceso de instalación se deben considerar algunos aspectos:

- El tipo de plataforma en donde se va a instalar el sistema.
- Las características técnicas de la computadora, está debe cumplir con los requerimientos del sistema.
- Un proceso de instalación automático, en donde se copien los archivos con la ruta correspondiente.
- La configuración de los posibles parámetros del sistema.

2.10.4 APROBACION FINAL.

Las pruebas para la aceptación de un sistema tienen que ayudar a demostrar que el sistema desarrollado satisface las necesidades que se establecieron en la definición del sistema. Algunos aspectos que se analizan para lograr la aprobación final son:

- Los requisitos que se verificarán.
- Las pruebas para los requisitos solicitados.
- La obtención del resultado esperado para cada prueba.

Una vez aprobada la instalación por el cliente, el sistema se entrega para iniciar la operación y se inicia la fase de mantenimiento, algunas de las actividades que se realizan dentro de esta etapa incluyen mejoras a las funciones desarrolladas, adaptación a nuevos ambientes y corrección de fallas menores en la programación.

2.10.5 IDENTIFICACION DE RESULTADOS Y DESVIACIONES.

Cabe mencionar que antes de liberar algún sistema se prueba de forma operativa y de manera exhaustiva durante un período de tiempo. Esto es, el mantenerlo trabajando de manera normal en la máquina donde va a instalarse definitivamente.

Es importante considerar que por la delicadeza del manejo de la información y la importancia de no parar las labores de la empresa, es necesario utilizar algún mecanismo que permita asegurar que la información se pueda seguir obteniendo oportunamente.

Una forma de realizar esto es tener funcionando el sistema anterior y el actual, e ir trabajando e introduciendo información en los dos sistemas paralelamente como si se trabajara normalmente, esto sirve para asegurar que el cliente no se quede sin la generación de la información debido a que falle el nuevo sistema.

Para el caso de ser el primer sistema para computadora, se instala y se captura la información y se observan los resultados obtenidos, pero paralelamente se sigue con el proceso anterior. Esto, para comprobar la validez de la información del nuevo sistema y así protegerse de posibles fallas del sistema implementado. Después de un período de tiempo de prueba, poco a poco se utiliza totalmente el sistema.

2.10.6 NORMAS DEL ASEGURAMIENTO DE LA CALIDAD DEL SISTEMA.

El proceso de calidad del sistema se aplica a lo largo de su desarrollo, Algunos elementos para lograr una calidad aceptable son:

- Los métodos y herramientas de análisis, diseño, codificación y pruebas.
- Las revisiones técnicas formales que se aplican en las etapas del desarrollo del proyecto.
- La elaboración de estrategias para realizar pruebas escalonadas.
- El control de la documentación que contengan los cambios realizados al sistema.
- Un procedimiento que fije los estándares utilizados en el sistema.
- Algunos mecanismos para medir la información.

Considerando los requerimientos solicitados y establecidos por el cliente y los elementos mencionados en los incisos anteriores se logra un producto con buena calidad.

Para considerar que un producto cumple con una calidad aceptable, es importante resaltar los siguientes puntos:

- Los requerimientos del sistema son la base fundamental para medir la calidad del programa. La inconsistencia con los requerimientos del usuario provoca que la calidad del producto sea deficiente.
- Los estándares que se definan deben establecer la forma como se aplica la ingeniería de software.
- El sistema debe ajustarse a los requerimientos del usuario.

Adicionalmente se contemplan otras actividades para lograr una mejor calidad del producto, las cuales se listan a continuación:

- Aplicar métodos y técnicas de control de calidad.
- Realizar revisiones técnicas formales.
- Apegarse a los estándares establecidos.
- Elaborar formas de control de cambios y modificaciones al sistema.
- Realizar pruebas al sistema.
- Realizar mediciones y estimaciones de la calidad.
- Elaborar registros e informes.

Un factor que afecta en la calidad del producto, son los cambios solicitados por el usuario durante el desarrollo. Es por esto, que es importante llevar un control de cambios aplicados durante el desarrollo del sistema y posteriormente durante la fase de mantenimiento al sistema.

2.10.7 REVISIONES TECNICAS FORMALES.

La calidad de un sistema no se considera posterior al desarrollo del sistema. Esta inicia desde el análisis y dentro de los requisitos establecidos por el usuario. Es por esto que el grupo encargado de la calidad del sistema, se auxilia de un conjunto de técnicas y herramientas que ayuden a conseguir una especificación del sistema con mayor calidad.

Una de las actividades que garantizan una mejor calidad del sistema, son las llamadas Técnicas de Revisión Formal (RTF). La RTF tiene como objetivo encontrar problemas en la calidad que presente el programa.

El grado en que se aplican los procedimientos y los estándares durante el proceso de la ingeniería del software varían. En algunos casos, los estándares los establece el cliente o ya existen normas definidas anteriormente. Los encargados del desarrollo del sistema son los responsables de vigilar que se cumpla con los estándares establecidos, esta actividad forma parte de una RTF.

Las revisiones no se aplican al final del proyecto, se aplican en diferentes puntos en el desarrollo y sirve para detectar errores que pueden ser eliminados en el momento en que se encuentren. Otra razón para utilizar RTF, es que aunque el personal involucrado en el desarrollo del proyecto sea eficiente, ellos buscan sus propios errores y por esto, otros errores se les escapan, por esta razón es conveniente que otra persona ajena realice una revisión externa y así poder encontrar el error que la otra persona no detecta.

Una RTF es una técnica efectiva que sirve para valorar la calidad del software. Un ejemplo es una presentación del diseño ante los clientes, ejecutivos y personal técnico. Las RTF son actividades que ayudan a garantizar la calidad del sistema.

Los puntos que a continuación se describen son algunos de los objetivos de las RTF:

- Detectar errores en las funciones, en la lógica o en la implementación del sistema.
- Verificar que el sistema cumple con los requerimientos.
- Garantizar que el sistema desarrollado se realizó de acuerdo con los estándares establecidos.
- Conseguir que el desarrollo del sistema sea uniforme.
- Lograr que los proyectos sean manejables.

Dentro de cualquier reunión para realizar una RTF, se deben contemplar las siguientes restricciones:

- Debe convocarse a la junta de revisión entre tres y cinco personas.
- Preparar con tiempo la reunión.
- La duración de la reunión debe ser menor de dos horas.

Con lo que se mencionó anteriormente, es conveniente que cada RTF se centre en una parte específica y pequeña del sistema. Esto es, en lugar de intentar revisar el diseño completo, se hacen revisiones por cada módulo o un pequeño grupo de módulos.

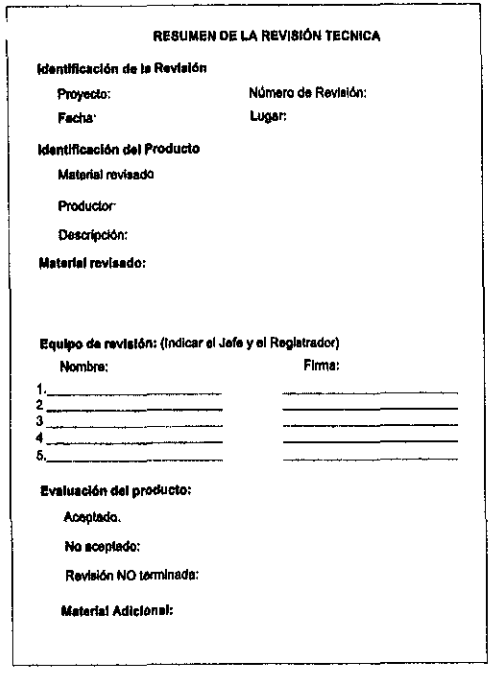
Registro e informe de revisión.

Durante las reuniones de la RTF, el encargado registra todas las modificaciones que surjan durante la revisión. Al final todas las modificaciones se resumen y se realiza una lista con todos los acuerdos a los que se llegó. Además, se prepara un resumen de la reunión de revisión. Este informe responde a tres preguntas básicas:

"1. ¿Qué fue revisado?.
2. ¿Quién lo revisó?.
3. ¿Qué se descubrió y cuáles son las conclusiones?."[37]

Un ejemplo de informe final de la RTF se ilustra en la FIG. 2.44. Este formato se adjunta con el registro histórico y puede ser distribuido al jefe del proyecto y otras personas involucradas en el mismo.

[37] Pressman Rogers; Ingeniería de software. Un enfoque práctico; McGraw Hill, 1993, México, pág. 589.



RESUMEN DE LA REVISIÓN TECNICA

Identificación de la Revisión

Proyecto: Número de Revisión:

Fecha: Lugar:

Identificación del Producto

Material revisado

Productor

Descripción:

Material revisado:

Equipo de revisión: (Indicar el Jefe y el Registrador)

Nombre: Firma:

1. _____ _____
2. _____ _____
3. _____ _____
4. _____ _____
5. _____ _____

Evaluación del producto:

Aceptado.

No aceptado:

Revisión NO terminada:

Material Adicional:

FIG. 2.44. Hoja resumen de la Revisión Técnica.

Adicionalmente, es necesario realizar una lista de actividades de la RTF, esto ayuda para:

- Identificar las áreas problemáticas del proyecto.
- Localizar el punto de acción para que los desarrolladores realicen las correcciones necesarias.

Directrices de la revisión.

Es importante establecer directrices para conducir las RTF, para que después sean distribuidas entre los revisores y finalmente aplicadas por el personal de desarrollo.

Algunas de las directrices a seguir son:

- Revisar el producto, no al productor. Se deben señalar los errores sin llegar a la confrontación.
- Establecer una agenda de trabajo y tratar de cumplirla.
- Limitar el debate y las impugnaciones.
- Enunciar las áreas con problemas, pero no intentar resolver cualquier problema que se ponga de manifiesto.
- Tomar notas escritas. Es importante que se tomen las declaraciones o la asignación de prioridades para ir registrando las medidas que se deban tomar.
- Limitar el número de participantes y preparar con anticipación la reunión.
- Desarrollar una lista de comprobaciones para que el producto pueda ser revisado.
- Disponer recursos y una planificación de tiempos para las reuniones.
- Repasar las revisiones anteriores. Estas pueden beneficiar para descubrir problemas en el propio proceso de revisión.

Las RTF son importantes para determinar la calidad de un producto, pero existen otros factores que nos ayudan a determinar la calidad que presenta el producto. Estos se clasifican en dos grupos:

- Factores que pueden ser medidos directamente.
- Factores que sólo pueden ser medidos indirectamente.

En cualquiera de los casos antes mencionados, se debe comparar el software, documentos y programas con alguna referencia para llegar a un indicativo de la calidad del sistema.

Los factores que afectan a la calidad del software se explican en los siguientes incisos:

- Corrección. El grado en que un programa realiza satisfactoriamente las especificaciones establecidas por los clientes.
- Fiabilidad. El grado en que se espera que los resultados del programa realicen sus funciones con la precisión requerida por el cliente.
- Eficiencia. La cantidad de recursos de computadora y de código requeridos por el programa para llevar a cabo las funciones.
- Integridad. El grado en que se controla el acceso al sistema o a los datos por personal no autorizado.
- Facilidad de uso. El esfuerzo requerido para aprender, trabajar, preparar la entrada e interpretar la salida de un programa.
- Facilidad de mantenimiento. El esfuerzo requerido para localizar y arreglar un error en un programa.
- Facilidad de prueba. El esfuerzo requerido para probar un programa de forma que se asegure que realiza las funciones requeridas.

- Portabilidad. El esfuerzo necesario para transferir el programa desde un hardware y/o entorno de sistemas a otro.
- Reusabilidad. El grado en que un programa (o partes de un programa) se puede reutilizar en otras aplicaciones.

Otro aspecto importante para medir la calidad de un software es establecer un esquema de graduación, que utiliza las siguientes características métricas:

Facilidad de auditoría.

La facilidad con que se puede comparar con los estándares establecidos.

Exactitud.

La precisión de los cálculos y el control del programa.

Completitud.

El grado en que se ha conseguido la total implementación de las funciones requeridas.

Concisión.

Lo compacto que es el programa en términos de líneas de código.

Consistencia.

El uso de un diseño uniforme y de técnicas de documentación a lo largo del desarrollo del proyecto.

Estandarización en los datos.

El uso de estructuras de datos y de tipos estándar a lo largo de todo el programa.

Tolerancia de errores.

El daño que se produce cuando el programa encuentra un error.

Eficiencia en la ejecución.

El rendimiento en tiempo de ejecución de un programa.

Modularidad.

La independencia funcional de los componentes del programa.

Facilidad de operación.

La facilidad para operar el programa.

Seguridad.

La disponibilidad de mecanismos para controlar o proteger los programas y la información.

Simplicidad.

El grado en que un programa puede ser entendido sin dificultad.

Facilidad de seguimiento.

La posibilidad de seguir paso a paso las instrucciones del programa hacia atrás, hasta llegar a los requerimientos.

Todo lo antes mencionado implica que la responsabilidad de garantizar la calidad del sistema, corresponde a todo el personal que trabaja en el desarrollo del sistema.

2.10.8 METRICA DE LA CALIDAD DEL SISTEMA.

Algunas de las métricas de calidad que se mencionaron son factores cualitativos, esto resulta difícil por la naturaleza subjetiva de las actividades que involucran a la ingeniería de software. Para resolver este problema, es necesario tener una definición más precisa de la calidad, además de una forma de obtener medidas cuantitativas de la calidad del sistema, como no existe un conocimiento absoluto no se puede esperar medir la calidad del sistema de forma exacta, ya que cada medida es parcialmente imperfecta.

2.10.9 CONFIABILIDAD DEL SISTEMA.

Es importante mencionar que la confiabilidad del programa depende de la calidad del sistema. Por ejemplo, si un programa presenta fallas frecuentemente en su funcionamiento la calidad del sistema no es buena, no importando si el resto de los factores de calidad del programa son aceptables.

La confiabilidad del sistema a diferencia de otros factores de calidad, puede ser medida o estimada mediante datos históricos o de desarrollo. Una definición de confiabilidad se puede definir en términos estadísticos como: *"la probabilidad de operación libre de fallos de un programa de computadora bajo un entorno determinado y durante un tiempo específico".*[38]

Medidas de la fiabilidad y de la disponibilidad.

En el caso de la confiabilidad del sistema todos las fallas se producen por problemas de diseño o de implementación, por otra parte, se puede pensar que existe un desgaste en el sistema por el uso que se le da, pero eso no se toma en cuenta por ser inexistente.

Para el caso de querer medir la confiablidad existen fórmulas que ayudan a realizar una estimación. Para un programa de computadora, se considera una medida de confiabilidad el tiempo medio entre fallos (TMEF), donde:

$$TMEF = TMDF + TMDR$$

Para esta ecuación las variables TMDF y TMDR corresponden al tiempo medio de falla y tiempo medio de reparación, respectivamente.

Adicionalmente para obtener una medida de confiablidad mayor se debe obtener una medida de la disponibilidad. La disponibilidad de un sistema es el porcentaje de que un programa opere de acuerdo con los requerimientos en un momento dado y que se define como:

$$Disponibilidad = TMDF / (TMDF + TMDR) 100 \%$$

La ecuación que define a la medida de fiabilidad del TMEF es dependiente del TMDF y al TMDR. Por otro lado la medida de disponibilidad es más sensible al TMDR, en donde se obtiene un porcentaje que nos define la disponibilidad. Los modelos de confiabilidad se usan para caracterizar y predecir el comportamiento del sistema.

[38] Pressman Rogers; Ingeniería de software, Un enfoque práctico; McGraw Hill,1993, México, pág. 589.

2.10.10 PROGRAMA DE MANTENIMIENTO.

El plan de mantenimiento se aplica posteriormente al desarrollo del sistema y debe ser planeando antes y durante el desarrollo del mismo, ésta no es una actividad aislada sino que es una parte integral del proyecto.

Durante el desarrollo del sistema se va elaborando la documentación de todas las actividades que se realizan, esto es útil al final del proyecto para hacer una correcta planeación del programa de mantenimiento.

Para identificar los documentos de dichas actividades, se utilizan los siguientes datos:

- El nombre del documento esquema.
- La relación entre documentos formales.
- La persona responsable para el chequeo de los documentos.
- La persona responsable para deliberar cada documento para el plan de mantenimiento.

Para ordenar la documentación del sistema se representa mediante una organización jerárquica, como se puede observar en la FIG. 2.45, en otras palabras la documentación generada se organiza de forma separada y ordenada, en su correspondiente módulo del proyecto.

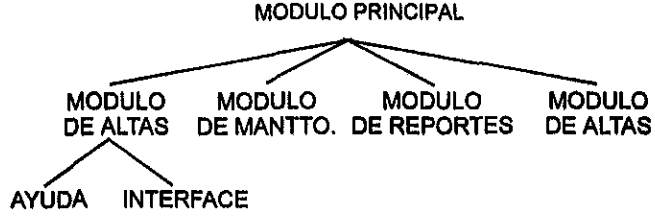


FIG. 2.45. Estructura jerárquica de la documentación.

En la FIG. 2.45 se observa la relación que existe entre los documentos de cada módulo del sistema, en esta parte el encargado del proyecto puede ir controlando la documentación necesaria para el mantenimiento.

Otro aspecto importante para desarrollar un plan de mantenimiento es definir un registro de lo más relevante de la configuración del sistema. Para esto se consideran las siguientes preguntas:

¿Qué características de equipo se requieren para ejecutar la nueva versión del *sistema*?.

¿Cuántas versiones han sido creadas y en qué fechas fueron creadas?.

¿Qué versiones del sistema son afectadas si se cambia algún componente del *sistema*?.

¿Cuántos cambios se requieren para pasar de una versión a otra?.

¿Cuántas fallas se presentan en la nueva versión?.

El plan de mantenimiento involucra procedimientos para realizar cambios en el control del sistema y en el manejo de las versiones de los archivos. En el mantenimiento se corrigen errores y se realizan pequeñas mejoras durante la vida del producto; la *implementación* de nuevas funciones y las adaptaciones a otros ambientes se consideran como nuevas actividades de desarrollo. Este tipo de modificaciones surgen como consecuencia de las necesidades y avances de la tecnología. Por ejemplo, los programas de computadoras cambian con el tiempo. Es por esto que el plan *mantenimiento envuelve el añadir* módulos y llevar a cabo alguna optimización del sistema. Para este caso, no sólo hay que cambiar la versión actual, sino también la versión anterior (que todavía sigue en uso) o las versiones futuras (que estén por liberarse). Además de los problemas que genera el realizar cambios para solucionarlos.

Por esta razón, se desarrollan mecanismos para evaluar, controlar y hacer las modificaciones necesarias.

El mantenimiento es mucho más que la corrección de errores, se *incluyen* cuatro actividades que se realizan tras liberar una versión del programa.

La primera actividad de mantenimiento se presenta cuando las pruebas del sistema no han descubierto el total de errores del *sistema. Esto surge cuando es utilizado el* programa y se encuentra algún error, el cual es informado al personal que desarrolló el sistema. Al proceso que incluye el diagnóstico y la corrección de uno o más errores se denomina mantenimiento correctivo.

La segunda actividad que presenta el mantenimiento, se produce debido al cambio tecnológico que existe en el mercado de hardware y software. Por otro lado, la vida útil del sistema puede fácilmente sobrepasar los diez años, haciéndose obsoleto para el entorno para el que fue desarrollado. Por esto, es necesario el mantenimiento adaptativo, esta actividad modifica el sistema para que interaccione adecuadamente con su entorno cambiante.

La tercera actividad que se aplica en el mantenimiento se realiza cuando el sistema tiene éxito. Esto es, a medida que se usa el sistema, se reciben de los usuarios recomendaciones sobre nuevas posibilidades, sobre modificaciones de funciones ya existentes y mejoras en general. Para esto se lleva a cabo el mantenimiento perfectivo.

La cuarta actividad se aplica periódicamente al sistema, en este tipo de mantenimiento se realizan actividades como revisiones periódicas, respaldos de información o actualización de archivos del sistema. A este tipo de mantenimiento se le llama preventivo.

En el momento que exista una amplia configuración, se empieza el mantenimiento con una evaluación de la documentación del sistema. A esto se le agregan las siguientes actividades que se tienen que realizar:

- Estudia el impacto de las correcciones o modificaciones requeridas y se traza un plan de actividades.
- Revisar si se modifica el diseño.
- Desarrollar nuevo código fuente.
- Realizar pruebas de regresión con la información contenida en la especificación de prueba.

Esta secuencia de actividades integran el mantenimiento del sistema y es el resultado de aplicar las técnicas de ingeniería del software.

Para el caso de mejoras futuras, hay que discutir la portabilidad del sistema. Durante las revisiones se evalúa el diseño de datos, el diseño estructurado y el diseño procedimental, para que sea fácil de modificar, tiene que ser modular y funcionalmente independiente. En las revisiones del código se cuida el estilo y la documentación interna y los factores que influyen para hacer más fácil el mantenimiento. Finalmente, cada paso de la prueba debe proporcionar anotaciones sobre las partes del programa que necesiten un mantenimiento preventivo antes de que el sistema sea formalmente liberado.

Las tareas que se asocian al mantenimiento del sistema principian a partir de que se realice una petición de mantenimiento. Inicialmente, se tiene que establecer una estructuración del mantenimiento, para esto se deben escribir procedimientos de evaluación y de información del sistema. Además, se establece un sistema de registro de información para las actividades de mantenimiento y se definen criterios de revisión y evaluación.

En el programa de mantenimiento primero se determina el tipo de mantenimiento que se llevará a cabo. En muchos casos el usuario ve una petición de mantenimiento como

una indicación de un error del sistema, mientras que el equipo de desarrollo ve la misma petición como una adaptación o una mejora.

Regularmente la evaluación de las actividades de mantenimiento se complica por la falta de datos que no estén registrados. Si se lleva un registro de información, se pueden realizar varias medidas del rendimiento del mantenimiento. A continuación se menciona una lista de posibles parámetros para medir el rendimiento:

- Número medio de fallos de procedimientos por ejecución del programa.
- Total de personas-horas por cada categoría de mantenimiento.
- Número medio de cambio por programa, lenguaje y tipo de mantenimiento.
- Número medio de personas-horas utilizadas para modificar código añadido o eliminarlo debido al mantenimiento.
- Media de personas-hora por lenguaje.
- Porcentaje de peticiones de mantenimiento por tipos.

Estas medidas proporcionan una base cuantitativa para la toma de decisiones sobre la técnica de desarrollo, la selección de un lenguaje y la previsión de esfuerzo para el mantenimiento.

CAPITULO 3

Estudio Inicial del Sistema

3. ESTUDIO INICIAL DEL SISTEMA

3.1 DEFINICION DEL SISTEMA.

Las metas económicas de casi todas las empresas están orientadas a mejorar los beneficios que perciba durante cierto período de tiempo. De ahí que orienten sus esfuerzos en la creación de programas de mejoramiento de los sistemas para el manejo de información como un recurso prioritario para el logro de sus objetivos. Por esta razón, una porción importante de todo programa de largo alcance, debe consistir en un análisis del sistema de información actual, para que descubra las mejorías que pueden hacerse con el esfuerzo necesario y gasto razonable, las cuales pueden ser fuente de ahorro inmediato en un volumen tal que signifique un peldaño hacia la constitución de un sistema de información administrativa que se baste por sí mismo. Más aún, esta clase de análisis suministrará la información necesaria para planear la transición del sistema anterior al nuevo, así como una base sólida con la cual comparar las necesidades del sistema.

Para el caso de estudio de la Empresa Inmobiliaria, el interés por analizar la situación actual surge a consecuencia de la falta de control de los inmuebles propiedad de la empresa. Por lo tanto, resulta necesario conocer cómo se encuentra operando la empresa, definir el problema, detectar cuál es la problemática del proceso y establecer una propuesta de solución.

Con base en la metodología mencionada en el capítulo II, los siguientes puntos nos permiten definir lo anterior y justificar la necesidad de implementar un sistema de cómputo para la Empresa Inmobiliaria.

En este caso lo llamaremos Sistema Integral de Administración Inmobiliaria, mismo que denominaremos SIAI de aquí en adelante.

3.1.1 DEFINICION DEL PROBLEMA.

La problemática que se detectó en la Empresa Inmobiliaria que representa nuestro caso de estudio, se describe a continuación:

El giro de la Empresa Inmobiliaria es el arrendamiento de inmuebles. Actualmente tiene en propiedad 753 inmuebles, los cuales se encuentran distribuidos en distintos estados de la República Mexicana. Estos inmuebles son administrados desde una oficina central, en la cual se mantiene un registro, tanto del inventario como de los mantenimientos realizados y por realizar a los mismos.

Para los inmuebles, se lleva un expediente que conforma su historial, en el cual se registran sus características y los cambios o modificaciones que van realizando a lo largo del tiempo. Dentro de los datos que forman parte de este expediente se encuentran: fotografías, planos, escrituras y demás documentos legales.

Esta información se encuentra archivada de acuerdo a la ubicación del inmueble, para posteriormente poder seleccionar los inmuebles que están disponibles dentro de alguna zona o colonia y tener la posibilidad de mostrarlos a los posibles clientes.

El cliente contacta a la Empresa Inmobiliaria telefónicamente y le indica sus requerimientos en cuanto a características del inmueble, por ejemplo: zona, número de habitaciones, m^2, monto de la renta, etc.

Una vez que el cliente ha elegido un inmueble, se genera el contrato.

Para operar óptimamente, los inmuebles deben recibir algún tipo de mantenimiento, el cual está clasificado en los tres grupos siguientes: permanentes o periódicos, preventivos y correctivos. Los mantenimientos periódicos son aquellos que se realizan con cierta periodicidad. Los mantenimientos preventivos son los que se realizan para prevenir las fallas y por último los mantenimientos correctivos son los que se llevan a cabo cuando ya se ha presentado la falla.

Para que éstos se lleven a cabo, es necesario formar un programa de mantenimiento en el cual se indican las partidas y conceptos que van a aplicarse.

Los programas de mantenimiento pueden ser individuales o globales. Son individuales cuando se aplican a un inmueble específico y es global cuando se aplican a un conjunto de inmuebles, ya sea que pertenezcan a una misma colonia o a un mismo conjunto habitacional.

Los mantenimientos son realizados por personal externo (contratistas), debido a que la empresa no cuenta con personal asignado para tales fines.

A estos programas de mantenimiento se les da seguimiento, para conocer su avance real. Para ello, de cada programa de mantenimiento se obtienen tres estimaciones: una

que realiza la empresa basándose en su propia experiencia, otra que realiza el contratista y por último la real.

La problemática radica principalmente en que:

- La información se captura en forma aislada, utilizando paquetería como hojas de cálculo, ocasionando con esto falta de comunicación y subdivisión entre las distintas áreas.
- Dado que la información se encuentra dispersa en diferentes computadoras, es evidente que los datos no interactúan entre sí, causando con esto la falta de oportunidad de la información, el retraso en el papeleo, así como la demora en la respuesta al cliente.
- Existe información redundante e inconsistente, lo cual causa pérdida de tiempo al tratar de integrar los datos cuando se requieren informes específicos.

3.1.2 JUSTIFICACION DEL SISTEMA.

Como se mencionó en la definición del problema, en la Empresa Inmobiliaria no existe control ni organización adecuada de la información, lo cual provoca duplicidad e inconsistencia de la misma.

Es necesario e importante contar con los datos al día, actualizados y unificados que permitan mantener el control de la administración de los inmuebles.

El Sistema Integral de Administración Inmobiliaria debe proporcionar la información oportunamente, además de mantener un control más eficiente sobre el inventario de inmuebles y la programación de mantenimientos.

En conclusión, la necesidad de crear un sistema que satisfaga las funciones principales de la operación de la Empresa Inmobiliaria para mantenerla en un alto grado de competitividad se fundamenta en los siguientes objetivos:

- Brindar un mejor servicio.
- Mejorar la capacidad de respuesta para servicios al cliente y áreas internas.
- Capacidad de respuesta en consultas al inventario de inmuebles.
- Obtención de reportes para la toma de decisiones.
- Mantener un control efectivo y ágil en la programación de mantenimientos.

3.1.3 METAS Y REQUERIMIENTOS DEL SISTEMA.

Metas.

Para que un sistema opere con eficiencia debe cumplir con las metas que justifiquen su implantación.

El Sistema Integral de Administración Inmobiliaria debe cumplir con las siguientes metas:

- Mejorar el control de la información.
- Mantener la seguridad de la información.
- Proporcionar accesos limitados a los diversos módulos.
- Interface gráfica y amigable para el usuario.
- Eliminar duplicidad de información.
- Reducir tiempos en operación.
- Reducir costos en operación.
- Realizar consultas e informes con oportunidad.
- Mejorar la comunicación entre las diversas áreas.

Requerimientos.

Los requerimientos básicos que debe cumplir el sistema para que opere en forma óptima, tanto en el presente como en un futuro inmediato, son los siguientes:

Confiabilidad.

Es necesario tener un sistema que proporcione información en forma eficiente, amigable, veraz y confiable.

Disponibilidad.

Es necesario que el sistema pueda ser accesado simultáneamente por varios usuarios dentro de la red y que el mantenimiento que se le proporcione o la incorporación de nuevos módulos no detenga la operación. El sistema debe operar en red porque así todos los usuarios podrán accesarlo en cualquier momento y con la frecuencia que lo requieran tanto para operarlo como para alimentar y disponer de la información de la base de datos.

Flexibilidad.

Es necesario que el sistema contemple la posibilidad de agregar nuevos módulos o nuevas opciones a futuro, sin que esto afecte a los ya existentes.

Expectativa de vida y potencial de crecimiento.

Es necesario que el sistema pueda adaptarse a los avances constantes de la tecnología, para así prolongar su vida útil y evitar su obsolescencia en el corto plazo.

Capacidad para recibir mantenimiento.

El sistema debe contar con documentación, estandarización de nombres y módulos que faciliten su mantenimiento.

Seguridad.

El sistema debe contar con las restricciones necesarias de acceso a los módulos, en base a las claves de seguridad asignadas a cada usuario.

Otros:

- El sistema deberá ser amigable, significando que su operación sea lo más sencilla posible sin causar problemas adicionales al usuario.
- Tener un sistema que facilite la captura y acopio de información.
- Tener un sistema que reduzca el tiempo de preparación de los reportes y sobre todo, que los tenga justo en el momento que se requieran.

3.1.4 RESTRICCIONES DEL SISTEMA.

- El sistema no toma en cuenta módulos que permitan el registro de los cobros y pagos derivados de los mantenimientos efectuados a los inmuebles, debido a que aún se están revisando los procedimientos manuales correspondientes.

- El sistema tiene restricciones en cuanto a espacio en disco duro y memoria RAM, dado que las imágenes digitalizadas (fotos de los inmuebles y planos) requieren mayores recursos para su operación.

- Por las causas mencionadas anteriormente, el sistema no podrá contener todos los documentos digitalizados que corresponden al inmueble tales como contratos, escrituras y demás documentos legales, debido a que ocuparían mucho espacio en disco, lo cual disminuiría el desempeño del sistema, en su primera etapa. Sin embargo se propone un proceso de captura de la información en medios que permitan el almacenamiento de grandes volúmenes de información, tales como discos duros.

3.1.5 CARACTERISTICAS DEL USUARIO.

El sistema está planteado para usuarios con conocimientos básicos en computación. Al tener esta aseveración como premisa del sistema, podemos garantizar que el uso del sistema no causará ningún problema para el usuario y tendremos la seguridad de que será herramienta útil para el mismo.

Dentro de la Empresa Inmobiliaria, los usuarios del SIAI, son en su mayoría, arquitectos e ingenieros civiles con experiencia en el uso de procesadores de palabra y hojas de cálculo. Aunque el personal que operará el sistema lleva a cabo sus tareas cotidianas en una red, no conoce las ventajas de compartir una base de datos ni los beneficios que puede obtener al utilizar un sistema integral de información.

Por lo anterior, es necesario capacitar a los usuarios para trabajar en un sistema en red, así como la secuencia que deberán observar para ejecutar los procesos correspondientes a cada área.

3.1.6 CRITERIOS DE ACEPTACION DEL SISTEMA.

El sistema será aceptado en el momento que permita realizar las actividades que ahora se llevan a cabo de forma manual, cumpliendo con las metas establecidas y respetando las restricciones definidas.

3.1.7 FUENTES DE INFORMACION.

La información concerniente para análisis del sistema, debe ser proporcionada por las áreas responsables de controlar el inventario de inmuebles, así como de las encargadas de dar seguimiento a los programas de mantenimiento.

3.2 DIAGNOSTICO DE LA SITUACION ACTUAL.

En la actualidad, la Empresa Inmobiliaria no cuenta con un sistema óptimo que ayude a controlar la información de los inmuebles. Los procesos se realizan manualmente y no existe una buena coordinación entre las diferentes áreas, por lo cual existen capturas duplicadas e inconsistencia en la información. Aunado a esto, la información no se encuentra concentrada en un punto común, sino que está dispersa y con distintos responsables.

El retraso en el manejo de la información trae consigo varias consecuencias, que se mencionan a continuación:

- Se generan retrasos en la respuesta al cliente, lo cual ocasiona que las personas interesadas busquen alguna otra empresa del ramo que les proporcione un servicio más eficiente.
- Se generan retrasos en los pagos a los contratistas, lo que significa que éstos no terminen su trabajo asignado en el plazo convenido y ello implica que el cliente no quede satisfecho.

Lo anterior conlleva a pérdidas económicas para la empresa, lo que hace que la situación sea cada vez más difícil, y se ha llegado a la conclusión de que debe haber un cambio estructural para que el flujo, concentración y manejo de la información sea eficiente y se generen beneficios tangibles y directos.

3.2.1 RECONOCIMIENTO DE RECURSOS.

El personal de la empresa cuenta con los recursos necesarios en cuanto a herramientas de trabajo se refiere (hardware y software), para operar satisfactoriamente el sistema.

A continuación, en la FIG. 3.1 se presenta el organigrama general y se describen las principales áreas que constituyen a la Empresa Inmobiliaria, enunciando la función que desempeña cada una de ellas.

ORGANIGRAMA GENERAL DE LA EMPRESA INMOBILIARIA

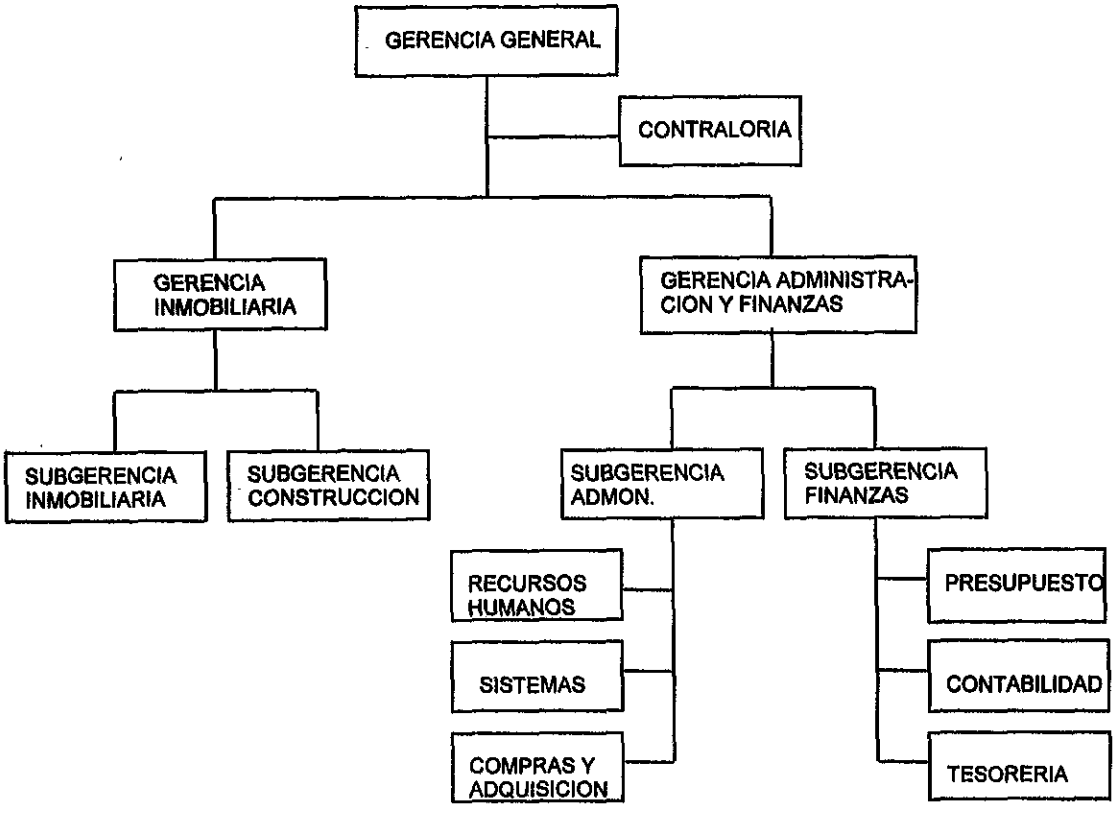


FIG. 3.1. Organigrama general de la Empresa Inmobiliaria.

Gerencia general.

Establece metas organizacionales, coordina y supervisa a todas las áreas que integran a la empresa con el fin de cumplir con los objetivos establecidos por la misma.

Gerencia de Administración y Finanzas.

Gerencia que se encarga de fijar las políticas y normas de operación de la empresa de inmuebles, cuidando que éstas se cumplan de acuerdo al programa financiero establecido dentro y fuera de la empresa.

Gerencia Inmobiliaria.

Gerencia encargada de administrar los inmuebles, además de llevar el control de contratistas, trámites legales y programación de mantenimientos.

Contraloría.

Area encargada de vigilar el cumplimiento de la normatividad de la empresa.

Subgerencia de Administración.

Area encargada de mantener el control de los recursos materiales y humanos de la empresa.

Subgerencia de Finanzas.

Area encargada de administrar los recursos financieros de la empresa.

Subgerencia Inmobiliaria.

Area encargada de mantener el control del inventario de los inmuebles, así como de los trámites legales para el buen funcionamiento de los mismos.

Subgerencia de Construcción.

Area encargada de realizar la programación de los mantenimientos, así como la adecuación de los inmuebles.

Presupuestos.

Area encargada de administrar el control presupuestal.

Recursos Humanos.

Area encargada de llevar a cabo la administración y optimización de los recursos humanos y materiales de la empresa, cuidando que éstas se apeguen a las políticas y normas establecidas por la empresa.

Sistemas.

Area encargada de planear, administrar, desarrollar, controlar e implementar procesos computarizados que automaticen con mayor eficiencia las principales tareas de la empresa, así como proporcionar soporte tanto al equipo de cómputo como a las áreas usuarias.

Compras y Adquisiciones.

Area encargada de la adquisición de bienes de consumo y bienes instrumentales.

Contabilidad.

Area encargada de planear, organizar y mantener permanentemente actualizados los procesos administrativos, flujos de documentación y el esquema de régimen contable de la empresa, a fin de producir información fiscal y el cumplimiento presupuestal, de forma completa confiable y oportuna que cubra todos los requerimientos de la empresa, así como las regulaciones emitidas por instancias gubernamentales.

Tesorería.

Area encargada de optimizar los recursos financieros, manejar el flujo de efectivo eficientemente para lograr el mínimo de excedentes y por consiguiente el mejor uso de dichos recursos.

En la FIG. 3.2 se puede apreciar de manera esquemática a la Empresa Inmobiliaria con la información desorganizada, donde cada quien maneja archivos y documentos con un equipo de cómputo independiente. Esto ocasiona que no se tenga control sobre el inventario de los inmuebles y que la petición de mantenimientos a los contratistas llegue con cierto retraso.

Todo ello entorpece las operaciones de la empresa, lo cual no permite una adecuada atención de los clientes y si deteriora la imagen de la misma.

En este esquema se plasman los principales problemas que se tienen, los cuales son básicamente: información descentralizada e inventario desorganizado, lo que conlleva a no cumplir oportunamente con los clientes y produce una situación crítica.

SITUACION ACTUAL

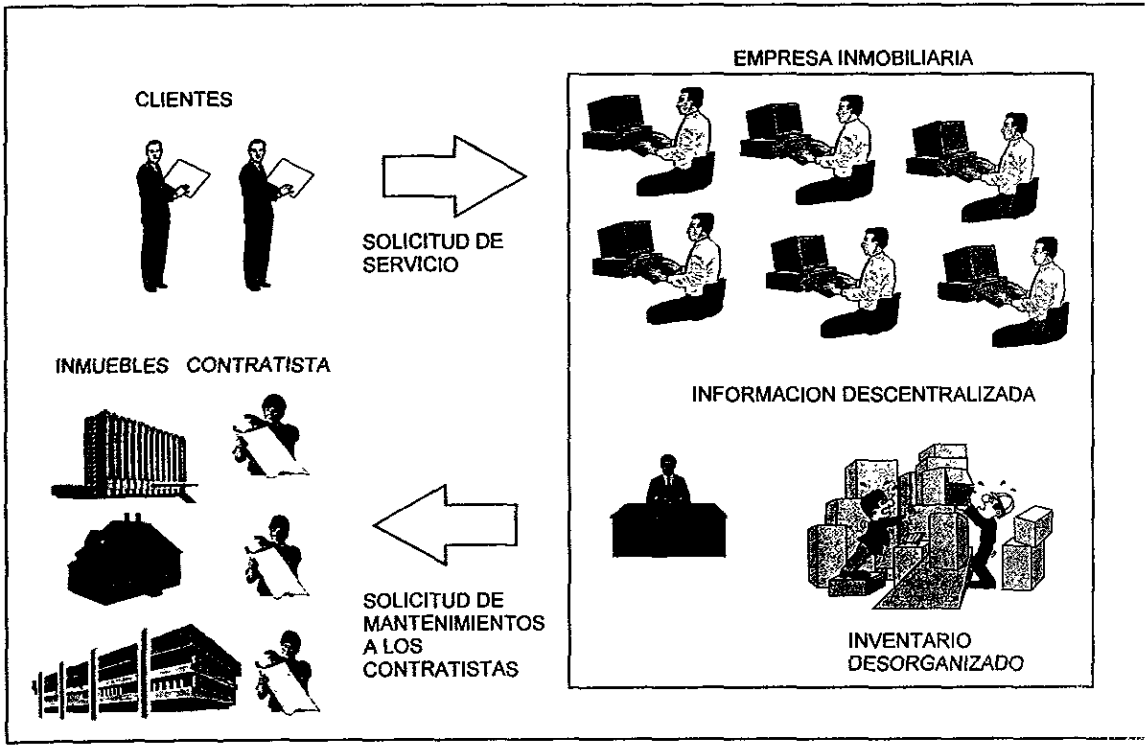


FIG. 3.2. La empresa inmobiliaria con información desorganizada.

La FIG. 3.3 muestra a la Empresa Inmobiliaria con la implantación de un sistema integral de cómputo en una red de computadoras. En este esquema puede apreciarse como a través de un sistema diseñado para funcionar en una red de cómputo, muchas tareas pueden simplificarse, otras evitarse y la mayoría mejorarse.

En primer lugar, se observa que la información se centraliza en un mismo punto (una base de datos), lo cual ayuda a evitar el aislamiento de la misma.

En segundo lugar, se advierte que los usuarios del sistema accesan la misma información, evitando de esta manera capturas innecesarias, ya que varios usuarios pueden tener la misma información en sus computadoras aunque clasificada de diferente manera.

Además, la comunicación entre los usuarios del sistema mejora, ya que ahora tienen que compartir los mismos datos y se hace necesario definir reglas para el flujo de la operación.

Evidentemente, estos puntos van a redundar en un mejor control de la información y esto a su vez en reducción de tiempos de operación.

Con este planteamiento se puede disponer de la información en cualquier momento, evitando retrasos en el arrendamiento de los inmuebles que puedan llevar a la empresa a incumplimientos en los contratos.

Permite al personal de la empresa tener un control sobre el inventario de los inmuebles, además de llevar la programación de mantenimientos ordenadamente, de tal forma que la petición a los contratistas sea oportuna y éstos puedan desempeñar su trabajo sin contratiempos.

En consecuencia, esta propuesta va encaminada a la satisfacción del cliente y al buen funcionamiento de la empresa en el ámbito inmobiliario.

SOLUCION PROPUESTA

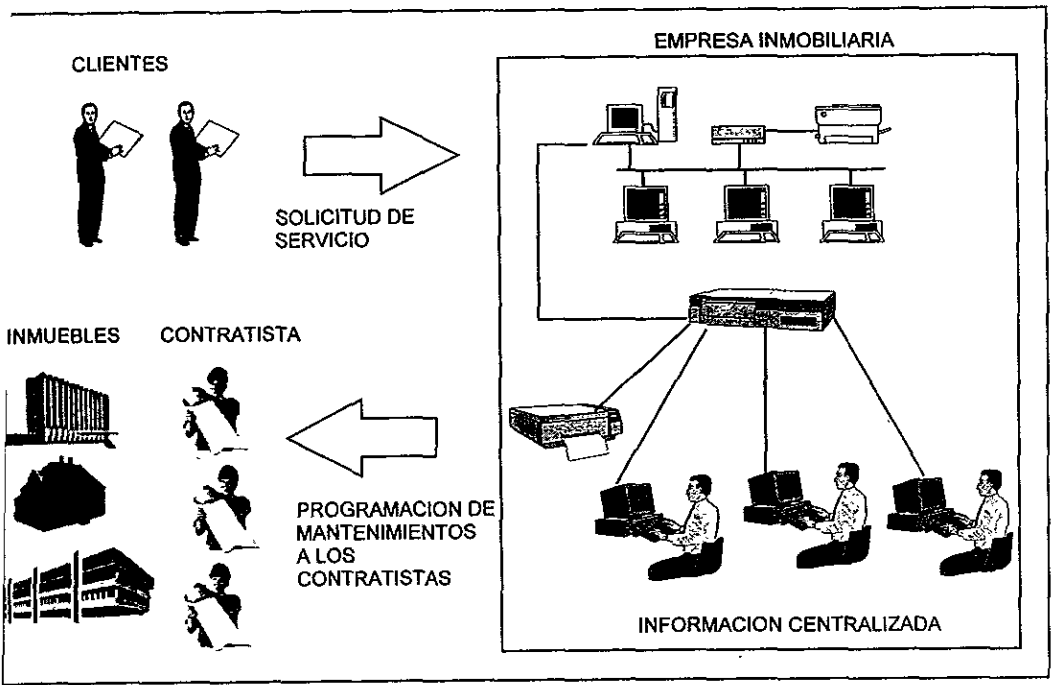


FIG. 3.3. La empresa inmobiliaria con la implantación de un sistema integral de cómputo en una red de computadoras.

3.2.2. TECNICAS USADAS PARA LA RECOPILACION DE INFORMACION EN EL DIAGNOSTICO DE LA SITUACION ACTUAL.

En cada fase del proceso de desarrollo del sistema de información, el analista utiliza ciertas herramientas y técnicas específicas para lograr sus metas y objetivos. Como todos los buenos artesanos, algunos analistas se valen con mayor frecuencia de una herramienta o técnica determinada; pero nosotros proponemos que se lleve a cabo un proceso sistemático e ingenieril basado en una metodología específica para obtener información.

Algunas de las herramientas más comunes que se utilizaron fue la entrevista.

La entrevista.

Dentro de una organización, la entrevista es una técnica muy significativa y con alto grado de productividad de que se dispone para recabar datos. En otras palabras, la entrevista es un intercambio que se efectúa cara a cara. Sirve para obtener información acerca de las necesidades y la manera de satisfacerlas. Por otra parte, la entrevista ofrece al analista una excelente oportunidad para establecer una corriente de simpatía con el personal usuario, lo cual es fundamental en el transcurso del estudio.

Entrevista con los directivos

FIG. 3.4. La entrevista.

Pues bien, de los métodos conocidos para la recopilación de la información se llevaron a cabo entrevistas con los responsables de cada una de las áreas que intervienen en el proceso de la administración inmobiliaria. Las personas entrevistadas fueron las siguientes: Gerente Inmobiliario, Subgerente inmobiliario, Subgerente de construcción, Jefe de Departamento del Area Inmobiliaria y analistas.

Las preguntas principales del cuestionario que se aplicó en cada una de las entrevistas se encuentran en el Anexo A.

3.3 ANALISIS DE FACTIBILIDAD.

Para poder tomar la mejor decisión en cuanto a la implantación de un sistema para automatizar eficientemente los procesos de la Empresa Inmobiliaria, se realizó una evaluación de los productos existentes en el mercado, que cumplieran con los requerimientos ya mencionados.

Como resultado de dicha evaluación se encontraron dos posibles soluciones:

1. Diseñar un sistema a la medida, que contemple los objetivos, requerimientos, metas y restricciones recopiladas en el análisis.
2. Comprar un programa comercial ya desarrollado y adaptar la operación de la Empresa Inmobiliaria al mismo.

En la búsqueda de sistemas existentes, se encontró únicamente uno que cumplía con los requerimientos establecidos por la Empresa Inmobiliaria, mismo que se presenta a continuación:

Análisis del sistema desarrollado a la medida.

Sistema 1.

Sistema desarrollado a la medida.

Análisis del sistema comercial ya existente en el mercado.

Sistema 2.

Empresa desarrolladora : Strategies.
Representante en México: Raxer, S.A. de C.V.
Software: Cadwin Facilities Management.
Dirección: Copa de Oro 40, Col. Ciudad Jardín 04370 México, D.F.
Teléfonos: 5 44 26 00 / 6 89 19 34.

Características generales del sistema.

Es un sistema que permite administrar el patrimonio inmobiliario de la empresa, a través de la integración de gráficos y datos. Ofrece las siguientes perspectivas:

- Gestión de superficies.
- Repartición de los costos operativos.
- Planificación del mantenimiento.
- Generación de estimaciones.
- Automatización de inventarios.
- Organización de los trabajos de mudanza.
- Simulación de reorganización.
- Reducción de los costos de contratistas.
- Anticipación a la evolución.
- Mejoramiento en los tiempos de respuesta de mantenimiento de las diversas redes.

Software.

- Cadwin FM.
- ORACLE Versión 6/7.
- PC NFS/TCPIP.

Plataformas en las que puede operar.

- MS-DOS (versión 5 o superior).
- Windows 3.11 o 95.
- Windows NT.
- Novell 3.xx/4.xx.
- UNIX.

Hardware.

Estación de trabajo.

Características:

- Procesador 486.
- 32 MB de RAM.
- 500 MB de disco duro.

Costo.

Por estación de trabajo: $15,000 USD (para plataforma de red).

Análisis de los componentes de factibilidad para los dos posibles sistemas.

Factibilidad técnica.

Sistema 1: Para el desarrollo del sistema se emplean el hardware y las herramientas de software con las que cuenta la empresa, en este caso Visual Basic y Access.

Sistema 2: Para la implementación de este sistema debe considerarse la adquisición de nuevo software, tanto de bases de datos (Oracle), como gráfico (Autocad) y nuevo hardware, lo cual implica capacitación en estas tecnologías.

Factibilidad económica.

Sistema 1: El costo de este sistema se basa fundamentalmente en lo que representa el desarrollo del sistema, ya que sólo implica costos adicionales por licencias de software (para el caso de Visual Basic) y ningún costo en hardware debido a que la Empresa Inmobiliaria ya cuenta con estos recursos. El desglose de los costos y tiempos se encuentra detallado en el Capítulo 4.

Sistema 2: La implantación de este sistema implica una inversión muy elevada debido a que el costo de cada licencia es de $15 000 USD (porque la empresa requiere un sistema bajo un ambiente de red) y por lo menos se requieren 5 licencias. Aunado a esto, deben adquirirse licencias de Oracle.

Debido a que este sistema funciona en un ambiente gráfico, es necesaria la adquisición de memoria RAM adicional para las computadoras, que actualmente cuentan con 8 Mbytes. También se hace necesaria la ampliación de los discos duros debido a que cuentan únicamente con 400 Mbytes. Además, debe considerarse que, ni el personal de sistemas ni el operativo, se encuentran capacitados para utilizar Oracle y AutoCad y que la capacitación implica invertir mayor tiempo y costo.

Factibilidad legal.

No existen transacciones que impliquen cuestiones legales. Existe un módulo que contiene toda la información legal referente a los inmuebles, pero ésta es únicamente descriptiva.

En cuanto a las licencias del software:

Sistema 1: Se cuenta con las licencias necesarias de Visual Basic y Access.

Sistema 2: Deben adquirirse las licencias respectivas de Oracle y las del sistema.

Factibilidad operacional.

Sistema 1: Debido a que es un sistema a la medida, se han ajustado los procedimientos a las necesidades de la Empresa Inmobiliaria tomando en cuenta la organización existente.

Sistema 2: Debido a que es un sistema comercial ya desarrollado, la Empresa Inmobiliaria debe adaptar sus procedimientos existentes al software y capacitar al personal en los nuevos procesos.

Factibilidad de programa.

Sistema 1: El arranque del sistema implica mayor tiempo, debido a que el sistema partirá de cero.

Sistema 2: El arranque del sistema sería relativamente rápido, debido a que es un sistema comercial listo para instalarse. Evidentemente debe hacerse un análisis para poder adaptar la operación de la empresa a los nuevos procesos.

Como puede apreciarse en el análisis anterior, el sistema dos satisface casi todos los requerimientos de la Empresa Inmobiliaria, siendo el alto costo uno de los inconvenientes. Esta opción es eliminada por el alto costo que implica su implantación y deja como única alternativa el desarrollo de un sistema a la medida.

3.4. ANALISIS DEL SISTEMA.

Proceso Actual.

La Empresa Inmobiliaria cuenta con una área encargada de mantener el control de los inmuebles: La Gerencia Inmobiliaria, la cual a su vez se subdivide en dos subgerencias: Construcción e Inmobiliaria.

La Subgerencia Inmobiliaria mantiene un registro de los inmuebles clasificados por zona, por tipo, por estado, etc.

Los programas de cómputo utilizados para el registro de la información consisten básicamente en hojas de cálculo.

El inconveniente es que el formato de registro no es el mismo para todos los inmuebles. Esto se explica de la siguiente manera: El trabajo de captura de los datos que conforman al inmueble es realizado por distintas personas que no han unificado criterios, lo cual ha ocasionado que cada una diseñe su propia hoja de captura conforme a sus necesidades y conocimientos.

Cabe mencionar que cada vez que se requiere una nueva clasificación de la información se vuelve a crear otra hoja con distribución diferente y los datos se vuelven a capturar, por varias personas y en computadoras distintas.

Aproximadamente cada 6 meses se requiere el reporte actualizado de todos los inmuebles. Para esto, se imprimen los reportes existentes y se revisan detalladamente para eliminar los inmuebles repetidos. Finalmente se juntan en otra hoja de cálculo. Esta tarea la llevan a cabo dos analistas.

Un tercer analista mantiene su propio registro con la información legal concerniente a los inmuebles y un cuarto analista se dedica a la elaboración de planos.

La Subgerencia de Construcción se encarga de supervisar que los inmuebles se encuentren en buenas condiciones, para lo cual lleva a cabo la programación de los mantenimientos y da el seguimiento necesario para que se ejecuten en los tiempos establecidos.

Cuenta con tres analistas, los cuales tienen a su cargo diversas zonas. Cada analista controla una zona, para la cual mantiene el registro de sus mantenimientos, avances y contratistas. Un inmueble que recibió un mantenimiento el año anterior, cuenta con un archivo respectivo de ese año. Para este año se volvería a crear una hoja nueva con los datos del inmueble y los conceptos del nuevo mantenimiento.

Al igual que en la Subgerencia Inmobiliaria en ocasiones es necesario reunir información de diversas zonas y se repite el proceso anterior de imprimir e inclusive capturar la información.

Es evidente que las computadoras están saturadas con hojas de cálculo con información clasificada de diversos conceptos, lo cual dificulta su unificación.

Proceso con el Sistema Integral de Administración Inmobiliaria (SIAI).

Con el SIAI las funciones antes mencionadas pueden simplificarse notablemente.

La Subgerencia Inmobiliaria únicamente va a capturar una sola vez los datos de un inmueble: datos generales y datos legales. Para evitar la duplicidad de la información se contará con un registro único para cada inmueble. Aunado a esto, podrá anexarse una fotografía digitalizada de cada inmueble, así como un plano arquitectónico.

Cada vez que un inmueble sufra alguna modificación, se guardarán estos cambios en archivos históricos. Los analistas del área podrán consultar todos los inmuebles simultáneamente y verán la misma información.

Para la elaboración del reporte semestral bastará con solicitar un informe que permita ser clasificado de distintas maneras.

Para la Subgerencia de Construcción el sistema proporcionará los mismos beneficios. Dado que el inmueble ya ha sido capturado por la Subgerencia Inmobiliaria, la Subgerencia de Construcción únicamente realizará la captura de los catálogos que permitirán elaborar los programas de mantenimiento, así como registrar y mantener un catálogo de contratistas.

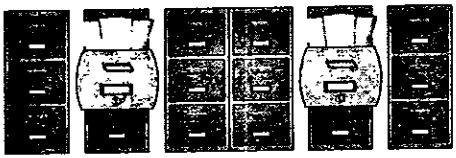
Los programas de mantenimiento deberán actualizarse continuamente para posteriormente elaborar comparativos.

Puede apreciarse inmediatamente los beneficios que proporcionaría a la Empresa Inmobiliaria contar con un Sistema de cómputo que permita controlar con eficiencia su información.

El contar con informes veraces y oportunos, permite a los altos mandos tomar mejores decisiones, lo cual se traduce a su vez en minimización de costos y tiempos.

Recopilación de Información.

La recopilación de información de documentos fuente, hojas de trabajo, informes, etc. es otro medio por el cual, el analista obtiene información durante la fase de análisis de sistemas. En esos documentos puede encontrar una idea de lo que se hace actualmente, cómo ha sido estructurado, de qué elementos no se dispone. Se hace referencia a la recopilación de documentos en la FIG. 3.5.



Otro medio para obtener información para el análisis, es la recopilación de documentos.

FIG. 3.5. Recopilación de Documentos.

Para la obtención de información referente a la Empresa Inmobiliaria se asistieron a juntas periódicas realizadas en las oficinas de la Gerencia Inmobiliaria

Los tópicos que se trataron en estas reuniones fueron:

- Datos generales de la información.
- Aspectos legales.
- Programación de mantenimientos.
- La estimación de costos.

En estas reuniones, la información obtenida fue la siguiente:

- Se nos describió la normatividad a la que está sujeta la empresa.
- Se nos proporcionaron los formatos en los que vacían su información.
- Se nos proporcionó la información que contienen los manuales de Conceptos de Mantenimiento y las Partidas que se manejan.
- Se nos proporcionaron los datos que contempla el inventario de los inmuebles.
- Se acordaron para cada uno de los documentos, los campos que se deberían llenar y el tipo de información que lleva cada uno de ellos.
- Se nos proporcionaron fotografías de las fachadas de algunos inmuebles de los que es propietaria la empresa.
- Se definió la metodología a seguir para la Programación de Mantenimientos.
- Se nos proporcionaron las características del equipo de cómputo con el que cuenta la empresa.
- Se proporcionó el procedimiento para la estimación de costos que se realizan en la empresa.
- Se acordó proporcionar una metodología a seguir para el cumplimiento de los programas de mantenimiento por parte de los contratistas.

Clasificación de la Información.

Al llegar a esta fase del análisis del sistema, se ha reunido una gran cantidad de requerimientos para definir cómo será el sistema y lo que debe hacer. Para lograr un

conocimiento más completo de los datos recolectados se pasa a otro nivel de análisis en donde se hace la clasificación de los mismos por características específicas. De los datos obtenidos en reuniones y entrevistas se hizo la siguiente clasificación de información:

La relativa a los documentos que se manejan.

- Formatos de Aspectos Legales.
- Formatos de Aspectos Físicos.
- Formatos de Datos Generales.
- Formatos de Componentes.
- Formatos de Partidas.

La relativa al manejo de las fotografías.

- Propuesta de software a utilizar para la captura de las fotografías.
- Tamaño de la fotografía a digitalizar.
- Procedimiento para la digitalización de las fotografías.
- Relación de las fotografías que conforman el inventario de inmuebles.

La relativa a la captura de la información.

- Características de la información de entrada.
- Definición del tipo de datos que se manejarán en cada campo.
- Clasificación de datos para cada documento.
- Definición del tipo de reportes.

La relativa al inventario de inmuebles.

- Determinación de la información a inventariar.
- Determinación de los catálogos a utilizar.
- Clasificación de los datos de los catálogos.

La relativa a la Programación de Mantenimientos.

- Descripción de tipos de mantenimiento.
- Determinación de los catálogos a utilizar.

La relativa a las consultas de la información.

- Procedimientos de consultas generadas por el usuario.

CAPITULO

4

Planeación del Sistema

4. PLANEACION DEL SISTEMA

4.1 EL ALCANCE DEL SISTEMA.

El sistema deberá cumplir con las metas, requerimientos y restricciones establecidas por el usuario, sin embargo, quedó establecido que no cubrirá la totalidad de la operación de las áreas de la Empresa Inmobiliaria que conforman: Cobros de Servicios Prestados y Facturación, Pagos de Gastos, Impuestos y Servicios Municipales, Administración y Finanzas, Recursos Humanos, Contabilidad, Tesorería, etc. ya que éstas cuentan con sistemas independientes para el control propio de su operación.

Por lo tanto, el sistema se enfocará a resolver las necesidades primordiales para el procedimiento general de una operación inmobiliaria. Esto es, se involucran tareas tales como: Registrar Datos Generales, Antecedentes, Inventario y Actualización del mismo, Programación de Mantenimientos, Estimación de Costos, Ejecución de Mantenimientos y Reparaciones, así como la generación de Reportes; además, los usuarios podrán diseñar el formato de consulta que ellos deseen.

Tomando en cuenta que se maneja un alto volumen de información y que debe estar centralizada en un solo lugar, el sistema deberá contar con una red que le permita operar en todas y cada una de las estaciones de trabajo de la manera más rápida, eficiente y confiable posible.

A continuación se presentan los alcances del sistema definidos de manera independiente para cada módulo.

El Sistema Integral de Administración Inmobiliaria contará con módulos que cubran los procesos de:

1. Inventario de Inmuebles.
2. Programación de Mantenimientos.
3. Estimación de Costos y Tiempos de Mantenimiento.
4. Ejecución de Mantenimientos y Reparaciones.

1. INVENTARIO DE INMUEBLES.

Permite registrar y mantener actualizado el inventario de inmuebles. Para cada inmueble es posible registrar sus datos generales, componentes (recámaras, cuarto de servicio, piscina, etc.), ampliaciones, reparaciones y gastos, así como los antecedentes de propiedad de uno o varios propietarios. Estará conformado por los siguientes procesos:

Inmuebles Datos Generales.

Se refiere a la captura y validación de los datos generales del inmueble, tales como: clave, tipo y descripción del inmueble, estado, ubicación, colonia, localidad, precio de venta, uso de suelo (indica en que se está utilizando el suelo: oficinas, comercio, casa-habitación), tipo de construcción (indica si la construcción es permanente ó provisional), sistema de construcción (indica los materiales de la construcción: concreto, metálica, mixta, prefabricada), estado de conservación (indica si el inmueble esta en excelente estado, bueno o regular) y antigüedad de la construcción.

Aspectos Legales.

Este proceso permitirá la captura y validación de la documentación legal que se maneja respecto a los Inmuebles, tales como: predio (indica si es un terreno), edificación (indica si existe alguna construcción), superficie del terreno, superficie construida, colindancias, valor al que se adquirió el inmueble, número de registro público de la propiedad, número del registro del catastro, número de boleta predial, número de boleta del agua, valor estimado del inmueble, licencia de construcción y licencia de uso de suelo.

Componentes.

El registro de componentes contempla el inventario de las partes que integran el Inmueble y sus características principales, tales como: número de recámaras, área del cuarto de servicio, capacidad de la cisterna, etc. Permitiendo hacer una clasificación en componentes generales y componentes a detalle. Los datos que se registran proporcionan al promotor y al cliente información de gran utilidad en el proceso de compraventa.

Gastos.

Procedimiento que permite almacenar en el sistema los gastos por concepto de reparaciones, impuestos y servicios que fue necesario erogar para la habilitación de los inmuebles.

Módulo Principal del Inventario de Inmuebles.

Este proceso se encargará de actualizar la información que proviene de los módulos antes mencionado; a su vez deberá cubrir los siguientes aspectos:

- Generar reportes que proporcionen información del Inventario de Inmuebles.
- Actualización de Inmuebles y datos generales. Se refiere a la actualización del inmueble por ampliaciones de la construcción o modificaciones de la misma. Asimismo, esta actividad permite la actualización de datos generales que cambian constantemente.
- Actualización consulta y reporte del histórico. Este proceso consiste en mantener un registro de los cambios que se han realizado al inmueble en cuanto a modificaciones y reparaciones.
- Consulta de datos del Inmueble. Permite consultar por pantalla la información de un inmueble seleccionado. La información se presenta a través de varias pantallas correspondientes a las secciones anteriores. Permitiendo al usuario diseñar la consulta de acuerdo a sus necesidades.
- Captura de fotografías y planos. Para este proceso se pretende utilizar una estación de trabajo y un scanner y consiste en el registro de fotografías de la fachada, así como de su plano arquitectónico para promoción. Las fotografías se deben almacenar en archivos.

2. PROGRAMACION DE MANTENIMIENTOS.

Este módulo proporciona los elementos necesarios para llevar a cabo la programación de mantenimientos preventivos y correctivos que se requieren para mantener en buen estado los inmuebles. Asimismo, este módulo permite a la empresa programar los mantenimientos para efectos de planeación.

La generación de programas de mantenimiento puede ser realizada especificando inmueble por inmueble o en forma masiva para un grupo de inmuebles definido.

Los datos contemplados en los programas de mantenimiento son:

- Partidas.
- Conceptos.
- Fechas de inicio y terminación.
- Unidad de medida.
- Duración.
- Contratista.
- Costo entre los datos más importantes.

Entre las partidas se encuentran:

- Albañilería.
- Aluminio y vidrio.
- Carpintería.
- Equipos.
- Impermeabilización.
- Instalaciones eléctricas.
- Instalaciones hidráulicas.
- Limpieza.
- Pintura.
- Varios.

Algunos conceptos de mantenimiento son:

- Aplanado.
- Boquillas.
- Limpieza de registros.
- Limpieza de tinacos.
- Ajuste de puertas.
- Pintura de muros.

Un programa de mantenimiento está conformado por los siguientes procesos:

Validar Partida.

Este módulo consiste en un conjunto de procesos que permitirán registrar, validar y actualizar las partidas, que requiere la empresa para producir sus programas de mantenimiento.

Validar Conceptos Mantenimiento.

Este módulo consiste en un conjunto de procesos que permitirán registrar, validar y actualizar los conceptos de mantenimiento, que requiere la empresa para producir sus programas de mantenimiento.

Validar Contratista.

Este módulo consiste en un conjunto de procesos que permitirán registrar, validar y actualizar los contratistas que requiere la empresa para producir sus programas de mantenimiento.

Validar Clave.

Este módulo consiste en un conjunto de procesos que permitirán registrar, validar y actualizar las claves, que requiere la empresa para producir sus programas de mantenimiento.

Cada uno de estos rubros constituye un catálogo del sistema que se alimentará y actualizará interactivamente con el usuario a través de pantallas.

Módulo Principal de Programación de Mantenimientos.

Este se encargará del control de los módulos antes mencionados y consta de los siguientes procesos:

Generar Programa de Mantenimiento Global y/o Individual.

El mantenimiento global consiste en conjuntar todas las actividades de mantenimiento pertenecientes a un mismo Tipo-Partida-Concepto de un grupo de inmuebles por contratista y el mantenimiento individual es particular de un inmueble. Dicho lo anterior el sistema deberá generar este tipo de programas.

Generación de Programas de Mantenimiento Correctivos y Preventivos.

El sistema permitirá generar individualmente los programas por tipo de mantenimiento.

Asignación de Inmuebles a Programas de Mantenimiento.

Por medio de este proceso es posible especificar los inmuebles que conformarán el Programa de Mantenimiento deseado. El sistema también permitirá asignar un grupo de inmuebles a un programa de mantenimiento, con lo cual se evita el trabajo de asignar inmueble por inmueble.

Generar reportes.

Este proceso permitirá al usuario obtener en forma impresa un Programa de Mantenimiento (correctivo o preventivo). El proceso imprimirá el(los) programa(s) que indique el usuario, sin importar si es un programa nuevo, en ejecución o concluido.Además de poder mostrar el Programa de Mantenimiento de un Inmueble en particular. El reporte podrá ser utilizado para dar aviso al cliente de los trabajos que van a ser realizados en el inmueble que renta.

3. ESTIMACION DE COSTOS Y TIEMPOS DE MANTENIMIENTO.

Este módulo permite incorporar a los Programas de Mantenimiento los Costos y Tiempos Estimados por la empresa para compararlos con los costos y tiempos ofrecidos por los contratistas. Estará *conformado* por los siguientes procesos:

Registrar Tiempos y Costos Base.

Este proceso se refiere al registro y actualización de costos y tiempos por unidad de medida que la empresa estima para calcular la duración y costo de los trabajos de mantenimiento. Dichos costos y duraciones sirven de base para compararlos con los costos y duraciones que propongan los contratistas.

Generación de Estimaciones de Costos.

Mediante este proceso se generan estimaciones de costos y duraciones de las actividades de un Programa de Mantenimiento seleccionado. Dichas estimaciones se calculan con los costos y tiempos base que determina la empresa.

En esta sección se cuenta con un reporte que muestra un Programa de Mantenimiento seleccionado con sus costos y tiempos estimados. El usuario puede pedir el reporte para cualquiera de los tipos de mantenimiento (correctivo o preventivo).

Módulo Principal de Estimación de Costos y Tiempos.

Este se encargará del control de los procesos antes mencionados, además, tomando en consideración que de un programa a otro pueden variar los costos y tiempos unitarios, se almacenarán los costos y tiempos de cada programa en el sistema. Esta facilidad nos permitirá controlar cada programa y además obtener información histórica para estudios económicos.

4. EJECUCION DE MANTENIMIENTOS Y REPARACIONES.

Este módulo permite registrar para cada programa de mantenimiento, los tiempos y costos de los trabajos que ha estimado el contratista. Asimismo, permite registrar los avances reales de los trabajos en ejecución y mostrar los avances por programa. También es posible registrar las reclamaciones al contratista por trabajos mal ejecutados y por materiales de baja calidad empleados. Abarca los procesos:

Registro de Tiempos y Costos Base.

Se refiere al registro y actualización de estimaciones de costo y duración para trabajos de mantenimiento solicitados a contratistas y también del registro y actualización de costos y tiempos reales que se van presentando durante la ejecución de los trabajos de mantenimiento. Dichos trabajos deben pertenecer a un Programa de Mantenimiento para que puedan ser almacenados. Las estimaciones de costo y tiempo de los contratistas constituyen el programa de trabajo aprobado que servirá para el control de los avances de obra, es decir, que son los costos y tiempos que serán comparados contra los costos y tiempos reales y que permitirán a la empresa tomar decisiones respecto al desempeño de los contratistas.

Generación de Estimaciones de Costos.

Con este proceso se generarán estimaciones de costos y duraciones de las actividades de un Programa de Mantenimiento. Dichas estimaciones se calculan con los costos y tiempos que determinan los contratistas y los resultados de los avances reales.

Reporte Comparativo de Tiempos.

Este proceso deberá obtener los datos que se registraron en los módulos anteriores para determinar los avances reales de las obras y calificar a los contratistas en función de las desviaciones a los programas de trabajo. Además con este reporte se obtendrá un comparativo de los avances que permitirá a la empresa tener resultados en forma gráfica, que en determinado momento es mejor para la presentación de los mismos a la gerencia.

Registrar Reclamaciones.

Este proceso permite almacenar en el sistema las reclamaciones a contratistas, registrando el Programa-Actividad-Inmueble-Contratista, el tipo de reclamación, el impacto causado (en costo y/o tiempo) y un renglón de observaciones. Esta información se utilizará para la evaluación de contratistas.

Módulo Principal de Ejecución de Mantenimientos.

Este se encargará del control de los módulos ya descritos, además de generar los reportes que permitirán a la Empresa Inmobiliaria tomar decisiones al respecto.

4.2 RECURSOS.

Como en toda fase de análisis se requiere determinar los recursos con los que cuenta y debe contar la empresa para el desarrollo e implementación del sistema. En este caso los recursos a saber son: recursos humanos, recursos de hardware y recursos de software.

Comenzaremos por mencionar los recursos humanos.

RECURSOS HUMANOS.

Se enumeran a continuación para cada una de las fases del proyecto:

1. Ingeniería de Sistemas.

- Descripción del recurso.

1. Definición del problema.	Se estima la colaboración de 2 personas.
2. Análisis de la Situación Actual.	Se estima la colaboración de 2 personas.
3. Análisis de Factibilidad.	Se estima la colaboración de 3 personas.
4. Análisis general del Sistema.	Se estima la colaboración de 4 personas.

2. Análisis del sistema.

- Descripción del recurso.

  Se estima la colaboración de 4 personas.

3. Diseño.

- Descripción del recurso.

  Se estima la colaboración de 4 personas.

4. Codificación.

- Descripción del recurso.

  Se estima la colaboración de 4 personas.

5. Pruebas.

- Descripción del recurso.

  Se estima la colaboración de 2 personas.

6. Mantenimiento.

- Descripción del recurso.

  Se estima la colaboración de 2 personas.

RECURSOS DE HARDWARE.

Debido a la variedad de tamaños y tipos de recursos de cómputo disponibles, el analista que selecciona o recomienda una fuente de hardware, software o servicios, debe contar con un método para evaluar las diferentes características de los equipos de cómputo.

En este caso la Empresa cuenta con equipo, el cual puede ser utilizado para el desarrollo e implementación del sistema.

En los recursos de hardware tenemos tres categorías:

1. Para el Sistema de Desarrollo.

Aquí se planea ocupar el siguiente equipo de cómputo durante todas las fases de desarrollo del sistema:

- Descripción del recurso.

  4 computadoras personales (Hewlett Packard) con las siguientes características:

  Procesador pentium a 75 Mhz.
  400 Mb en disco duro.
  8 Mb en RAM.
  4 equipos multimedia (tarjeta de sonido, bocinas, unidad de CD-ROM y software de multimedia).

2. La máquina objetivo.

Dado que se pretende centralizar la información en un sólo lugar para que la actualización, manejo y consulta de la misma sea efectivo, es necesario implantar una arquitectura cliente-servidor. Se presentan a continuación las características deseables del equipo sobre el que se pretende ejecutar el sistema.

- Descripción del recurso.

  1 Servidor Hewlett Packard con las siguientes características:

  Procesador pentium a 100 Mhz.
  1 Gb en Disco Duro.

32 Mb en RAM.

3. Para la operación del sistema.

- Descripción del recurso.

15 estaciones de trabajo con tarjeta de red integrada y las siguientes características:

Procesador pentium a 75 Mhz.
400 Mb en disco duro.
8 Mb en RAM.

4. Comunicaciones.

- Descripción del recurso.

Topología Ethernet.
Configuración Estrella.
Protocolo IPX/SPX.
Cableado: UTP nivel 5.

5. Otros.

- Descripción del recurso.

1 Scanner Hewlett Packard con las siguientes características:

Cama plana.
Tamaño Oficio.
Resolución: blanco y negro 600 DPI, color 300 DPI.

4 equipos multimedia (tarjeta de sonido, bocinas, unidad de CD-ROM y software de multimedia). Mismos que se utilizarán en la etapa de desarrollo.

1 Impresora Laserjet 4v con las siguientes características:

16 páginas por minuto.
Tamaño de la hoja doble carta.
Resolución 600 DPI.

El equipo con el que cuenta la Empresa Inmobiliaria cumple con las características descritas anteriormente, por lo tanto se está asegurando que se cumple con los requerimientos de procesamiento y capacidad de almacenamiento necesarios para obtener los beneficios esperados.

RECURSOS DE SOFTWARE.

Cuando una organización adquiere un software específico, se vuelve dependiente del proveedor, por lo que el proceso de seleccionar el software adecuado aumenta su importancia. Debido a ello y a las necesidades planteadas por la Empresa, se hicieron las consideraciones necesarias y se determinó que el software adecuado es el siguiente:

1. Etapa de desarrollo.

- Descripción del recurso.

  Microsoft Windows 95.
  Microsoft Visual Basic Versión 4.0.
  Microsoft Access Versión 7.0.
  Visioner Paper Port (para procesamiento de imágenes).
  Software para multimedia.

2. Implantación del sistema.

- Descripción del recurso.

  Microsoft Windows 95.
  Visioner Paper Port (para procesamiento de imágenes).
  Sistema Operativo Netware 4.1, 15 usuarios.

4.3 ESTIMACION DE COSTOS.

Actualmente se conocen diferentes métodos para la estimación de costos de un sistema, éstos ya fueron mencionados. En este caso no podemos basarnos en experiencias anteriores ya que se trata de un sistema de cómputo nuevo.

Por lo tanto, se realizaron cotizaciones del equipo de cómputo a utilizar, del software, licencias requeridas y del personal requerido en las fases de desarrollo, implementación e instalación del sistema.

A continuación, en la TABLA 4.1 se presenta un estimado de costos de los recursos humanos.

Etapa	Recursos	Días	Costo/día ($)	Total ($)
Análisis y diseño.	4	65	500.00	130,000.00
Codificación.	4	15	300.00	18,000.00
Pruebas, instalación y mantenimiento.	2	30	300.00	18,000.00
Otros.	2	110	75.00	16,500.00
Costo Total.				182,500.00

TABLA 4.1. Estimación de costos de los recursos humanos.

Los costos se estimaron de acuerdo a un diagrama de Gantt que se elaboró con tiempos para cada actividad.

En cuanto a recursos de hardware se refiere, tenemos las cotizaciones del equipo en las dos fases: para el desarrollo del sistema y para el equipo en que se pretende instalar. Se presentan en la TABLA 4.2.

Cant.	Equipo	Costo Unitario	Total ($)
	ETAPA DE DESARROLLO		
4	Computadoras Personales.	14,792.00	59,168.00
	ETAPA DE INSTALACION Y LIBERACION		
1	Servidor HP.	60,000.00	60,000.00
15	Estaciones de trabajo.	15,300.00	229,500.00
1	Instalación del servidor y dejar a punto.	2,000.00	2,000.00
1	Scanner con software.	5,000.00	5,000.00
4	Equipo Multimedia con software.	1,292.00	5,168.00
1	Impresora.	25,000.00	25,000.00
	Costo Total.		385,836.00

TABLA 4.2. Costos estimados de los recursos de hardware.

Respecto a los recursos de software tenemos que considerar el que se utilizará en la etapa de desarrollo y en la implantación de sistema, contemplando las licencias necesarias, para ello se tienen las cotizaciones que se presentan en la TABLA 4.3.

Etapa	Recursos	Total ($)
Desarrollo.	MS Windows 95.	1,100.00
	MS Visual Basic V.4.0 (4 licencias).	14,372.00
	MS. Access V.7.0 (1 licencia).	468.00
Implantación.	S.O. Novell Netware 4.1 para 15 usuarios.	25,000.00
	Costo Total.	40,940.00

TABLA 4.3 Costos estimados de los recursos de software.

Todos los costos de los tres tipos de recursos son estimados. Para calcular el costo del proyecto, sólo se tomaron en cuenta los recursos en los que invertiría la empresa, haciendo las siguientes aclaraciones:

Para los recursos humanos:

La Empresa Inmobiliaria no dispone de dichos recursos para realizar el proyecto SIAI, por lo tanto, se toma en cuenta el costo total de los recursos humanos.

Para los recursos de hardware:

La Empresa Inmobiliaria ya cuenta con todo el hardware requerido para la realización de este proyecto, por lo tanto, el costo es nulo.

Para los recursos de software:

La Empresa Inmobiliaria solo invertirá en tres licencias del compilador Visual Basic 4.0, el resto del software requerido se tiene disponible.

En resumen, el costo total de proyecto se presenta en la TABLA 4.4.

Recursos	Costo Total ($)
Humanos.	182,500.00
Hardware.	0.00
Software.	10,779.00
Total.	193,279.00

TABLA 4.4 Costo Total de los recursos para el proyecto SIAI.

4.4 HERRAMIENTAS DE CONTROL DE AVANCE.

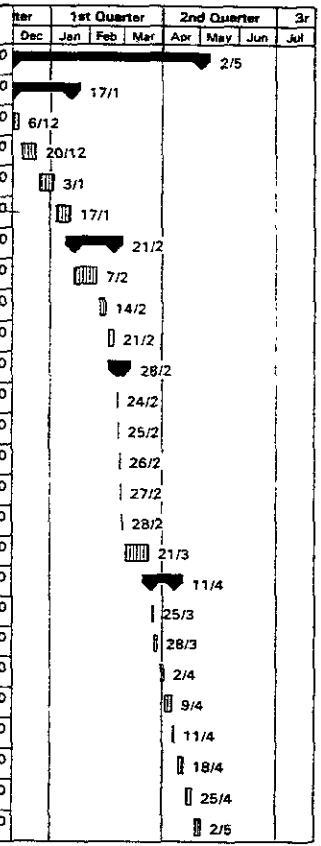
En la realización del sistema es necesario elaborar un plan de trabajo, dicho plan debe describir los mecanismos que se ocuparán para lograr las metas y requisitos. Por ejemplo, el objetivo de entregar los productos finales a tiempo se puede expresar en términos de alcanzar logros importantes del proyecto a tiempo. Un logro es un hecho significativo en el ciclo de vida del desarrollo del producto de programación; ejemplos de logros son terminación del análisis de requisitos, terminación del diseño e integración y prueba con éxito de los componentes del sistema.

Pues bien, para la planeación del Sistema Integral de Administración Inmobiliaria (SIAI) es necesario contemplar todas las actividades que se llevarán a cabo y el tiempo de desarrollo de cada una, desde la fase de definición del problema, recopilación de la información, análisis, diseño, codificación, pruebas, instalación. capacitación, etc.

Para lograr las metas se elaboró un plan de trabajo, utilizando las herramientas disponibles y descritas anteriormente.

A continuación, se presentan en la FIG. 4.1 el Diagrama de GANTT y en la FIG. 4.2 el PERT para la planeación del desarrollo del Sistema Integral de Administración Inmobiliaria, en el que se desglosan cada una de las actividades a desarrollar con tiempos fijos, aunque con holgura para evitar traslape y retraso en la terminación del proyecto SIAI.

#	Actividad	Duración	Fecha Inicio	Fecha Término
1	Proyecto SIAI	110d	2/12/96 8:00	2/5/97 17:00
2	Ingeniería de Sistemas	35d	2/12/96 8:00	17/1/97 17:00
3	Definición del Problema	5d	2/12/96 8:00	6/12/96 17:00
4	Análisis de la Situación Actual	10d	9/12/96 8:00	20/12/96 17:00
5	Análisis de Factibilidad	10d	23/12/96 8:00	3/1/97 17:00
6	Análisis del Sistema (Opciones de Solución)	10d	6/1/97 8:00	17/1/97 17:00
7	Análisis	25d	20/1/97 8:00	21/2/97 17:00
8	Recopilación de Información	15d	20/1/97 8:00	7/2/97 17:00
9	Clasificación de la Información	5d	10/2/97 8:00	14/2/97 17:00
10	Determinación de los req. del usuario	5d	17/2/97 8:00	21/2/97 17:00
11	Diseño	5d	24/2/97 8:00	28/2/97 17:00
12	Elaboración de Diagramas de Flujo de Datos	1d	24/2/97 8:00	24/2/97 17:00
13	Elaboración de Diccionario de Datos	1d	25/2/97 8:00	25/2/97 17:00
14	Elaboración de Diagramas de Entidad-Relación	1d	26/2/97 8:00	26/2/97 17:00
15	Elaboración de Minespecificaciones	1d	27/2/97 8:00	27/2/97 17:00
16	Elaboración de Carta de Estructura	1d	28/2/97 8:00	28/2/97 17:00
17	Codificación	15d	3/3/97 8:00	21/3/97 17:00
18	Pruebas	15d	24/3/97 8:00	11/4/97 17:00
19	De Unidad	2d	24/3/97 8:00	25/3/97 17:00
20	De Integración	3d	26/3/97 8:00	28/3/97 17:00
21	De Validación	3d	31/3/97 8:00	2/4/97 17:00
22	De Volumen	5d	3/4/97 8:00	9/4/97 17:00
23	Prueba General del Sistema	2d	10/4/97 8:00	11/4/97 17:00
24	Instalación	5d	14/4/97 8:00	18/4/97 17:00
25	Capacitación	5d	21/4/97 8:00	25/4/97 17:00
26	Mantenimiento	5d	28/4/97 8:00	2/5/97 17:00



Critical Progress Summary

Noncritical Milestone ◆ Rolled Up ◇

FIG. 4.1. Diagrama de GANTT del proyecto SIAI

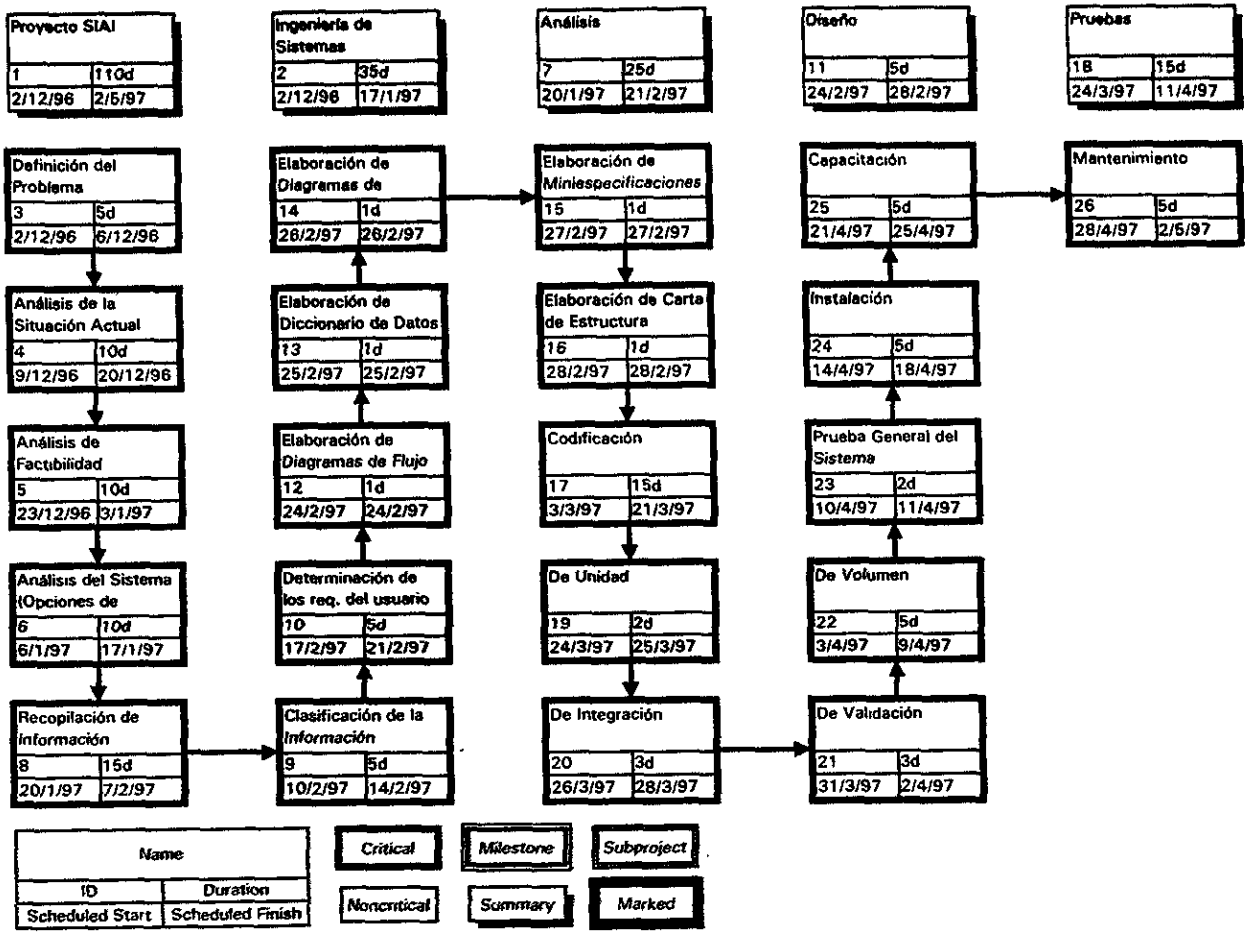


FIG. 4.2. PERT del proyecto SIAI.

CAPITULO 5

Análisis y Especificación Estructurada del Sistema

5. ANALISIS Y ESPECIFICACION ESTRUCTURADA DEL SISTEMA

Una de las etapas más importantes en el desarrollo de sistemas de información es sin duda el análisis, ya que es donde se realiza la conversión del procedimiento manual "hablado" a una representación gráfica, compacta y fácil de entender. Esta representación brinda un panorama de los procesos y flujo de información que se deben considerar para resolver la necesidad del usuario.

El propósito de este capítulo es realizar el análisis de los requerimientos y procedimientos actuales del usuario y áreas involucradas en la operatividad de la Empresa Inmobiliaria.

Se utilizarán las herramientas del análisis estructurado, como son: diagramas de flujo de datos, diagrama entidad-relación, diccionario de datos y miniespecificaciones.

Para realizar el análisis de los requerimientos del usuario, lo más conveniente es colocar en secuencia los procedimientos que se requieran para la utilización del sistema. Para ello, es necesario emplear una herramienta gráfica de modelado de sistemas, en este caso se plasmarán en diagramas de flujo de datos, los procesos que se requieren para llevar a cabo una tarea específica.

Debido a la información proporcionada por los usuarios y áreas participantes en la Empresa Inmobiliaria y como producto del análisis se generaron 70 DFD. Por cuestiones de seguridad y tratando de ser concretos, se presentan los DFD's para los procedimientos más representativos, siempre dentro de una metodología para lograr su propósito.

Se proponen dos procesos globales, en los cuales se concentran las actividades principales de operación de la empresa. Dichos procesos se pueden identificar como:

1. Inventario de Inmuebles.
2. Programación de Mantenimientos.

Estos procesos estarán constituidos por subprocesos de manera tal que permitan realizar una serie de tareas para un fin común.

Los objetivos que persiguen cada uno de estos procesos son los siguientes:

1. Inventario de Inmuebles.

Este proceso tendrá como función principal registrar los datos generales del inmueble, sus aspectos legales, sus componentes, los gastos por concepto de reparaciones, impuestos y servicios, además de mostrar fotografías de la fachada y de los planos y también tiene la capacidad de mantener actualizado el inventario.

2. Programación de Mantenimientos.

La función de este proceso es proporcionar los elementos necesarios para llevar a cabo la programación de mantenimientos que se requieran para conservar en buen estado los inmuebles. Incluyendo para ello, el registro y actualización de las partidas y los conceptos de mantenimiento, así como asignar los inmuebles a los programas.

5.1 EL DIAGRAMA DE FLUJO DE DATOS (DFD).

El Diagrama de Flujo de Datos es una herramienta gráfica que nos describe el flujo de la información y cómo se transforma de un proceso a otro, en este caso serán representados a continuación los procesos antes mencionados.

Para realizar los diagramas de flujo se utilizó el paquete Microsoft PowerPoint V.4.0, el cual tiene funciones que cumplen perfectamente con el propósito para dibujar este tipo de diagramas, además tiene la ventaja de ejecutarse bajo Microsoft Windows, que nos muestra un ambiente gráfico y amigable al usuario.

La simbología, lectura y reglas para la construcción de un DFD ya fueron detallados, por lo tanto, nos concretaremos a explicar brevemente el significado de algunos de ellos y no su interpretación de acuerdo a su representación gráfica.

Primero dibujaremos el DFD que contempla los dos puntos con los flujos de datos involucrados en el procedimiento general. Después dibujaremos el DFD de los procesos principales de cada uno de los módulos anteriores, seguiremos con los DFD del siguiente nivel y así sucesivamente, hasta llegar a los DFD terminales. A los diagramas de más alto nivel se le asignará un número de secuencia, los diagramas del siguiente nivel tendrán un número consecutivo con el prefijo del módulo al que pertenecen seguidos de un dígito con el siguiente dígito consecutivo. Esta regla se seguirá para los niveles inferiores subsecuentes.

Sin embargo, como se puede observar en el diagrama de la FIG. 5.1, aparecen tres módulos más de los ya mencionados. La razón principal de este hecho es que aparece el módulo principal para el manejo de todo el sistema y los otros dos son necesarios para el manejo de la información del inventario y de los programas de mantenimiento.

Por lo tanto estos módulos vienen a complementar a los dos primeros y de esta manera se obtiene el total del sistema.

DFD. Nivel 1.

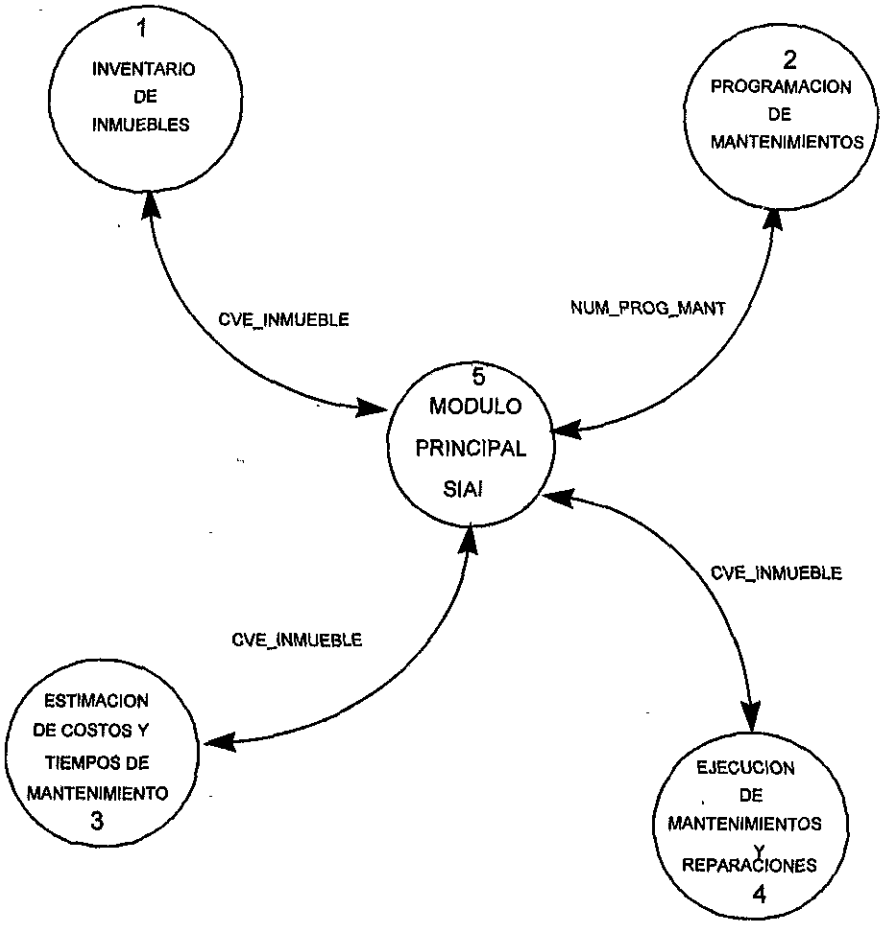


FIG. 5.1. DFD Nivel 1 Sistema Integral de Administración Inmobiliaria.

1. Inventario de inmuebles.

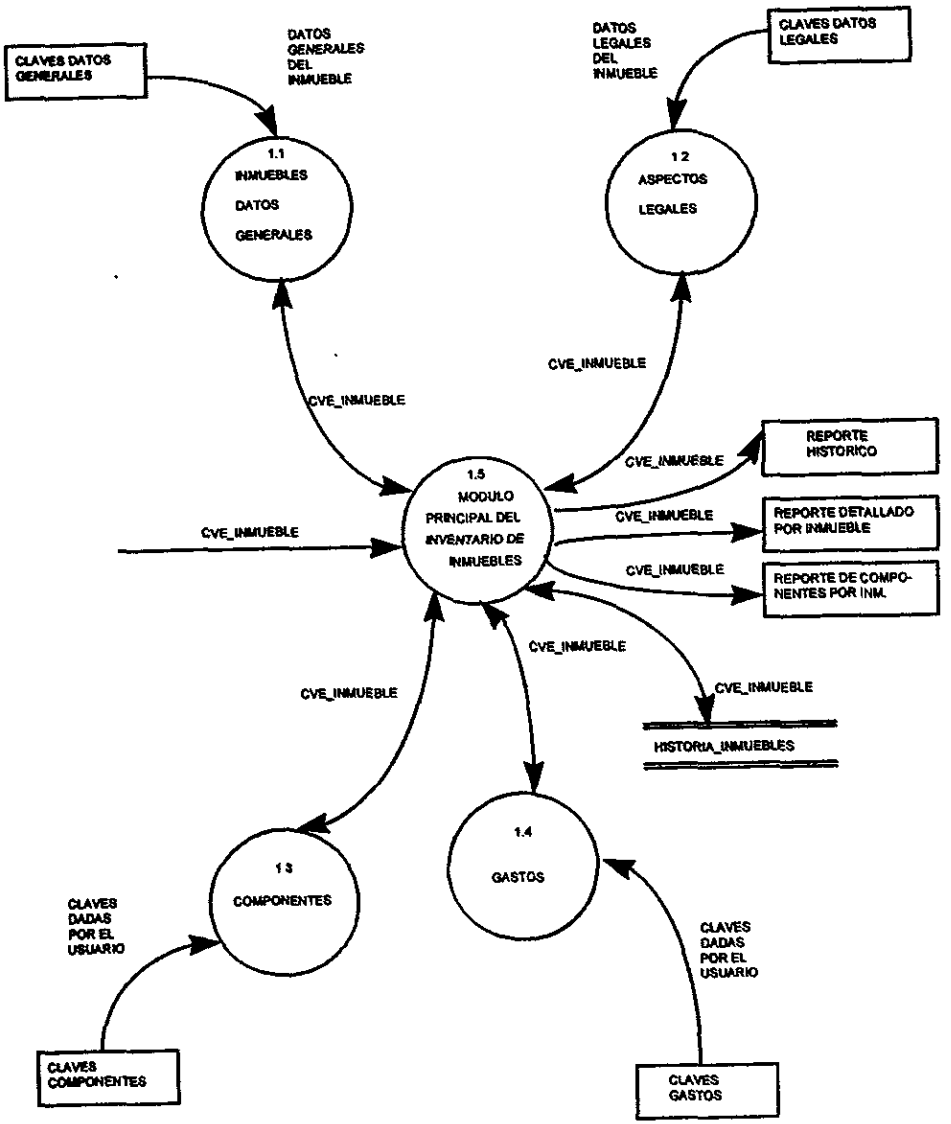


FIG. 5.2. DFD Nivel 2 Inventario de Inmuebles.

Como podemos ver en la FIG. 5.2, el módulo Inventario de Inmuebles consta de cinco procesos principales:

1. Inventario de Inmuebles.
    1.1 Inmuebles Datos Generales.
    1.2 Aspectos Legales.
    1.3 Componentes.
    1.4 Gastos.
    1.5 Módulo Principal del Inventario de Inmuebles.

En el diagrama se muestran los datos de entrada y salida, así como las entidades externas que interactúan con los procesos.

La representación de este proceso no termina aquí, los procesos pueden contener subprocesos y éstos a su vez otros, y así sucesivamente hasta llegar a los terminales.

Los subprocesos correspondientes a cada uno de los anteriores procesos se describen a continuación:

1.1 Inmuebles Datos Generales.
    1.1.1. Validaciones Datos Generales.
        1.1.1.1 Registrar Clave Tipo de Inmueble.
        1.1.1.2 Actualizar Cve_Tipo.
        1.1.1.3 Registrar Clave Ciudad.
        1.1.1.4 Actualizar Cve_Ciudad.
        1.1.1.5 Registrar Clave de Clasificación.
        1.1.1.6 Actualizar Cve_Clasif.
        1.1.1.7 Registrar claves de Archivo_Foto Archivo_Plano.
        1.1.1.8 Actualizar Archivo_Foto Archivo_Plano.
        1.1.1.9 Registrar Clave Estado.
        1.1.1.10 Actualizar Cve_Estado.
        1.1.1.11 Registrar Clave de Localidad.
        1.1.1.12 Actualizar Cve_Localidad.
1.2 Aspectos Legales.
    1.2.1 Validaciones Aspectos Legales.
        1.2.1.1 Registrar Clave de Inmueble.
1.3 Componentes.
    1.3.1 Validaciones y Clasificación de Componentes.
        1.3.1.1 Registrar Clave de Componente Detalle.
        1.3.1.2 Actualizar Cve_Comp_Det.
        1.3.1.3 Registrar Clave de Componente General.
        1.3.1.4 Actualizar Cve_Comp_Gen.
        1.3.1.5 Registrar Clave de Inmueble.
        1.3.1.6 Actualizar Cve_Inmueble.

1.4 Gastos.
1.4.1 Validaciones Gastos .
1.4.1.1 Registrar Clave de Gasto.
1.4.1.2 Actualizar Cve_Gasto Cve_Inmueble.
1.5 Módulo Principal del Inventario de Inmuebles.
1.5.1 Llamar Datos.
1.5.2 Generar Reportes.
1.5.3 Actualización de Inmuebles y Datos Generales.
1.5.4 Actualización Consulta y Reporte del Histórico.
1.5.5 Consulta de Datos del Inmueble.
1.5.6 Capturar Planos y Fotografías.

Un aspecto importante es que las entradas y salidas de una burbuja, coinciden con las de la burbuja que le corresponden de nivel inmediato anterior siguiendo con este criterio hasta llegar al nivel 1.

De acuerdo con la simbología, las entidades externas que participan en este procedimiento son: Claves Datos Generales, Claves Datos Legales, Claves Componentes, Claves Gastos, Reporte Histórico, Reporte Detallado por Inmueble y Reporte de Componentes por Inmueble.

Ahora se presentan los DFD's derivados de cada uno de los subprocesos de los procesos principales del módulo 1.

1. Inventario de inmuebles.
    1.1 Inmuebles datos generales.
        1.1.1 Validaciones datos generales.

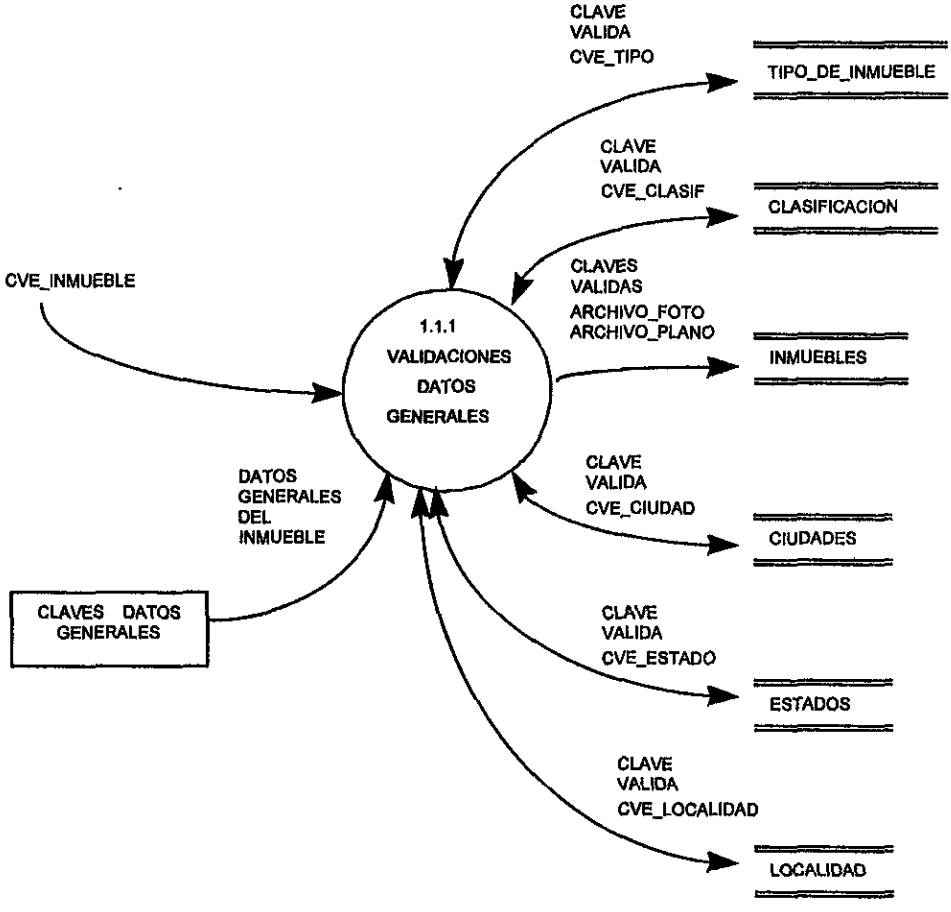


FIG. 5.3. DFD nivel 3 Inmuebles Datos Generales.

Se presenta a continuación el DFD del proceso 1.2 Aspectos Legales que pertenece al Inventario de Inmuebles.

1. Inventario de inmuebles.
   1.2 Aspectos legales.
      1.2.1 Validaciones aspectos legales.

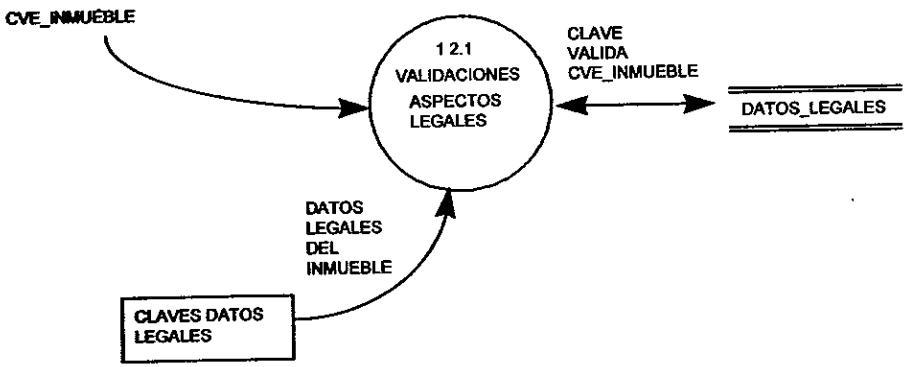


FIG. 5.4. DFD nivel 3 Aspectos Legales.

A continuación se presenta la FIG. 5.5 que corresponde al subproceso:
1.3 Componentes.

1. Inventario de inmuebles.
    1.3 Componentes.
        1.3.1 Validaciones y clasificación de componentes.

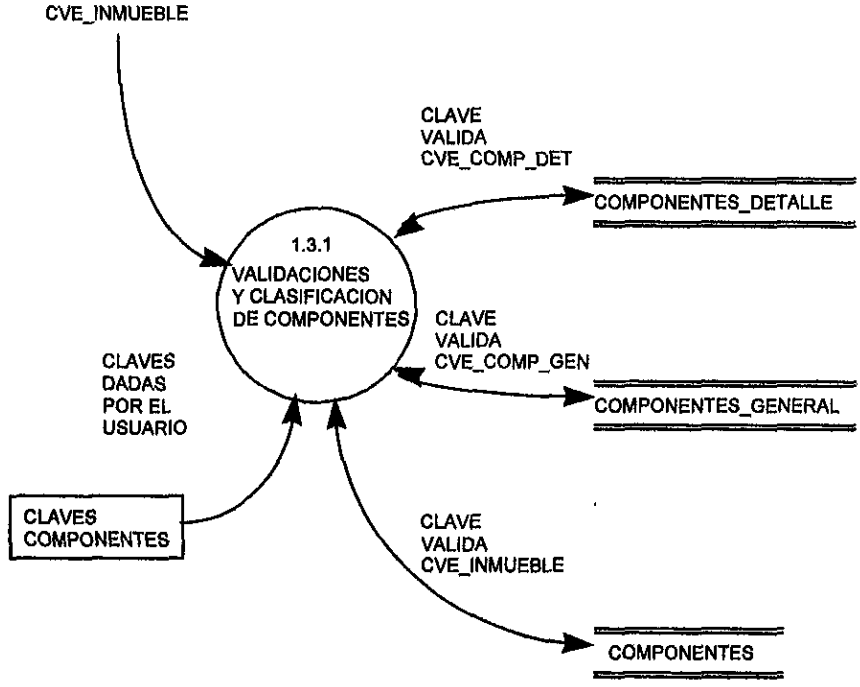


FIG. 5.5. DFD nivel 3 Componentes.

A continuación se presenta la FIG. 5.6 que corresponde al subproceso 1.4 Gastos.

1. Inventario de inmuebles.
   1.4 Gastos.
      1.4.1 Validaciones gastos.

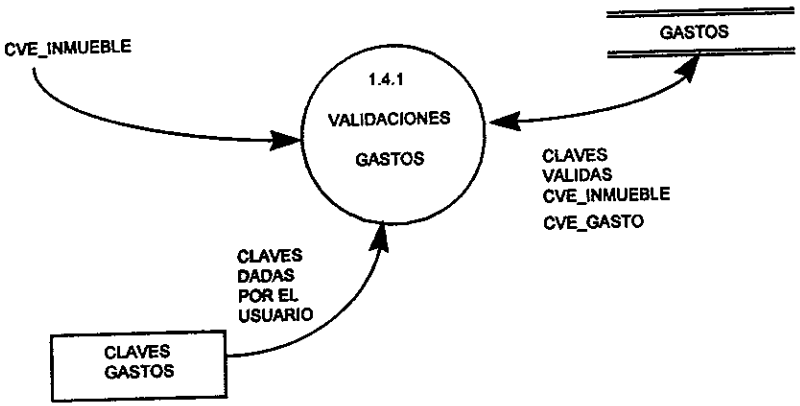


FIG. 5.6. DFD nivel 3 Gastos.

A continuación se presenta la FIG. 5.7 que corresponde al subproceso 1.5 Módulo Principal del Inventario de Inmuebles.

1. Inventario de inmuebles.
   1.5 Módulo principal del inventario de inmuebles.

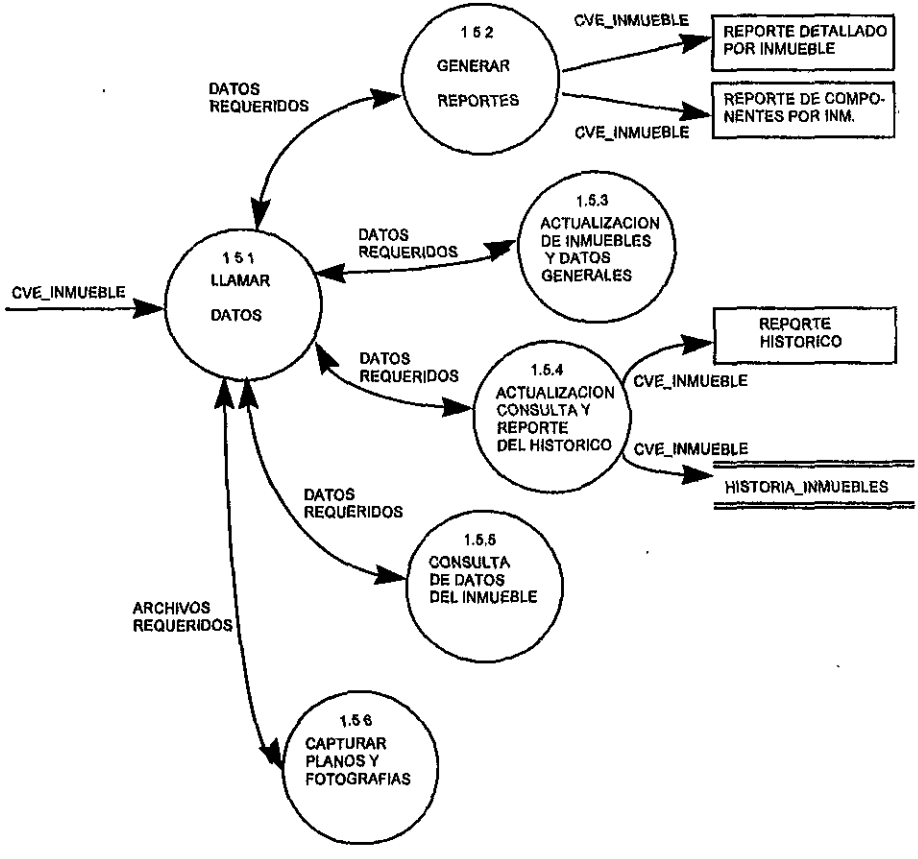


FIG. 5.7. DFD nivel 3 Módulo Principal del Inventario de Inmuebles.

El Módulo 2, Programación de Mantenimientos, consiste principalmente en lo siguiente: una vez que se tienen las claves de las partidas, los conceptos de mantenimiento y los contratistas, se procede a realizar una serie de validaciones, para posteriormente generar la programación de los mantenimientos y esto se encuentra en los siguientes subprocesos:

2. Programación de Mantenimientos.
    2.1 Validar Partida.
    2.2 Validar Conceptos de Mantenimiento.
    2.3 Validar Contratista.
    2.4 Valida Clave.
    2.5 Módulo Principal de Programación de Mantenimientos.

De la misma forma que en el módulo Inventario de Inmuebles, los subprocesos del módulo Programación de Mantenimientos son los siguientes:

2.1 Validar Partida.
    2.1.1 Registrar Partida.
    2.1.2 Actualizar Partida.
2.2 Validar Conceptos de Mantenimiento.
    2.2.1 Registrar Concepto de Mantenimiento.
    2.2.2 Actualizar Concepto de Mantenimiento.
2.3 Validar Contratista.
    2.3.1 Registrar Contratista.
    2.3.2 Actualizar Contratista.
2.4 Valida Clave.
    2.4.1 Registrar Clave del Programa de Mantenimiento.
    2.4.2 Actualizar Clave.
2.5 Módulo Principal de Programación de Mantenimientos.
    2.5.1 Llamar Claves.
    2.5.2 Generar Programa de Mantenimiento Global y/o Individual.
    2.5.3 Generación de Programas Permanentes, Correctivos y Preventivos.
    2.5.4 Asignación de Inmuebles a Programas de Mantenimiento.
    2.5.5 Generar Reportes.

2. Programación de mantenimientos.

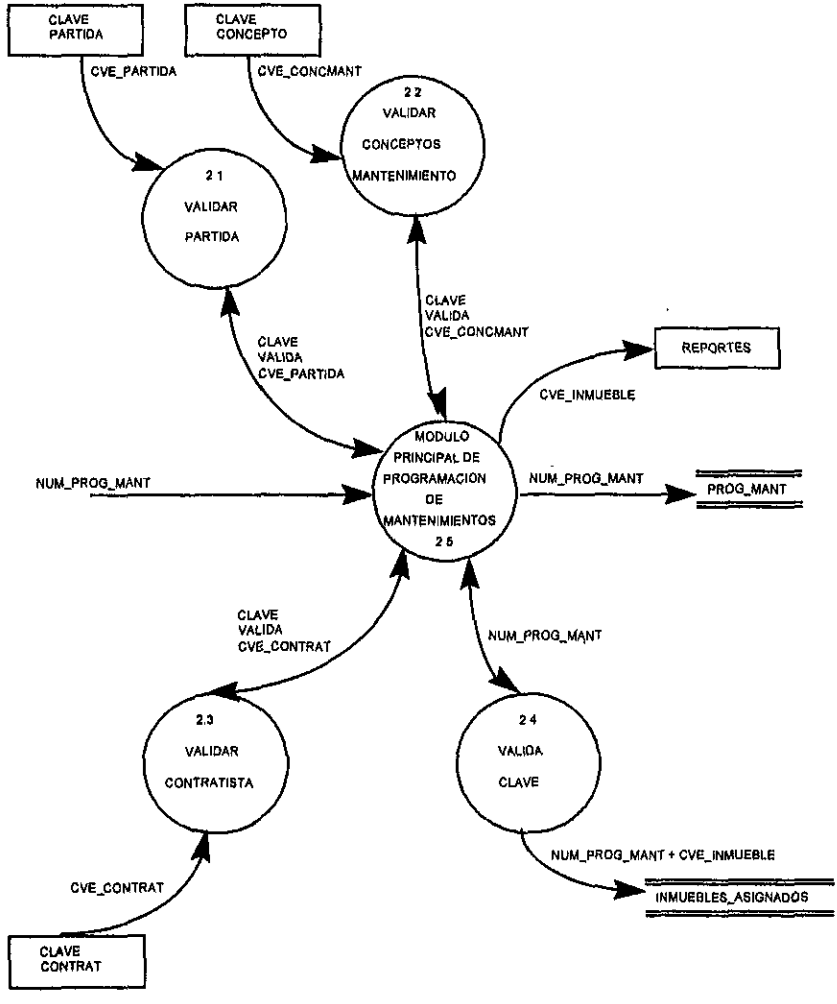


FIG. 5.8. DFD nivel 2 Programación de Mantenimientos.

2. Programación de mantenimientos.
   2.1 Validar partida.

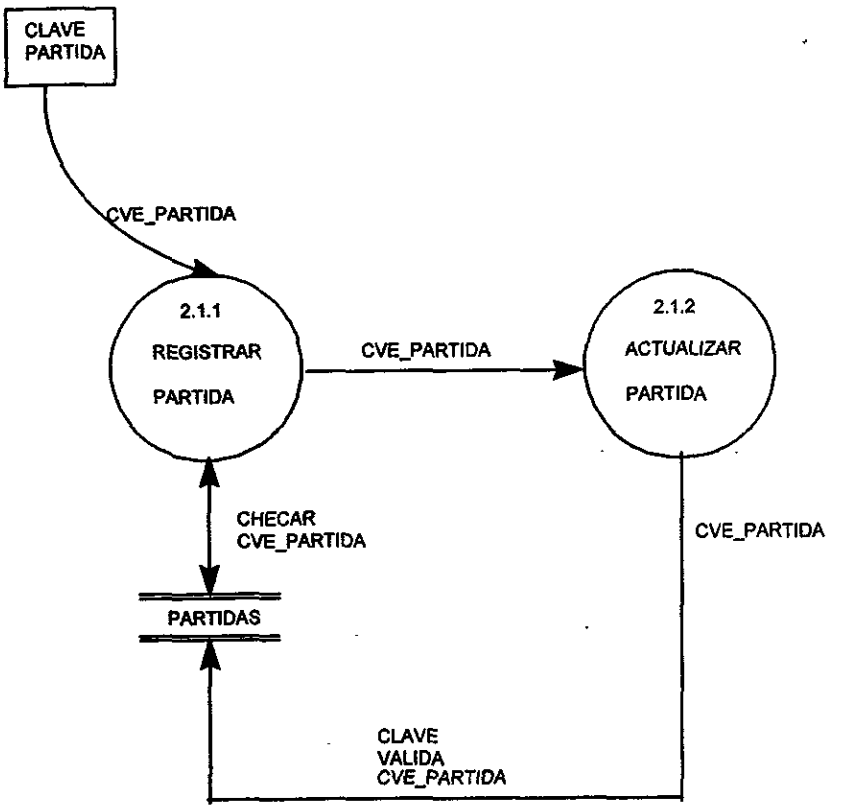


FIG. 5.9. DFD nivel 3 Validar Partida.

2. Programación de mantenimientos.
2.2 Validar conceptos de mantenimiento.

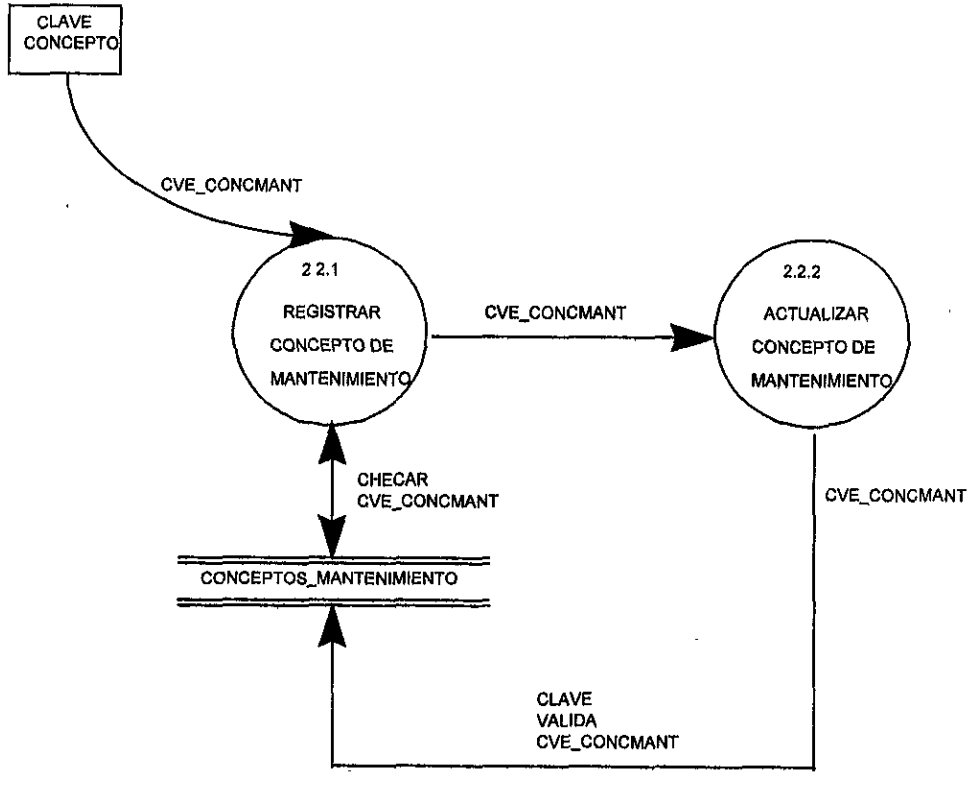


FIG 5.10. DFD nivel 3 Validar Conceptos de Mantenimiento.

2. Programación de mantenimientos.
   2.3 Validar contratista.

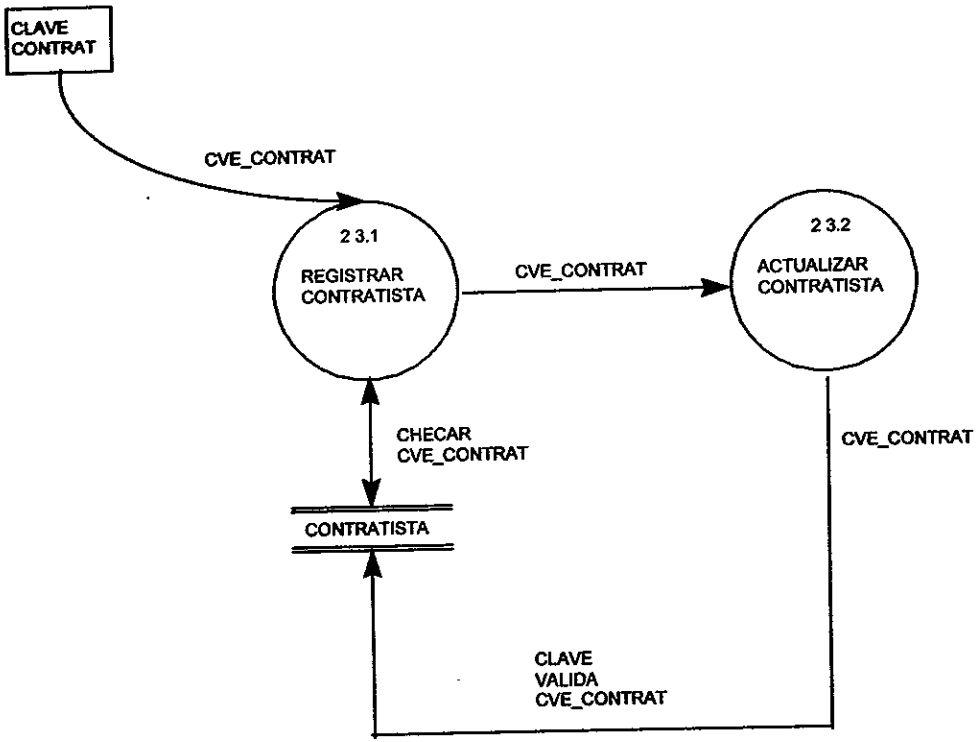


FIG. 5.11. DFD nivel 3 Validar Contratista.

2. Programación de mantenimientos.
2.4 Valida clave.

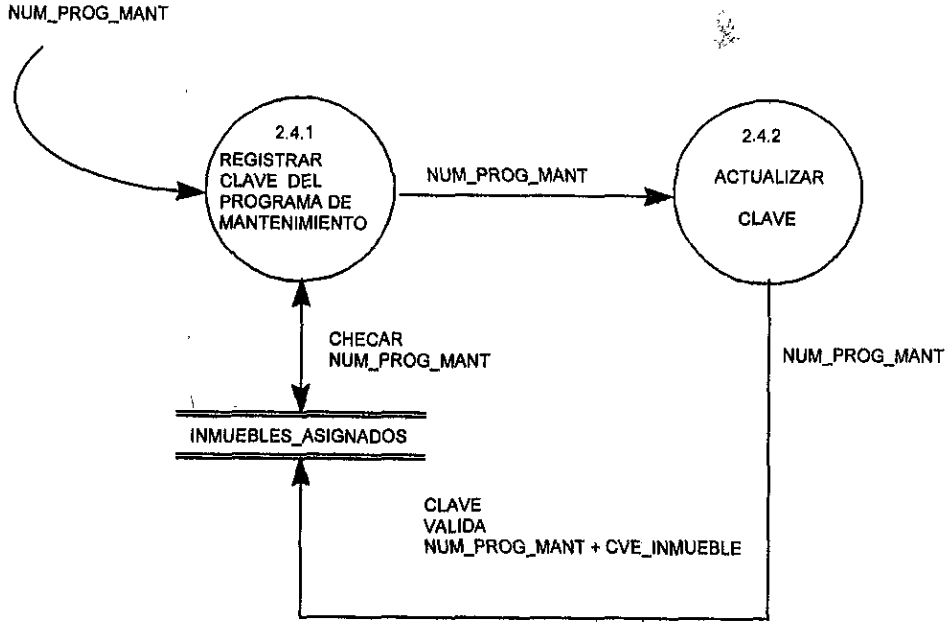


FIG. 5.12. DFD nivel 3 Valida Clave.

2. Programación de mantenimientos.

2.5 Módulo principal de programación de mantenimientos.

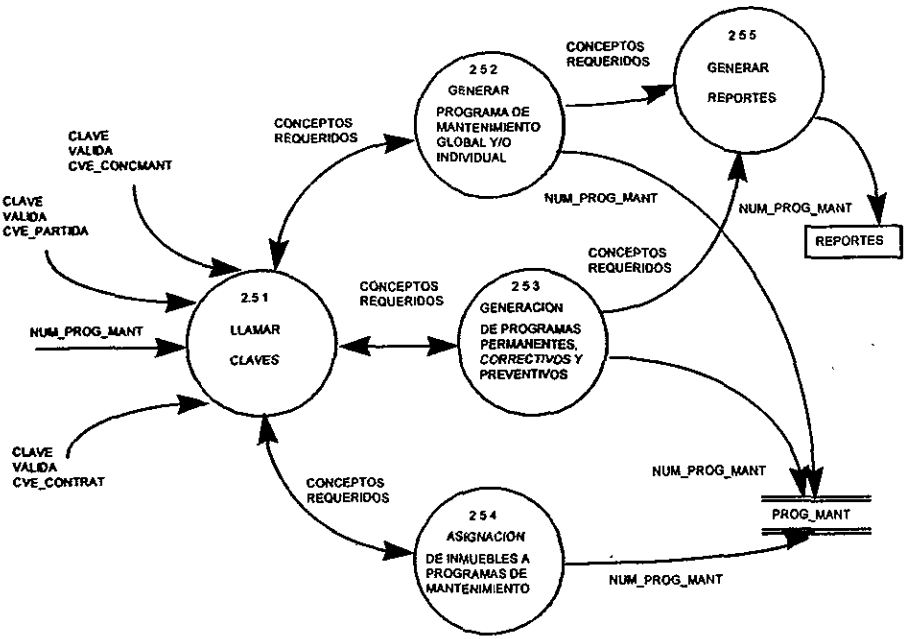


FIG. 5.13. DFD nivel 3 Módulo Principal de Programación de Mantenimientos.

5.2 EL DICCIONARIO DE DATOS.

El diccionario de datos es otra herramienta del análisis, donde se elabora un listado que contiene todos los datos acerca del sistema, se describen las entradas, salidas y lo que hay en cada tabla.

Como se explicó anteriormente, es recomendable utilizar un paquete que permita la construcción del diccionario de datos con la finalidad de emplear todos los recursos que ayuden a desarrollar el diccionario. Por lo tanto, se utilizó el manejador de base de datos Microsoft Access 7.0 para desarrollarlo. El diccionario de datos del SIAI se presenta en la parte correspondiente a los anexos del presente trabajo.

5.3 EL DIAGRAMA DE ENTIDAD-RELACION.

El siguiente paso en el diseño estructurado del sistema es el diagrama entidad-relación, el cual se elabora cuando se han cubierto las etapas anteriores. Para elaborar este diagrama se optó por utilizar un paquete de software especializado, con el fin de obtener de manera automática el diagrama entidad-relación ligado directamente a la base de datos del sistema.

Dado que en la actualidad es muy recomendable utilizar la tecnología informática existente, como una herramienta más para solucionar problemas y mejorar muchas actividades de cualquier área además de organizar y controlar información y documentos, optamos por utilizar un paquete de software para construir el diagrama entidad-relación, no sin antes asegurarnos de que cumpliera con los propósitos establecidos en la referencia teórica de la presente tesis y de señalar la ventajas que esto representa como son:

a) El diagrama puede obtenerse de manera automática en cualquier momento.
b) El diagrama puede ir actualizándose en relación a la base de datos conforme el sistema crezca.

Por lo anterior y además de que el software está disponible en la empresa, es decir, no tuvo que adquirirse, el diagrama de entidad-relación del Sistema Integral de Administración Inmobiliaria (SIAI) se construyó con el paquete ERWIN (entidad-relación para Windows) versión 2.6 y es el que se presenta en la FIG. 5.14.

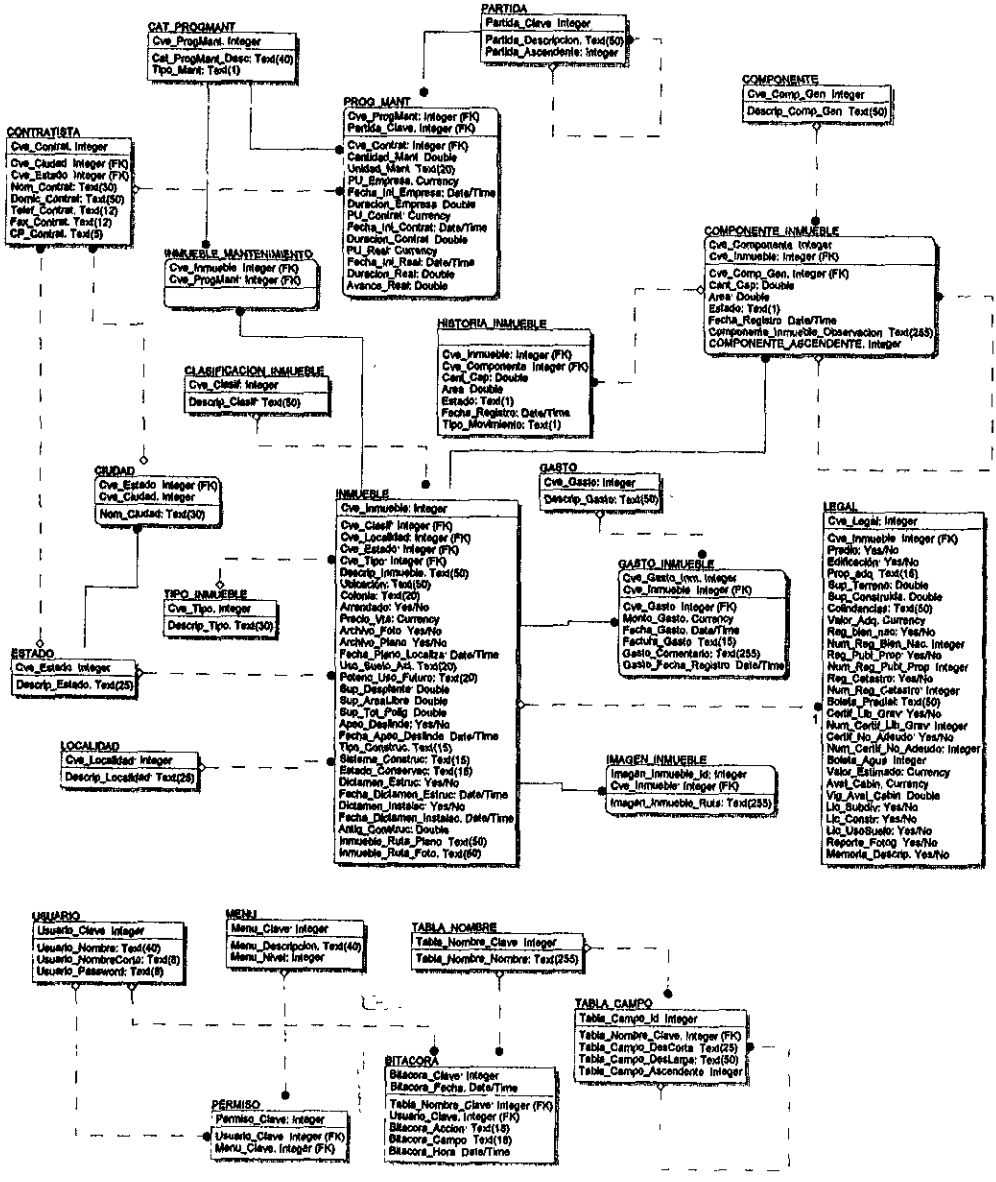


FIG. 5.14. Diagrama entidad-relación del SIAI.

Como puede apreciarse, el diagrama entidad-relación del SIAI cumple con las funciones primordiales de describir la distribución de los datos almacenados en el sistema y representar las entidades y sus relaciones. Aquí se puede distinguir cuáles son las principales entidades con las que el sistema no podría funcionar sin ellas. Tal es el caso de las entidades: INMUEBLES, PROGRAMAS DE MANTENIMIENTO, CONTRATISTAS, etc., por mencionar algunas de ellas. También se pueden apreciar las entidades, que por sus relaciones, son menos importantes en el sistema pero necesarias para el adecuado funcionamiento y máximo aprovechamiento de la información que resulta del sistema.

Cabe mencionar que se integran como entidades: USUARIOS, GRUPOS, NOMBRES DE LAS TABLAS, MENUS, PERMISOS DE GRUPO y BITACORAS, las cuales no forman parte de los procesos de la empresa, pero sí del sistema, ya que son con las que trabaja el módulo de seguridad del sistema, mismo que se implementó a petición de los usuarios y es un requerimiento más del sistema.

El diagrama cumple con los cuatro componentes de un diagrama entidad-relación:

1. Tipos de objetos.

Existen entidades representadas en el diagrama las cuales representan objetos del mundo real.

2. Relaciones.

También pueden apreciarse todas las relaciones existentes que hay entre los datos del sistema; éstas se hacen a través de llaves primarias o foráneas.

3. Indicadores asociativos de tipo objeto.

Estos son los que representan algo que funciona como entidad (sin llegar a serlo) y como relación a la vez y uno de estos indicadores dentro del sistema es el siguiente:

La relación asociada entre las entidades INMUEBLES y PROGRAMAS DE MANTENIMIENTO son el(los) inmueble(s) asignados a un programa de mantenimiento, el cual sin la asignación de un contratista no puede llevarse a cabo. En este ejemplo la relación antes mencionada puede considerarse como un indicador asociativo de tipo objeto.

4. Indicadores de subtipo/supertipo.

Este componente del diagrama entidad-relación puede ejemplificarse con la entidad inmuebles porque dentro de dicha entidad existe una clasificación importante que es el tipo de inmueble. Por lo tanto puede considerarse como indicador de tipo supertipo a la entidad inmuebles y como indicadores de tipo subtipo inmuebles propios, inmuebles administrados e inmuebles consignados para su venta y esto lo confirma el hecho de

que en algún momento dado puede darse un trato diferente a cada tipo de inmueble dentro del sistema.

5.4 MINIESPECIFICACIONES.

Las miniespecificaciones describen lo que sucede en cada burbuja del diagrama de flujo de datos. Estas se escribieron en Español estructurado, siguiendo con la metodología propuesta.

Se desarrollaron las miniespecificaciones del módulo Inventario de Inmuebles, esto involucra al mismo tiempo la descripción de la entrada al sistema y el llamado desde el menú principal al módulo de Inventario, que se estructura en: inmuebles (datos generales) e información legal (datos o aspectos legales). Por lo tanto la descripción de las miniespecificaciones contempla las siguientes formas:

- fpassw.
- finicio.
- fprincipal.
- fDatosGen.
- fDatLeg.

Forma fpassw.

Forma que pide el password de entrada al sistema.

PROCEDIMIENTO cAceptar_Click
  DESCARGAR todas las formas contenidas actualmente en memoria
  MOSTRAR la forma de Inicio para captura de password
  Liberar la memoria para poder cargar sin problema el sistema
  DESCARGAR la forma de Inicio
  MOSTRAR el menú Principal
FIN DE PROCEDIMIENTO.

PROCEDIMIENTO cCancelar_Click
  DESCARGAR los formatos en memoria
  SALIR del sistema
FIN DE PROCEDIMIENTO.

PROCEDIMIENTO cTerminar_Click
  DESCARGAR los formatos en memoria
  SALIR del sistema
FIN DE PROCEDIMIENTO.

Forma finicio.

Forma que carga una pantalla de presentación del sistema.

```
PROCEDIMIENTO Form_Load
   Salto y retorno de carro
   ASIGNAR gsNewLine = Chr(13) & Chr(10)
FIN DE PROCEDIMIENTO.
```

Forma fprincipal.

Forma que carga el menú principal del sistema.

```
PROCEDIMIENTO ShowTip
   HACER CASO intbtnselected
   CASO 1
      ASIGNAR atributos para botón de Documento Nuevo
      lblToolTip.Top = imgFileNewButton.Top * 2
      lblToolTip.Left = imgFileNewButton.Left + 500
      lblToolTip.Caption = "Programa de Mantenimientos"
      lblToolTip.Visible = True
   CASO 2
      ASIGNAR atributos para botón de Abrir Documento
      lblToolTip.Top = imgFileOpenButton.Top * 2
      lblToolTip.Left = imgFileOpenButton.Left + 500
      lblToolTip.Caption = "Mi opción dos"
      lblToolTip.Visible = True
   FIN CASO
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO imgFileNewButton_Click
   RESTAURA con datos de inicialización, el botón de abrir nuevo archivo
   MOSTRAR la forma fasigna
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO imgFileOpenButton_Click
   RESTAURA con datos de inicialización, el botón de abrir archivo
FIN DE PROCEDIMIENTO.
```

```
PROCEDIMIENTO mnuConfig_Click
   LLAMAR al procedimiento de configuración de impresora en el control de dialogo
   común.
   CMDialog1.Flags = &H40  ' Sólo el dialogo de configuración de impresora.
   MOSTRAR la forma de configuración de impresora
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO mnuEjecucion_Click
   MOSTRAR la forma frmEjecuta
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO mnuGenerador_Click
   MOSTRAR la forma frmQuery
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO mnuInmuebles_Click
   MOSTRAR la forma fDatosGen
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO mnuLegal_Click
   MOSTRAR la forma DatLeg
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO mnuProgMantenimiento_Click
   MOSTRAR la forma fProgramación
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO mnuProgramacion_Click
   MOSTRAR la forma fasigna
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO mnuSalir_Click
   SALIR del sistema
FIN DE PROCEDIMIENTO.
```

```
PROCEDIMIENTO picToolbar_MouseMove
   INICIALIZAR atributos del temporizador
   lblToolTip.Visible = False
   Timer1.Enabled = False
FIN DE PROCEDIMIENTO.
```

Forma FdatosGen.

Forma que carga la pantalla para el manejo de los datos generales del sistema.

```
PROCEDIMIENTO Ctrl2Reg
   ASIGNAR a registro de paso los atributos de texto actuales
   r_Inmueble.Descrip_Inmueble = txtDescrip
   r_Inmueble.Ubicación = txtubica
   r_Inmueble.Colonia = txtcol
   r_Inmueble.Arrendado = txtarrenda
   r_Inmueble.Uso_Suelo_Act = txtsueloact
   r_Inmueble.Potenc_Uso_Futuro = txtusofut
   r_Inmueble.Sup_Desplante = txtsupdesplante
   r_Inmueble.Antig_Construc = txtantigcons
   r_Inmueble.Sup_Tot_Polig = txttotpolig
   r_Inmueble.Precio_Vta = txtpvta
   r_Inmueble.Sup_AreaLibre = txtarealibre
   SI no se encuentra el tipo de inmueble en la lista ENTONCES
      r_Inmueble.Cve_Tipo = cmbTipoInmueble.ItemData(cmbTipoInmueble.ListIndex)
   FIN SI
   SI no se encuentra la clave de localidad en la lista ENTONCES
      r_Inmueble.Cve_Localidad = cmbLocalidad.ItemData(cmbLocalidad.ListIndex)
   FIN SI
   SI no se encuentra la clave de Estado en la lista ENTONCES
      r_Inmueble.Cve_Estado = cmbEdo.ItemData(cmbEdo.ListIndex)
   FIN SI
FIN DE PROCEDIMIENTO.


FUNCION Inmo_Grabar (Clave)
   SI r_Inmueble.existe ENTONCES
      ACTUALIZAR datos de la base con los temporales
   OTRO
      ADICIONAR registro a la tabla inmuebles
   FIN SI
FIN DE FUNCION.
```

```
FUNCION inmo_leer(clave)
   BUSCAR registro de acuerdo a la clave dada
   resultado = False
   SI existe el registro con la clave igual ENTONCES
      LLENAR registro temporal con datos de la tabla de inmuebles
      resultado = True
   OTRO
      r_Inmueble.existe = False
   FIN SI
   inmo_leer = resultado
FIN DE FUNCION.


PROCEDIMIENTO limpiactrl
   INICIALIZAR con valores nulos los campos del inmueble
   txtDescrip = gsNULL_STR
   TipoInmueble = ""
   txtubica = gsNULL_STR
   txtcol = gsNULL_STR
   localidad = gsNULL_STR
   Edo = ""
   txtarrenda = gsNULL_STR
   txtsueloact = gsNULL_STR
   txtusofut = gsNULL_STR
   txtsupdesplante = gsNULL_STR
   txtantigcons = gsNULL_STR
   txttotpolig = gsNULL_STR
   txtpvta = gsNULL_STR
   txtarealibre = gsNULL_STR
   Archivo_f = gsNULL_STR
   Archivo_P = gsNULL_STR
   fecha_pl = ""
   apeo = gsNULL_STR
   fecha_ad = ""
   dictamen_e = gsNULL_STR
   fecha_de = ""
   dictamen_i = gsNULL_STR
   fecha_di = ""
FIN DE PROCEDIMIENTO.
```

```
PROCEDIMIENTO reg2ctrl
   ASIGNA datos temporales al registro de inmuebles
   txtDescrip = r_Inmueble.Descrip_Inmueble
   txtubica = r_Inmueble.Ubicación
   txtcol = r_Inmueble.Colonia
   txtarrenda = r_Inmueble.Arrendado
   txtsueloact = r_Inmueble.Uso_Suelo_Act
   txtusofut = r_Inmueble.Potenc_Uso_Futuro
   txtsupdesplante = r_Inmueble.Sup_Desplante
   txtantigcons = r_Inmueble.Antig_Construc
   txttotpolig = r_Inmueble.Sup_Tot_Polig
   txtpvta = r_Inmueble.Precio_Vta
   txtarealibre = r_Inmueble.Sup_AreaLibre
   SI el tipo de construcción es permanente ENTONCES
      optPermanente.Value = True
   OTRO
      optPrivisional.Value = True
   FIN SI
   HACER CASO r_Inmueble.Sistema_Construc
         CASO optConcreto.Caption
               optConcreto.Value = True
         CASO optMetalica.Caption
               optMetalica.Value = True
         CASO optMixta.Caption
               optMixta.Value = True
         CASO optPrefab.Caption
              optPrefab.Value = True
   FIN CASO
   HACER CASO r_Inmueble.Estado_Conservac
         CASO optExcelente.Caption
               optExcelente.Value = True
         CASO optBueno.Caption
               optBueno.Value = True
         CASO optRegular.Caption
               optRegular.Value = True
         CASO optMalo.Caption
               optMalo.Value = True
         CASO optDefec.Caption
               optDefec.Value = True
   FIN CASO
FIN DE PROCEDIMIENTO.
```

```
FUNCION Trae_Nombre
   En caso de error ir a etiqueta ErrHandler
   OBTENER el nombre del archivo gráfico y abrirlo
   Salir de la FUNCION
   ErrHandler:
      Desplegar mensaje de error
FIN DE FUNCION.


PROCEDIMIENTO TraeDatoCtrl
   SI clave = "" ENTONCES
      Salir del Procedimiento
   FIN SI
   CONTAR el numero de registros de la tabla
   SI los registros contados > 0 ENTONCES
      LLENAR registro de control
   FIN SI
FIN DE PROCEDIMIENTO.


FUNCION Validar
   VALIDAR una variable que sirve como bandera
   Validar = True
FIN DE FUNCION.

PROCEDIMIENTO Cerrar_Click
   DESCARGAR los formatos de la memoria
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO cmdConinm_Click
   resultado = query_busca("con_inmuebles")
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO cmdFoto_Click
   SI el inmueble tiene fotografia ENTONCES
      lblFoto = Trae_Nombre()
   FIN SI
FIN DE PROCEDIMIENTO.
```

```
PROCEDIMIENTO cmdGrabar_Click
   VALIDAR datos para grabar información
   SI Validar() ENTONCES
      Ctrl2Reg
      rdo = Inmo_Grabar(xclave)
      SI rdo ENTONCES
         MsgBox "Datos grabados correctamente", vbInformation
      FIN SI
   FIN SI
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO cmdPlano_Click
   OBTENER el nombre del plano, si este existe
   SI el inmueble contiene planos ENTONCES
      lblPlano = Trae_Nombre()
   FIN SI
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO dictamen_e_Change
  VALIDAR Dictamen
  dictamen_e = Format(dictamen_e, ">")
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO dictamen_e_LostFocus
   CAPTURAR dato para dictamen
   SI dictamen_e <> "S" y dictamen_e <> "N" ENTONCES
     DESPLEGAR mensaje "Valor inválido. Escriba : S/N", 48, "¡E R R O R!"
    dictamen_e = "N"
    dictamen_e.SetFocus
   FIN SI
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO dictamen_i_Change
  VALIDAR Dictamen
  dictamen_i = Format(dictamen_i, ">")
FIN DE PROCEDIMIENTO.
```

```
PROCEDIMIENTO dictamen_i_LostFocus
   CAPTURAR dato para dictamen
   SI dictamen_i <> "S" y dictamen_i <> "N" ENTONCES
    DESPLEGAR mensaje "Valor inválido. Escriba : S/N", 48, "¡E R R O R!"
    dictamen_i = "N"
    dictamen_i.SetFocus
   FIN SI
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optBueno_Click
   ASIGNAR Estado de conservación
   r_Inmueble.Estado_Conservac = optBueno.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optConcreto_Click
   ASIGNAR Tipo de construcción
   r_Inmueble.Sistema_Construc = optConcreto.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optDefec_Click
   ASIGNAR Estado de conservación
   r_Inmueble.Estado_Conservac = optDefec.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optExcelente_Click
   ASIGNAR Estado de conservación
   r_Inmueble.Estado_Conservac = optExcelente.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optMalo_Click
   ASIGNAR Estado de conservación
   r_Inmueble.Estado_Conservac = optMalo.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optMetalica_Click
   ASIGNAR Tipo de construcción
   r_Inmueble.Sistema_Construc = optMetalica.Caption
FIN DE PROCEDIMIENTO.
```

```
PROCEDIMIENTO optMixta_Click
   ASIGNAR Tipo de construcción
   r_Inmueble.Sistema_Construc = optMixta.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optPermanente_Click
   ASIGNAR Tipo de construcción
   r_Inmueble.Tipo_Construc = optPermanente.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optPrefab_Click
   ASIGNAR Tipo de construcción
   r_Inmueble.Sistema_Construc = optPrefab.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optPrivisional_Click
   ASIGNAR Tipo de construcción
   r_Inmueble.Tipo_Construc = optPrivisional.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO optRegular_Click
   ASIGNAR estado de conservación del inmueble
   r_Inmueble.Estado_Conservac = optRegular.Caption
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO xclave_LostFocus
   SI IsNumeric(xclave) ENTONCES
      rdo = inmo_leer(xclave)
      LLENAR los combos
      llena_cmb cmbTipoInmueble, "Descrip_Tipo", "Cve_Tipo", "cat_tipo_de_inmueble"
      llena_cmb cmbLocalidad, "Descrip_Localidad", "Cve_Localidad", "cat_localidad"
      llena_cmb cmbEdo, "Descrip_Estado", "Cve_Estado", "cat_estados"

      SI No rdo ENTONCES
         msg = MsgBox("No se encontró la clave" & gsNewLine & "¿Desea darlo de
          alta?", vbQuestion + vbYesNo)
         HACER CASO msg
               CASO vbYes
                     limpiactrl
               CASO vbNo
         FIN CASO
      OTRO
```

```
        LLAMAR reg2ctrl
        LLAMAR TraeDatoCtrl cmbTipoInmueble, r_Inmueble.Cve_Tipo, "Cve_Tipo", "
        "cat_tipo_de_inmueble"
        LLAMAR TraeDatoCtrl cmbLocalidad, r_Inmueble.Cve_Localidad,
        "Cve_Localidad", "CAT_LOCALIDAD"
        LLAMAR TraeDatoCtrl cmbEdo, r_Inmueble.Cve_Estado, "Cve_Estado",
        "CAT_ESTADOS"
      FIN SI
    FIN SI
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO TraeDatoCtrl
  SI clave = "" ENTONCES
    Salir del Procedimiento
  FIN SI
  SI numero de registros de la tabla > 0 ENTONCES
    ctrl.Text = mirs.Fields(1)
  FIN SI
FIN DE PROCEDIMIENTO.
```

Forma fDATLEG

Forma que carga la pantalla para el manejo de los datos legales del sistema

```
FUNCION inmo_leerleg(clave)
  BUSCAR registro de inmueble con clave dada
  resultado = False
  SI existe registro de inmueble ENTONCES
    ASIGNAR datos de registro a registro temporal
    resultado = True
  OTRO
    r_InmuebleLeg.existe = False
  FIN SI
  inmo_leer = resultado
FIN DE FUNCION.


PROCEDIMIENTO Reg2CtrlLeg
  ASIGNAR datos de registro temporal a datos del inmueble
FIN DE PROCEDIMIENTO.
```

```
PROCEDIMIENTO cmdBusca_Click
   BUSCAR la consulta del inmueble
   resultado = query_busca("con_inmuebles")
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO cmdCerrar_Click
   DESCARGAR formatos de memoria
FIN DE PROCEDIMIENTO.


PROCEDIMIENTO txtClave_LostFocus
   SI IsNumeric(txtClave) ENTONCES
      rdo = inmo_leerleg(txtClave)
      SI No rdo ENTONCES
         DESPLEGAR mensaje ("No se encontró la clave" & gsNewLine & "¿Desea darlo
         de alta?", vbQuestion + vbYesNo)
         HACER CASO msg
               CASO vbYes
                     LimpiaCtrlLeg
               CASO vbNo
         FIN CASO
      OTRO
         LLAMAR Reg2CtrlLeg
      FIN SI
   FIN SI
FIN DE PROCEDIMIENTO.
```

CAPITULO 6

Diseño Estructurado del Sistema

6. DISEÑO ESTRUCTURADO DEL SISTEMA

El capítulo muestra la representación gráfica del Sistema Integral de Administración Inmobiliaria, SIAI, en la que se podrá observar el diseño modular que tiene el sistema, así como la independencia que existe entre los módulos.

En esta etapa del desarrollo del sistema, ya se tienen determinados los *requerimientos* solicitados por el usuario, por medio de los diagramas de flujo de datos, diagrama de *entidad-relación*, el *diccionario de datos* y *español estructurado*. Básicamente el propósito del capítulo es mostrar el diseño estructurado del sistema, a través de su carta de estructura, además de la forma en que se presentan visualmente los módulos al usuario, una vez terminados.

6.1 LA CARTA DE ESTRUCTURA (CDE).

La carta de estructura del Sistema Integral de Administración Inmobiliaria, SIAI, nos muestra la representación gráfica del sistema y su disposición modular.
Para la realización de la CDE se utilizó el paquete Microsoft Power Point en su versión 4.0. El motivo de utilizar este programa es la variedad de funciones para dibujar y realizar presentaciones rápidas y con mejor calidad. En la CDE del sistema se utilizó una numeración consecutiva ascendente para definir los niveles más bajos de la carta, iniciando en el nivel más alto de la carta de estructura con el número 1.

En primera instancia mostraremos la carta de estructura en el nivel 1 en la FIG. 6.1. Aquí se observa la parte principal del menú del sistema. En este caso el nivel más alto se representa como la raíz de nuestra CDE.

FIG. 6.1. Nivel 1 de la carta de estructura.

Después de observar el nivel más alto de la carta de estructura, el siguiente paso es pasar a un nivel subordinado o nivel 2, de la CDE, FIG. 6.2.

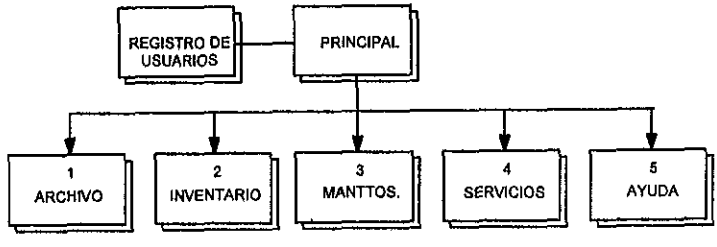


FIG. 6.2. Carta de estructura en el nivel 2.

En la FIG. 6.2 se observan los módulos *subordinados del* módulo PRINCIPAL, estos módulos son:

1. Módulo de archivo.
2. Módulo de inventario.
3. Módulo de mantenimientos.
4. Módulo de servicios.
5. Módulo de ayuda.

Estos cinco módulos están integrados por subprocesos, los cuales se describen a continuación:

1. Módulo de archivo.

   1.1 Compactar base de datos.
   1.2 Reparar base de datos.
   1.3 Seguridad.
   1.4 Configurar impresora.
   1.5 Salir.

2. Módulo de Inventario.

   2.1 Inmuebles.
   2.2 Información legal.
   2.3 Gastos.
   2.4 Publicidad.

3. Módulo de Mantenimiento.

   3.1 Programación.
   3.2 Avance.
   3.3 Estadísticas.

4. Módulo de Servicios.

   4.1 Generador de consultas.
   4.2 Catálogos.
   4.3 Opciones.

5. Módulo de Ayuda.

   5.1 Contenido.
   5.2 Buscar ayuda acerca de.
   5.3 Acerca de.

A partir de este nivel se observa una organización jerárquica entre los elementos. Los módulos tienen una distribución en forma de organigrama, cada módulo va ligado a un nivel superior y cada módulo tiene otros niveles subordinados hacia abajo, correspondiendo al nivel 2.

Para describir el nivel 3, se iniciará con el módulo de ARCHIVO, su carta de estructura se muestra en la FIG. 6.3.

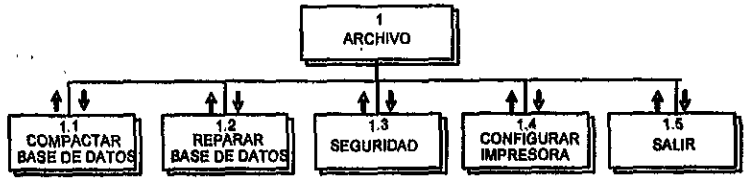


FIG. 6.3. Nivel 3, Módulo de ARCHIVO.

En la FIG. 6.3 se observan los cinco subprocesos del módulo de ARCHIVO. A continuación se describen las funciones que realiza cada uno de ellos:

1. Módulo de ARCHIVO.

En este módulo se tienen las herramientas para el manejo y respaldo de las bases de datos, las cuales se describen a continuación:

1.1 Compactar base de datos. Esta herramienta es necesaria para realizar respaldos de la información, la compactación de la base de datos nos ayuda a reducir el espacio que pueda ocupar la base de datos.

1.2 Reparar base de datos. Esta herramienta nos ayuda a restaurar las bases de datos compactadas, es decir, recupera la información del archivo de respaldo.

1.3 Seguridad. Esta opción actualiza los permisos de acceso a los usuarios del sistema. Aquí se abarcan los siguientes procesos:

- Altas, bajas y cambios de grupos.
- Altas, bajas y cambios de usuarios.
- Permisos para los menús del sistema.

1.4 Configurar impresoras. Esta opción configura las características de la impresora que se utilizará para obtener los reportes impresos del sistema. Se configura tamaño y orientación del papel, también se seleccionan los controladores de las impresoras.

El siguiente módulo que se desarrollará es el de INVENTARIO, al igual que el anterior la CDE en su nivel 3 se observa en la FIG. 6.4.

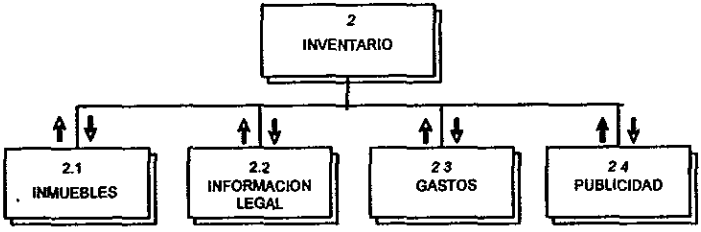


FIG. 6.4. Módulo de INVENTARIO en el nivel 3.

En el nivel 3, se observan los cuatro procesos subordinados del módulo de INVENTARIO, los cuales son: Inmuebles, Información legal, Gasto y Publicidad. A continuación se explica brevemente la función de cada submódulo:

2. Módulo de INVENTARIO.

2.1 Inmuebles. Esta opción se encarga del control de todos los datos generales de cada inmueble, entre los que se encuentran: la ubicación, el uso de suelo, la antigüedad, precio y tipo de inmueble, además del control de inventario.
2.2 *Información legal. En esta opción se capturan* los antecedentes de la propiedad, predio, por quién fue adquirida, superficie y predio.
2.3 Gastos. En esta opción puede capturarse y consultarse el monto de los diferentes gastos que se aplican a los inmuebles, tales como servicios de luz, agua, predial.
2.4 *Publicidad. En esta opción se pueden observar fotos del inmueble*, además de información referente al mismo.

El módulo de MANTENIMIENTO se muestra en la FIG. 6.5, aquí se observan los procesos relacionados a dicho módulo.

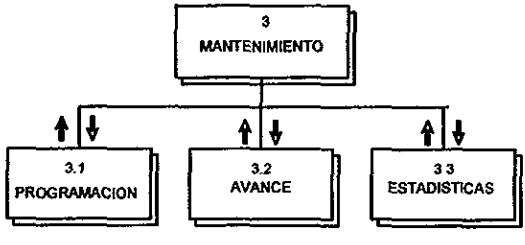


FIG. 6.5. Módulo de MANTENIMIENTO, carta de estructura nivel 3.

3. Módulo de MANTENIMIENTO.

3.1 Programación. Este módulo se encarga de la programación de mantenimientos, en este proceso se asigna un número de programa de mantenimiento a los inmuebles, para darles seguimiento y asignar los recursos necesarios para su ejecución.
3.2 Avance. Este módulo permite registrar el avance por partida para los mantenimientos programados.
3.3 Estadísticas. En este módulo se muestran gráficamente el avance real y programado por parte de la empresa.

La carta de estructura del módulo de SERVICIOS se muestra en la FIG. 6.6.

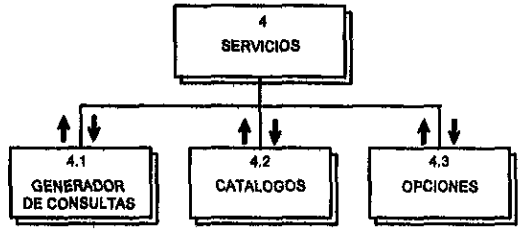


FIG. 6.6. Carta de estructura del módulo de SERVICIOS en el nivel 3.

4. Módulo de SERVICIOS.

4.1 Generador de consultas. Puede generar reportes o consultas personalizadas.
4.2 Catálogos. Módulo donde se actualizan los catálogos siguientes: Estados, partidas, tipos de inmueble, ciudad, contratistas, localidades, gastos, componentes y clasificación de inmuebles.
4.3 Opciones. Este módulo permite diseñar el encabezado para los reportes, da opción de elegir la presentación de la información, por ejemplo los decimales con los que se desea el reporte.

El último módulo descrito en la CDE, corresponde al de AYUDA, el cual se muestra en la FIG. 6.7.

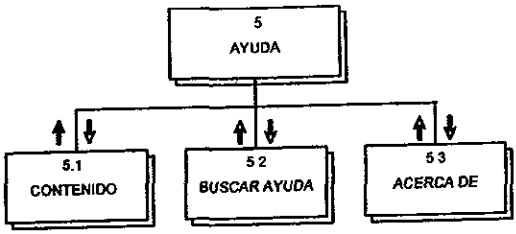


FIG. 6.7. Carta de estructura del módulo de AYUDA en nivel 3.

5. Módulo de AYUDA.

En este módulo se consultan dudas o aclaraciones sobre el manejo del sistema, además de información y conceptos utilizados en el SIAI.

5.1 Contenido. Muestra un índice del contenido de la ayuda del sistema.
5.2 Buscar ayuda. Busca una cadena de caracteres en especial.
5.3 Acerca de. Esta opción muestra la versión y los derechos de autor del sistema.

Hasta este punto se ha desarrollado la CDE del SIAI en su nivel 3, el siguiente paso es el desglose de cada uno de los módulos antes mostrados, al cual le corresponde el nivel 4. Primero iniciaremos con el módulo 1, en este caso el único proceso que contiene más elementos subordinados es el módulo 1.4, que corresponde a la opción de configurar impresora, su carta se muestra en la FIG. 6.8.

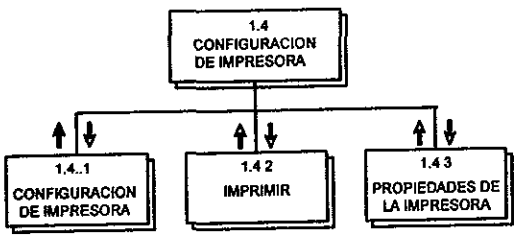


FIG. 6.8. Nivel 4 del módulo de ARCHIVOS.

Las funciones de estos submódulos se describen a continuación:

1.4 Configuración de impresora.

1.4.1 Configuración de impresora. Este proceso se encarga de seleccionar el tipo de impresora, tamaño y orientación del papel.
1.4.2 Imprimir. Generar la impresión.
1.4.3 Propiedades de la impresora. Para seleccionar el papel y la calidad de impresión.

La CDE del módulo 1.1 Inmuebles del INVENTARIO se presenta en la FIG. 6.9.

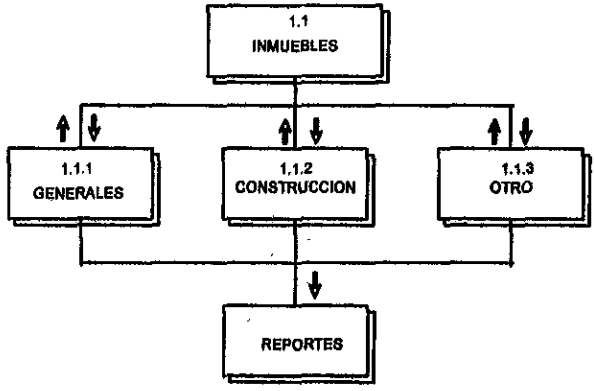


FIG. 6.9 Módulo 1.1 INVENTARIO (Inmuebles), nivel 4 de abstracción.

Como se observa en la FIG. 6.9 los procesos subordinados al módulo de INVENTARIO son: Generales, Construcción y Otro, aquí se observa la estructura jerárquica que guarda la CDE del SIAI.

A continuación se describen brevemente algunas de las funciones que realizan los submódulos.

1.1 Inmuebles.

1.1.1 Generales. En esta opción se captura la ubicación del inmueble desglosado en calle, colonia, localidad y estado, la antigüedad, el precio y tipo de inmueble que es considerado.
1.1.2 Construcción. Describe el tipo de material y construcción del inmueble, así como una evaluación del estado de conservación y la superficie total del inmueble.

1.1.3 Otro. En esta opción se marcan los dictámenes técnicos (estructural e instalaciones), además de manejar los archivos de los planos y fachadas de los inmuebles.

El siguiente submódulo es el de Información Legal, su CDE se representa en la FIG. 6.10.

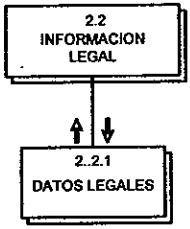


FIG. 6.10. Módulo de INVENTARIOS Información Legal 2.2.

2.2 Información Legal.

2.2.2 Datos legales. Estos datos se refieren a la información de las escrituras, predio, propietario, etc.

Respecto al módulo de INVENTARIO son los únicos submódulos que presentan un nivel 4 de abstracción.

El módulo 3. MANTENIMIENTO , tiene la CDE de nivel 4, en el submódulo de Programación 3.1, FIG. 6.11.

FIG. 6.11. Módulo de MANTENIMIENTO 3, submódulo 3.1 de Programación.

A continuación se describen algunas de las funciones que realiza este proceso.

3.1 Programación.

3.1.1 Asignación de Programa de mantenimiento. Este submódulo asigna un número al programa de mantenimiento seleccionado describe en qué consiste dicho programa.

Siguiendo con la descripción de nivel 4 de la CDE, el siguiente módulo a desarrollar es el módulo de 4. SERVICIOS, con el submódulo de Catálogos 4.2. La FIG. 6.12 muestra la CDE del submódulo 4.2 Catálogos.

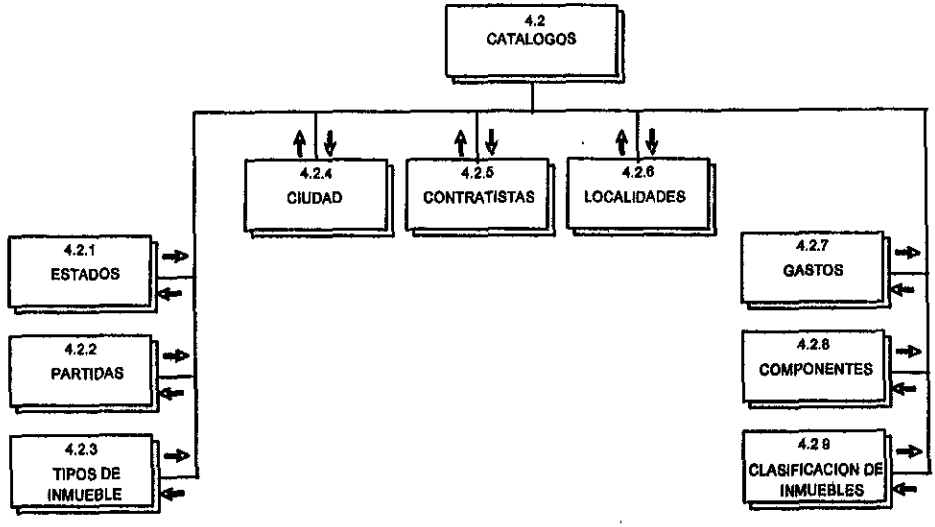


FIG. 6.12. Carta de estructura nivel 4 del módulo de SERVICIOS.

4.2 Catálogos.

4.2.1 Estados. En este catálogo se realizan las altas y modificaciones a los estados donde se encuentran ubicados los inmuebles.
4.2.2 Partidas. En este catálogo se encuentran las partidas que maneja la Empresa Inmobiliaria para los mantenimientos.
4.2.3. Tipos de Inmueble. Este catálogo contiene el tipo de inmuebles que se manejan en la Empresa que son los propios.
4.2.4 Ciudad. En este catálogo se realizan las altas y modificaciones a las ciudades en donde se encuentran los inmuebles de la Empresa.
4.2.5 Contratistas. En este catálogo se encuentran los contratistas que trabajan para la Empresa.
4.2.6. Localidades. En este catálogo se realizan las altas y modificaciones a las localidades en donde se encuentran ubicados los inmuebles.
4.2.7 Gastos. En este catálogo se encuentran los gastos que generan los inmuebles.

4.2.8 Componentes. En este catálogo se encuentran los componentes que forman parte de los inmuebles. Aquí se pueden actualizar y modificar los mismos.

4.2.9 Clasificación de Inmuebles. En este catálogo se encuentra la clasificación de los inmuebles que son: casas, edificios, hoteles y terrenos.

Como se observa en las figuras ya expuestas, cada submódulo realiza una sola función dentro del sistema.

6.2 MODULARIDAD.

Como se mencionó, la modularidad consiste en separar el problema en partes más pequeñas. Para el caso del SIAI se observa en la FIG. 6.2 la carta de estructura del sistema en su nivel 2. Aquí se representan los cinco módulos que integran al SIAI, en donde cada uno realiza una función, la cual sólo se reconoce por el nombre del módulo. Por ejemplo, el módulo de ARCHIVO realiza procesos que se relacionan con el manejo y control de los archivos de información.

En el caso del módulo de INVENTARIO realiza sólo funciones relacionadas al inventario de los inmuebles. Para el módulo de MANTENIMIENTO, éste realiza la programación y control de los programas de mantenimiento sin afectar su operación a otros módulos del sistema.

Revisando en niveles más bajos de cada módulo, se observa que a su vez están integrados por otros submódulos, esto conduce a concluir que los submódulos finales realizan sólo un proceso dentro del sistema. Como se observa en la FIG. 6.9, aparece un submódulo de Construcción, el cual sólo maneja información referente al estado de conservación de la construcción.

Siguiendo con los otros módulos hasta sus últimos niveles de abstracción, es importante hacer notar, como la carta de estructura mostrada en la FIG. 6.9, cuenta con una estructura jerárquica y con módulos independientes entre sí.

La modularidad del sistema, se observa en los CUADROS 6.1, 6.2, 6.3, 6.4 y 6.5 a continuación, los cuales muestran segmentos de pseudocódigo en donde se observan algunos procedimientos que se agrupan por el módulo que ejecutan.

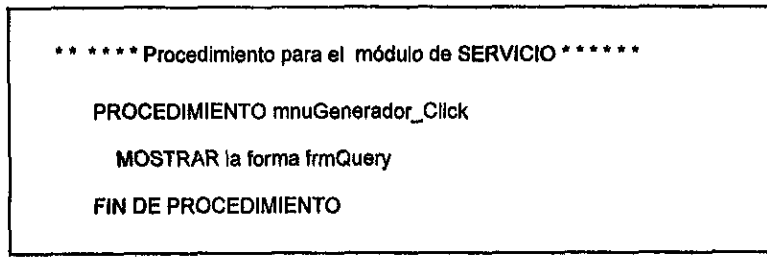
```
********* Procedimiento para el módulo de ARCHIVO *********

    PROCEDIMIENTO imgFileNewButton_Click
      RESTAURA con datos de inicialización el botón de abrir nuevo
    archivo
      MOSTRAR la forma fasígna
    FIN DE PROCEDIMIENTO

    PROCEDIMIENTO imgFileOpenButton_Click
      RESTAURA con datos de inicialización el botón de abrir archivo
    FIN DE PROCEDIMIENTO

    PROCEDIMIENTO mnuConfig_Click
    LLAMAR al procedimiento de configuración de impresora en el control
    de dialogo común.
      CMDialog1.Flags = &H40  ' Sólo el dialogo de configuración de
    impresora.
      MOSTRAR la forma de configuración de impresora
    FIN DE PROCEDIMIENTO
```

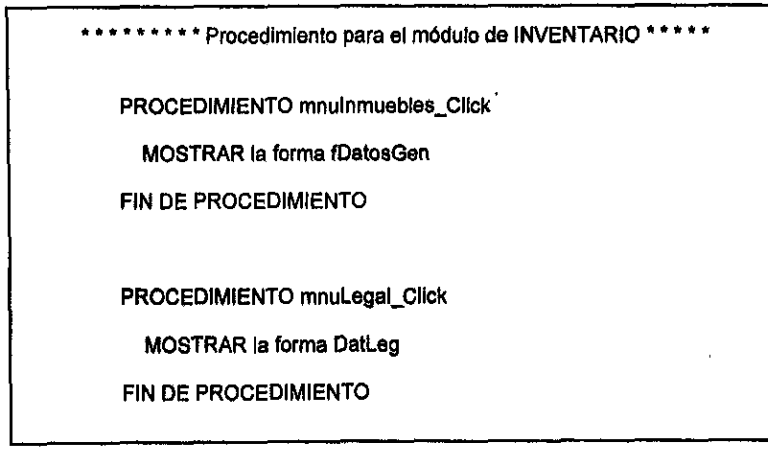
CUADRO. 6.1. Segmento de pseudocódigo del SIAI, módulo ARCHIVO.



```
** **** Procedimiento para el  módulo de SERVICIO ******

    PROCEDIMIENTO mnuGenerador_Click
      MOSTRAR la forma frmQuery
    FIN DE PROCEDIMIENTO
```

CUADRO. 6.2. Segmento de pseudocódigo del SIAI, módulo SERVICIOS.



```
********* Procedimiento para el módulo de INVENTARIO *****

    PROCEDIMIENTO mnuInmuebles_Click
      MOSTRAR la forma fDatosGen
    FIN DE PROCEDIMIENTO


    PROCEDIMIENTO mnuLegal_Click
      MOSTRAR la forma DatLeg
    FIN DE PROCEDIMIENTO
```

CUADRO. 6.3. Segmento de pseudocódigo del SIAI, módulo INVENTARIO.

```
*** Procedimiento para el módulo de MANTENIMIENTO ***

  PROCEDIMIENTO mnuProgMantenimiento_Click
    MOSTRAR la forma fProgramación
  FIN DE PROCEDIMIENTO
  PROCEDIMIENTO mnuEjecucion_Click
    MOSTRAR la forma frmEjecuta
  FIN DE PROCEDIMIENTO

  PROCEDIMIENTO mnuProgramacion_Click
    MOSTRAR la forma fasigna
  FIN DE PROCEDIMIENTO
```

CUADRO. 6.4. Segmento de pseudocódigo del SIAI, módulo MANTENIMIENTO.

```
********* Procedimiento para salir del sistema *********

  PROCEDIMIENTO mnuSalir_Click
    SALIR del sistema
  FIN DE PROCEDIMIENTO
```

CUADRO. 6.5. Segmento de pseudocódigo del SIAI, módulo SALIR.

En este pseudocódigo se observa como cada segmento de código ejecuta un módulo del sistema, por ejemplo, los procedimientos que utiliza el módulo de INVENTARIO, son fDatosGen y DatLeg, los cuales corresponden a la opción de Inmuebles y a información legal del módulo de INVENTARIO, respectivamente.

Estos módulos los observamos en la FIG. 6.13, que es la pantalla del módulo INVENTARIO que contiene los submódulos Inmuebles, Información Legal, Gastos y Publicidad.

FIG. 6.13. Módulo de INVENTARIO, opciones de Inmuebles, Información Legal, Gastos y Publicidad.

El diseño modular facilita la realización de modificaciones en los módulos. Además el modificar algún módulo no provoca efectos secundarios en los otros elementos, es más fácil detectar errores y reutilizar alguno de los mismos.

Una forma de ver la modularidad del sistema es que está integrado por procesos bien definidos, los cuales tienen funciones que realizan un solo proceso. Esto se observa en la FIG. 5.7 del diagrama de flujo de datos del sistema, el cual corresponde al módulo principal del Inventario de inmuebles, donde se observa como existen procesos con una sola función. Por ejemplo, el proceso que realiza la consulta de los datos del inmueble sólo realiza esa función.

Por último, se presenta la pantalla de entrada del SIAI, en la cual se observan los módulos principales del sistema.

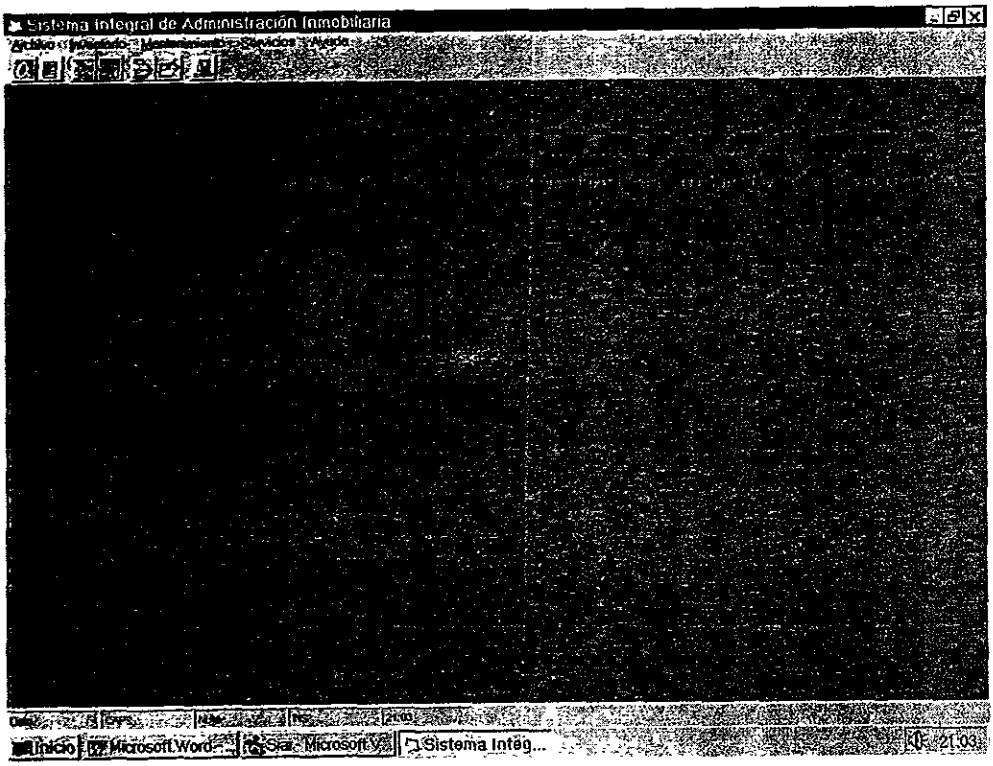


FIG. 6.14. Pantalla principal del SIAI, módulos que integran al sistema.

6.3 COHESION.

Los módulos desarrollados se diseñaron utilizando procedimientos, en donde se maneja el paso de parámetros. De esta forma se realiza el intercambio de datos entre módulos, como se observa en el CUADRO 6.6, el parámetro CLAVE es la información que se pasa a la función Inmo_leer.

```
Función inmo_leer(clave)
   Busca registro de acuerdo a la clave dada
   resultado = False
   Si existe el registro con la clave igual Entonces
      Llenar registro temporal con datos de la tabla de inmuebles
      resultado = True
   En caso contrario
      r_inmueble.existe = False
   Fin de si
   inmo_leer = resultado
Fin de función
```

CUADRO. 6.6. Segmento de pseudocódigo de la función INMO_LEER.

Adicionalmente, en el CUADRO 6.7 se observa el paso de parámetros entre funciones, en donde la función Inmo_Grabar realiza la función de grabar los datos del inmueble por medio del parámetro CLAVE.

```
Función Inmo_Grabar (Clave)
  Si r_Inmueble.existe Entonces
    Actualizar datos de la base con los temporales
  En caso contrario
    Adicionar registro a la tabla inmuebles
  Fin de si
Fin de función
```

CUADRO. 6.7. Segmento de pseudocódigo de la función INMO_GRABAR.

En este pseudocódigo se observa, como la función realiza sólo el proceso de grabar los datos de un inmueble. Este a su vez es llamado por otra función a través del paso de parámetros.

También se observa la cohesión entre los elementos del SIAI, por medio del diagrama de flujo de datos mostrado en la FIG. 5.7. Donde los procesos de generar reportes, actualizar inmuebles, consulta a los inmuebles y captura de planos se encuentran relacionados para realizar una sola función.

Por lo anterior se deduce que el sistema tiene una cohesión funcional, ya que los procesos realizan una sola función.

6.4 ACOPLAMIENTO.

El sistema SIAI tiene acoplamiento débil, debido al uso de parámetros, los cuales sirven únicamente para pasar información entre los procedimientos. Esta característica se observa en la FIG. 5.7 donde el nombre del parámetro que se pasa es CVE_INMUEBLE, que es la única entrada que se requiere para ese módulo. Con ello se muestra que el flujo de información entre procesos solamente pasa los datos necesarios al siguiente proceso del módulo.

Por lo anterior, se deduce que el sistema tiene un acoplamiento bajo entre sus módulos debido a la poca interacción que existe entre los mismos. Lo que significa que el sistema SIAI tiene acoplamiento de datos.

CAPÍTULO 7

Desarrollo del Sistema

7. DESARROLLO DEL SISTEMA

7.1 LA PROGRAMACION SISTEMATICA.

Recordando el concepto de programación sistemática explicado en el capítulo 2, la programación sistemática es una técnica basada en la teoría de crear una estructura de programa a partir de estructuras de control que son métodos para especificar el orden en que las instrucciones de un algoritmo se ejecutarán.

De acuerdo a lo anterior se contemplaron diversos lenguajes de programación existentes en el mercado que trabajaran con las estructuras de control básicas (secuencia, selección y repetición) para hacer la selección del lenguaje de programación que se utilizaría para desarrollar el Sistema Integral de Administración Inmobiliaria (SIAI).

Así mismo, se verificó que con dichos lenguajes pudieran emplearse varias o todas las estructuras de control para manejo de ciclos: **FOR**, **WHILE-DO** y **REPEAT-UNTIL** y construcciones de decisión: **IF THEN**, **IF THEN ELSE** y **CASE**.

Los lenguajes de programación disponibles en el mercado que se tomaron en cuenta que tienen la característica de poder desarrollar programas con la programación sistemática se presentan en la TABLA 7.1.

LENGUAJE DE PROGRAMACION	FABRICANTE
Visual Basic	Microsoft
Visual Fox Pro	Microsoft
Visual C	Microsoft
Delphi	Borland
C++	Borland
Dbase IV	Borland
Oracle	Oracle
Fox Pro para Windows	Microsoft
Clipper	Computer Asociates

TABLA 7.1. Lenguajes de programación disponibles en el mercado que abarcan la programación sistemática.

Cabe señalar que cuando se tomaron en cuenta todos los factores para seleccionar el lenguaje de programación con el que se desarrolló el sistema como: costo, ambientes operativos bajo los cuales operará el sistema, características que poseen para trabajar en red y arquitectura cliente servidor, adaptación al análisis y diseño del sistema, etc., esta lista se redujo.

7.2 CLASES Y CARACTERISTICAS DE LOS LENGUAJES DE PROGRAMACION.

7.2.1 CLASES DE LOS LENGUAJES DE PROGRAMACION.

En este punto de determinó cuál sería el software para el sistema en los dos niveles: el front-end y el back-end.

Front-end.

El front-end, como ya se ha mencionado, es el software con el que se desarrolla la aplicación. Para determinar cuál sería el front-end empleado para el sistema se tomaron en cuenta los siguientes aspectos:

1. Costo.
2. Adaptación al análisis y diseño del sistema.
3. Cumplimiento de los conceptos de programación sistemática.
4. Correr bajo ambiente Windows y bajo Novel netware 4.1 que es el sistema operativo de red donde se instalará el sistema.
5. Portabilidad.
6. Incluír un entorno de programación y acceso a todas sus herramientas dentro de este entorno.
7. Incluír entre sus herramientas: depurador, editor de menús, diseñador de reportes y creador de formas o pantallas.
8. Manejo de gran volumen de datos y la consistencia de ellos.
9. Equilibrio en cuanto a velocidad entre el volumen de información y el manejo de transacciones.
10. Soporte de bases de datos relacionales y por consiguiente del lenguaje estructurado de consulta SQL (Structured Query Languaje).
11. Creación de programas ejecutables.
12. Soporte a características de multimedia como manejo de imágenes y archivos de sonido.
13. Características para operar en red y arquitectura cliente-servidor.
14. Conocimiento del personal del Departamento de Sistemas de la empresa.
15. Expectativa de vida del lenguaje de programación.

Las opciones existentes en el mercado de lenguajes de programación que se tomaron en cuenta para la selección se muestran en la TABLA 7.2.

LENGUAJE DE PROGRAMACION	FABRICANTE
Visual Basic	Microsoft
Visual Fox Pro	Microsoft
Delphi	Borland

TABLA 7.2. Lenguajes de programación disponibles en el mercado que se tomaron en cuenta y cumplen con los aspectos requeridos.

Se eliminaron como posibles opciones, respecto de la TABLA 7.1:

VISUAL C y C++.

Una razón de peso por la que no se tomaron en cuenta éstos dos lenguajes es que son orientados a objetos y tanto el análisis como el diseño del Sistema Integral de Administración Inmobiliaria están basados en el análisis y diseño estructurado. Por lo tanto, no hubiera sido conveniente tratar de adaptar un lenguaje orientado a objetos con un analisis y diseño estructurados.

ORACLE para Novell Netware.

ORACLE es reconocido como la mejor base de datos del mercado en ambientes operativos como UNIX y también, por mucho, la mas costosa. Por lo mismo nunca se consideró como una fuerte opción y hay que recordar que la versión para Novell Netware es muy reciente en comparación a las demás opciones, también el personal de la empresa no está capacitado en la operación de ORACLE.

DBASE IV.

Es un lenguaje que tiene poco tiempo de estar disponible en ambiente WINDOWS, no tiene grandes expectativas de vida y no soporta programación orientada a eventos al nivel de Visual Basic 4.0 entre otras razones. Se sabe que en la década pasada Dbase era el lenguaje Xbase por excelencia, pero en los últimos años se ha quedado claramente atrás respecto a lenguajes de programación orientados a objetos, eventos, multiplataformas, etc.

CLIPPER V.5.2.

Por las mismas razones que DBASE IV y por no soportar bases de datos relacionales ni el lenguaje SQL, además de carecer de un entorno de programación a la altura de sus competidores.

FOX PRO para Windows V.2.6.

Por ser una versión descontinuada del lenguaje a pesar de estar presente todavía en muchas aplicaciones de las empresas.

Back-end.

El back-end, como ya se ha mencionado, es el software con el que se crea, administra y se da mantenimiento a la base de datos. Para determinar cuál sería el back-end empleado para el sistema se tomaron en cuenta los siguientes aspectos:

1. Costo.
2. Correr bajo ambiente Windows y bajo Novel netware 4.1 que es el sistema operativo de red donde se instalará el sistema.
3. Compatibilidad con el lenguaje de programación en el que se desarrollará el sistema en cuanto a la base de datos.
4. Además de ser un sistema creador y administrador de base de datos, permita dar mantenimiento y realizar transacciones en la misma.
5. Hacer posible la creación de formularios, obtener consultas y reportes y crear módulos para explotar la base de datos.
6. Propiedades de seguridad y accesos para la base de datos y tablas que la componen.
7. Nivel alto y confiable en cuanto a la integridad de los datos.
8. Soporte de bases de datos relacionales.
9. Características para operar en red y arquitectura cliente-servidor.
10. Conocimiento del personal del Departamento de Distemas de la empresa.
11. Poder operarse fácilmente y en ambiente amigable para el usuario.
12. Expectativa de vida del software.

De las opciones que existen en el mercado sólo se tomo en cuenta una: Microsoft Access en la versión del paquete integrado OFFICE para WINDOWS 95.

Se eligió Microsoft Access porque además de cumplir con los aspectos antes mencionados posee características que lo hacen ser un ideal back-end para el Sistema de Administración Inmobiliaria, entre otras se puede mencionar que no es tan robusto como otros (por ejemplo el SQL-Server de Microsoft o el Administrador de bases de datos de Oracle) pero aún así puede satisfacer las necesidades de crecimiento del SIAI.

También se eligió porque la empresa cuenta con él así que el costo es nulo, el personal del Departamento de Sistemas ya está capacitado en él y es un software con altas expectativas de vida y muestra de ello es que se recientemente salió al mercado Access 97 incluído en el paquete integrado OFFICE 97 y es totalmente compatible con Visual Basic 4.0 que es el front-end que finalmente fue elegido para desarrollar el SIAI.

7.2.2 CARACTERISTICAS DE LOS LENGUAJES DE PROGRAMACION.

Los tres lenguajes de programación presentados en la TABLA 7.2 poseen herramientas con las cuales puede ser desarrollado el SIAI, además cumplen en gran medida con los aspectos requeridos. Sin embargo, analizando mas a fondo las características de cada uno de ellos y prestando más atención en algunos aspectos relevantes, se llegó a la conclusión de que la mejor opción es Visual Basic, por lo tanto, el lenguaje elegido para desarrollar el Sistema Integral de Administración Inmobiliaria es Visual Basic V. 4.0.

Se eligió Visual Basic porque además de cumplir con todos los aspectos requeridos, cuenta con características que lo hacen ser el ideal front-end para el Sistema Integral de Administración Inmobiliaria.

También se eligió porque la empresa cuenta con él, así que parte del costo es nulo, el personal del Departamento de Sistemas de la Empresa Inmobiliaria ya está capacitado en él y es un software con altas expectativas de vida y muestra de ello es que recientemente salió al mercado la versión 5.0 de Visual Basic.

Es además orientado a eventos y uno de los lenguajes con más experiencia dentro del ambiente WINDOWS y el de mayor aceptación dentro de las empresas que cuentan o está entre sus planes contar con una aplicación administrativa bajo ambiente WINDOWS y que opere en red y/o arquitectura cliente-servidor.

CARACTERISTICAS DE VISUAL BASIC V.4.0.

- Acceso a bases de datos con el **Control Data**. En esta versión sólo se puede utilizar para abrir objetos de bases de datos existentes. Sin embargo, se pueden crear bases de datos, tablas e índices ejecutando la aplicación **Data Manager**.
- Visualización y manipulación de datos de otras aplicaciones Windows utilizando el control **OLE** (Enlace e incrustación de objetos). La manera de usar OLE en Visual Basic es mediante el control OLE de la Caja de Herramientas. Este control permite que el usuario muestre en pantalla datos de otra aplicación y que edite los datos en la misma aplicación donde fueron creados.
- Acceso y manipulación de objetos incrustados y enlazados suministrados por otras aplicaciones utilizando OLE Automation.
- Control para utilizar las cajas de diálogo comunes utilizadas (abrir, guardar como, imprimir, color y fuentes).
- Menús desplegables flotantes.
- Opción de guardar el proyecto automáticamente antes de que se ejecute.
- Creación de discos de distribución para aplicaciones con **Setup Wizard**.
- Soporte para accesar a bases de datos. Visual Basic implementa el acceso a los datos, incorporando el mismo mecanismo de base de datos que Microsoft Access. De esta forma, Visual Basic puede acceder a muchas bases de datos estándar,

como Microsoft Access, Microsoft Fox Pro, dBASE, Paradox, Oracle y Microsoft SQL Server.
- Programa **Crystal Reports** para la creación de informes, listados y documentos utilizando la información de una base de datos.
- **Outline** (Control esquema) para visualizar los elementos de una lista de manera jerárquica. Por ejemplo, mostrar directorios y archivos en una estructura de árbol.
- Conectarse con una base de datos utilizando **drivers** (controladores) ODBC. Utilizando Visual Basic y drivers **ODBC** (Open Database Connectivity), se puede conectar una base de datos a tablas de una base de datos ODBC, tal como Microsoft SQL Server u Oracle. Antes de conectarse a una base de datos ODBC, es necesario utilizar el programa de instalación de Visual Basic para instalar el driver ODBC apropiado.
- Creación de controles personalizados con el **CDK** (Control Development Kit) y de controles personalizados en tres dimensiones, como cajas de texto, botones de click, marcos, botones de opción, controles para comunicaciones y control de conteo entre otros.
- Creación de archivos de ayuda con el Help Compiler. Ayuda en línea como la que poseen las aplicaciones de Windows.
- Manejo de Multimedia.
- Ofrece una gran capacidad y velocidad en su sofisticado debugger.
- Permite manipular otras aplicaciones para utilizarlas como componentes en aplicaciones propias (Ej. Word, Excell, Project, etc.), siempre y cuando dichas aplicaciones soporten OLE automation.
- Interprete poderoso que permite la detección de errores al momento de edición de programa.
- Maneja aplicaciones Cliente-Servidor.

A continuación se mencionan algunas de las razones por las cuáles no se eligieron los otros lenguajes de programación.

- La Empresa Inmobiliaria cuenta con Visual Basic V. 4.0, así que únicamente tiene que adquirir las licencias restantes para desarrollador. Por otra parte, la elección de cualquier otro lenguaje representaba una inversión extra para la empresa.
- El personal del Departamento de Sistemas de la Empresa Inmobiliaria no está capacitado en ninguno de los otros lenguajes que se eligieron como posibles opciones al nivel que lo está en Visual Basic.

VISUAL FOX PRO V.5.

Es un lenguaje con grandes características de programación estructurada porque es totalmente compatible con su antecesor Fox Pro para Windows, pero no existen muchas aplicaciones todavía en las empresas que lo respalden y aunque supera a Visual Basic en cuanto a características de programación orientada a objetos, éstas no se utilizarían porque el Sistema Integral de Administración Inmobiliaria se analizó y diseño de manera estructurada y no orientado a objetos.

DELPHI.

Es una de las opciones que contaron con todo lo requerido, su costo es menor al de otros, cuenta con una rápida y creciente popularidad y tiene buenas expectativas de vida, pero finalmente habría que invertir en su adquisición y en la capacitación del personal del Departamento de Sistemas.

7.3 LAS HERRAMIENTAS DE PROGRAMACION.

7.3.1 CARACTERISTICAS, VENTAJAS Y DESVENTAJAS DEL COMPILADOR VISUAL BASIC 4.0.

El desarrollo de aplicaciones con interface gráfica de usuario hace suponer que el futuro en la industria de las computadoras se dirige hacia *ambientes gráficos*. Un ejemplo de ello son las aplicaciones de Windows que cuentan con una interface gráfica consistente e intuitiva. Con lo cual, el usuario dispone de más tiempo para dominar la *aplicación sin preocuparse* por dominar comandos y opciones. Microsoft al crear el ambiente gráfico Windows, requirió de un sistema de diseño que explotara esta interface de manera sencilla, sin tener que utilizar un lenguaje más especializado para manipular sistemas. Con esto surge una nueva técnica: la programación orientada a eventos, la cual se creó pensando que es más importante el aspecto y la respuesta del usuario que la programación del código que permite manipular la entrada y la salida de la información; ya que la interface siempre ha llevado más tiempo en diseñarse que el *problema a resolverse en sí*.

Así, aparece en 1987 Visual Basic 1.0, orientado al diseño rápido y eficaz de interfaces basadas en controles (botones, barras de desplazamiento, listas desplegables, etc.) que captan los datos necesarios para el programa, el manejo de ventanas, el código asociado a los eventos y además de propósito general, resultando el arrastre de controles a la forma una herramienta poderosa para el diseño visual de pantallas de captura y *despliegue de datos*.

Antes, desarrollar aplicaciones para Windows requería de expertos programadores en C. Ahora, gracias a Visual Basic, es posible programar en menor tiempo cualquier aplicación, sin importar su complejidad. Los errores de programación no se generan tan frecuentemente, son más sencillos de detectar y corregir. Esto no significa eliminar el C o el ensamblador en la programación para Windows, ya que aún se requiere de herramientas disponibles *únicamente en estos lenguajes*. Visual Basic es un lenguaje desarrollado para proveer al programador de un método rápido y sencillo para el desarrollo de aplicaciones Windows. Incluye diversas herramientas para el diseño de aplicaciones gráficas y un lenguaje basado en sus antecesores Basic y QuickBasic.

Arranque de Visual Basic.

Para arrancar Visual Basic se tienen varias alternativas, *sin embargo*, la *manera más* sencilla de hacerlo es con un doble click sobre su ícono como se indica en la FIG. 7.1.

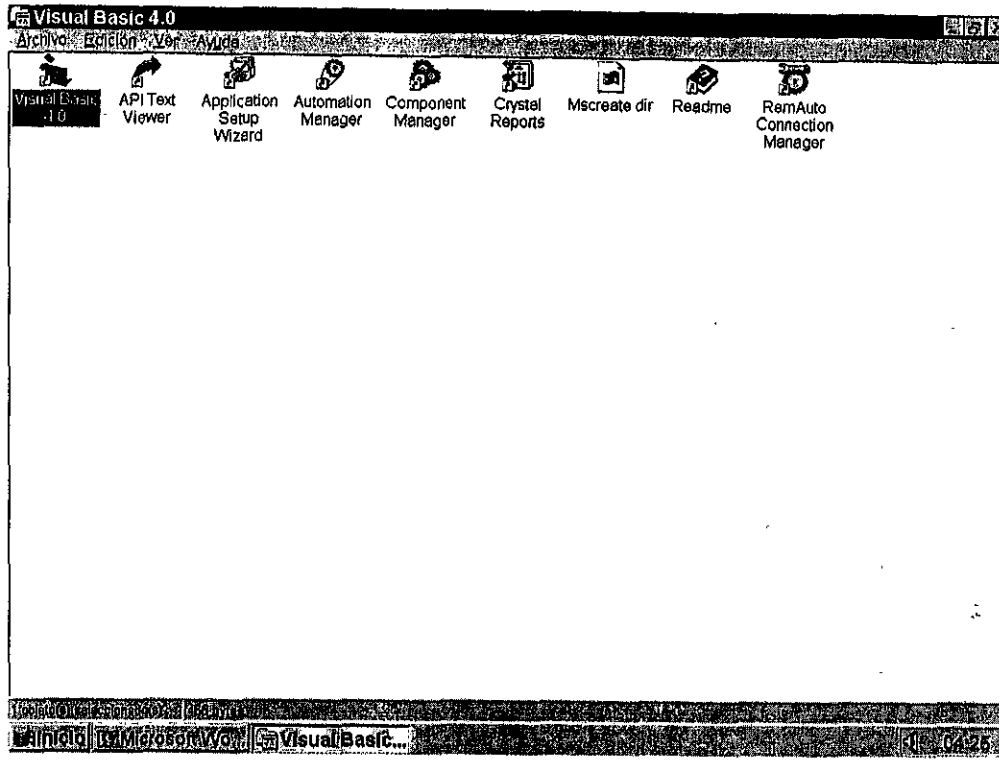


FIG. 7.1. Arranque de Visual Basic desde el icono de Visual Basic.

Entorno de Visual Basic.

Después de arrancar Visual Basic, aparecen las cinco ventanas principales de Visual Basic, las cuales se mencionan a continuación y se presentan en la FIG. 7.2:

1. Ventana Principal.
2. Ventana de Forma.
3. Caja de Herramientas.
4. Ventana de Proyecto.
5. Ventana de Propiedades.

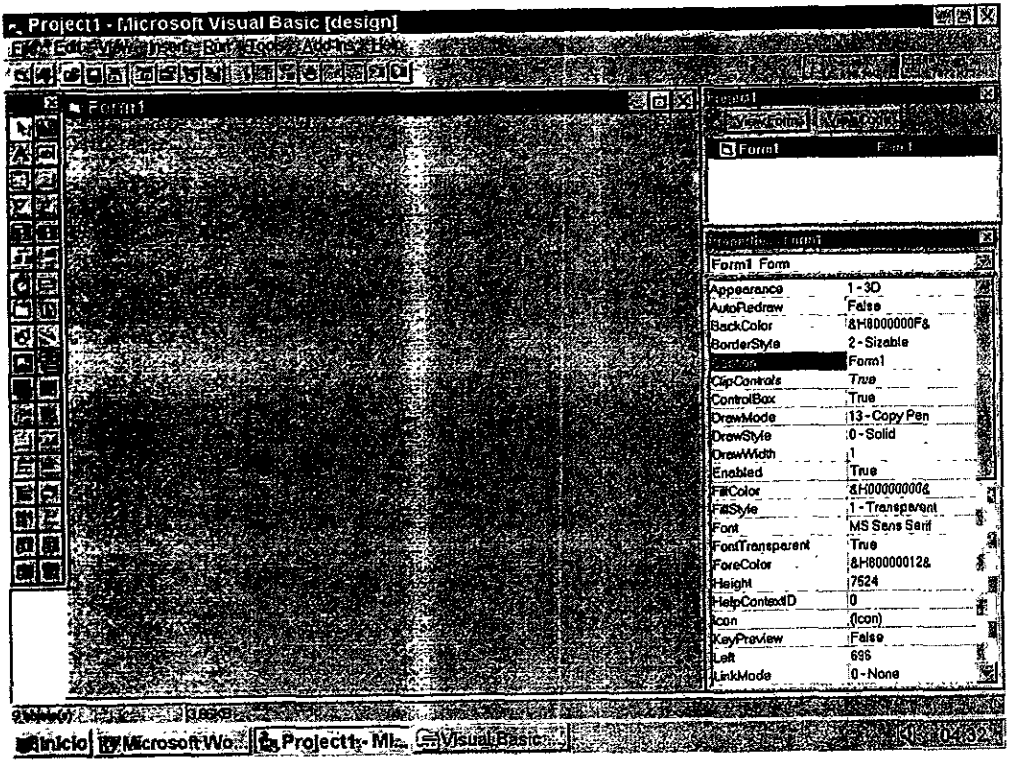


FIG. 7.2. Las cinco ventanas principales de Visual Basic.

1. Ventana Principal.

Se encuentra en la parte superior de la pantalla y contiene los menús de Visual Basic:

File (Archivo)	Crea, abre, guarda, agrega, borra e imprime proyectos y formas, crea también archivos ejecutables.
Edit (Edición)	Deshace y reace acciones, copia, pega y corta la selección hecha, busca y remplaza, entre otros comandos.
View (Vista)	Contiene las opciones para activar o desactivar las diferentes ventanas y la barra de herramientas de la ventana principal.
Insert (Insertar)	Sirve para insertar en el proyecto procedimientos, formas, módulos, etc.

Run (Ejecutar)	Contiene todas las opciones del depurador de Visual Basic.
Tools (Herramientas)	Este menú contiene opciones para escribir expresiones de Visual Basic que se evaluarán en cada procedimiento y/o módulo del proyecto cada vez que se ejecute el código de éstos. También contiene el editor de menús de Visual Basic y opciones personalizar el entorno, el proyecto y el editor de Visual Basic.
Add-Ins (Agregar-Insertar)	En este menú puede invocarse al administrador de bases de datos de Visual basic y al diseñador de reportes (Crystal reports).
Help (Ayuda)	Da acceso a toda la ayuda disponible de Visual Basic, para el entorno, herramientas y el lenguaje de programación.

La barra de título de la ventana principal tendrá el nombre del proyecto que esté abierto en ese momento, además Visual Basic permite saber mediante ésta si se encuentra diseñando o ejecutando una aplicación. De esta manera, aparecerá **Microsoft Visual Basic [design]** en el caso de que esté en el modo de diseño o **Microsoft Visual Basic [run]** si se está en el modo de ejecución.

La ventana principal, además contiene la barra de herramientas de Visual Basic, la cual es personalizable y está conformada por botones. Estos botones son atajos para los comandos que se usan con más frecuencia.

La Barra de Herramientas se encarga del diséño, ejecución y depuración de aplicaciones principalmente. Además esta barra permite activar las tareas más comunes sin la necesidad de utilizar los menús. Dado que cada elemento de la Barra de Herramientas tiene también una combinación de teclado para la misma tarea, la elección de un sistema u otro para activarla es cuestión de comodidad. La barra de herramientas de la ventana principal se muestra en la FIG. 7.3.

Las tareas que corresponden a estos botones son:

Botón New Form.	Permite crear una nueva forma.
Botón New Module.	Permite crear un nuevo módulo.
Botón Open Project.	Permite abrir un proyecto existente.
Botón Save Project.	Permite guardar el proyecto actual.

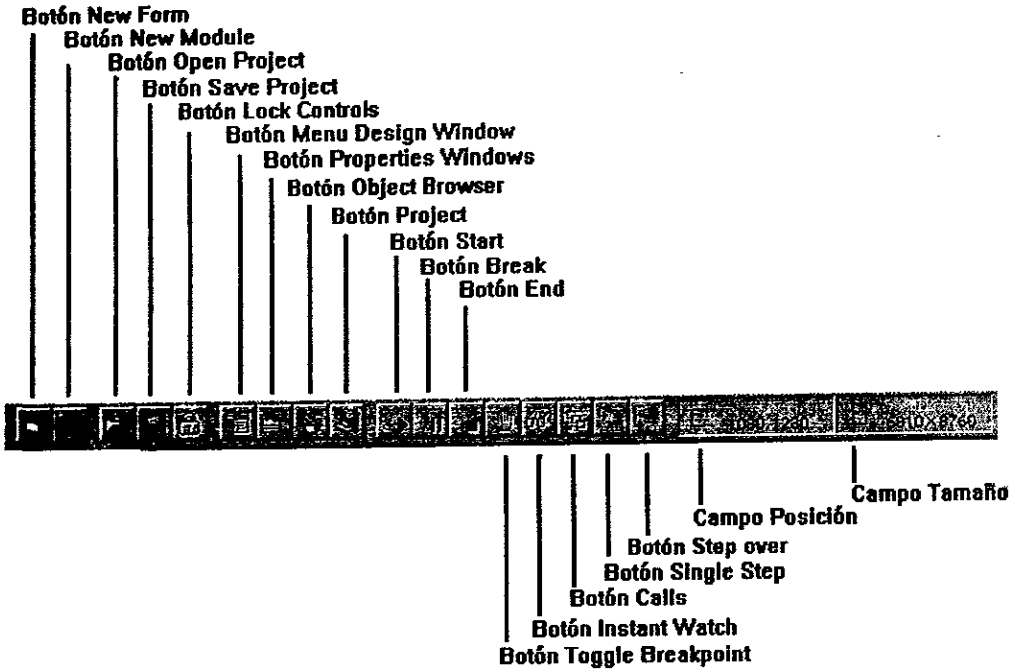


FIG. 7.3 La Barra de Herramientas de la ventana principal.

Botón Lock Controls.	Bloquea y desbloquea los controles en la forma activa.
Botón Menu Design Window.	Permite visualizar la Ventana de Diseño de Menús.
Botón Properties Window.	Permite visualizar la Ventana de Propiedades.
Botón Object Browser.	Despliega las clases, propiedades y métodos disponibles en las librerías de objetos y módulos y procedimientos en el proyecto activo.
Botón Project.	Abre la ventana Project.
Botón Start Button.	Permite ejecutar la aplicación en modo de diseño.
Botón Break.	Permite hacer una pausa durante la ejecución de la aplicación.

Botón End.	Permite detener la ejecución y volver al modo de diseño.
Botón Toggle Breakpoint.	Permite introducir una pausa en la línea actual.
Botón Instant Watch.	Permite visualizar el valor del elemento seleccionado en la ventana de código.
Botón Calls.	Permite visualizar la estructura de las llamadas activas.
Botón Single Step.	Permite ejecutar sentencias paso a paso.
Botón Step Over.	Permite ejecutar un procedimiento o una sentencia cada vez.

Por último a la derecha de la Barra de Herramientas en la Ventana Principal se encuentran dos campos que sirven para indicar la posición y el tamaño del objeto que se está seleccionando en ese momento en la Ventana de Forma.

2. Ventana de Forma.

Se localiza en el centro de la ventana principal, y es una ventana que muestra una forma en blanco lista para diseñarse con el título de form1.

La Ventana de Forma ocupa la mayor parte del centro de la pantalla. En ella es donde se personaliza y diseña la forma que verán los usuarios; es decir, la interface de usuario para el proyecto o la parte del proyecto que el usuario ve y con la que interactúa.

Cuando se inicia un proyecto nuevo, Visual Basic crea una forma vacía con el título de Form1, en ella se dibujan los objetos o controles que forman parte de la forma.

La apariencia de la forma se diseña eligiendo controles de la Caja de Herramientas y colocándolos en ella. Esto es, con ayuda del cursor, se selecciona el control que se quiere agregar a la interface, se arrastra a la forma, y por último se coloca en el lugar adecuado.

3. Caja de Herramientas.

Se localiza en la parte izquierda de la ventana principal y contiene todos los controles disponibles en Visual Basic.

La Caja de Herramientas contiene herramientas básicas para desarrollar aplicaciones, éstas se pueden utilizar para situar botones de comandos, botones de texto y otros controles en las aplicaciones. Cada herramienta de la caja crea un único control, la caja de herramientas se presenta en la FIG. 7.4.

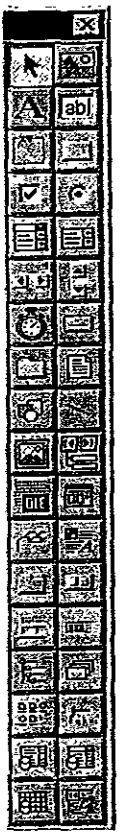


FIG. 7.4 La ventana de la Caja de Herramientas.

Los controles de la Caja de Herramientas de izquierda a derecha y de arriba hacia abajo son:

Pointer o puntero.

Se utiliza para manipular los controles que se tienen en la forma. Con el puntero se puede seleccionar, mover y ajustar el tamaño de los controles.

Picture Box o cuadro de dibujo.

Se utiliza para visualizar una imagen que se dibujó utilizando código o que se importó de algún archivo.

Label o etiqueta.

Se utiliza para crear texto que no pueda ser modificado por el usuario, y tiene la finalidad de informar al usuario sobre lo que puede hacer y sobre la función que tiene cada control.

Text Box o cuadro de texto.

Se utiliza para crear áreas dentro de la forma en las que el usuario pueda escribir o visualizar texto.

Frame o marco.

Se utiliza para realzar el aspecto de la forma. También se utiliza para agrupar objetos relacionados entre sí.

Command Button o botón de comando.

Se utiliza para crear botones que tienen asociado un comando u orden. Esta orden se ejecuta cuando el usuario hace *click* sobre el botón.

Check Box o cuadro de comprobación.

Permite crear un cuadro de comprobación que utiliza el usuario para seleccionar una opción. De esta manera se pueden seleccionar varias opciones de un grupo.

Option Button o botón de opción.

Permite crear botones de opción que el usuario utiliza para seleccionar una opción entre varias. Así, sólo se puede seleccionar una opción de un grupo de ellas.

Combo Box o cuadro combinado.

Combina las características de una caja de texto y de una lista. Esto permite al usuario elegir un elemento de varios, escribiéndolo directamente en la caja de texto o seleccionándolo de la lista.

List Box o cuadro de lista.

Este control crea cuadros de lista que ponen a disposición del usuario un conjunto de elementos, de los cuales elegirá uno.

Horizontal Scroll Bar (barra de desplazamiento horizontal) y Vertical Scroll Bar (barra de desplazamiento vertical).

Permiten crear barras de desplazamiento que a menudo son utilizadas en cajas de texto y ventanas para desplazar información hacia abajo o hacia arriba de la ventana, o hacia la izquierda o hacia la derecha de ventana. Pero también pueden utilizarse como controles de ventana. Una barra de desplazamiento representa un valor entero. Cada barra de desplazamiento tiene un botón que se desplaza a lo largo de la misma. La posición inicial corresponde a un valor mínimo, la posición final corresponde a un valor máximo y cualquier otra posición es un valor intermedio.

Timer o reloj.

Permite activar procesos a intervalos regulares de tiempo.

Drive List Box o cuadro de lista de unidades.

Permite crear cuadros de lista de unidades que se utilizan para visualizar la lista de unidades de disco disponibles. Esto, con el fin de seleccionar alguna.

Directory List Box o cuadro de lista de directorio.

Permite crear cuadros de lista de directorios que se utilizan para visualizar los directorios que el usuario puede accesar.

File List Box o cuadro de lista de archivos.

Permite crear cuadros de lista de archivos que se utilizan para visualizar los archivos de un determinado directorio que el usuario puede tener acceso.

Shape o figura.

Se utiliza para añadir rectángulos, cuadros, elipses o círculos a una forma.

Line o linea.

Se utiliza para añadir líneas rectas a una forma.

Image o imagen.

Se utiliza cuando se requiere visualizar una imagen que se dibujó utilizando código o que se importó de algún archivo. Se diferencia de la herramienta de cuadro de dibujo principalmente en la forma de la presentación.

Data o datos.

Permite conectarse a una base de datos existente y visualizar su información en la forma.

OLE.

Permite incrustar datos en una aplicación.

Commond Dialog o diálogo común.

Permite utilizar cajas de diálogo estándar, tales como: abrir, guardar como, color, fuentes e imprimir.

Sstab.

Es un control que permite presentar varios diálogos o pantallas de información de manera sencilla en una misma área de la pantalla.

RichTextBox.

Es un control que permite al usuario capturar y editar texto, también provee de avanzadas opciones para formatos de texto.

Tabstrip.

Es un control similar a las secciones de un libro o las etiquetas de un grupo de folders. Se pueden definir múltiples páginas para la misma área de la ventana en la aplicación.

ToolBar o barra de herramientas.

Es un control que contiene una colección de botones usados para crear la barra de herramientas asociada con la aplicación.

StatusBar o barra de estado.

Es un control que se coloca usualmente en la parte de abajo de una forma y muestra varios estados de la aplicación.

Progress Bar o barra de progreso.

Es un control que muestra el progreso de una operación o de un proceso en un rectángulo que se va llenando conforme avanza el proceso.

TreeView.

Despliega una lista jerárquica de nodos, cada uno de los cuáles consiste en una etiqueta y un mapa de bits opcional.

ImageList.

Sirve para poder referenciar imágenes por su índice o una tecla.

ListView.

Despliega los componentes de una lista de objetos utilizando una de cuatro diferentes vistas, puede desplegarse cada elemento de la lista con un ícono y texto.

Slider.

Este control despliega una ventana en la cual pueden seleccionarse valores dentro de un rango, este slider puede arrastrarse a lo largo de los diferentes valores.

DBlist y Dbcombo.

Estos controles difieren de los controles listbox y combobox ya que se llenan automáticamente y pueden accesar directamente a un elemento de la lista sin código adicional.

Dbgrid o rejilla.

Despliega y habilita la manipulación de una serie de renglones y columnas que representan registros y campos del objeto RecordSet de Visual Basic, el cual contiene los registros de una tabla o el resultado de un query.

MSRDC.

Provee el acceso a datos almacenados en forma remota a través de un ODBC.

4. Ventana de proyecto.

En ésta ventana, con el título del proyecto activo, se encuentran en forma de lista todos los módulos y formas que componen el proyecto.

La Ventana de Proyecto es una forma de organizar los archivos necesarios para ejecutar la aplicación de Visual Basic que se desarrolla; es decir, contiene un listado de todas las formas y módulos contenidos en la aplicación; ya que es muy común que las aplicaciones de Visual Basic compartan código o formas personalizadas. Cada proyecto (aplicación) puede tener varias formas, y el código que activa los controles de una forma es archivado con ésta en archivos separados. El código general de programación compartido por todas las formas de una aplicación puede ser dividido en varios módulos (código general), que también se archivan separadamente.

La Ventana de Proyecto contiene dos botones: View Form (para ver la forma) y View Code (para ver el código y formatearlo).

Por omisión, siempre se mostrará la forma correspondiente cuando se seleccione un archivo en la Ventana de Proyecto.

5. Ventana de Propiedades.

Se localiza debajo de la ventana de proyecto y es en la cual se definen todas las propiedades de las formas y controles usados en el proyecto.

La Ventana de Propiedades permite modificar los valores de una propiedad como podrían ser el tamaño y la posición de una forma. Sin embargo, algunas propiedades se encuentran restringidas a ciertos valores como por ejemplo, el que un objeto sea visible o no, en este caso, sólo puede ser ejecutado como verdadero o falso. La ventana de propiedades se muestra en la FIG. 7.5.

El cuadro de lista desplegable que se encuentra en la parte superior de la Ventana de Propiedades se llama cuadro Object (objetos). Este muestra el nombre de todos los objetos de la aplicación y sus tipos. Debajo del cuadro Object se encuentran el cuadro Settings (ajustes) y la lista de Propiedades. Esta lista permite desplazarse por todas las propiedades del objeto que se muestran en el Cuadro Object y ver el valor actual de cada propiedad. Si se quiere cambiar el valor, se escribe un dato nuevo en el Cuadro Settings o se elige un nuevo valor predefinido de una lista desplegable, dependiendo de la propiedad a cambiar.

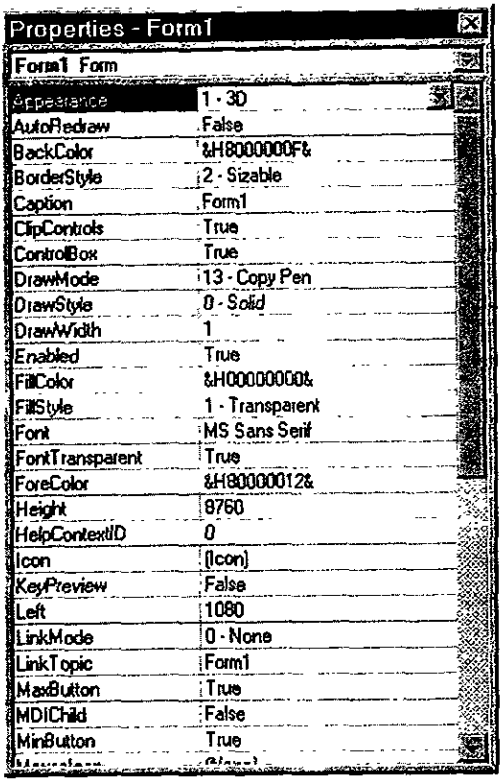


FIG. 7.5. La Ventana de Propiedades.

Diseño de una aplicación para Windows.

El lenguaje de Visual Basic incluye un conjunto de características que lo hacen un lenguaje de programación de fácil manejo para el desarrollo de aplicaciones Windows.

Las líneas de programa se ejecutarán cuando se active un evento (suceso). Un evento se define como una acción que activará otra, por ejemplo: un doble *click* del mouse sobre un botón o la selección de alguna opción.

Para el manejo de archivos, se tiene la posibilidad de crear, salvar y borrar archivos. El acceso a los archivos puede ser de manera secuencial (para archivos de tipo texto), de manera aleatoria (para archivos con registros de tamaño idéntico) o con acceso binario (para archivos con registros de diferente tamaño).

Visual Basic permite generar archivos ejecutables, lo que proporciona la comodidad de tener archivos portables. También, cuenta con una ayuda en línea, lo que facilita su uso, y por consecuencia, su aprendizaje.

En Visual Basic, es posible agregar a una aplicación una ayuda en linea como en muchas aplicaciones para Windows.

Para esto, es necesario tener instalada la versión profesional de Visual Basic, que proporciona el Help Compiler (compilador de ayuda) y adicionalmente se necesita un procesador de textos que soporte archivos con extensión .rtf (Rich Text Format).

Dadas las características del lenguaje, los pasos necesarios a seguir para desarrollar una aplicación en Visual Basic son:

1. **Definir el problema.** Realizar un diseño general de la aplicación que se quiere desarrollar. Esta es una de las partes más importantes en la creación de una nueva aplicación.
2. **Dibujar la interface.** Crear una nueva aplicación, mover y ajustar el tamaño por omisión de la forma y dibujar los controles necesarios, basándose en el diseño general.
3. **Designar propiedades.** Se definen las propiedades de la forma y de los controles que formarán la forma. Esto se hace en la Ventana de Propiedades.
4. **Escribir el código.** Se escribe el código para cada uno de los objetos, uniéndolo a la forma y a los controles, para que la aplicación ejecute las tareas que se definieron.
5. **Crear el programa ejecutable.** Se salva la aplicación y se verifica ésta, empleando las opciones de depuración de errores. Por último se crea el archivo ejecutable.

1. Definir el problema.

La base esencial de la programación en Visual Basic son los objetos (formas y controles), que permiten diseñar sin programar y su interface gráfica de usuario. Para construir una aplicación que incluya varias formas, controles, archivos, menús e incluso algunos gráficos, será necesario, como primer paso, definir el problema, haciendo un diseño general de la aplicación a desarrollar.

El diseño de diagramas antes de diseñar un programa es muy importante, sobre todo en el caso de procedimientos no detallados; ya que permite concretar las ideas vagas en pasos bien definidos. También es más fácil identificar puntos que se han pasado por alto o que están incompletos, por ejemplo, se pueden tener procesos comunes que puedan compartir código. Si el diseño es de un nivel alto, se puede implementar cada parte del diseño sin definir su función, e implementar los detalles una vez que se tiene la estructura del programa principal.

Dentro del diseño general, es importante definir las partes que se utilizarán más frecuentemente, así como el diseño de las opciones del menú de la aplicación. El trabajar en la estructura general del programa antes de comenzar a escribir el código permitirá ahorrar tiempo; ya que de esta manera, es posible conocer qué procedimientos se utilizan en más de una ocasión a lo largo del programa, y así, únicamente escribir el procedimiento una sola vez.

2. Dibujar la interface.

Para diseñar la interface, primero se crea una nueva aplicación, ajustando el tamaño por omisión de la forma.

a) Se crea una forma. Para crear una nueva forma se ejecuta el comando New Form (Nueva Forma) del menú File.
b) Se añaden controles a la forma. Para dibujar los controles se utiliza la Caja de Herramientas de Visual Basic.

También es posible incluir cualquier imagen prediseñada o una dada por un ícono de Visual Basic. Además, se pueden añadir menús desplegables para facilitar al usuario el acceso a una amplia variedad de comandos seleccionables. La interface de usuario se diseña en la Ventana de Formas agregando los controles necesarios. Todo esto, de acuerdo con las características que se requieren para la aplicación. Una vez que se tienen los controles en la forma, se establecen las propiedades de la forma y de los controles. De esta manera, el usuario podrá interactuar con la aplicación.

El diseño de menús adecuados hace que las aplicaciones sean mucho más amigables al usuario. Por ejemplo, en Windows se aprecian los menús de despliegue, donde se colocan opciones relativas al manejo del sistema sin que interfieran con el espacio visible que se tiene. Para diseñar este tipo de menús, se emplea la Ventana del Editor de Menús que permite construir hasta seis niveles de menús y añadir menús desplegables. La ventana del Editor de Menús se muestra en la FIG. 7.6.

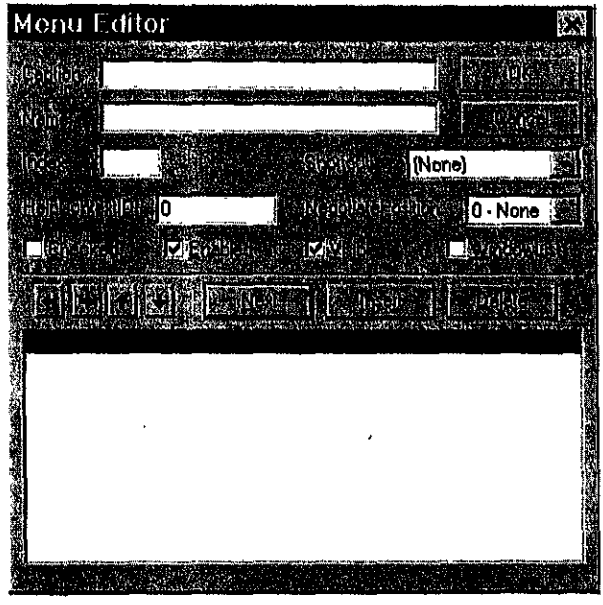


FIG. 7.6 Ventana del Editor de Menús.

3. Designar propiedades.

Una vez terminados la interface, formas y controles necesarios, se especifican las propiedades para los objetos creados. Es decir, se definirán las propiedades de la forma y controles. Esto se hace con la ayuda de la Ventana de Propiedades.

4. Escribir el código.

Para añadir código a un programa se selecciona el comando New Module (Nuevo Módulo) del menú File de Visual Basic. El módulo contendrá las declaraciones globales y las rutinas compartidas por distintas formas que compondrán la aplicación. En el código se declaran las variables que usará el programa. El Editor de Código es una ventana que se abre al efectuar un doble click en una forma. Aquí es donde se introduce el código que indica a Visual Basic cómo responder a un suceso. En la parte superior de la ventana se encuentran dos cuadros uno de Object (objeto) y otro de Proc (procedimiento). El de Object lista los objetos, mientras que el de Proc lista todos los procedimientos o sucesos que puede reconocer el objeto que se encuentra en el cuadro Object. El editor de código se muestra en la FIG. 7.7.

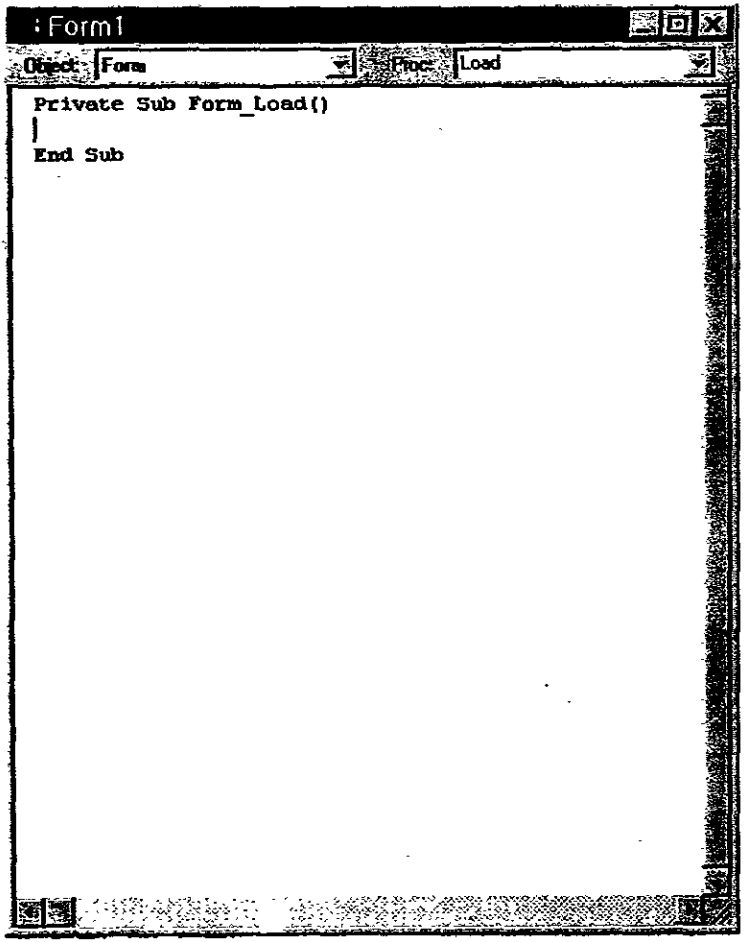


FIG. 7.7 El Editor de Código (Ventana de código).

Una vez que se ha terminado el código, es necesario unir el código correspondiente a cada una de las formas y controles, para que la aplicación responda a las acciones del usuario. Bajo Windows, una aplicación es conducida por eventos y orientada por objetos. Esto significa que se liga el código escrito para un objeto determinado a un evento que puede ocurrir sobre un objeto, de tal forma, que cuando ocurra el evento, el código se ejecute. Al código asociado con cada control para un determinado evento, se le conoce como un procedimiento manejado por un evento (código invocado cuando un objeto reconoce que ha ocurrido un evento determinado).

Las formas y los diferentes tipos de controles reconocen tres eventos producidos por el ratón, estos son: MouseDown, MouseUp y MouseMove.

MouseDown.	Cuando el usuario pulsa cualquier botón.
MouseUp.	Cuando el usuario suelta cualquier botón.
MouseMove.	Cada vez que el usuario mueve el puntero del ratón a una nueva posición.

Es necesario aclarar que una forma reconocerá un evento del ratón cuando el puntero del mismo esté en una zona en la que no hay ningún control; y un control reconocerá un evento del ratón cuando el puntero del ratón esté sobre el propio control. Por ejemplo, una forma de emitir una orden o comando es pulsando un botón. Es decir, esta acción invocará al procedimiento conducido por el evento *click* de botón pulsado.

5. Crear el programa ejecutable.

Una vez que la aplicación ha sido verificada, que tiene el aspecto deseable, y que su ejecución transcurre satisfactoriamente, se puede crear un archivo ejecutable que permita correr la aplicación fuera del entorno de Visual Basic.

Para guardar la aplicación como un archivo ejecutable, se ejecuta el comando **Make EXE File** (Crear el archivo ejecutable) del menú **File**. Un archivo como este requerirá de Windows y de un archivo **VBRUN???.DLL** para poder ejecutarse.

Si el archivo .EXE se guarda en otro directorio distinto de Visual Basic, será necesario también guardar una copia del archivo VBRUN???.DLL en el mismo directorio. Este archivo se encuentra en el directorio windows/system.

En un proyecto que contenga varias formas o en un proyecto en el que se deba producir una inicialización antes de que se muestre una forma en pantalla, es necesario que el programa de Visual Basic comience ejecutando un procedimiento específico, en vez de sólo mostrar la primera forma. En este caso se deberá dar al procedimiento el nombre de **Main** (Principal) y colocarlo en un módulo.

En Visual Basic se cuenta con herramientas que permiten analizar como se desarrolla la ejecución de la aplicación. Esto se lleva a cabo mediante la depuración de errores.

Depuración de errores.

Durante el diseño y ejecución de una aplicación se tienen tres modos de trabajo, estos modos son: Design, Run o Break.

El modo de **Design** (diseño) se caracteriza por ser el modo en el que se realiza la mayor parte del trabajo para crear una aplicación. En este modo se diseñan las formas,

se dibujan controles, se escribe código; se utiliza la Ventana de Propiedades para ver o modificar las propiedades de los objetos, pero no se puede ejecutar el código.

El modo **Run** (ejecución) se caracteriza por ser el modo donde la aplicación toma el control. En este modo se puede ver el código, pero no se puede modificar.

El modo **Break** (pausa) se caracteriza por ser el modo que permite ejecutar la aplicación paso a paso. En este modo se puede editar código, examinar variables y propiedades, avanzar un paso más en la ejecución, ejecutar la aplicación hasta el final, iniciar la ejecución, o finalizar la misma.

En la barra de título de Visual Basic se puede observar el modo en el que se encuentra en ese momento.

En Visual Basic se cuenta con utilerías que permiten analizar cómo se desarrolla la ejecución de la aplicación desde un lugar a otro. Esto es una manera de determinar qué ocurre en una aplicación que aparentemente parece correcta, pero donde los resultados mostrados no son satisfactorios. Las técnicas de depuración empleadas por Visual Basic incluyen, puntos de parada, ejecución paso a paso, y la posibilidad de visualizar valores de variables y de propiedades.

Para comenzar a depurar una aplicación, es necesario estar en el modo de pausa, y se emplean los botones de la Barra de Herramientas.

El Botón Start (ejecutar).

Sirve para ejecutar la aplicación en modo de diseño.

El Botón Break (detener).

Sirve para hacer una pausa durante la ejecución de aplicación.

El Botón End (fin).

Sirve para detener la ejecución y volver al modo de diseño.

El Botón Procedure Step (saltar procedimiento).

Sirve para introducir una pausa en la línea actual.

El Botón Single Step (paso a paso).

Sirve para visualizar el valor del elemento seleccionado en la Ventana de Código.

El Botón Calls (llamar).

Sirve para visualizar la estructura de las llamadas activas.

El Botón Instant Watch (punto de visualización).

Sirve para ejecutar una sentencia o comando cada vez (ejecución paso a paso).

El Botón Toggle Breakpoint (activar/desactivar punto de interrupción).

Sirve para ejecutar un procedimiento o una sentencia a la vez.

La FIG 7.8 muestra los botones de la Barra de herramientas del depurador.

Cuando se crea una aplicación, se pueden presentar tres clases de errores: errores de sintaxis, errores en tiempo de ejecución y errores lógicos.

Errores de sintaxis.

Los errores de sintaxis son el resultado de escribir incorrectamente una sentencia. Si se tiene activa la opción **Syntax Checking** (orden Environment del menú Options), Visual Basic detectará estos errores tan pronto como se produzcan; esto es, en el momento de escribir una sentencia incorrecta en la Ventana de Código.

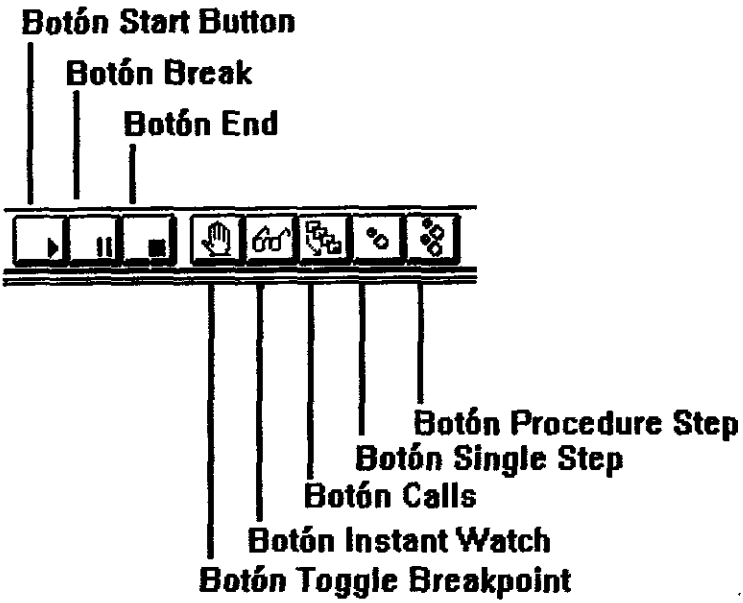


FIG. 7.8 Botones de la Barra de herramientas del depurador.

Errores en tiempo de ejecución.

Los errores en tiempo de ejecución ocurren durante la ejecución de una aplicación, una sentencia intenta una operación que es imposible realizar. Un ejemplo típico de esto es la división por cero. Una buena programación exige que ésta se anticipe a muchos de estos errores, manipulándolos adecuadamente.

Errores lógicos.

Los errores lógicos que se producen cuando la aplicación es sintácticamente correcta, ejecutándose ésta sin producirse errores en tiempo de ejecución, pero los resultados que se obtienen no son los correctos. Para detectar este tipo de errores, es necesaria una minuciosa depuración, analizando cada parte de código y verificando resultados inmediatos.

VENTAJAS Y DESVENTAJAS DE VISUAL BASIC 4.0.

Trabajar con Visual Basic 4.0 representa diversas ventajas, además de las características antes mencionadas listaremos algunas más.

Ventajas:

- Es un sistema productivo para crear soluciones en Windows.
- Controles Visuales preconstruidos por terceros.
- Acceso a Bases de Datos Relacionales y de otros fabricantes a través de ODBC.
- Permite ensamblar fácil y rápidamente una interface de usuario con componentes prefabricados.
- Soporta diversos tipos de Manejador de Bases de Datos.
- Interface al usuario amigable.
- Manejo de ayudas en linea robustas.
- Aplicaciones escritas que toman ventaja de grandes arreglos y un espacio limitado en strings (cadenas).
- Visual Basic incluye el mismo motor (Jet Engine) de la Base de Datos Access, la cual provee acceso simultáneo con Foxpro, dBASE, Paradox, SQL, etc.
- Un usuario inexperto puede ser productivo con algunas semanas de utilización de Visual Basic.
- Puede ser utilizado como un servidor de OLE Automation.

Desventajas:

- Visual Basic es poco flexible en el sentido de que se tiene que auxiliar de lenguajes mas robustos como el C o Pascal para hacer por ejemplo manipulación de memoria, accesos directos al Sistema Operativo, etc.
- Su código no es compatible con plataformas no Windows.

OPERADORES ARITMETICOS.

Los operadores más familiares son los que llevan a cabo operaciones aritméticas simples. La TABLA 7.3 muestra una lista de los operadores aritméticos de los que dispone Visual Basic.

Operador	Operación
+	Suma
-	Resta
*	Multiplicación
^	Exponenciación
/	División en coma flotante
\	División entera
Mod	Módulo

TABLA 7.3. Operadores aritméticos.

OPERADORES RELACIONALES.

Los seis operadores relacionales aparecen en la TABLA 7.4. Cuando se comparan dos valores, el resultado de la operación es un valor booleano o lo que es lo mismo, verdadero o falso. Visual Basic dispone de las constantes internas llamadas True o False que representan los valores enteros -1 y 0 respectivamente.

Operador	Comparación
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
=	Igual que
<>	Distinto que

TABLA 7.4. Operadores relacionales.

OPERADORES LOGICOS.

Estos operadores tienen valores booleanos como operandos, y devuelven valores booleanos, la TABLA 7.5 muestra la lista con los operandos lógicos en su orden de precedencia.

Operador	Operación
Not	No lógico
And	Y lógico
Or	O lógico
Xor	O exclusivo
Eqv	Equivalencia lógica
Imp	Implicación lógica

TABLA 7.5. Los operadores lógicos.

SINTAXIS PARA PROCEDIMIENTOS Y FUNCIONES.

Para llamar a un procedimiento o función se debe utilizar su nombre, utilizaremos el término procedimiento para referirnos a los servicios que no devuelven un valor, y el término función a aquellos que sí lo devuelven. Visual Basic dispone de muchas funciones internas que pueden usarse sin necesidad de definirlas. Algunas de estas funciones se describen en la TABLA 7.6.

Función	Valor devuelto
Abs	El valor absoluto de un número
Asc	El código ASCII ó ANSI de un carácter
Chr$	El carácter correspondiente a un código ASCII ó ANSI dado
Cos	El coseno de un ángulo
CurDir$	El nombre del directorio de trabajo actual
Date$	La fecha actual como una cadena de texto
Format	Una fecha o un número con formato convertido en una cadena de texto.
InputBox	El texto introducido en un cuadro de diálogo por un usuario
Len	El número de caracteres de una cadena de texto
Mid$	Una porción seleccionada de una cadena de texto
Now	La fecha y hora actuales
Rnd	Un número aleatorio
Sin	El seno de un ángulo
Sqr	La raíz cuadrada de un número
Str$	Un número convertido en una cadena de texto
Time$	La hora actual como una cadena de texto
Val	El valor numérico de una cadena de texto dada

TABLA 7.6. Algunas funciones internas de Visual Basic.

SENTENCIAS.

Las sentencias Remark (o Rem) se ponen en los programas para explicar lo que hace el código. No son ni ejecutadas ni procesadas por Visual Basic. En consecuencia, no ocupan espacio dentro del código compilado.

Hay dos formas de indicar una sentencia Remark. La primera es empleando la palabra clave Rem:

```
Sub Command1_Click()
  Rem Los comentarios describen lo que el procedimiento hace aquí
End Sub
```

El segundo método de indicar una sentencia Remark es mediante el empleo del apóstrofo (').

```
Sub Command1_Click()
  ' Los comentarios describen lo que el procedimiento hace aquí
End Sub
```

Sentencia end.

Cuando Visual Basic procesa una sentencia End el programa se para. Si se está ejecutando un programa, se vuelve al entorno de desarrollo.

Sentencia If.

En el código de un programa se crea lo que se denomina una ramificación condicional, utilizando la declaración If. La sintaxis de la declaración If en Visual Basic es:

```
If expresión booleana Then
        [declaración]...
[Else
        [declaración]...]
End If
```

Sentencia Do.

La declaración Do de Visual Basic puede procesar repetidas veces un conjunto de instrucciones, la sintaxis es la siguiente:

```
Do
        [declaración]...
Loop
```

La palabra clave Do marca el comienzo de una declaración compuesta, la cual incluye todas las declaraciones que haya hasta la palabra clave Loop. Las instrucciones se ejecutan en orden hasta que se llega a la palabra clave Loop; en ese punto la ejecución comienza de nuevo al inicio del bloque que contiene la declaración Do.

El conjunto de instrucciones que se repiten se denominan un bucle. Cuando un programa ejecuta repetidamente esas declaraciones, se dice que está iterando o haciendo un bucle.

Se puede añadir una cláusula While a la declaración Do o a la declaración Loop. La palabra clave While debe ir seguida de una condición en forma de expresión booleana. Si la expresión es False, se saltan las instrucciones que vienen a continuación, hasta la palabra Loop, y la ejecución continúa en las instrucciones que van detrás de la palabra Loop.

La palabra clave While puede asociarse con la palabra Loop en lugar de hacerlo con Do, y lograr un efecto ligeramente distinto.

Poniendo la cláusula While al final del bucle se asegura que las instrucciones del interior del mismo se ejecutarán al menos una vez, debido a que la prueba que se realiza para determinar su finalización viene detrás de la palabra clave Loop.

Sentencia For.

La sentencia For es una declaración especializada de Visual Basic que gestiona bucles que tienen un contador que aumenta o decrece regularmente. Tiene la siguiente sintaxis:

```
For variable = primervalor To ultimovalor [Step incremento]
        [declaración]...
Next variable
```

7.3.2 MODULO MUESTRA DEL SISTEMA.

Se toma como muestra uno de los módulos del sistema.

Módulo Catálogos.

Este se encuentra en el Menú Servicios del Menú Principal del SIAI. La forma que se presenta al ejecutar este módulo y que tiene todos los controles involucrados es frmcatalogos y se presenta en la FIG. 7.9.

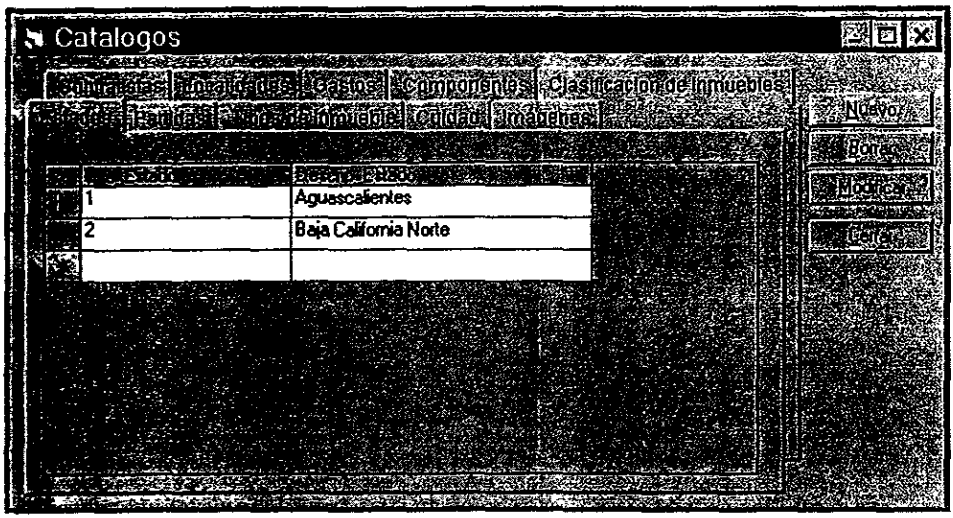


FIG. 7.9. Forma del módulo catálogos.

La forma es frmcatalogos.frm y contiene los siguiente controles:

- Un control SSTab.

  SSTab1 Control SSTab con un título para cada catálogo.

- Un control DBGrid (rejilla) para cada catálogo.

  grdCiudad Control DBGrid para catálogo de ciudades.
  grdClasInm Control DBGrid para catálogo de clasificación de inmuebles.
  grdComponente Control DBGrid para catálogo de componentes.
  grdContratista Control DBGrid para catálogo de contratistas.
  grdEstado Control DBGrid para catálogo de estados.

grdGasto	Control DBGrid para catálogo de gastos de inmuebles.
grdLocalidad	Control DBGrid para catálogo de localidades.
grdTinm	Control DBGrid para catálogo de tipos de inmueble.

- Dos controles TreeView.

TvwImagen	Control TreeView para catálogo de imágenes de los inmuebles.
TvwPartida	Control TreeView para catálogo de partidas.

- Cuatro controles CommandButton.

CmdNuevo	Control CommandButton para agregar un registro.
CmdBorrar	Control CommandButton para borrar un registro.
CmdModificar	Control CommandButton para modificar un registro.
CmdCerrar	Control CommandButton para cerrar la forma.

A continuación se presenta el código fuente de cada objeto de la forma con sus procedimientos para cada evento.

Objeto: (General) **Procedimiento: (Declarations)**

```
Option Explicit
Dim queForma As Integer
```

Objeto: (General) **Procedimiento: Borra_Reg**

```
Function Borra_Reg(ByVal iClave, sCampo, sTabla) As Boolean
    Dim rpta As Boolean
    Dim sQuery As String

    On Error GoTo HndError

    sQuery = DELETE_STR & FROM_STR & sTabla & WHERE_STR & vbLf
    sQuery = sQuery & sCampo & "=" & iClave

    rpta = False
    If SQLEXEC(sQuery) = NULL_INTEGER Then
        rpta = True
    End If

    Borra_Reg = rpta
    Exit Function

HndError:
    MsgBox Err.Description
    Borra_Reg = rpta
End Function
```

Objeto: CmdBorrar **Procedimiento:** Click

```
Private Sub cmdBorrar_Click()
    Dim rpta As Integer
    Dim process As Boolean

    rpta = Confirma_Eliminar

    Select Case rpta
    Case 6 'se elimina
        Select Case queForma
        Case 0
            process=Borra_Reg(grdContratista.Columns(0).Text,    "cve_Contrat",
"Contratista")
        Case 1
            process=Borra_Reg(grdLocalidad.Columns(0).Text,    "cve_localidad",
"Localidad")
        Case 2
            process=Borra_Reg(grdGasto.Columns(0).Text, "cve_gasto", "Gasto")
        Case 3
            process=Borra_Reg(grdComponente.Columns(0).Text,    "cve_comp_gen",
"Componente")
        Case 4
            process=Borra_Reg(grdClasInm.Columns(0).Text,          "cve_Clasif",
"Clasificacion_Inmueble")
        Case 5
            process=Borra_Reg(grdEstado.Columns(0).Text,          "cve_Estado",
"Estado")
        Case 6
            process=Borra_Reg(Left$(tvwPartida.SelectedItem.Key,
InStr(tvwPartida.SelectedItem.Key, ">") - 1), "Partida_Clave", "Partida")
        Case 7
            process=Borra_Reg(grdTInm.Columns(0).Text,              "cve_tipo",
"Tipo_Inmueble")
        Case 8
            process=Borra_Reg(grdCiudad.Columns(0).Text,          "cve_ciudad",
"Ciudad")

        End Select

    Case 7
        'no se elimina

    End Select

    If process Then
        bandera(queForma) = False
        MsgBox "Registro Borrado"
    Else
        MsgBox "No se pudo borrar el registro"
    End If

End Sub
```

Objeto: CmdCerrar **Procedimiento: Click**

```
Private Sub cmdCerrar_Click()
    Unload Me
End Sub
```

Objeto: CmdModificar **Procedimiento: Click**

```
Private Sub cmdModificar_Click()
    Dim sTag As String
    Select Case queForma
    Case 5
        TagInit sTag, grdEstado.Columns(0).Text
        TagInsert sTag, grdEstado.Columns(1).Text
        TagInsert sTag, queForma
        frmEstado.Tag = sTag
        frmEstado.Show MODAL
    Case 0
        TagInit sTag, grdContratista.Columns(0).Text
        TagInsert sTag, queForma
        frmContratista.Tag = sTag
        frmContratista.Show MODAL
    Case 1
        TagInit sTag, grdLocalidad.Columns(0).Text
        TagInsert sTag, grdLocalidad.Columns(1).Text
        TagInsert sTag, queForma
        frmLocalidad.Tag = sTag
        frmLocalidad.Show MODAL
    Case 3
        TagInit sTag, grdComponente.Columns(0).Text
        TagInsert sTag, grdComponente.Columns(1).Text
        TagInsert sTag, queForma
        frmComp.Tag = sTag
        frmComp.Show MODAL
    Case 4
        TagInit sTag, grdClasInm.Columns(0).Text
        TagInsert sTag, grdClasInm.Columns(1).Text
        TagInsert sTag, queForma
        frmCatClas.Tag = sTag
        frmCatClas.Show MODAL
    Case 2
        TagInit sTag, grdGasto.Columns(0).Text
        TagInsert sTag, grdGasto.Columns(1).Text
        TagInsert sTag, queForma
        frmGasto.Tag = sTag
        frmGasto.Show MODAL
    Case 6
        TagInit            sTag,            Left$(tvwPartida.SelectedItem.Key,
InStr(tvwPartida.SelectedItem.Key, ">") - 1)
        TagInsert sTag, tvwPartida.SelectedItem.Text
        TagInsert sTag, queForma
        frmPartida.Tag = sTag
        frmPartida.Show MODAL
    Case 7
        TagInit sTag, grdTInm.Columns(0).Text
        TagInsert sTag, grdTInm.Columns(1).Text
```

```
        TagInsert sTag, queForma
        frmTipoInm.Tag = sTag
        frmTipoInm.Show MODAL
    Case 8
        TagInit sTag, grdCiudad.Columns(0).Text
        TagInsert sTag, grdCiudad.Columns(2).Text
        TagInsert sTag, queForma
        frmCiudad.Tag = sTag
        frmCiudad.Show MODAL

    End Select

End Sub
```

Objeto: CmdNuevo **Procedimiento: Click**

```
Private Sub CmdNuevo_Click()

    Select Case queForma
    Case 5
        frmEstado.Tag = queForma
        frmEstado.Show MODAL
    Case 0
        frmContratista.Tag = queForma
        frmContratista.Show MODAL
    Case 1
        frmLocalidad.Tag = queForma
        frmLocalidad.Show MODAL
    Case 3
        frmComp.Tag = queForma
        frmComp.Show MODAL
    Case 4
        frmCatClas.Tag = queForma
        frmCatClas.Show MODAL
    Case 2
        frmGasto.Tag = queForma
        frmGasto.Show MODAL
    Case 6
        Dim sTag As String

        TagInit sTag, queForma
        TagInsert          sTag,          Left$(tvwPartida.SelectedItem.Key,
InStr(tvwPartida.SelectedItem.Key, ">") - 1)
        frmPartida.Tag = sTag
        frmPartida.Show MODAL
    Case 7
        frmTipoInm.Tag = queForma
        frmTipoInm.Show MODAL
    Case 8
        frmCiudad.Tag = queForma
        frmCiudad.Show MODAL
    End Select

End Sub
```

Objeto: Form **Procedimiento: Activate**

```
Private Sub Form_Activate()
    Dim i As Integer

    'asegura que entre a todos los tab
    For i = 0 To UBound(bandera)
        bandera(i) = False
    Next i

End Sub
```

Objeto: Form **Procedimiento: Load**

```
Private Sub Form_Load()

    Me.Height = 4620
    Me.Width = 8895

End Sub
```

Objeto: grdCiudad **Procedimiento: UnboundReadData**

```
Private Sub grdCiudad_UnboundReadData(ByVal RowBuf As RowBuffer, StartLocation
As Variant, ByVal ReadPriorRows As Boolean)
    GrillaLee RowBuf, StartLocation, ReadPriorRows
End Sub
```

Objeto: grdClasInm **Procedimiento: UnboundReadData**

```
Private Sub grdClasInm_UnboundReadData(ByVal RowBuf As RowBuffer,
StartLocation As Variant, ByVal ReadPriorRows As Boolean)
    GrillaLee RowBuf, StartLocation, ReadPriorRows
End Sub
```

Objeto: grdComponente **Procedimiento: UnboundReadData**

```
Private Sub grdComponente_UnboundReadData(ByVal RowBuf As RowBuffer,
StartLocation As Variant, ByVal ReadPriorRows As Boolean)
    GrillaLee RowBuf, StartLocation, ReadPriorRows
End Sub
```

Objeto: grdContratista **Procedimiento: UnboundReadData**

```
Private Sub grdContratista_UnboundReadData(ByVal RowBuf As RowBuffer,
StartLocation As Variant, ByVal ReadPriorRows As Boolean)
    GrillaLee RowBuf, StartLocation, ReadPriorRows
End Sub
```

Objeto: grdEstado **Procedimiento: UnboundReadData**

```
Private Sub grdEstado_UnboundReadData(ByVal RowBuf As RowBuffer, StartLocation
As Variant, ByVal ReadPriorRows As Boolean)
    GrillaLee RowBuf, StartLocation, ReadPriorRows
End Sub
```

Objeto: grdGasto **Procedimiento: UnboundReadData**

```
Private Sub grdGasto_UnboundReadData(ByVal RowBuf As RowBuffer, StartLocation
As Variant, ByVal ReadPriorRows As Boolean)
    GrillaLee RowBuf, StartLocation, ReadPriorRows
End Sub
```

Objeto: grdLocalidad **Procedimiento: UnboundReadData**

```
Private Sub grdLocalidad_UnboundReadData(ByVal RowBuf As RowBuffer,
StartLocation As Variant, ByVal ReadPriorRows As Boolean)
    GrillaLee RowBuf, StartLocation, ReadPriorRows
End Sub
```

Objeto: grdTInm **Procedimiento: UnboundReadData**

```
Private Sub grdTInm_UnboundReadData(ByVal RowBuf As RowBuffer, StartLocation
As Variant, ByVal ReadPriorRows As Boolean)
    GrillaLee RowBuf, StartLocation, ReadPriorRows
End Sub
```

Objeto: SSTab1 **Procedimiento: Click**

```
Private Sub SSTab1_Click(PreviousTab As Integer)
    Dim sQuery As String

    queForma = SSTab1.Tab
     Select Case queForma
        Case 0
            If Not bandera(queForma) Then
                sQuery = SELECT_STR + "cve_contrat, nom_contrat,
telef_contrat" + FROM_STR + vbLf
                sQuery = sQuery + "Contratista"
                GrillaLlena sQuery, grdContratista
                bandera(queForma) = True
            End If
        Case 1
            If Not bandera(queForma) Then
                sQuery = SELECT_STR + "*" + FROM_STR + vbLf
                sQuery = sQuery + "Localidad"
                GrillaLlena sQuery, grdLocalidad
```

```
                bandera(queForma) = True
            End If
        Case 5
            If Not bandera(queForma) Then
                sQuery = SELECT_STR + "*" + FROM_STR + vbLf
                sQuery = sQuery + "Estado"
                GrillaLlena sQuery, grdEstado
                bandera(queForma) = True
            End If
        Case 3
            If Not bandera(queForma) Then
                sQuery = SELECT_STR + "*" + FROM_STR + vbLf
                sQuery = sQuery + "Componente"
                GrillaLlena sQuery, grdComponente
                bandera(queForma) = True
            End If
        Case 4
            If Not bandera(queForma) Then
                sQuery = SELECT_STR + "*" + FROM_STR + vbLf
                sQuery = sQuery + "Clasificacion_Inmueble"
                GrillaLlena sQuery, grdClasInm
                bandera(queForma) = True
            End If
        Case 2
            If Not bandera(queForma) Then
                sQuery = SELECT_STR + "*" + FROM_STR + vbLf
                sQuery = sQuery + "Gasto"
                GrillaLlena sQuery, grdGasto
                bandera(queForma) = True
            End If
        Case 6
            If Not bandera(queForma) Then
                Dim nodX As Node
                tvwPartida.Nodes.Clear
                'adiciona las propiedades al nodo
                Set nodX = tvwPartida.Nodes.Add(, , "0>" & "RAIZ", "Partidas",
"RAIZ")
                nodX.Tag = "RAIZ"
                tvwPartida_NodeClick nodX
                bandera(queForma) = True
            End If
        Case 7
            If Not bandera(queForma) Then
                sQuery = SELECT_STR + "*" + FROM_STR + vbLf
                sQuery = sQuery + "Tipo_Inmueble"
                GrillaLlena sQuery, grdTInm
                bandera(queForma) = True
            End If
        Case 8
            If Not bandera(queForma) Then
                sQuery = SELECT_STR + "a.cve_ciudad, b.descrip_estado,
a.nom_ciudad" + FROM_STR + vbLf
                sQuery = sQuery + "Ciudad a, Estado b"
                sQuery = sQuery + WHERE_STR + "a.cve_estado = b.cve_estado"
                GrillaLlena sQuery, grdCiudad
                bandera(queForma) = True
            End If
    End Select
End Sub
```

Objeto: tvwImagen **Procedimiento: NodeClick**

```
Private Sub tvwImagen_NodeClick(ByVal Node As Node)
    'habilita manejador de errores
    On Error GoTo tvwImagen_NodeClick

    Dim nodX As Node
    Dim rdoQuery As rdoResultset
    Dim szCampoUno As String
    Dim sClave As String
    Dim sSql As String
    Dim iCount As Integer
    Dim iImagenId As Integer

    If Node.Tag = "RAIZ" Then
        sSql = SELECT_STR + "Cve_Inmueble, Descrip_Inmueble,0" + vbLf
        sSql = sSql + FROM_STR + "Inmueble" + vbLf
    Else
        sSql    =    SELECT_STR    +    "A.Cve_Inmueble,    B.Imagen_Inmueble_Ruta,
B.Imagen_Inmueble_Id" + vbLf
        sSql = sSql + FROM_STR + "Inmueble A, Imagen_Inmueble B" + vbLf
        sSql = sSql + WHERE_STR + vbLf
        sSql = sSql + "A.Cve_Inmueble = B.Cve_Inmueble" + vbLf
        sSql = sSql + AND_STR + "A.Cve_Inmueble =" + Node.Tag
    End If

    'si tiene hijos sal
    If Node.Children > 0 Then Exit Sub

    Set rdoQuery = BuildCursor(sSql)
    For iCount = 0 To rdoQuery.RowCount - 1
        szCampoUno = rdoQuery(1)
        sClave = rdoQuery(0)
        iImagenId = rdoQuery(2)

        Set nodX = tvwImagen.Nodes.Add(Node.Key, _
                tvwChild, _
                sClave & ">" & iImagenId, szCampoUno, _
                "RAIZ", "RAIZ")

        nodX.Tag = rdoQuery(0)
        rdoQuery.MoveNext
    Next iCount

    Node.Expanded = True
    Exit Sub

tvwImagen_NodeClick:
    Select Case Err
    Case 91

    Case 35602
    Case Else
        MsgBox Err.Description
    End Select
End Sub
```

Objeto: tvwPartida **Procedimiento: NodeClick**

```
Private Sub tvwPartida_NodeClick(ByVal Node As Node)
    'habilita manejador de errores
    On Error GoTo tvwPartida_NodeClick

    Dim nodX As Node
    Dim rdoQuery As rdoResultset
    Dim szCampoUno As String
    Dim sClave As String
    Dim sSql As String
    Dim iCount As Integer
    Dim iClave_Comp As Integer

    If Node.Tag = "RAIZ" Then
        sSql = SELECT_STR + "*" + vbLf
        sSql = sSql + FROM_STR + "Partida" + vbLf
        sSql = sSql + WHERE_STR + vbLf
        sSql = sSql + "Partida_Clave = Partida_Ascendente" + vbLf
    Else
        sSql = SELECT_STR + "*" + vbLf
        sSql = sSql + FROM_STR + "Partida" + vbLf
        sSql = sSql + WHERE_STR + vbLf
        sSql = sSql & "Partida_Ascendente = " & Node.Tag
        sSql = sSql & AND_STR & "Partida_Clave <> Partida_Ascendente"
    End If

    'si tiene hijos sal
    If Node.Children > 0 Then Exit Sub
    'adiciona las unidades de negocio

    Set rdoQuery = BuildCursor(sSql)
    For iCount = 0 To rdoQuery.RowCount - 1
        szCampoUno = rdoQuery(1)
        sClave = rdoQuery(0)
        iClave_Comp = rdoQuery(2)
        Set nodX = tvwPartida.Nodes.Add(Node.Key, _
                tvwChild, _
                sClave & ">" & iClave_Comp, szCampoUno, _
                "RAIZ", "RAIZ")

        nodX.Tag = rdoQuery(0)
        rdoQuery.MoveNext
    Next iCount

    Node.Expanded = True
    Exit Sub

tvwPartida_NodeClick:
    MsgBox Err.Description

End Sub
```

Subrutina GrillaLee.

```
Sub GrillaLee(ByVal RowBuf As Object, StartLocation As Variant, ByVal
ReadPriorRows As Boolean)
    On Error GoTo ManError

    Dim Bookmk As Variant
     Bookmk = StartLocation

    Dim RelPos As Integer
    If ReadPriorRows Then
        ' la grilla es reconsultada
        RelPos = -1
    Else
        ' la grilla es reconsultada antes de ser leida
        RelPos = 1
    End If

    Dim RowsFetched As Integer
    RowsFetched = 0
    Dim i%, j%

    ' loop por cada renglon empezando en cero
    For i% = 0 To RowBuf.RowCount - 1

        ' obtiene el identificador para el renglon
        Bookmk = GetRelativeBookmark(Bookmk, RelPos)

        If IsNull(Bookmk) Then Exit For
        ' si el siguiente identificador es nulo, es EOF o BOF
        ' sale de la funcion e indica cuantos renglones
        ' fueron leidos

        For j% = 0 To RowBuf.ColumnCount - 1
        ' obtiene los datos del arreglo

            RowBuf.Value(i%, j%) = GetUserData(Bookmk, j%)
        Next j%

        RowBuf.Bookmark(i%) = Bookmk

        ' incrementa el contador para el siguiente renglon
        RowsFetched = RowsFetched + 1
    Next i%

    ' le dice a la grilla cuantos renglones se leyeron
    RowBuf.RowCount = RowsFetched
    Exit Sub

ManError:
    MsgBox Err.Description
End Sub
```

7.3.3 HERRAMIENTAS DE PUESTA A PUNTO.

Todas las herramientas de puesta a punto que se utilizaron para el Sistema Integral de Administración Inmobiliaria vienen incluídas en Visual Basic. No fue necesario emplear alguna herramienta externa porque el depurador, el editor de menús, el diseñador de reportes y el diseñador de pantallas de Visual Basic son herramientas muy completas y nos permitieron dar un muy buen funcionamiento y presentación de menús, pantallas y reportes, así como detectar errores e inconsistencias a través del depurador. Por lo tanto, todos estos elementos de Visual Basic fueron nuestras herramientas de puesta a punto.

7.4 RECURSOS DE MULTIMEDIA.

En el Sistema Integral de Administración Inmobiliaria (SIAI) el módulo de *Inventario de* Inmuebles muestra con imágenes la fotografía del inmueble y el plano de ubicación del inmueble. Estas imágenes las toma el sistema de archivos ya procesadas a la vez de reproducir un archivo de sonido, el cual toma también ya procesado.

A continuación se describe el procedimiento empleado para procesar las imágenes y archivos de sonido y el software empleado para ello:

Archivos de imágenes.

1. Tener la fotografía del inmueble y/o plano de ubicación.
2. Escanear la fotografía a través de un Scanner de cama plana utilizando el paquete *Visioner Paper* Port V. 3.01.
3. Grabar la fotografía escaneada en un archivo con formato pcx ó gif.
4. Procesar el archivo de imagen con formato pcx ó gif para ajustar colores y resolución utilizando el paquete Photoshop.
5. Guardar la imagen final en un archivo con formato bmp.

Archivos de sonido.

Una vez seleccionada la música se utilizó el paquete Digital Audio Transport para crear los archivos de sonido. El procedimiento fue el siguiente:

1. Definir los parámetros para la grabación, estos fueron:

   a) Frecuencia: 22,000 Hz.
   b) Calidad de sonido: estéreo.
   c) Volumen de grabación.

2. Reproducir el disco compacto y grabar durante 14 segundos.

3. Grabar el archivo con formato wav.

CAPÍTULO 8

Documentación

8. DOCUMENTACION

8. DOCUMENTACION.

8.1 MANUAL DEL USUARIO.

El SIAI cuenta con un manual de usuario en línea, por medio del cual a través de un menú el usuario puede seleccionar el contenido de la ayuda y elegir cualquier tema referente al manejo del mismo. Este manual en línea se puede accesar en cualquier momento de la ejecución. Por tal motivo, el manual del usuario que se presenta a continuación, abarca únicamente los siguientes aspectos:

- Como entrar al SIAI.
- Como operar la pantalla de acceso al SIAI.
- Las áreas de las que se compone la pantalla principal del SIAI.
- La descripción de cada una de las opciones del menú principal del SIAI.
- Notación general para menús, ayuda y pantallas de actualización del SIAI.
- Como salir del SIAI.
-

Entrada al SIAI.

Para correr el sistema se tienen dos opciones:

Opción uno:

a) Hacer click con el botón izquierdo del mouse en el icono SIAI.

Opción dos:

a) Ir al menú Inicio de Windows 95 y elegir la opción Ejecutar, como se muestra en la FIG. 8.1.

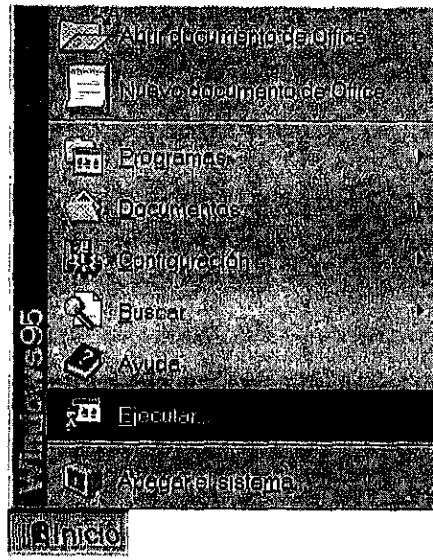


FIG. 8.1. Menú Inicio de Windows 95.

b) En el cuadro de texto Abrir de la ventana Ejecutar, teclear la ruta completa y el nombre del programa para correr el SIAI, como se muestra en la FIG. 8.2.

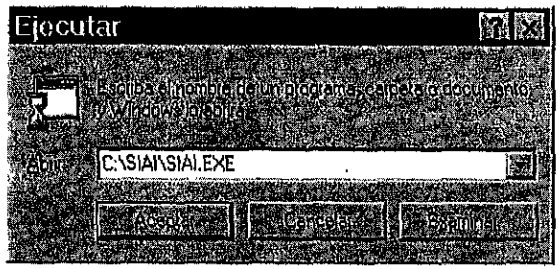


FIG. 8.2. Ventana Ejecutar de la opción ejecutar del Menú Inicio.

c) Hacer click en el botón Aceptar de la ventana Ejecutar.

Pantalla de acceso al SIAI.

Cuando se corre el SIAI, lo primero en aparecer es la pantalla de acceso al sistema, la cual se presenta en la FIG. 8.3.

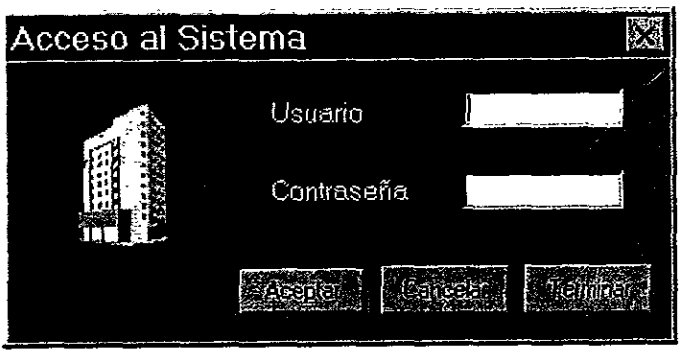


FIG. 8.3. Pantalla de acceso al sistema.

Para entrar al sistema deberán proporcionarse los datos requeridos de *Usuario* y *Contraseña* y oprimir <Enter>, o bien hacer click en *Aceptar*, *Cancelar* o *Terminar*, según sea el caso.

Pantalla principal del SIAI.

Si el nombre de usuario y la contraseña son correctos, aparece la pantalla principal del sistema, la cual se presenta en la FIG. 8.4 con las partes que la componen.

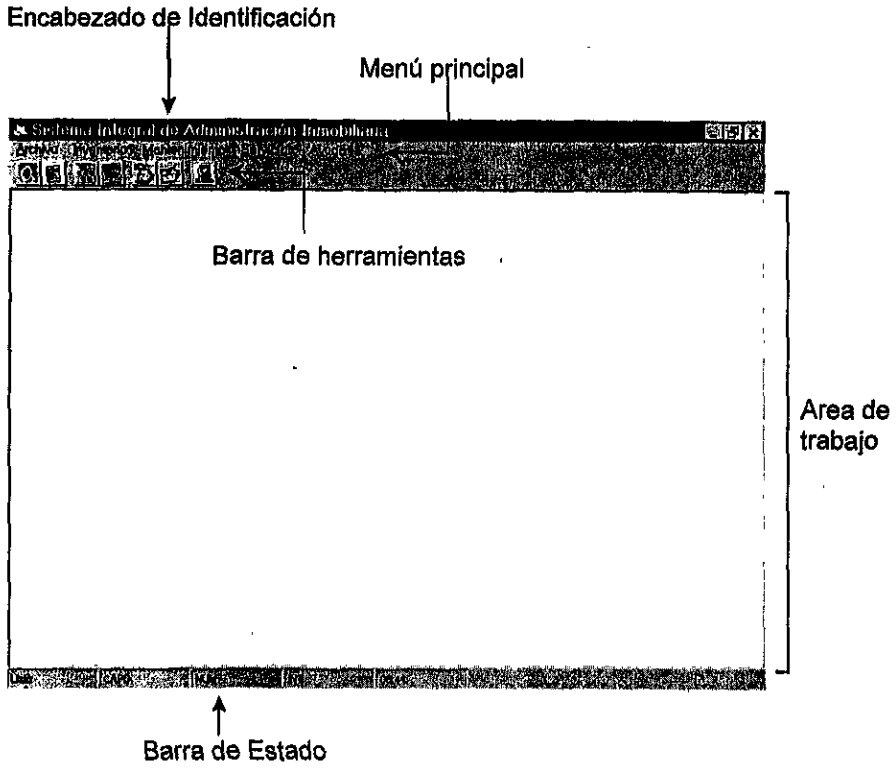


FIG. 8.4 Pantalla principal del SIAI y partes que la componen.

Descripción de cada una de las opciones del Menú Principal del SIAI.

MENU ARCHIVO.

Este módulo tiene las herramientas para el manejo y respaldo de la base de datos, para la seguridad del sistema, para la configuración de la impresora y para salir del sistema. Y tiene las siguientes opciones: *Compactar base de datos, Reparar base de datos, Seguridad, Configurar impresora* y *Salir*.

El menú *Archivo* se presenta en la FIG. 8.5.

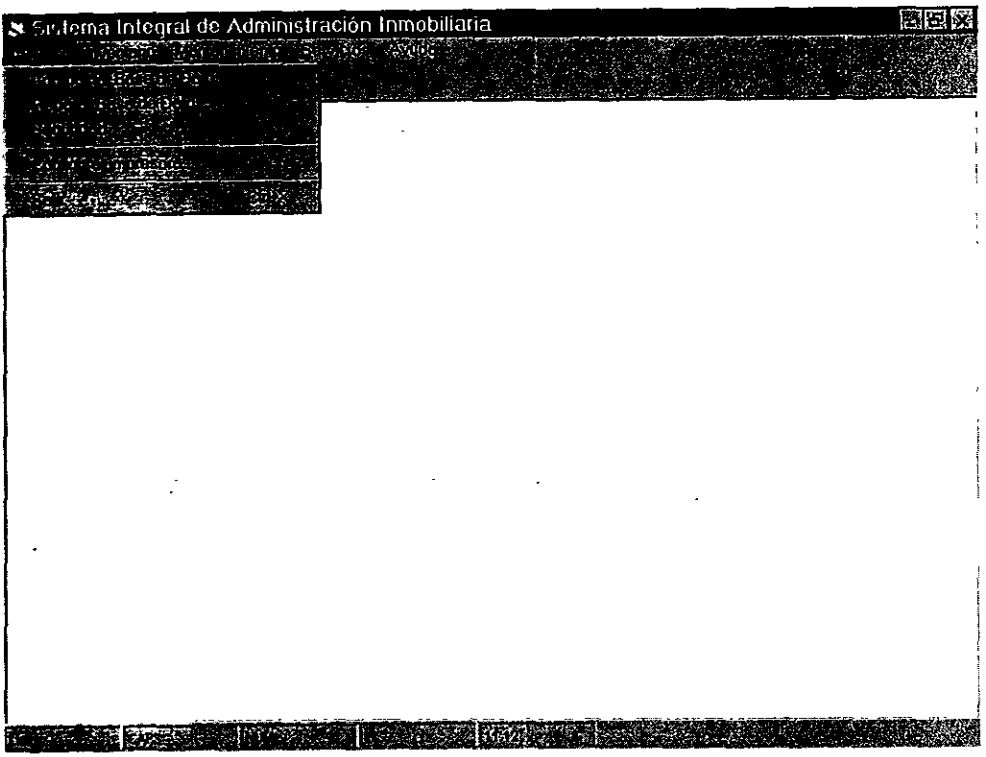


FIG. 8.5 Menú Archivo.

MENU INVENTARIO.

Este módulo del sistema se encarga del Inventario de los Inmuebles y tiene las siguientes opciones: *Inmuebles, Información legal, Gastos y Publicidad*, las cuales permiten registrar y mantener actualizado el inventario. Con la opción *Inmuebles* es posible registrar sus datos generales, construcción, archivos fotográficos de los inmuebles, *Información legal* permite registrar los aspectos legales y la opción *Gastos* registra los gastos por concepto de servicios. La opción de *Publicidad* muestra al cliente las imágenes disponibles de los inmuebles.

El menú *Inventario* se presenta en la FIG. 8.6.

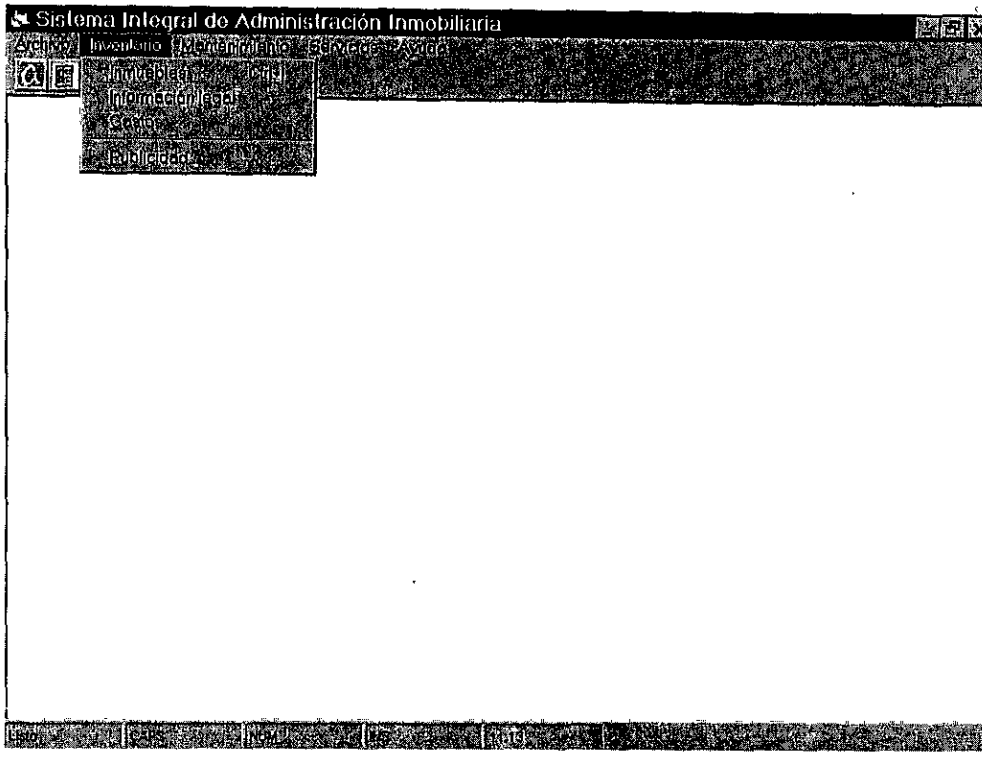


FIG. 8.6 Menú Inventario.

MENU MANTENIMIENTO.

Es el módulo del sistema que proporciona los elementos necesarios para realizar la programación de mantenimientos y tiene las siguientes opciones: *Programación, Avance y Estadísticas*. Aquí se pueden observar los avances en cada uno de los programas, tanto real como programado. Además, se muestran los avances por medio de gráficas de barras.

El menú *Mantenimiento* se presenta en la FIG. 8.7.

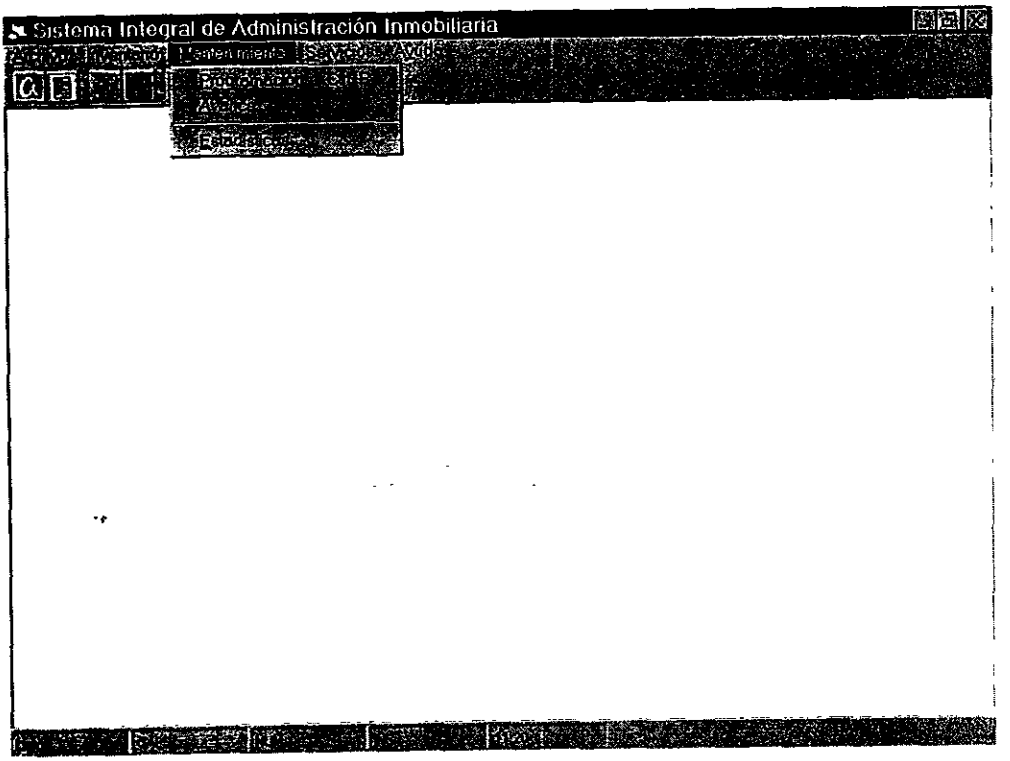


FIG. 8.7 Menú Mantenimiento.

MENU SERVICIOS.

Este módulo del sistema tiene las siguientes opciones: *Generador de Consultas, Catálogos y Opciones.* Proporcionan al usuario la posibilidad de generar reportes y consultas personalizadas con diferentes criterios, además pueden actualizarse los catálogos del sistema que son: Contratistas, localidades, gastos, componentes, clasificación de inmuebles, estados, partidas, tipos de inmuebles y ciudad. También se pueden configurar los parámetros para los reportes como, encabezado, formato numérico de los datos.

El Menú *Servicios* se presenta en la FIG. 8.8.

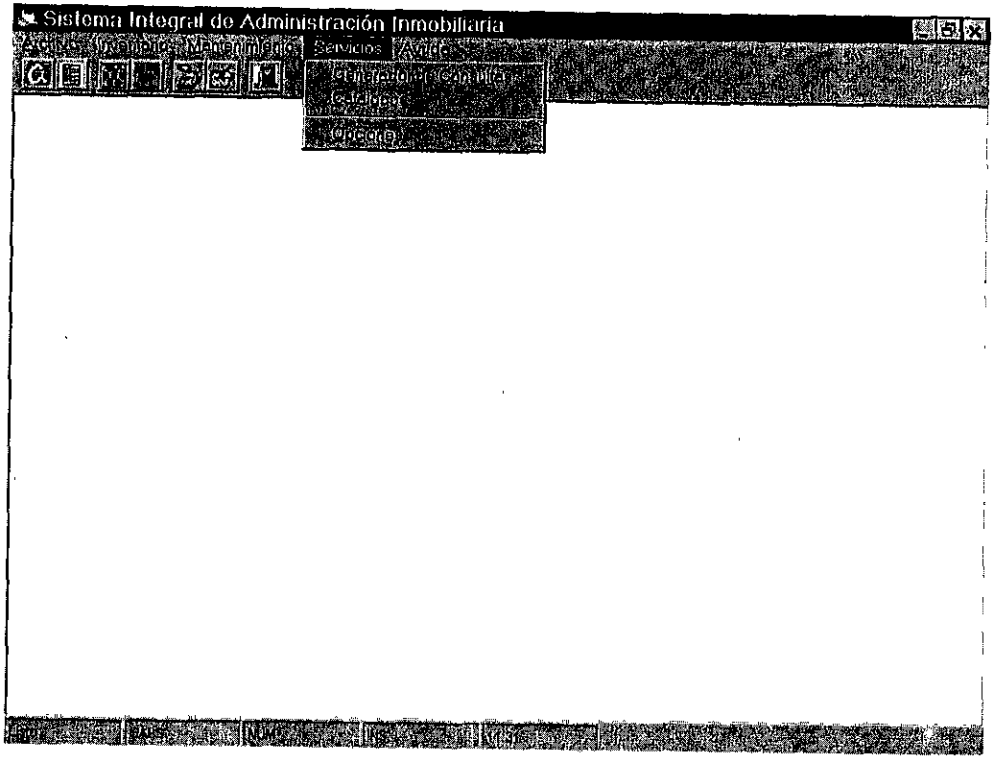


FIG. 8.8 Menú Servicios.

MENU AYUDA.

Es el módulo del sistema que proporciona al usuario la ayuda acerca del sistema y tiene las siguientes opciones: *Contenido*, *Buscar ayuda acerca de* y *Acerca de.* Además cabe aclarar que el sistema cuenta con ayuda en línea disponible para el usuario en el momento que lo requiera.

El Menú *Ayuda* se presenta en la FIG. 8.9.

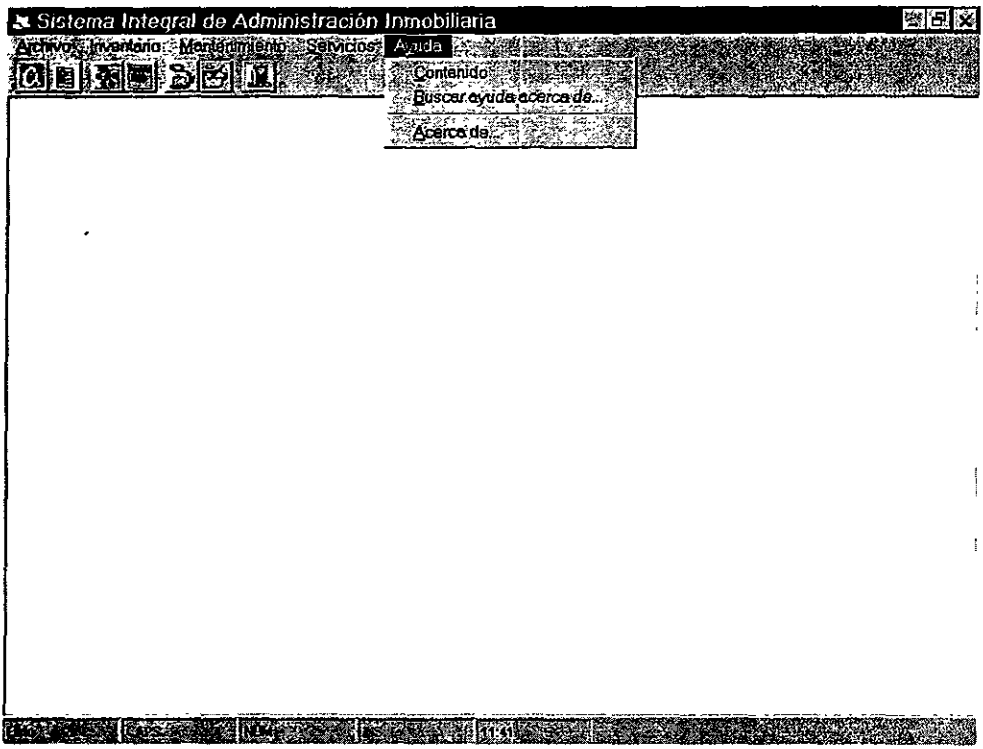


FIG. 8.9 Menú Ayuda.

Notación general para menús.

Para entrar a cada una de las opciones de los menús contenidos en este sistema, existen tres formas que a continuación se explican:

1. Presionar <F10> y se iluminará la barra del menú principal, posicionarse con las flechas de movimiento del cursor sobre la opción deseada y oprimir la tecla <Enter>.
2. Presionar la tecla <Alt > y teclear la letra subrayada de la opción deseada y oprimir la tecla <Enter>.
3. Hacer *click* con el botón izquierdo del mouse.

Para salir de cualquier opción de los menús se oprime la tecla <Esc>, interrumpiendo automáticamente la opción en curso y regresando a la opción anterior.

Notación general para ayuda.

Todas las opciones de este sistema cuentan con una ayuda; para activarla, basta con posicionarse sobre la palabra buscada y presionar F1, ésta nos despliega en pantalla los datos existentes acerca del tema. Existen además, dos formas de consultar la ayuda:

1. Activar el menú *Ayuda* del menú principal y seleccionar la opción *Contenido*. Aquí se debe teclear la palabra completa o las primeras letras de la palabra, el sistema posicionará el cursor en el renglón correspondiente a la palabra buscada o el cursor será posicionado en el primer renglón correspondiente a las primeras letras de la palabra. Si no existe un registro que empiece con la letra tecleada, aparecerá el mensaje de error *"Esta palabra no se encuentra en el índice. Escriba otra palabra o seleccione una de la lista"*.

2. Activar el menú *Ayuda* del menú principal y seleccionar la opción *Buscar ayuda acerca de*, aquí se puede navegar por toda la pantalla de ayuda mediante las flechas de movimiento del cursor y oprimir <Enter> cuando esté colocado sobre el renglón deseado.

Notación general para pantallas de actualización.

En todas las pantallas de actualización y consulta del SIAI existen botones que indican las acciones necesarias para dar de alta, modificar o borrar registros, de manera que el proceso a seguir es el siguiente:

Para alta de registros:

- Oprimir <Enter> o hacer *click* en el botón Nuevo de la pantalla correspondiente.
- Proporcionar los datos que se van pidiendo en la captura.
- Oprimir <Enter> o hacer *click* en los botones Aceptar o Grabar, según sea el caso.

Para modificación registros:

- Seleccionar el registro a modificar.

  Esto se lleva a cabo haciendo *click* en el botón de búsqueda -?-, lo cual hace que se *muestre una lista* con los registros existentes, entonces hay que posicionarse en el registro a modificar y Oprimir <Enter> o hacer *click* en el botón Seleccionar.
- Proporcionar o modificar los datos que se van pidiendo en la captura.
- Oprimir <Enter> o hacer *click* en los botones Aceptar o Grabar, según sea el caso.

Para el caso de alta, modificación o borrado de registros en los catálogos, los registros existentes aparecen desde el momento de seleccionar el catálogo, entonces, sólo hay que posicionarse en el registro y Oprimir <Enter> o hacer *click* en el botón correspondiente a la acción deseada.

Todas las pantallas del SIAI realizan validaciones a la información y muestran el mensaje correspondiente a acciones no válidas llevadas a cabo por los usuarios.

También se muestran mensajes que corresponden a acciones llevadas a cabo por el sistema, por ejemplo: cuando se lleva a cabo una alta, modificación o borrado de un registro.

Además, las pantallas cuentan con el botón Cerrar o con el botón Cancelar, los cuales pueden cerrar la ventana activa y cancelar la operación sin que se realizan cambios en la información.

Salida del sistema.

Existen tres formas para salir del sistema.

1. Activar el menú *Archivo* del menú principal y seleccionar la opción *Salir*, como se muestra en la FIG. 8.10.

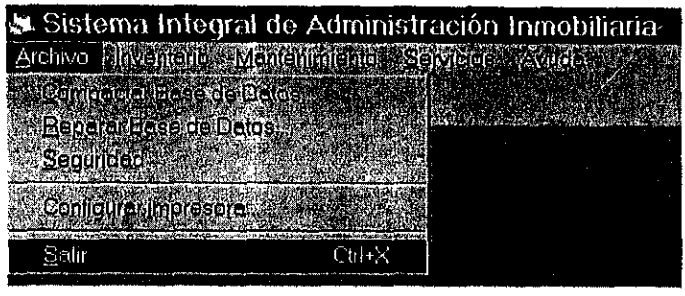


FIG. 8.10 Salida del SIAI mediante el menú Archivo.

2. Hacer *click* en el botón Salir del SIAI de la barra de herramientas del SIAI, como se muestra en la FIG. 8.11.

FIG. 8.11 Salida del SIAI mediante la barra de herramientas.

3. Oprimir <*Ctrl*> <*X*>. Esto lo regresará al sistema operativo.

8.2 MANUAL DE OPERACION.

El Sistema Integral de Administración Inmobiliaria, SIAI, desempeña las funciones especificas de la administración inmobiliaria para los inmuebles propiedad de la empresa.

Para el desarrollo del sistema, se utilizó el análisis y diseño estructurado que comprende los siguientes puntos:

a) Ingeniería de Sistemas.
b) Análisis.
c) Diseño.
d) Codificación.
e) Pruebas.
f) Instalación.
g) Capacitación.
h) Mantenimiento.

Cada uno de estos puntos fue desarrollado completamente en los capítulos ya descritos con anterioridad y están basados en la teoría expuesta. Por lo tanto cada una de las diferentes etapas mencionada forma parte del manual de operación del sistema, porque en ellas se abarcan los requerimientos y restricciones del sistema, los diagramas que muestran el flujo de datos y la descripción de los procesos del sistema, todo lo anterior, apegándose a la administración inmobiliaria.

CAPITULO 9

Pruebas y Confiabilidad del Sistema

9. PRUEBAS Y CONFIABILIDAD DEL SISTEMA

9.1 PRUEBA DE UNIDADES Y PRUEBA DE INTEGRACION.

Este proceso se concentra en probar cada módulo como una entidad individual. Aquí, se presentan las pruebas realizadas al sistema SIAI, las cuales permiten conocer y concluir que el sistema alcanza y supera los lineamientos para el aseguramiento de la calidad.

Para la realización de estas pruebas se generaron datos tipo prueba. Así, para el módulo de Inventario de Inmuebles, las características de los datos fueron entre otras las siguientes: tipo de dato incorrecto, longitud nula, etc. Como ejemplo, al precio de venta, que es numérico, se le asignaron caracteres alfanuméricos. El sistema respondió adecuadamente validando el dato. En otro registro de datos se generaron datos nulos, esto es, cuando existen campos en la base de datos que son requeridos y el sistema hallaba el primer dato nulo, validaba e informaba al usuario del error, y continuaba validando los demás datos. Este proceso se realizó para las altas de los inmuebles para por lo menos 50 registros.

Las pruebas para los datos de tipo numérico se probaron principalmente en sus límites superiores para comprobar el rango válido de almacenamiento, respondiendo el sistema adecuadamente. Asimismo, las longitudes en los datos de tipo cadena se variaron en mayores longitudes permitidas y los resultados fueron los esperados.

Por otro lado, se formularon casos especiales para detectar flujos inapropiados. Por ejemplo; se procedió a dar de alta un inmueble en sus datos generales pero no en sus datos legales; también se probó cuando un inmueble no tenía registrados sus componentes. En todos estos casos, el sistema respondió como se esperaba.

Las variables locales y globales se comprobaron para lo que fueron declaradas (llevar un contador, almacenar temporalmente dátos; etc.), sin obtener terminación de bucles inapropiada o inexistente, variables de bucles modificadas de forma inapropiada o cualquier otro tipo de error.

Cuando en las pruebas de modificaciones y consultas se alteraban los datos en sus tipos y longitudes, el flujo de datos continuaba comportándose establemente.

En otras pruebas de unidad que se realizaron están las pruebas al módulo de programación de mantenimientos, un módulo donde, una vez dados los inmuebles de alta, éstos requieren de mantenimiento (pintarlos, ampliarlos, etc.), los datos generados sirvieron para ver cómo se iba actualizando la interfaz conforme el mantenimiento avanzaba. Las pruebas más importantes en este módulo fueron en las cantidades numéricas y sus límites, así como en los datos de tipo fecha.

Los resultados obtenidos en la prueba de unidad fueron los esperados hasta la conclusión del mantenimiento.

Para este modulo, conforme se realiza el mantenimiento, los datos van sufriendo cambios. Por eso resulta interesante determinar como la interfaz y el control de la misma se está llevando a cabo. Esto se refleja en las gráficas de avance y estadísticas.

La prueba de integración utilizada fue la integración descendente donde se integran los módulos hacia abajo o a lo ancho de la aplicación. En la FIG. 9.1 se muestra la estructura jerárquica del sistema.

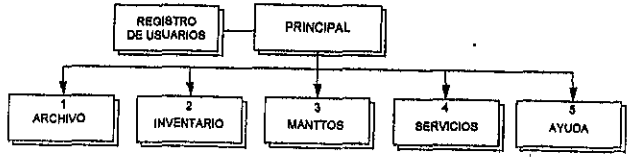


FIG. 9.1. Estructura del SIAI en su nivel 1.

El primer módulo que se probó fue Datos generales, que funciona como el conductor de pruebas principal sobre los módulos subordinados. Una vez probado, se incorporó el módulo de Datos legales. Posteriormente el módulo de Gastos. En este punto se supuso que datos de los catálogos necesarios estaban datos de alta. El flujo de los datos fue el esperado. El módulo de publicidad se probó suponiendo que las imágenes estaban ya validadas y asignadas por el módulo de catálogos donde se liga la imagen con el inmueble.

A continuación se probó el módulo Mantenimiento de inmuebles con sus respectivos módulos subordinados. En este punto se puso mucha atención en el flujo de la información, ya que conforme avanza el programa, los datos se van modificando y los resultados pueden estar variando hasta llegar al 100% del mantenimiento. Los resultados y el flujo de información fueron como se esperaba.

Así, se probaron todos los módulos, hasta el módulo de más profundidad y el sistema se comportó en el flujo de información y en desempeño de una manera aceptable.

9.2 PRUEBA DE VALIDACION.

La validación se logró cuando el sistema logró las expectativas del cliente y la especificación de requisitos.

El plan para las pruebas fue el de liberarle al cliente módulo por módulo y que el mismo usuario final fuera el que validara todos los requisitos. Después surgió una versión alfa para detectar errores que sólo el usuario final puede detectar. A partir de esta versión, se corrigieron los errores detectados y se liberó una versión nueva llamada beta que fue probada también junto con el usuario final. De esta versión se obtuvieron los resultados esperados especificados en el diseño, tanto en rendimiento como en interfaces.

9.3 PRUEBA DE VOLUMEN.

Por otro lado, la prueba de volumen se realizó con aproximadamente 1000 registros que es el promedio de inmuebles administrados por la compañía, el tiempo de respuesta y el volumen de transacciones fue aceptable de acuerdo con los requerimientos.

CAPITULO 10

Instalación, Mantenimiento y Aseguramiento de la Calidad del Sistema

10. INSTALACION, MANTENIMIENTO Y ASEGURAMIENTO DE LA CALIDAD DEL SISTEMA

10.1 PLAN DE INSTALACION.

Este requiere de un proceso donde todos los elementos necesarios se integran para que el sistema funcione y realice los procesos requeridos, esto involucra un proceso de compilación de archivos del sistema y controles de Visual Basic 4.0. Primero, se identificaron los elementos que componen el sistema, los cuales se muestran en la TABLA 10.1. Una vez identificados todos los componentes necesarios para que opere el SIAI, se instaló en el servidor de archivos el administrador de base de datos Access 7.0 y la base de datos de sistema sin información alguna. Se carga el administrador como usuario único de la base de datos.

También se instaló el ODBC en las máquinas de los clientes y se verificaron los siguientes puntos:

- Plataforma donde se instaló el sistema.
- Características de las PC's.

Después se realizó un instalador de la aplicación mediante un generador de discos de instalación, los cuales se instalan en las diferentes máquinas de la empresa que funcionan como cliente. El instalador, verifica las versiones de archivos instalados en dichas máquinas, así como la existencia de la aplicación. Una vez instalado el sistema, se realizaron pruebas comunicación entre los clientes y el servidor.

El software seleccionado para realizar el instalador forma parte de Visual Basic 4.0 y tiene entre sus principales funciones, tiene las siguientes:

- Generar los discos de instalación con los archivos compactados.
- Registrar los controles necesarios y crear los directorios especificados, verificar el espacio en disco y archivos que existan con el mismo nombre.
- Asistir al usuario para que el proceso sea lo más sencillo posible y transparente.
- Definir la configuración ideal para las PC's.

Finalmente se procedió a realizar las pruebas de comunicación entre el sistema y la base de datos, dando por terminado el proceso de instalación del sistema.

ARCHIVO	DESCRIPCION
VB40032.DLL	Runtime para aplicaciones Visual Basic 4.0
ven2232.olb	Librería de objetos para aplicaciones Visual Basic
olepro32.dll	DLL para soporte de propiedades OLE
msvcrt20.dll	Librería de runtime
msvcrt40.dll	Librería de runtime
ctl3d32.dll	Controles de Windows 3D
VB4ES32.DLL	Visual Basic 4.0 Recursos internacionales
OC30.DLL	DLL para runtime OLE
THREED32.OCX	Control OLE de 32 Bits
MFC40.DLL	Libreria Compartida por aplicaciones Windows
MFC40LOC.DLL	MFC de recursos especificos de lenguaje
DBLIST32.OCX	Control OLE de 32 Bits
COMDLG32.OCX	Control OLE de 32 Bits
TABCTL32.OCX	Control OLE de 32 Bits
GRAPH32.OCX	Control OLE de 32 Bits
gsw32.exe	Servidor de gráficos
gswdll32.dll	DLL del servidor de gráficos
COMCTL32.OCX	Control OLE de 32 Bits
CCTLES32.DLL	DLL del control OLE COMCTL32
SSCALB32.OCX	Control de día
SSCALA32.OCX	Controles Mes/Año/Combo Fecha
MSRDO32.DLL	Control rdoEngine
MSMASK32.OCX	Control OLE de 32 Bits
DBGRID32.OCX	Control OLE de 32 Bits
GRDKRN32.DLL	DLL del control DBGRID32
MCI32.OCX	Control OLE de 32 Bits
CRYSTL32.OCX	Control OLE de 32 Bits
CRPE32.DLL	DLL del control CRYSTL32
CRXLAT32.DLL	DLL para Crystal Reports ToWords (solamente ingles)
V2DDSKES.DLL	DLL de destino de exportación a disco para Crystal Reports
V2FDIFES.DLL	DLL de formato de exportación DIF para Crystal Reports
V2FRECES.DLL	DLL de formato de exportación de registros para Crystal Reports
V2FSPVES.DLL	DLL de formato de exportación de valores separados por un carácter para Crystal Reports
V2FTXTES.DLL	DLL de formato de exportación de texto para Crystal Reports
q2bdaoES.dll	DLL de bases de datos físicas de Crystal Reports para Access de Microsoft
q2cdaoES.dll	DLL de diccionario físico de Crystal Reports para Access de Microsoft
q2rdaoES.dll	DLL de directorio físico de Crystal Reports para Access de Microsoft
q2sodbES.dll	DLL de servidor físico de Crystal Reports para ODBC
sndrec32.exe	Accesorio Grabadora de sonido
SPIN32.OCX	Control OLE de 32 Bits
GEAR32SO.OCX	Control OLE de 32 Bits
Sial.exe	Ejecutable de la aplicación de Administración Inmobiliaria
Sial.hlp	Archivo de ayuda de la aplicación de Administración Inmobiliaria

TABLA 10.1. Elementos que componen el sistema.

10.2 APROBACION FINAL.

Como se pudo apreciar en las pruebas, los resultados obtenidos fueron los planeados y con las versiones alfa y beta el usuario final comprobó que lo que el sistema le ofrece es lo que él esperaba. Esto lo comprueba con la administración de la información y las salidas que el sistema ofrece. Además de la facilidad de uso, la seguridad en la información, entre otras.

Una vez instalado el sistema se comenzó a operar de forma real. Como resultado de las pruebas el sistema fue aprobado por el personal de la empresa involucrado, tanto a nivel operativo, como administrativo y gerencial.

10.3 IDENTIFICACION DE RESULTADOS Y DESVIACIONES.

Una vez instalado y aprobado el sistema, se inició una etapa de trabajo de operación normal, pero por cualquier eventualidad se tenía que venir trabajando como antes para asegurar que la operación no se detuviera. Esta forma de operar logro percibir aún más los beneficios que el sistema va a proporcionar a corto y largo plazo.

Esto quiere decir que el sistema no tiene ninguna desviación respecto a los objetivos planteados inicialmente y además los resultados identificados coinciden con los alcances del sistema.

10.4 REVISIONES TECNICAS FORMALES.

La calidad del sistema se empezó a valorar desde la fase de análisis. Al personal directivo de la empresa inmobiliaria se le hicieron presentaciones del diseño para poder detectar problemas en las funciones, en la lógica y en los requerimientos.

El personal de desarrollo estableció estándares en la programación, en nombres de variables, etc.

Durante las sesiones de revisiones técnicas se utilizó el formato descrito en la sección 2.10.7.

En un ambiente de cordialidad y respeto se resolvieron los problemas que se presentaban y entre los puntos sobre los cuales se puso mayor cuidado están los siguientes:

- Desarrollo de una lista de comprobaciones.
- Repasar las revisiones anteriores.
- Contar con un moderador para las sesiones.

Entre las características más importantes que se cuidaron para tener un producto de calidad se encuentran las siguientes:

Integridad.

El modelo relacional de la base de datos y la misma base de datos nos ayudan a tener la máxima integridad de datos.

Facilidad de uso.

El esfuerzo requerido para aprender el sistema fue realmente muy corto aproximadamente una semana en sesiones diarias de dos horas. Además de la facilidad que se tiene en la lectura de las salidas del sistema, el manual en línea demostró su eficiencia cuando se presentó algún problema en el usuario.

Completitud.

Los requerimientos de los usuarios fueron contemplados casi al 100%.

Consistencia.

Los documentos existentes en el proyecto tienen un estándar que se definió desde la primera fase. Así, se tiene un proyecto con sus documentos uniformes.

Modularidad.

Los módulos del sistema pueden trabajar de forma independiente, contemplando claro, parámetros iniciales y datos de entrada correctos, lo que permite la reusabilidad de los mismos así como futuras actualizaciones.

Simplicidad.

El diseño estructurado permite generar sistemas muy sólidos presentados con una estructura simple, este es el caso del SIAI. En el código cualquier persona con los conocimientos básicos de programación y de la herramienta Visual Basic 4.0 puede entender la forma en que está escrito el sistema.

Seguridad.

El acceso al Sistema está restringido para cada usuario dependiendo de los permisos que el administrador del sistema haya asignado, así como de las autorizaciones del gerente. Con esta característica se tiene la protección de módulos no autorizados.

10.5 CONFIABILIDAD DEL SISTEMA.

En el transcurso de la etapa más importante del desarrollo de un sistema: el análisis, se cuidó que el sistema final fuera sumamente confiable y esto se ve reflejado en el diseño.

El tiempo medio entre fallos es el siguiente:

TMDF = 6 meses
TMDR = 0.5 meses
TMEF = 6 + 0.5 = 6.5 meses

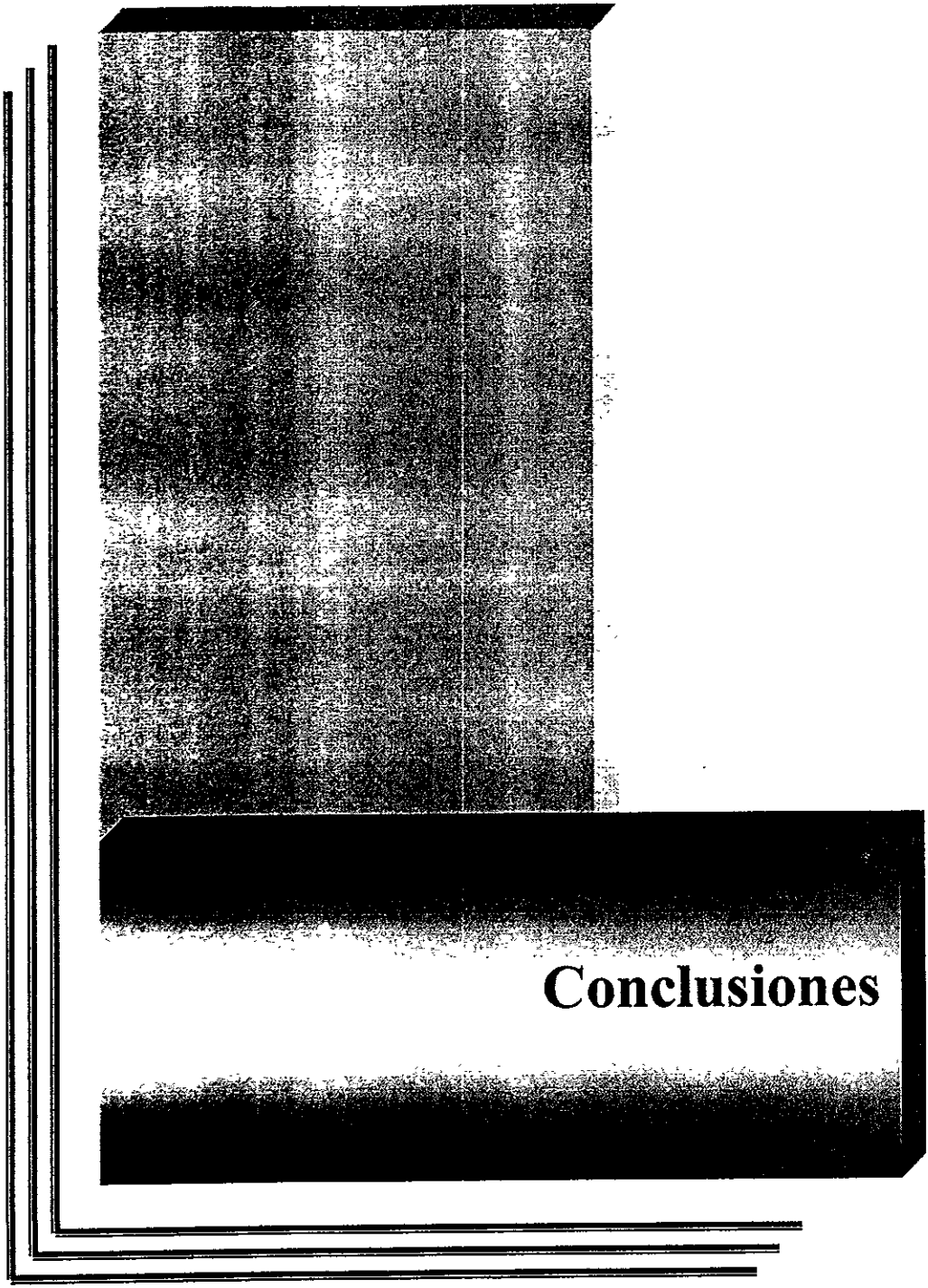
Por otro lado la disponibilidad es la siguiente:

TMDF = 6
TMDR = 0.5

Disponibilidad = 6 / (6+0.5) * 100 %
= 92.3076 %

Podemos observar que el sistema es altamente confiable, aún así se tiene un plan de mantenimiento correctivo, donde en coordinación con la gerencia y usuarios se realizó un diagnóstico para establecer fechas y horarios de corrección.

Para este proyecto no se tienen planeados mantenimientos adaptativos, ni mantenimientos sobre nuevas funciones a menos que la empresa inmobiliaria lo solicitara al grupo de desarrollo.



Conclusiones

CONCLUSIONES

En primer lugar, es muy importante destacar que el solo hecho de crear un producto que resuelve de manera determinante una problemática real resulta bastante positivo y provechoso. Tomando en cuenta, que el Sistema Integral de Administración Inmobiliaria está diseñado a la medida de la Empresa Inmobiliaria, el problema de la información en dicha Empresa ha quedado resuelto.

Dicho lo anterior y asumiendo que muchas empresas en nuestro país tienen un problema similar en mayor o menor grado, la mejor solución para el problema de administrar, controlar, almacenar y explotar la información es la creación de sistemas de información computarizados como el SIAI, creado bajo una metodología de análisis y diseño estructurado, desarrollado con un lenguaje de programación que hoy por hoy tiene el soporte mas grande en la industria del software y aprobado por quienes serán los más beneficiados de la realización de este proyecto: los usuarios.

Así, el SIAI es el resultado de varios factores; el primero de ellos: reconocer la existencia de un problema, después vienen: el planteamiento de una solución computarizada, la designación de recursos para que el sistema sea posible y por último la creación del sistema. El SIAI es la combinación acertada de aplicar la metodología del análisis y diseño estructurado junto con la utilización de una herramienta de vanguardia como lo es Visual Basic 4.0, además del aprovechamiento de las ventajas que proporciona la arquitectura cliente-servidor. El nivel tecnológico con el que ha sido construido el SIAI es muy alto.

Como en todo proyecto, al hacer una evaluación final se obtienen conclusiones, que para el caso del SIAI agruparemos en tres rubros.

- Conclusiones acerca de la metodología empleada para el análisis y el diseño.
- Conclusiones acerca del lenguaje de programación, la forma en que se eligió para desarrollar el proyecto y el desarrollo del sistema.
- La participación del usuario y la interrelación que tuvo con el equipo de análisis y desarrollo.

Conclusiones acerca de la metodología empleada.

Definitivamente siempre será mejor hacer un sistema de información si se utiliza una técnica para el análisis y el diseño. La técnica del análisis y diseño estructurado nos ha permitido contar con todas las etapas necesarias que requiere un proyecto de esta naturaleza. Es una técnica muy completa pero a la vez flexible porque todos los pasos a seguir toman el nivel de profundidad que exige el problema a resolver, los documentos que genera esta técnica son de gran trascendencia, como: los diagramas de flujo de datos, el diagrama entidad-relación y el diccionario de datos por citar algunos, pero algo muy importante es que el uso de todos estos documentos no se limita al analista o al programador sino que son también elementos que le sirven de mucho a la empresa para identificar el flujo de información existente, asignar funciones a su personal y desarrollar procedimientos.

Por otra parte, con el empleo de la técnica del análisis y diseño estructurado es fácil determinar la problemática y definir los objetivos del proyecto, también es posible saber el estado actual del proyecto ya que sugiere el uso de herramientas de control de avance.

También se concluye que el trabajo del programador y en general de todos los participantes en el proyecto tomaría un alto grado de complejidad si no se hubiera empleado esta técnica. En resumen, no encontramos ninguna desventaja y sí muchas ventajas por haber empleado esta técnica.

Conclusiones acerca del lenguaje de programación, la forma en que se eligió para desarrollar el proyecto y el desarrollo del sistema.

El éxito del SIAI se debe también a que en cada etapa se tomaron decisiones atinadas y la elección del lenguaje de programación para el desarrollo del sistema no fue la excepción.

Una conclusión relevante es que para poder elegir un buen lenguaje deben considerarse muchos aspectos que tengan que ver no sólo con el sistema, sino también con el entorno, la plataforma y el equipo donde se va a implementar. Cabe señalar que aunque el factor económico sigue siendo el de mayor peso, existen en la actualidad muchos parámetros importantes que pueden hacer que una inversión económica fuerte quede plenamente justificada.

Después de utilizar Visual Basic 4.0 estamos convencidos de que es un compilador muy potente, versátil y amigable, con un entorno de programación excepcional por todas las herramientas de las que dispone y además fácil de aprender. Es muy completo.

En la etapa de desarrollo resultó muy valioso contar con una carta de estructura la cual sirve de base para realizar los procedimientos y subrutinas que permite el lenguaje.

Queda de manifiesto también que es mejor liberar un proyecto módulo por módulo que todo junto, porque así, se puede ir haciendo pruebas y se obliga al usuario a participar activamente lo cual da como resultado que se hagan observaciones sobre los procesos y eso lleva a dar mantenimiento al sistema con el fin de que cumpla totalmente con todos los requisitos del usuario.

Conclusiones acerca de la participación del usuario y la interrelación que tuvo con el equipo de análisis y desarrollo.

La conclusión principal de este rubro es que cuando se crea un nuevo sistema de información siempre va ser más útil y de mejor calidad si se toma en cuenta la opinión de todos los involucrados.

El usuario operativo manifiesta sus inquietudes, su opinión es importante porque él es quien va a tener el mayor contacto con el sistema.

El usuario administrativo da su punto de vista, su opinión es importante porque él sabe en que consisten los procedimientos de la empresa.

Nosotros, en este caso como analistas y programadores también opinamos, nuestra opinión es importante porque conociendo el problema podemos sugerir diferentes opciones de solución empleando la tecnología existente y determinando que ventajas y desventajas tienen las diferentes opciones podemos decir cual es mejor y cuanto cuesta.

Finalmente, los directores y gerentes opinan, su opinión es importante porque ellos toman las decisiones.

Como conclusión final, se establece que a pesar de la existencia de obstáculos normales en este tipo de proyectos tales como: resistencia al cambio y apatía por parte de algunos usuarios, el SIAI ha resultado un éxito desde el momento en que se liberó y empezó a operar, obteniéndose de él grandes beneficios como: ahorro de tiempo para todos los procesos, mejor organización en las funciones del personal, cambio de mentalidad en los usuarios (ahora son más intuitivos y con iniciativa), etc. Todo esto paulatinamente se irá reflejando en la productividad de la empresa y por consiguiente la Empresa Inmobiliaria será más competitiva.

También queda establecido que un Ingeniero en Computación está capacitado no sólo para resolver problemas técnicos relacionados con su área, sino para crear verdaderas soluciones que involucren cualquier disciplina y ramo de la industria. Además queda comprobado que un Ingeniero en Computación puede manejar a un nivel muy alto cualquier lenguaje de un área específica por ajeno que parezca.

Finalmente, es bastante satisfactorio el haber enseñado, pero también aprendido de éste proyecto.

Bibliografía

BIBLIOGRAFIA

Burch, John G., Grundnitski, Gary.
Diseño de sistemas de información.
Megabyte, México, 1996.

Elmasri. Navate.
Sistemas de bases de datos conceptos fundamentales.
Addison Wesley, E.U.A., 1997.

Fairley, Richard.
Ingeniería de software.
Mc Graw Hill, México, 1996.

Fraterth, Harold.
El gran libro de la multimedia con cd-rom.
Marcombo, 1995.

Kendall, Kenneth E.
Análisis y diseño de sistemas.
Prentice Hall, México, 1996.

Martin, James.
Organización de las bases de datos.
Prentice Hall, México, 1996.

Pereña, Jaime.
Dirección y gestión de proyectos.
Díaz de Santos, México, 1995.

Pressman, Roger S.
Ingeniería de software, un enfoque práctico.
McGraw Hill, México,1995.

Sheldon, Tom.
Novell netware 4 manual de referencia.
Mc Graw Hill, España, 1994.

Sommerville, Ian.
Software engineering.
Adison-Wesley Iberoamericana, 1996.

Stoltz, Kevin.
Todo acerca de redes de computación
Prentice Hall, México, 1996.

Vaughan, Tay.
Todo el poder de multimedia.
Mc Gral Hill, México, 1996.

Yourdon, Edward.
Anállsis estructurado moderno.
Prentice Hall, México, 1995.

Anexos

ANEXOS

ANEXO A. CUESTIONARIO QUE SE APLICO A LOS USUARIOS EN LAS ENTREVISTAS DURANTE LA ETAPA DE ANALISIS.

El siguiente cuestionario fue aplicado a usuarios de diferentes niveles y de las áreas correspondientes, se incluye en cada pregunta la respuesta más representativa.

También se agrupan las preguntas de acuerdo al proceso al que corresponden.

INVENTARIO DE INMUEBLES.

Registro de datos generales.

1. Especificar todos los datos generales para registrar un inmueble.

   Tipo de inmueble, ubicación (localidad, entidad, municipio), propietario, notario avalúos, datos de boleta predial, área de construcción, componentes, conjunto.

2. ¿Qué tipos de inmuebles se manejan?

   - Casas de diversos tipos (habitación, visitas, etc.).
   - Edificios.
   - Departamentos.
   - Hoteles.

3. ¿Cuántos inmuebles se manejan aproximadamente?

   Aproximadamente 1000.

4. Definir los inmuebles propios, administrados y consignados.

   Se van a manejar únicamente los inmuebles propios.

5. ¿Cómo se administran los inmuebles?

   La administración de los inmuebles consiste en una serie de procesos en los que están involucradas diversas áreas de la empresa.

6. ¿Es posible dar de baja un inmueble?

   Si, cuando se vende.

7. ¿Cuál información legal debe registrarse?

   Anexo con todos los datos.

Registro de antecedentes.

8. ¿Qué datos deben registrarse en el registro de antecedentes?, especificar todos.

   Nombre del dueño anterior, escrituras del inmueble, valor catastral, uso de suelo, período de ocupación, notario que expidió la escritura, no. de notaría, estado del inmueble.

Registro de gastos.

9. ¿A qué gastos se refiere el Registro de Gastos? (aclarar si son gastos relacionados con mantenimiento u otro tipo de gastos, aclarar también a qué impuestos y servicios se refiere).

   Se refiere a todos gastos necesarios para habilitar el inmueble, tales como predial, agua, luz y/o reparaciones que en su caso sean requeridas para el perfecto funcionamiento del inmueble como: pintura, cambiar chapas, revisión de tuberías y cableado eléctrico, etc.

Actualización de inmuebles.

10. ¿Cuáles son los datos que pueden modificarse al actualizar un inmueble?

Todos los datos generales.

Consulta de datos del inmueble.

11. Definir criterios de búsqueda para las consultas.

Los inmuebles podrán consultarse por:

- Un inmueble específico.
- Tipo de inmueble.
- Status del inmueble.

12. Definir qué datos deben mostrarse para las consultas (pantalla) y para los reportes (papel).

Deberán incluirse los datos generales, componentes, antecedentes y los gastos.

PROGRAMACION DE MANTENIMIENTOS.

13. ¿Qué es un programa de mantenimiento?

Son los trabajos de servicios necesarios para cuidar el inmueble, programado en las fechas que se requieren o les corresponda, por ejemplo: pintura, impermeabilización, etc.

14. Definir los datos necesarios para registrar un programa de mantenimiento.

Contar con un catálogo de conceptos, un catálogo de partidas, unidades, cantidades, precios unitarios, importes y fechas.

15. ¿Existe un catálogo de claves de partidas y conceptos de mantenimiento?

Sí.

16. ¿Quién realiza el catálogo y qué criterio se utiliza?

La empresa se encargará de actualizar los catálogos conforme lo requiera, para lo cual deberá definirse un responsable y el criterio existente por el momento es asignar claves diferentes a cada elemento de los catálogos.

17. ¿Cuáles son las partidas que se manejan?

Actualmente existe un catálogo de partidas.

18. ¿Cuáles son los conceptos de mantenimiento?

También existe un catálogo de conceptos.

19. ¿Se les proporciona mantenimiento a todos los inmuebles (propios, dados en administración o dados en consignación para venta)?

Sí.

20. ¿En qué consiste el mantenimiento global?

Se refiere al mantenimiento que se le proporciona a un grupo de inmuebles, ya sea por colonia o por conjunto.

21. ¿En qué consiste el mantenimiento individual?

Es el mantenimiento que se le proporciona a un solo inmueble.

Estimación de costos y tiempos de mantenimiento.

22. ¿Cómo se asignan los contratos por mantenimiento?

Por concurso.

23. ¿Es necesario registrar la licitación?

Sí.

24. Definir qué se va a capturar.

Para registrar un programa de mantenimiento deben capturarse:

Datos generales:

Concepto del programa de mantenimiento, ubicación, contratista, No. de contrato, importe del contrato, fecha del contrato, número de estimación, fecha contrato, período que abarca.

Conceptos:

No. de concepto, descripción, unidad, precio unitario.

25. ¿Cómo se calcula el costo del mantenimiento por la zona geográfica?

El sistema no va a calcular los costos por la zona, simplemente va a registrar un programa de mantenimientos estimado por la empresa, otro estimado por el contratista y otro con los avances reales del programa.

26. *¿Cómo se reportan los avances?*

Deberán alimentarse los avances de un programa de mantenimiento en el módulo de programas de mantenimiento reales.

30. ¿Qué son costos base y costos unitarios?

Costo base es el costo estimado de un concepto calculado por la empresa y el segundo es el costo por unidad de medida de cualquier concepto.

ANEXO B. EL DICCIONARIO DE DATOS.

A continuación se presentan para cada tabla de la base de datos los campos con sus características: nombre del campo, descripción, tipo y si es o no requerido.

Tabla Inmueble.

Cve_Inmueble	Identificador del inmueble Entero Requerido
Cve_Clasif	Identificador de la clasificaciòn del inmueble Entero Requerido
Cve_Localidad	Identificador de la localidad donde está ubicado el inmueble Entero Requerido
Cve_Estado	Identificador del estado donde está ubicado el inmueble Entero Requerido
Cve_Tipo	Identificador del tipo de inmueble Entero Requerido
Descrip_Inmueble	Descripción completa del inmueble Texto Requerido
Ubicación	Ubicación del inmueble Texto No Requerido
Colonia	Colonia donde se encuentra el inmueble Texto No Requerido
Arrendado	Indica si el inmueble está arrendado Lógico Requerido Valor por omisión: Falso
Precio_Vta	Precio de venta del inmueble Moneda No requerido
Archivo_Foto	Indica si se tiene fotografía del inmueble Lógico Requerido Valor por omisión: Falso
Archivo_Plano	Indica si se tiene fotografía del plano del inmueble Lógico Requerido Valor por omisión: Falso
Fecha_Plano_Localiza	Fecha del plano del inmueble Fecha Requerido Valor por omisión: Fecha del sistema

Uso_Suelo_Act	Uso de suelo actual del inmueble Texto No Requerido
Potenc_Uso_Futuro	Uso futuro potencial del inmueble Texto No Requerido
Sup_Desplante	Superficie del área que se va a construir Doble No Requerido
Sup_AreaLibre	Superficie del área sin construir Doble No Requerido
Sup_Tot_Polig	Superficie total poligonal que forma el inmueble Doble No Requerido
Apeo_Deslinde	Indica si se han definido áreas para banquetas Lógico Requerido Valor por omisión: Falso
Fecha_Apeo_Deslinde	Fecha en que se definió el apeo Fecha Requerido Valor por omisión: Fecha del sistema
Tipo_Construc	Especifica si el tipo de construccón es permanente o provisional Texto Requerido Valor por omisión: Permanente
Sistema_Construc	Especifica si el sistema constructivo es concreto, metálico, mixto o prefabricado Texto Requerido Valor por omisión: Concreto
Estado_Conservac	Especifica si el estado de conservación es excelente, bueno, regular, malo o defectuoso Texto Requerido Valor por omisión: Excelente
Dictamen_Estruc	Indica si se tiene documento donde quedan establecidas las condiciones de los esfuerzos a los que esta sometida la estructura Lógico Requerido Valor por omisión: Falso
Fecha_Dictamen_Estruc	Fecha en que se definió el dictamen estructural Fecha Requerido Valor por omisión: Fecha del sistema
Dictamen_Instalac	Indica si se tiene documento donde quedan establecidas las condiciones en que se encuentran las instalaciones Lógico Requerido Valor por omisión: Falso
Fecha_Dictamen_Instalac	Fecha en que se definió el dictamen de las instalaciones Fecha Requerido Valor por omisión: Fecha del sistema

Antig_Construc	Antigüedad de la construcción Doble No requerido
Inmueble_Ruta_Plano	Ruta donde se encuentra el archivo del plano del inmueble Texto No requerido
Inmueble_Ruta_Foto	Ruta donde se encuentra el archivo de la fotografía del inmueble Texto No requerido

Tabla Legal.

Cve_Legal	Identificador de la clave de datos legales del inmueble Entero Requerido
Cve_Inmueble	Identificador del inmueble Entero Requerido
Predio	Indica si es un terreno Lógico Requerido Valor por omisión: Falso
Edificación	Indica se existe alguna construcción Lógico Requerido Valor por omisión: Falso
Prop_adq	Forma en la cual se adquirió el inmueble Texto No requerido
Sup_Terreno	Dimensiones del terreno Doble No requerido
Sup_Construida	Dimensiones construidas Doble No requerido
Colindancias	Indica con que limita el inmueble Texto No requerido
Valor_Adq	Valor por el cual se adquirió el inmueble Moneda No requerido
Reg_bien_nac	Indica si existe el registro de bienes nacionales Lógico Requerido Valor por omisión: Falso
Num_Reg_Bien_Nac	Número del registro de bienes nacionales Entero No requerido
Reg_Publ_Prop	Indica si existe el registro publico de la propiedad Lógico Requerido Valor por omisión: Falso

Num_Reg_Publ_Prop	Número del registro público de la propiedad Entero No requerido
Reg_Catastro	Indica si existe el registro de catastro Lógico Requerido Valor por omisión: Falso
Num_Reg_Catastro	Número del registro de catastro Entero No requerido
Boleta_Predial	Numero de la boleta predial Texto No requerido
Certif_Lib_Grav	Indica si existe certificado de libertad de gravamen Lógico Requerido Valor por omisión: Falso
Num_Certif_Lib_Grav	Número del certificado de libertad de gravamen Entero No requerido
Certif_No_Adeudo	Indica si existe certificado de no adeudo Lógico Requerido Valor por omisión: Falso
Num_Certif_No_Adeudo	Número del certificado de no adeudo Entero No requerido
Boleta_Agua	Número de la boleta del agua Entero No requerido
Valor_Estimado	Valor estimado del inmueble Moneda No requerido
Aval_Cabin	Valor del avalúo proporcionado por CABIN Moneda No requerido
Vig_Aval_Cabin	Tiempo de duración del avalúo dado por CABIN Doble No requerido
Lic_Subdiv	Indica si pueden existir subdivisiones en el inmueble Lógico Requerido Valor por omisión: Falso
Lic_Constr	Indica si se tiene permiso para construir Lógico Requerido Valor por omisión: Falso
Lic_UsoSuelo	Indica si se tiene permiso al uso de suelo de la construcción Lógico Requerido Valor por omisión: Falso
Reporte_Fotog	Indica si existe un reporte fotográfico Lógico

	Requerido Valor por omisión: Falso
Memoria_Descrip	Indica si existe un escrito donde se especifica la forma y el contenido de como esta hecha la construcción Lógico Requerido Valor por omisión: Falso

Tabla Localidad.

Cve_Localidad	Identificador de la localidad Entero Requerido
Descrip_Localidad	Descripción de la localidad Texto Requerido

Tabla Estado.

Cve_Estado	Identificador del estado Entero Requerido
Descrip_Estado	Descripción del estado Texto Requerido

Tabla Tipo_Inmueble.

Cve_Tipo	Identificador del tipo de inmueble Entero Requerido
Descrip_Tipo	Descripción del tipo de inmueble Texto Requerido

Tabla Ciudad.

Cve_Estado	Identificador del estado donde se encuentra la ciudad Entero Requerido
Cve_Ciudad	Identificador de la ciudad Entero Requerido
Nom_Ciudad	Nombre de la ciudad Texto Requerido

Tabla Clasificacion_Inmueble.

Cve_Clasif	Identificador de la clasificación del inmueble Entero Requerido
Descrip_Clasif	Descripción del inmueble Texto Requerido

Tabla Imagen_Inmueble.

Imagen_Inmueble_Id	Identificador de la imagen del inmueble Entero Requerido
Cve_Inmueble	Identificador del inmueble Entero Requerido
Imagen_Inmueble_Ruta	Ruta del directorio donde se encuentra la imagen del inmueble Texto Requerido

Tabla Gasto_Inmueble.

Cve_Gasto_Inm	Identificador del gasto del inmueble Entero Requerido
Cve_Inmueble	Identificador del inmueble Entero Requerido
Cve_Gasto	Identificador del catálogo de gastos Entero Requerido
Monto_Gasto	Monto que causo el gasto Moneda Requerido
Fecha_Gasto	Fecha en que se generó el gasto Fecha Requerido Valor por omisión: Fecha del sistema
Factura_Gasto	Número de la factura que soporta el gasto Texto Requerido
Gasto_Comentario	Comentarios acerca del gasto Texto No requerido
Gasto_Fecha_Registro	Fecha en que se registró el gasto Fecha Requerido Valor por omisión: Fecha del sistema

Tabla Gasto.

Cve_Gasto	Identificador del gasto
	Entero
	Requerido
Descrip_Gasto	Descripción del gasto
	Texto
	Requerido

Tabla Componente_Inmueble.

Cve_Componente	Identificador del componente del inmueble
	Entero
	Requerido
Cve_Inmueble	Identificador del inmueble
	Entero
	Requerido
Cve_Comp_Gen	Identificador del catálogo de componentes
	Entero
	Requerido
Cant_Cap	Cantidad o capacidad del componente
	Doble
	No requerido
Area	Area del componente
	Doble
	No requerido
Estado	Estado del componente
	Texto
	No requerido
Fecha_Registro	Fecha de registro del componente
	Fecha
	Requerido
	Valor por omisión: Fecha del sistema
Componente_Inmueble_Observacion	Observaciones acerca del componente
	Texto
	No requerido
Componente_Ascendente	Indica de qué componente forma parte
	Entero
	Requerido

Tabla Historia_Inmueble.

Cve_Inmueble	Identificador del inmueble
	Entero
	Requerido
Cve_Componente	Clave del componente
	Entero
	Requerido
Cant_Cap	Cantidad o capacidad del componente
	Doble
	No requerido

Area	Area del componente Doble No requerido
Estado	Estado del componente Texto No requerido
Fecha_Registro	Fecha de registro del componente Fecha Requerido Valor por omisión: Fecha del sistema
Tipo_Movimiento	Tipo de movimiento del componente, indica si se dió de baja o modificó Texto Requerido Valor por omisión: B = Baja

Tabla Componente.

Cve_Comp_Gen	Identificador del componente Entero Requerido
Descrip_Comp_Gen	Descripción del componente Texto Requerido

Tabla Prog_Mant.

Cve_ProgMant	Identificador del programa de mantenimiento Entero Requerido
Partida_Clave	Identificador de la partida del programa de mantenimiento Entero Requerido
Cve_Contrat	Identificador del contratista Entero Requerido
Cantidad_Mant	Cantidad a la cual se la va a dar mantenimiento Doble Requerido
Unidad_Mant	Unidad de mantenimiento(L, M, M2) Texto No requerido
PU_Empresa	Precio unitario del concepto propuesto por la empresa Moneda Requerido
Fecha_Ini_Empresa	Fecha de inicio del concepto propuesta por la empresa Fecha Requerido Valor por omisión: Fecha del sistema
Duracion_Empresa	Duración del concepto propuesto por la empresa Doble Requerido

PU_Contrat	Precio unitario del concepto propuesto por el contratista Moneda Requerido
Fecha_Ini_Contrat	Fecha de inicio del concepto propuesta por el contratista Fecha Requerido Valor por omisión: Fecha del sistema
Duracion_Contrat	Duración del concepto propuesto por el contratista Doble Requerido
PU_Real	Precio unitario del concepto real Moneda No requerido
Fecha_Ini_Real	Fecha de inicio del concepto real Fecha Requerido Valor por omisión: Fecha del sistema
Duracion_Real	Duración del concepto real Doble No requerido
Avance_Real	Avance del concepto en unidades Doble No requerido

Tabla Partida.

Partida_Clave	Identificador de la partida Entero Requerido
Partida_Descripcion	Descripción de la partida Texto Requerido
Partida_Ascendente	Indica de que partida forma parte Entero Requerido

Tabla Cat_ProgMant.

Cve_ProgMant	Identificador del programa de mantenimiento Entero Requerido
Cat_ProgMant_Desc	Descripción del programa de mantenimiento Texto Requerido
Tipo_Mant	Tipo de mantenimiento Texto Requerido

Tabla Inmueble_Mantenimiento.

Cve_Inmueble	Identificador del inmueble Entero Requerido
Cve_ProgMant	Identificador del programa de mantenimiento Entero Requerido

Tabla Contratista.

Cve_Contrat	Identificador del contratista Entero Requerido
Cve_Ciudad	Identificador de la ciudad donde reside el contratista Entero Requerido
Cve_Estado	Identificador del estado donde reside el contratista Entero Requerido
Nom_Contrat	Nombre del contratista Texto Requerido
Domic_Contrat	Domicilio del contratista Texto No requerido
Telef_Contrat	Número de teléfono del domicilio del contratista Texto No requerido
Fax_Contrat	Número fax del contratista Texto No requerido
CP_Contrat	Código postal del domicilio del contratista Texto No requerido

Tabla Usuario.

Usuario_Clave	Identificador del usuario Entero Requerido
Usuario_Nombre	Nombre largo del usuario Texto Requerido
Usuario_NombreCorto	Nombre corto del usuario Texto Requerido
Usuario_Password	Clave del usuario para accesar al sistema Texto Requerido

Tabla Menu.

Menu_Clave	Identificador del menú Entero Requerido
Menu_Descripcion	Descripción del menú Texto Requerido
Menu_Nivel	Indica el nivel de profundidad del menú Texto Requerido

Tabla Permiso.

Permiso_Clave	Identificador del permiso Entero Requerido
Usuario_Clave	Identificador del usuario Entero Requerido
Menu_Clave	Identificador del menú Entero Requerido

Tabla Tabla_Nombre.

Tabla_Nombre_Clave	Identificador de la tabla Entero Requerido
Tabla_Nombre_Nombre	Nombre de la tabla Texto Requerido

Tabla Bitacora.

Bitacora_Clave	Identificador de la bitacora del usuario Entero Requerido
Bitacora_Fecha	Fecha en que se efectuó alguna operación del usuario Fecha Requerido Valor por omisión: Fecha del sistema
Tabla_Nombre_Clave	Tabla a la que acceso el usuario Entero Requerido
Usuario_Clave	Identificador del usuario Entero Requerido
Bitacora_Accion	Acción realizada por el usuario Texto Requerido

Bitacora_Campo	Campo donde se efectuó alguna operación Texto Requerido
Bitacora_Hora	Hora en la que se realizó la acción Fecha/Hora Requerido Valor por omisión: Fecha y hora del sistema

Tabla Tabla_Campo.

Tabla_Campo_Id	Identificador del campo de la tabla Entero Requerido
Tabla_Nombre_Clave	Nombre del campo Entero Requerido
Tabla_Campo_DesCorta	Descripción corta del campo Texto Requerido
Tabla_Campo_DesLarga	Descripción larga del campo Texto Requerido
Tabla_Campo_Ascendente	Indica a que tabla pertenece el campo Entero Requerido