30
2ej

UNIVERSIDAD NACIONAL
AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

CASE Y SISTEMAS DE INFORMACION

TESIS PROFESIONAL
Que para obtener el Titulo de
INGENIERO EN COMPUTACION
presenta

TELMA GALLO SANCHEZ
JAIME RODRIGUEZ DIAZ

Dir. de Tesis:
INGENIERO CRISTOBAL PEÑA

México, D. F. 1993.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

CONTENIDO

I.INTRODUCCION.

En un principio los sistemas se desarrollaban de una forma en la cual el programador, era el único capaz de corregir y modificar dicho sistema, su comunicación con los usuarios era mínima, por lo cual realizaba el sistema según lo que él creía se le había pedido.

En infinidad de ocasiones los sistemas no brindaban al usuario los resultados deseados exigiendo de inmediato modificaciones iniciándose una larga etapa de mantenimiento que abarcaba a veces más de la mitad del tiempo de vida del sistema, dando lugar a sistemas muy complejos y de una gran extensión.

Dichos sistemas no contaban con un metodología mediante la cual se buscara tener un buen programa; su principal objetivo era que funcionara, puesto que muchas veces el tiempo con el que se contaba para su desarrollo era mínimo y se exigían resultados casi inmediatos.

Se vislumbró la necesidad de implementar una serie de reglas que guiaran el desarrollo de un sistema, para que éste fuera consistente, confiable y fácilmente comprensible por otra persona que no fuese su programador original.

Los programadores comenzaron a introducir una serie de técnicas y metodologías cuyo fin era lograr una estructura más coherente y confiable.

Este análisis servirá como punto de partida para el desarrollo de un sistema utilizando herramientas que la tecnología actual nos proporciona, como son bases de datos, interfases amigables y herramientas CASE.

Además se intenta realizar una discusión sobre las diversas técnicas estructuradas que deben de utilizarse para desarrollar sistemas con buena calidad, de fácil modificación, reducir su costo de mantenimiento, obtener un buen tiempo de desarrollo, lograr tener un mejor control de flujo y por todo lo anterior, un menor costo del sistema final.

Todas estas técnicas son base fundamental de la filosofía tomada por una serie de herramientas llamadas de cuarta generación, que contemplan todas las características antes mencionadas para el desarrollo de sistemas.

Estas son denominadas herramientas CASE (Computer Aided Systems Enginnering) cuyo término se mencionó anteriormente y facilitan el desarrollo de un sistema evitando un largo y tedioso trabajo a mano, volviéndolo sencillo y rápido con el uso de una computadora.

Se comentarán las diferentes técnicas de diagramación que se usan en estas herramientas (pues son herramientas gráficas), se realizará un breve bosquejo histórico de sus orígenes y hará una comparación entre diversas herramientas CASE comerciales existentes, mencionando características importantes de cada uno de éstos.

El uso de técnicas gráficas para el modelaje de datos han ayudado a crear y representar de una manera más real la información que se maneja en una empresa, evitando así ambigüedades de textos narrativos y explicaciones muy largas.

Finalmente, mediante una de estas herramientas en específico procederemos al desarrollo de un sistema abarcando sus diferentes etapas por las cuales debe atravesar para el logro de un sistema de buena calidad y posteriormente se procederá a implementarlo en un lenguaje bajo ambiente Windows.

II. METODOLOGIAS DE PROGRAMACION.

En los primeros días, la programación se veía como un arte. Existían pocos métodos formales y pocas personas los usaban. El programador aprendía normalmente su oficio mediante prueba y error. El mundo del software era virtualmente indisciplinado y sigue siendo en alguna medida.

A finales de los 60's, Dijkstra y otros propusieron el uso de un conjunto de construcciones lógicas con las que podría formarse cualquier programa. Las construcciones reforzaban el "mantenimiento del dominio funcional" . Esto es, cada construcción tenía una estructura lógica predecible, se entraba a ella por el principio y se salía por el final, y facilitando al lector seguir de una manera más fácil el flujo procedimental.

OBJETIVOS PRIMARIOS.

Los objetivos primarios de las técnicas estructuradas son los siguientes:

-Incrementar la velocidad en el desarrollo de sistemas.

-Simplificar los programas y su desarrollo.

-Contar con programas de gran calidad y con un comportamiento predecible.

-Reducir el costo del desarrollo del sistema.

-Crear programas sencillos de modificar (de fácil mantenimiento).

OBJETIVOS SECUNDARIOS.

-Descomponer problemas complejos y construir otros de una manera más sencilla.

-Lograr la simplicidad de diseño. Un buen diseño usualmente permite interfaces limpias entre módulos simples; un mal diseño tiene modelos complejos de interacción.

-Control de Complejidad. Es importante dividir los diseños complejos de tal manera que el personal pueda manejar esta complejidad sin errores. El control de complejidad se puede lograr minimizando el número de interacciones entre módulos separados y estandarizando la estructura de control.

-Obtener una idea clara sobre el sistema a desarrollar.

-Uso de técnicas de diagramación tan claras como sea posible. Los buenos diagramas son de gran ayuda en la comprensión de problemas complejos.

-Incrementar la comunicación con los usuarios finales. Muchos sistemas fallan u operan pobremente porque no conocen los requerimientos de los usuarios.

-Lograr una unidad de arquitectura. Un buen diseño aumenta la limpieza y unifica la estructura. Los principios de diseño e implementación se aplican al sistema completo.

-Empleo consistente en los métodos de enseñanza. El análisis, el diseño, y las técnicas de programación necesitan ser fáciles de enseñar. Las técnicas deben ser dóciles para cursos de entrenamiento y aplicables a todo tipo de programas en una instalación.

-Empleo de estructuras de Control estándares que puedan ser convertidas a un código con un esfuerzo mínimo. Los lenguajes de cuarta generación facilitan la representación de tales estructuras de control con mucho menos trabajo que los lenguajes de tercera generación.

-Lograr una buena comunicación entre la gente que desarrolla el sistema. Uno de los más grandes problemas a los que se enfrentan los programadores es ésta comunicación. La técnica debe minimizar la necesidad de interacciones complejas entre los

miembros del equipo y complementar esta interacción a través de documentos descriptivos.

-Utilizar técnicas que trabajen tanto para sistemas grandes como para pequeños. Los grandes sistemas requieren mucho más control de interacciones entre los módulos.

-Minimizar errores. En general, se cometen pocos errores si las herramientas a utilizar están bien diseñadas.

-Detectar errores tan pronto como sea posible. Los errores implicarán un costo menor si son detectados en la etapa de especificaciones, más que durante el diseño; en la etapa de diseño más que en la etapa de codificación; o en la etapa de codificación y depuración más que en el mantenimiento.

En la primera década de las técnicas estructuradas, algunos de los objetivos antes mencionados no fueron cumplidos, por ejemplo, lograr una comunicación con los usuarios finales. El creciente énfasis en la participación del usuario final ha conducido a un mayor desarrollo en los lenguajes, prototipos y manejo de centros de información.

EVOLUCION.

Las Técnicas Estructuradas se han desarrollado desde una Metodología Codificada (Programación Estructurada) hasta llegar a Técnicas incluyendo análisis, diseño, y metodologías de prueba, asi como las herramientas para documentación y manejo de conceptos.

Las técnicas estructuradas fueron introducidas en los últimos años de los 60s y se comercializaron a principios de los 70s, después del trabajo de F.Terry Baker y Harlan Mills en IBM. Desde tal fecha, éstas técnicas han ganado gran popularidad y han causado un gran impacto en el arte de la programación.

La figura 1 lista algunas de las técnicas más conocidas y su evolución en el tiempo.

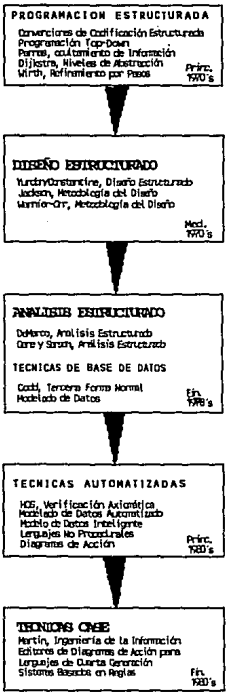


Figura 1. Evolución de las Técnicas Estructuradas dentro de la Tecnología CASE.

Programación Estructurada.

A principios de los 70s, las técnicas estructuradas se enfocaron al programa mismo. La estandarización de la forma fué la clave. La noción de estandarización fue aplicada a construir el control del programa y sus módulos.

Diseño Estructurado.

A mitad de los 70s la filosofía estructurada implementó la fase de diseño. La noción de estandarización fue aplicada a la solución de problemas en el proceso como una introducción de organización y disciplina en el diseño del programa. Además de que las técnicas estructuradas se concentraron en la visión de un nivel de instrucción detallado, se enfocaron también a utilizar módulos de programas como la estructura básica de diseño.

Análisis Estructurado.

Las técnicas estructuradas para análisis de sistemas y especificación de requerimientos se desarrollaron hasta finales de los 70s, rápidamente se difundió el uso de diagramas de flujo de datos.

Se introdujeron diferentes técnicas estructuradas, resultado de una composición de metodologías, estrategias y herramientas ofreciendo una técnica sistemática en el desarrollo de software.

Ingeniería de la Información.

La Ingeniería de Información se aplicó a las técnicas estructuradas no sólo a un sistema sino a la empresa completa. El fin es crear un esquema en el cual se desarrollen sistemas por separado.

Técnicas Automatizadas.

A principios de los 80s, existía una crisis en la programación por la poca productividad; las computadoras y particularmente las microcomputadoras se desarrollaron rápidamente y su costo fué reduciéndose considerablemente.

Las computadoras eran empleadas para crear, editar, expander, y cambiar los diagramas estructurados. Se utilizaban como directorios y enciclopedias como ayuda al programador.

Un problema con los lenguajes de programación tradicionales es que permiten a los programadores crear estructuras que posteriormente no pueden ser validadas.

FILOSOFIA DE LAS TECNICAS ESTRUCTURADAS.

La filosofía básica de las técnicas estructuradas logró formalizar las buenas prácticas de la programación que se aplican en muchos aspectos de desarrollo de sistemas más que en la misma programación.

La filosofía estructurada original está compuesta de las siguientes estrategias para solución de problemas:

-Principio de Abstracción.
-Principio de Formalidad.
-Concepto de División y Conquista.
-Concepto de Orden Jerárquico.

Principio de Abstracción.

Sin el principio de abstracción, quizás la evolución en la programación estructurada no se hubiera dado. La abstracción es la concepción o visión de algo separado de su realidad; es una simplificación de factores describiendo lo que se está haciendo sin explicar cómo se esta haciendo. La abstracción nos permite imaginar la solución de un problema sin ser inmediatamente restringido con detalles irrelevantes de la realidad.

La abstracción es la mejor herramienta que se ha encontrado para manejar la complejidad que implica la cantidad de piezas interrelacionadas, y utilizando este principio podemos ver un programa en niveles.

Cada nivel de abstracción esta compuesto de un grupo de funciones que están lógicamente relacionados. Las

funciones pertenecientes al mismo nivel y a diferentes niveles se pueden comunicar por dos reglas especificas:

1.Los niveles más bajos no pueden llamar a niveles más altos; pero por el contrario, los niveles más altos pueden invocar a niveles más bajos para desarrollar alguna tarea.

2.Cada nivel tiene sus propios datos, los cuales pertenecen exclusivamente a funciones de cada nivel y no pueden ser accesados por funciones de otro nivel. Los datos son explícitamente pasados de un nivel a otro.

Principio de Formalidad.

El segundo principio fundamental de la filosofía estructurada es el principio de Formalidad. La palabra formalidad sugiere una técnica rigurosa y metódica. Esto es lo que implica precisamente en la filosofía estructurada; significa que el desarrollo del programa es guiado por procedimientos sistemáticos cuyo resultado es consecuencia de cada uno de los procedimientos que han sido rigurosamente definidos, y que una evaluación paso a paso nos determina si un procedimiento ha sido ejecutado correctamente y si el resultado producido es el correcto.

El ciclo de vida del software es un ejemplo de la aplicación de este principio de formalidad. Este ciclo de vida define el proceso de la programación como una secuencia de fases como podemos observar en la figura 2.2; para cada fase del ciclo se definen detalladamente procedimientos paso por paso.

Concepto de División y Conquista.

Este concepto implica la solución de problemas complejos con su división en problemas más pequeños, independientes y que a su vez sean fáciles de entender y resolver.

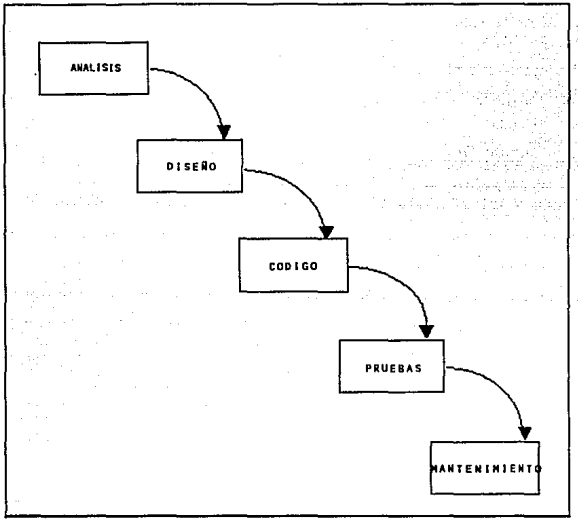


Figura 2.2.El ciclo de vida del Software como un proceso multipaso consistente de cinco fases secuenciales básicas: Análisis, Diseño, Codificación, Prueba y Mantenimiento.

Concepto de Orden Jerárquico.

Este concepto está estrechamente relacionado con el concepto de División y Conquista. Además de facilitar el entendimiento del problema dividiéndolo en partes más pequeñas, el arreglo de éstas es igualmente importante para su comprensión. El programa puede ser

comprendido y construido nivel por nivel, donde cada nivel tiene más detalles.

INGENIERIA DE SOFTWARE.

En los primeros días de la Informática, los sistemas basados en computadoras se desarrollaban usando técnicas orientadas al hardware. Los desarrolladores de proyectos se centraban en el hardware, debido a que era el factor principal. Para controlar los costos del hardware los responsables instituyeron controles formales y estándares técnicos. Exigían un análisis y diseño completo antes de que se construyera el hardware. Medían el proceso para determinar dónde podían hacerse mejoras. Dicho sencillamente, aplicaban los controles , métodos y herramientas que reconocemos como ingeniería del hardware. Desgraciadamente, el software era simplemente un añadido.

Hoy, la distribución de costos para el desarrollo de sistemas informáticos ha cambiado dramáticamente. El software, en vez del hardware, es normalmente el elemento principal del costo. En las pasadas décadas los ejecutivos y muchos practicantes técnicos se habían hecho las siguientes preguntas :

¿ Por qué lleva tanto tiempo terminar los programas?

¿ Por qué el costo es tan alto ?

¿ Por qué no podemos encontrar todos los errores antes de darle el software a nuestros clientes ?

¿ Por qué tenemos dificultad en medir el progreso conforme se desarrolla el software?

¿ Por qué el costo del mantenimiento es tan elevado?

Estas, y muchas otras preguntas, son una manifestación de las características del software y la

forma en que se desarrolla, un problema que ha llevado a la adopción de la práctica de la ingeniería del software como una disciplina para desarrollar software de alta calidad para sistemas basados en computadoras.

La Ingeniería de Software se podría definir como sigue :

"El establecimiento y uso de principios de ingeniería robustos, orientados a obtener económicamente software que sea fiable y funcione eficientemente sobre máquinas reales".

La Ingeniería de Software abarca un conjunto de tres elementos claves - métodos, herramientas y procedimientos - que facilitan al gestor controlar el proceso del desarrollo del software y suministrar a quienes practiquen dicha disciplina las bases para construir software de alta calidad de una forma práctica.

Los métodos de la Ingeniería de Software suministran el "cómo" construir técnicamente el software. Estos abarcan las tareas de planificación y estimación de proyectos; análisis de los requerimientos del sistema y del software; diseño de estructuras de datos, arquitectura de programas y procedimientos algorítmicos; codificación; prueba y mantenimiento.

Las herramientas de la Ingeniería del Software suministran un soporte automático o semiautomático para los métodos. Hoy existen heramientas para soportar cada uno de los métodos mencionados anteriormente; una de éstas herramientas son los CASE, que nos sirven para guardar la información relevante sobre el análisis y diseño, entre otras cosas.

Los procedimientos de la Ingeniería del Software definen la secuencia en la que se aplican los métodos, las entregas (documentos, informes, formas, etc) que se requieren , los controles que ayudan a asegurar la calidad y coordinar los cambios, y las guías que facilitan a los gestores del software establecer su desarrollo.

El proceso de desarrollo del software contiene tres fases genéricas: definición, desarrollo y mantenimiento.

La fase de definición se enfoca sobre el "qué" y abarca los siguientes pasos : Análisis del sistema, planificación del proyecto de software y análisis de requerimientos.

La fase de desarrollo se enfoca sobre el "cómo", y se aplican los tres pasos concretos siguientes : Diseño del Software, Codificación y Prueba del Software.

La fase de mantenimiento se enfoca sobre el cambio que va asociado con una correción de errores, adaptaciones requeridas por la evolución del entorno del software y modificaciones debidas a los cambios de los requerimientos del cliente para reforzar o aumentar el sistema. En esta fase se encuentran tres tipos de cambios : Corrección , Adaptación y Aumento.

PRINCIPIOS BASICOS DE INGENIERIA DE SOFTWARE.

Algunas veces es difícil distinguir entre la Ingeniería del Software y las Técnicas Estructuradas. La Ingeniería del Software es el estudio de los principios y su aplicación al desarrollo y mantenimiento de sistemas. Esta ingeniería de software y las técnicas estructuradas son ambas colecciones y herramientas de metodologías de Software.

La Ingeniería de Software incluye además cuatro principios.

- Principio de Ocultamiento.
- Principio de Localización.
- Principio de Integridad Conceptual.
- Principio de Completitividad.

Principio de Ocultamiento.

El principio de ocultamiento se utiliza para guiar el proceso de descomposición funcional. Para tener control sobre la complejidad de un problema debe ser

modularizado, y las interfases entre los módulos deberán ser tan simples como sea posible.

Principio de Localización.

El Principio de localización está relacionado con la agrupación lógicamente relacionada con módulos físicos. Esto se aplica a datos y procesos; los arreglos, registros, subrutinas y procedimientos son todos aplicaciones del principio de localización.

Principio de Integridad Conceptual.

El Principio de Integridad Conceptual es quizás el principio más importante de la Ingeniería de Software; sigue una filosofía de diseño y arquitectura consistente. Esto proporciona la cualidad de entendimiento más que cualquier otro factor.

Principio de Completitividad.

La completitividad implica que se ha incluido todo y que nada se ha olvidado. El principio de completitividad direcciona su atención a la importancia de revisar que un sistema conozca completamente todos las necesidades y requerimientos, por ejemplo, sus datos y funciones.

El Ambiente de Base de Datos.

Mientras los principios de la programación estructurada y el diseño de procesos fueron surgiendo, otro grupo dentro de la computación también apareció: las bases de datos y los sistemas de información.

Anteriormente, los datos tenían una estructura inherente que era independiente de cómo eran procesados. Un sistema de manejo de base de datos permite que muchos procesos diferentes utilicen los mismos datos y tengan diferentes visiones de los mismos datos.

Esto derivó el **Principio de Independencia Lógica de Datos:** Los modelos de datos representando la estructura

lógica inherente de datos, debe ser formalmente designada e independiente de las técnicas de acceso físico y distribución de datos.

En una buena instalación de procesamiento de datos, tales modelos de datos serán las herramientas fundamentales del procesamiento de datos. Muchos programas utilizados manejan los mismos modelos de datos. Los programadores y analistas utilizaban un diccionario de datos común. Además, el administrador de datos, el cuál era independiente de los proyectos individuales, era el custodio de los modelos de datos y del diccionario. Era necesario considerar el análisis de los datos formalmente antes de que los programas fueran diseñados. Esto es, una tarea básica, rigurosa y nada difícil para el analista de sistemas.

La planeación estratégica de los datos en toda una empresa se hizo una necesidad. La planeación, la definición y estructuración eran indispensables para el intercambio de datos entre procesos así como su manejo para obtener la información más importante.

A partir de esta serie de ideas y las metodologías para la implementación de sistemas, surge la ingeniería de información.

INGENIERIA DE LA INFORMACION.

La Ingeniería de Software aplica técnicas estructuradas a un proyecto. La Ingeniería de Información aplica técnicas estructuradas a la empresa como un todo o a grandes sectores de ésta.

Ya que la empresa es muy compleja, la planeación, análisis, diseño y construcción, no pueden llevarse a cabo sin la ayuda de herramientas automatizadas. La Ingeniería de Información se ha definido con respecto a las técnicas automatizadas, como sigue :

" Un conjunto entrelazado de técnicas automatizadas en las cuales los modelos de la empresa, los modelos de

datos y modelos de procesos se construyen sobre una base de conocimientos inteligente y se usan para crear y mantener los sistemas de procesamiento de datos."

En el procesamiento de datos tradicional, los sistemas se construían independientemente. Eran incompatibles unos con otros, tenían datos incompatibles y se podían ligar con mucha dificultad. Algunas empresas tienen cientos de aplicaciones incompatibles, todas difíciles y caras de mantener y además con necesidades de información no satisfechas.

Con la Ingeniería de Información se crean planes y modelos de alto nivel y se construyen sistemas separados de acuerdo a estos planes y modelos. Particularmente importante son los modelos de datos de las áreas de negocios. Asociados con ésto está un modelo de los procesos en cada área. Los modelos representan la estructura que se representa en la computadora.

Es importante remarcar que a la base sobre la cual descansa la Ingeniería de Información, es al modelo de Datos. La representación lógica de los datos se puede diseñar de tal forma que sea estable. Lo anterior será la llave para soportar otros elementos de la Ingeniería de Información tales como los procesos. Estos, al contrario de los datos, son menos estables, por tanto más susceptibles a cambios.

Las fases de la Ingeniería de la Información son las siguientes :

Planeación Estratégica de Información.

Análisis del Area de Negocios.

Diseño de Sistemas.

Construcción.

A continuación se mencionan los Principios básicos de la Ingeniería de Información. Estos principios convergen con los principios de la filosofía estructurada así como los principios de la Ingeniería de Software.

Principio de Análisis riguroso de datos.

Los datos tienen una estructura inherente. El análisis de datos, el cuál identifica formalmente ésta estructura, se debe realizar antes de que se designe el proceso lógico.

Principio de Independencia de Datos.

La representación de modelos de datos a la estructura lógica debe ser formalmente designada, independientemente de la manera en que los datos van a ser utilizados e independientemente de la estructura física y distribución de los datos.

Principio de Planeacion Estratégica de Datos.

Los datos requieren una planeación, definición y estructuración de tal manera que éstos puedan ser intercambiados entre los procesos.

Principio de Acceso del usuario final.

Los usuarios finales deben proporcionar herramientas para accesar a las bases de datos que ellos mismos utilicen, sin programar.

En general, éstos principios se aplican al diseño y especificación de los procesos más que la codificación del programa.

Automatización del Análisis, Diseño y Codificación.

La programación es una de las labores más intensas y propensas a errores humanos. Un principio de la automatización del diseño debe ser que la programación debe ser protegida tanto como sea posible aplicando los siguientes principios:

Gráficos.

El diseño asistido por computadora requiere de buenas gráficas. El analista y el usuario final deben emplear los diagramas de los sistemas que sean tan significativos y de fácil entendimiento como sea posible.

Manejo de Complejidad.

El reto para muchos que desarrollan alta tecnología es el manejo de la complejidad. Esto se aplica al futuro del software, diseño de chips, redes de computadoras, robots y muchas otras áreas.

Cambios en los Lenguajes de Computación.

El diseñador de chips se vio en la necesidad de emplear nuevos "lenguajes" a medida que el manejo de la complejidad crecía; el programador también necesitaba adoptar nuevos lenguajes.

A principios de los 70s, se observó un flujo de nuevos lenguajes conocidos como de Cuarta Generación, cuyo objetivo era obtener calidad en los resultados mucho más rápido que los lenguajes convencionales como COBOL, PL/I, FORTRAN y PASCAL. Algunos lenguajes de cuarta generación surgieron con la idea de simplificar la creación de ciertas aplicaciones de manera que los usuarios finales, más que los programadores, pudieran obtener los resultados deseados.

Ahora estos lenguajes de cuarta generación (o al menos la mayoría de ellos) permiten generar pantallas, reportes y bases de datos y actualizar datos con técnicas no procedurales.

El principio del diseño de automatización es que el lenguaje debe ser seguido de las técnicas de diseño.

Los siguientes principios se aplican al diseño de sistemas asistido por computadora recordando que no niegan los principios anteriores pero sí cuentan con ellos, alcanzando altos niveles de automatización.

Necesidad de Automatización con Técnicas de diseño poderosas.

El cerebro humano está limitado en su capacidad de manejar con detalle y sin errores, la complejidad y los cálculos matemáticos. Cuando una computadora está lista para esto, los diseñadores pueden ir más allá de los confines del cerebro humano.

Necesidad de Gráficos en el Diseño Asistido por Computadora.

Todos el análisis estructurado y su diseño requieren de gráficas. Una herramienta computarizada debe facilitar la construcción, edición, refinamiento, expansión y extracción de los diagramas de diseño de manera que la complejidad sea manejable.

Una herramienta computarizada debe guiar al diseñador.

Las herramientas CASE deben incorporar metodologías y guiar al diseñador por medio de secuencias que reflejen el buen diseño; además, debe solicitar al diseñador información con la cual sea capaz de generar el código.

La programación debe ser protegida tanto como sea posible.

Los diagramas de diseño se deben de descomponer dentro de otros con más detalle hasta que el código pueda ser generado automáticamente con ellos. Los reportes, bases de datos, diálogos, actualización de datos, etc., debe ser específico para la generación automática de código.

Los lenguajes deben ir junto con las técnicas de diseño.

Los nuevos lenguajes de computación (Cuarta Generación) están evolucionando de una manera muy rápida; es necesario, por ésto, emplear una construcción correcta para que las mejores técnicas de diseño automatizado sean reflejadas en el lenguaje.

Un principio vitalmente importante en el análisis y diseño de sistemas es la participación activa de los usuarios finales. Esto se puede lograr de muchas maneras:

1.Los usuarios finales deben de discutir los procedimientos que se estan llevando a cabo a fin de saber si les son útiles o no.
2.Los usuarios deben participar en la planeación de requerimientos (Joint requirements planning-JRP) y diseño de aplicación (Joint application design-JAD) en el comienzo del desarrollo del ciclo de vida.
3.Los usuarios deben de ser capaces de leer y criticar los diseños creados por el departamento de DP.

4.Los usuarios deben proporcionar herramientas con las cuales ellos mismos puedan crear sus aplicaciones.
5.Los usuarios deben tener un perfecto conocimiento del proceso de administración de datos, definición y modelado de datos compartidos.

Técnicas Estructuradas accesibles al usuario.

Para que los usuarios participen en el análisis, diseño y administración de datos, es muy recomendable emplear técnicas estructuradas que sean tan accesibles como sea posible. Para esto, se requieren buenas técnicas de diagramación. Los usuarios deben mencionar los procedimientos que requieren para poderlos discutir y modificar dentro del análisis del sistema

El Proceso de Diseño.

La figura 2.3 representa la manera en que las técnicas estructuradas pueden ser utilizadas. Primero, existe un bosquejo del proceso con la participación del usuario final. Los delineamientos pueden comenzarse en un pizarrón o papel para después ser transferidos a la pantalla de la computadora. En esta etapa, debe existir tanta interacción con el usuario como sea posible. Cuando el diseñador ha corregido todos los errores detectados, se genera y se prueba el código ejecutable.

PROGRAMACION ESTRUCTURADA.

A finales de los 60s era obvio que el software que se desarrollaba era muy complejo. Era difícil diseñarlo, escribirlo, examinarlo y virtualmente imposible entenderlo y mantenerlo. Con la necesidad de crear nuevas tecnologías, surge la programación estructurada, la cual es una metodología que permite estructurar y llevar una disciplina en la forma del programa, en su diseño, en la codificación y pruebas.

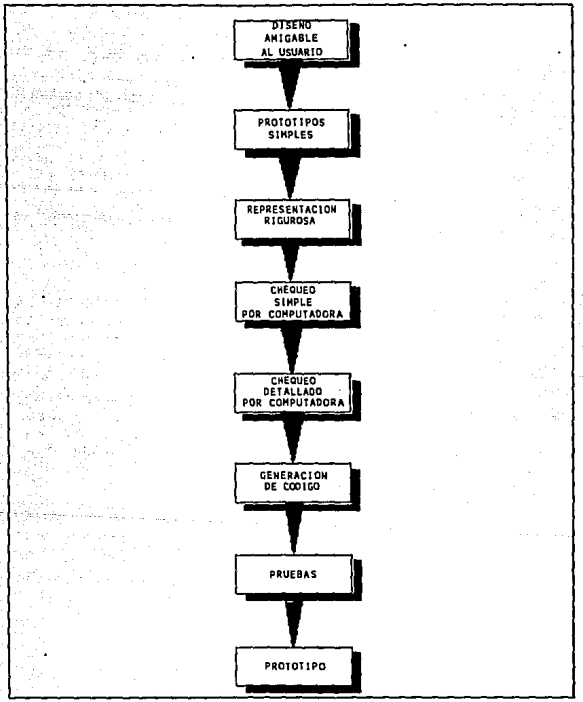


Figura 3. El Proceso de diseño con herramientas Computarizadas y con la participación del usuario.

El objetivo de la programación estructurada fue resolver esta crisis, proporcionando una disciplina de programación con las siguientes características:

-Realizar lecturas del programa.
-Realizar una correspondencia entre el programa fuente y el ejecutable tan trivial como sea posible.
-Fomentar la localidad de estructuras de control y uso de datos.
-Reducir la complejidad del programa simplificando los patrones de control.
-Deshabilitar los programas que tengan que ser leídos de comienzo a fin sin un control de saltos.

Lograr Eficiencia en el programa.

-Desarrollar programas eficientes con respecto a los requerimientos generales, así como al tiempo de ejecución y mantenimiento.

Acrecentar la seguridad del Programa.

-Construir el programa con la idea de que no será necesario depurar.
-Proporcionar el programa correcto como parte del proceso de construcción.
-Establecer un nuevo nivel de precisión en la programación.

Proporcionar una disciplina en la programación.

-Sistematizar el proceso de programación.
-Forzar a una integridad en el sistema.

Reducir el costo de Programación.

-Permitir a los programadores un control total sobre grandes cantidades de código.
-Incrementar la productividad del programador.

El objetivo principal de la programación estructurada fue resolver la crisis por la que atravesaba el

software a finales de los 60s proporcionando una disciplina acompañada de:

-Dar seguridad en el programa.
-Minimizar la complejidad del programa.
-Simplificar su posterior mantenimiento.
-Dar una disciplina en la metodología de programación.

En el sentido más limitado, el concepto de programación estructurada está relacionado únicamente con la forma del programa y el proceso de codificación. Esto es, una serie de convencionalismos que el programador puede seguir para obtener el código estructurado. Las reglas de codificación imponen restricciones en el uso de estructuras de control, estructuras de datos, composición de módulos, y documentación. En este sentido, la programación estructurada se enfoca en temas tales como:

1. Programación de bajo nivel con la sentencia GOTO.
2. La programación con las tres construcciones de control básicas.
3. La convención de la codificación estructurada aplicada a un lenguaje en particular.
4. La forma de un programa estructurado.

En un sentido más amplio, la programación estructurada direcciona tanto los métodos de programación como la forma del programa. La programación estructurada implica la siguiente metodología estructurada con el fin de diseñar e implementar el programa. Los temas incluyen:

1. La programación modular.
2. Niveles de abstracción.
3. Programación Bottom-Up y Top-Down.
4. Sistematización del proceso de programación.

En el sentido más amplio, el concepto de programación estructurada acompaña al proceso entero de programación.

Esto implica una disciplina en cada etapa de la implementación de un sistema y la organización de los programadores.

Las principales características de un programa estructurado son de una forma jerárquica; están restringidas por un conjunto de estructuras de control.

Las tres construcciones básicas introducidas son la secuenciación, selección e iteración.

Secuenciación.

Esta estructura se utiliza para controlar la ejecución del programa, las instrucciones se ejecutan una trás otra en el mismo orden en el que aparecen en el código fuente (Ver figura 2.4).

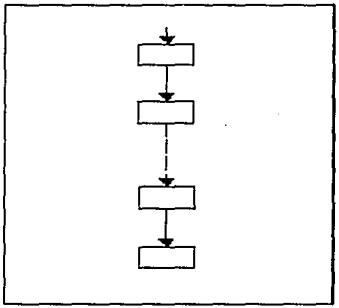


Figura 2.4.Cuando se utiliza la construcción de control de secuencia, las sentencias son ejecutadas una trás otra en el mismo orden como aparecen el código fuente.

Selección.

Cuando esta estructura se utiliza una condición se prueba. Si la condición es verdadera, un conjunto de

instrucciones se ejecuta; si la condición es falsa otro conjunto de instrucciones se ejecuta. Ambos conjuntos caen en un punto en común para continuar la ejecución. La figura 2.5 muestra la estructura de selección.

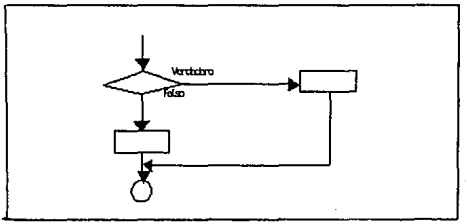
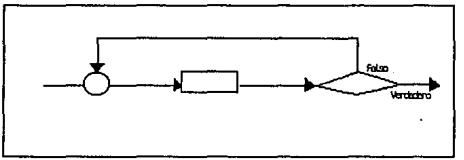


Figura 2.5. Estructura de Selección.

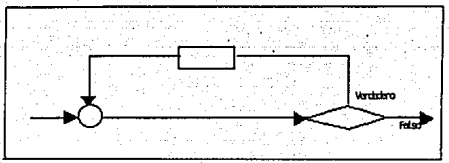
Iteración.

Una condición de terminación se prueba. Si la condición es falsa, el "loop" termina y la ejecución continua con la siguiente instrucción. Si la condición es verdadera, un bloque de instrucciones se ejecuta y la condición se prueba nuevamente.

Las construcciones estructuradas pueden estar anidadas una en otras. La figura 2.6 muestra la construcción de iteración.



DO UNTIL.



DO WHILE.

Figura 6.La construcción de iteración es utilizada para ejecutar una serie de instrucciones un número entero de veces (n 0). Existen dos formas básicas: UNTIL y DO WHILE.

Estas tres construcciones son fundamentales en programación estructurada - una técnica de diseño importante en el campo más amplio de lo que hemos aprendido a llamar Ingeniería de Software.

Las construcciones estructuradas se propusieron para limitar el diseño procedimental de software a un pequeño número de operaciones predecibles. Las métricas de complejidad indican que el uso de construcciones estructuradas reduce la complejidad de los programas y por tanto facilita la legibilidad, prueba y mantenimiento.

El objetivo principal de la programación estructurada es producir programas de alta calidad a bajo costo.

Las características más importantes de un programa estructurado son las siguientes:

El programa está dividido en un conjunto de módulos de manera jerárquica.

Cada uno de los módulos debe tener una sola entrada y una sola salida y cada uno de ellos debe regresar al módulo que lo invocó. Además la construcción de cada módulo está estandarizado al uso de las estructuras de

control básicas antes mencionadas: secuenciación, selección e iteración.

La documentación se requiere en el código fuente para definir la función del programa así como para explicar la función de cada módulo , sus estructuras de datos y sus relaciones con otros módulos en el programa.

Cada módulo oculta los detalles poco importantes a los de arriba en jerarquía. (Programación por refinamientos sucesivos).

III. TECNICAS DE DIAGRAMACION.

La buena diagramación juega un papel muy importante en el diseño de sistemas complejos y el desarrollo de sistemas. Es esencial el diseño asistido por computadora y la tecnología CASE.

Si una sola persona desarrollara un sistema, los diagramas que él utilizara serían claros y de gran ayuda para él. Una mala selección de una técnica de diagramación puede inhibir lo que él está pensando. Una buena opción, puede acelerar su trabajo y proporcionar calidad en los resultados.

Cuando varias personas están trabajando en un sistema o programa, los diagramas son una herramienta de comunicación esencial. Se necesita una técnica de diagramación formal para que los desarrolladores puedan trabajar en módulos distintos de tal manera que al juntarlos, exista precisión.

Durante la depuración, la diagramación clara es una herramienta invaluable en la comprensión de la manera en que el programa sa va a desarrollar.

Las técnicas de diagramación estructurada tienen muchas ventajas. Primero, combinan notaciones narrativas con gráficos para incrementar su comprensión. Los gráficos son especialmente útiles dado que tienden a ser menos ambiguos que la descripción narrativa. También, porque son más concisos y pueden ser dibujados en menos tiempo que el que nos tomaría escribir un documento narrativo conteniendo la misma cantidad de información.

Segundo, las técnicas de diagramación estructurada soportan la técnica de desarrollo Top-Down, la cual puede describir un sistema o programa en diferentes grados de mayor detalle durante cada paso del proceso de descomposición funcional. Pueden clarificar los pasos y los resultados de éste proceso proporcionando una manera estandarizada en la descripción lógica procedural y las estructuras de datos.

Uno de los problemas en computación, es que es fácil hacer un enredo. Las técnicas basadas matemáticamente están evolucionando de manera que la computadora nos auxilie en la creación de especificaciones sin ambigüedades, inconsistencias y omisiones.

Lo mejor de CASE es que impone una disciplina en analistas y diseñadores, asegurando de ésta manera, que los sistemas complejos sean mejor manejados y de fácil modificación.

1.- Diagrama de Descomposición.

La mayoría de las técnicas estructuradas utilizan la técnica de descomposición para refinar información completándola en niveles sucesivos de detalle. Una estructura de árbol, es un ejemplo del empleo de esta técnica.

La descomposición jerárquica puede ser aplicada a estructuras de organización, estructuras de sistema, de programas, de archivos y estructuras de reporte. El término descomposición funcional se aplica a funciones más que a datos; sin embargo, algunas veces son dibujados diagramas similares para la descomposición de ambos, datos y funciones.

Los diagramas de descomposición ayudan a crear y mantener diagramas para:

Descomposición Organizacional .- Para ayudar a describir las estructuras jerárquicas de la organización.

Descomposición de Datos .- Para mostrar cómo un grupo general de datos (áreas sujeto) se dividen en entidades de datos más específicas.

Descomposición de Procesos .- Para describir el análisis de procesos en subprocesos.

El tipo más común de descomposición funcional es la estructura de árbol que relaciona las funciones y no los datos que estas funciones están utilizando.

El segundo tipo de descomposición muestra el tipo de datos que son entrada y salida de cada función. Si ésto

se maneja por computadora, la máquina puede checar que los datos consumidos y producidos por cada nodo funcional sea consistente a través de toda la estructura.

El tercer tipo es más completo. Este permite únicamente cierto tipo de descomposición., la cual está sujeta a reglas precisas que son definidas por axiomas matemáticos. La estructura que resulta puede ser completamente verificada para asegurar que ésta sea internamente consistente y correcta.

2.- Diagrama de Entidad-Relación.

Las entidades son sujetos de información (personas, lugares, cosas, eventos, ideas, etc) acerca de las cuales se necesita guardar información. Estos diagramas proveen una manera gráfica de describir los requerimientos de datos así como sus relaciones. En éste también se incorporan las reglas del negocio.

Los diagramas de entidad-relación hacen que el crear y cambiar el modelo de datos de la empresa sea fácil.

Modelo de entidad relación (Modelo E-R).

Este modelo fue propuesto por Peter Chen, en 1976. Es usado como el modelo principal sobre el cual se soporta el diseño lógico de una BD. Este modelo es independiente del ambiente de hardware, de tal forma que no existen restricciones físicas para su implantación. Un modelo de este tipo debe ser consistente, preciso, completo, claro y breve.

Este modelo permite representar información en términos de : ENTIDADES, sus ATRIBUTOS y asociaciones entre las ocurrencias de sus entidades, llamadas RELACIONES.

ENTIDADES:

Son los objetos principales acerca de los cuales se almacena información.

Ejemplos de entidades físicas: personas, empledos, automovil, libros, etc.

Ejemplos de entidades conceptuales: organizaciones, contratos, etc.

Una entidad se representa como un rectángulo:

Los nombres de las entidades irán dentro del rectángulo y serán dados en singular.

TIPOS DE ENTIDADES:

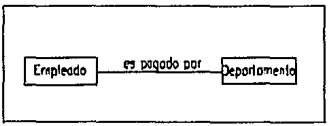
FUNDAMENTAL: entidad base que no depende de otra entidad para su existencia.
ATRIBUTIVA: es una entidad que depende de otra entidad para su existencia.
ASOCIATIVA: es una entidad que describe la relación entre dos entidades y el mantenimiento de datos acerca de esa relación.

ATRIBUTOS.

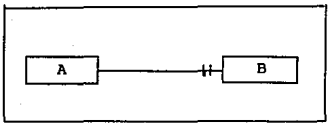
Son las características y propiedades que las entidades pueden tener.

RELACIONES.

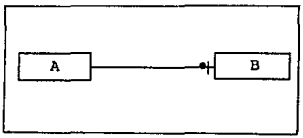
Es una conexión o asociación existente entre entidades. Las relaciones se expresan como una frase con verbo.
Ejemplo:



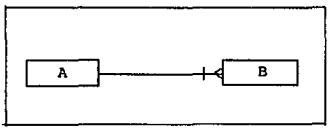
ASOCIACIONES BASICAS.



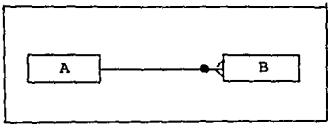
Para cada ocurrencia de A hay una y solo una ocurrencia de B.



Para cada ocurrencia de A hay una o cero ocurrencias de B.



Para cada ocurrencia de A hay una o muchas ocurrencias de B.

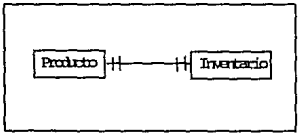


Para cada ocurrencia de A hay cero, una o muchas ocurrencias de B.

CARDINALIDAD DE LAS ASOCIACIONES.

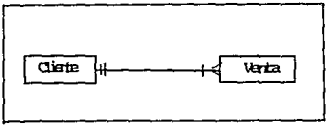
1:1 UNO A UNO.

Cada producto está asociado con solo un registro de inventario, y cada registro de inventario esta asociado con solo un producto.



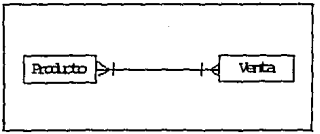
1:M UNO A MUCHOS

Cada cliente se puede asociar con muchas ventas, pero cada venta puede ser solo asociada a un cliente.

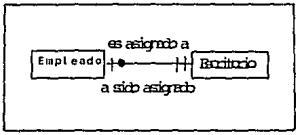


M:M MUCHOS A MUCHOS.

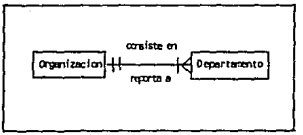
Un producto puede ser asociado con muchas ventas, y una venta puede ser asociada a muchos productos, al menos uno, posiblemente más.



EJEMPLOS DE ASOCIACIONES CON CARDINALIDADES.

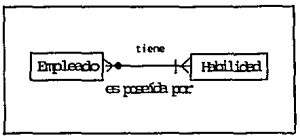


Y se lee de izq. a der: un empleado es asignado a uno y solo un escritorio.
Ahora de der. a izq.: un escritorio a sido asignado a cero o un empleado.



Y se lee de izq a der.: una organización consiste de uno o muchos departamentos.

Ahora de der. a izq.: un departamento reporta a una y solo una organización.



Y se lee de izq. a der. : un empleado tiene una o muchas habilidades.
Ahora de der. a izq.: una habilidad es poseida por cero, uno o muchos empleados.

IDENTIFICADOR (llave de una entidad).

Una llave es el conjunto mínimo de atributos que identifican de manera única a cada ocurrencia de una entidad.

Una llave debe de ser única. Una llave puede estar formada por un solo atributo o por la concatenación de varios atributos.

Llave foránea.- Se le llama así a aquel(llos) atributo(s) que forman parte de una entidad y son llave de otra.

Ejemplo. Tenemos las siguientes entidades con sus atributos:

Proyecto (Código de proyecto, tipo de proyecto, descripción)
Empleado (Número de empleado, nombre)
Asignación de Proyecto (Código de proyecto, Número de empleado,
Fecha de iniciación, tiempo asignado)

Las llaves para cada una de éstas entidades son:

Para la entidad Proyecto: Código de proyecto;
Para Empleado: Número de empleado y,

Para Asignación de Proyecto: Su llave estará formada por Código de proyecto y Número de empleado. En donde estos atributos por separado son llaves foráneas, pues cada uno de ellos es llave de otra entidad.

OBJETOS QUE NO SON ENTIDADES.

Procesos: Ejecuta acciones en una entidad.
Son ejemplos: Una factura de materiales, un presupuesto, una póliza de garantía.
Cálculos: Derivaciones de atributos de una entidad.
Son ejemplos: El cálculo de una edad promedio, el valor neto de un artículo
Reportes: Presenta hechos acerca de una o más entidades.
Son ejemplos: Un estado de resultados, una cédula de proyectos.

Hechos acerca de entidades: Describe características de las entidades.
Son ejemplos: Un número telefónico, una fecha de contratación.

DIFERENCIA ENTRE EL CONCEPTO DE ARCHIVO Y ENTIDAD.

Archivo: Conjunto de información relacionada lógicamente entre sí, un archivo puede contener información de varios objetos.
Ejemplo: Un archivo de claves que contiene el nombre y la clave de personas aseguradas, de agentes de seguros, etc.

Entidad: Al igual que un archivo una entidad almacena información, pero a diferencia de este , la información almacenada por una entidad es concerniente a un solo objeto.
Ejemplo: Del ejemplo anterior, podemos obtener la entidad persona asegurada que contendrá el nombre y clave de asegurados.

ARCHIVOS DE CLAVES Y CATALOGOS.

Claves: Este archivo se puede dividir en varías entidades con información propia, convirtiéndose, posteriormente cada uno de ellos en un archivo de catálogo.

Consideremos el ejemplo de un archivo que contiene el nombre y clave de clientes asi como nombre y clave de vendedores de manera indistinta. Podríamos dividir este archivo en dos entidades distintas: Una de clientes y una de vendedores las cuales a su vez, formarán parte del archivo de catálogo.

ARCHIVOS HISTORICOS Y TEMPORALES.

Archivos históricos: Estos no se encuentra plasmados en los diagramas de E-R de cada uno de los sistemas ya que su existencia es únicamente de control, por lo que si aparecerán a nivel proceso, pero no a nivel datos.
Archivos temporales: También son archivos que aparecerán únicamente a nivel proceso, ya que son auxiliares en la obtención de algún producto del sistema y no son fundamentales en la representación del modelo de datos.

DATOS REPETIDOS.

La idea fundamental en este tipo de análisis es eliminar la redundancia de datos en el modelo de datos. No se espere encontrar en cada una de las entidades atributos que pueden obtenerse de otras entidades por relación entre ellas mismas.

CLASIFICACION DE ENTIDADES Y ATRIBUTOS.

La siguiente es una guía para clasificar entidades y atributos, con la idea de obtener un diseño de Base de Datos "eficiente".

1. Las entidades poseen información descriptiva; los atributos no.

A. Si existe información que describe a un objeto, el objeto debe ser clasificado como una entidad.

B. Si solo se requiere un identificador para describir a un objeto, el objeto debe ser clasificado como un atributo.

Ejemplo. Tenemos la entidad "Persona" cuyos atributos son:

2. Los atributos multivaluados deben ser clasificados como entidades.

Si más de un valor de un atributo descriptor corresponde a un valor de un identificador, este atributo descriptor debe ser clasificado como una entidad, en lugar de un atributo, aun y cuando no podamos reconocerle atributos propios.

3. Convertir en una entidad a un atributo que tenga una relación muchos-a-uno con una entidad.

Si un atributo descriptor en una entidad tiene una relación muchos-a-uno con otra entidad, el atributo descriptor debe ser clasificado como una entidad, aún y cuando no podamos reconocerle descriptores propios.

4. Asociar atributos a las entidades que ellos describen más directamente.

5. Evitar los identificadores compuestos en la medida que sea posible.

Si una entidad ha sido definida con un identificador compuesto, (de más de un atributo) y los componentes del identificador son a su vez identificadores de otras entidades diferentes, entonces hay que eliminar a esta entidad. El objeto a que se hace referencia se deberá definir como una relación, en un paso posterior.

Si una entidad ha sido definida con un identificador compuesto, pero los componentes del identificador no son identificadores de otras entidades, existen dos soluciones posibles :

A. Eliminar la entidad y definir nuevas entidades con los componentes del identificador compuesto como identificadores de la entidad, y en un paso subsiguiente definir una relación para representar este objeto.

B. Mantener la entidad con el identificador compuesto, si esta entidad es razonablemente "natural".

3.- Diagrama de Flujo de Datos.

Los diagramas de flujo de datos muestran los procesos y el flujo de datos entre estos procesos. Un diagrama de flujo de datos se utiliza como primer paso en una forma de diseño estructurado. Es una herramienta primaria para dibujar los componentes procedurales básicos y los datos que se manejan entre estos.

Estos diagramas pueden representar cualquier número de niveles de detalle. Siguiendo una metodología de "arriba a abajo" se pueden descomponer progresivamente los diagramas de flujo de datos, comenzando con el diagrama de contexto (diagrama 0), y terminando con los procesos elementales.

El diagrama de flujo de datos muestra el flujo de datos dentro de un sistema lógico, pero no nos proporciona un control o secuencia de la información.

Un diagrama de flujo de datos (DFD) es una representación en red de un sistema mostrando los procesos e interfaces de datos entre éstos. El DFD se construye a partir de cuatro componentes básicos: El flujo de datos, el proceso, el almacenamiento de datos y las relaciones.

El **flujo de datos** delinea el flujo de datos a través de un sistema de procesos. La dirección del flujo de datos se indica por las flechas. Los datos son identificados por nombre, con el nombre escrito a un costado de la flecha correspondiente; por ejemplo:

Productos Ordenados

---------->

El **proceso** es un componente procedural en el sistema. Opera en (o transforma) datos. Por ejemplo, puede desarrollar operaciones lógicas o aritméticas en datos para producir resultados. Cada proceso se representa por un círculo o un rectángulo con puntas redondeadas en el DFD. El nombre del proceso se escribe dentro del círculo. Se debe seleccionar un nombre clave para definir la operación desarrollada por el proceso.

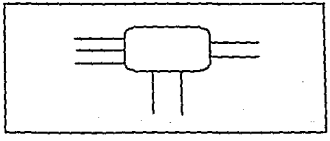
El **almacenamiento de datos** representa un archivo lógico. Se dibuja con un par de líneas paralelas en el DFD. El nombre de los datos almacenados se escribe dentro de las líneas.

El terminator muestra el origen de datos utilizados por el sistema y el último receptor de datos producido por el sistema. El origen de datos se le llama fuente, mientras que el receptor de datos un vertedero. Para representar el terminator se utiliza un rectángulo.

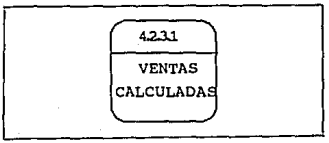
La notación que se va a emplear para el desarrollo del sistema es en base a las convenciones adoptadas por Gane y Sarson cuyos diagramas varían ligeramente de las popularizadas por Yourdon y De Marco.

Gane y Sarson dibujan un proceso como un rectángulo con puntas redondeadas.

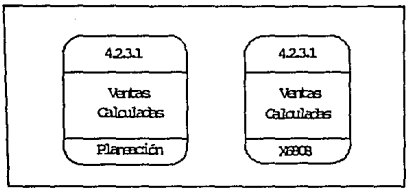
En el dibujo computarizado, la máquina puede más facilmente ligar múltiples flechas al bloque:



En lo alto del bloque, se coloca un número de bloque u otro identificador para lo que está escrito:

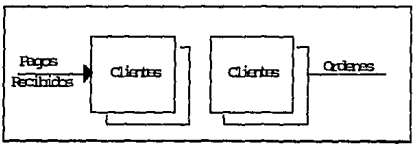


En la parte inferior, el diseñador puede opcionalmente dibujar una localidad física donde el proceso tome lugar o el nombre del programa que ejecuta el proceso:

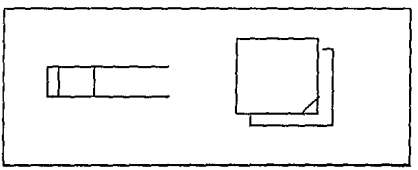


El almacenamiento de datos es dibujado con un bloque en la parte izquierda y en el cual se puede colocar un número o identificación. Puede ligar éstos a un modelo de datos:

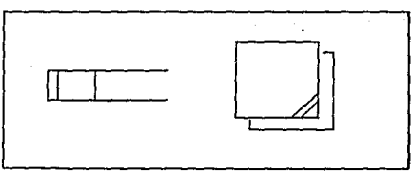
Una fuente o destinación externa de datos, se puede dibujar con un doble cuadrado:



Para simplificar la cantidad de líneas en un dibujo, una fuente externa o destinación puede aparecer varias veces. Si un grupo de datos aparece dos veces se dibuja una línea vertical del lado izquierdo del bloque. Si un bloque externo aparece dos veces, se dibuja una línea vertical en la esquina inferior derecha:



Sí además de éste, aparecen tres veces, se dibujan entonces dos líneas:

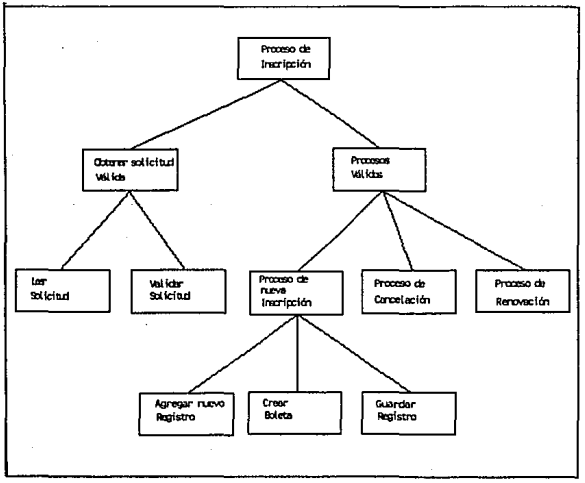


Si aparecen N veces, se dibujarán N-1 líneas.

4.- Diagramas de Estructura.

Este diagrama, popularizado por Yourdon, Constantine, Page-Jones y otros, es útil para darnos una visión completa de la estructura de un programa o un sistema. Muestra la jerarquía de los módulos subordinados, así como la información que se pasa entre éstos, es decir, es un árbol que muestra los módulos y sus relaciones.

La siguiente figura muestra una carta de estructura para un sistema de inscripción. El propósito de este



sistema es realizar las operaciones de inscripciones contra un archivo de inscripción maestro. Existen tres tipos de transacciones: nuevas inscripciones, renovaciones y cancelaciones. Cada operación primero se validada y después se procesa contra el archivo maestro. Para nuevas inscripciones, se construye un registro base así como su factura correspondiente.

Para renovaciones, se actualiza la fecha de expiración y también su correspondiente factura. Para cancelaciones, el registro se borra y se re-edita.

El programa esta representado como una serie de módulos ordenados jerárquicamente. Aquellos módulos que desarrollan tareas más importantes dentro del programa, son colocados en niveles de más alta jerarquía. Observando los niveles de menor jerarquía, éstos se les asignarán a los módulos de los niveles procedentes y que desarrollan tareas más pequeñas. Por ejemplo, la tarea OBTENER LA VALIDACION DE PRE-INSCRIPCION está compuesta por dos sub-tareas, LEER PRE-INSCRIPCION y VALIDAR PRE-INSCRIPCION.

Los bloques básicos de una carta de estructura son cajas rectangulares y flechas como un medio de conexión entre éstas. A pesar de que la estructura más pequeña posible está formada por una caja, la mayoría de los diagramas contienen muchas cajas así como flechas de conexión.

Cada caja rectangular en una carta de estructura representa un módulo. Logicamente, un módulo es una tarea que el programa desarrolla, tal como sumar un nuevo registro o crear una factura. Físicamente, un módulo es implementado como una secuencia de instrucciones de programación ligadas por un punto de entrada y un punto de salida.

Se debe seleccionar un nombre descriptivo para explicar la tarea que el módulo esta desarrollando. Después del nombre del módulo, la carta de estructura no proporciona información del contenido de los módulos.

Los módulos están interrelacionados por una estructura de control. La carta de estructura muestra las interelaciones con los arreglos en niveles y conectando estos niveles por flechas.

Una flecha dibujada entre dos módulos en niveles sucesivos indica un tiempo de ejecución, donde el control del programa es transferido de un módulo al segundo en dirección de la flecha. Es decir, el primer módulo invoca o llama al segundo módulo.

Un módulo puede invocar a diferentes módulos (o tener diferentes hijos). Dado que la carta de estructura no nos muestra una secuencia, no podemos saber en qué orden el padre invoca a sus hijos.

En algunos casos, el desarrollador del programa puede utilizar librerías de módulos predefinidas.

Cuando el control es transferido entre dos módulos, generalmente los datos también son transferidos, los cuales pueden ser transferidos en una u otra dirección. La dirección se realiza dibujando una pequeña flecha. Los nombres de los datos son escritos junto a la flecha para identificar que datos fueron trasladados.

Las reglas que se manejan para la estructura de control del programa se pueden resumir de la siguiente manera:

1. Existe uno y solamente un módulo en la parte más alta de la carta de estructura (nivel 1), donde el control se origina. Este módulo es llamado raíz.

2. Desde raíz, el control se lleva de manera descendente, nivel por nivel a los otros módulos. El control siempre retrocede al invocar a un módulo. Además, cuando termina la ejecución del programa, el control regresa a raíz.

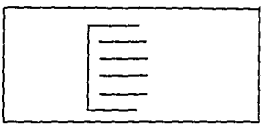
3. Existe una sóla relación de control entre dos módulos. Esto quiere decir que si un módulo A invoca a un módulo B, éste no puede invocar al modulo A. Por su parte, el módulo A no puede invocarse asi mismo.

5.- Diagrama de Acción (Miniespecificaciones).

Este diagrama, creado y promovido por James Martin, es una excelente herramienta para describir claramente los detalles de un proceso en una forma estructurada.

Los diagramas de acción son gráficas y notaciones narrativas lo suficientemente completas como para representar la lógica de un programa en detalle. Ejemplos de estos diagramas de acción, son los diagramas de estructura, los diagramas de Warnier-Orr y diagramas de flujo con una lógica detallada. Estos fueron originalmente diseñados para facilitar la enseñanza a los usuarios finales y para asistir a los usuarios en aplicaciones con lenguajes de cuarta generación.

Un módulo del programa se dibuja con un paréntesis:



Estos paréntesis cuadrados son la estructura básica de los diagramas de acción y sirven para describir la secuencia de operaciones. Lo paréntesis pueden ser de cualquier longitud, con el fin de detallar tanto como se requiera.

Dentro de los paréntesis se encuentra una secuencia de operaciones. El anidamiento de múltiples paréntesis indican una estructura jerárquica de procesos y subprocesos.

Los diagramas de acción fueron diseñados para resolver algunos problemas relacionados con otras técnicas de diagramación. Fueron diseñados para tener las siguientes propiedades:

1. Son más fáciles y rápidas para dibujar y cambiar.

2. Son buenos para el diseño manual y para la edición computarizada.

3. Una técnica sencilla proporciona desde una visión general y más alta a los detalles de un nivel de codificación.

4. Es fácil de enseñar a los usuarios finales; fomentan a que los usuarios finales desarrollen su capacidad dentro de la examinación y diseño de procesos lógicos detallados.

IV. HERRAMIENTAS COMERCIALES CASE.

La Ingeniería de Software asistida por Computadora (CASE) está pasando de su etapa inicial dado el pronto desarrollo de la industria. Este largo período de incubación, el cual principalmente implicó la aceptación del concepto de CASE, está terminando.

Las empresas están aceptando CASE porque es una herramienta importante en el desarrollo y mantenimiento de problemas con lo que se reducen los esfuerzos en el manejo de información y sistemas de tiempo real.

CASE utiliza una tecnología más simple y efectiva. CASE sirve como punto focal para incrementar la comunicación dentro de la organización, la cual proporciona el desarrollo de sistemas efectivos.

El primer paso hacia la efectiva implementación de CASE es establecer una técnica o metodología, ya sea comprada o creada por uno mismo, que defina las etapas del proceso de desarrollo así como los elementos de diseño necesarios para movernos de manera sucesiva a través de todo el proceso.

Una vez establecida la metodología de desarrollo, el manejador MIS (Manejador de Sistemas de Información) debe seleccionar un producto CASE que pueda soportarlo. Además, el sistema CASE debe ser compatible con las herramientas existentes de software direccionando todas las áreas al ciclo de vida del desarrollo del sistema.

CASE, por otro lado, organiza y automatiza la representación gráfica de datos y procesos a través del sistema estimulando la entrada detallada de los manejadores y usuarios finales durante el proceso de diseño. Dado que las herramientas CASE muestran instantáneamente los cambios en datos y procesos dentro del sistema, el manejo, los usuarios y grupos de diseño pueden manejar diferentes modelos de diseño con el fin de obtener de manera óptima los mejores requerimientos particulares.

CASE, por lo tanto, aparece como una fuerza cohesiva en la organización, junto con la estrategia de manejadores, la realidad práctica de los usuarios

finales y los cambios en el diseño de la Ingeniería de Software.

Como siguiente punto dentro de nuestro análisis en éste capítulo, mencionaremos algunas de las características de mayor importancia con las que debe contar una herramienta CASE, las cuales servirán como base para comparar algunas de las distintas herramientas existentes hasta el momento.

Características generales.

- Facilita al usuario la construcción de diagramas para planeación, análisis o diseño en pantalla.

- Solicita información acerca de los objetos en los diagramas y de las relaciones entre estos objetos, de modo que un conjunto completo de información es construido.

- Guarda el significado de los diagramas, más que el diagrama en sí, en un depósito.

- Checa la exactitud e integridad de los diagramas.

- Facilita al usuario el empleo de múltiples tipos de diagramas representando diferentes fases en el análisis o diseño.

- Representa especificaciones y programas con diagramas mostrando loops, condiciones, estructuras CASE y otros enunciados de la programación estructurada.

- Obliga a diseñar y modelar estructuradamente.

- Coordina la información en múltiples diagramas, checando que sean consistentes y juntos tengan exactitud e integridad.

- Guarda la información en un depósito central compartido por todos los analistas y diseñadores.

- Coordina la información en el depósito central, asegurando la consistencia entre el trabajo de todos los analistas y diseñadores.

Razones para usar una herramienta CASE

Mediante el uso de una herramienta CASE se pueden lograr los siguientes beneficios:

- Estandarización de las metodologías de Análisis y Diseño.

- Incremento de la productividad.

- Documentación actualizada.

- Información centralizada.

- Mejora la comunicación dentro del grupo de desarrollo y con los usuarios finales utilizando un lenguaje común.

- Automatiza actividades de desarrollo.

- Verifica el cumplimiento de reglas de consistencia y congruencia y con ésto se asegura la precisión de la información.

- Ambientes amigables para el manejo de gráficos.

Para la selección de la herramienta CASE adecuada es aconsejable tomar en cuenta las siguientes criterios:

-Requerimientos de equipo.
-Metodololgías manejadas.
-Costo.
-Facilidad de uso.
-La no dependencia de lenguajes y Bases de Datos.
-Calidad y diversificación de reportes.
-Manejo del diccionario de datos.
-Verificación de consistencia.
-Herramientas para desarrollo de prototipos.
-Interfaz gráfica.
-Navegación entre distintos diagramas.
-Normalización de datos.

Es conveniente mencionar que en la selección de la herramienta CASE utilizada en el sistema a desarrollar en capítulos posteriores, se trató de tomar en cuenta

cuál de ellas cumplía de mejor forma con los criterios anteriores.

Como se podrá observar, existen herramientas que llegan a cumplir la mayoría de estos criterios pero el acceso a ellas nos fue limitado, por lo que se seleccionó la herramienta de más fácil adquisición pero no inadecuada.

EVALUACION DE HERRAMIENTAS COMERCIALES CASE.

En base a los puntos antes mencionados para la selección de la herramienta CASE adecuada, se procederá a analizar diversas de las herramientas que se pueden encontrar actualmente en el mercado.

Dentro del gran mercado de distribuidores de herramientas CASE, se encuentra la compañía Chen & Associates, Inc., la cual proporciona una serie de herramientas de Ingeniería Inversa para PC para diferentes DBMSs. Tales herramientas son las siguientes: La herramienta de Ingeniería Inversa -Reverse Engineering Tool-, ER Designer, DDS-Link, Normalizer, SchemaGen y Autodraw.

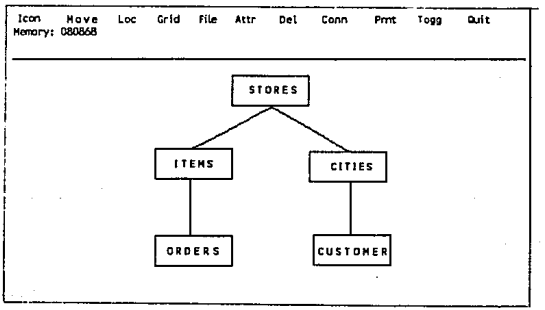
Reverse Engineering Tool.

La herramienta de Ingeniería Inversa para IMS captura las descripciones de la base de datos de una aplicacion IMS existente y posteriormente la convierte en un Diagrama de Entidad-Relación y/o un diagrama IMS jerárquico, los cuales pueden ser desplegados gráficamente, como se observa en las figuras 4.1. y 4.2.

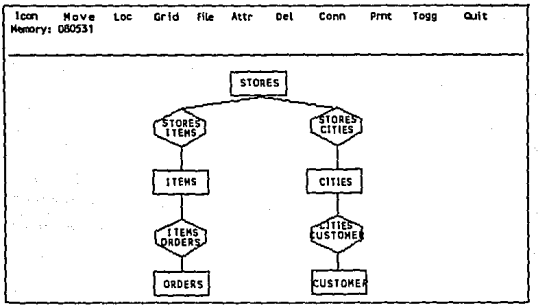
De esta forma obtenemos un diagrama de ER representando el modelo de datos conceptual para dicha aplicación IMS. Este diagrama de ER servirá como un importante documento para futuras modificaciones y otras actividades de mantenimiento.

La información desplegada como un diagrama de ER es también almacenada en el diccionario de ER por otros módulos como son: el ER-Disigner, Normalizer, SchemaGen y DDS-link diseñados por la misma compañia. Así, la aplicación IMS existente puede ser normalizada y/o traducida a descripciones de base de datos o otras DBMSs (ej. DB2) o cargada al diccionario de datos de un mainframe (ej. IBM's DB/DC, etc).

En conclusión, la Reverse Engineering Tool de Chen puede ayudar en el rediseño de una base de datos IMS o migrar a otros DBMSs.



4.1. Diagrama jerárquico IMS



4.2. Diagrama correspondiente ER

DDS-Link.

DDS-Link brinda un método fácil y rápido de conversión de datos de un diagrama de ER en formatos de datos apropiados a través de un diccionario de datos (DDS) en un equipo mainframe.

Los formatos del diccionario de datos que actualmente DDS-Link soporta son los siguientes: MSP's DataManager/DisignManager, IBM DB/DC Data Dictionary, Computer Associates IDD Dictionary, DEC CDD+ y un formato genérico.

Los archivos creados por DDS-Link pueden ser cargados a un equipo mainframe a través de un programa de transferencia de datos de una PC-a un-mainframe como son Kermit o PC-Talk. Los archivos son entonces leídos por el sistema de diccionario de datos.

El DDS-Link puede también ser usado como una utileria para emitir reportes. Aunque un generador de reportes

esta también disponible en el Diccionario Chen, los usuarios pueden usar los archivos de datos creados por DDS-Link (en formato ASCII estándar) para construir su propio generador de reportes o para accesar los formatos con datos importados para su sistema de diccionario de datos.

Normalizer.

Normalizer es uno de los primeros y mas sofisticados paquetes de software comercial para normalización de datos para PC. Comparado con productos similares para equipos mainframe, Normalizer presenta muchas de las funcionalidades de productos similares pero con una interfase mas amigable al usuario.

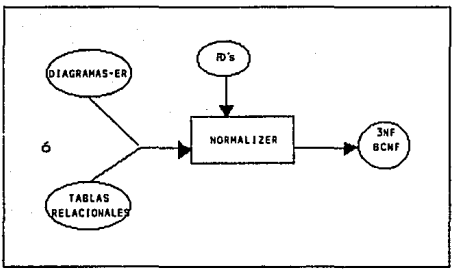
Normalizer es una herramienta de validación para el modelado de datos, la cual forza a una rigurosa disciplina dentro del mismo modelado. Además, puede ser utilizada para validar los resultados obtenidos del modelado top-down de diagrama Entidad-Relación.

Normalizer puede también usarse en el modelado de procesos. Analiza las agrupaciones de datos resultantes de un diseño de sistemas basados en una técnica de modelado de procesos como es el Diagrama de Flujo de Datos (DFD). Las estructuras de archivos resultantes tendrán pocas anomalías en actualización o recuperación.

Normalizer puede ser usado como una sola herramienta o como parte del Diseñador ER, una herramienta para diagramación Entidad-Relación que ofrece Chen & Associates. Acepta modelos de ER creados por el diseñador ER y genera sus respectivas relaciones normalizadas. Las dependencias funcionales (FD's) son definidas ya sea de manera interactiva o recuperadas de archivos existentes.

Normalizer normaliza las relaciones a la Tercera Forma Normal (3NF) o a la Forma Normal de Boyce-Codd (BCNF). Normalizer se asegura que las relaciones complejas sean descompuestas en otras mas simples en las cuales los elementos de datos sean agrupados y la duplicación minimizada. Normalizer también soporta las

modificaciones del diagrama de ER para reflejar los resultados en la descomposición de entidades. Normalizer no solo verifica que una relación cumpla satisfactoriamente las condiciones de forma normal, sino que descompone y normaliza una relación en varias relaciones normalizadas.(Figura 4.3.)



4. 3. Flujo de Información en Normalizer.

SchemaGen.

SchemaGen convierte los diagramas de ER (ERD's) en un esquema para base de datos para una DBMS destino. Los esquemas creados por SchemaGen pueden ser cargados en un mainframe o micro-computadoras en las cuales resida el DBMS destino.

Los beneficios que se obtienen de usar SchemaGen son:

1.- La vista lógica de los datos capturados en un diagrama de ER son trasladados automáticamente en un esquema físico de un DBMS. La semántica de los datos (en particular las relaciones entre entidades) es incorporada en el algoritmo interno de SchemaGen de

manera que se asegure que el esquema generado es semánticamente correcto.

2.- Las reglas sintácticas de esquemas de DBMS son forzadas por SchemaGen; de este modo, el esquema producido seguramente será sintácticamente correcto.

Los esquemas para los DBMS que pueden ser generados son : DB2, SQL/DS, IMS, Oracle, Ingres, ADABASE, IDMS/R, DATACOM/DB, Nomad, Model 204, UNISYS/RDMS, Focus, Teredata, Sybase, Informix, DEC/Rdb, DG/SQL, dBASE, ZIM, db_VISTA, Progress, HP/IMAGE, PACE, y más.

Diccionario de Datos.

El diccionario de datos en el Workbench de Chen esta hecho para manejar y controlar el análisis del sistema y el proceso de diseño. El diccionario está estructurado para mantener consistencia y control en cada fase del ciclo de vida del desarrollo de un sistema. Por ejemplo, la verificación de consistencia es efectuada cada vez que un diagrama de ER es cambiado y el diccionario sigue la asignación de atributos y objetos de los diagramas de ER.

Los comandos del diccionario son comprensibles y fáciles de entender: Load, permite cargar un diccionario al sistema y permite la creación de nuevos dicccionarios; Process permite al usuario almacenar un diagrama de ER en un diccionario, crear o modificar un objeto, y/o crear, modificar o borrar un atributo; Reports se usa para generar ya sea amplios reportes para diagramas de ER o reportes específicos sobre porciones del diagrama (por ejemplo atributos, reglas o cardinalidad, etc.) o para reportar sobre atributos, objetos y diagramas dentro del diccionario, Statistics para recuperar la configuración del diccionario y del hardware. Todas las herramientas dentro del Workbench de Chen tienen interface con el diccionario.

Autodraw.

AutoDraw automáticamente dibuja un diagrama de ER de entradas textuales. AutoDraw permite al usuario el proporcionar fácilmente los nombres de entidades, relaciones y atributos de un diagrama de ER y entonces transferir éstos al Diseñador ER para desplegarlos como diagramas.

AutoDraw acepta entradas textuales del usuario en forma de archivos tipo texto y genera los archivos de datos necesarios para ser leídos por el Diseñador ER para desplegar un diagrama de ER en la pantalla. Los diagramas que son creados pueden ser modificados según las necesidades de cada usuario (figura 4.4.).

Entrada a AutoDraw.

```
Item_Order      Orders     Items      (1,N)      (1,1)
In_City         Cities     Items
Hold            Stores     Items      (N,N)      (1,N)
Located         Cities     Stores
Live_in         Cities     Customer   (1,N)      (1,N)
Make            Orders     Customer
```

SAMPLE1.AUT

```
#Item_Order
Order_No        C,6        Y
* Identification of a particular order must be
unique.
Item_Id         C,6        Y
* Identification of a particular item ordered.
Quantity_Held   N,5        N
* Numeric string 1-5   digits in length
```

SAMPLE1.ATR

Figura 4.4.

Diagrama ER creado

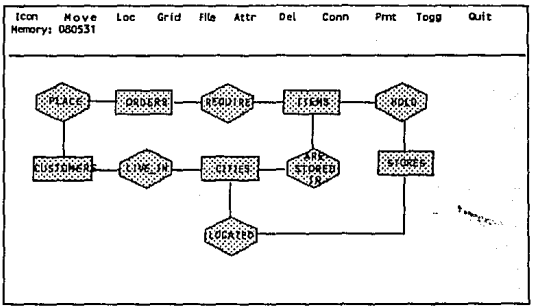


Figura 4.5.

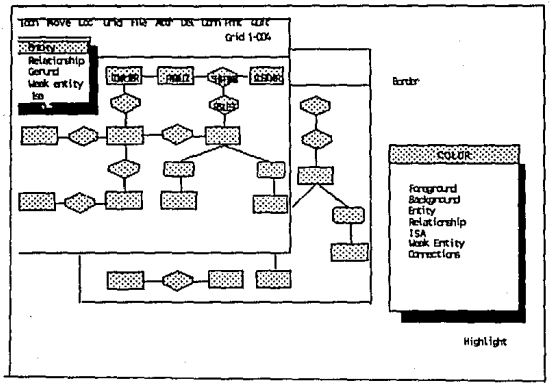
Ambiente Avanzado Para el Modelador ER.

*Soporta construcciones avanzadas: Entidades, relaciones, Entidades débiles y Gerundios (Figura 4.6.).
*Maneja: Datos relacionales (Cardinalidad).
*Soporta: Relaciones totales, recursivas e ISA.
*Manipula: Listas de atributos (incluyendo nombres, formatos, alias, etc) para objetos definidos.
*Permite: Comentarios en pantallas completas o simples anotaciones para atributos y objetos definidos.
*Ejecuta: Chequeo contínuo de posibles errores.

Generador de Reportes.

Los reportes completos y detallados para los diagramas ER son proporcionados desde el Módulo del Diccionario Chen.

ER-Designer.



4.6. Pantalla principal de ER-Designer con diferentes tipos de despliegue de cardinalidad.

Características ER-Designer.

- Menú para el control del ambiente del programa. Turn on u off: despliega parámetros, entrada de relaciones binarias, soporte de mouse, entrada de relaciones recursivas, chequeo de consistencia.

- Usuarios frecuentes. La configuración de los archivos se puede definir para controlar el ambiente del programa.

- Soporte completo de 16 colores en sistema con capacidad EGA.

- Verificación contínua de errores evitando condiciones innecesarias y mantenimiento de consistencia de datos.

- Configuración automática del programa para el hardware de gráficos eliminando la necesidad de alguna instalación por parte del usuario.

- Capacidad para editar cuatro diagramas al mismo tiempo.

- Diccionario de Datos integrado.

Parámetros Soportados por el Sistema.

Máximo número de atributos/ Celdas:	No existe límite. (Depende del espacio en disco)
Tamaño de las ventanas:	5x6, 8x10, 40x40.
Objetos soportados:	Entidades,relaciones, entidades débiles, Gerundios , ISA.
Tipo de Datos:	Integer, smallint, float, decimal, hexadecimal, binario, char, varchar, gráfico, lógico, date, time, money
Ambiente:	MS/PC-DOS 2.X O 3.X
Hardware Mínimo:	IBM PC, XT, AT o PS/2, disco duro de 640K, tarjeta para gráficos (IBM, CGA, EGA, VGA, Hércules y otras)

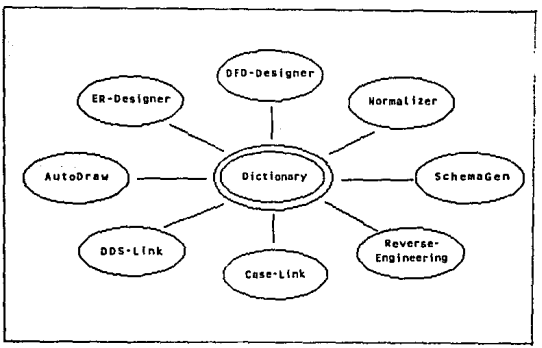
Nuevas Características del Modelador de ER (Versión 2).

1.- Diccionario de datos :

* Diccionario activo para compartimiento de datos y control de integridad con facilidades de importar y exportar datos (Figura 4.7.).
* Proveé un mecanismo de control central para todos los módulos de los productos Chen.

2.-Diagramador de Flujo de Datos (Data Flow Diagrammer):

* Proveé las notaciones de Yourdon & DeMarco o Gane & Sarson.
* Multinivel en diagrama de flujo de datos (DFD´s) con chequeo de consistencia.



4.7. Relación de Herramientas con el Diccionario de Datos.

3.- ER-Designer:

* Proveé seguridad con PASSWORD de entrada.
* Control en el manejador de proyectos con diccionario dividido entre los diferentes diagramas.
* Buena interfase de usuario.
  ** Soporte para usuarios con o sin mouse.
  ** Diccionario de datos amigable.
* Soporte de impresoras láser, matriz de puntos y plotters.
* Expansión de pre-conjuntos de funciones en el archivo de configuración con la característica de auto-save.
* El máximo número de atributos está limitado solo por el tamaño del disco.
* Ventanas adicionales de 8 x 10.
* Adición de múltiples niveles del diagrama de ER: se puede hacer un Zoom interno de un objeto para definir otro diagrama de ER completo para proporcionar una descripción interna del objeto agregado.
* Nombres de objetos y atributos con diferentes tipos de letra. Se puede usar una mezcla de letras mayúsculas y minúsculas en la misma palabra para nombres de relaciones.

4.- Normalizer.

* Las nuevas tablas normalizadas serán automáticamente almacenadas en el diccionario. Una entidad que no esté en tercera forma normal será descompuesta en entidades adicionales en el diccionario de ER.
* Soporte de mouse.

5.- Autodraw.

* El archivo de entrada puede ser convertido a un diagrama de ER automáticamente.
* Aquellos archivos de entrada que no puedan ser convertidos a un diagrama de ER automáticamente, serán convertidos a un diagrama de ER sin las líneas de conexión, permitiendo al usuario usar el comando "connect" de ER-Designer para conectar los objetos.
* Soporte para mouse.

6.- SchemaGen.

* Soporte adicional de DBMS's (como son ADABAS, DMS/R, Progress, dbVISTA, Model 204 and, soon, DBMS de DEC/Unisys/HP/Data General).
* Soporte de mouse con interfase de usuario mejorada.
* ZIM SchemaGen.
**Soporte para cargar y descargar información del diccionario de ZIM.

7.- DDS-Link.

* Soporta mas sistemas de diccionarios de datos (como DEC's CDD+ y otros).

8.- Productos Reverse-Enginering:

* Ayuda en la migración de datos (Por ejemplo: de IMS a DB2).
* Reverse-IMS, Reverse-Oracle, Reverse-Ingres, Reverse-Model204.
* Reverse-DatacomDB, Reverse-IDD.

Information Engineering Workbench IEW.

IEW es una serie de herramientas integradas de Ingeniería de Software asistidas por computadora para la planeación, análisis y diseño de sistemas de información.

Cada herramienta consiste de una serie de diagramas integrados para la captura y revisión de categorías de información. Las entradas se llevan a cabo a través de un verificador lógico llamado Coordinador del Conocimiento y almacenadas en una base de conocimiento llamada Enciclopedia.

Dentro de IEW existen principalmente tres series de herramientas :

- El módulo de Planeación (Planning Workstation-IEW/PWS) Su principal próposito es definir y dar prioridades a proyectos para después estudiarlos e implementarlos. Esto lo realiza capturando altos niveles de información para la planeación de empresas en base a aciertos y factores críticos de ésta.

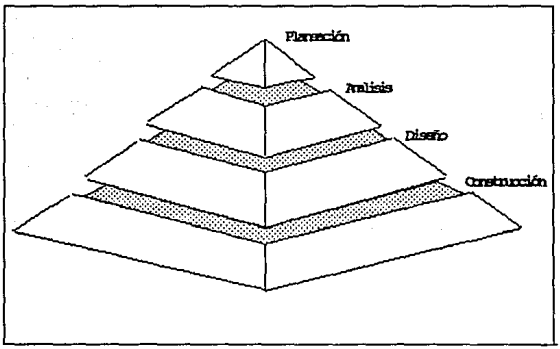
- El módulo de Análisis (Analysis Workstation-IEW/AWS) refina los procesos y datos del modelado de datos. Determina qué procesos es necesario correr en una área de negocios determinada, cómo se relacionan estos procesos y qué datos se requieren.

- El módulo de Diseño (Design Workstation-IEW/DWS) . Define cómo los procesos seleccionados en el área de negocios son o serán implementados en procedimientos específicos (nivel físico) y cómo trabajarán estos procedimientos. El diseño de procedimientos requiere la participación del usuario final.

Existe una herramienta opcional, el módulo de Construcción que se encarga de la implementación del trabajo realizado en los módulos anteriores con generación de código en Cobol.

En donde sea posible, la información definida en una herramienta previa, puede ser utilizada por herramientas de mas bajo nivel. Por ejemplo, las entidades y sus atributos pueden ser descritas en el módulo de planeación y más adelante refinadas en el de Análisis; las mini-especificaciones de procesos bosquejadas en el módulo de Análisis pueden ser perfeccionadas como código de programa en el módulo de Diseño (Figura 4.8.) .

Todos los módulos no solo almacenan, mantienen y analizan el significado de imágenes, sino también de diagramas. La tecnología de Inteligencia Artificial se utiliza para integrar el significado lógico en una Enciclopedia.

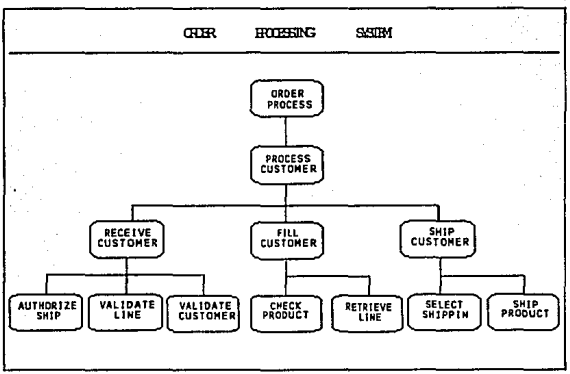


4.8. Las cuatro etapas del desarrollo de un sistema.

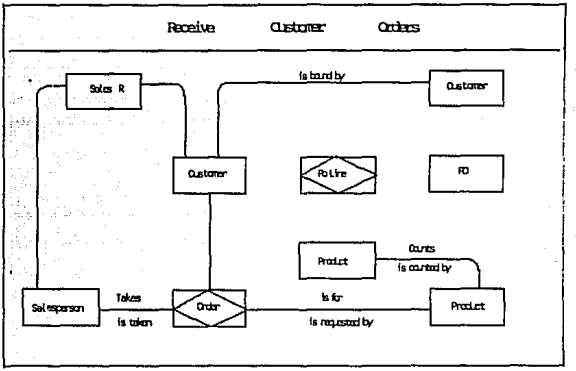
Cada Workstation contiene diferentes herramientas de diagramación integradas, estas son:

Planning Workstation:
Diagramador de descomposición (Figura 4.9.),
Diagramador de Entidad (Figura 4.10.)
Diagramador de Matriz. (Figura 4.11.)

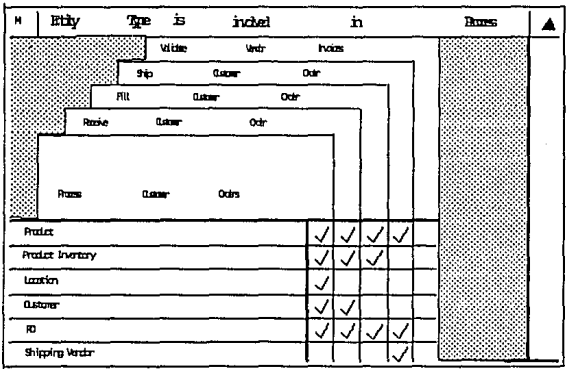
Analysis Workstation:
Diagramador de Acción (Figura 4.12.a.),
Diagramador de flujo de datos (Figura 4.12.b.),
Diagramador de descomposición,
Diagramador de Entidad.



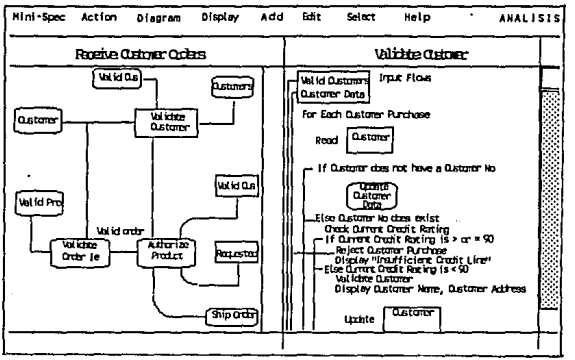
4.9.Diagrama de Descomposición.



4.10. Diagrama de Entidad.



4.11. Matriz de Asociación.



4.12.a. 4.12.b.

Design Workstation:

Diagramador de Acción,
Diagramador de Estructura de Datos,
Diagramador de Bases de Datos,
Diagramador de Bases de Datos Jerárquico,
Diagramador de Presentación,
Diagramador de Base de Datos Relacional,
Diagramador de Cartas de Estructura.

- Requerimientos de equipo.

* Se requiere de equipo IBM PC/ AT ó compatible.
* Sistema Operativo DOS versión 3.1 ó mayor.
* Mínimo 5 MB de Memoria RAM.

* 20 MB de Memoria en disco duro para software del producto.
* Monitor gráfico con tarjeta CGA, EGA ó VGA.
* Soporta mouse PS/2 de IBM ó el mouse de Microsoft.
* Soporte de impresoras EPSON de alta resolución, IBM Color Jetprinter y otras.
* Impresoras LaserJet.

- Metodologías.

* Incluye metodologías tales como Yourdon; Gane/ Sarson; Técnicas de Diagramación para Cartas de Estructura, Diagramas de Descomposición, Diagramas de Entidad-Relación, Diagramas de Acción y Matrices de Planeación.

- Facilidad de uso.

* El manejo de la herramienta es sencillo si se cuenta con ciertos conocimientos básicos sobre la teoría de diseño de sistemas.
* Es accesible en el manejo de los diversos elementos de los diferentes diagramas. Además, genera su propio ambiente semejante a Windows.

-Independencia de la Metodología.

* IEW no está ligado a ninguna metodología en específico; no se pretende reemplazar las metodologías existentes sino trabajar en un ambiente donde la ingeniería vista como disciplina se utilice para integrar lo ya existente.

- Calidad y diversificación de reportes.

* Presenta una gran variedad de generación de reportes, ya que cada uno de los diferentes módulos cuenta con esta facilidad. La alta calidad de éstos permite identificar factores importantes mediante subtitulación.

- Manejo de gráficos y ventanas.

* Esta herramienta utiliza lo más actual de la tecnología de interfases. Utiliza el mouse así como las capacidades de ventanas. Los menús "pull-down" permiten posicionarnos en los comandos

en lugar de recordar cuales son y se pueden utilizar. Algunos diagramas se pueden desplegar en la pantalla al tiempo que se puede ver el flujo de información de un diagrama a otro. Y así como otras herramientas de diagramación, IEW permite guardar diagramas en la pantalla mientras se ve la definición de algún objeto del diagrama.

* Mouse, Menus y Diálogos. El **mouse** es esencial para la diagramación; ésto es, tanto para posicionarse en un objeto o moverlo, como para abrir un menú.
Un **menú** es una lista de acciones que pueden ser desarrolladas. Los títulos del menú (Pull-down) son generalmente verbos que se leen junto con las opciones del menú. Cuando un Workstation requiere información adicional para hacer algo o encontrar un problema se presenta un mensaje en una caja llamada **diálogo**.

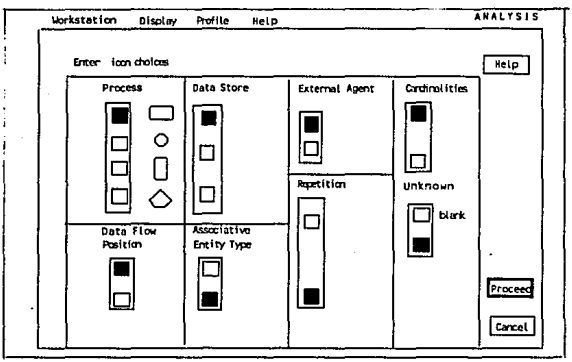


Fig. 4.14. Iconos y sus propiedades.

* Los iconos para diferentes objetos y sus propiedades pueden ser seleccionados para reflejar las convenciones establecidas (Fig. 4.14.).

El diálogo Profile permite determinar la apariencia de iconos para los siguientes objetos:

En el módulo de Planeación:

- Tipos de Entidad Asociativa.
- Funciones y procesos.
- Cardinalidades de relación.

En el módulo de Análisis:

- Tipos de Entidad Asociativa.
- Cardinalidades.
- Flechas para el flujo de datos.
- Almacenamiento de datos.
- Agentes externos.
- Procesos.
- Bloques de repetición.

En el módulo de Diseño:

- Módulos.
- Nombres de parámetros o símbolos.

Este mismo diálogo da la opción de cambiar el largo y ancho de los iconos para asignar nombres de gran longitud.

En el módulo de planeación permite también definir una serie de características por defecto:

-Determinar el número de niveles visibles cuando se abre un diagrama de descomposición.
- Ocultar o mostrar nombres de relación en diagramas de entidad.

Para el módulo de Análisis, las características por defecto que se permiten son:

- Mostrar los corchetes en la parte izquierda en los diagramas de acción.

- Definir el número de niveles visibles cuando se abre un diagrama de descomposición.
- Mostrar los nombres del flujo de datos en un diagrama de flujo de datos.

En el módulo de Diseño, los valores por defecto que se pueden determinar son:

- Mostrar la entidad, sus atributos y relaciones en los diagramas de estructura de datos.
- Mostrar únicamente los tipos de datos y formato de columnas en los diagramas de estructura de datos.

-Manejo de colores .

* Al trabajar con varios diagramas a la vez , es posible apreciar las capacidades de manejo de colores de IEW. El uso de colores facilita la identificación de diferentes diagramas en la pantalla permitiendo trazar una línea de trabajo al estar observando varios diagramas al mismo tiempo.

-Tecnología de Inteligencia Artificial.

* IEW está basado en la tecnología de inteligencia artificial, la cual se implementa utilizando una Enciclopedia inteligente - una base de conocimiento que es común, siendo el punto de integración del conjunto de herramientas de IEW. La inteligencia de las herramientas se basa en un sistema basado en reglas que aseguran la integridad y consistencia de la información guardada en la Enciclopedia, y proveé revisión de errores en tiempo real para mejorar la calidad del trabajo.

- Manejo del Diccionario de datos.

* Enciclopedia.- Es importante distinguir entre "diccionarios" y Enciclopedia. Los diccionarios contienen nombres y descripciones de datos, campos. procesos, variables, etc. La Enciclopedia captura nombres y definiciones e incluye todas las interrelaciones y propiedades entre estos objetos. La Enciclopedia es una representación completa de la empresa y los modelos de sistemas.

La información se guarda en la Enciclopedia a través de los diagramas que se dibujan con IEW. Es la información descrita por los diagramas la que se almacena (tipo de objeto, nombre, descripción, relaciones, asociaciones ...) y no el mapa de bits de la gráfica (cajas, textos, flechas ...). Los diagramas se derivan de la Enciclopedia cada vez que se hace un requerimiento, asegurándose que ellos reflejan la información más actual en la Enciclopedia.

Otra de las ventajas de la Enciclopedia sobre los diccionarios de datos es el uso de la inteligencia artificial. Por medio de ésta IEW permite rápida y fácilmente convertir información compleja, de la enciclopedia a los diagramas.

La Enciclopedia almacena toda la información de cada objeto únicamente una vez, incluyendo el número o tipos de diagramas en el cual éste aparece. Cuando se cambia la definición de un objeto en un diagrama, el cambio se realiza automáticamente en cada uno de los demás diagramas en los cuales el objeto aparezca.

Los diferentes tipos de diagramas desplegados por la Enciclopedia van de acuerdo con los diferentes aspectos o niveles de información del desarrollo del sistema. Por ejemplo, para describir de una manera más completa el contenido de un archivo de datos en un diagrama de flujo de datos, se puede crear un diagrama de entidad que muestre los datos, sus relaciones y cómo se involucran con este almacenamiento de datos.

- Consistencia Automática.

* El coordinador del conocimiento mantiene consistencia entre los diagramas; por ejemplo si dos diagramas diferentes contienen un elemento en común y el usuario cambia uno de ellos, el resto de los diagramas cambiarán automáticamente.

- Arquitectura Abierta.

* IEW tiene la capacidad de importar y exportar información, tomando ventaja de la herramienta en conjunción con las capacidades de cualquier otro software. Se ha utilizado esta característica de la herramienta para desarrollar interfases para

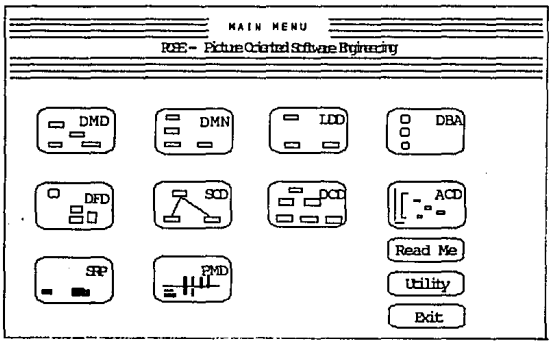
diversos manejadores de bases de datos,diccionario de datos, lenguajes de cuarta generación.

- Modularidad Opcional.

  *No es necesario iniciar el desarrollo de sistemas con la secuencia planeación, análisis, diseño y construcción. La modularidad significa que se puede usar aquellas herramientas que se necesiten para resolver los problemas actuales e ir agregándolas conforme se vayan necesitando.

POSE (Picture Oriented Software Engineering).

POSE es un sistema CASE modular que ayuda a los analistas y diseñadores en las fases críticas del ciclo de vida del desarrollo de un sistema. POSE consiste de diez herramientas de análisis y diseño desarrolladas para proporcionar la calidad de los sistemas de información. Estos diez módulos ofrecen herramientas y métodos los cuales pueden ser combinados y controlados según los requerimientos del diseñador de sistemas (Fig. 4.15.)



4.15. POSE Picture Oriented Software Engineering.

- Requerimientos de equipo.

* Se requiere de un equipo PC- XT , PC-AT , PS/2 o compatible.
* Sistema Operativo DOS versión 3.0 o mayor.
* Mínimo 640 KB de memoria RAM.

* 10 MB de espacio en disco duro para el software del producto.
* Monitor gráfico con tarjeta CGA, EGA o VGA.
* Mouse de dos o tres teclas.
* Puerto serial para el mouse.
* Impresora compatible con EPSON o IBM.
* Impresora HP-Laserjet.

-Metodologías.

* Entre sus metodologías incluye Yourdon, Gane/Sarson, además de varias técnicas de diagramación todas compatibles con IEW.

- Facilidad de uso.

* Esta herramienta es de manejo sencillo, la mayor parte de su manejo se hace seleccionando una de las diversas opciones con las que cuenta mediante un click de mouse.

* Los elementos de los diversos diagramas ya sea de Entidad-Relación, de Flujo de Datos, de las Cartas de Estructura o de los Diagramas de Descomposición son fáciles de manipular permitiendo por lo general un rápido desarrollo del diagrama que se está realizando; el trazado de líneas para el flujo, relaciones, diagramas de descomposición y cartas de estructura puede presentar algún problema si se desea que ésta línea no presente imperfecciones.

- Dependencia de lenguajes y bases de datos.

* No tiene este tipo de dependencias aunque puede generar esquemas para diferentes bases de datos pero cuenta con un módulo extra mediante el cual puede generar código COBOL.

- Calidad y diversificación de reportes.

* Los reportes tienen una calidad bastante aceptable.
* Cuenta con una buena variedad de reportes para los Diagramas de Entidad-Relación, son reportes sencillos pero con información útil.

* Para los reportes de Diagramas de Flujo de Datos, Diagramas de Descomposición y Cartas de Estructura la variedad de reportes es menor pero los reportes con los que cuenta muestran los datos más importantes que describen al diagrama en cuestión.

- Manejo del Diccionario de Datos.

* Esta herramienta mantiene ligas entre los diagramas y el Diccionario de Datos de tal forma que todas las figuras, diagramas y reportes que sean consistentes y exactos son reflejo del contenido del Diccionario de Datos, es por tanto, el corazón de la herramienta. Las diez herramientas de POSE para el diseño de manejo de datos y para el manejo de procesos dentro del análisis y diseño del sistema son:

POSE-DMD

Data Model Diagrammer) Diagramador de Modelo de Datos. Automatiza el proceso de modelado de datos. A su vez, soporta el modelo de datos Entidad-Relación definido por la Ingeniería de Información. Este modelo de datos se puede descomponer en diferentes modelos separados o diferentes modelos pueden ser integrados en un modelo de datos completo.Proporciona una interfase para manipular, definir y documentar los modelos de datos gráficamente definiendo entidades, los atributos de estas entidades, las asociaciones entre entidades y la cardinalidad de sus asociaciones.

Con POSE-DMD se pueden crear modelos de datos que reduzcan los errores en el diseño y las inneficiencias. Posee un registro multi-formas con el cual localiza y despliega entidades y sus relaciones. Examina los atributos de un modelo de datos y determina aquellos que sean redundantes. Para cada modelo de datos se pueden crear mas de 1000 entidades; para cada entidad, se pueden especificar mas de 100 atributos.

POSE-DMN

(Data Model Normalizer) Normalizador de Modelos de Datos. Normaliza los Modelos de Datos DMD de POSE, en base a dependencias funcionales proporcionadas por el usuario. Esto es, estabiliza las estructuras de datos minimizando el impacto al momento de realizar cambios. Debe ser utilizado durante el proceso de modelado de datos para asegurar la correcta estabilidad del mismo.

Produce una lista de Atributos Entidad-Relación Normalizada en la Tercera Forma Normal. En ésta misma lista, automáticamente incluye atributos y llaves que por descuido hayan sido omitidos del modelo de datos. También, automáticamente detecta dependencias transitivas entre entidades. En la lista E-R Normalizada, identifica nuevas entidades y atributos que pueda ser añadidos para remover estas dependencias. Además, detecta y combina entidades con las mismas llaves.

POSE-LDD

(Logical Database Designer) Diseñador de Base de Datos Lógico. Este módulo guía hacia la optimización de un modelo de datos desarrollando un análisis de operaciones. Se verifica que se soporte las operaciones solicitadas y que las mismas utilicen de manera correcta el modelo de datos. Así, determina las diferencias entre el modelo de datos y las operaciones solicitadas. Subraya aquellas entidades que no sean accesadas en cualquier operación, los patrones de acceso que no contengan ninguna asociación y las asociaciones que no sean utilizadas. POSE-LDD determina cuáles atributos de las entidades dentro de un modelo de datos deben ser utilizados como llaves de acceso primarias o secundarias. Simultáneamente despliega el modelo de datos y Mapas de Uso de Operaciones (TUMs) asociados en múltiples ventanas. Facilita la visualización de mas de 9 ventanas, permitiendo que diferentes TUMs sean desplegados simultáneamente. Un TUM muestra las entidades incluidas en una operación así como el curso de su navegación. Además, captura información esencial del diseño de la base de datos tal como: Puntos de entrada de datos,

estructuras de datos, manejo de volúmenes y el acceso lógico.

POSE-DBA

(Database Aid) Ayuda de Base de Datos. El diseño físico de una base de datos normalmente produce una serie de esquemas dentro de un lenguaje de definición de datos (Data Definition Language DDL) para la implementación en tarjeta en un DBMS. Para producir estos esquemas, el módulo DDL requiere que el modelo de datos esté completamente definido. Los esquemas generados pueden ser utilizados por DB2, SQL/DS, ADABAS, AS/400, FOCUS y ORACLE.

POSE-DCD

(Decomposition Diagrammer) Diagramador de Descomposición. Este módulo permite crear, manipular y documentar diagramas de descomposición. Se pueden crear diagramas que capturen la estructura jerárquica de un sistema y toda la información esencial acerca de su funcionalidad.

Para obtener diagramas de descomposición más completo y fáciles de entender, el módulo permite la construcción de diagramas de descomposición estándar, incluyendo construcciones para: Opcionalidad, cardinalidad, condiciones, exclusividad mutua y secuenciación.

Con el fin de asegurar la precisión consistencia de los diagramas de descomposición, POSE-DCD informa cuando se han violado las reglas de diagramación, permitiendo corregir errores fundamentales en el diseño de estructuras ANTES de que el diseño detallado de un sistema se comience.

Para satisfacer las necesidades de diseño y documentación, se maneja el cambio de tamaños y formas de los objetos.

POSE-DFD

(Data Flow Diagrammer) Diagramador de Flujo de Datos. POSE-DFD brinda una serie de funciones interactivas para la diagramación de flujo de datos multi-nivel. También genera reportes analíticos para asegurar la consistencia y balance de niveles dentro de los diagramas de flujo de datos. Se pueden crear y mantener más de 30 "diagramas de contexto" o áreas funcionales. Para cada diagrama de contexto, se pueden crear mas de 12 niveles de diagramas de flujo de datos, cada uno definiendo más de 99 procesos. Este módulo facilita la navegación a través de múltiples niveles de diagramas. Como una ayuda, se puede seleccionar el diagrama del nivel que va a ser desplegado desde una jerarquía gráfica de todos los diagramas de flujo de datos en un contexto seleccionado. Para el chequeo de consistencia y balanceo, los objetos en los diagramas de alto nivel son automáticamente identificados y pueden ser manejados como "objetos flotantes" en diagramas de nivel inferior.

Para evitar la duplicación, los objetos son definidos y documentados una sola vez. Después, pueden ser utilizados cualquier número de veces en el mismo o diferente diagrama de flujo de datos.

POSE-SCD

(Structure Chart Diagrammer) Diagramador de Cartas de Estructura. Mediante este módulo, se puede diseñar, organizar y documentar la arquitectura de los procesos del sistema de una manera estructurada.
Con la biblioteca de funciones de POSE-SCD, se crean módulos reusables del programa para utilizarlos en nuevas cartas de estructuras. Además, estos módulos pueden contener rutinas reutilizables o aún programas enteros y su organización se lleva a cabo dentro de una serie de librerias.

En adición, POSE-SCD permite que una parte del diagrama de la carta de estructura sea extraída o "descompuesta" en una nueva carta con el fin de que sea utilizada en otros diagramas.

POSE-ACD

(Action Chart Diagrammer) Diagramador de Cartas de Acción. Realiza el desarrollo de mini-especificaciones en un formato estructurado conteniendo solo la información necesaria para la subsecuente programación de la aplicación.
Este diagramador utiliza convenciones gráficas y de símbolos para estructurar las especificaciones del programa con una metodología Top-Down. De esta manera, se puede ver la lógica de los programas desde el nivel mas alto de diseño hasta el nivel mas bajo.
POSE-ACD puede ser utilizado para crear estructuras de control detalladas (construcciones como if...then, case, do...while y do...until) representando la lógica procedural de operaciones empresariales.

POSE-SRP

(Screen Report Prototyper) Prototipos de pantallas. POSE - SRP permite desarrollar prototipos de la aplicación sistemas aún en las primeras fases del ciclo de vida del desarrollo de un sistema. Con esto se demuestra la naturaleza y capacidades del sistema propuesto y los reportes que podrá producir. Se pueden crear y mantener imágenes de pantalla y los campos de entrada de datos pueden ser predefinidos tanto en pantalla como en reportes. Las definiciones detalladas para estos campos deben ser especificadas por el diseñador del sistema derivadas de la definición de entidades del modelo de datos en el módulo POSE-DMD.

Las pantallas pueden ser ligadas una a otra para formar diálogos de pantalla. La secuencia de las pantallas es determinada por el usuario. El usuario puede además navegar a través del diagrama en un orden natural. Las validaciones de la edición de pantalla se pueden hacer junto con la entrada de datos. Esto incluye campos opcionales contra los requeridos, campos alfanuméricos contra numéricos, longitud del campo y justificación.

POSE-SRP cuenta con elementos para:

- La configuración del perfil, la cual consiste de registros que permiten marcar los valores por default; por ejemplo: para campos de entrada, campos de salida, tamaño de la pantalla, atributos, etc.

- La generación de pantallas, que incluye: definiciones de campos y características; operaciones por bloques como borrar o mover algún bloque; operaciones del archivo como salvar, renombrar, borrar y copiar archivos, etc.

- La generación de reportes, que permite manejar hojas hasta de 132 columnas por 66 líneas; facilidades para la numeración de página, definición de tipo de columna así como el manejo de la hora y fecha.

- Facilidades de prototipo y utilidades. Muestra al usuario como deberían aparecer las pantallas durante la ejecución del sistema y cómo podrían interactuar con otras pantallas para desarrollar las funciones del usuario. Los valores de los datos pueden ser capturados y almacenados para que más adelante se utilicen en alguna sesión del diseño del prototipo y

- Utilidades, las cuales consisten en una serie de rutinas que permiten al usuario manipular pantallas y reportes como un todo. Algunas de las rutinas que se brindan son: copiar, renombrar, listar, imprimir, importar y exportar.

POSE-PMD

(Planning Matrix Diagrammer) Diagramador de Matrices de Planeación. POSE-PMD automatiza la diagramación de matrices, lo cual es especialmente útil en las primeras etapas de planeación del ciclo de vida del desarrollo de un sistema. Para hacerlo, permite crear, actualizar, documentar y obtener matrices como parte integral de las etapas de análisis y planeación.

El pool de datos reside en el diccionario de datos, lo cual significa que los datos pueden ser compartidos por diferentes matrices.

El usuario de POSE en su Versión 4.2 tiene acceso a la información que se encuentra dentro del diccionario de datos por medio de las funciones de Import y Export para el módulo de Data Model Diagrammer, por medio de la función Export para los módulos de Data Flow Diagrammer y Screen Report Prototyper.

La Versión 4.2 de POSE cuenta con un módulo opcional llamado Data Model Bridge que permite que haya transferencia de datos bidireccional entre POSE e Information Engineering Workbench (IEW). POSE por medio del puente puede tomar información del depósito central de datos de IEW y colocarla dentro de su Data Model Diagrammer para realizar tareas de modelado de datos y de normalización.

-Verificación de consistencia.

* Aparentemente realiza verificaciones de consistencia, pero se puede observar que en los Diagramas de Flujo de Datos al pasar del diagrama de contexto a un DFD hijo no guarda consistencia de los flujos de datos que entran al diagrama hijo pues una vez dentro del hijo, un flujo se puede usar cuantas veces se desee.

-Herramientas para el desarrollo de prototipos.

* Este paquete cuenta con un módulo que sirve para el diseño de prototipos de pantallas del sistema y generación de reportes.

-Interfaz gráfica.

* La interfaz gráfica es buena pero limitada pues se pierde legibilidad en diagramas de gran tamaño.

* Manejo de ventanas. Para desplegado simultáneo de distintas tareas en el módulo LDD.

* En los Diagramas de Flujo de Datos al pasar los flujos de datos de un diagrama padre a un hijo, debido a que se tienen que incluir aquellos agentes externos que aportan el flujo, el tamaño del diagrama aumenta y se pierde legibilidad.

* Una ventaja que presenta es la de contar con seis escalas de reducción y amplificación del tamaño del diagrama por medio del Zoom.

* Uso de distintos colores para diferenciación de los diversos elementos que componen un diagrama.

* En los Diagramas de Entidad-Relación los nombres de las relaciones son de tipo texto permitiendo además poner nombre a ambos sentidos de la relación, pero tiene como desventaja que internamente solo se trata de un texto, no es el nombre de la relación.

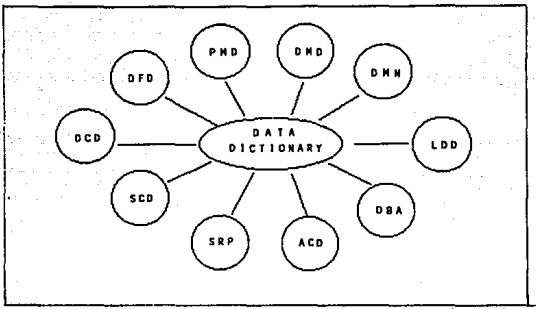
* Lentitud en el trazo de una línea de relación, para lograr que tenga una calidad aceptable.

- Navegación entre los distintos diagramas.

* La navegación entre los distintos diagramas se da indirectamente ya que para pasar de un diagrama a otro se debe de abandonar el módulo de POSE en el que nos encontramos trabajando para accesar el módulo donde se encuentre el diagrama que se desea accesar.

- Normalización de datos.

* POSE cuenta con un módulo (DMN) que usando entidades, atributos,asociaciones y dependencias funcionales capturadas en el modelo de datos creado en el módulo de DMD obtiene una lista normalizada de atributos de un diagrama de Entidad-Relación en tercera forma normal.(3NF).



4.16. Relación de módulos de POSE con su Diccionario de Datos.

SYSTEM ARCHITECT.

- Requerimientos de equipo.

* IBM PC's y compatibles que soporten Micosoft Windows.
* Opera bajo Microsoft Windows 2.11 o 3.0
* Mínimo de 640 KB de memoria principal.
* Se recomiendan 10 MB de espacio en disco duro.
* Soporta cualquier mouse, adaptador, impresora, etc. soportado por Microsoft Windows.
* Adaptador gráfico soportado por Windows (CGA no es recomendable).

- Metodologías.

* DeMarco/Yourdon, Gane/Sarson, Ward & Mellor y otras varias técnicas de diagramación en su

mayoría compatibles con IEW. A diferencia de IEW cuenta con el diagrama de flujo tradicional.

- Facilidad de uso.

* System Architect es muy amigable al usuario pues al trabajar bajo ambiente Windows cuenta con todas las facilidades que Windows tiene, haciendo uso de las facilidades que el mouse trae consigo.
* Manipulación sencilla de los elementos de los diversos diagramas con los cuales se trabaja. El trazado de las diversas líneas para flujos de datos, relaciones, diagramas de descomposición y cartas de estructura, es bastante sencillo y estético.
* La impresión de diagramas es de buena calidad reduciendo un diagrama a un tamaño tal que su impresión pueda ser hecha en una hoja.

- Dependencia de lenguajes y bases de datos.

* No está diseñado para un lenguaje o base de datos en particular.

- Calidad y diversificación de reportes.

* Los reportes con los que cuenta son variados y presentan características importantes del diagrama del que se esta solicitando.
* El usuario de Sysarch puede definir varios reportes mediante el uso de SQL.

- Manejo del diccionario de datos.

* Las definiciones para los objetos gráficos en los diagramas es centralizada en un diccionario de datos/enciclopedia, para que los cambios hechos en un lugar puedan ser automáticamente reflejados en otros lugares. El diccionario de datos es una construcción lógica que consiste de definiciones de símbolos y relaciones entre símbolos categorizados en tipos. El diccionario de datos usa formatos de archivos tipo dBase III

Plus, por ésto se puede tener acceso a él mediante dBase III, exportando nuestro diccionario de datos a código ASCII.

-Revisión de consistencia.

* Revisa la consistencia, ésto lo podemos observar al momento de generar un hijo a un proceso dentro de un diagrama de flujo de datos, al entrar a la pantalla del hijo de dicho proceso podemos observar que se encuentran una serie de flujos de datos que son los que entran y los que salen del proceso padre estos flujos aparecen para asegurar que cada fujo que entre sea utilizado en el diagrama hijo y que de éste salgan todos los flujos que deben de salir del diagrama padre.

* Un problema que se presenta es que una vez generado el hijo al actualizar el padre con algún flujo nuevo que entre o salga de éste, estos flujos no se actualizarán en el diagrama hijo. Una posible solución es la de agregar al diagrama hijo los agentes externos mismos de los cuales provienen o hacia los cuales salen los flujos de datos agregados (ésta solución más que válida para el sistema, persigue fines prácticos al diseñador para salvar esta desventaja).

- Herramientas para el desarrollo de prototipos.

* System Architect no cuenta con este tipo de herramientas.

- Interfaz gráfica.

* La interfaz gráfica de System Architect es bastante buena, pues al trabajar bajo ambiente Windows aprovecha gran parte del nivel gráfico de éste, dándole una buena calidad y elegancia al acabado del diagrama con el cual se esté trabajando.

* La representación gráfica de los elementos del diagrama es bastante aceptable, el trazo de relaciones, flujos de datos, etc., es muy rápido y sencillo además que el acabado que se da a éstos

elementos es bueno y con poco esfuerzo si se hace uso de la opción "Straigth-Orthogonal" para una línea esta opción esta incluida en System Architect.

* En los diagramas de Entidad-Relación las relaciones solo aceptan que se les de nombre en un sentido.

* Se puede tener una vista del diagrama de acuerdo a los deseos del diseñador, ésto es si el diagrama por su extención no se puede apreciar en una pantalla existe la opción de reducirlo a diferentes tamaños hasta que podamos apreciarlo totalmente en la pantalla, pudiendo tener una vista muy general de la distribución de los diversos elementos del diagrama claro al ir aumentando el nivel de reducción se van perdiendo más detalles.

-Navegación entre los distintos diagramas.

* Para navegar entre los diversos diagramas normalmente dentro de System Architect se tiene que abandonar un diagrama para llamar al otro que se desea. Para evitar el paso de cargado del nuevo diagrama y contar con todos los diagramas de un sistema relacionados entre sí se puede implementar un estilo de navegación semejante a IEW; esto se haría de la siguiente forma: dentro del hijo de algún elemento del diagrama se crea otro tipo de diagrama el cual sea la representación de otra de las fases de diseño del sistema que se esta generando y así sucesivamente se crea un hijo de uno de los elementos de éste nuevo diagrama y se vuelve ha hacer lo mismo. Al momento de navegar entre los distintos diagramas únicamente lo que se tendrá que hacer es moverse entre los distintos niveles de padres e hijos existentes en el sistema.

- Normalización de datos.

* System Architect no realiza normalización sobre el diagrama de entidad-relación, lo único que hace

es verificar que el modelo se encuentre hecho en primera, segunda o tercera forma normal y si no cumple la normalización, manda una serie de mensajes de error señalando porque no cumple con la forma normal que se le pidió verificar. El número de mensajes es limitado.
Las funciones de dibujo nos permiten crear diagramas fáciles de entender y modificarlos como sea necesario para mejor claridad.

TABLA COMPARATIVA ENTRE IEW,

POSE Y SYSTEM ARCHITECT.

IEW	POSE	SYSTEM ARCHITECT
Requerimientos de equipo.		
PS/2, PC´s-XT, AT o compatible.	PC"s- XT, AT o compatible.	PC- AT o compatible.
DOS Versión 3.3 o mayor.	DOS Versión 3.0 o mayor.	DOS Versión 3.1 o mayor.
5 MG RAM.	640 KB RAM.	640 KB RAM.
20 MB Disco Duro.	10 MB Disco Duro.	10 MB Disco Duro.
Monitor VGA, EGA o CGA.	Monitor CGA, EGA o VGA.	Monitor soportado por Windows. (VGA, CGA - no recomendado).

Mouse.	Mouse.	Mouse.
Impresora compatibles con EPSON o IBM y HP-Laserjet.	Impresora compatibles EPSON o IBM y HP-Laserjet.	Impresoras soportadas por Windows.
Metodologías.		
Yourdon, Gane/Sarson, Cartas/Estructura, Diagramas de Descomposición, Diagramas de E-R, Diagramas de Acción, Matrices de Planeación.	Yourdon, Gane/Sarson, Diagramas de Descomposición, Cartas de Estructura, Matrices de Planeación, Diagrmas de Acción.	DeMarco/ Yourdon, Gane/Sar, Ward& Mellor, Diagramas de E-R, Diagramas de Descom-posición, Cartas de Estructura Diagramas de Estado, Diagrama de Flujo.
Facilidad de uso.		
Poco complicado.	Sencillo.	Muy sencillo.
Dependencia de lenguajes y bases de datos.		
No existe.	No existe.	No existe.

Calidad diversificación de reportes.

Excelente calidad.	Buena calidad Gran variedad de reportes.	Calidad aceptable, varios reportes

Manejo del Diccionario de Datos.

Hace uso de la Enciclopedia.	Si.	Si.

Verificación de Consistencia.

Si tiene.	Si tiene.	Si tiene.

Herramientas para el desarrollo de prototipos.

Si.	Si.	Si.

Interfaz gráfica.

Muy buena.	Buena.	Muy buena, aprovecha todas las facilidades que brinda el ambiente Windows.

Navegación entre los distintos diagramas.			
	Muy fácil.	Sencilla.	Sencilla.
Normalización de datos.			
	Si normaliza.	Si normaliza.	Si normaliza.

De la anterior tabla comparativa se desprende la decisión de utilizar como herramienta CASE a System Architect, no sólo por las características, requerimientos de equipo y ventajas que presenta, sino también por ser la única herramienta que tuvimos a nuestro alcance, principal razón por la que fué seleccionada.

V. ANALISIS.

El Análisis es el proceso de definición de requerimientos para la solución a un problema. Durante la etapa de análisis se examinan las necesidades de los usuarios, y las propiedades que el sistema debe cubrir para su posterior definición. El mejor camino a seguir para el análisis de un sistema es la especificación funcional.

En muchos proyectos de desarrollo de software, la fase de análisis se ha evitado porque no brinda en sí una fuente de estructuración con los métodos y técnicas existentes. Los requerimientos del análisis se enfocan en la inrterfase entre el usuario y la computadora.

Pero los requerimientos no son estáticos. Cambian con el tiempo, la gente y nuevas tecnologías. En la práctica, la fase de análisis del desarrollo de un programa se involucra (participa) durante el resto del ciclo de vida y debe, al menos en parte, ser repetida como el usuario y el desarrollador aprendan más a manera que el problema sea resuelto.

Durante el análisis, los requerimientos del problema se definen y entonces se procesa la solución del mismo. Primero, el problema es estudiado para determinar sus partes y sus inter-relaciones. Es importante entender el problema antes de tomar una solución. Si la estructura de la solución tomada no se asemeja a la estructura del problema, el sistema resultante será difícil de cambiar y a su vez mantener.

El resultado de la fase de análisis es la especificación del sistema, el cuál describe cómo el sistema conocerá los requermientos del problema. Una especificación incluye definiciones para reportes, estructuras de datos, bases de datos, archivos externos, archivos internos, tablas internas, componentes funcionales e interfases con otros sistemas - en pocas palabras, los componentes del sistema y las interfases conectadas a estos componentes. Además, esta especificación debe ser precisa, formal y con opción a ser examinada. Por ejemplo, si se dice "el sistema

tendrá capacidad de procesamiento de 100 transacciones por segundo", será una especificación bien definida y probable.

La completitividad y corrección de las especificaciones afectan el proceso del desarrollo del software entero. La especificación es utilizada para desarrollar programas de trabajo, distribución de recursos humanos, planes de prueba y la documentación del usuario. Si la especificación es incompleta o incorrecta, puede causar retrasos en el programa de trabajo, una etapa de pruebas pobre y una documentación, por consiguiente, incorrecta.

Anteriormente las técnicas estructuradas ignoraban los métodos de solución de problemas y los resultados relacionados con el usuario. El enfoque era hacia la solución técnica de problemas del software asociados con la codificación y prueba. Después de haber realizado muchos sistemas que tecnicamente eran completos, aún existían fallas debido a los pocos requerimientos que se cubrían. Todas las estrategias básicas de solución de problemas dentro de la filosofía estrcturada -división y conquista, abstracción y organización jerárquica -fueron incluídos en el proceso de análisis del sistema. La técnica estructurada más conocida y ampliamente utilizada para análisis de requerimientos y especificación es conocidad como Análisis Estructurado.

El análisis estructurado utiliza los métodos de descomposición top-down y Entidad-Relación para definir los requerimientos del sistema. Es un objetivo producir una especificación estructurada que proporcione un modelo conciso y fácil de entender del sistema.

OBJETIVO:

Diseño y desarrollo de un sistema de información alimentado por las principales fuentes de información de la empresa para su proceso y la obtención oportuna y confiable de informes financieros y contables a fin de poder realizar un análisis más profundo de ellos y de sus causas.

ANALISIS

ANTECEDENTES:

La subdirección de información financiera del área de planeación es la encargada de elaborar la carpeta de resultados de la compañía, proceso que representa una ardua labor de recopilación, captura y procesamiento de la información. Además de un detallado proceso de investigación de las causas del estado de la información.

El proceso de elaboración de la carpeta de resultados ha sufrido varios cambios que han beneficiado a los analistas y al área tanto en oportunidad como en veracidad de información, estos cambios los ha marcado el uso del equipo de cómputo por medio de paquetes y por medio de archivos pre-procesados por el área de sistemas: pero aún así es tanto el volumen de información que los analistas dedican gran parte de su tiempo a procesar (recopilación, captura, selección, ordenamiento, etc.) la información que necesitan para la creación de los cuadros de información y disponen de poco tiempo para su análisis y conclusiones.

El área de planeación, percatándose de esta problemática plantea la posibilidad de contar con una base de datos para concentrar la información más relevante de la compañía y explotarla por medio de un sistema adecuado, por lo que en conjunto con el área de sistemas, se evaluaron los productos desarrollados por algunas consultorías.

Durante la evaluación de los productos presentados por las consultorías, se observó que era necesario para desarrollar un sistema de información para finalmente decidir desarrollar el sistema dentro de la compañía conjuntamente entre las áreas de planeación y sistemas.

Se desea que el sistema de informacion contenga la mayor cantidad de información básica para la compañia y que sea útil para todas o gran parte de las áreas.

Por ser planeación un área que maneja gran cantidad de información y por ser de suma importancia para ella y la dirección general el informe de resultados, se optó por iniciar el proyecto desarrollando los procesos para la obtención financiera, ésto es, automatizar la generación de los cuadros de información de la carpeta que se entrega al consejo de administración.

ANALISIS

ESTRUCTURA PROPUESTA PARA LA GENERACION DE CUADROS:

PRIMAS DIRECTAS.

- Ventas totales.
- Ventas por ramo.
- Ventas por moneda.
- Negocios importantes del mes.
- Negocios importantes al mes.

REASEGURO.

- Primas cedidas.
- Cobertura exceso de perdida.

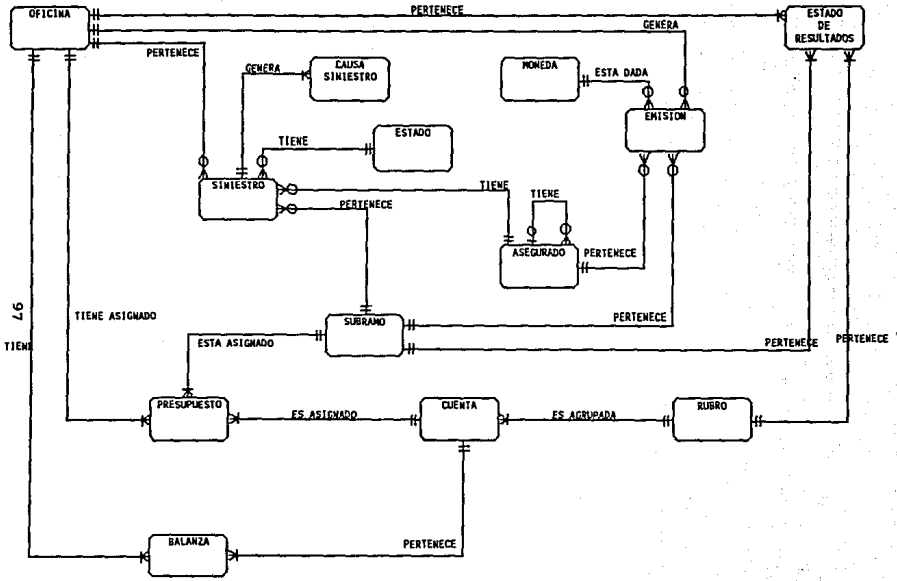
COSTO DE ADQUISICION.

COSTO DE SINIESTRALIDAD.

- Siniestralidad por ramo.
- Composición de los siniestros importantes.
- Siniestros importantes del mes.
- Siniestros importantes al mes.

DIAGRAMA DE ENTIDAD-RELACION.

Después de diversas entrevistas con el personal encargado de elaborar la carpeta de resultados pudimos identificar las diferentes fuentes de información de las cuales obtenían los datos necesarios para el desarrollo de los cuadros correspondientes. El siguiente diagrama entidad-relación no es sino el resultado obtenido a partir de estas entrevistas. Lo reportes anexos se generaron a partir de este diagrama y nos muestran nombres de entidades, sus atributos y sus relaciones.


OFICINA
PERTENECE
ESTADO DE RESULTADOS
GENERA
PERTENECE
GENERA
CAUSA SINIESTRO
MONEDA
ESTA DADA
EMISION
TIENE
ESTADO
SINIESTRO
TIENE
TIENE
PERTENECE
ASEGURADO
PERTENECE
97
TIENE ASIGNADO
SUBRAMO
PERTENECE
TIENE
ESTA ASIGNADO
PERTENECE
PERTENECE
PRESUPUESTO
ES ASIGNADO
CUENTA
ES AGRUPADA
RUBRO
BALANZA
PERTENECE

DICCIONARIO DE DATOS.

Dentro del Diccionario de Datos estarán contenidas las definiciones de los elementos, para éste caso las definiciones serán de nuestro diagrama de Entidad-Relación. Es decir, para cada una de nuestras entidades habrá una definición en nuestro diccionario.

La defincón se hace en función de los componentes de cada entidad. Por ejemplo, la entidad ESTADO se define por los siguientes elementos:

ESTADO+DESCEDO

El formato que se utilizará en el diccionario será que: en cada una de las definiciones estarán los elementos de la entidad. El signo **+** nos indica la subsecuencia de otro elemento y **@** indica que el elemento forma parte de la llave de la entidad.

ENTIDAD:	ASEGURADO
ELEMENTOS:	@ NOCLIENTE +
	DESCCLIENTE +
	TIPOCLIENTE +
	CLIENTEASOCIADO

ENTIDAD:	BALANZA
ELEMENTOS:	@1 ANIO +
	@2 MES +
	@3 CIA +
	@4 SCIA +
	@5 SSCIA +
	@6 OFNA +
	SALDO

ENTIDAD:	CAUSASINIESTRO
ELEMENTOS:	@ CAUSA +
	DESCCAUSA

ENTIDAD:	CUENTA
ELEMENTOS:	@1 CIA +
	@2 SCIA +
	@3 SSCIA +
	RUBRO +
	SRUBRO +
	DESCCTA

ENTIDAD:	EMISION
ELEMENTOS:	OFNA +
	@1 MONEDA +
	RAMO +
	SRAMO +
	@2 NOPOLIZA +
	@3 NOENDOSO +
	TIPOEND +
	ASIENTO +
	FECHAEMI +
	NOCLIENTE +
	INIVIG +
	FINVIG +

ENTIDAD:	EMISION (SIIE)
ELEMENTOS:	MTORECARG +
	MTOREDAUT +
	MTOPRIMA +
	MTOCOMIS +
	@4 TIPOCAMBIO

ENTIDAD:	ESTADO
ELEMENTOS:	@ ESTADO +
	DESCEDO

ENTIDAD:	ESTADO DE RESULTADOS
ELEMENTOS:	@1 RUBRO +
	@2 SRUBRO +
	@3 RAMO +
	@4 SRAMO +
	OFNA +
	@5 ANIO +
	@6 MES +
	CANTIDAD

ENTIDAD:	MONEDA
ELEMENTOS:	@ MONEDA +
	DESCMON

ENTIDAD:	OFICINA
ELEMENTOS:	@ OFNA + DESCOFNA

ENTIDAD:	PRESUPUESTO
ELEMENTOS:	@1 ANIO + @2 MES + OFNA + @3 CIA + @4 SCIA + @5 SSCIA + @6 RAMO + @7 SRAMO + MONTO

ENTIDAD:	RUBRO
ELEMENTOS:	@1 RUBRO + @2 SRUBRO + DESCRUBRO

ENTIDAD:	SUBRAMO
ELEMENTOS:	@1 RAMO + @2 SRAMO + DESCSRAMO

ENTIDAD:	SINIESTRO
ELEMENTOS:	OFNA +
	RAMO +
	SRAMO +
	@1 NORECLAM +
	FECHARECLAM +
	NOPOLIZA +
	ESTADO +
	LOCALIDAD +
	CAUSA +
	@2 FECHAMOV +
	TIPOMOV +
	IMPMOV +
	MONEDA +
	TIPOCAMBIO +
	NOCLIENTE

DESCRIPCION DE ENTIDADES.

ASEGURADO:

Contiene los datos necesarios para la identificación de los diferentes clientes.

BALANZA:

Contiene el estado de situación financiera en las diferentes fechas.

CAUSA SINIESTRO:

Contiene una breve descripción de las diferentes causas de un siniestro.

CUENTA:

Catálogo de cuentas de ingresos y egresos.

EMISION:

Contiene los datos de las diferentes pólizas emitidas por la compañía.

ESTADO:

Catálogo de estados.

ESTADO DE RESULTADOS:

Contiene los datos del estado finaciero de perdidas y ganancias en los diferentes períodos.

MONEDA:

Catálogo de monedas manejadas.

OFICINA:

Catálogo de las diferentes oficinas existentes incluye sucursales y matriz.

PRESUPUESTO:

Contiene los datos del estado financiero previsto a ser ejercido.

RUBRO:

Contiene los números clave mediante los cuales se identifica una de las cuentas existentes.

SINIESTRO:

Almacena la información de los diferentes siniestros ocurridos.

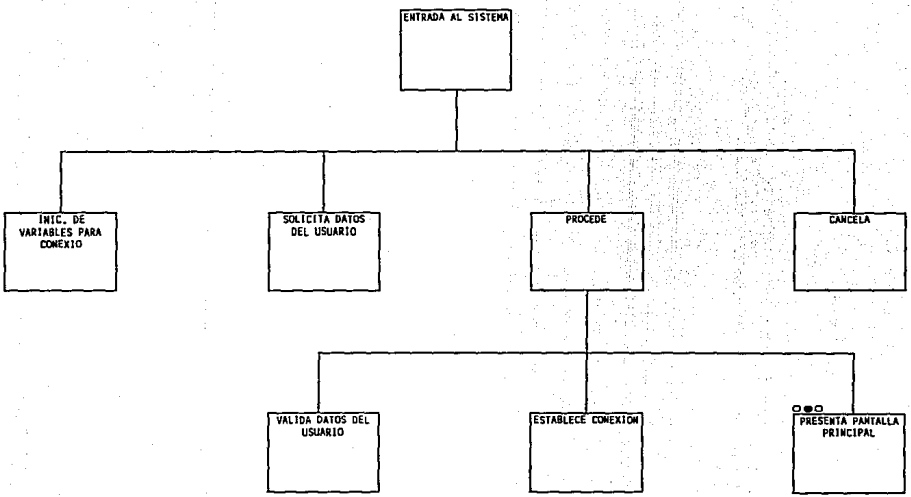
SUBRAMO:

Catálogo de ramos y subramos.

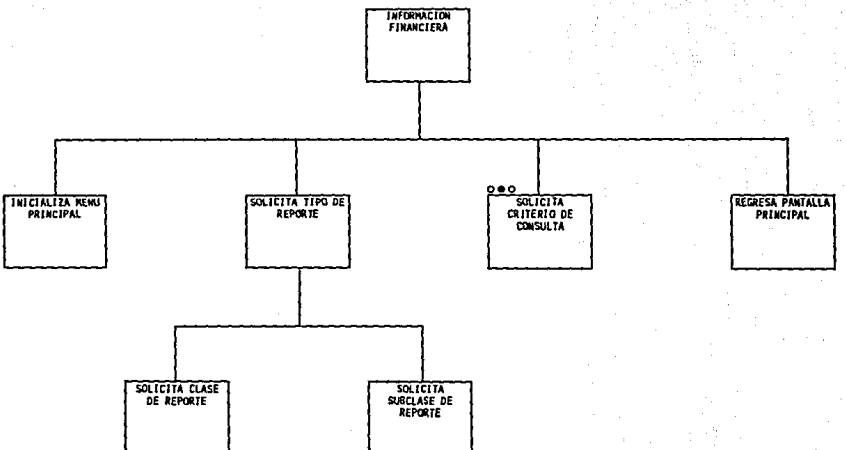
ANALISIS

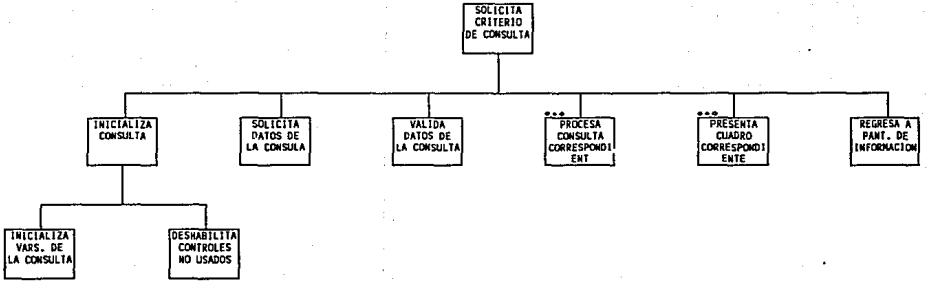
DIAGRAMA DE DESCOMPOSICION.

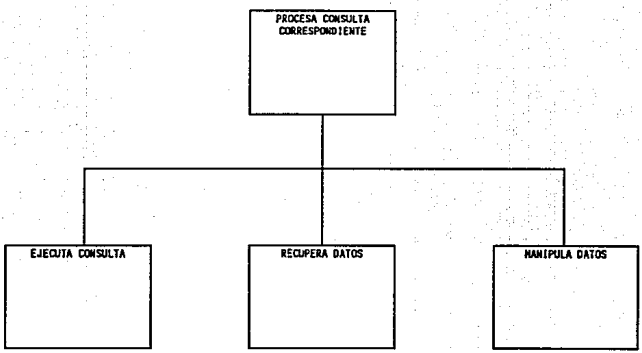
Habiendo analizado el proceso uililizado para la generación de los diferenes cuadros que conforman la carpeta se llegó al siguiente diagrama de descomposición de dichos procesos (Ver página siguiente).

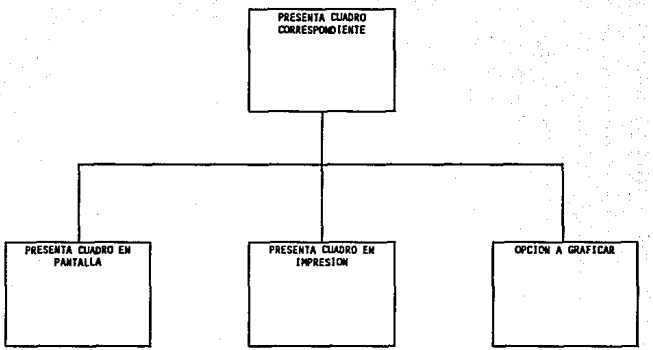

ENTRADA AL SISTEMA
INIC. DE VARIABLES PARA CONEXIO
SOLICITA DATOS DEL USUARIO
PROCEDE
CANCELA
VALIDA DATOS DEL USUARIO
ESTABLECE CONEXION
PRESENTA PANTALLA PRINCIPAL

PRESENTA PANTALLA
PRINCIPAL
INFORMACION FINANCIERA
FINALIZA SESION


INFORMACION FINANCIERA
INICIALIZA MENU PRINCIPAL
SOLICITA TIPO DE REPORTE
SOLICITA CRITERIO DE CONSULTA
REGRESA PANTALLA PRINCIPAL
SOLICITA CLASE DE REPORTE
SOLICITA SUBCLASE DE REPORTE


SOLICITA CRITERIO DE CONSULTA
INICIALIZA CONSULTA
SOLICITA DATOS DE LA CONSULA
VALIDA DATOS DE LA CONSULTA
PROCESA CONSULTA CORRESPONDIENT
PRESENTA CUADRO CORRESPONDIENTE
REGRESA A PANT. DE INFORMACION
INICIALIZA VARS. DE LA CONSULTA
DESHABILITA CONTROLES NO USADOS


PROCESA CONSULTA CORRESPONDIENTE
EJECUTA CONSULTA
RECUPERA DATOS
MANIPULA DATOS


PRESENTA CUADRO CORRESPONDIENTE
PRESENTA CUADRO EN PANTALLA
PRESENTA CUADRO EN IMPRESION
OPCION A GRAFICAR

VI.DISEÑO.

Un objetivo de la Ingeniería de Programación es la capacidad de desarrollar sistemas rápidos, es decir, se debe ser capaz de implantar procedimientos computacionales rápidamente.

Toma tiempo desarrollar los modelos de datos iniciales y los procesos de los modelos del Análisis del Area Empresarial (Business Area Analyisis - BAA). Una vez hecho ésto, los modelos pueden ser utilizados más adelante como una base. Dentro de este concepto, es deseable que los sistemas fueran diseñados y construídos rápidamente. Para incrementar la velocidad en el desarrollo, se necesita el uso de herramientas CASE para el diseño del sistema el cual facilite el prototipo, las técnicas "rápidas" y la metodología de Cuadros. El diseño bajo estas herramientas se deberá acoplar a un generador de código. El código de la base de datos, los códigos de control de trabajo así como los exámenes de datos y la documentación deberán ser generados.

El término I-CASE, CASE integrado, es utilizado para equipos en los cuales las herramientas de planeación y modelación son integradas como herramientas de diseño y un generador de código, todos usando la misma enciclopedia. Esto es esencial para incrementar la rapidez en el desarrollo. La información BAA será utilizada directamente para asistir en el diseño, y las herramientas de diseño y el generador de código deberán ser integrados.

Las herramientas necesarias para el diseño e implantación son los siguientes:

Diagrama de Descomposición. Los diagramas de descomposición permiten la visión a un nivel alto del diseño que será sucesivamente desglosado en otros niveles con más detalle.

Diagrama de Acción (Mini-Especificaciones). Los diagramas de acción facilitan la construcción de procedimientos estructurados así como su código también estructurado.

Generador de Reportes. Un generador de reportes debe permitir que la estructura y el formato de un reporte sean creados rápidamente y que los cálculos derivados de los campos se incluyan en el reporte.

Generador de Código. Un generador de código debe crear un código ejecutable con un nivel de especificación tan alto como sea posible.

Diagrama de Flujo de Datos. Los diagramas de flujo de datos muestran el flujo de datos entre los módulos de procedimientos o programas.

El diseño del sistema comienza, pues, en la extracción de información relevante contenida en la enciclopedia y establecida en la Planeación de la Estrategia de Información (Information Strategy Planning). Esta información es tomada dentro en una de las herramientas con la cual se inicia el diseño de patallas, reportes y prototipos. Las funciones que el sistema debe desarrollar son establecidas en base a pláticas con los usuarios finales; el diseño inicial es hecho en el ambiente del mismo usuario. El diseño es refinado, los usuarios dan su opinión sobre los prototipos y es entonces cuando se genera el código a partir del diseño.

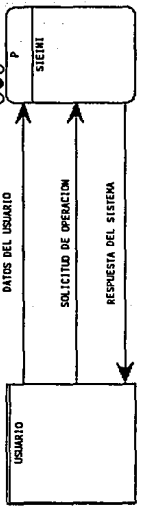
La construcción e integración de los sistemas de información necesarios dentro de la empresa son perfeccionados con la sintetización de los modelos y diseños que mucha gente aporta. Las herramientas CASE hacen posible esta integración y construcción con una enciclopedia central.

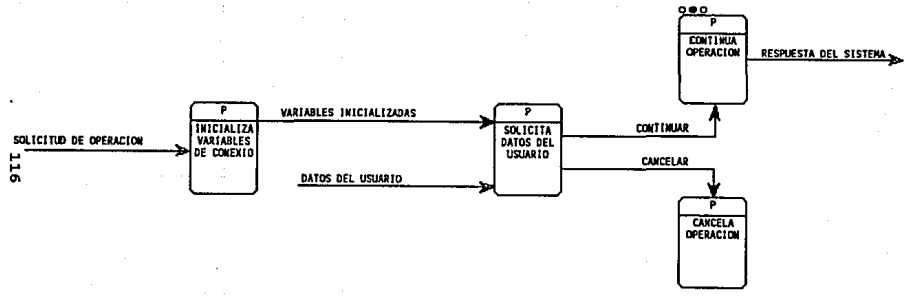
El diseñador debe extraer información de la enciclopedia central para formar una enciclopedia propia de su conjunto de herramientas, con la cual trabaje en un ambiente local y con la que posteriormente se coordine hacia la enciclopedia central.

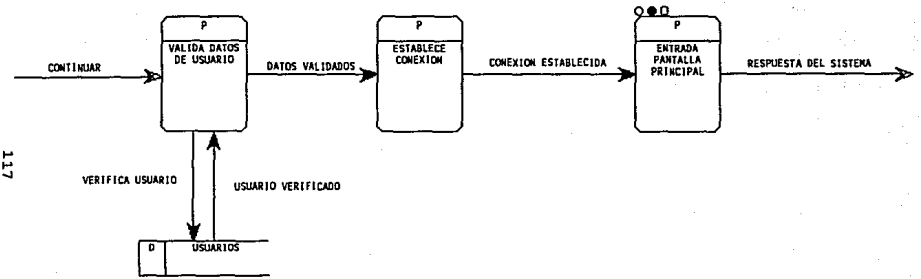
Diagrama de Flujo de Datos.

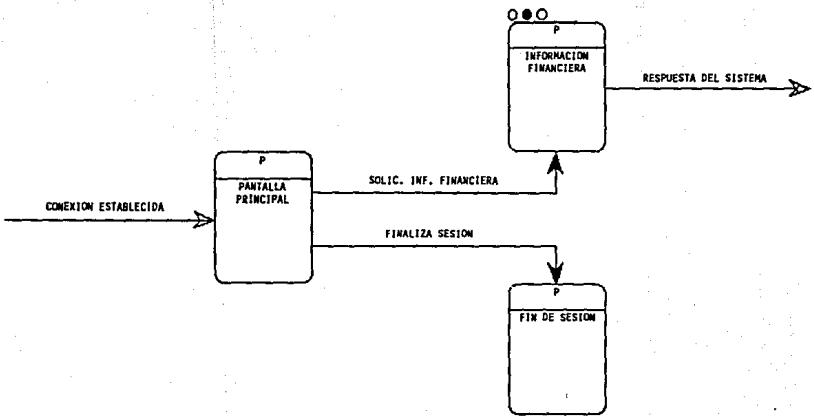
Una vez identificados los diferentes procesos que intervendrán en el sistema, es necesario entrar más en

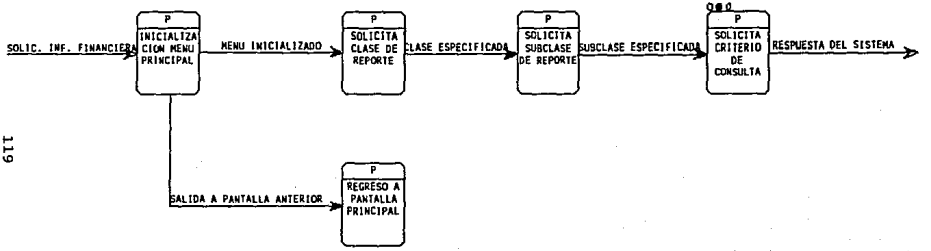
detalle tomando en cuenta la información que cada proceso necesita para su correcto desempeño así como la que éste manda a procesos subsecuentes; ésto lo podemos observar en lo que se conoce como diagrama de flujo de datos. Para el sistema de información dicho diagrama es el siguiente (Ver siguiente página):

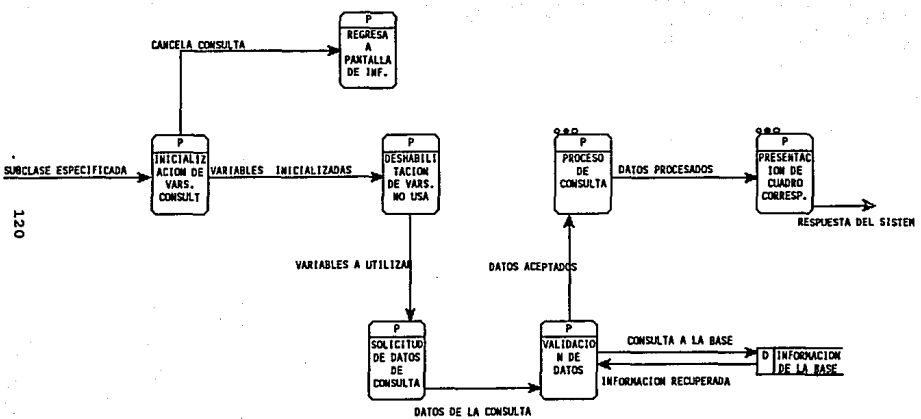

SIEINI
DATOS DEL USUARIO
SOLICITUD DE OPERACION
RESPUESTA DEL SISTEMA
USUARIO

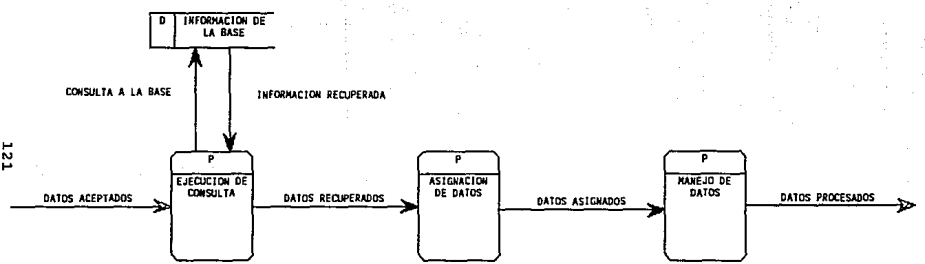

P
INICIALIZA VARIABLES DE CONEXIO
VARIABLES INICIALIZADAS
DATOS DEL USUARIO
P
SOLICITA DATOS DEL USUARIO
CONTINUAR
CANCELAR
P
CONTINUA OPERACION
RESPUESTA DEL SISTEMA
P
CANCELA OPERACION
SOLICITUD DE OPERACION

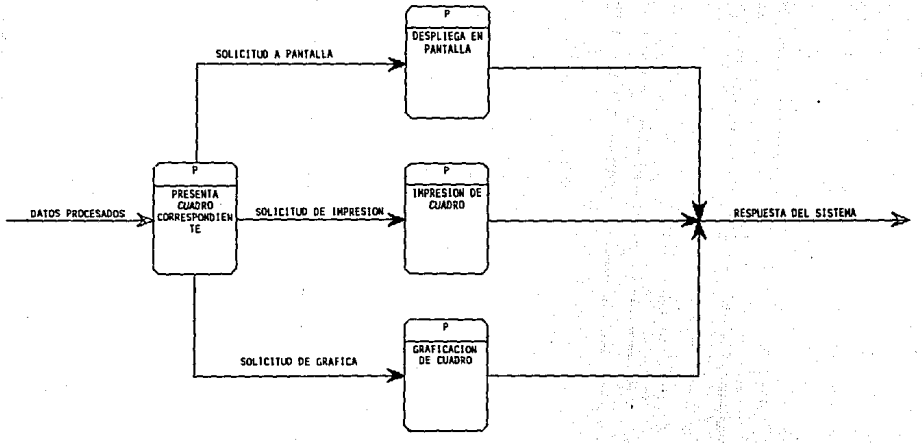

P
VALIDA DATOS DE USUARIO
CONTINUAR
DATOS VALIDADOS
P
ESTABLECE CONEXION
CONEXION ESTABLECIDA
P
ENTRADA PANTALLA PRINCIPAL
RESPUESTA DEL SISTEMA
VERIFICA USUARIO
USUARIO VERIFICADO
D
USUARIOS


P
INFORMACION FINANCIERA
RESPUESTA DEL SISTEMA
P
PANTALLA PRINCIPAL
SOLIC. INF. FINANCIERA
CONEXION ESTABLECIDA
FINALIZA SESION
P
FIN DE SESION


P
INICIALIZACION MENU PRINCIPAL
P
SOLICITA CLASE DE REPORTE
P
SOLICITA SUBCLASE DE REPORTE
P
SOLICITA CRITERIO DE CONSULTA
P
REGRESO A PANTALLA PRINCIPAL
SOLIC. INF. FINANCIERA
MENU INICIALIZADO
CLASE ESPECIFICADA
SUBCLASE ESPECIFICADA
RESPUESTA DEL SISTEMA
SALIDA A PANTALLA ANTERIOR


P
REGRESA A PANTALLA DE INF.
CANCELA CONSULTA
P
INICIALIZACION DE VARS. CONSULT
SUBCLASE ESPECIFICADA
VARIABLES INICIALIZADAS
P
DESHABILITACION DE VARS. NO USA
P
PROCESO DE CONSULTA
DATOS PROCESADOS
P
PRESENTACION DE CUADRO CORRESP.
RESPUESTA DEL SISTEM
VARIABLES A UTILIZAR
DATOS ACEPTADOS
P
SOLICITUD DE DATOS DE CONSULTA
P
VALIDACION DE DATOS
CONSULTA A LA BASE
D
INFORMACION DE LA BASE
INFORMACION RECUPERADA
DATOS DE LA CONSULTA


D
INFORMACION DE LA BASE
CONSULTA A LA BASE
INFORMACION RECUPERADA
P
EJECUCION DE CONSULTA
P
ASIGNACION DE DATOS
P
MANEJO DE DATOS
DATOS ACEPTADOS
DATOS RECUPERADOS
DATOS ASIGNADOS
DATOS PROCESADOS


P
DESPLIEGA EN PANTALLA
SOLICITUD A PANTALLA
P
PRESENTA CUADRO CORRESPONDIENTE
DATOS PROCESADOS
SOLICITUD DE IMPRESION
P
IMPRESION DE CUADRO
RESPUESTA DEL SISTEMA
P
GRAFICACION DE CUADRO
SOLICITUD DE GRAFICA

Mini-Especificaciones.

Dentro de cada proceso incluído en el diagrama de flujo de datos se realizan diversas operaciones sobre la obtención de datos, su. manejo y los resultados que se obtienen. Todo ésto se observa de manera más clara en las miniespecificaciones correspondientes a los procesos del sistema las cuales se muestran a continuación.

```
Entrada al sistema (en pantalla principal)

Si se solicita finalizar sesión
     Regresa a Sistema Operativo
Sino
     Entra al Sistema Financiero
Endif;


Para toda entrada al sistema

Seleccionar clase de reporte ó regreso a pantalla
principal

Si se solicita reporte
     Ejecutar subrutina para definir ramos
     Desplegar clase de reporte
     Inicializar lista de subclase de reporte
Dependiendo de la clase de reporte seleccionado
     Despliega lista de subclase de reporte
Enddo;
     Seleccionar criterio de consulta
Sino
      Regresa a pantalla principal
Endif;

Para toda subclase especificada
     Inicialización de variables para consulta
     Selección para consultar ó regresar a
     pantalla anterior
Si se desea consultar cuadro
     Deshabilitar variables no utilizadas por tipo
     de reporte
     Cargar datos  (mes, año, oficina)
     Cargar Ramo
     Cargar subramo
     Validar datos (mes, año, oficina)
```

```
        Procesar datos
          Armar consulta
          Enviar consulta
        Presentar cuadro correspondiente
          Recupera datos de consulta
          Manipula datos
        Sino
          Regresa a pantalla anterior
        Endif;


      Para cualquier selección de cuadro
         Seleccionar presentación de cuadro
         Se despliega en pantalla
           Si se imprime cuadro
              Carga formato de salida
               Manda a impresora
         Endif;
         Si se desea graficar
             Envia conjunto de datos
             Graficar datos
                   Si se desea imprimir gráfica
                 Manda a impresora
                      Endif;
         Endif;
End.
```

VII. IMPLEMENTACION.

Requerimientos de Hardware y Software.

Para poder implementar este sistema el servidor debe cumplir con los siguientes requerimientos de Hardware:

- Que cuente con un procesador 80386 o superior
- Tenga como memoria mínima 8 Mb,
- Tener como mínimo 30 Mb en disco duro para instalación del software necesario,
- Contar con tarjeta para red capaz de soportar los protocolos TCP/IP o Netbeui.

Cada Estación de trabajo debe cumplir con los siguientes requerimientos de Hardware:

- Un procesador 80286 o superior,
- Un mínimo de memoria Ram de 4Mb,
- Un mínimo de 20 Mb de espacio disponible en disco duro para instalación del software necesario.
- Al igual que el Servidor debe contar con tarjeta para red capaz de soportar los protocolos TCP/IP o Netbeui.

Dentro del Software necesario el servidor debe contar con:

- Sistema operativo MS OS/2 Versión 1.2 o superior,
- Microsoft LAN Manager 1.2 o superior y
- SQL Server 2.0 o superior.

Dentro del Software necesario la estación de trabajo debe contar con:

- Sistema operativo MS-DOS 4.01 o superior,
- Microsoft LAN Manager 1.2
- SQL Server 2.0 o superior,
- Windows Ver 3.0 o superior,
- Visual Basic 1.0 (lenguaje de programación bajo ambiente Windows) o superior,
- Visual Basic Profesional Toolkits y
- Visual Basic Library para SQL Server (VBSQL).

Una vez que se cuenta con el HW y SW necesario se procede a la implantación del sistema que se dividirá en las siguientes fases :

DISEÑO DE LA BASE DE DATOS.

Este diseño se realizó en base a la información contenida en el diccionario de datos del diagrama de entidad-relación realizado en nuestra fase de análisis, tomando en cuenta la estructura necesaria para la definición de las tablas de nuestra base de datos en SQL SERVER éstas quedan definidas de la siguiente forma:

```
create table EdoRes
(Rubro          smallint        not null,
Rubro           smallint        not null,
Ramo            smallint        not null,
SRamo           smallint        not null,
Ofna            smallint        not null,
Anio            smallint        not null,
Mes             smallint        not null,
Monto           money           not null)

create unique index CveEdoRes
on EdoRes
(Anio,
Mes,
Rubro,
SRubro,
Ramo,
SRamo,
Ofna)

create table Presupuesto
(Anio           smallint        not null,
Mes             smallint        not null,
Ofna            smallint        not null,
Cta             smallint        not null,
SCta            smallint        not null,
SSCta           smallint        not null,
Ramo            smallint        not null,
SRamo           smallint        not null,
Monto           money           not null)
```

```
create unique index CvePresupuesto
on Presupuesto
(Anio,
Mes,
Cta,
SCta,
SSCta,
Ramo,
SRamo)


create table Cuenta
(Cta            smallint        not null,
SCta            smallint        not null,
SSCta           smallint        not null,
SRubro          smallint        not null,
Rubro           smallint        not null,
DescCta         char(40)        not null)

create unique index CveCuenta
on Cuenta
(Cta,
SCta,
SSCta)


create table Rubro
(Rubro          smallint        not null,
SRubro          smallint        not null,
DescRubro       char(56)        not null)

create unique index CveRubro
on Rubro
(Rubro,
SRubro)


create table Emision
(Ofna           smallint        not null,
Moneda          smallint        not null,
Ramo            smallint        not null,
SRamo           smallint        not null,
NoPoliza        int             not null,
NoEndoso        int             not null,
TipoEnd         char(1)         not null,
Asiento         char(1)         not null,
```

```
FechaEmi        char(4)         not null,
NoCliente       char(12)        not null,
IniVig          char(6)         not null,
FinVig          char(6)         not null,
MtoRecarg       money           not null,
MtoRedAut       money           not null,
MtoPrima        money           not null,
MtoComis        money           not null,
TipoCambio      money           not null)


create index CvePoliza
on Emision
(NoPoliza,
NoEndoso,
Moneda)

create table Siniestro
(Ofna           smallint        not null,
Ramo            smallint        not null,
SRamo           smallint        not null,
NoReclam        int             not null,
FechaReclam     char(6)         not null,
NoPoliza        int             not null,
Estado          smallint        not null,
Localidad       char(20)        not null,
Causa           char(3)         not null,
FechaMov        char(6)         not null,
TipoMov         smallint        not null,
ImpMov          money           not null,
Moneda          smallint        not null,
TipoCambio      money           not null,
NoCliente       char(40)        not null)


create index CveSin
on Siniestro
(NoReclam,
FechaMov)

create table Estado
(Estado         smallint        not null,
DescEdo         char(22)        not null)
```

```
create unique index CveEdo
on Estado
(Estado)

create table CausaSin
(Causa          smallint        not null,
DescCausa       char(50)        not null)

create unique index CveCausa
on CausaSin
(Causa)

create table Moneda
(Moneda         smallint        not null,
DescMon         char(20)        not null)

create unique index CveMoneda
on Moneda
(Moneda)

create table Asegurado
(NoCliente      char(12)        not null,
DescCliente     char(40)        not null)

create unique index Cve Asegurado
on Asegurado
(NoCliente)

create table Balanza
(Anio           smallint        not null,
Mes             smallint        not null,
Cta             smallint        not null,
SCta            smallint        not null,
SSCta           smallint        not null,
Ofna            smallint        not null,
Saldo           money           not null)
```

```
create index CveBal
on Balanza
(Anio,
Mes,
Cta,
SCta,
SSCta,
Ofna)


create table Oficina
(Ofna           smallint        not null,
DescOfna        char(30)        not null)


create unique index CveOfna
on oficina
(Ofna)


create table SRamo
(Ramo           smallint        not null,
SRamo           smallint        not null,
DescSRamo       char(35)        not null)

create unique index CveSRamo
on SRamo
(Ramo,
SRamo)
```

ELABORACION DE PANTALLAS Y MENUES.

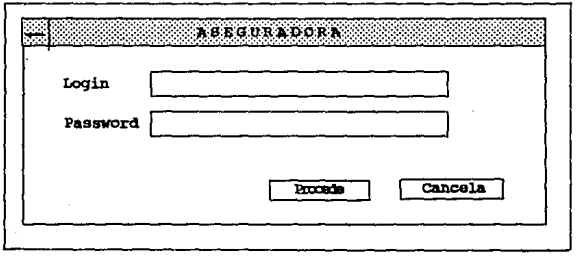
La elaboración de las diferentes pantallas y menúes se realizó dentro Visual Basic, que nos permite indicar y definir qué objetos va a contener dicha pantalla, indicar títulos representativos de la función de cada objeto, la ubicación de cada uno de ellos dentro de la pantalla, el color mismo de la pantalla y de algunos de los objetos que participan en ella, para posteriormente a cada objeto asociarle el código correspondiente de la rutina que se encargara de disparar la invocación de dicho objeto.

Las diferentes pantallas y menúes que componen el Sistema de Información Estratégica de la Aseguradora son las siguientes:

- Pantalla de seguridad de acceso al sistema
- Menú Principal
- Pantalla de selección de consultas
- Pantalla para elaboración de consulta y despliegue de resultados
- Pantalla para gráficas

A continuación se describirá a cada pantalla y su función en particular.

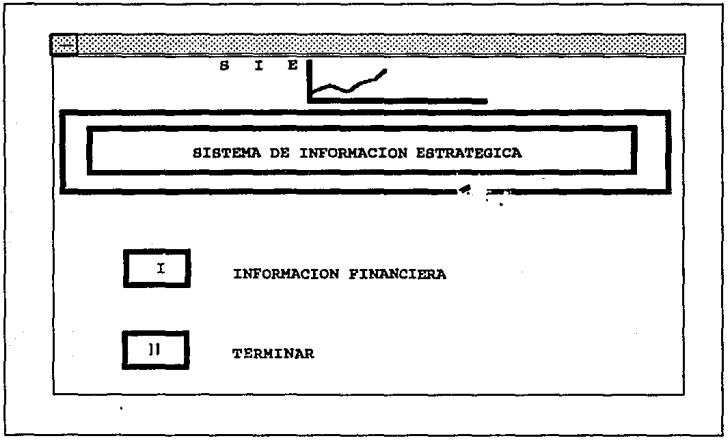
La primera pantalla (Fig. 7.1.) tendrá como fin dar cierta seguridad a la información contenida en la base de datos del sistema, ya que será una pantalla donde el usuario proporcione sus datos y estos al ser validados por el sistema den acceso unicamente a los usuarios autorizados.



7.1. Pantalla de acceso al Sistema.

En la segunda pantalla contamos con un menú de dos opciones, la primera de estas opciones nos llevará directamente a la tercera pantalla que es el menú de consultas permitidas por el sistema, la segunda opción

nos permitirá dar por terminada la sesión con el sistema (Fig. 7.2.).



7.2. Pantalla Principal del Sistema.

La tercera de nuestras pantallas aparecerá como ya se mencionó con anterioridad al dar un click en la primera opción de la segunda pantalla, ésta se presentará sobrepuesta a la segunda pantalla y corresponde al menú de consultas permitidas por el sistema, en éste seleccionaremos el tipo de consulta que deseamos realizar; existen cuatro tipos de información que podremos accesar que son los que se encuentran en la lista de la caja que se encuentra a la izquierda de esta pantalla, al seleccionar una de estas opciones en la caja de la derecha aparecerá la lista de las diferentes consultas que se pueden realizar con la opción seleccionada (Fig. 7.3.).

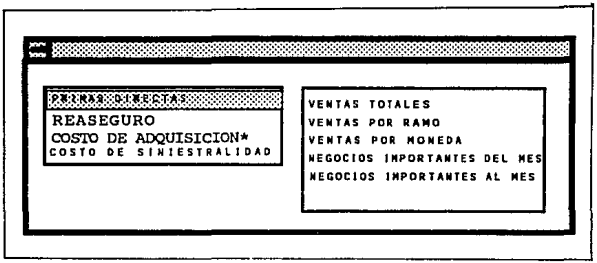


Fig.7.3. Pantalla de Selección del Cuadro correspondiente.

Al seleccionar el tipo de información que se desea consultar, se presentará la cuarta pantalla la cual está compuesta a su vez de dos diferentes pantallas cada una de éstas con una función distinta.

La primera de sus pantallas será usada para proporcionar a aquel que realiza la consulta el medio de comunicación por el cual se le especifiquen al sistema las restricciones con que debe cumplir la información a presentar, esta primera pantalla también nos permitirá solicitar una impresión del cuadro a generar, una vez proporcionadas todas las restricciones y al presionar el botón de PROCEDE esta pequeña pantalla desaparecerá y dejará libre la segunda pantalla en la cual se presentará el cuadro solicitado con su formato correspondiente.

Esta segunda pantalla además de presentar el cuadro correspondiente tendrá un ícono que nos permitirá desplegar una gráfica del cuadro obtenido ya sea en tipo de pie o de barras, el botón de QUERY nos presentará de nueva cuenta la pantalla pequeña permitiéndonos realizar otra consulta distinta pero del mismo tipo de cuadro de información seleccionado de la tercera pantalla del sistema, el botón de FIN nos

llevara a la tercera pantalla para dar la opción de solicitar algún otro tipo de información (Fig.7.4.).

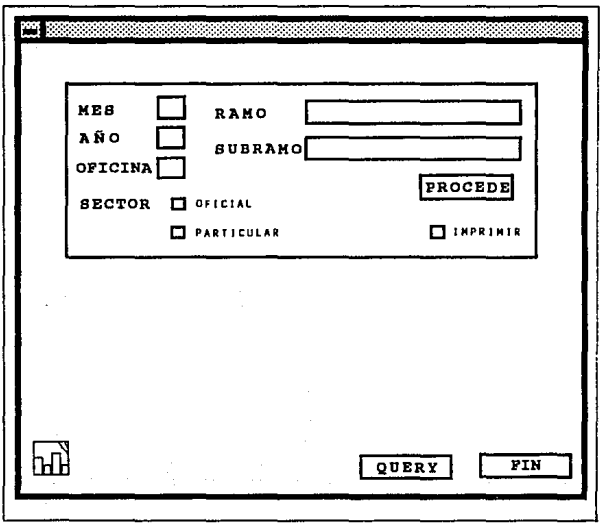


Fig. 7.4. Pantalla de Consulta y presentación del cuadro correspondiente.

Al dar un click del mouse sobre el ícono de gráficas nos llevará a una quinta pantalla la cual para los cuadros que cuenten con esta función nos desplegará una comparación en gráficas de pie de los datos correspondientes al año pasado y al presente año, si se

presiona el botón de **Barras** nos desplegará esta misma comparación pero en una gráfica de este tipo, con el botón de **Pie** se podrá regresar a la gráfica anterior.

El botón de **Impresión** nos permitirá obtener una impresión de dichas gráficas, el botón de **Fin** nos ocultará esta pantalla y nos llevará a la tercera pantalla del sistema (Fig. 7.5.).

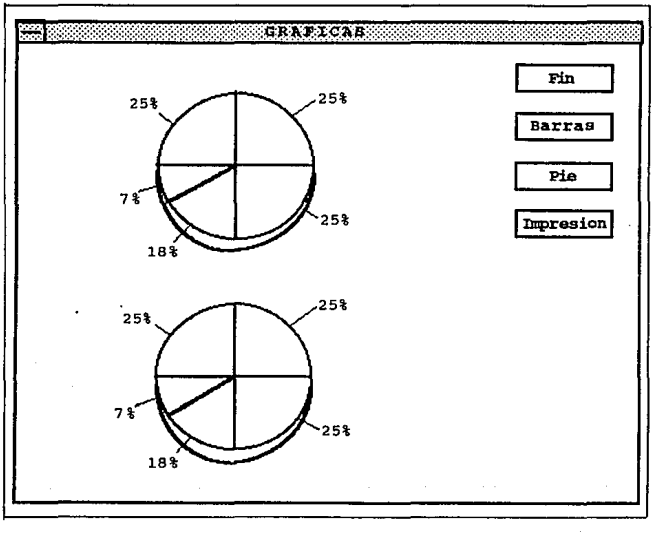


Fig.7.5. Pantalla de Gráficas.

IMPLEMENTACION

Para dar por terminada la sesión al momento de regresar a la tercera pantalla como ya se mencionó, ésta se encuentra sobrepuesta a la segunda, se dará un click del mouse dentro del área correspondiente a la segunda pantalla, automáticamente será cerrada la tercera y podremos seleccionar la segunda opción que nos permitirá terminar la sesión.

PROGRAMACION DEL SISTEMA.

.Por tratarse de un sistema dirigido al sector ejecutivo de la compañía, se decidió realizar la programación del sistema en un lenguaje que nos permitiera desarrollar aplicaciones que pudieran ejecutarse bajo ambiente windows que es hasta cierto punto elegante y fácil de comprender. El lenguaje seleccionado fue Visual Basic de Microsoft el cual permite desarrollar fácilmente este tipo de aplicaciones, asociando código a diversos objetos como son formas, botones, iconos, menúes, etc; dicho código se ejecuta al dar un click del mouse sobre el objeto seleccionado.

PRUEBAS.

Los objetivos de las actividades de pruebas son valorar y mejorar la calidad de los productos del trabajo generados durante el desarrollo y modificación del software. Los atributos de la calidad deben ser la corrección, la perfección, la consistencia, la confiabilidad, la utilidad, la eficacia, el apego a los estándares y la eficacia de los costos totales.

Para probar el funcionamiento del sistema dentro de las siguientes tablas que son las que mes a mes recibirán nuevos datos se cargo información referente a 1991 y 1992 hasta el mes de Junio:

TABLA	DESCRIPCION
EdoRes	Estado de Resultados
Emisión	Emisión de Polizas
Siniestro	Siniestros Ocurridos
Balanza	Balanza
Presupuesto	Presupuesto Asignado (Solo 1992)

También las siguientes tablas de la base fueron cargadas con la información que a cada una le corresponde, pero a diferencia de las anteriores ésta rara vez recibe nuevos datos.

TABLA	DESCRIPCION
Oficina	Catálogo de oficinas
CausaSin	Catálogo de Causas de Siniestros
Estado	Catálogo de Estados
Moneda	Catálogo de Monedas Existentes
Asegurado	Catálogo de Asegurados
Subramo	Catálogo de Ramos y Subramos
Cuenta	Catálogo de Cuentas
Rubro	Catálogo de Rubros

Se generaron por medio del sistema los cuadros de la carpeta del consejo de diversos meses del año de los años con que contaba la Base de Datos del Sistema de Información Estratégica, los cuadros obtenidos por este medio fueron comparados con los que el departamento de planeación había generado manualmente mediante captura de datos en hojas de cálculo, se detectaron algunos errores en el cálculo de porcentajes de algunos cuadros los cuales fueron ya corregidos, también se detectó información errónea dentro de la base de datos la cual se pidió fuese reprocesada.

Una vez corregidos estos errores se generaron nuevamente los cuadros del Sistema de Información y se compararon con los realizados por Planeación, los cuadros del Sistema y los de Planeación coincidían casi totalmente, las diferencias que se presentaban ahora

eran por redondeo de cantidades hecho en los cuadros de obtenidos por Planeación o por ajustes realizados a los montos contenidos en la base de datos, al verificar que estas diferencias no eran generadas por información incorrecta el sistema fue aceptado.

CONCLUSIONES

El sistema de información estratégica (SIE) esta orientado principalmente al personal ejecutivo de una compañia de seguros, por lo cual se tomó la decisión de desarrollarlo bajo un lenguaje que permitiera implementar aplicaciones que se ejecutaran bajo un ambiente amigable, para este caso, ambiente Windows, el cual es de fácil comprensión por personas ajenas al medio de sistemas.

El lenguaje seleccionado fue Visual Basic que nos permitió desarrollar dicho sistema con las características deseadas y no nos fue difícil de entender; todo ésto no quiere decir que fue el mejor lenguaje, pero si, cumplió con nuestros requerimientos.

Sobre la herramienta CASE seleccionada -Sysarch- podemos decir que ésta no cumplió en su totalidad, pero si satisfactoriamente con el fin de facilitarnos el análisis y el diseño del sistema, pues pensamos que en un futuro, cuando se desee modificar o actualizar el SIE, todo lo desarrollado bajo esta herramienta facilitará su mantenimiento y actualización.

En lo que se refiere a SQL Server, es importante mencionar que su rendimiento se puede incrementar si su plataforma se encontrase en un equipo de mejores características, ya que su velocidad de respuesta en el equipo Vectra utilizado como servidor disminuye notablemente al agregar a la red un mayor número de usuarios.

Finalmente, creemos que la experiencia adquirida en el desarrollo de este sistema, nos ha permitido aprender el uso, de lo que sentimos, son herramientas de actualidad, como son las herramientas CASE, los manejadores de bases de datos y el lenguaje de programación bajo ambiente Windows Visual Basic.

BIBLIOGRAFIA.

Martin James, Carma McClure, *Structured Techniques* . EU: Prentice Hall, 1988.

SYSTEM ARCHITECT, *User Guide* . EU: Popkin Software & Systems Incorporated, 1989.

SYSTEM ARCHITECT, *Tutorial* EU: Popkin Software & Systems Incorporated, 1989.

POSE, *Overview* . EU: Computer Systems Advisers, Inc., 1990.

DBMS. Magazine January 1991. pags. 44-82.

MOCROSOFT VISUAL BASIC, Programmer's Guide. EU: Microsoft, 1991.