11
2ej

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE CIENCIAS

APLICACION DE ANALISIS Y DISEÑO DE SISTEMAS MEDIANTE HERRAMIENTAS CASE

T E S I S

QUE PARA OBTENER EL TITULO DE:

M A T E M A T I C O

P R E S E N T A

JOSE ANTONIO HERNANDEZ AYUSO

MEXICO, D. F. 1993

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

TABLA DE CONTENIDO

INTRODUCCION i

CAPITULO 1. SISTEMAS DE INFORMACION.

1.1 Introducción 1
1.2 Generalidades de la Teoría de Sistemas 3
1.3 Ingeniería de Sistemas 5
1.4 Ciclo de Vida de un Sistema 8
1.4.1 Estudio de Factibilidad 8
1.4.2 Planeación 8
1.4.3 Desarrollo 11
1.4.4 Instalación 18
1.4.5 Mantenimiento 20
1.5 Problemática de los Sistemas de Información 22
Bibliografía del Capítulo 30

CAPITULO 2. METODOLOGIAS PARA EL DESARROLLO DE SISTEMAS.

2.1 Antecedentes 31
2.2 Clasificación de las Técnicas Estructuradas 34
2.3 Metodologías más Utilizadas 36
2.4 Análisis Estructurado de Tom De Marco 43
2.4.1 Objetivos 43
2.4.2 Componentes de la Especificación 44
2.4.3 Construcción de la Especificación 54
2.5 Diseño Estructurado de Yourdon 56
2.5.1 Objetivos 56
2.5.2 Metodología 57
2.5.3 Evaluación del Modelo 62
2.5.4 Implantación del Diseño 64
Bibliografía del Capítulo 65

INTRODUCCION.

Desde los inicios de la computación, se ha perseguido con afán el objetivo de construir sistemas confiables y eficientes. Algunos destacados investigadores han propuesto técnicas que permitan acercarse al cumplimiento de dicho objetivo, que en mayor o menor grado han sido seguidas por los grupos de trabajo.

Sin embargo los vicios y las malas costumbres que han enraizado entre los analistas y programadores, así como la poca participación de los usuarios involucrados, han obstaculizado el camino hacia la calidad. Como resultado de estos defectos, el desarrollo de sistemas ha avanzado mucho más lentamente que otra disciplina relacionada con él como es el desarrollo de equipo de cómputo.

Resulta innegable que el avance del Hardware ha sido impresionante, pasando en unas pocas décadas de equipos enormes instalados en organizaciones privilegiadas, a computadoras personales ubicadas en todas partes. Por el contrario, el proceso de creación de un sistema se enfrenta hoy casi a los mismos problemas que lo acosaban hace veinte o veinticinco años, por lo que todo esfuerzo encaminado a impulsar su desarrollo debe ser tomado en cuenta seriamente.

Las herramientas CASE representan una posibilidad atractiva para el mejoramiento de los sistemas, es posible que con ellas se logre por fin crear sistemas de alta calidad, pero para utilizarlas correctamente es preciso conocer su filosofía. Este trabajo tiene por objeto mostrar el alcance de este tipo de herramientas dentro de la Ingeniería de Sistemas.

El primer capítulo está dedicado a los Sistemas de Información, en él se analiza la situación actual del desarrollo de sistemas así como las características que diversos investigadores de la Ingeniería de Sistemas establecen como deseables en el ciclo de vida de un sistema común. Los problemas que se presentan a los equipos de desarrollo, los obligan a apartarse en mayor o menor grado del modelo teórico ideal, por lo que se establece la **necesidad** de contar con métodos y herramientas que los auxilien en su labor.

Algunos de los métodos surgidos con este enfoque, son discutidos en el segundo capítulo, mostrando sus características más relevantes y mencionando las principales herramientas de que hacen uso. Este capítulo constituye la **base teórica** del trabajo y es indispensable para comprender la aplicación de las herramientas CASE.

En el tercer capítulo se desarrollan los conceptos de la Tecnología CASE, tales como su origen histórico, sus categorías y sus componentes principales. Asimismo, se describe una herramienta CASE en particular (IEW/WS), que es utilizada posteriormente como ejemplo y con la cual es posible apreciar los **beneficios** que esta tecnología puede ofrecer al desarrollo de sistemas.

Finalmente, los conceptos incluidos en las primeras tres partes del trabajo, se llevan a la **práctica** en el capítulo cuatro. En él se ejemplifica el análisis y diseño de un sistema, utilizando algunos de los métodos descritos en el segundo capítulo y aprovechando las ventajas de la herramienta CASE descrita en el capítulo tres.

Capítulo 1

SISTEMAS DE INFORMACION

1.1 INTRODUCCION.

A lo largo de su historia, el hombre ha desarrollado para su supervivencia y comodidad, las más diversas actividades que han evolucionado desde la caza y recolección hasta los viajes espaciales y los trasplantes de órganos. Esta evolución ha sido posible gracias a que a través de sus observaciones e investigaciones, la humanidad ha ido acumulando toda la información que forma la base de su conocimiento.

En un principio, el conocimiento no era registrado en forma alguna y era transmitido de boca en boca, de un pueblo a otro y de generación en generación. De esta manera, la información acerca de los hábitos de los animales o de las características de las diferentes plantas era conservada por los hombres más experimentados quienes enseñaban a sus discípulos todo lo posible. Sin embargo, cuando las actividades comenzaron a diversificarse (agricultura, ganadería, comercio, artes, etc.) surgió la necesidad de registrar de alguna manera todos los datos que pudieran ser útiles posteriormente. Esta necesidad quedó satisfecha en gran parte gracias a la invención de la escritura por medio de la cual se crearon códices, libros, mapas y demás fuentes de consulta más perdurables y confiables que la simple tradición oral. De esta manera la información referente a un tema en particular estaba disponible en cualquier momento, si bien sólo era accesible para un grupo más o menos

reducido de personas.

En el presente siglo, la diversificación del quehacer humano ha llegado mucho más lejos y basta imaginarnos la cantidad de datos que son necesarios para administrar una universidad, operar un banco u organizar un campeonato de futbol, para comprender porqué el hombre ha buscado la mejor manera de almacenar y organizar su información utilizando archivos en papel, carpetas clasificadoras, microfichas, etc. En los últimos años, se ha incrementado notablemente la capacidad humana para el procesamiento de la información por lo que en la mayoría de la organizaciones han surgido departamentos especializados cuyo objetivo es el diseño, operación y/o administración de sistemas de información.

Una de las definiciones más aceptadas de lo que es un Sistema de Información es la que lo establece como un conjunto organizado de procesos que al ejecutarse proporcionan información para la toma de decisiones o el control de la organización.

De acuerdo con esto es posible mencionar un gran número de sistemas de información presentes en las organizaciones que nos rodean y que diariamente observamos trabajar, tal vez sin darnos cuenta de lo que representan realmente. Por ejemplo, en una tienda de tamaño mediano podemos observar de vez en cuando a los empleados haciendo conteos de mercancía en anaqueles, a los cajeros hacer reportes de las ventas y formas de pago e incluso a algún supervisor recibir quejas y sugerencias de los clientes. Toda esta información se concentra en un solo lugar, posiblemente la gerencia, en donde será utilizada para conocer la marcha de la empresa.

Al hacerse más complejas las organizaciones, la obtención de información para la toma de decisiones se hizo a su vez un proceso más elaborado que difícilmente puede satisfacerse mediante procesos manuales, por lo que los Sistemas de Información se trasladaron a un medio que cuenta con la capacidad para satisfacer las necesidades en cuanto a volumen de información almacenada y procesada, velocidad de respuesta, confiabilidad y versatilidad: la computadora. Este tipo de

Sistemas de Información computarizados se encuentran actualmente en todo tipo de organizaciones, por ejemplo: supermercados (inventarios, proveedores), universidades (historias académicas, bibliotecas), bancos (cajeros automáticos, manejo de cuentas).

Dada su creciente importancia y tomando en cuenta el hecho de que un sistema de información computarizado es por lo general más difícil de modificar que un sistema manual, es comprensible que haya surgido un área de investigación conocida como Ingeniería de Sistemas (también llamada Ingeniería de Software) que dedica sus esfuerzos a la correcta especificación y diseño de los sistemas de información computarizados.

En las siguientes secciones se discutirá el origen y características más relevantes de la Ingeniería de Sistemas.

1.2 GENERALIDADES DE LA TEORIA DE SISTEMAS.

La Ingeniería de Sistemas tiene sus orígenes en la Teoría General de Sistemas la cual define un sistema como un conjunto organizado, interactivo, interdependiente e integrado de componentes con una meta común, y subraya la necesidad de examinar todas sus partes. Dentro de esta definición debe destacarse la idea de que las componentes del conjunto están interrelacionadas, ya que ésta es una de las bases de la Teoría General de Sistemas al señalar que el estudio de las partes aisladas no equivale a la comprensión del sistema integral, puesto que éste debe ser visto como un todo y no como la simple agregación de elementos. Esta afirmación se contrapone a la clásica visión analítica de los eventos pues sustenta que algunas propiedades de los sistemas, las llamadas constitutivas no son observadas en el estudio individual de las componentes que se limita a tratar las características sumativas, es decir, aquellas que no se alteran dentro o fuera del sistema.

Una de las consecuencias de la idea anterior es el hecho de que la Teoría General de Sistemas reconoce a éstos como escenarios en donde intervienen diferentes ramas del saber humano, por lo que pone énfasis en los aspectos de comunicación entre los especialistas de las diversas ciencias que se ven involucrados en un sistema. Para apreciar la veracidad de este punto, considérese una organización tan sencilla en apariencia como un restaurante en donde pueden intervenir expertos en las áreas de nutrición, psicología de ventas, cocina, relaciones humanas, arquitectura, etc., es fácil imaginar la forma en que caminaría un negocio de este tipo en el que no existiera comunicación entre las partes.

Otro punto destacado en la caracterización de un sistema por parte de la Teoría General de Sistemas es la existencia de un objetivo o meta a lograr, los cuales no siempre son identificables con facilidad. Las diferentes partes de un sistema interactúan con miras a lograr el objetivo general de la mejor manera posible.

En este contexto, el medio ambiente y los recursos son elementos adicionales para los sistemas. El medio ambiente no es controlado por el sistema, sino que por el contrario lo limita y define su comportamiento bajo diferentes situaciones. Los recursos son los medios de que se valen los sistemas para realizar las actividades necesarias para el logro de sus objetivos.

La administración de un sistema incluye actividades de planeación y control. La planeación consiste en el establecimiento de metas, elaboración de estrategias para el uso de recursos y para enfrentar el medio ambiente. El control se refiere a la ejecución de los planes, el flujo de la información y el autocontrol del sistema.

Para ejemplificar estos puntos considérese un equipo de voleibol desde el punto de vista de la Teoría General de Sistemas. En este *sistema* desde luego que el objetivo es vencer al equipo rival aunque pueden existir metas secundarias como agradar al público o clasificar en un lugar determinado en un campeonato; el medio ambiente está integrado por el equipo contrario, el árbitro, la condición de la cancha, etc.; los re-

cursos son las diferentes habilidades de los jugadores para los diferentes puestos; la planeación es realizada por el entrenador que decide cuáles jugadores utilizará y la táctica a emplear; el control puede ejercerse a través del capitán del equipo que debe motivar a sus compañeros así como transmitir las órdenes del entrenador. También el punto inicial referente a analizar el sistema como un todo y no únicamente como la unión de sus partes queda ejemplificado aquí pues la calidad de un equipo de pelota no es función exclusiva de la calidad de sus jugadores sino también de la capacidad que demuestren para interactuar unos con otros.

Finalmente debe mencionarse que para los investigadores de la Teoría General de Sistemas estos están ligados entre sí, a través de sus entradas y salidas formando sistemas más complejos. Es por esto que los datos que alimentan un sistema así como las respuestas que se esperan de él deben estar definidas claramente. Utilizando el razonamiento inverso, frecuentemente es posible ver un sistema como una comunidad de sistemas menores que al interactuar van estableciendo las bases del sistema estudiado.

1.3 INGENIERIA DE SISTEMAS.

Como ya se mencionó, la Ingeniería de Sistemas nace como una respuesta a la necesidad de crear sistemas de una manera ordenada, que minimice los problemas producidos por las prácticas viciadas de los diferentes integrantes de un equipo de desarrollo. Se trata de una disciplina que reúne conceptos de varias ciencias como son Computación, Matemáticas, Psicología y Administración, entre otras, y cuya base está constituida por una serie de técnicas empleadas para resolver o implementar la solución de problemas de manera independiente a la aplicación de que se trate.

Esta independencia de la aplicación permite que la Ingeniería de Sistemas sea utilizada para el desarrollo de sistemas con di-

ferentes características como pueden ser: la Programación de Sistemas (compiladores, editores); sistemas empresariales (cuentas por cobrar, costos unitarios); sistemas científicos con algoritmos complejos o análisis numérico profundo y otro tipo de sistemas como juegos o reconocimiento de caracteres.

Una premisa importante de este enfoque consiste en reconocer que los grandes sistemas de información no son simples extrapolaciones de los pequeños sistemas, sino que al aumentar el tamaño, los problemas y los recursos, el control debe ser más estricto para obtener resultados satisfactorios. La Ingeniería de Sistemas proporciona los elementos para establecer ese control. Para ello considera que los sistemas cumplen un ciclo de vida que abarca el tiempo durante el cual se desarrollan así como su período de utilidad.

Diferentes autores dividen el ciclo de vida de los sistemas en diferentes etapas y subetapas, haciendo énfasis en algunas de ellas, sin embargo todos tienden a un modelo similar al siguiente:

CICLO DE VIDA DE UN SISTEMA.

Estudio de factibilidad.

Planeación.

- Planeación del sistema.
- Análisis y definición de requerimientos.

Desarrollo.

- Diseño preliminar del sistema.
- Diseño detallado del sistema.
- Programación.
- Pruebas modulares del sistema.
- Pruebas integradas del sistema.

Instalación.

- Instalación.

Capacitación.

Operación.

Mantenimiento y mejoras

Adicionalmente debe reconocerse que el desarrollo de un sistema no se limita a la producción de los programas que van a operarse, sino que debe considerarse también la documentación para instalar, utilizar y mantener los programas. Siguiendo el modelo del ciclo de vida descrito anteriormente, la documentación deseable estaría integrada por:

Estudio de factibilidad.

Plan del sistema.

Especificación de requerimientos.

Diseño preliminar.

Diseño detallado.

Documentación de programas.

Plan de pruebas.

Documentación para capacitación.

Manual del usuario.

Plan de reportes para mantenimiento.

Tanto las fases del ciclo de vida como los documentos generados en ellas son aspectos relevantes para el contenido de este trabajo por lo que se describen más detalladamente a continuación.

1.4 CICLO DE VIDA DE UN SISTEMA.

1) Estudio de factibilidad.

En esta primera fase se estudian las posibilidades económicas y técnicas para desarrollar un sistema. En el caso de un sistema manual que desea automatizarse, se revisan los procesos existentes para proponer sistemas alternativos. Un estudio costo-beneficio ayuda en la selección de una de las alternativas. Los resultados se reflejan en un documento llamado Estudio de Factibilidad.

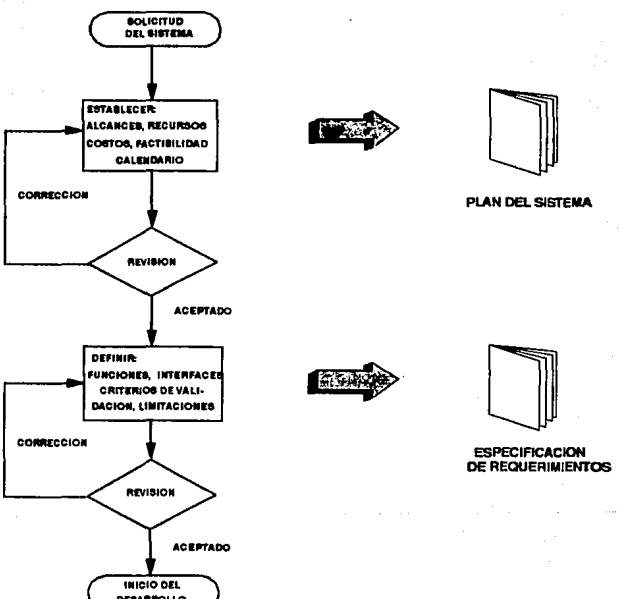
2) Planeación.

Generalmente esta fase se divide en dos subetapas conocidas como planeación y análisis, cada una de las cuales produce un documento de importancia. Esta situación se ilustra en el diagrama de la siguiente página.

2.a) Planeación del sistema. El objetivo principal de esta temprana etapa consiste en identificar plenamente el sistema, es decir, a partir de una solicitud generalmente informal por parte del usuario el analista debe establecer cuál es exactamente el problema que tiene que atacarse. Esta operación no siempre es fácil ya que frecuentemente las fronteras de acción e influencia de un sistema no están definidas con claridad y se sobreponen a las de otros sistemas.

Una vez identificado el problema, se acordarán junto con el usuario los alcances del nuevo sistema, definiendo las principales funciones que se incluirán y las cuestiones que se mantendrán fuera del sistema. En una organización compleja este punto reviste gran importancia, pues sería un error intentar elaborar un sistema que abarcara todos los aspectos susceptibles de ser sistematizados. Lo más práctico es reducir el campo de acción de los sistemas, de manera que cubran una porción coherente de las actividades de la empresa y dejen lo demás bajo la responsabilidad de sistemas complementarios.

FASE DE PLANEACION



Aunque es altamente recomendable la participación de los usuarios en todas las etapas del desarrollo del sistema, es en las fases de planeación donde la comunicación entre los usuarios y los analistas debe acentuarse al máximo con el objeto de que ambos se aseguren que están siendo comprendidos perfectamente. Un error u omisión en el sistema producido por una falta de comunicación en las primeras etapas, puede ser descubierto hasta que el sistema está ya muy avanzado, acarreando costo y trabajo extra que serán mayores mientras más tiempo pase antes de su detección.

El documento producido durante la planeación del sistema (Plan del sistema) debe incluir además observaciones acerca de los recursos que serán necesarios para producirlo (recursos humanos, de software y hardware), una estimación del costo implicado en su elaboración y una propuesta de calendarización de las entregas. Obviamente los recursos, costos y calendarización deben ser realistas y ajustarse a los límites determinados en el estudio de factibilidad.

2b) Análisis y definición de requerimientos. En esta etapa de la fase de planeación se persiguen los siguientes objetivos principales:

- Proporcionar una base para el desarrollo del sistema estableciendo el flujo y la estructura de la información.
- Describir las interfaces o funciones.
- Describir los límites para el sistema.
- Establecer y sostener la comunicación entre el usuario y el analista.

Dentro del primer objetivo, es útil incluir un modelo conceptual representado mediante alguna notación gráfica en el que se observe el comportamiento externo del sistema, así como el flujo general de los datos. De igual forma es conveniente definir las características de la base de datos del sistema, para especificar claramente la información que se requerirá a lo largo del proceso.

La descripción de interfaces consiste en establecer las entradas y salidas del sistema, contemplando todos aquellos reportes que deberán provenir o alimentar a otros sistemas. Las funciones deben describirse de manera general con el propósito de dar a conocer concretamente qué es lo que se espera que el sistema proporcione al usuario.

Las limitantes del sistema -también llamadas requisitos no funcionales- se incluyen en esta etapa con el fin de confirmar las apreciaciones realizadas en la etapa de planeación.

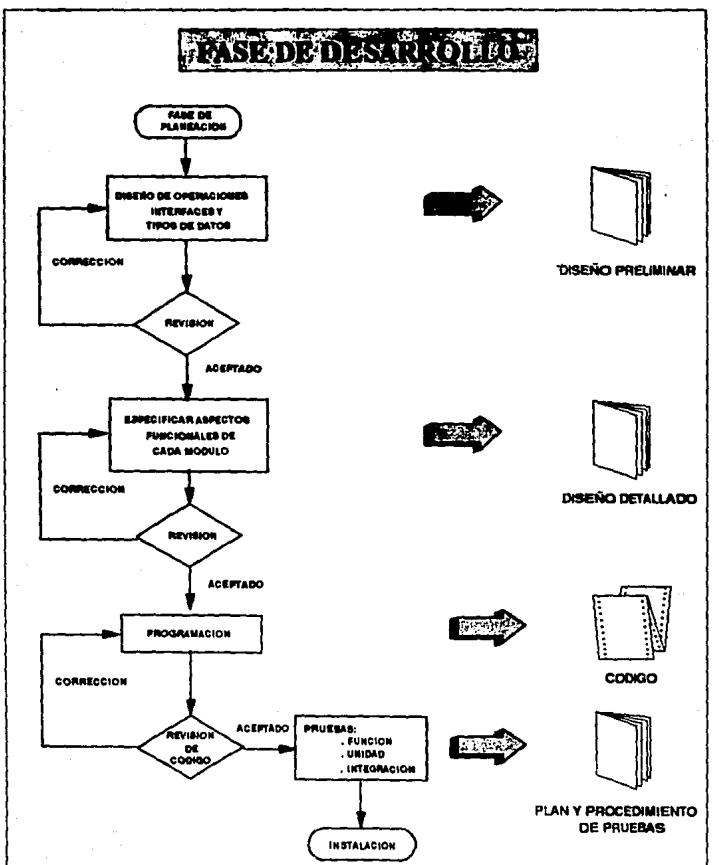
El documento de los requerimientos del sistema no es un documento de diseño. en el sentido de que establece lo que ha de hacer el sistema y no cómo ha de hacerlo. Se presenta al usuario para que lo discuta con el analista y confirme la visión que éste tiene del sistema o bien manifieste su desacuerdo. Como esta última situación se presenta con más frecuencia. dicho documento debe ser fácil de modificar, ya que el análisis de requerimientos es por naturaleza una fase en la que el estrecho contacto con el usuario obliga a revisiones sucesivas de su contenido, lo que no es un inconveniente sino todo lo contrario pues todas las diferencias de puntos de vista que se resuelvan aquí evitarán contratiempos en etapas posteriores.

3) Desarrollo.

El desarrollo suele dividirse en diseño preliminar. diseño detallado. programación y prueba del sistema. de acuerdo a la secuencia mostrada en la figura de la siguiente página.

3.a) Diseño preliminar del sistema. En esta parte del ciclo de vida se efectúa un diseño de alto nivel del sistema enfocándose en los siguientes puntos:

- Especificación de interfaces internas. Se determinan las conexiones entre diferentes procesos dentro del sistema sin hacer énfasis en la manera en que cada proceso transforma los datos de entrada en los de salida.


FASE DE DESARROLLO
FASE DE PLANEACION
DISEÑO DE OPERACIONES INTERFACES Y TIPOS DE DATOS
CORRECCION
REVISION
ACEPTADO
DISEÑO PRELIMINAR
ESPECIFICAR ASPECTOS FUNCIONALES DE CADA MODULO
CORRECCION
REVISION
ACEPTADO
DISEÑO DETALLADO
PROGRAMACION
CORRECCION
REVISION DE CODIGO
ACEPTADO
PRUEBAS:
. FUNCION
. UNIDAD
. INTEGRACION
CODIGO
PLAN Y PROCEDIMIENTO DE PRUEBAS
INSTALACION

- Especificaciones operacionales. Definen las transformaciones que ocurren en cada proceso, por medio de las cuales se convierten los datos de entrada en los de salida requeridos.

- Especificación de tipos de datos abstractos. La información detallada de los datos se oculta mediante el empleo de tipos de datos abstractos. con el objeto de centrar la atención en los aspectos de más alto nivel.

3.b) Diseño detallado del sistema. En este punto se procede a una descripción detallada de todos los procedimientos del sistema. El diseño final se obtendrá tras un proceso repetitivo en el que se van revisando las diferentes versiones. hasta llegar a una que cumpla satisfactoriamente con la Especificación de Requerimientos y a un nivel de detalle lo suficientemente bajo como para pasar de manera natural a la programación.

Por lo general un diseño detallado sigue las etapas que se mencionan a continuación:

a) Establecer los subsistemas en que se divide el proyecto de una manera lógica.
b) Dividir cada subsistema en componentes individuales.
c) Diseñar los programas o procesos en base a componentes relacionadas.
d) Refinar los componentes hasta el detalle de los algoritmos.

El diseño no se preocupa exclusivamente de los programas sino que debe contemplar también otros aspectos como son: los canales de comunicación entre las partes del sistema. los archivos y la estructura de datos y las pruebas a efectuar para su comprobación.

La comprobación del diseño debe centrarse en dos interrogaciones principales que son: saber si se resolvió el problema adecuadamente. es decir. si se cumplió con todos los requerimientos dentro de

las cotas factibles y saber si se solucionó el problema correcto, o sea, si las necesidades del usuario quedan realmente satisfechas con el sistema a desarrollar.

Para facilitar la labor del diseñador, y con el objeto de que logre el mejor diseño posible, han surgido una serie de metodologías de entre las cuales pueden destacarse las siguientes:

Diseño Funcional Descendente. En él se comienza por diseñar el funcionamiento en un alto nivel y se avanza progresivamente en la especificación hasta llegar a un diseño lo suficientemente detallado. Esta es la metodología más utilizada actualmente y está representada por las teorías de Diseño Estructurado de Yourdon y la de Refinamiento por Pasos de Wirth.

Diseño Orientado a Objetos. El sistema se ve como un conjunto de objetos con su propio conjunto de operaciones asociadas (Booch 1983).

Diseño controlado por los datos. Plantea que la estructura del sistema debe reflejar la estructura de los datos que procesa y que por tanto puede deducirse del análisis de entradas y salidas (Jackson 1975, Warnier 1977).

No es posible señalar categóricamente a una metodología como la mejor de todas pues al parecer algunas se adaptan mejor que otras a ciertas necesidades o a ciertos aspectos del software. La opinión generalizada es que una combinación de metodologías durante el diseño de un sistema puede resultar provechosa al utilizar los aspectos ventajosos de cada una.

Dadas las características de la actividad de diseño, resulta difícil si no imposible determinar si el diseño de un sistema es el mejor o más adecuado, ya que ello depende de los recursos con que se cuenta, de los objetivos prioritarios y de la naturaleza del software. Sin embargo puede decirse que un *buen* diseño es aquel que facilita el mantenimiento

posterior del sistema, por ejemplo mediante la definición de unidades coherentes y poco acopladas. Entendemos que las unidades son coherentes si todos sus elementos son relevantes dentro del contexto de la unidad y poco acopladas si no son fuertemente dependientes unas de otras de modo que al modificar una se afecten sólo mínimamente las demás.

3.c) Programación. Dado que la Ingeniería de Sistemas tiene como objetivo computarizar un sistema de información, la programación puede verse como su meta final, no porque aquí termine su trabajo, sino porque es en este momento cuando el sistema estudiado se convierte en un sistema computarizado.

Cuando se han seguido adecuadamente los pasos preliminares del ciclo de vida del sistema, la programación debe surgir de manera natural a partir del diseño detallado, sin embargo, esto no significa que sea una actividad sencilla, por el contrario, un buen programador se forma únicamente con la práctica ya que los programas no sólo deben *funcionar* sino que deben hacerlo de manera correcta y eficiente. Además, como ya se dijo, los programas del sistema no representan el fin de las actividades del equipo de desarrollo, éstas continuarán al ser necesario el mantenimiento del sistema, por ello los programas deben realizarse con un estilo que garantice su legibilidad, pues es casi seguro que en el futuro surja la necesidad de revisar el código.

Algunas características que pueden ayudar en la realización de programas más legibles son las siguientes:

- Utilizar nombres de datos y subprogramas significativos que tengan relación con lo que representan.
- Escribir programas modulares y reutilizar código siempre que sea posible por medio de subrutinas y funciones.
- Estructurar adecuadamente los ciclos evitando saltos bruscos en el flujo del programa.
- Observar los usos comunes al escribir código como sangrado de

proposiciones y utilización de letras mayúsculas y minúsculas.

- Agregar comentarios siempre que se crea oportuno, sobre todo en aquellas porciones oscuras del programa.

La utilización de todas ellas conducirá a programas más fáciles de entender.

Existen básicamente dos métodos para efectuar la programación del sistema, el ascendente y el descendente. Como su nombre lo indica el ascendente comienza por la programación de las funciones de nivel más bajo, para combinarlas a medida que se van desarrollando los niveles superiores. El descendente desarrolla primero los niveles superiores sin incluir los detalles de los procesos de bajo nivel hasta que se ha avanzado hacia abajo lo suficiente. Como en otros casos, no hay forma de calificar alguno de estos enfoques como superior, debido a que en diferentes circunstancias puede preferirse el uso de uno u otro.

El código fuente del sistema debidamente comentado se añade a los demás informes para integrar la documentación.

Finalmente debe considerarse que la programación está limitada desde luego por las características del lenguaje de programación, por ello al seleccionar el lenguaje o los lenguajes que se emplearán dentro de una organización, deben cuidarse aspectos tales como: la facilidad de traducir el diseño al código, la eficiencia del compilador, la existencia de herramientas de desarrollo y la facilidad de mantenimiento, teniendo en cuenta que los sistemas que se desarrollen en el futuro se basarán muy probablemente en esos lenguajes.

3.d) Pruebas y Depuración. Las pruebas del sistema son el proceso mediante el cual se ejecutan los programas con datos similares a los reales para tratar de encontrar errores o insuficiencias. La depuración consiste en la localización del error y posterior corrección del código.

De las definiciones anteriores podemos deducir que el objetivo de las pruebas es hacer fallar al sistema, que una buena prueba es aquella que tiene una alta probabilidad de descubrir un error y que una prueba tiene éxito si localiza un error. Cuando las pruebas fallan no queda demostrado que el sistema ya no contenga errores, pues es posible que estos existan sin haber sido descubiertos. Es por esto que las pruebas deben ser diseñadas adecuadamente.

Para realizar una prueba se siguen comúnmente los siguientes pasos:

- Se tiene a mano la Especificación de Requerimientos.
- Se diseña y efectúa la prueba.
- Se comparan los resultados obtenidos contra los esperados.
- Se realizan las correcciones necesarias a la vez que se va estableciendo el grado de confiabilidad del software.

Una práctica recomendable para el diseño de una prueba es la de partir los datos de entrada en clases de equivalencia intentando abarcar todos los casos posibles, o al menos un gran número de ellos. Por ejemplo, si la entrada es un dato booleano incluir valores verdaderos y falsos; si el programa espera un valor en un rango determinado, proporcionarle valores dentro y fuera del rango (menores y mayores); etc. Al diseñar una prueba de esta manera realmente se está intentando viajar por las diferentes ramas establecidas por las proposiciones de control de flujo del programa (if, while, for). Es necesario hacer un esfuerzo por recorrer todas las ramas con el fin de detectar tantos errores como sea posible, comúnmente los errores hallados de esta manera son errores lógicos o tipográficos.

Otro método adecuado para el diseño de pruebas consiste en el análisis de los valores límite de los datos de entrada. Ya que la experiencia demuestra que el número de errores presentes en un programa es mayor cuando los datos se encuentran cerca de sus valores límite, pueden

proporcionarse entradas cercanas a los límites permitidos y analizar el comportamiento del sistema. De esta manera se verifica principalmente la integridad externa del programa, localizando errores de inicialización o terminación, funciones incorrectas y errores en estructura de datos. Tanto el análisis de valores límite como la partición en categorías pueden aplicarse también a las salidas para el diseño de pruebas exitosas.

Las pruebas del sistema deben hacerse por etapas:

- Pruebas de funciones.
- Pruebas de módulos.
- Pruebas de integración.

La integración se realiza gradualmente agregando una sola función o módulo a la vez ya que de esta manera se facilita la ubicación de los errores. Al igual que en la programación, existen enfoques ascendentes y descendentes para efectuar pruebas de integración.

Esta fase de desarrollo del sistema debe quedar documentada en una Especificación de Pruebas en la que se incluyan puntos como los siguientes:

- Alcance de la prueba (límites y terminación).
- Plan de la prueba (diseño, estrategia general de integración).
- Procedimiento (Descripción detallada de los procedimientos requeridos para llevar a cabo el plan).
- Resultados esperados.
- Resultados de la prueba.

4) Instalación.

Comúnmente, la fase de instalación se divide en tres subetapas que tienen como objetivo entregar el sistema a la persona o grupo de personas que estarán encargadas de operarlo.

4.a) Instalación. Cuando. en opinión del equipo de desarrollo, el sistema está listo, es el momento de transferirlo de las máquinas donde fue desarrollado a aquellas en donde operará regularmente. Se copian los programas. se crean los archivos necesarios y, de ser necesario, se instalan los periféricos adicionales.

Es importante volver a probar el sistema una vez instalado en las computadoras definitivas, para cerciorarse de que las características del hardware y software son las indicadas y que los programas se comportan según lo previsto.

4.b) Capacitación. La importancia de la capacitación salta a la vista si pensamos en que. por lo general, un nuevo sistema computarizado se crea para sustituir procesos obsoletos o inadecuados que en muchas ocasiones incluyen una serie de procesos manuales. Por lo tanto frecuentemente la llegada del nuevo sistema implica la realización de tareas que los usuarios no están acostumbrados a realizar.

En los casos más drásticos, algunos usuarios no conocerán siquiera las bases para utilizar una computadora. por lo que se deberá comenzar la capacitación desde ese nivel de detalle. En cualquier caso una buena capacitación es escencial para el funcionamiento del sistema y ésta debe abarcar conceptos tales como la manera y el formato en que se proporcionan datos al programa. el orden de ejecución de los procesos. el flujo de información entre los módulos del sistema, etc. Adicionalmente si el sistema contempla el empleo de algún dispositivo periférico nuevo para el usuario. por ejemplo una nueva impresora o una unidad de respaldo. la capacitación debe ampliarse para cubrir los puntos referentes al uso de ese nuevo dispositivo.

Es aconsejable que no sea una sola persona la que se capacite en el empleo del sistema. sino que se instruya a un número mayor pues de esta forma se evita la dependencia por parte del sistema hacia un único usuario capacitado. cuya ausencia por cualquier situación, obligaría al equipo de desarrollo a regresar a esta etapa del ciclo de vida del sistema.

Para facilitar el aprendizaje, pueden entregarse al usuario a manera de documentación ejemplos del funcionamiento del sistema en los que se reflejen los diferentes aspectos operativos, ilustrando las entradas que se esperan y las salidas que se obtendrán.

4.c) Operación. Se conoce como fase de operación a aquella durante la cual el sistema es utilizado en forma regular para producir resultados. Esta fase tiene una duración variable que puede ir desde unos pocos días en sistemas desarrollados para proyectos muy específicos, hasta varios años en sistemas de gran estabilidad.

En este período un documento que desempeña un papel principal es el Manual del Usuario, el cual especifica detalladamente los pasos que debe seguir un operador para conseguir que cada proceso del sistema funcione adecuadamente. Por ejemplo en un sistema operado a través de menús, el manual del usuario dirá como elegir una opción del menú, que objetivo tiene cada opción, que tipo de entradas se esperan, que salidas se obtendrán, etc.

Una característica relevante de la operación es el hecho de que los usuarios suelen sugerir mejoras o solicitar modificaciones al sistema conforme van utilizando los diferentes módulos y observan los aspectos de funcionamiento de éstos. Estas sugerencias y solicitudes se evalúan y en caso de proceder dan origen a lo que se conoce como Mantenimiento del Sistema.

5) Mantenimiento. El mantenimiento de un sistema es la actividad mediante la cual se incorporan al proyecto original correcciones o mejoras solicitadas por los usuarios o descubiertas tardíamente. Los recursos consumidos por las diferentes clases de mantenimiento son cuantiosos, a tal grado que algunos autores estiman que podrían representar hasta un 60% de los costos totales de la vida del sistema.

Dependiendo de su origen, el mantenimiento se clasifica en tres tipos:

Correctivo. Es el que corrige errores que permanecieron ocultos durante las fases de prueba y que aparecen al operar el sistema, lo que hace que en gran número de ocasiones sea de naturaleza urgente. Aunque no es posible hacerlo desaparecer por completo, este tipo de mantenimiento logra reducirse sustancialmente mediante el empleo de la Ingeniería de Sistemas.

Adaptativo. Por ser el medio ambiente de los sistemas un entorno dinámico, a menudo ocurre que se presenten nuevas necesidades o restricciones que fuercen a modificaciones de diversos tipos. Este tipo de modificaciones suelen ser visibles con suficiente tiempo para ser efectuadas antes de que el sistema falle.

Estructurado. Es el mantenimiento que se realiza con el fin de incorporar mejoras al sistema. Es común que estas mejoras sean solicitadas por los usuarios y que se agrupen para ser manejadas en bloque dando lugar a versiones sucesivas del sistema.

El mantenimiento debe ser documentado mediante reportes en los que se indique el tipo de mantenimiento efectuado, la causa u origen de la modificación y las acciones implementadas para llevarla a cabo, pero sobre todo, el mantenimiento debe reflejarse en la documentación previa, modificando el diseño del sistema, los listados de programas, el manual de usuario y en fin todos aquellos papeles que resulten afectados de manera que siempre representen el estado actual del sistema.

A pesar de que suele ser descrita como la última actividad del ciclo de vida de un sistema, en realidad el mantenimiento es sólo un estado intermedio que puede dar lugar a nuevo análisis, rediseño de algunos componentes, nueva programación o reprogramación y nuevas pruebas, conformando un camino cíclico y de ninguna manera rectilíneo.

1.5 PROBLEMATICA DE LOS SISTEMAS DE INFORMACION.

Hasta ahora se ha descrito el camino seguido por un sistema de información cuando se desarrolla con un enfoque de Ingeniería de Sistemas. Sin embargo, en muchos casos la práctica no concuerda con la teoría y los sistemas se desarrollan sin seguir metodología alguna, lo que se traduce en la omisión o reducción de una o varias fases del ciclo de vida y en la construcción de productos deficientes que arrastran una serie de graves problemas.

En esta sección se describirán algunos de los problemas que se observan con mayor frecuencia, tratándolos de acuerdo al orden aproximado en el que aparecen en los sistemas de una organización.

Planeación y Estimación Inadecuadas. Uno de los mayores errores que se tienen en el área de sistemas es el de querer complacer las expectativas de los usuarios *importantes* sin estudiar si se está en posibilidades o no de cumplir con ellas. En este punto hay responsabilidad por ambas partes, pues si bien es cierto que son muy contadas las organizaciones que pueden permitirse el lujo de crear un sistema con tiempo y recursos ilimitados, también es verdad que muchas veces los directivos imponen al desarrollo de sistemas barreras artificiales con el fin de reducir sus costos puesto que es difícil apreciar la utilidad de un sistema que no producirá beneficios a corto plazo. Por otro lado, el equipo encargado de desarrollar un sistema, con tal de no ver cancelado su proyecto, se compromete a entregar resultados en un tiempo demasiado breve y con un consumo de recursos menor del que en realidad necesita.

El resultado de este *optimismo* es que el sistema no avanza a la velocidad planeada y los días transcurren sin que el calendario de entregas se cumpla. Entonces, se reconoce el error y se hacen nuevas estimaciones, o bien, se decide incrementar el número de personas que intervienen en el proyecto, en un esfuerzo desesperado por tratar de

regularizar el sistema, lo cual casi nunca se logra puesto que el incrementar el tamaño del equipo acarrea otros problemas -principalmente de comunicación- que terminan por obstaculizar aún más la marcha del proyecto.

En otras ocasiones, la calendarización del avance del sistema y la estimación de recursos son erróneas debido a una inadecuada comprensión del problema y no a la existencia de limitaciones por parte de los usuarios. De cualquier modo el problema es similar al anterior y las consecuencias igual de lamentables.

La experiencia aconseja que durante la estimación de tiempo y recursos que serán necesarios para el desarrollo de un sistema, se utilice una visión pesimista de la situación, pues no solamente es difícil que todos los procesos funcionen perfectamente al primer intento, sino que, dependiendo de la complejidad del sistema, casi puede asegurarse que en algún momento el desarrollo se detendrá por obstáculos imprevistos.

Análisis Insuficiente. Es común que se solicite a un departamento de informática el desarrollo de un sistema urgente, es decir un sistema que ya debería de estar trabajando. En estos casos el camino más socorrido es el de abreviar la etapa de análisis y pasar lo antes posible al desarrollo. Esta situación desde luego tiene sus inconvenientes, pues parte de la suposición de que el sistema puede comenzar a desarrollarse a partir de una definición más o menos general del problema.

Tarde o temprano el o los diseñadores se dan cuenta de que esta suposición es falsa, ya que aún hay aspectos nebulosos del sistema pendientes de definir. El desarrollo se detiene, se regresa a la etapa de análisis y se intenta de nuevo.

En otras ocasiones, se diseñan las partes ambiguas de acuerdo a corazonadas, lo que pospone la manifestación del problema hasta las etapas finales del ciclo, cuando regresar a la etapa de análisis es demasiado costoso aunque inevitable.

Falta de Entendimiento Entre el Usuario y el Analista. Aún cuando se disponga de tiempo suficiente para efectuar el análisis del sistema, existe otro factor que puede decrementar la calidad de dicho análisis: el entendimiento entre los usuarios y los analistas.

Cuando esta comunicación no es la adecuada, lo cual es muy frecuente, el sistema que se está desarrollando empieza a diferir del que el usuario realmente necesita y desea. Mientras más tiempo pase antes de que el analista se de cuenta de las diferencias, las correcciones serán más complicadas y costosas, pero de no hacerse se terminará por construir un sistema que será subutilizado o no será utilizado por completo.

¿Cómo es posible que aún existiendo estrecha colaboración entre el analista y el usuario, el primero no logre captar las verdaderas necesidades del segundo?. Existen dos respuestas principales para esta pregunta, la primera de ellas radica en el hecho de que la mayor parte de las veces los usuarios son personas ajenas al área informática y el *lenguaje* que utilizan difiere del empleado por los analistas. Esto lleva a ambigüedades ocasionadas por términos o expresiones que tienen diferente significado para cada una de las partes, y la imagen del sistema que se van formando comienza a ser distinta. Para atacar este problema se ha sugerido la utilización de lenguajes formales, es decir, notaciones rigurosas que tengan una sola interpretación sin importar la formación de quien las interprete, sin embargo, ello requiere un esfuerzo adicional, principalmente por parte del usuario, para familiarizarse con su manejo.

La segunda respuesta al problema de comunicación usuario - analista tiene que ver con las técnicas deficientes aplicadas a las entrevistas en donde el analista intenta obtener toda la información posible acerca de los requerimientos del usuario. En estas pláticas, suele dejarse de lado información relevante, en ocasiones por olvido, por resistencia del usuario a aceptar el sistema, o porque supone que algunos datos son obvios y no necesita hacerlos explícitos. Por este motivo, las preguntas de las entrevistas deben ser planeadas cuidadosamente y efectuadas con suficiente tacto para lograr la cooperación del usuario.

Ausencia de Diseño. El principal problema de la fase de diseño de los sistemas, es que muchas veces ni siquiera existe. En buen número de los proyectos en donde una sola persona o un grupo reducido actúan como analistas- diseñadores-programadores-operadores sucede que tras efectuarse el análisis del problema se piensa que se tienen los suficientes elementos para empezar la codificación. Los archivos de datos se van diseñando conforme se van necesitando, el flujo de los programas se establece arbitrariamente confiando en la experiencia y una vez creados varios programas se van conjuntando en módulos de acuerdo a sus características.

Si bien los sistemas así desarrollados pueden trabajar satisfactoriamente, lo hacen a un alto costo, ya que presentan múltiples problemas que no trascienden debido a que la persona que lo desarrolló está casi siempre allí para corregirlos y ponerlos a trabajar de nuevo con algunos *parches* adicionales. Lo que no se toma en cuenta es el tiempo que se ha ido utilizando para realizar esas pequeñas o no tan pequeñas composturas y que acumulado seguramente supera ampliamente al que hubiera sido necesario para realizar un buen diseño.

Al observar un programa detenidamente, es posible establecer si tiene su origen en un diseño detallado o si nació por generación espontánea, pues éstos últimos tienden a ser desorganizados, usan algoritmos menos eficientes y por lo general no están documentados, convirtiéndose en una fuente de problemas cuando deben ser modificados por una persona distinta a su creador.

Comunicación Deficiente Entre Equipos de Trabajo. Este tipo de problemas acosan principalmente a los sistemas de grandes proporciones en los que interviene un equipo de personas muy extenso. Si la comunicación se pierde, los problemas no se hacen esperar: aparición de versiones distintas de las mismas funciones, uso de datos no actualizados, duplicación de trabajo, retrasos en el desarrollo por esperas innecesarias, etc.

Los investigadores del área de Ingeniería de Sistemas, reco-

miendan que los canales de comunicación se reduzcan tanto como sea posible, es decir que no se permita el traslado de información a través de dos miembros cualesquiera de diferentes equipos de trabajo, sino que se establezcan vías autorizadas y exclusivas para tal fin. Es decir, que se aplique una jerarquía lo suficientemente estricta que garantice el control de la información. Si un equipo modifica un programa o un archivo, lo comunica a los demás a través de sus portavoces; si alguien requiere procesos que están siendo desarrollados por otro equipo, debe hacérselo saber por medio de las personas autorizadas, etc. De esta manera la responsabilidad del control de la información se concentra en un pequeño grupo de personas que pueden comunicarse satisfactoriamente.

Disparidad de Criterios al Desarrollar. De nuevo este es un problema que se presenta con más frecuencia en sistemas grandes que en sistemas reducidos, y afecta principalmente la fase de codificación.

Cuando un sistema no sigue determinados estándares durante su programación, especialmente en las interfaces con el usuario, causa la impresión de ser un sistema integrado a fuerza con programas de diferentes partes. Por ejemplo es preferible un sistema en el que el usuario vea en el mismo lugar de la pantalla la información similar de los diferentes módulos a uno en donde a veces se despliega en la parte superior, otras veces en la parte media y otras en la inferior. De la misma manera, la entrada de datos debe ser similar siempre que sea posible pues si se utilizan diferentes métodos indistintamente, no sólo se pierde calidad en la presentación sino que se dificulta el aprendizaje de la operación del sistema.

Para el equipo que efectuará el mantenimiento, también es mucho más cómodo que el sistema conste de programas desarrollados bajo una sola filosofía, pues de esta manera será posible aplicar la experiencia obtenida previamente en la modificación de un módulo cualquiera.

Considerar al Código Como Unico Producto a Entregar. Si bien el código es la parte más importante de un sistema computarizado, existen otros productos con los que el usuario debe contar para explotarlo de manera óptima.

Por ejemplo el *Manual de Operación* es un documento básico sin el cual el usuario nunca dejará de depender del área de informática para realizar sus funciones. No es raro encontrarse con un sistema del cual se utilizan sólo una parte de las opciones debido a que se desconoce el funcionamiento de las demás porque el grupo que lo implementó entregó únicamente una pequeña nota en donde se explicaba a grandes rasgos el modo de empleo.

Quizá por su formación, los analistas y programadores rehuyen la tarea de escribir manuales, pues siempre hay algo más importante que hacer que sentarse ante un papel a describir un sistema, los manuales se dejan para después hasta que la organización se acostumbra a operar sin ellos.

Falta de Documentación. Este es un problema similar al anterior pero a una escala mayor, no sólo se omitió la documentación para el usuario, sino que no se escribió documentación alguna. Las fases de planeación, análisis, diseño y programación se realizaron pero no se registraron o no se hizo a conciencia, para entender cualquier aspecto del sistema es necesario interpretar lo que dicen los programas fuente.

Esta situación se agrava generalmente porque se manifiesta cuando la persona que hizo los programas y el equipo que intervino en el análisis y diseño ya no se encuentran dentro de la organización. Un nuevo analista y/o programador debe corregir algún error o añadir alguna opción al sistema y se encuentra con que tiene que empezar desde cero o peor, puesto que no cuenta con herramientas que le ayuden a aplicar sus ideas sino que tiene que interpretar lo que el anterior programador pensaba al codificar sus programas, con lo que el tiempo requerido para completar el objetivo se multiplica.

Documentación Desactualizada. Tal vez peor que el caso anterior es el de los sistemas que cuentan con una documentación que no corresponde a su estado actual, sino que se estancó en una de las primeras versiones.

Cuando se modifica un programa, es común que no se reflejen los cambios en los diferentes documentos que son afectados, en ocasiones porque el cambio no se considera significativo, porque no se tiene la costumbre de analizar un sistema hacia atrás para determinar la magnitud de una modificación o simplemente porque se deja para después. Pero se olvida que al acumularse una serie de cambios aunque sean cambios pequeños, el impacto sobre el sistema se hace notorio, en ocasiones considerablemente.

¿Que sucede cuando un analista se basa en una documentación inconsistente con el sistema?. Por lo general sucede que el analista trabaja mucho más, pues recorre caminos equivocados u obsoletos, realiza cambios que presentan efectos secundarios imprevistos y termina por olvidarse de la documentación y meterse de lleno al código fuente.

Dificultad de Mantenimiento. En la mayoría de los sistemas sino es que en todos, la fase de mantenimiento es la más extensa, y su excesiva longitud se hace más pesada porque frecuentemente va aunada a su dificultad. Un sistema es difícil de mantener por todos los problemas mencionados en esta sección, que de una manera o de otra se reflejan en esta etapa del ciclo de vida.

El mantenimiento de un sistema sólo puede facilitarse cuando se corrigen las deficiencias que se arrastran durante su desarrollo, un enfoque adecuado de cada parte de la vida de un sistema, redundará en un costo relativamente bajo en su mantenimiento, al disminuir la cantidad de tiempo y recursos que es necesario dedicarle.

Además es importante modificar la mentalidad del equipo humano que se encarga del mantenimiento, haciéndosele sentir que esta actividad es tan importante como las demás y que la creatividad no

desaparece en ella (lo que realmente es cierto). Así el esfuerzo consumido en el mantenimiento se ubicará en el destacado lugar que le corresponde.

Baja Calidad de los Sistemas Computarizados. Cuando no se sigue una metodología como la propuesta por la Ingeniería de Sistemas. es muy difícil que el producto obtenido cumpla con los requerimientos de calidad impuestos por una organización. El sistema presentará con toda seguridad carencias, errores. poca eficiencia y en fin una serie de características que lo calificarán como un mal sistema.

Desafortunadamente. un mal sistema puede hacer que un usuario sin experiencia en cómputo. se olvide de los beneficios que le han proporcionado otros sistemas o que no escuche las ventajas que han recibido otras personas gracias a ellos y se vuelva desconfiado con su información y prefiera procesarla manualmente.

Todos los que estamos involucrados de una forma o de otra en el proceso de desarrollo de sistemas computarizados. estamos obligados a hacer nuestro máximo esfuerzo por ampliar su ámbito de operación. lo que se producirá como una consecuencia lógica al ir creciendo el número de sistemas que satisfacen plenamente los objetivos para los que fueron creados. Los puntos que se tratan en este trabajo tienen el fin de facilitar el desarrollo de sistemas de alta calidad aplicando los principios de la Ingeniería de Sistemas a través de herramientas computarizadas.

BIBLIOGRAFIA DEL CAPITULO

- Ian Sommerville. "Ingeniería de software". Addison-Wesley Iberoamericana. 2a. Edición 1988.

- Roger S. Pressman. "Ingeniería del software: un enfoque práctico", McGraw Hill. 2a. Edición 1989.

- Henry C. Lucas Jr.. "The analysis, design and implementation of information systems". McGraw Hill. 3a. Edición 1985.

- Frederick P. Brooks Jr.. "The mythical man-month". Addison-Wesley. 1979.

- Warren Keuffel. "Solving the right problem", Data Based Advisor, Agosto 1991. pp 134-137.

- Marc Rettig. "Nobody reads documentation". Communications of the ACM. Vol. 34 No. 7. Julio 1991. pp 19-24.

- Warren Keuffel. "Build it right the first time". Data Based Advisor". Junio 1991. pp 102-108.

- Susan Perschke. "10 steps to better programming". Data Based Advisor. Junio 1991. pp 112-113.

Capítulo 2

METODOLOGIAS PARA EL DESARROLLO DE SISTEMAS.

2.1 ANTECEDENTES.

Como se mencionó en el capítulo anterior, los problemas a los que se enfrenta el desarrollo de sistemas son muy diversos y son consecuencia de diferentes factores. Estos factores se han ido multiplicando con la amplia difusión que han tenido las computadoras en todo tipo de organizaciones, que reclaman la producción de sistemas de mejor calidad en un menor tiempo.

Es entonces cuando los analistas y programadores comienzan la búsqueda de métodos que le brinden una mayor disciplina a su labor, y en esa búsqueda recurren a las metodologías o técnicas estructuradas.

Una metodología estructurada es *un conjunto de reglas, métodos y postulados utilizados para organizar el enfoque de solución de problemas, listando, diagramando y documentando todos sus pasos.*[1] Dada esta definición, es comprensible por qué las técnicas estructuradas ayudan a estandarizar y sistematizar el desarrollo y mantenimiento de sistemas, pues lo enfocan desde el punto de vista de una disciplina de ingeniería y no como la reunión de esfuerzos de personas aisladas.

[1] Carma McClure, "The CASE for structured development".

Los objetivos principales de las metodologías estructuradas son los siguientes:

- Producir sistemas de alta calidad con comportamiento totalmente predecible.
- Crear sistemas cuya modificación y mantenimiento sean sencillos.
- Simplificar el proceso de producción de programas y los programas mismos.
- Proporcionar un control efectivo en el proceso de desarrollo y facilitar las estimaciones de tiempo y recursos.
- Acelerar el desarrollo de sistemas.
- Reducir los costos de desarrollo.

Para alcanzar estos objetivos. las diferentes metodologías estructuradas deben efectuar una serie de mejoras en el desarrollo de sistemas. las cuales pueden ser consideradas como objetivos secundarios destacando entre ellas las que se detallan a continuación:

a) Descomposición de problemas complejos en otros progresivamente más simples. Con esto se consigue partir el modelo del sistema en varios módulos que cumplen funciones específicas y que están organizados jerárquicamente.

b) Control de complejidad. Busca reducir el número de interacciones entre módulos diferentes. Comúnmente los buenos diseños son aquellos cuya simplicidad es mayor.

c) Empleo de las técnicas de diagramación más claras. La fuerza de las técnicas estructuradas radica en buena medida en su eficiencia para utilizar el lenguaje gráfico para comunicar las ideas de persona a persona.

d) Incremento en la legibilidad de los programas. La claridad y legibilidad de los programas es esencial para que los encargados del

mantenimiento realicen una labor adecuada. La estandarización del estilo de programación contribuye a aumentar la claridad.

e) Mejoramiento de la comunicación con los usuarios. Las técnicas utilizadas deben permitir al usuario manifestar claramente sus necesidades y verificar el diseño del sistema antes de comenzar con la implantación.

f) Empleo de métodos consistentes y de fácil enseñanza. Con el objeto de ser utilizadas extensivamente dentro de una organización, las técnicas estructuradas deben poder transmitirse con relativa facilidad y deben poder aplicarse a todos los tipos de sistemas existentes en ella.

g) Comunicación entre los miembros del equipo de desarrollo. Las metodologías deben minimizar las comunicaciones complejas entre los analistas y siempre que sea posible utilizar representaciones formales.

h) Minimizar el número de participantes en los equipos de desarrollo. En los equipos pequeños no suelen presentarse con tanta frecuencia problemas ocasionados por las interacciones humanas.

i) Localizar los errores en las primeras etapas. Como se mencionó anteriormente, los errores son más costosos mientras más tiempo transcurra antes de su detección, por lo que se intenta minimizar su número en cada etapa.

j) Facilitar la administración de los datos. Esto se logra mediante una definición de datos uniforme y el establecimiento de las estructuras de datos correctas.

k) Establecer un control eficaz del proyecto. Al disciplinar el desarrollo del sistema se facilita el seguimiento de las diferentes actividades que componen el proyecto.

2.2 CLASIFICACION DE LAS TECNICAS ESTRUCTURADAS.

Las metodologías estructuradas no han permanecido estáticas desde su aparición en los años 70, sino que han venido evolucionando para cubrir una parte cada vez mayor del desarrollo de sistemas. Así, en un principio se enfocaron únicamente a la producción de programas y a encontrar la manera en que éstos debían ser escritos para ser más comprensibles. Las técnicas aparecidas en esta época constituyen la llamada **Programación Estructurada**.

A mediados de la misma década, los conceptos de organización y disciplina, que dieron vida a la Programación Estructurada, se extendieron a la fase de diseño. La atención se centró en un nivel más alto del proceso de solución de problemas, buscando la correcta definición de módulos, así como de las interfaces entre ellos. Las metodologías provenientes de esta etapa forman la corriente conocida como **Diseño Estructurado**.

El **Análisis Estructurado** surge más adelante, cuando se hace patente la necesidad de mejorar la especificación de requerimientos de los sistemas, con el fin de reducir el número de errores causados por la incorrecta comprensión del problema.

Finalmente, la **Ingeniería de Información** intenta aplicar técnicas estructuradas no sólo al desarrollo de los sistemas sino a la organización vista como un todo, considerando cómo interactúan los sistemas, el personal, la información, etc.

Dadas sus características, las tres primeras etapas de la evolución de las metodologías estructuradas (Programación Estructurada, Diseño Estructurado y Análisis Estructurado), son catalogadas como pertenecientes a la Ingeniería de Sistemas mientras que la última forma por sí sola una tendencia más amplia, que comienza en un nivel de abstracción más alto.

Independientemente del bloque al que pertenecen por su enfoque del ciclo de vida de los sistemas, las metodologías pueden clasificarse de acuerdo a su orientación en tres grupos: **orientadas a los procesos, orientadas a los datos y orientadas a la información.**

Las metodologías orientadas a los procesos consideran que éstos son los componentes básicos de los sistemas y que por lo tanto deben ser definidos primeramente. Los datos se definen a continuación de los procesos. La herramienta principal de estas metodologías es el diagrama de flujo, que muestra el camino seguido por los datos desde la entrada, a través de los procesos, hasta la salida.

El enfoque orientado a los datos, considera las entradas y salidas como lo más importante del sistema. Se definen primero las estructuras de datos y a partir de ellas se derivan los procesos necesarios para convertir las entradas en las salidas requeridas. Las metodologías estructuradas con esta orientación hacen uso de los diagramas de árbol para representar tanto a los datos como a los procesos.

En la corriente orientada a la información, se define primero un modelo que represente el uso de la información a lo largo de la organización y se analizan a nivel general sus objetivos y necesidades. Basándose en este modelo se van construyendo los sistemas individuales.

Otra forma alternativa de dividir a las metodologías estructuradas es de acuerdo al tipo de sistemas que pueden producir. La mayor parte de ellas pueden utilizarse para construir **sistemas de información**, mientras que con algunas de ellas pueden desarrollarse **sistemas de tiempo real**, aunque éstas suelen ser variaciones de las primeras.

Los sistemas de información se caracterizan por tener estructuras de datos complejas y grandes volúmenes de entradas y salidas. Por el contrario, los sistemas de tiempo real tienen estructuras de datos simples y poco volumen de entradas, y deben responder a una señal externa produciendo una salida dentro de un tiempo establecido.

2.3 METODOLOGIAS MAS UTILIZADAS.

A continuación se listan las seis metodologías estructuradas más difundidas[2], cabe señalar que el nombre por el que se les conoce comúnmente es el de su creador.

METODOLOGIA	AUTOR
ANALISIS ESTRUCTURADO	DE MARCO.
ANALISIS ESTRUCTURADO	GANE - SARSON.
DISEÑO ESTRUCTURADO	YOURDON.
DISEÑO ESTRUCTURADO	JACKSON.
INGENIERIA DE INFORMACION	MARTIN.
DESARROLLO DE SISTEMAS	WARNIER - ORR.

El cuadro siguiente muestra la clasificación de estas seis metodologías de acuerdo a los diferentes aspectos tratados con anterioridad:

[2]Carma McClure. "CASE is software automation".

CLASIFICACION DE LAS METODOLOGIAS ESTRUCTURADAS MAS COMUNES

	YOURDON	GANE SARSON	DE MARCO	WARNIER ORR	JACKSON	MARTIN
ORIENTACION						
Datos	–	–	–	●	●	–
Información	–	–	–	–	–	●
Procesos	●	●	●	–	–	–
TIPO DE SISTEMAS						
Información	●	●	●	●	●	●
Tiempo Real	●	●	●	–	●	–
INGENIERIA						
Sistemas	●	●	●	●	●	–
Información	–	–	–	–	–	●
TIPO DE METODOLOGIA						
Diseño estructurado	●	–	–	●	●	●
Análisis Estructurado	–	●	●	●	–	●

Las características principales de cada una de estas metodologías, se describen brevemente en esta sección, adicionalmente en las secciones siguientes se ahonda en el tratamiento de las metodologías más relevantes para el presente trabajo.

ANALISIS ESTRUCTURADO. Se consideran representativos de esta corriente los trabajos de Tom De Marco y de Gane-Sarson. que presentan similitudes pues su objetivo es producir una especificación estructurada del sistema, basándose en una organización jerárquica descendente y en el uso de herramientas gráficas para la comunicación y la documentación.

La especificación resultante se compone de lo siguiente:

i) Un conjunto de diagramas de flujo de datos, que presentan los procesos del sistema y los datos que constituyen las interfaces. El primer diagrama es la representación del sistema desde el nivel más alto y los diagramas subsecuentes lo van descomponiendo en niveles más detallados.

ii) Un diccionario de datos en donde se encuentra la definición de todas las entidades que aparecen en los diagramas de flujo.

iii) La especificación de los procesos, que describe detalladamente las actividades internas de los procesos señalados en los diagramas.

La metodología de De Marco consta de los siete pasos siguientes:

1) Construir un modelo físico que represente al sistema actual.
2) Construir. mediante diagramas de flujo de datos. el modelo lógico equivalente.
3) Desarrollar el modelo lógico del nuevo sistema, teniendo como punto de partida el modelo anterior y especificando los nuevos procesos

manuales y automatizados.

4) Producir uno o varios modelos físicos posibles para implantar el sistema con distintos niveles de automatización.
5) Realizar estimaciones de costo y calendario de todos los modelos propuestos.
6) Seleccionar uno de los modelos basándose en las estimaciones anteriores.
7) Crear la especificación estructurada definitiva integrando los diagramas, el diccionario y la especificación de procesos.

Dentro de esta secuencia, el tercer paso es el más importante, por lo tanto es el que requiere más atención y recursos.

Por su parte, la metodología Gane-Sarson presenta los cinco pasos siguientes:

1) Construir un modelo lógico del sistema actual.
2) Elaborar el modelo lógico del nuevo sistema mediante diagramas de flujo de datos, diccionario de datos y especificación de procesos, asimismo elaborar un modelo lógico de la base de datos que represente el contenido de los almacenes de datos.
3) Diseñar físicamente la base de datos.
4) Elaborar un modelo físico para el nuevo sistema.
5) Integrar la especificación definitiva.

DISEÑO ESTRUCTURADO DE YOURDON. La metodología Yourdon se aplica al diseño general del sistema, así como al diseño detallado de los programas. Su producto principal es la gráfica de estructura (*Structure Chart*) en la que se muestra la estructura general del sistema de una manera jerárquica, señalando los procesos que componen los programas y los datos que los comunican.

Está integrada por los cuatro pasos siguientes:

1) Elaborar el diagrama de flujo de datos en el que se represente el sistema a diseñar mediante el flujo de datos entre sus procesos.
2) Generar la gráfica de estructura. A partir del diagrama de flujo de datos anterior, representar el diseño de programas como una jerarquía de procesos.
3) Evaluar el diseño, midiendo la coherencia y el acoplamiento de los módulos.
4) Preparar la implantación del diseño. Realizar el diseño físico de los programas mediante la división del diseño lógico en unidades.

En este proceso, el segundo paso es el más importante y en él se efectúa el grueso del trabajo de diseño. Yourdon propone dos estrategias para la conversión de los diagramas de flujo de datos en la gráfica de estructura: el análisis de transformaciones y el análisis de transacciones. En el primero, los diagramas de flujo se dividen en tres partes: ramas aferentes que son las entradas, procesos lógicos llamados transformaciones centrales y ramas deferentes o de salida. En el análisis de transacciones, se define en el diagrama de flujo de datos un proceso que realiza la transformación central y una serie de módulos de transacciones.

DISEÑO ESTRUCTURADO DE JACKSON. Esta metodología se utiliza también para el diseño del sistema, difiere de Yourdon en su orientación a los datos y no a los procesos o flujos de información. Su enfoque consiste en definir la estructura de los programas a partir de la estructura de datos.

Como todas las metodologías estructuradas, se basa en el uso de técnicas gráficas, utilizando principalmente las siguientes:

i) Diagrama de red del sistema (*System-Network Diagram*) en el que se describe el flujo de datos entre programas.

ii) Diagrama estructurado de árbol (*Tree Structured Diagram*) que representa de manera jerárquica la estructura de programas

y datos.

iii) Texto estructurado que es una forma de pseudocódigo empleada para la definición detallada de programas.

Jackson supone la existencia previa de una especificación completa del sistema, y a partir de ella efectúa el diseño correspondiente mediante los siguientes cuatro pasos:

1) Definición de datos. A través de un diagrama de red se representan todas las corrientes de entrada y salida de datos, a continuación cada corriente se describe jerárquicamente con un diagrama de árbol.

2) Definición del programa. Las estructuras de datos obtenidas en el primer paso se combinan para formar la estructura del programa, identificando todas las correspondencias entre componentes de la estructura de datos y encontrando las relaciones consumo-producción (cuáles entradas deben consumirse para producir cuáles salidas). La estructura del programa se construye de acuerdo a estas relaciones. La verificación del diseño se lleva a cabo regresando la estructura del programa a sus corrientes de datos individuales.

3) Definición de operaciones. Se listan las operaciones requeridas por el programa para convertir las entradas en salidas y cada operación se sitúa en el lugar apropiado de la estructura del programa. La corrección se establece verificando que todas las entradas se consuman y todas las salidas se produzcan.

4) Definición detallada. En este paso se traslada a pseudocódigo el diagrama de estructura del programa junto con sus operaciones.

Para que el diseño de programas complejos sea posible con esta metodología, éstos deben dividirse de antemano en programas más sencillos a los cuales se les aplica de manera individual el procedimiento

descrito anteriormente.

INGENIERIA DE INFORMACION DE MARTIN. Esta metodología permite construir sistemas de información siguiendo paso a paso todas sus fases, desde el nivel más alto posible que es el estudio de la organización en donde operará el sistema.

Las herramientas gráficas más importantes en esta metodología son los diagramas de entidad-relación, los diagramas de árbol y los de flujo de datos. Está formada por las cuatro etapas que se describen a continuación:

Etapa 1 (Planeación). Se concentra en el estudio de alto nivel de la organización, definiendo modelos de sus funciones y de sus datos. Identifica todas las entidades de datos existentes. la manera en que se relacionan y las áreas donde son producidas y consumidas, representándolas en un diagrama de entidad-relación. La estructura organizacional y sus funciones se representan en un diagrama de árbol llamado Diagrama de Descomposición.

Etapa 2 (Análisis). A partir de sus funciones globales, la organización se divide en áreas locales. Los procesos y datos necesarios para cumplir los objetivos de la organización dentro de un área en particular, se definen extrayendo del diagrama general de entidad-relación la parte correspondiente a dicha área y derivando del diagrama de descomposición procesos locales que se representan con diagramas de flujo de datos.

Etapa 3 (Diseño). En esta etapa. la metodología se vuelve similar a las descritas previamente y se dedica al diseño de los procedimientos y estructuras de datos necesarios para implantar los procesos definidos en la etapa 2.

Etapa 4 (Construcción). Transfiere el diseño lógico del sistema a un diseño físico. incluyendo programas. bases de datos y documentación.

METODOLOGIA WARNIER-ORR. También llamada Metodología para el Desarrollo de Sistemas Estructurados por los Datos, se le conoce por sus siglas en inglés DSSD. Esta es una metodología orientada a los datos. su filosofía consiste en derivar completamente la estructura de datos a partir de los datos de salida. y la estructura de los procesos a partir de la estructura de los datos.

Su principal herramienta son los diagramas de Warnier, que consisten en conjuntos de llaves ({) anidadas y que se utilizan para representar la estructura jerárquica de los procesos así como el flujo de los datos y actividades dentro de ellos.

Contempla tres fases principales:

a) Planeación del Proyecto. Desarrolla el plan del sistema a desarrollar compaginando sus objetivos con los de la empresa.

b) Definición de Requerimientos. Se establecen los requerimientos de los usuarios y de la organización y se asocian con las salidas del sistema. En un diagrama de entidad-relación se representan todas las entidades del sistema y las transacciones entre ellas con el fin de establecer su alcance.

c) Diseño. Usando como base las salidas ya definidas lógicamente. se diseña la base de datos lógica y a continuación los procesos lógicos de actualización. Ambas definiciones se realizan mediante diagramas Warnier. Finalmente se realiza el diseño físico de la base de datos y de los procesos.

2.4 ANALISIS ESTRUCTURADO DE TOM DE MARCO.

En la presente sección se describen los pormenores de la metodología de Tom de Marco, que como se señaló anteriormente, corresponde a la corriente llamada **Análisis Estructurado**, por enfocarse al estudio de los requerimientos del sistema.

2.4.1 OBJETIVOS.

El trabajo realizado en la fase de análisis, debe ser considerado como un puente entre el usuario y el desarrollo del sistema, en donde el producto más importante es la especificación de requerimientos. El Análisis Estructurado busca que esta especificación presente determinadas características que le permitan realizar la conexión entre ambos extremos de forma más eficiente.

La especificación de requerimientos se convierte en una especificación estructurada con las siguientes cualidades:

a) Presenta sus conceptos gráficamente. El lenguaje gráfico permite visualizar el sistema a desarrollar más fácilmente, pero lo que es más importante, ayuda a resolver definitivamente el eterno problema de incomprensión entre el analista y el usuario, estableciendo un lenguaje común que favorece la colaboración entre ambos. Una especificación gráfica es preferible sobre una que haga uso exclusivo del texto en sus descripciones.

b) Particiona el problema a sistematizar. Su enfoque descendente le permite ir mostrando diferentes niveles del sistema, comenzando por el más general y detallándolo progresivamente, creando módulos o subsistemas cada vez más simples.

c) Es de fácil mantenimiento. En la especificación estructurada, la redundancia es controlada y minimizada, por lo que al surgir la necesidad de modificaciones pueden establecerse claramente las partes afectadas, sin correr el riesgo de actualizar la especificación sólo parcialmente.

d) Describe el sistema de manera precisa. A través de diagramas de flujo de datos especifica precisamente los procesos, a la vez que las interfaces entre ellos son descritas con igual exactitud en el diccionario de datos. Si por el contrario se utilizara el lenguaje tradicional para describir los mismos procesos y grupos de datos, se introduciría inevitablemente cierto grado de ambigüedad.

e) Su enfoque es lógico no físico. Al eliminar restricciones impuestas por el medio ambiente físico se consiguen dos importantes ventajas: proteger la especificación contra modificaciones en los equipos y proporcionar a los encargados del diseño la mayor libertad posible para cumplir con su parte del trabajo.

Todas estas cualidades de la especificación estructurada ayudan al analista a minimizar la probabilidad de introducir errores graves en la especificación, lo que se traducirá a su vez en reducción de los costos originados por el mantenimiento, que como se mencionó en su oportunidad llega a convertirse en el principal consumidor de recursos en el desarrollo de los sistemas.

2.4.2 COMPONENTES DE LA ESPECIFICACION ESTRUCTURADA.

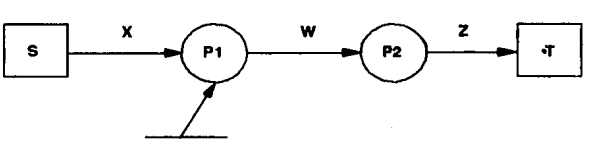
Para conseguir sus objetivos, el Análisis Estructurado de Tom de Marco construye la especificación estructurada en base a tres componentes principales: un conjunto de diagramas de flujo de datos para la representación de procesos y sus interfaces, un diccionario de datos con la definición de todas las entidades mencionadas en los diagramas y una serie de descripciones (mini-especificaciones) de la lógica

de los procesos representados en los diagramas. Cada una de estas partes integrantes de la especificación se trata más detenidamente a continuación.

DIAGRAMAS DE FLUJO DE DATOS (DFD). Un DFD es la representación del sistema en forma de red. Está constituido por los siguientes elementos:

- Flujos de datos. Son las vías a través de las cuales fluyen paquetes de información de composición determinada. Constituyen las interfaces entre las demás componentes y se representan mediante un vector con nombre cuya dirección es significativa.
- Procesos. Son las transformaciones que convierten los datos de entrada en los datos de salida. Se representan por medio de círculos.
- Archivos. Son almacenes temporales de datos. Están representados por una línea recta con nombre.
- Fuentes y Receptores. Son personas u organizaciones fuera del sistema que son el origen o que reciben su información. Se les representa a través de rectángulos.

En el siguiente ejemplo se muestran las 4 componentes de los DFD:



La manera correcta de interpretar un DFD es comenzando con los flujos de datos y de ahí a los procesos, el ejemplo mostrado debe

leerse así: *La información X se obtiene de la fuente S y es transformada en W por el proceso P1 que requiere para ello acceso al archivo F. W es transformada subsecuentemente en Z por el proceso P2. Finalmente Z es entregada al receptor T.*

Los nombres asignados a flujos de datos y procesos deben ser únicos y representativos de la información que contienen o del trabajo que realizan respectivamente. Cuando los nombres no son suficientemente explícitos, es probable que se haya elegido erróneamente el modelo del sistema.

Cuando un sistema es demasiado complejo, es necesario crear subsistemas para poder describirlo completamente. Cada proceso del diagrama original se descompone en nuevos procesos y flujos de datos que forman un segundo DFD . Si los procesos en este segundo diagrama continúan siendo muy elaborados, se descomponen sucesivamente tantas veces como sea necesario, produciéndose un conjunto de DFD estructurado por niveles.

El primer diagrama del conjunto es llamado diagrama de contexto y su propósito es determinar el dominio del sistema mostrando los flujos de datos que corren hacia su interior y hacia el exterior.

Los niveles intermedios del conjunto están formados por procesos que a su vez pueden ser particionados nuevamente. Generalmente existirán varios niveles intermedios.

El último nivel contiene procesos que no admiten más descomposiciones y que son llamados funciones primitivas. En una situación óptima, una función primitiva contará únicamente con un flujo de entrada y uno de salida.

El conjunto completo de DFD debe cumplir con la regla de balanceo que establece lo siguiente:

Todos los flujos de entrada en un diagrama derivado deben estar representados, en el nivel superior, por los mismos flujos de entrada al proceso asociado. Similarmente, los flujos de salida deben coincidir para el diagrama derivado y el proceso asociado.

Una vez elaborado el conjunto de DFD estructurado por niveles, cada uno de estos niveles debe verificarse para comprobar su corrección y utilidad.

Las causas más comunes de que un DFD sea incorrecto pueden resumirse en esta relación:

- Flujos faltantes. Cuando en el diagrama no se encuentra información indispensable para que un proceso funcione de la manera prevista. Este problema es atacado por la regla de conservación de información que estima que *todo proceso debe ser capaz de construir su salida utilizando exclusivamente tanto la información contenida en los flujos que entran a él, como información constante.*
- Flujos superfluos. Son aquellos que estando presentes en algún DFD no son utilizados en ninguna forma, por lo que su existencia sólo aumenta la complejidad de las interfaces.
- Procesos faltantes. Las transformaciones consideradas no son suficientes para convertir las entradas en las salidas requeridas, reflejando con esto la necesidad de considerar más procesos.
- Errores entre niveles. No se respeta la regla de balanceo, al aparecer o desaparecer flujos de datos entre un nivel y otro.
- Errores de concepción. Este es el tipo de error más difícil de detectar y a la vez el más grave, se refiere al hecho de que aún cuando los DFD sean sintácticamente correctos, no reflejen realmente las necesidades del área. Para evitar esta situación debe contarse con la colaboración del usuario responsable.

En lo referente a la utilidad de los DFD, deben cuidarse dos aspectos principales: que las interfaces que describen tengan un grado de complejidad razonable y que los nombres de todas sus entidades sean significativos. El primer punto es importante ya que los diagramas serán utilizados como herramienta de comunicación entre todas las personas involucradas, y si se cuenta con interfaces demasiado complicadas no se conseguirá transmitir las ideas adecuadamente. Por otro lado, cuando los procesos o flujos tienen nombres poco sugerentes, es síntoma de que es necesario buscar una partición que resulte más adecuada.

DICCIONARIO DE DATOS. El diccionario de la especificación es un almacén de información acerca de la información. También puede verse como un conjunto riguroso de definiciones de todos los elementos de los DFD. El diccionario debe contener principalmente definiciones para los flujos de datos, los archivos y los procesos de acuerdo a la siguiente relación:

Existirá una entrada única en el Diccionario de Datos para cada flujo que aparezca en cualquier nivel del conjunto estructurado de DFD; existirá una entrada única para cada archivo que haya sido referenciado en los DFD y existirá una y sólo una entrada para cada función primitiva del nivel inferior del conjunto de DFD.

Para definir una entidad cualquiera, se debe determinar a qué clase pertenece y adicionalmente señalar características particulares que la distingan de las demás.

Los flujos de datos se definen comúnmente como la suma o asociación de diferentes componentes, que a su vez se definen como nuevas asociaciones hasta llegar a un nivel de componentes primitivas que se definen de manera única mediante una tabla de valores posibles. Estas componentes primitivas son llamadas elementos de información y no admiten mayor descomposición. Este proceso es similar al utilizado en los DFD con los procesos.

Para la definición de los archivos, se contará con una entrada similar a la de los flujos de datos en la que se señale su composición y modo de acceso. pero adicionalmente se incluirán gráficas que permitan apreciar las relaciones existentes entre los diferentes archivos. Estas gráficas son los llamados Diagramas de Estructura de Datos y en ellos se representa cada archivo mediante un bloque y las ligas entre los archivos con flechas entre los bloques, cuya dirección indica la dirección de la liga.

Las entradas para los procesos existirán únicamente para las funciones primitivas y contendrán una identificación del proceso al que se refieren y que lo relacione con el DFD que constituye su entorno. La descripción detallada del proceso consistirá en una miniespecificación con las características que se mencionarán más adelante.

Teniendo en cuenta que uno de los objetivos del Análisis Estructurado es construir una especificación no redundante. el diccionario de datos no debe contener redundancia alguna. lo que se aplica a la información contenida en sus diferentes entradas. así como a la información ya representada en alguna otra parte de la especificación como son los DFD. Para lograr esto debe observarse la siguiente regla:

La información acerca de la composición de las entidades se ubicará en el Diccionario de Datos, la información del contenido y procesamiento de las entidades se incluirá en la documentación de los procesos (miniespecificaciones) y el tránsito de la información en el sistema se representará en los DFD.

Otras propiedades indispensables en el diccionario de datos son facilidad de mantenimiento, facilidad de acceso y establecimiento de convenciones ortogonales para el desarrollo de definiciones. donde la ortogonalidad implicará que una entidad puede ser descrita de una y solo una manera.

Obsérvese el ejemplo de la figura 2.2. En él destacan tres puntos importantes:

EJEMPLO DE CONSTRUCCION DE DEFINICIONES EN EL DICCIONARIO DE DATOS

REGISTRO_CONTRIBUYENTES = LETRAS_REGISTRO +
NUMEROS_REGISTRO + HOMO_CLAVE

LETRAS_REGISTRO = LETRAS_APELLIDO_PATERNO +
INICIAL_APELLIDO_MATERNO +
LETRA_NOMBRE

LETRAS_APELLIDO_PATERNO = INICIAL_APELLIDO_PATERNO +
SIGUIENTE_VOCAL_AP_PATERNO

LETRA_NOMBRE = [INICIAL_NOMBRE
X]

NUMEROS_REGISTRO = ANIO_NACIMIENTO + MES_NACIMIENTO
+ DIA_NACIMIENTO

HOMO_CLAVE = [A,B,...,Z] + [A,B,...,Z]

FIGURA 2.2

i) Las entradas del diccionario de nivel superior (REGISTRO_CONTRIBUYENTES, LETRAS_REGISTRO, LETRAS_APELLIDO_PATERNO, etc) son definidas en base a otras entradas de menor nivel.

ii) Algunas entradas son consideradas lo suficientemente básicas para no requerir definición. (INICIAL_APELLIDO_MATERNO, INICIAL_NOMBRE, MES_NACIMIENTO, etc.)

iii) No se indica en qué casos una entidad toma un valor y en qué casos otro valor, ya que al especificarse ese detalle en las miniespecificaciones, se introduciría redundancia (LETRA_NOMBRE).

ESPECIFICACION DE PROCESOS (MINIESPECIFICACION). El centro del trabajo del Análisis Estructurado consiste en la especificación de los procesos, puesto que los diagramas de flujo y el diccionario de datos, si bien son partes importantes del documento final, no efectúan una especificación propiamente dicha. Por lo tanto la fase de análisis no estará completa hasta que no se concluyan las miniespecificaciones.

El conjunto de especificaciones de procesos presenta las siguientes características dentro del Análisis Estructurado:

- Existe una miniespecificación por cada función primitiva en el nivel inferior del conjunto estructurado de diagramas de flujo.
- Cada miniespecificación describe la forma en que se transforman los datos de los flujos que entran a la función primitiva en los datos que salen de ella.
- Las miniespecificaciones indican la idea de la transformación y no un método para llevarla a cabo.
- Al igual que los demás elementos del documento no debe existir redundancia en las miniespecificaciones, ni entre ellas ni tomando en cuenta los diagramas y el diccionario.

- **El método empleado en la miniespecificación debe ser ortogonal, en el sentido descrito con anterioridad.**

Para construir las miniespecificaciones, se cuenta con tres herramientas principales: el Lenguaje Natural Estructurado, las Tablas de Decisión y los Arboles de decisión.

Lenguaje Natural Estructurado. Para emplear este método de especificación de procesos se elige un subconjunto del lenguaje natural suficientemente limitado para eliminar las ambigüedades propias del idioma. De igual manera la sintaxis se reduce estableciendo reglas estrictas de construcción de oraciones.

En general se utilizarán únicamente los siguientes vocablos:

- Verbos en forma imperativa o infinitiva.
- Términos contenidos en el Diccionario de Datos. Los nombres de los objetos serán los asignados a las diferentes entidades y los calificadores serán los posibles valores que dichas entidades pueden asumir.
- Palabras reservadas para la construcción de expresiones lógicas, tales como *mientras, si, y, o,* etc.

En lo referente a la estructura del lenguaje (sintaxis) se utilizarán exclusivamente las siguientes construcciones:

- Construcciones Secuenciales. Se componen de una o más acciones que se ejecutan una tras de otra sin interrupción. Se forman con sentencias que contienen verbos imperativos.
- Construcciones de Repetición. Es un bloque de acciones que se efectúa una y otra vez hasta que se cumpla una condición determinada. Están constituidas por una sentencia condicional que controla la repetición de las acciones subordinadas.

- **Construcciones de decisión. Uno o más bloques de acciones de los cuales solo uno se realiza en una situación determinada. Están compuestas por una sentencia condicional seguida por el bloque de acciones que se ejecutan si la condición es verdadera, opcionalmente se añaden una expresión de separación tal como *sino* o *en otro caso* y el bloque de acciones a efectuar si la condición es falsa.**

Para que las construcciones anteriores conformen un proceso bien estructurado se requiere que cualquier combinación de ellas comience en un solo punto e igualmente termine en un sólo punto, pues esto garantizará que el comportamiento del proceso es siempre predecible.

Independientemente de lo restringido que resulte el subconjunto del lenguaje seleccionado, el lenguaje natural estructurado debe conservar dos características vitales: debe ser lo suficientemente formal para eliminar cualquier ambigüedad posible y debe ser lo suficientemente informal para no resultar incomprensible al usuario, pues en esta temprana etapa del desarrollo la opinión del usuario es fundamental y dicha opinión no podría ser formulada si no se entiende a fondo el contenido de la Especificación Formal.

Las tablas de decisión, son herramientas útiles para especificar procesos en donde la(s) acción(es) a ejecutar depende(n) de combinaciones de condiciones. Facilitan la visualización del proceso al existir varias alternativas que resultaría complejo describir textualmente.

Finalmente los árboles de decisión son la representación gráfica de las tablas de decisión y auxilian al analista de manera similar. Las tres herramientas pueden mezclarse dentro de un mismo documento, empleando en cada caso la que resulte más conveniente.

2.4.3 CONSTRUCCION DE LA ESPECIFICACION FORMAL.

Como se mencionó anteriormente, la metodología de De Marco contempla siete pasos para la construcción de la especificación. A continuación se revisan estos siete pasos:

i) Construcción del modelo físico actual. A través de diagramas de flujo se representa la manera en que las cosas se llevan a cabo en la actualidad. En esta primera etapa se contempla información de tipo físico con el fin de establecer un paralelo entre el modelo y la realidad. Entre la información que suele incluirse para que el usuario ubique el modelo, se encuentran los nombres de oficinas, localización, nombres de personas, archivos y dispositivos de almacenamiento, equipo y método para realizar el proceso.

ii) Se elimina la información física para convertir el modelo en un modelo lógico. Para esto deben detectarse las características físicas asociadas con los procesos, los flujos de datos y los archivos.

La información física contenida en los procesos puede ser eliminada sustituyendo en el diagrama de flujo el círculo que representa al proceso con el diagrama asociado de nivel inmediato inferior, hasta llegar a un nivel de detalle en el que se haya eliminado toda referencia física.

Para construir una estructura lógica de los archivos, se utiliza el proceso denominado NORMALIZACION DE TERCERA FORMA. Este proceso no será detallado en el presente trabajo dado que su magnitud requeriría un capítulo aparte, basta decir que consiste a grandes rasgos en registrar todas las referencias a los datos contenidos en los diferentes flujos, sustituirlas por el conjunto mínimo de datos necesario para cumplir las necesidades del proceso y finalmente agruparlos en archivos de acuerdo a sus características lógicas.

En los flujos de datos se eliminan los conceptos referentes a cuestiones físicas, como son reportes y formas, reemplazándose con los datos que los conforman.

Hasta aquí se ha utilizado exclusivamente información proporcionada por el usuario, por lo que el modelo del sistema refleja únicamente el estado actual de las cosas con nada o casi nada de trabajo creativo.

iii) Construcción del nuevo modelo lógico. Este resulta el paso central de la construcción de modelos, ya que incorpora las nuevas características solicitadas para el sistema como pueden ser automatización de procesos manuales, mejoras a procesos actuales, establecimiento de nuevos procesos, etc. En este momento el analista emplea al máximo su capacidad y experiencia.

Dentro de este punto destaca el establecimiento del dominio de cambio dentro del marco de referencia, es decir las modificaciones a realizar deberán estar contenidas completamente en el área ya estudiada, lo que garantiza que las interfaces son completamente conocidas.

El usuario juega nuevamente un papel importante en el desarrollo de este modelo pero ahora ya no es visto sólo como fuente de información, sino que pasa a asumir un papel de inspector del trabajo, cuya opinión tiene que ser tomada en cuenta y su visto bueno debe conseguirse para continuar la construcción del sistema. Esto origina que este paso se repita una y otra vez, lo que evita problemas a largo plazo.

iv) Se establecen modelos físicos. Se agregan al modelo pequeñas indicaciones físicas entre las que destaca la interfaz hombre-máquina, que delimita la parte a automatizar del sistema. No hay que olvidar que el sistema está compuesto tanto por las partes automatizadas como por las manuales, si diferentes modelos presentan diferentes rangos de automatización se obtendrán diversos modelos físicos a considerar.

v) y vi) Se realiza un análisis de costo-beneficio para determinar hasta dónde es deseable automatizar el sistema, eligiéndose el modelo correspondiente.

vii) Se unen las diferentes partes de la especificación en un todo que representará finalmente el sistema que el analista propone desarrollar y que el usuario ha aceptado como la solución más adecuada a sus necesidades.

2.5 DISEÑO ESTRUCTURADO DE YOURDON.

Los conceptos manejados por Yourdon en su metodología de Diseño Estructurado se ampliarán a continuación por considerarse necesario para el desarrollo de los capítulos siguientes.

2.5.1 OBJETIVOS.

Como ya se dijo, el Diseño Estructurado es el proceso de diseñar los componentes de un sistema y las interrelaciones entre esos componentes de la mejor manera posible.

Esta corriente busca elaborar un diseño que tenga las siguientes características:

- Que haga uso eficiente de los recursos disponibles.
- Que sea confiable, lo que se traducirá en un sistema de fallas mínimas.
- Que facilite el mantenimiento cuando se presenten fallas.
- Que permita modificaciones al variar las necesidades.

- Que sea flexible, permitiendo variaciones al mismo tema.
- Que sea de utilidad y de fácil uso por parte de los interesados.

Para el diseño de componentes, Yourdon basa su trabajo en el hecho confirmado por la experiencia, de que la implantación, mantenimiento y modificación de un sistema se facilitan cuando sus partes corresponden a partes bien definidas del problema y las relaciones entre componentes representan siempre relaciones en el problema real.

La descomposición en módulos se justifica mediante la llamada Ley Fundamental de la Ingeniería de Sistemas:

Dados dos módulos P y Q, una función M para medir su tamaño y una función C para medir el costo de su desarrollo se tiene que:

$$M(P + Q) > M(P) + M(Q) \text{ y por lo tanto}$$
$$C(P + Q) > C(P) + C(Q)$$

De lo anterior se desprende que al descomponer un sistema en módulos progresivamente más pequeños, el costo del desarrollo disminuye. Sin embargo este proceso tiene un límite, pues si los módulos se reducen en exceso, el costo originado por comunicación entre ellos eleva demasiado el costo total del sistema.

2.5.2 METODOLOGIA.

El producto principal de la metodología de Yourdon es la Gráfica de Estructura del sistema, que como su nombre indica, representa la manera en que los módulos están organizados dentro de él.

Las gráficas de estructura son árboles o diagramas jerárquicos que definen la estructura general de un sistema o programa mostrando los módulos que lo integran y las relaciones entre ellos.

Los módulos que realizan tareas de alto nivel se encuentran en los niveles superiores y los que realizan tareas de detalle aparecen en los niveles inferiores.

La representación gráfica de un módulo es un rectángulo en el cual se escribe un nombre representativo de la parte del problema real que representa. Los diferentes módulos conforman una estructura de control al colocarse en diferentes niveles entre los que existen relaciones representadas con flechas. Estas relaciones implican que el control se transfiere de un módulo a otro en la dirección de las flechas.

Algunas características relevantes de las relaciones intermódulo son:

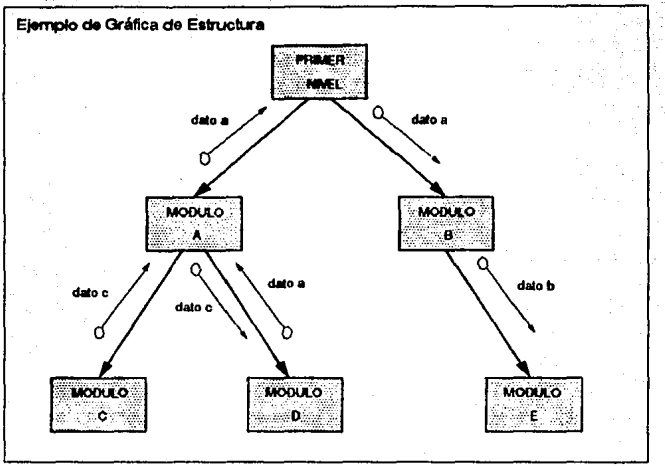
- Sólo puede haber una relación entre dos módulos dados.
- Cuando un módulo finaliza su labor, retorna el control al que hizo la llamada.
- Varios módulos pueden llamar a un sólo módulo de nivel inferior.
- Solo se permite la existencia de un módulo en el primer nivel de la jerarquía.

Al mismo tiempo que el control, se efectúa la transferencia de datos, que puede presentarse en cualquier dirección. Esta transferencia se señala con una pequeña flecha adjunta a la que indica la relación entre los módulos, acompañada del nombre de la información trasladada.

Dentro de estas gráficas se definen cuatro tipos de módulos:

i) Módulos aferentes. Toman información de sus subordinados y la trasladan hacia sus superiores. Los flujos así formados son conocidos como flujos aferentes.

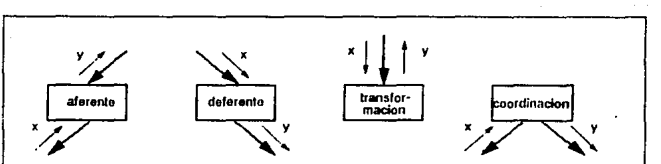
ii) Módulos deferentes. Reciben la información de sus superiores y la trasladan a sus subordinados, formando los llamados flujos



deferentes.

iii) Módulos de transformación. Su objetivo principal es transformar la información que reciben de sus superiores y devolverla una vez transformada.

iv) Módulos coordinadores. Administran y coordinan la ejecución de los demás módulos.



Para generar la Gráfica de Estructura, debe partirse de un modelo del sistema efectuado a través de diagramas de flujo de datos

tal como se describieron en la sección anterior al describir el Análisis Estructurado.

En estos diagramas, generalmente es posible apreciar la existencia de una o más transformaciones centrales que crean las principales salidas del sistema a partir de entradas preparadas previamente. Esta característica da origen al **Análisis de Transformaciones**, cuyas fases son las que se listan a continuación:

Identificación de elementos aferentes y deferentes.

Factorización de alto nivel.

Factorización de flujos aferentes, deferentes y de transformación.

Identificación de flujos aferentes y deferentes. Los elementos aferentes son aquellos que pueden considerarse aun como entradas al sistema, pero que han sido despojados de todas sus características físicas y constituyen una entidad lógica. Por el contrario, los elementos deferentes son las salidas del sistema antes de poseer información física.

Para su identificación se comienza por las fronteras del sistema, adentrándose cada vez más en él hasta localizar un flujo que ya no pueda ser considerado como de entrada o salida.

Una vez identificados los flujos aferentes y deferentes, quedarán entre ambos un conjunto de transformaciones que serán designadas transformaciones centrales por ser las que realizan el trabajo principal del sistema.

Factorización de alto nivel. En la gráfica de estructura se asigna un módulo coordinador que efectuará toda la tarea del sistema a través de llamadas a módulos subordinados. Para cada elemento identificado en los diagramas de flujo, se crea en la gráfica un módulo aferente, deferente o de transformación según corresponda, subordinado al módulo coordinador principal.

Factorización de flujos. Para factorizar un módulo aferente, se localiza la transformación que produce la información requerida por las transformaciones centrales y se le asigna un módulo de transformación subordinado al módulo aferente de primer nivel. Para cada una de las entradas requeridas por este nuevo módulo se crea un submódulo aferente. que será después refinado mediante el mismo proceso. Esta misma secuencia se sigue con razonamientos inversos para la factorización de módulos deferentes.

El caso de los módulos de transformación es tratado de manera diferente. descomponiendo el módulo en transformaciones menores que unidas produzcan la transformación original.

Otra técnica auxiliar en la conversión de los Diagramas de Flujo a la Gráfica de Estructura es el **Análisis de Transacciones.** Esta técnica es sugerida por aquellas transformaciones que convierten un flujo de datos entrante en varias salidas discretas.

En su forma general, un módulo de transacción puede modelarse como un sistema de cuatro niveles:

a) Procesador de transacciones. Recibe las señales del módulo superior y controla la ejecución de la transacción correspondiente.

b) Nivel de transacciones. Está compuesto por las diferentes transacciones existentes.

c) Nivel de acciones. Submódulos cuya función es realizar las acciones necesarias para cada transacción.

d) Nivel de detalle. Contienen los pasos detallados para las acciones del nivel inmediato superior.

2.5.3 EVALUACION DEL MODELO.

Yourdon propone la evaluación del diseño efectuado, a través de la presencia de dos importantes propiedades: el acoplamiento y la cohesión. Dado que la segunda es deseable y la primera no, un diseño será considerado bueno si posee un alto grado de cohesión y un bajo índice de acoplamiento.

Acoplamiento. Puede decirse que dos módulos son independientes si cada uno puede funcionar correctamente sin la presencia del otro. Mientras más deba saberse acerca de un módulo B para ejecutar el módulo A, más conectados estarán entre sí. El acoplamiento es una medida de la fuerza de interconexión entre dos módulos.

Existen cuatro factores que pueden hacer variar el grado de acoplamiento total de un sistema:

a) Tipo de la conexión. Si todas las conexiones de un sistema se restringen a transferencias de control completamente parametrizadas hacia el encabezado o interfaz de activación de los módulos, entonces el sistema está conectado de manera mínima. Si por el contrario, en algunos módulos existen referencias a elementos internos de otros módulos (conexiones patológicas), entonces el acoplamiento del sistema se incrementa considerablemente.

b) Complejidad de las interfaces. Se considera que entre más compleja es la interfaz o interconexión entre dos módulos, más acoplados se hallan éstos, puesto que para entender qué parámetros deben utilizarse, se requiere conocer más detalles internos del módulo que será activado.

c) Tipo información fluyendo en la conexión. Cuando la información que va de un módulo a otro está constituída exclusivamente por datos, el acoplamiento es menor que cuando se compone de datos de control.

d) Instante de acoplamiento. Dos módulos pueden acoplarse a tiempo de ejecución, de carga, de ligado o de compilación. Se considera que el acoplamiento es menos intenso cuando es más tardío.

Cohesión. Esta es una propiedad que establece qué tan relacionados están los componentes de un mismo módulo. Al integrarse las partes relacionadas entre sí en una sola unidad, disminuyen las relaciones entre diferentes unidades, por lo que mientras más cohesión presente un sistema, más se reducirá el acoplamiento.

Existen diferentes tipos de cohesión, los cuales se listan a continuación en orden creciente de importancia:

- Coincidental.
- Lógica.
- Temporal.
- De procedimiento.
- Comunicacional.
- Secuencial.
- Funcional.

Cohesión coincidental. Se presenta cuando no hay ninguna relación aparente entre elementos de un módulo, es decir su reunión parece ser casual.

Cohesión lógica. Se presenta en aquellos módulos cuyas componentes pertenecen a la misma clase de funciones por lo que a primera vista parece lógico colocarlas juntas.

Cohesión temporal. Implica que todas las ocurrencias de todos los elementos de proceso agrupados en un módulo se producen dentro del mismo período de tiempo durante la ejecución.

Cohesión de procedimiento. Los elementos asociados bajo este principio se localizan en la misma unidad de procedimiento, por ejemplo una iteración o una condición.

Cohesión comunicacional. Los elementos asociados comunicacionalmente actúan sobre los mismos datos de entrada o bien producen las mismas salidas.

Cohesión secuencial. En este caso, los datos obtenidos por la ejecución de una componente sirven como entrada al siguiente elemento de proceso.

Cohesión funcional. Este es el tipo ideal de asociación, ya que en ella todos y cada uno de los elementos de proceso agrupados, forman parte integral y son esenciales para la realización de una sola función.

El grado de cohesión total de un módulo es el grado más alto que puede aplicarse a todas sus componentes.

2.5.4 IMPLANTACION DEL DISEÑO.

El punto final de la metodología de Diseño Estructurado, es la construcción de unidades físicas a partir de los módulos ya definidos. El objetivo de estas unidades es el de permitir la ejecución del sistema en el equipo de cómputo establecido.

La técnica utilizada para establecer esta agrupación, es conocida como Análisis de Procedimientos (*Procedural Analysis*), y su política general consiste en incluir dentro de la misma unidad los módulos conectados por referencias utilizadas frecuentemente.

Dicha frecuencia puede ser estimada en tres formas principales: Analizando las iteraciones, para incluir en la misma unidad

al módulo superior y al módulo subordinado cuando la referencia se efectúa dentro de una iteración; estimando el volumen de referencias entre dos módulos, incluyéndolos en la misma unidad cuando el volumen es considerable; analizando las proposiciones condicionales para valorar la frecuencia con que las transferencias de control entre módulos se producen o no.

Por el contrario, algunos módulos ameritan ubicarse en unidades separadas, tal es el caso de funciones opcionales, que en ocasiones no llegan a activarse durante sesiones completas del sistema, así como módulos que se utilizan una sola vez en la ejecución del sistema, y que una vez que terminan su labor no son requeridos en la memoria.

Es importante señalar que la tarea de agrupar módulos en unidades debe realizarse cuando el diseño finaliza, pues así se asegura que las decisiones de diseño estuvieron tan libres como fue posible de puntos de vista impuestos por el nivel físico.

BIBLIOGRAFIA DEL CAPITULO

- James Martin, Carma McClure, "Structured Techniques the Basis for CASE". Prentice Hall, 1988.

- Carma McClure, "CASE is Software Automation", Prentice Hall, 1989.

- Tom DeMarco, "Structured Analysis and System Specification", Yourdon, Inc., 1979

- Edward Yourdon, Larry L. Constantine, "Structured Design", Prentice Hall, 1979.

Capítulo 3

TECNOLOGIA CASE.

3.1 ORIGENES.

La palabra CASE corresponde a las siglas de la expresión inglesa *Computer Aided Software Engineering*, y se utiliza para designar la tecnología por medio de la cual se automatizan las metodologías estructuradas de desarrollo de sistemas, desde el paso de planeación hasta el de mantenimiento, poniendo especial énfasis en el análisis y el diseño.

Esto significa que no se trata de una nueva disciplina, sino que se basa en la combinación de las dos tecnologías tradicionales existentes: las técnicas o metodologías estructuradas y las herramientas computarizadas. Las diferentes metodologías estructuradas presentan la ventaja de ser ampliamente utilizadas y de tener una eficiencia reconocida, sin embargo en la mayor parte de los casos contienen procesos laboriosos o tediosos, como la creación y actualización de distintos tipos de diagramas o la especificación de reportes, que las hacen ser subutilizadas. Es en estos procesos donde las herramientas computarizadas entran en acción inyectando nueva vida a la Ingeniería de Sistemas.

Los objetivos de la tecnología CASE buscan satisfacer las diferentes necesidades de los equipos de desarrollo de software, y entre ellos pueden mencionarse los siguientes:

- Fomentar la aplicación de la Ingeniería de Sistemas.
- Hacer más prácticas la metodologías estructuradas.
- Hacer más eficiente el consumo de recursos.
- Incrementar la calidad de los sistemas.
- Automatizar la administración de proyectos.
- Liberar al analista de las tareas menos relevantes.
- Facilitar la reutilización de componentes.
- Incrementar la portabilidad de los sistemas.
- Mejorar el desarrollo de sistemas.

De estos puntos, el último es el más extenso, debido a que la tecnología CASE pretende realizar gran variedad de mejoras a las características del desarrollo de sistemas, entre las cuales destacan las siguientes:

- Estandarizarlo y formalizarlo.
- Maximizar su productividad.
- Simplificar sus diferentes etapas.
- Incrementar su confiabilidad.
- Promover un mayor control en sus partes.
- Acelerar su terminación.
- Reducir sus costos.
- Automatizar la verificación de errores.
- Acelerar la implantación de prototipos.
- Estandarizar la documentación.
- Generar automáticamente la documentación.
- Generar el código de manera automática.
- Mejorar la interfaz hombre-máquina.
- Facilitar el mantenimiento.

Si la tecnología CASE ayuda a resolver estas necesidades entonces es clara la importancia que debe tener para las organizaciones en donde se desarrollan sistemas computarizados y en particular para los analistas encargados de esa tarea.

En la búsqueda del cumplimiento de estos objetivos debe destacarse una vez más el hecho de que las herramientas computarizadas, que llamaremos herramientas CASE, son únicamente una parte de la tecnología CASE cuya base teórica son las metodologías estructuradas. El analista que no cuente con una formación adecuada estará casi tan limitado al utilizar herramientas CASE como lo estaría de no utilizarlas y no puede esperarse por el simple hecho de contar con ellas, que empezará a producir sistemas de calidad superior, en el mejor de los casos, producirá más rápidamente los mismos sistemas deficientes que acostumbra.

Por lo tanto, antes de intentar utilizar una herramienta CASE, es primordial la comprensión a fondo de una o varias metodologías estructuradas, teniendo en cuenta que el impacto principal al introducir la tecnología CASE en una organización no es tanto tecnológico como psicológico, ya que se comprometerá a los analistas a seguir un método en su trabajo y a olvidarse de la improvisación y la inconsistencia.

Algunos ejemplos de herramientas CASE utilizadas en las diferentes etapas del desarrollo de sistemas son las siguientes:

- Diagramadores para producir diagramas estructurados y especificaciones gráficas.
- Creadores de pantallas y reportes para realizar prototipos.
- Diccionarios y manejadores de bases de datos para fácil acceso a sistemas de información.
- Generadores de código.
- Generadores de documentación.

Como es lógico suponer, estas herramientas no aparecieron simultáneamente, sino que han seguido un proceso de evolución producto del cambio en las necesidades más apremiantes de los analistas y programadores. De esta manera es posible trazar una breve historia en el surgimiento de las herramientas CASE de acuerdo al cuadro siguiente:

A	B	C	D
PRINCIPIOS DE LOS 80	MEDIADOS DE LOS 80	FINES DE LOS 80	PRINCIPIOS DE LOS 90
DIAGRAMADORES Y DOCUMENTADORES	VERIFICACION DE DIAGRAMAS	INTEGRACION DISEÑO - CODIGO	REUTILIZACION DE MODULOS

A) Hacia principios de los años 80 aparecen en el mercado documentadores y diagramadores. Los primeros analizan los programas fuente de un sistema para producir comentarios generales de identificación; información relativa a los archivos utilizados: número, tipo y aparición de variables empleadas; estructura jerárquica de procedimientos y funciones. Los diagramadores crean diagramas de flujo de datos, diagramas de entidad-relación, gráficas de estructura y otros. Diferentes herramientas manejan diferentes metodologías, entre ellas algunas de las más utilizadas son: Jackson, Yourdon y Warnier. En esta etapa las herramientas CASE se enfocan a la documentación y a estrechar la comunicación entre grupos de trabajo para incrementar la productividad.

B) A mediados de la década de los 80 se introdujo la verificación automática de diagramas estructurados de acuerdo a las reglas establecidas por cada metodología, así como su almacenamiento en enciclopedias. que son repositorios de información en donde se almacena no sólo un diccionario de datos común que señale los atributos de los datos empleados, sino también información más significativa como relaciones entre entidades, diagramas y su significado, historia de modificaciones, etc. En este lapso, la preocupación se centró en el análisis y el diseño para mejorar la calidad de los sistemas.

C) Se integra la automatización del diseño con la generación automática del código, al hacerlo se realza aún más el papel de la enciclopedia, que es el medio por el cual la información puede fluir desde los diagramas de especificación de sistemas hasta los programas.

D) A principios de la década de los 90 la reutilización aparece como filosofía de trabajo y no se refiere exclusivamente al código sino a los modelos de datos, prototipos y especificaciones.

Paralelamente a la tecnología CASE, originada en los Estados Unidos, ha surgido en Europa un concepto similar llamado *Integrated Project Support Environment (IPSE)*, el cual se concentra en la infraestructura de las organizaciones de la que forman parte la administración de proyectos, la comunicación dentro de los equipos de trabajo y las funciones de los departamentos. Los problemas de calidad y productividad asociados al desarrollo de sistemas, se atacan en IPSE tratando de optimizar la comunicación entre los grupos participantes.

La tecnología IPSE supone que todos los participantes en el proceso de desarrollo de un sistema tienen acceso a un medio ambiente automatizado. La clave consiste en asegurarse de que la totalidad de los miembros, desde el administrador del proyecto hasta los analistas y programadores estén ligados a la misma estructura de comunicación.

3.2 CATEGORIAS DE HERRAMIENTAS CASE.

Herramientas individuales. Los primeros pasos en la tecnología CASE consistieron en herramientas utilizadas por un solo usuario en una sola fase del proceso de desarrollo de sistemas. Estas herramientas, que actualmente continúan utilizándose, cumplen diferentes funciones pero permanecen aisladas unas de otras, proporcionando únicamente pequeñas mejoras en la calidad y productividad del desarrollo de sistemas.

En general las herramientas individuales pertenecen a alguno de los siguientes grupos:

- Herramientas de diagramación para representar gráficamente modelos de la realidad.
- Generadores de pantallas y reportes para presentar la interfaz con el usuario.
- Generadores de código para producir programas ejecutables a partir de especificaciones gráficas.
- Generadores de documentación para programas ya existentes.
- Diccionarios y sistemas de manejo de datos para almacenar la información técnica del proyecto.

Estructuras CASE (*Frameworks*). La necesidad de integración de herramientas CASE ha sido resuelta hasta ahora de diferentes maneras. Una de ellas ha sido a través de estructuras que proporcionan un ambiente en el cual es posible integrar herramientas que tradicionalmente trabajan por separado. Las facilidades que suelen proporcionar son principalmente una interfaz común con el usuario, un almacén central de información y mecanismos para la comunicación entre herramientas.

Las estructuras CASE más simples y a la vez más utilizadas son las que han surgido de alianzas entre diferentes fabricantes, su función principal es trasladar datos entre los almacenes de información de las herramientas involucradas. Los métodos utilizados para tal fin son las opciones de importar y exportar datos, o bien, la traducción de archivos de diferentes formatos.

En resumen, las estructuras CASE no realizan ninguna función determinada para el desarrollo de sistemas, sino que su objetivo principal es crear en el usuario la impresión de que las herramientas que utiliza comparten la misma interfaz, el mismo formato de archivos y que pueden utilizarse en el mismo ambiente de hardware.

Conjuntos integrados de herramientas CASE (*Toolkits*). Cuando un solo producto agrupa una serie de herramientas pertenecientes a un solo fabricante, el resultado es un paquete mucho más conexo que multiplica la utilidad de cada herramienta si se tomara por separado. Con estos conjuntos de herramientas es posible automatizar toda una fase del ciclo de vida de un sistema, las facilidades que proporcionan varían dependiendo de a cuál de esas fases se enfocan. A continuación se mencionan sus principales subcategorías:

Conjuntos integrados para análisis. Su objetivo es automatizar la creación de una especificación del sistema, que contenga la definición de pantallas y reportes, la estructura de los datos y la definición de componentes funcionales. Para cumplir con este objetivo, estos conjuntos deben contar con herramientas de los siguientes cuatro tipos:

- Herramientas de diagramación estructurada que permitan trazar, manipular y almacenar diagramas estructurados. Como mínimo, diagramas de flujo de datos y diagramas entidad-relación.
- Herramientas para crear prototipos que muestren al usuario las pantallas, reportes y menús que incluirá el sistema.
- Repositorios de información para almacenar todos los datos concernientes a la especificación del sistema.
- Verificadores de especificaciones que cotejen la integridad y consistencia de los diagramas.

Conjuntos integrados para diseño. Soportan tanto el diseño lógico como el físico de la base de datos y de los procesos. Deben proporcionar diagramas de árboles jerárquicos y diagramas de lógica de procedimientos.

Conjuntos integrados para programación. Tienen como propósito producir programas documentados y probados que cumplan con la especificación de requerimientos. Las herramientas deseables para estos conjuntos son las siguientes:

- Diagramas de árboles jerárquicos.
- Diagramas de lógica de procedimientos.
- Repositorio de información.
- Generador de código fuente.
- Analizadores de código.
- Manipuladores de archivos.
- Generadores y analizadores de cobertura de pruebas.
- Localizadores de errores.

Conjuntos integrados para mantenimiento. Tienen por objeto corregir, modificar, ampliar, analizar o efectuar ingeniería de sistemas a la inversa[1] sobre sistemas existentes para extender su período de vida útil. Presentan las siguientes facilidades:

Para evaluación de programas:

- Herramientas para control de proyectos.
- Monitores de cobertura de pruebas.
- Monitores de desempeño.
- Comparadores de archivos.

Para análisis e Ingeniería de Sistemas Inversa:

- Analizadores de programas para producir referencias cruzadas.
- Depuradores de errores.
- Documentadores y reformateadores.

[1] La Ingeniería de Sistemas a la Inversa consiste en partir de los programas fuente para obtener sus especificaciones y modelos lógicos. Es utilizada en aquellas organizaciones que buscan estandarizar el desarrollo de sus sistemas, incluyendo a los ya existentes para los cuales no se hayan observado íntegramente las diferentes fases del ciclo de vida.

- Herramientas para aplicar a la inversa la Ingeniería de Sistemas, llevando un programa a su especificación.
- Convertidores para trasladar programas a nuevos ambientes.

Ambientes integrados CASE (*Workbench*). Esta es la forma más ambiciosa de integración de herramientas CASE, ya que pretende automatizar por completo el desarrollo y mantenimiento de sistemas así como la administración de proyectos. La salida de un módulo que se enfoque a una parte específica del desarrollo es proporcionada automáticamente como entrada al siguiente módulo. El producto final es el sistema computarizado y su documentación.

Los diferentes ambientes CASE trabajan en diferentes plataformas de hardware, diferentes sistemas operativos y soportan diferentes metodologías estructuradas, así como diferentes tipos de sistemas a desarrollar. Estas características llevan a una clasificación más detallada de los ambientes integrados CASE:

PLATAFORMA DE HARDWARE:	METODOLOGIA SOPORTADA	TIPO DE APLICACION
Mainframe	Ingeniería de sistemas	Sistemas de información
Mini	Ingeniería de información	Sistemas de tiempo real
Pc/Workstation		

Las características principales que debe presentar un Ambiente Integrado CASE son las que se listan a continuación:

- Cobertura del ciclo de vida
- Generación automática de código.
- Soporte de metodologías
- Verificación de errores

- Repositorio de información
- Integración de herramientas
- Alta capacidad gráfica
- Soporte de prototipos

Aunque todas estas herramientas han sido tratadas con más o menos detalle en este capítulo, dada su trascendencia para este trabajo se comentarán por separado en la siguiente sección.

Una clasificación alternativa de las herramientas CASE, que también está ampliamente difundida en la bibliografía del tema, considera tres grupos principales dependiendo del tipo de tarea que efectúen: herramientas CASE de alto nivel (*Upper CASE*), herramientas CASE de nivel medio (*Middle CASE*) y herramientas CASE de bajo nivel (*Lower CASE*).

Herramientas CASE de alto nivel (*Upper CASE*). Se concentran en el estudio del medio ambiente en que deben operar los sistemas y automatizan la planeación de los mismos.

Herramientas CASE de nivel medio (*Middle CASE*). Son aquellas que se emplean para automatizar las fases de análisis y diseño del ciclo de vida de un sistema. Su papel es definir qué debe hacer el sistema sin detallar cómo hacerlo. Para esto incluyen diagramas de flujo de datos y de entidad-relación en la etapa de análisis así como definición de los elementos del sistema en diccionarios de datos en la etapa de diseño.

Herramientas CASE de bajo nivel (*Lower CASE*). Facilitan la automatización de las actividades de diseño físico e implementación del sistema. Son las herramientas que producen realmente los programas verificados y los ponen en funcionamiento.

3.3 COMPONENTES DE UN AMBIENTE CASE.

Para que un ambiente integrado CASE explote al máximo la capacidad de la computadora, proporcionando al analista un mayor número de facilidades para la automatización de su labor, deben estar presentes las componentes que se ilustran en la figura 3.3 y que se discuten a continuación:

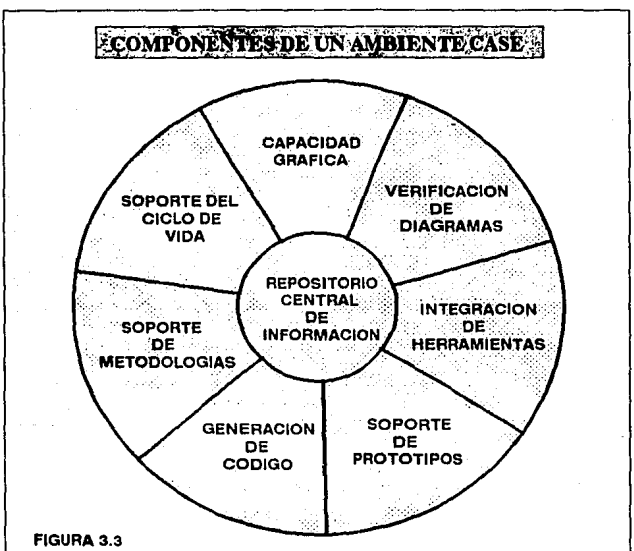


FIGURA 3.3

Alta capacidad gráfica. Las primeras etapas del desarrollo de un sistema, están dominadas por actividades que involucran a personas con diferentes formaciones que necesitan comunicarse con claridad. Una alternativa que ha probado ser adecuada para facilitar esta comunicación es el empleo del lenguaje gráfico.

El lenguaje gráfico es más rico que el textual, ya que en una gráfica o diagrama puede representarse gran cantidad de información cuyo significado es asimilado por el receptor más rápidamente.

En el desarrollo de la Ingeniería de Sistemas, los diagramas han desempeñado un papel muy importante al ser utilizados como herramientas de comunicación o para aclarar ideas. Su utilidad se basa en el hecho de que un diagrama es a la vez menos ambiguo y más comprensible que un texto.

Puede considerarse que los diagramas básicos que conforman el conjunto mínimo de diagramación que debe contener un ambiente CASE son:

a) Diagramas de flujo de datos para análisis.

b) Diagramas de modelos de datos para diseño de bases de datos.

c) Gráficas de estructura para diseño de programas.

Adicionalmente, dependiendo de las aplicaciones a desarrollar, pueden requerirse los siguientes diagramas:

a) Para análisis: Diagramas de flujo de control, tablas de decisión, diagramas de composición de procedimientos.

b) Para diseño: Diagramas Warnier-Orr, diagramas de transición de estados, diagramas para procesos o pseudocódigo.

c) Para diseño de bases de datos: Diagramas entidad-relación, diagramas de estructuras de datos.

Verificación de diagramas. Cada uno de los diagramas mencionados anteriormente posee sus propias reglas de construcción y sintaxis, las cuales deben ser observadas para obtener resultados adecuados. La verificación de diagramas es la herramienta que permite que, dentro del ambiente CASE, los diagramas estructurados sean algo más

que simples dibujos utilizados como documentación, y se conviertan en el punto de partida de todos los demás procesos.

Los tipos de verificación que debe proporcionar un ambiente CASE son:

- Verificación de sintaxis y tipos, que asegura que los símbolos gráficos están siendo utilizados según la función que la metodología les define.

- Verificación de consistencia e integridad, que revisa si todos los objetos referenciados han sido definidos previamente.

- Verificación de composición funcional, que mide la complejidad de los procesos.

- Verificación de consistencia entre niveles, que relaciona diferentes diagramas que se refieren a un mismo aspecto del sistema, localizando posibles inconsistencias entre ellos.

- Verificación de satisfacción de requerimientos, que confronta una especificación del sistema contra los diseños propuestos para asegurar su concordancia.

Almacenamiento de información. El repositorio de información es el centro del ambiente CASE alrededor del cual se agrupan las demás componentes. Este dispositivo de almacenamiento es mucho más que un simple diccionario, ya que guarda más tipos de información que únicamente datos. Por ejemplo, es importante que la herramienta CASE almacene no sólo los diagramas sino el significado de los símbolos gráficos así como su información completa y su relación lógica con otros símbolos.

A través del repositorio del ambiente CASE puede obtenerse información como la siguiente:

- Descripción del flujo de datos.
- Lista de entradas y salidas para cada proceso.
- Lista de objetos actualizados a partir de una fecha dada.
- Historia de modificaciones de un objeto.
- Relación de los diagramas donde aparece un proceso.
- Relación de procesos llamados por un módulo dado.
- Personas responsables de cada entidad.
- Objetos no definidos en el sistema.
- Objetos definidos pero no utilizados.
- Impacto al modificar o eliminar una entidad.

Es por esta característica que el repositorio central es la base para actividades medulares de la Ingeniería de Sistemas automatizada como son las siguientes:

- Integración de Herramientas CASE
- Control de consistencia e integridad de la especificación.
- Traspaso de información entre los módulos.
- Generación de la documentación del sistema.
- Estandarización de documentación.
- Generación de código.
- Control y administración de proyectos.
- Reutilización de procesos y tipos de datos.

Integración de herramientas. Como ya se ha mencionado, la integración de herramientas es lo que distingue a los ambientes CASE de los demás tipos. Esta integración puede y debe producirse a diferentes niveles:

- Interfaz con el usuario común a todas las herramientas.

- Transferencia eficiente de datos entre herramientas.
- Integración de las fases de desarrollo a través de una representación del sistema almacenada por el repositorio.
- Programas auxiliares incorporados como procesadores de texto, ayudas de escritorio, etc.

Soporte del ciclo de vida. Las herramientas aisladas se enfocan a un proceso específico de alguna fase del ciclo de vida de un sistema. Por el contrario, el poderío de los ambientes integrados CASE radica en el hecho de que cuentan con suficientes herramientas para poder hacer un seguimiento de todo el proceso de construcción de un sistema.

Sin embargo, la práctica común es que los ambientes integrados CASE den mayor énfasis a las fases de análisis y diseño, debido a que reconocen la importancia de eliminar los errores desde el principio cuando aún no son tan costosos.

Soporte de prototipos. Los prototipos tienen como función proporcionar al usuario una idea de cómo se relacionará con el sistema, cuando éste se encuentra aún en una etapa temprana de su desarrollo. Con esto se consigue involucrar al mismo usuario en el análisis del sistema, de manera que cuando reciba el producto final no encuentre discrepancias entre lo que en realidad deseaba y necesitaba y lo que le fue proporcionado.

Las principales herramientas para la construcción de prototipos son:

- Constructores de pantallas y reportes.
- Creadores de menús.

Generación automática de código. Esta componente se encuentra en el lado opuesto del ciclo de vida de un sistema de donde

se encuentra la construcción de prototipos. Siendo los programas el objetivo final del equipo de desarrollo del sistema, lo ideal es que éstos se generen automáticamente como consecuencia de un proceso seguido ordenadamente.

Aún cuando los generadores de código están todavía restringidos a unos cuantos lenguajes y máquinas, la confiabilidad y eficiencia de los programas que producen, permiten suponer que en un futuro próximo estas herramientas serán extendidas para cubrir las necesidades de un mayor número de analistas y programadores.

Soporte de metodologías estructuradas. A lo largo de este capítulo se ha hecho especial énfasis en que la base teórica de la tecnología CASE está formada por las diferentes metodologías estructuradas que han dado vida a la Ingeniería de Sistemas, y que su estudio reviste gran importancia para la correcta utilización de las diferentes herramientas. La automatización de las metodologías se efectúa a dos niveles:

a) Preparación de documentación.

b) Seguimiento paso a paso de la metodología.

La documentación está integrada por los diferentes diagramas empleados por las metodologías, de manera que el conjunto de diagramas que ofrece una herramienta CASE define las metodologías que soporta.

El seguimiento paso a paso, auxilia al usuario en el correcto uso de la metodología. Esto implica la necesidad de entendimiento pleno de la metodología inmersa en la herramienta. El analista que emplea una herramienta CASE con este tipo de automatización de la metodología, es forzado a seguir los procesos en el orden establecido.

Este factor se considera muchas veces un inconveniente, pues el analista no puede elegir la metodología que le parezca más apropiada

para cada parte del desarrollo del sistema. Por otro lado esta aparente desventaja, se ve compensada porque al especializarse en una metodología en particular, las herramientas CASE cuentan con la posibilidad de efectuar análisis más elaborados.

3.4 AMBIENTE INTEGRADO IEW.

En esta sección se señalarán las características más relevantes del ambiente integrado CASE conocido con las siglas IEW (*Information Engineering Workbench*).

Como todos los ambientes integrados, IEW busca auxiliar al analista de sistemas en todas las fases del desarrollo, para lo cual cuenta con tres módulos integrados pero a la vez independientes llamados estaciones de trabajo (*Workstations*), estos módulos se enfocan a la planeación (*Planning Workstation*) , el análisis (*Analysis Workstation*) y el diseño (*Design Workstation*).

El módulo de Planeación contiene herramientas para representar los objetivos de la organización y coordinar los elementos del medio ambiente en que operará el sistema. El módulo de Análisis permite modelar los datos y procesos necesarios para integrar un sistema acorde con los requerimientos del usuario. El módulo de Diseño describe cómo se implantarán físicamente tales procesos y datos.

Además del **seguimiento del ciclo de vida** del sistema instrumentado a través de sus tres estaciones, IEW cuenta con la mayoría de las componentes descritas en la sección 3.3.

En lo referente a **capacidad gráfica**, proporciona diferentes tipos de diagramas dentro de cada estación o módulo, los cuales se detallarán a continuación:

Principales diagramas del Módulo de Planeación. Los dia-

gramas que proporciona para crear un modelo de la organización son: Diagramas de Descomposición, para describir las componentes de la organización en forma jerárquica, Diagramas de Entidad-Relación para representar objetos cuyos datos fluyen dentro de la organización y las relaciones entre ellos. y Diagramas de Matriz para representar asociaciones y propiedades comunes entre los objetos del sistema.

Principales diagramas del Módulo de Análisis. Para obtener un modelo de los datos y procesos que conformarán el sistema, este módulo contiene Diagramas de Flujo de Datos, Diagramas de Entidad-Relación, Diagramas de Descomposición y Diagramas de Acciones.

Los Diagramas de Flujo de Datos representan las actividades o procesos y los datos que fluyen entre ellos. Regularmente se utilizan diagramas que representan diferentes niveles de abstracción, comenzando desde la visión más general posible del sistema que se va partiendo hasta llegar al nivel de detalle más bajo.

Los Diagramas de Entidad-Relación representan las entidades relevantes para el sistema, junto con sus atributos y relaciones que constituyen la columna vertebral de la información manejada por el sistema. Los Diagramas de Descomposición se emplean para dividir sistemas complejos en subcomponentes, pudiendo aplicarse tanto a los datos como a los procesos. Los Diagramas de Acciones hacen uso de un lenguaje estructurado para describir textualmente los diferentes procesos de que se compone el sistema.

Principales diagramas del Módulo de diseño. Con el fin de convertir el modelo lógico, creado en la estación de análisis, en una descripción física detallada, se cuentan con los siguientes diagramas para procedimientos y datos: Gráficas de Estructura. Diagramas Modulares de Acciones, Diagramas de Estructura de Datos y Diagramas de Bases de Datos, entre otros.

Las Gráficas de Estructura representan la organización jerárquica de los módulos y programas que componen el sistema, mostrando

las llamadas a funciones o subrutinas y los parámetros que las conectan con el procedimiento principal.

Los Diagramas Modulares de Acciones explican detalladamente la lógica de los procedimientos en cada módulo definido en la gráfica de estructura.

Los Diagramas de Estructura de Datos describen la estructura de todos los datos utilizados en un módulo, utilizando tablas cuyas columnas contienen los atributos de cada entidad de datos física.

Los Diagramas de Bases de Datos describen la manera en que se implantan los archivos y las relaciones existentes entre los componentes de la base de datos.

Las diferentes herramientas para el procesamiento de diagramas, proporcionan facilidades de edición con el objeto de minimizar el tiempo empleado en su construcción y su modificación. Entre tales facilidades destacan, la reubicación de iconos, el escalamiento y las opciones para impresión.

Para **verificación de diagramas**, IEW incluye un elemento llamado *Knowledge Coordinator* que posee gran cantidad de reglas predefinidas e impide que el usuario incorpore información incompleta e inconsistente.

También se cuenta con una serie de reportes basados en la información proporcionada por el analista, que van más allá de ser simples listados de datos, y efectúan verificaciones de diferentes aspectos de la construcción de los sistemas.

Dentro del módulo de análisis existen los siguientes reportes que permiten diagnosticar errores potenciales dentro de los diagramas de flujo de datos:

- Reporte de análisis de conectividad (*Connectivity analysis report*).

Lista todos los flujos de datos inválidos.

- Reporte de conservación de datos. (*Data conservation report*). Señala aquellos diagramas que contienen objetos o flujos que no respetan la ley de conservación de los datos.

- Reporte de análisis de trayectorias de datos. (*Data Flow course analysis report*). Describe las inconsistencias de un flujo de datos entre los distintos niveles de abstracción de la especificación.

La eliminación de errores dentro del módulo de diseño, puede efectuarse basándose en los siguientes reportes:

- Reporte de análisis de llamadas (*Call analysis report*). Compara los parámetros dentro de un módulo con los parámetros proporcionados por los módulos de más alto nivel.

- Reporte de análisis de descripción de la base de datos. (*Database description analysis report*). Lista las descripciones de las bases de datos físicas que son utilizadas en los diagramas de descripción de datos lógicos.

- Reporte de análisis de especificación de programas (*Program specification analysis report*). Relaciona el contenido de los diagramas de descripción de la base de datos con los segmentos correspondientes de las especificaciones de programas.

- Reporte de segmentos no referenciados (*Unattached segment report*). Señala las entradas de la enciclopedia para los que no existe referencia en ningún diagrama de base de datos.

Para el **almacenamiento de información** se utiliza la enciclopedia, que reúne todos los elementos, diagramas y reportes producidos y proporciona información detallada de cada entidad en cada diagrama, utilizando para ello técnicas de inteligencia artificial.

Dada la importancia que el almacén de datos tiene para el ambiente integrado CASE, la enciclopedia de IEW está rodeada de funciones auxiliares que le permiten reducir la redundancia y eliminar inconsistencias entre objetos utilizados en diferentes etapas del desarrollo. Entre estas funciones auxiliares caben destacar: el control de seguridad mediante el cual puede restringirse la actualización de su contenido; el procedimiento para listar los objetos contenidos en la enciclopedia relacionados con etapas en particular y la capacidad de crear diferentes enciclopedias que sean utilizadas específicamente por un proyecto o grupo de trabajo, con la facultad de integrarlas posteriormente en una sola.

La integración de las herramientas se consigue a través de una interfaz de usuario común a todas las estaciones. Dicha interfaz permite el empleo de menús, diálogos, manejo del ratón y proporciona ayuda en línea acerca de las posibles acciones a realizar.

Por otra parte, los módulos pueden utilizarse secuencialmente, constituyendo la salida de cada uno la entrada del sucesor inmediato, incluyéndose en esta serie **generadores de código** a los que puede ligarse el módulo de diseño.

Para el **desarrollo de prototipos**, la estación de diseño cuenta con un diagramador para el trazado de pantallas, con el cual es posible diseñar las pantallas asociadas a cada módulo con el fin de que el usuario pueda apreciar el comportamiento esperado de las pantallas que integrarán la interfaz con el producto final.

Finalmente, IEW no es una excepción en su relación con las **metodologías estructuradas**, éstas son su fundamento teórico, por lo que el analista debe estar familiarizado con algunas de ellas para explotar al máximo su capacidad, principalmente con el Análisis Estructurado de Tom de Marco, el Diseño Estructurado de Yourdon y la Ingeniería de Información de James Martin.

Este último aspecto resalta la categoría de *herramienta* de

IEW, ya que para su utilización es necesaria una capacitación previa que abarque los diferentes aspectos de la Ingeniería de Sistemas.

Para concluir esta sección es oportuno señalar que la instalación de IEW puede llevarse a cabo en una computadora personal, que sin embargo deberá contar con importantes recursos de disco duro y memoria extendida y expandida, al igual que un ratón y una impresora con buena resolución.

BIBLIOGRAFIA DEL CAPITULO

- Carma McClure, "CASE is Software Automation", Prentice Hall, 1989.

- Carma McClure, "The CASE for Structured Development", PC Tech Journal, Agosto 1988, pp 51-67

- Warren Keuffel, "CASE for the rest of us", Computer Language. Enero 1991, pp 25-29

- John Burke, "Tough CASE", HP Professional, Julio 1991, pp 30-36

- Andrew Topper, "Building Up to IEW/WS", PC Tech Journal, Septiembre 1988, pp 110-123.

- KnowledgeWare, "Information Engineering Workbench/Workstation Basics Book", Release 5.0, Septiembre 1988

Capítulo 4

APLICACION DE CONCEPTOS.

En este capítulo se llevarán a la práctica los conceptos teóricos que se han presentando hasta el momento. Para tal fin se ha elegido un escenario real que se describirá brevemente en la primera parte, mencionando los problemas que han surgido en dicho escenario, estableciéndose la necesidad de diversos sistemas de cómputo. Finalmente se detallará el desarrollo de uno de estos sistemas, siguiendo conceptos de las metodologías revisadas en el capítulo dos y utilizando herramientas como las presentadas en el capítulo tres.

En particular, el desarrollo del sistema se llevará a cabo dentro del ambiente CASE de IEW y se incluirán únicamente reportes y diagramas producidos por sus tres estaciones de trabajo.

4.1 MARCO DE REFERENCIA.

Hace algunos años, la crisis financiera en que se encontraba el país obligó a muchas empresas mexicanas a buscar créditos en el extranjero para poder hacer frente a sus compromisos. Adicionalmente este mismo desequilibrio económico desestabilizó el tipo de cambio del peso frente al dólar, por lo que las empresas encontraron serias dificultades para conseguir las divisas necesarias para realizar sus pagos.

El riesgo de una devaluación que convirtiera los créditos denominados en monedas extranjeras en deudas imposibles de pagar, obligó al gobierno federal a implementar un programa que asegurara a los empresarios la disponibilidad de divisas suficientes, a la vez que les concediera cobertura contra riesgos cambiarios.

A grandes rasgos, este programa consistió en lo siguiente: las empresas establecidas en el país que hubieran contraído adeudos en moneda extranjera con entidades financieras del exterior, podían suscribir uno o varios contratos con el Banco de México. A través de estos contratos las empresas aseguraban el pago oportuno de sus obligaciones con un tipo de cambio controlado, siempre y cuando ellas realizaran los pagos correspondientes en moneda nacional.

Para la recepción de abonos por parte de las empresas y para la realización de pagos a los acreedores, el Banco de México se sirvió de las instituciones bancarias del país, que se convirtieron en operadoras de los contratos.

De esta manera, las empresas efectúan depósitos en moneda nacional en las cuentas que los bancos les asignan para tal fin. Mensualmente, los bancos reúnen los fondos captados en este proceso y los transfieren a cuentas registradas en el Banco de México. En correspondencia, éste transfiere un monto equivalente en dólares a las cuentas de moneda extranjera propiedad de cada banco, desde donde cada institución efectúa los pagos a los acreedores.

Un esquema conceptual de esta situación es el diagrama de entidad-relación de la figura 4.1, en él se aprecian las principales entidades que participan en el modelo así como sus relaciones.

Para instrumentar este programa, el Banco de México instituyó un fideicomiso con las siguientes funciones principales:

- Administrar los contratos de cobertura cambiaria.

- **Calcular y recibir de los bancos las aportaciones de las empresas en moneda nacional.**
- **Calcular y proporcionar a los bancos las cantidades en moneda extranjera a entregar a los acreedores.**
- **Efectuar inversiones con los fondos captados.**

Cada una de estas funciones se compone a su vez de una serie de procesos, los cuales se enlistan en la figura 4.2 asociados a la función de la que forman parte. Los procesos de administración interna del fideicomiso se agrupan bajo una nueva función llamada ***Administración Interna.***

La estructura ideada para hacer frente a estos procesos se muestra en la figura 4.3 mediante un diagrama de descomposición. En la matriz de la figura 4.4, las unidades organizacionales que integran el fideicomiso se asocian con los procesos a su cargo.

Como en todas las organizaciones, comenzaron a surgir una serie de problemas que obstaculizaban en mayor o menor grado el desarrollo de los diferentes procesos. En la tabla de la figura 4.5 puede observarse una relación de los problemas más apremiantes relacionándolos con los procesos que resultan afectados por ellos.

Tras un simple análisis visual de la tabla mencionada, puede establecerse que tres de los problemas existentes tienen un área de influencia mayor. Dos de ellos (*Gran volumen de operaciones* y *Elevado número de casos especiales*) son producto de la magnitud del programa de cobertura y deben ser resueltos o minimizados por la totalidad de los sistemas. Por el contrario, el tercer caso (*Incongruencia entre directorios*), es producto de una organización deficiente y puede ser eliminado mediante la creación de un sistema centralizado que proporcione a todas las oficinas la información que requieren acerca del directorio de entidades, asegurando la oportunidad y unicidad de la información.

En modo de reporte, la importancia de solucionar el pro-

blema de incongruencia entre directorios, puede observarse al relacionar las unidades organizacionales con los procesos a su cargo y a su vez éstos con los problemas que los afectan mediante un reporte ordenado jerárquicamente como el que se incluye en la figura 4.6. Analizando dicho reporte, puede apreciarse que únicamente dos unidades (Oficina Técnica y Subdirección de Contabilidad) no son afectadas por este problema.

Para comprender mejor por qué este problema afecta tan profundamente a la organización, examinemos tres entidades de nuestro modelo: Empresa, Acreedor y Banco. En las figuras 4.7, 4.8 y 4.9 se desglosan los datos de estas entidades que son de interés para el programa de cobertura, así como las relaciones en donde intervienen. La mayor parte de estos datos son almacenados en directorios por las diferentes oficinas para ser utilizados en determinados procesos de la operación. Cuando la información contenida en el directorio de una oficina no coincide con la de los demás, los procesos se entorpecen mientras se determina la razón de la incongruencia y se decide cual es la información real.

La construcción de un sistema que controle el directorio de empresas, bancos y acreedores, beneficiaría directamente la ejecución de quince procesos y el desempeño de siete oficinas, que además se verían liberadas de la tarea de mantenimiento de información del directorio, pues ésta se efectuaría una sola vez en toda la organización. Debido al amplio radio de acción de este problema y a los beneficios prácticos que reportaría su solución, se eligió para ejemplificar en este capítulo el análisis y diseño de sistemas a través de herramientas CASE.

Desde luego no todas las unidades requieren de toda la información, en la figura 4.10 se observa en qué procesos interviene cuando menos un atributo de las diferentes entidades. Sin embargo, un sistema centralizado proporcionará a cada quien lo que necesite, e incluso, siempre que sea posible, suministrará sin costo extra, información adicional a la que cada oficina ya maneja, es decir, pondrá la información de unas al alcance de otras.

Figura 4.1

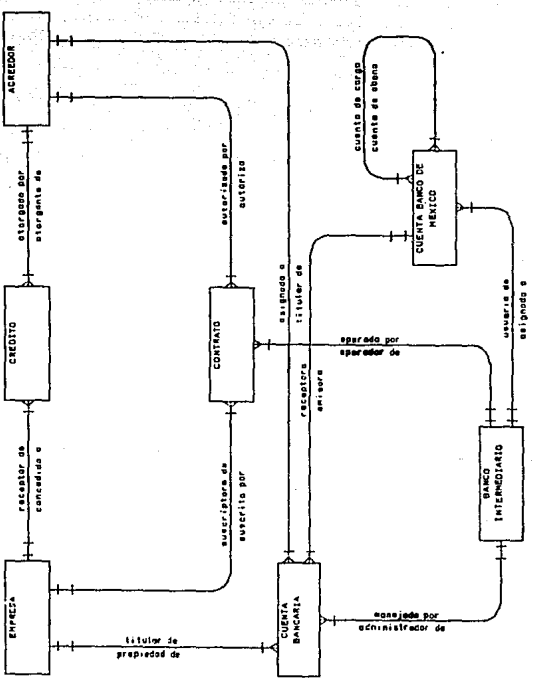
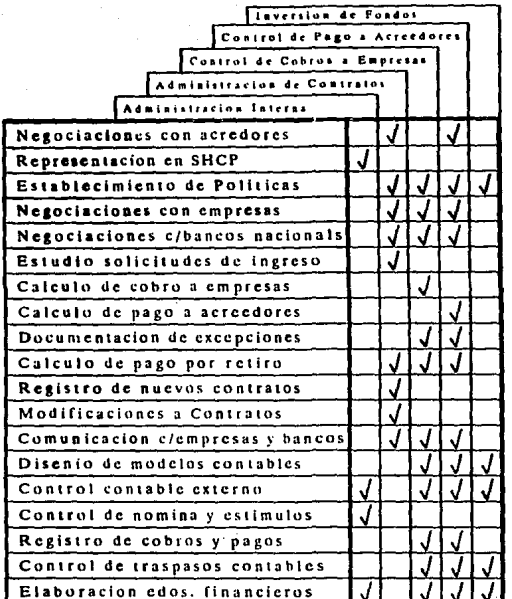

AGREDOR
otorgado por
otorgante de
CREDITO
suscrito por
suscriptora de
CONTRATO
autorizado por
autoriza
cuenta de cargo
cuenta de abono
CUENTA BANCO DE MEXICO
asignado a
titular de
operado por
operador de
usuario de
asignado a
BANCO INTERMEDIARIO
receptora de
concedido a
EMPRESA
titular de
propiedad de
CUENTA BANCARIA
manejada por
administrador de
Entity Model
Nov 14, 1992 20:36:12

Figura 4.2



	Administracion Interna	Administracion de Contratos	Control de Cobros a Empresas	Control de Pago a Acreedores	Inversion de Fondos
Negociaciones con acreedores		√		√	
Representacion en SHCP	√				
Establecimiento de Politicas		√	√	√	√
Negociaciones con empresas		√	√	√	
Negociaciones c/bancos nacionals		√	√	√	
Estudio solicitudes de ingreso		√			
Calculo de cobro a empresas			√		
Calculo de pago a acreedores				√	
Documentacion de excepciones			√	√	
Calculo de pago por retiro		√	√	√	
Registro de nuevos contratos		√			
Modificaciones a Contratos		√			
Comunicacion c/empresas y bancos		√	√	√	
Diseño de modelos contables			√	√	√
Control contable externo	√		√	√	√
Control de nomina y estimulos	√				
Registro de cobros y pagos			√	√	
Control de traspasos contables			√	√	√
Elaboracion edos. financieros	√		√	√	√
Tramite de documentacion		√	√	√	
Control de personal	√				
Admon. recursos materiales	√				
Desarrollo/mantenimiento sistema	√	√	√	√	
Asesoria informat./capacitacion	√				
Adquisicion e instalacion equipo	√				
Seleccion de programas inversion					√
Diseño de contratos		√			

Process is a part of Function

May 22, 1992 20:37:57

DIRECCION FICORCA

Hoy 15, 1992 18:28:45

Figura 4.3

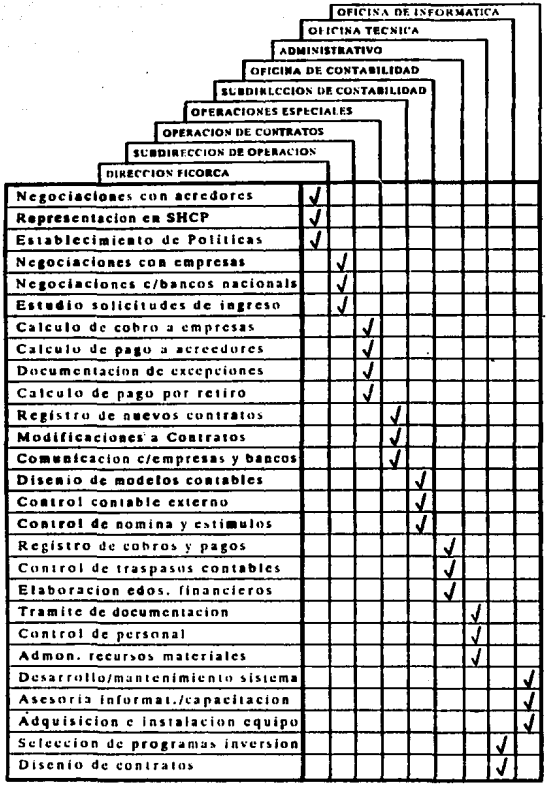
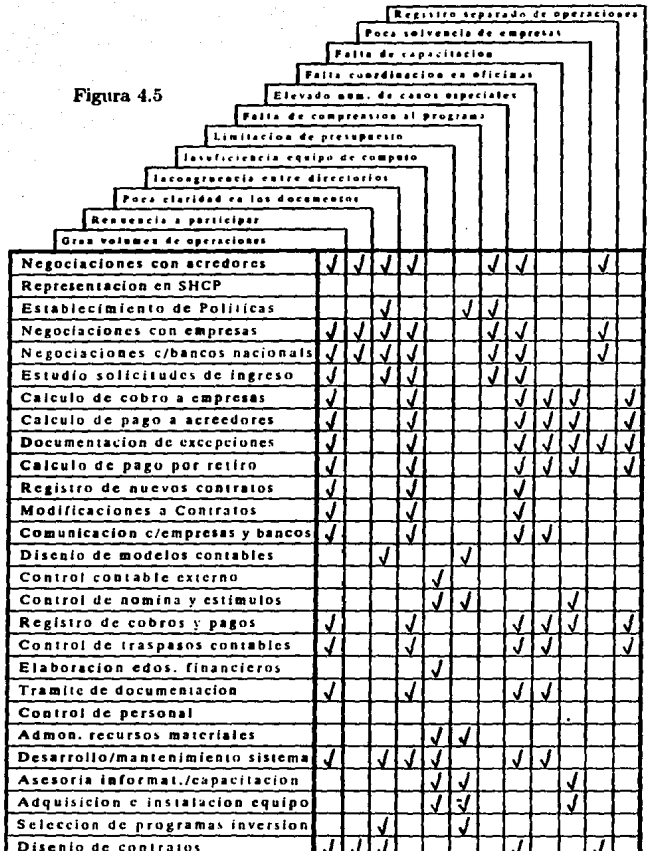

OFICINA DE INFORMATICA
OFICINA TECNICA
ADMINISTRATIVO
OFICINA DE CONTABILIDAD
SUBDIRLCCION DE CONTABILIDAD
OPERACIONES ESPECIALES
OPERACION DE CONTRATOS
SUBDIRECCION DE OPERACION
DIRECCION FICORCA
Negociaciones con acredores
Representacion en SHCP
Establecimiento de Politicas
Negociaciones con empresas
Negociaciones c/bancos nacionals
Estudio solicitudes de ingreso
Calculo de cobro a empresas
Calculo de pago a acreedores
Documentacion de excepciones
Calculo de pago por retiro
Registro de nuevos contratos
Modificaciones a Contratos
Comunicacion c/empresas y bancos
Diseno de modelos contables
Control contable externo
Control de nomina y estimulos
Registro de cobros y pagos
Control de traspasos contables
Elaboracion edos. financieros
Tramite de documentacion
Control de personal
Admon. recursos materiales
Desarrollo/mantenimiento sistema
Asesoria informat./capacitacion
Adquisicion e instalacion equipo
Seleccion de programas inversion
Diseno de contratos

Figura 4.4

Figura 4.5



Process is affected by Problem	Gran volumen de operaciones	Renuencia a participar	Poca claridad en los documentos	Incongruencia entre directorios	Insuficiencia equipo de computo	Limitacion de presupuesto	Falta de comprension al programa	Elevado num. de casos especiales	Falta coordinacion en oficinas	Falta de capacitacion	Poca solvencia de empresas	Registro separado de operaciones
Negociaciones con acredores	✓	✓	✓	✓			✓	✓			✓	
Representacion en SHCP												
Establecimiento de Politicas			✓			✓	✓					
Negociaciones con empresas	✓	✓	✓	✓			✓	✓			✓	
Negociaciones c/bancos nacionals	✓	✓	✓	✓			✓	✓			✓	
Estudio solicitudes de ingreso	✓		✓	✓			✓	✓				
Calculo de cobro a empresas	✓			✓				✓	✓	✓		✓
Calculo de pago a acreedores	✓			✓				✓	✓	✓		✓
Documentacion de excepciones	✓			✓				✓	✓	✓	✓	✓
Calculo de pago por retiro	✓			✓				✓	✓	✓		✓
Registro de nuevos contratos	✓			✓				✓				
Modificaciones a Contratos	✓			✓				✓				
Comunicacion c/empresas y bancos	✓			✓				✓	✓			
Diseño de modelos contables			✓			✓						
Control contable externo					✓							
Control de nomina y estimulos					✓	✓				✓		
Registro de cobros y pagos	✓			✓				✓	✓	✓		✓
Control de traspasos contables	✓			✓				✓	✓			✓
Elaboracion edos. financieros					✓							
Tramite de documentacion	✓			✓				✓	✓			
Control de personal												
Admon. recursos materiales					✓	✓						
Desarrollo/mantenimiento sistema	✓		✓	✓	✓			✓	✓			
Asesoria informat./capacitacion					✓	✓				✓		
Adquisicion e instalacion equipo					✓	✓				✓		
Seleccion de programas inversion			✓			✓						
Diseño de contratos	✓	✓	✓					✓			✓	

Hierarchy Report

The selected associations with the implied nesting are:

1 Organizational Unit is responsible for Process
2 Process is affected by Problem

ORGANIZATIONAL UNIT DIRECCION FICORCA
IS RESPONSIBLE FOR PROCESS Negociaciones con acreedores
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta de comprension al programa
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Poca claridad en los documentos
IS AFFECTED BY PROBLEM Poca solvencia de empresas
IS AFFECTED BY PROBLEM Renuencia a participar
IS RESPONSIBLE FOR PROCESS Representacion en SHCP
IS RESPONSIBLE FOR PROCESS Establecimiento de Politicas
IS AFFECTED BY PROBLEM Falta de comprension al programa
IS AFFECTED BY PROBLEM Limitacion de presupuesto
IS AFFECTED BY PROBLEM Poca claridad en los documentos

ORGANIZATIONAL UNIT OFICINA DE INFORMATICA
IS RESPONSIBLE FOR PROCESS Desarrollo/mantenimiento sistema
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta coordinacion en oficinas
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Insuficiencia equipo de computo
IS AFFECTED BY PROBLEM Poca claridad en los documentos
IS RESPONSIBLE FOR PROCESS Asesoria informat./capacitacion
IS AFFECTED BY PROBLEM Limitacion de presupuesto
IS AFFECTED BY PROBLEM Insuficiencia equipo de computo
IS AFFECTED BY PROBLEM Falta de capacitacion
IS RESPONSIBLE FOR PROCESS Adquisicion e instalacion equipo
IS AFFECTED BY PROBLEM Falta de capacitacion
IS AFFECTED BY PROBLEM Insuficiencia equipo de computo
IS AFFECTED BY PROBLEM Limitacion de presupuesto

ORGANIZATIONAL UNIT OFICINA TECNICA
IS RESPONSIBLE FOR PROCESS Seleccion de programas inversion
IS AFFECTED BY PROBLEM Poca claridad en los documentos
IS AFFECTED BY PROBLEM Limitacion de presupuesto
IS RESPONSIBLE FOR PROCESS Diseño de contratos
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Poca claridad en los documentos
IS AFFECTED BY PROBLEM Poca solvencia de empresas
IS AFFECTED BY PROBLEM Renuencia a participar

Figura 4.6

ORGANIZATIONAL UNIT SUBDIRECCION DE OPERACION
IS RESPONSIBLE FOR PROCESS Negociaciones con empresas
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta de comprension al programa
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Poca claridad en los documentos
IS AFFECTED BY PROBLEM Poca solvencia de empresas
IS AFFECTED BY PROBLEM Renuencia a participar
IS RESPONSIBLE FOR PROCESS Negociaciones c/bancos nacionals
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta de comprension al programa
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Poca claridad en los documentos
IS AFFECTED BY PROBLEM Poca solvencia de empresas
IS AFFECTED BY PROBLEM Renuencia a participar
IS RESPONSIBLE FOR PROCESS Estudio solicitudes de ingreso
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta de comprension al programa
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Poca claridad en los documentos

ORGANIZATIONAL UNIT SUBDIRECCION DE CONTABILIDAD
IS RESPONSIBLE FOR PROCESS Diseno de modelos contables
IS AFFECTED BY PROBLEM Limitacion de presupuesto
IS AFFECTED BY PROBLEM Poca claridad en los documentos
IS RESPONSIBLE FOR PROCESS Control contable externo
IS AFFECTED BY PROBLEM Insuficiencia equipo de computo
IS RESPONSIBLE FOR PROCESS Control de nomina y estimulos
IS AFFECTED BY PROBLEM Falta de capacitacion
IS AFFECTED BY PROBLEM Insuficiencia equipo de computo
IS AFFECTED BY PROBLEM Limitacion de presupuesto

ORGANIZATIONAL UNIT OPERACION DE CONTRATOS
IS RESPONSIBLE FOR PROCESS Calculo de cobro a empresas
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta coordinacion en oficinas
IS AFFECTED BY PROBLEM Falta de capacitacion
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Registro separado de operaciones
IS RESPONSIBLE FOR PROCESS Calculo de pago a acreedores
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta coordinacion en oficinas
IS AFFECTED BY PROBLEM Falta de capacitacion
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Registro separado de operaciones

Figura 4.6 (Cont.)

IS RESPONSIBLE FOR PROCESS Documentacion de excepciones
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta coordinacion en oficinas
IS AFFECTED BY PROBLEM Falta de capacitacion
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Poca solvencia de empresas
IS AFFECTED BY PROBLEM Registro separado de operaciones
IS RESPONSIBLE FOR PROCESS Calculo de pago por retiro
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta coordinacion en oficinas
IS AFFECTED BY PROBLEM Falta de capacitacion
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Registro separado de operaciones

ORGANIZATIONAL UNIT OPERACIONES ESPECIALES
IS RESPONSIBLE FOR PROCESS Registro de nuevos contratos
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS RESPONSIBLE FOR PROCESS Modificaciones a Contratos
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS RESPONSIBLE FOR PROCESS Comunicacion c/empresas y bancos
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta coordinacion en oficinas
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios

ORGANIZATIONAL UNIT OFICINA DE CONTABILIDAD
IS RESPONSIBLE FOR PROCESS Registro de cobros y pagos
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta coordinacion en oficinas
IS AFFECTED BY PROBLEM Falta de capacitacion
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Registro separado de operaciones
IS RESPONSIBLE FOR PROCESS Control de traspasos contables
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta coordinacion en oficinas
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS AFFECTED BY PROBLEM Registro separado de operaciones
IS RESPONSIBLE FOR PROCESS Elaboracion edos. financieros
IS AFFECTED BY PROBLEM Insuficiencia equipo de computo

Figura 4.6 (Cont.)

ORGANIZATIONAL UNIT ADMINISTRATIVO
IS RESPONSIBLE FOR PROCESS Tramite de documentacion
IS AFFECTED BY PROBLEM Elevado num. de casos especiales
IS AFFECTED BY PROBLEM Falta coordinacion en oficinas
IS AFFECTED BY PROBLEM Gran volumen de operaciones
IS AFFECTED BY PROBLEM Incongruencia entre directorios
IS RESPONSIBLE FOR PROCESS Control de personal
IS RESPONSIBLE FOR PROCESS Admon. recursos materiales
IS AFFECTED BY PROBLEM Limitacion de presupuesto
IS AFFECTED BY PROBLEM Insuficiencia equipo de computo

Figura 4.6 (Cont.)

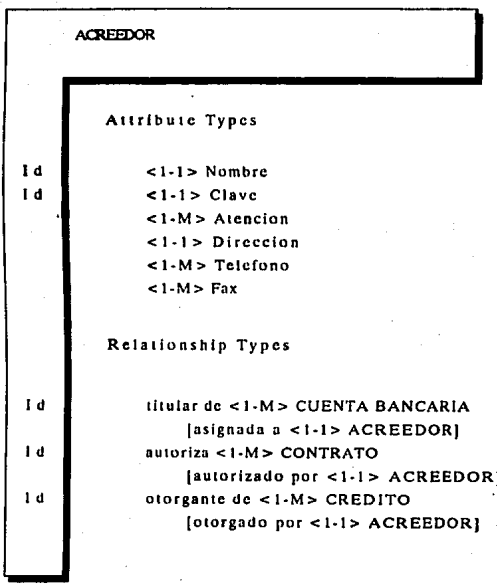
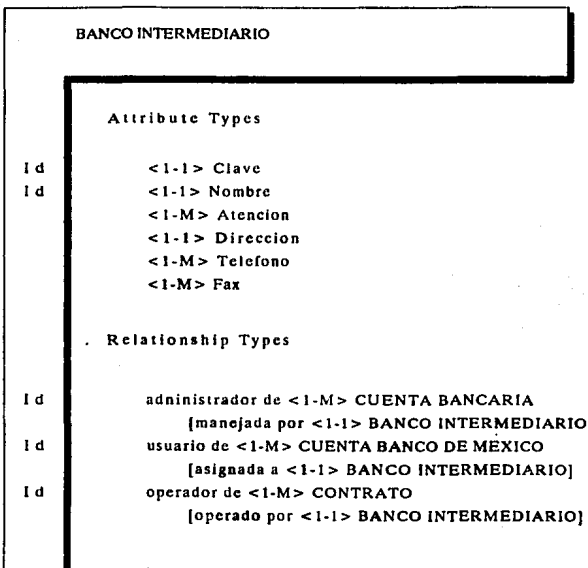

ACREEDOR
Attribute Types
Id
Id
<1-1> Nombre
<1-1> Clave
<1-M> Atencion
<1-1> Direccion
<1-M> Telefono
<1-M> Fax
Relationship Types
Id
titular de <1-M> CUENTA BANCARIA
[asignada a <1-1> ACREEDOR]
Id
autoriza <1-M> CONTRATO
[autorizado por <1-1> ACREEDOR]
Id
otorgante de <1-M> CREDITO
[otorgado por <1-1> ACREEDOR]
Context: None (Entity Model)
June 1, 1992 19:35:34

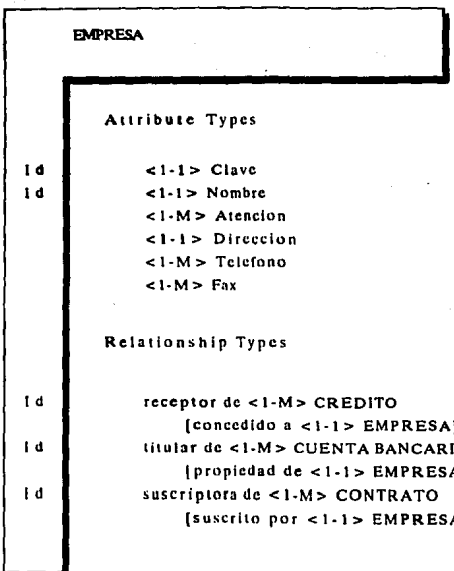
Figura 4.7



Context: None (Entity Model)

June 1, 1992 19:36:27

Figura 4.8



Context: None (Entity Model)

June 1, 1992 19:32:31

Figura 4.9

Process Involves Entity Type

June 2, 1992 15:39:54

Figura 4.10

4.2 ANALISIS DEL SISTEMA.

En el segundo capítulo se trató el Análisis Estructurado de Tom De Marco. En la presente sección se muestra el análisis del sistema de directorios o catálogos siguiendo la metodología de De Marco y bajo el ambiente CASE de IEW.

Se recordará que para De Marco una especificación estructurada consta de tres partes: un conjunto de diagramas de flujo, un Diccionario de Datos que especifique el contenido de los flujos y un conjunto de miniespecificaciones que detallen los pasos seguidos en los procesos de más bajo nivel.

Asimismo los primeros pasos en la construcción de una especificación estructurada consisten, según esta metodología, en la elaboración del modelo físico y del modelo lógico del sistema actual, para a continuación establecer el modelo lógico y el modelo físico del nuevo sistema a desarrollar.

4.2.1 MODELO ACTUAL DEL SISTEMA.

Se incluye en este trabajo sólo un modelo para el sistema actual que aún cuando tiene información física (como nombres de documentos por ejemplo) puede considerarse como modelo lógico puesto que los flujos están completamente definidos en base a la información que trasladan y no al formato de dicha información.

El modelo está integrado por las cuatro partes que se enlistan a continuación y para las cuales se incluyen algunos comentarios a manera de explicación:

a) Diagrama de descomposición.

b) Diagramas de flujo de datos.

c) Diccionario de datos.

d) Miniespecificaciones.

Diagrama de descomposición. Aún cuando estrictamente hablando esta herramienta no forma parte de la especificación estructurada, su inclusión se justifica por dos razones: en primer lugar, permite apreciar la descomposición gradual de los procesos, con lo que se facilita el seguimiento del conjunto estructurado de Diagramas de Flujo y en segundo lugar, simplifica la identificación de los procesos primitivos o de más bajo nivel y para los que debe incluirse una miniespecificación.

El Diagrama de Descomposición que se muestra aquí, se divide en dos páginas por razones de presentación. En cada una de las páginas se desarrollan sólo dos de las ramas principales de la jerarquía mientras que otras dos permanecen compactadas, lo que se muestra con dos puntos debajo de ellas (páginas 109 y 110).

Diagramas de flujo. Dentro de esta componente se incluyen únicamente los diagramas de los niveles superiores. El nombre del proceso que describen puede apreciarse en la parte inferior izquierda, mientras que su numeración dentro de los diferentes niveles aparece en la esquina inferior derecha.

Son particularmente importantes los diagramas 1.2, 1.3 y 1.4 correspondientes a los procesos de Registro. Operación y Contabilidad de Contratos respectivamente. En ellos se aprecia la creación y mantenimiento de tres diferentes catálogos de datos generales de empresas y acreedores: Catálogo de Control. de Operación y de Contabilidad. Esta multiplicidad de la información y de procesos. es el origen del problema de incongruencia detectado en la fase de planeación y que intenta resolver el nuevo sistema.

Asimismo, puede apreciarse que la información operativa se incorpora al archivo llamado PIPA en el proceso de Registro de Contratos y más tarde al archivo MAESTRO DE CONTRATOS durante

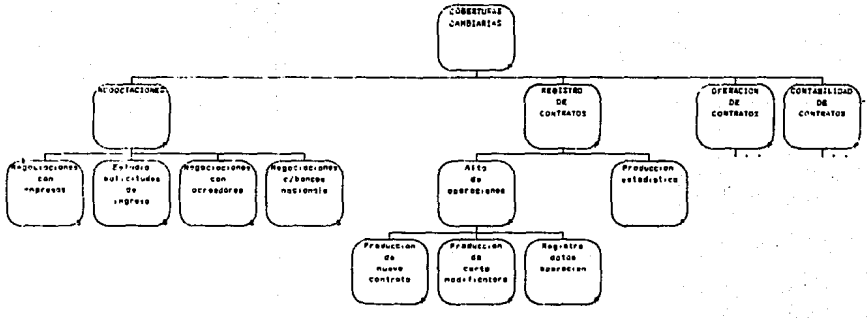
Operación de Contratos, produciéndose con ésto el riesgo de diferencias entre los registros de diferentes unidades organizacionales.

Diccionario de datos. La descripción del contenido de un flujo de datos se realiza listando sus componentes, mismas que pueden ser: entidades, atributos de las entidades, relaciones entre entidades y otros flujos de datos. Cada entrada del diccionario se presenta en dos partes, en el primer renglón la expresión *"Flow Expresion of Data Flow"* es seguida por el nombre del flujo de datos a describir, en los siguientes renglones se listan todas sus componenetes.

Se emplean las siguientes convenciones: un nombre con mayúsculas representa una de las entidades identificadas en el modelo Entidad-relación de la figura 4.1 del presente capítulo (CREDITO, EMPRESA); los atributos de dichas entidades se denotan con el nombre de la entidad seguido de un punto y el nombre del atributo (ACREEDOR.clave); varios atributos de una misma entidad pueden agruparse como en EMPRESA.(clave,nombre,teléfono); las relaciones entre dos entidades se denotan con la estructura ENTIDAD1.nombre-relacion.ENTIDAD2; finalmente los nombres aislados en letras minúsculas se refieren a otros flujos de datos.

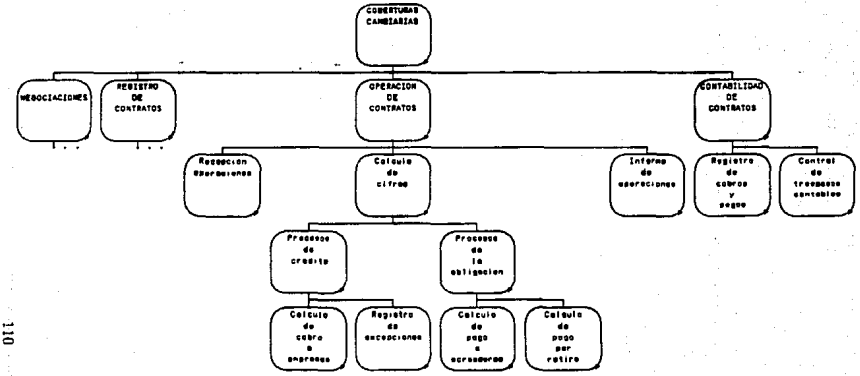
Miniespecificaciones. En los diagramas de descomposición, puede apreciarse la existencia de 16 procesos elementales para los que se produjeron 16 miniespecificaciones. Se incluyen en esta sección cinco de ellas que se consideran representativas del sistema, y en las que es posible valorar el empleo de los diferentes catálogos.

DIAGRAMAS DE DESCOMPOSICION
(MODELO ACTUAL)



COBERTURAS CAMBIARIAS

September 21, 1992 16:27:03



COBERTURAS CAMBIARIAS

September 17, 1982 10:24:41

DIAGRAMAS DE FLUJO

(MODELO ACTUAL)

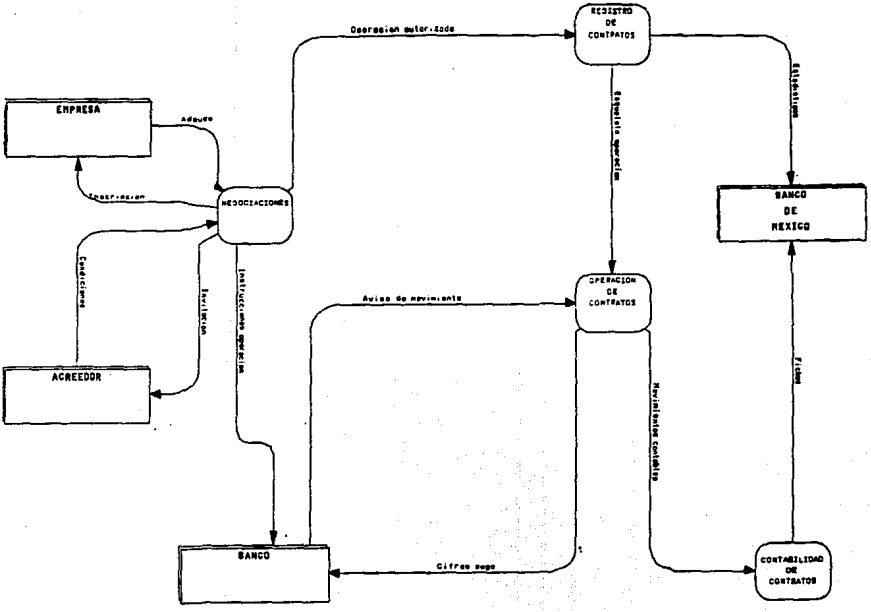


Diagrama 1

COBERTURAS CAMBIARIAS

September 1, 1992 19:21:08

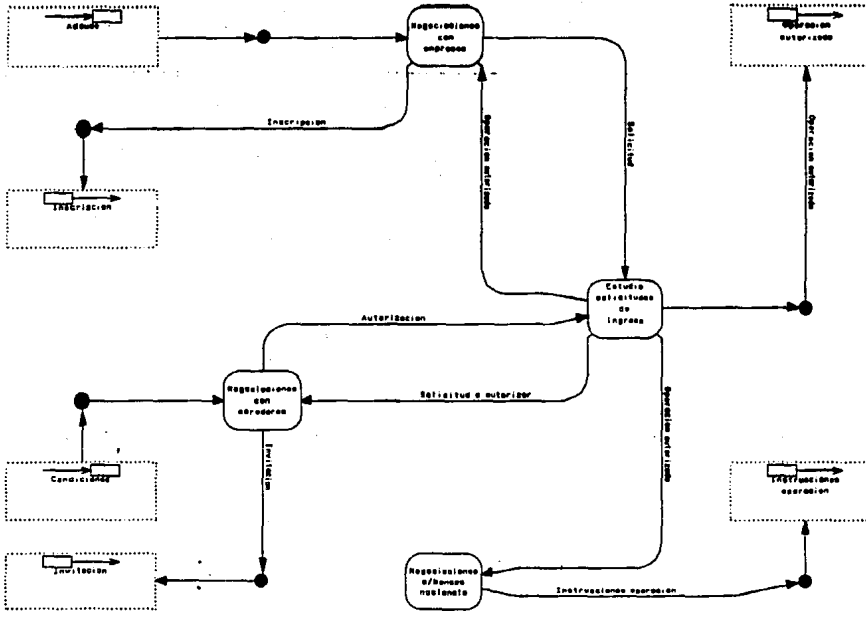


Diagrama 1.1

NEGOCIACIONES

Septiembre 1, 1992 10:50:52

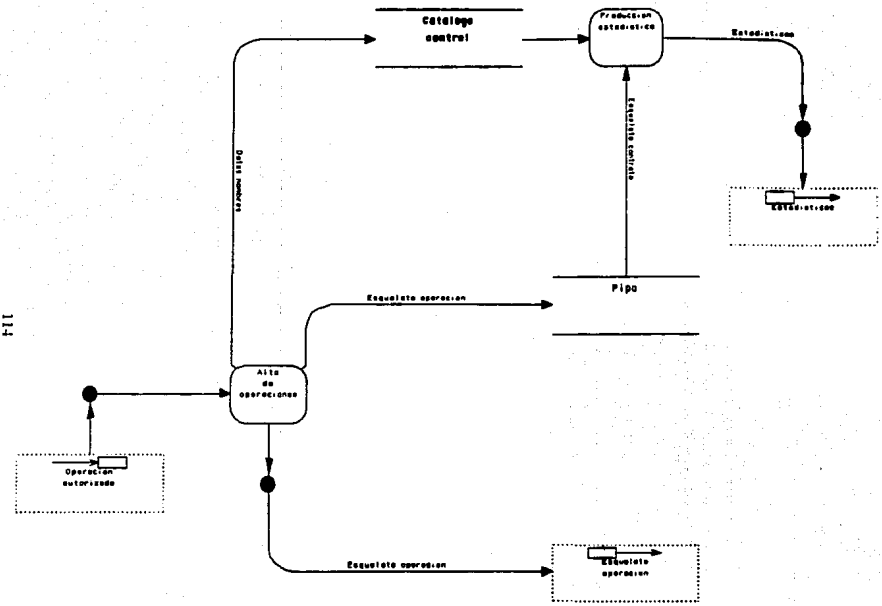

Catálogo central
Producción estadística
Estadísticas
Datos maestros
Estadísticas centrales
Estadísticas
Pipa
Esqueleto operación
Alta de operaciones
Operación autorizada
Esqueleto operación
Esqueleto operación

Diagrama 1.2

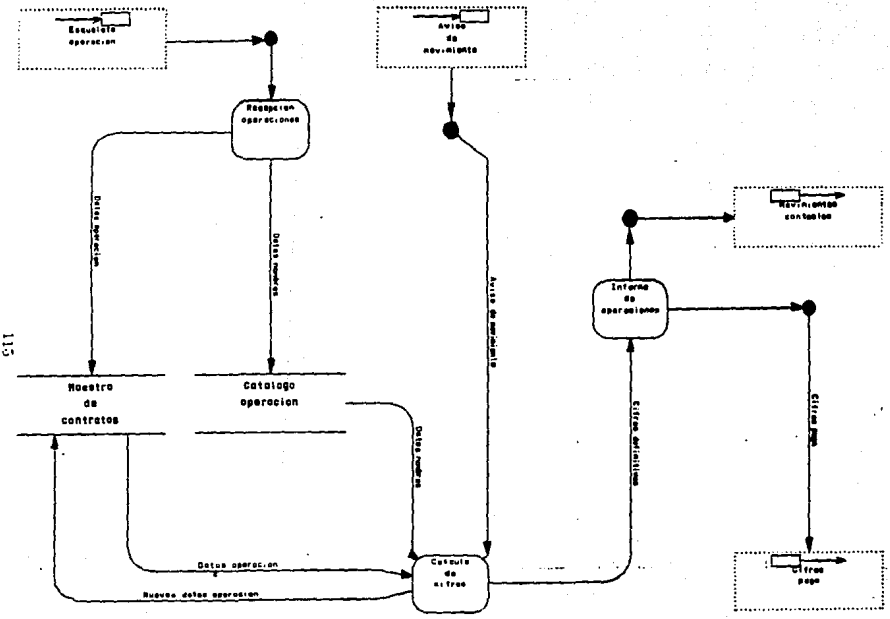
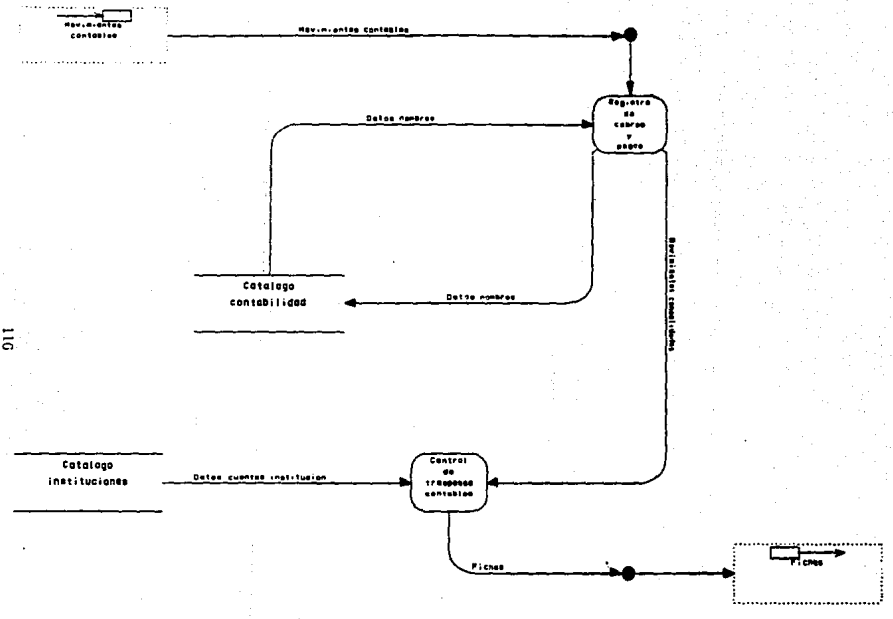


Diagrama 1.3

OPERACION DE CONTRATOS

September 8, 1993 17:26:29



CONTABILIDAD DE CONTRATOS

September 29, 1992 14:31:43

Diagrama 1.4

DICCIONARIO DE DATOS

(MODELO ACTUAL)

Flow Expression Report

Flow Expression of Data Flow Adeudo
CREDITO.

Flow Expression of Data Flow Autorizacion
Solicitud a autorizar + CONTRATO.autorizado por.ACREEDOR

Flow Expression of Data Flow Aviso de movimiento
CONTRATO.Folio + EMPRESA.(Clave,Nombre) + BANCO INTERMEDIARIO.(Clave,Nombre) + ACREEDOR.(Clave,Nombre)

Flow Expression of Data Flow Carta modificatoria
ACREEDOR. + BANCO INTERMEDIARIO. + EMPRESA. + CONTRATO.

Flow Expression of Data Flow Cifras definitivas
Cifras pago + Movimientos contables

Flow Expression of Data Flow Cifras pago
CONTRATO.Folio + EMPRESA.(Clave,Nombre) + ACREEDOR.(Clave,Nombre) + BANCO INTERMEDIARIO.(Atencion,Direccion,Fax,Clave,Nombre)

Flow Expression of Data Flow Condiciones
CONTRATO.firma

Flow Expression of Data Flow Contrato
CONTRATO. + EMPRESA. + BANCO INTERMEDIARIO. + ACREEDOR.

Flow Expression of Data Flow Datos cuentas institucion
CUENTA BANCO DE MEXICO.

Flow Expression of Data Flow Datos nombres
BANCO INTERMEDIARIO.(Clave,Nombre,Atencion,Direccion,Fax) +
EMPRESA.(Clave,Nombre,Atencion,Direccion,Fax) +
ACREEDOR.(Clave,Nombre,Atencion,Direccion,Fax)

Flow Expression of Data Flow Datos operacion
CONTRATO.(Folio,Monto,Plazo,Tipo) + BANCO
INTERMEDIARIO.Clave + EMPRESA.Clave + ACREEDOR.Clave

Flow Expression of Data Flow Esqueleto contrato
Datos nombres + Datos operacion

Flow Expression of Data Flow Esqueleto modificacion
Datos nombres + Datos operacion

Flow Expression of Data Flow Esqueleto operacion
Esqueleto contrato | Esqueleto modificacion

Flow Expression of Data Flow Estadisticas
CONTRATO.(Monto,Plazo,Tipo) + ACREEDOR.(Clave,Nombre) +
EMPRESA.(Clave,Nombre)

Flow Expression of Data Flow Fichas
CUENTA BANCO DE MEXICO.(Numero,Saldo)

Flow Expression of Data Flow Inscripcion
CONTRATO.suscrito por.EMPRESA

Flow Expression of Data Flow Instrucciones operacion
CONTRATO. + EMPRESA. + ACREEDOR. + CONTRATO.operado por.BANCO INTERMEDIARIO

Flow Expression of Data Flow Invitacion
Solicitud a autorizar

Flow Expression of Data Flow Movimientos consolidados
CONTRATO.Folio + BANCO INTERMEDIARIO.Clave

Flow Expression of Data Flow Movimientos contables
CONTRATO.Folio + BANCO INTERMEDIARIO.(Clave,Nombre) + ACREEDOR.(Clave,Nombre)

Flow Expression of Data Flow Nuevos datos operacion
Datos operacion

Flow Expression of Data Flow Operacion autorizada
Esqueleto contrato | Esqueleto modificacion

Flow Expression of Data Flow Solicitud
CREDITO. + EMPRESA. + ACREEDOR. + CREDITO.concedido a.EMPRESA + CREDITO.otorgado por.ACREEDOR

Flow Expression of Data Flow Solicitud a autorizar
Solicitud + CONTRATO.(Monto,Plazo,Tipo)

MINIESPECIFICACIONES

(MODELO ACTUAL)

* REGISTRO DATOS OPERACION

Propósito: Producir archivos con la información medular de los contratos

Para cada OPERACION AUTORIZADA

Si operacion = CONTRATO

Resumir CONTRATO en DATOS OPERACION

Extraer DATOS NOMBRES de CONTRATO

Registrar en archivo PIPA los DATOS OPERACION

Si DATOS NOMBRES no registrados

Registrar DATOS NOMBRES en CATALOGO CONTROL

Si OPERACION = CARTA MODIFICATORIA

Extraer de CARTA MODIFICATORIA nuevos DATOS OPERACION

Extraer de CARTA MODIFICATORIA nuevos DATOS NOMBRES

Reescribir en archivo PIPA nuevos DATOS OPERACION

Si modificación = corrección

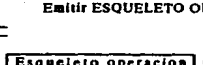
Reescribir DATOS NOMBRES en CATALOGO CONTROL

Si modificación = cambio

Si DATOS NOMBRES no registrados

Registrar DATOS NOMBRES en CATALOGO CONTROL

Emitir ESQUELETO OPERACION para OPERACION DE CONTRATOS



Datos nombres
Esqueleto contrato
Input Flows

* PRODUCCION ESTADISTICAS
  Propósito: Generar información que refleje el estado de avance del programa

* Se producen estadísticas globales
  (p.e. montos totales por CONTRATO.TIPO o ACREEDOR.TIPO)

Mientras no sea fin de archivo PIPA
  Leer un registro
  Acumular cifras por CONTRATO.TIPO o ACREEDOR.TIPO

Generar cifras finales

* Se producen estadísticas particulares
  (p.e. saldos por ACREEDOR o EMPRESA)

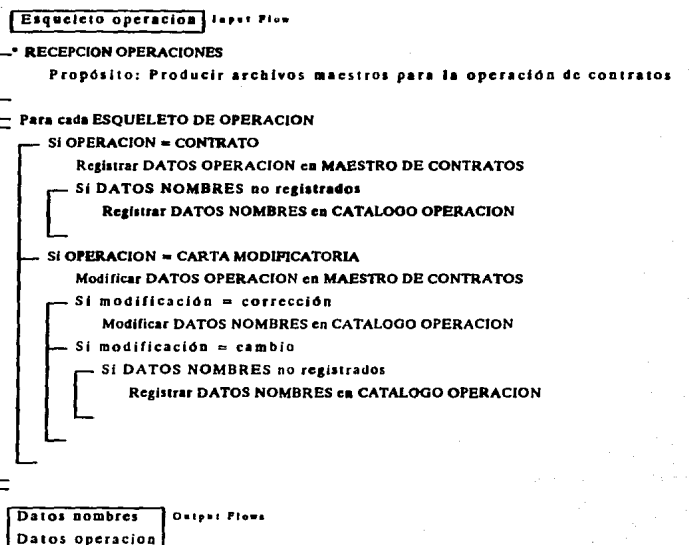
Leer ACREEDOR.NOMBRE o EMPRESA.NOMBRE de CATALOGO CONTROL
Seleccionar registros de PIPA a incluir
Mientras haya registros a incluir
  Acumular sus cifras

Generar cifras finales

Estadísticas
Output Flow

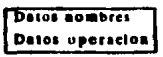
Produccion estadistica

September 21, 1992 18:48:53



Recepcion operaciones

September 21, 1992 18:49:42



* CALCULO DE COBRO A EMPRESAS

Propósito: Determinar los monto a cobrar periódicamente a las EMPRESAS

Para cada CONTRATO.FOLIO en MAESTRO DE CONTRATOS

Determinar si se le cobra este periodo

Si le toca pago

Calcular amortización a cobrar

Calcular intereses a cobrar

Si CONTRATO al corriente

Actualizar CONTRATO.SALDO en MAESTRO DE CONTRATOS

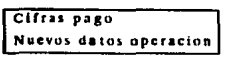
En caso contrario

Calcular amortizaciones vencidas

Calcular intereses moratorios

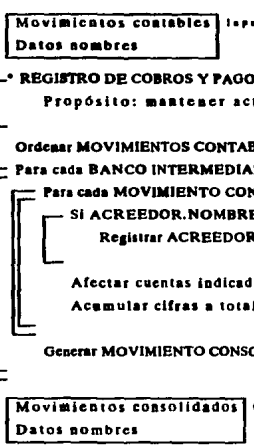
Extraer DATOS NOMBRES de CATALOGO OPERACION

Generar CIFRAS DEFINITIVAS



Calculo de cobro a empresas

September 21, 1992 18:55:59



Registro de cobros y pagos

September 21, 1992 19:23:36

4.2.2 NUEVO MODELO DEL SISTEMA.

En el modelo actual del proceso completo de Coberturas cambiarias, se aprecia el origen del problema de incongruencia entre directorios detectado en la fase de planeación: cada área registra los datos de empresas y personas en un archivo diferente. Como cada captura se efectúa partiendo de diferentes papeles de trabajo, el riesgo de diferencias en la información se multiplica con cada directorio existente.

Por lo tanto, en el nuevo modelo se plantea la existencia de un catálogo único que sustituya los tres existentes, y de un solo proceso de registro que elimine las tareas duplicadas.

Este modelo tiene la ventaja de que la lógica de una gran parte de los procesos de más bajo nivel no se modifica, ya que en ellos sólo se requiere de la información sin ser relevante de dónde provenga.

El esfuerzo desperdiciado en el mantenimiento de catálogos múltiples se enfoca ahora a la explotación más completa del único catálogo existente, produciéndose reportes útiles a todas las áreas con los que no se contaba anteriormente.

Debe notarse que también se elimina la duplicidad de los datos de operación de los contratos de cobertura, ya que ahora existirá un sólo maestro de contratos que se utilizará tanto con fines operativos como estadísticos. Sin embargo, la información operativa se trata con menor detalle con el objeto de resaltar los flujos correspondientes al catálogo de empresas y acreedores.

Al igual que en el caso del modelo anterior, la especificación se presenta según la corriente del Análisis Estructurado, dividida en las siguientes partes:

a) Diagrama de Descomposición. Se presenta en dos páginas (130-131), siendo ahora la raíz el proceso denominado *Dominio*

del Sistema de Directorios. En la primera parte se desglosan los nuevos procesos de Registro y Contabilidad de Contratos y en la segunda la nueva Operación de Contratos.

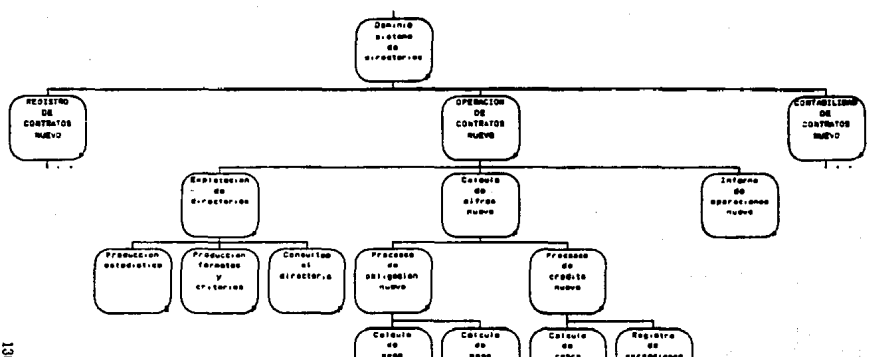
b) Diagramas de flujo. En el primer diagrama se muestra el dominio del sistema de directorios dentro del contexto de las Coberturas Cambiarias. En el diagrama 2.1 se representa el siguiente nivel de descomposición, destacándose la creación del Catálogo Unico y del Maestro de Contratos Unico, lo que constituye la aportación principal del nuevo modelo.

Se anexa también el tercer nivel del Conjunto Estructurado de Diagramas de Flujo que permite apreciar las principales transformaciones en el Registro, Operación y Contabilidad de Contratos (Diagramas 2.1.1, 2.1.2 y 2.1.3). Finalmente se incluye el diagrama correspondiente a la explotación del directorio, que es un proceso nuevo en su mayor parte (Diagrama 2.1.2.1).

c) Diccionario de datos. Elaborado de acuerdo a las convenciones ya mencionadas en el desarrollo del modelo actual.

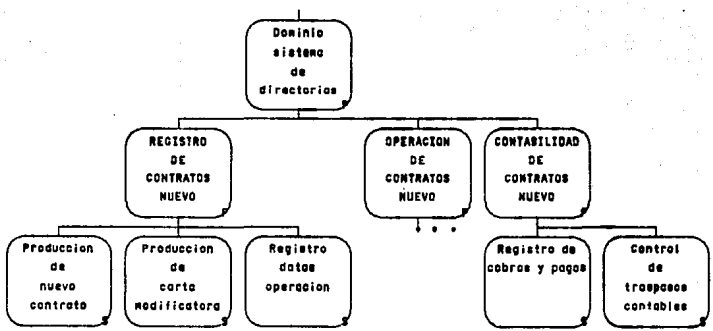
d) Miniespecificaciones. Dentro de las especificaciones más interesantes, se incluyen el Registro de Datos de Operación, en donde se genera la información para el Catálogo Unico y para el Maestro de Contratos: la Producción de Formatos y Criterios para la explotación del catálogo y el Cálculo de Cobro a Empresas que muestra cómo la lógica de los procesos de más bajo nivel no se ve afectada por las modificaciones realizadas.

DIAGRAMAS DE DESCOMPOSICION
(NUEVO MODELO)



Dominio sistema de directorios

October 8, 1992 13:32:38

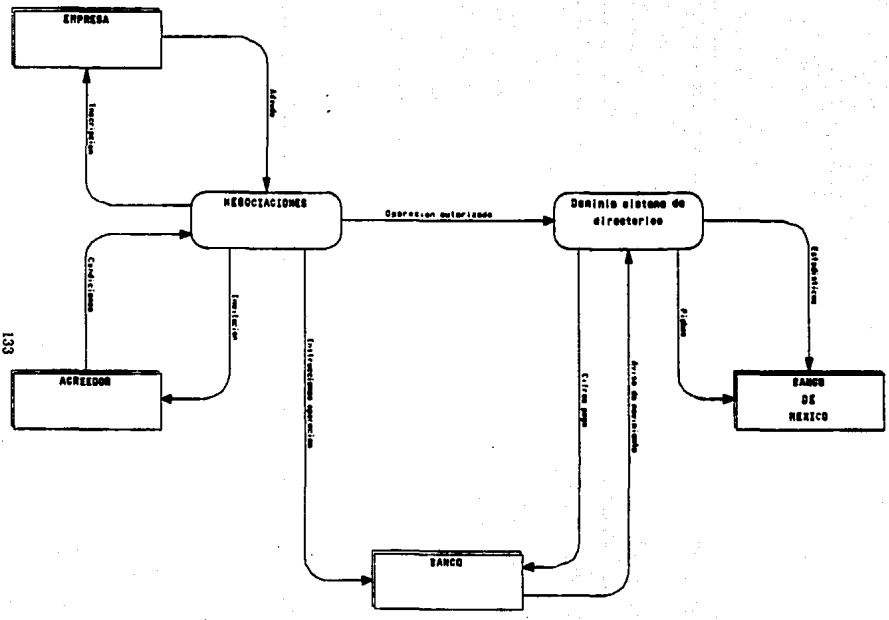


Dominio sistema de directorios

October 6, 1992 13:31:46

DIAGRAMAS DE FLUJO

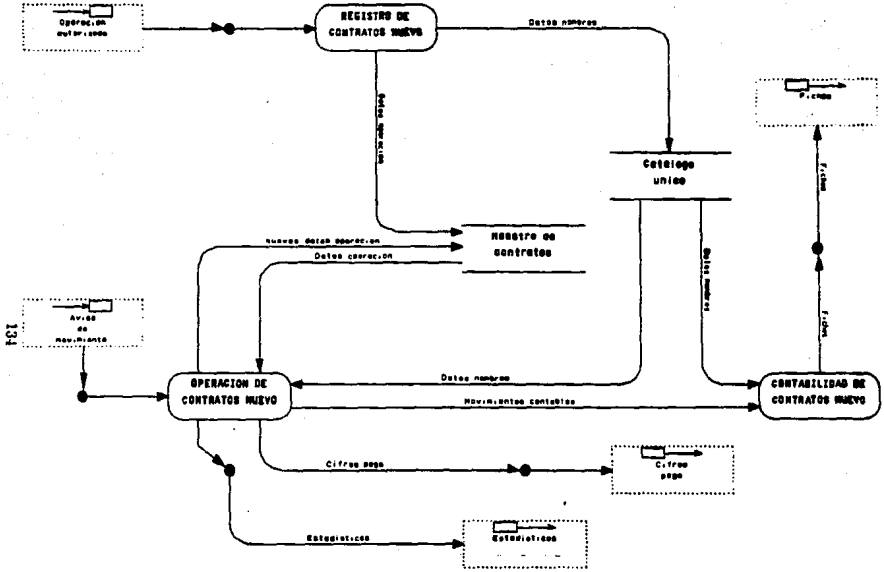
(NUEVO MODELO)



COBERTURAS CAMBIARIAS NUEVO

October 1, 1992 10:52:06

Diagrama 2



134

Domino sistema de directorios

October 8, 1992 13:08:54

Diagrama 2.1

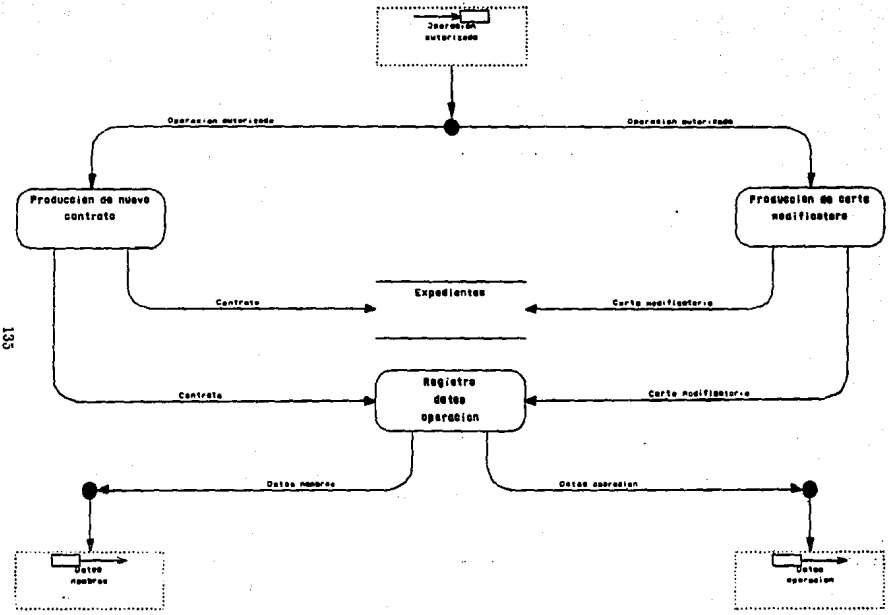
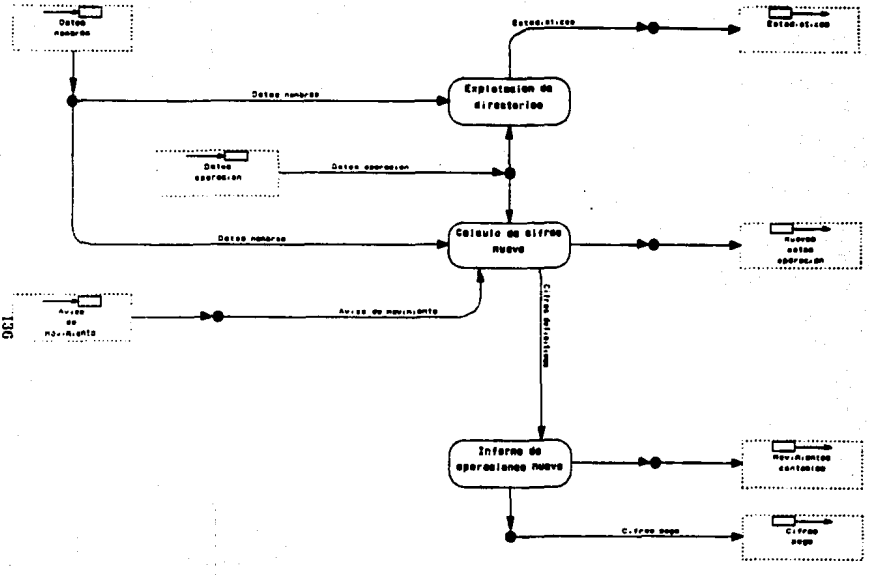


Diagrama 2.1.1

REGISTRO DE CONTRATOS NUEVO

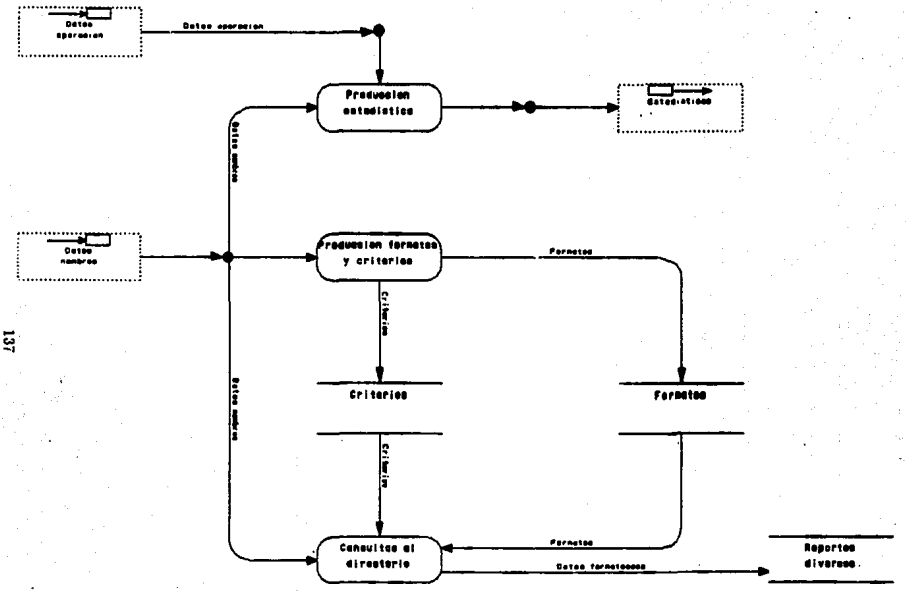
Octubre 1, 1992 19:30:24



OPERACION DE CONTRATOS NUEVO

October 1, 1992 20:09:01

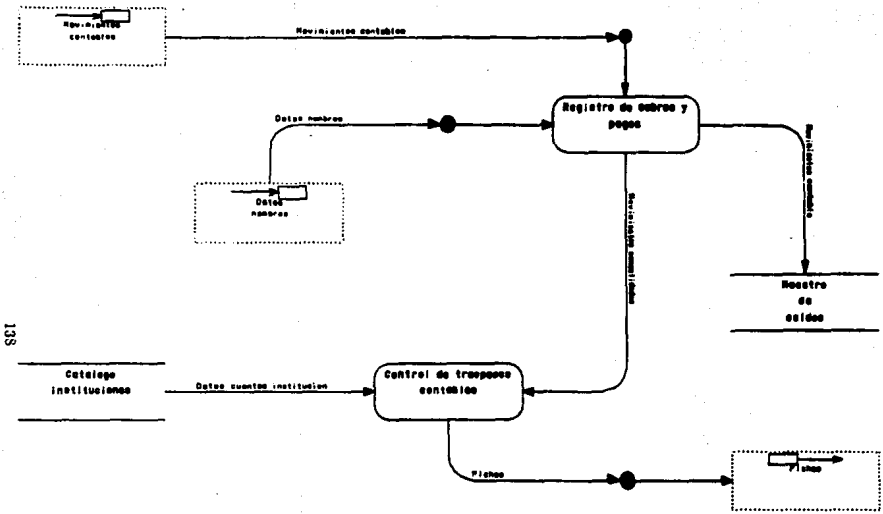
Diagrama 2.1.2



Explotacion de directorios

October 9, 1992 13:21:18

Diagrama 2.1.2.1



CONTABILIDAD DE CONTRATOS NUEVO

October 8, 1993 10:10:10

Diagrama 2.1.3

DICCIONARIO DE DATOS

(NUEVO MODELO)

Flow Expression Report

Flow Expression of Data Flow Adeudo
CREDITO.

Flow Expression of Data Flow Aviso de movimiento
CONTRATO.Folio + EMPRESA.(Clave,Nombre) + BANCO INTERMEDIARIO.(Clave,Nombre) + ACREEDOR.(Clave,Nombre)

Flow Expression of Data Flow Carta modificatoria
ACREEDOR. + BANCO INTERMEDIARIO. + EMPRESA. + CONTRATO.

Flow Expression of Data Flow Cifras definitivas
Cifras pago + Movimientos contables

Flow Expression of Data Flow Cifras pago
CONTRATO.Folio + EMPRESA.(Clave,Nombre) + ACREEDOR.(Clave,Nombre) + BANCO INTERMEDIARIO.(Atencion,Direccion,Fax,Clave,Nombre)

Flow Expression of Data Flow Condiciones
CONTRATO.firma

Flow Expression of Data Flow Contrato
CONTRATO. + EMPRESA. + BANCO INTERMEDIARIO. + ACREEDOR.

Flow Expression of Data Flow Criterios
Nombres de campos + Valores de campos + Operadores de comparacion

Flow Expression of Data Flow Datos cuentas institucion
CUENTA BANCO DE MEXICO.

Flow Expression of Data Flow Datos formateados
Datos nombres + Criterios + Formatos

Flow Expression of Data Flow Datos nombres
BANCO INTERMEDIARIO.(Clave,Nombre,Atencion,Direccion,Fax) + EMPRESA.(Clave,Nombre,Atencion,Direccion,Fax) + ACREEDOR.(Clave,Nombre,Atencion,Direccion,Fax)

Flow Expression of Data Flow Datos operacion
CONTRATO.(Folio,Monto,Plazo,Tipo) + BANCO INTERMEDIARIO.Clave + EMPRESA.Clave + ACREEDOR.Clave

Flow Expression of Data Flow Estadisticas
CONTRATO.(Monto,Plazo,Tipo) + ACREEDOR.(Clave,Nombre) + EMPRESA.(Clave,Nombre)

Flow Expression of Data Flow Fichas
CUENTA BANCO DE MEXICO.(Numero,Saldo)

Flow Expression of Data Flow Formatos
Nombres de campos + Longitudes de campos + Coordenadas de campos

Flow Expression of Data Flow Inscripcion
CONTRATO.suscrito por.EMPRESA

Flow Expression of Data Flow Instrucciones operacion
CONTRATO. + EMPRESA. + ACREEDOR. + CONTRATO.operado por.BANCO INTERMEDIARIO

Flow Expression of Data Flow Invitacion
Solicitud a autorizar

Flow Expression of Data Flow Movimientos consolidados
CONTRATO.Folio + BANCO INTERMEDIARIO.Clave

Flow Expression of Data Flow Movimientos contables
CONTRATO.Folio + BANCO INTERMEDIARIO.(Clave,Nombre) + ACREEDOR.(Clave,Nombre)

Flow Expression of Data Flow Nuevos datos operacion
Datos operacion

Flow Expression of Data Flow Operacion autorizada
Esqueleto contrato | Esqueleto modificacion

MINIESPECIFICACIONES

(NUEVO MODELO)

Carta modificatoria
Contrato

Input Flows

```
* REGISTRO DATOS OPERACION
    Propósito: Producir archivos con la información medular
    de los contratos. Estos archivos serán utilizados por todos
    los demas procesos.

Para cada OPERACION AUTORIZADA
  Si operacion = CONTRATO
      Resumir CONTRATO en DATOS OPERACION
      Extraer DATOS NOMBRES de CONTRATO
      Registrar en archivo MAESTRO DE CONTRATOS los DATOS OPERACION
    Si DATOS NOMBRES no registrados
        Registrar DATOS NOMBRES en CATALOGO UNICO

  Si OPERACION = CARTA MODIFICATORIA
      Extraer de CARTA MODIFICATORIA nuevos DATOS OPERACION
      Extraer de CARTA MODIFICATORIA nuevos DATOS NOMBRES
      Reescribir en archivo MAESTRO DE CONTRATOS nuevos DATOS OPERACION
    Si modificación = corrección
        Reescribir DATOS NOMBRES en CATALOGO UNICO
    Si modificación = cambio
      Si DATOS NOMBRES no registrados
          Registrar DATOS NOMBRES en CATALOGO UNICO
```

Datos nombres
Datos operacion

Output Flows

Datos nombres Input Flow
Datos operacion

* **PRODUCCION ESTADISTICAS**
Propósito: Generar información que refleje el estado de avance del programa

* **Se producen estadísticas globales**
(p.e. montos totales por CONTRATO.TIPO o ACREEDOR.TIPO)

Mientras no sea fin de archivo MAESTRO DE CONTRATOS
Leer un registro
Acumular cifras por CONTRATO.TIPO o ACREEDOR.TIPO

Generar cifras finales

* **Se producen estadísticas particulares**
(p.e. saldos por ACREEDOR o EMPRESA)

Leer ACREEDOR.NOMBRE o EMPRESA.NOMBRE de CATALOGO UNICO
Seleccionar registros de MAESTRO DE CONTRATOS a incluir
Mientras haya registros a incluir
Acumular sus cifras

Generar cifras finales

Estadísticas Output Flow

Produccion estadistica

October 5, 1992 17:30:00

Datos nombres Input Flow

```
* PRODUCCION FORMATOS Y CRITERIOS
      Propósito: Permitir al usuario la creación de reportes
      de acuerdo a sus necesidades particulares

*Produccion de formatos

Mientras MAS_CAMPOS = si
      Elegir NOMBRE DE CAMPO
      Elegir COORDENADAS DE CAMPO
      Determinar área ocupada en base a COORDENADAS DE CAMPO y LONGITUD DE CAMPO
      Preguntar por MAS_CAMPOS

Almacenar FORMATO
*Producción de criterios

Mientras MAS_CONDICIONES = si
      Elegir NOMBRE DE CAMPO
      Elegir OPERADOR DE COMPARACION
      Establecer VALOR DE CAMPO
      Preguntar por MAS_CONDICIONES

Almacenar CRITERIO
* Esta rutina debe ser flexible para poder utilizarse con otras bases de datos.
```

Formatos
Criterios
Output Flows

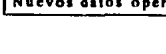
Produccion formatos y criterios

October 6, 1992 11:35:30


```
* CALCULO DE COBRO A EMPRESAS
    Propósito: Determinar los monto a cobrar periódicamente a las EMPRESAS

Para cada CONTRATO.FOLIO en MAESTRO DE CONTRATOS
    Determinar si se le cobra este periodo
    Si le toca pago
        Calcular amortización a cobrar
        Calcular intereses a cobrar
        Si CONTRATO al corriente
            Actualizar CONTRATO.SALDO en MAESTRO DE CONTRATOS
        En caso contrario
            Calcular amortizaciones vencidas
            Calcular intereses moratorios

        Extraer DATOS NOMBRES de CATALOGO UNICO
        Generar CIFRAS DEFINITIVAS
```




```
* calculo de cobro a empresas

(October 5, 1992  17:35:29
```

4.3 DISEÑO DEL SISTEMA.

Los Diagramas de Flujo de Datos integrantes de la Especificación Estructurada, desarrollada durante la fase de análisis, sirven como base para la construcción de las Gráficas de Estructura de los diferentes subsistemas, de acuerdo a la metodología de diseño estructurado de Yourdon.

En esta sección se muestran las Gráficas de Estructura de los tres principales subsistemas contenidos en el Dominio del Sistema de Directorios, generadas mediante la identificación de componentes aferentes, deferentes y de transformación de los respectivos Diagramas de Flujo de Datos.

Para comenzar se describe la Estructura del Registro de Contratos (figura 4.3.1), cuya rama deferente es de gran importancia dentro del contexto de este trabajo, puesto que en ella se conforman el archivo Maestro de Contratos y el Catálogo Unico. El segundo subsistema que se describe es el de Cálculo de Cifras ubicado dentro de la Operación de Contratos (figura 4.3.2), en esta gráfica es importante observar las ramas aferentes en donde se accesan los archivos centrales. Este tipo de accesos y por lo tanto el código contenido en los módulos correspondientes, es similar en otras partes del sistema.

Finalmente se incluye la Gráfica de Estructura del módulo de Explotación de Directorios, que por razones de presentación se separa en dos páginas, mostrándose en cada una la descomposición de una de las ramas principales, mientras que la otra permanece compactada, lo que se señala con dos puntos en su parte inferior. En la figura 4.3.3 puede observarse la composición del módulo de consultas, cuya rama de transformación efectúa la extracción de la información para el usuario. Por otra parte, la figura 4.3.4 permite apreciar los diferentes módulos que integran la Producción Estadística y en donde nuevamente es la rama deferente la que tiene relación con el Catálogo Unico.

Para facilitar el estudio de la estructura del sistema, los parámetros incluidos en las llamadas entre módulos sólo se indican mediante pequeñas flechas, sin embargo, para aquellas situaciones que requieren un análisis más profundo de dichos parámetros, IEW permite la impresión de Gráficas de Estructura con los parámetros de entrada y salida a cada módulo señalados explícitamente. Esta característica se muestra en las figuras 4.3.5 a 4.3.8 que, siendo iguales a las que las anteceden, muestran los nombres de los parámetros indicando si son de entrada o de salida mediante cabezas de flecha apuntando hacia abajo o hacia arriba respectivamente.

Con el enfoque de Yourdon, utilizado para el diseño de estos procesos, se construyen módulos que buscan ser lo suficientemente específicos para representar una función concreta como puede ser Escribir un Registro Maestro, y a la vez lo suficientemente básicos como para ser trasladados a código directamente.

Estas propiedades permiten que algunos módulos que se repiten en diferentes partes del sistema, queden perfectamente identificados, facilitando la reutilización de código que de otra manera hubiera sido duplicado con la consiguiente pérdida de tiempo y esfuerzo. Tal es el caso del módulo de Lectura de Datos del Catálogo, cuya función es leer del Catalogo Unico los datos almacenados bajo la clave que se le proporcione, sin importar si estos datos serán utilizados para reportar los datos del cobro, las estadísticas del sistema o una consulta particular.

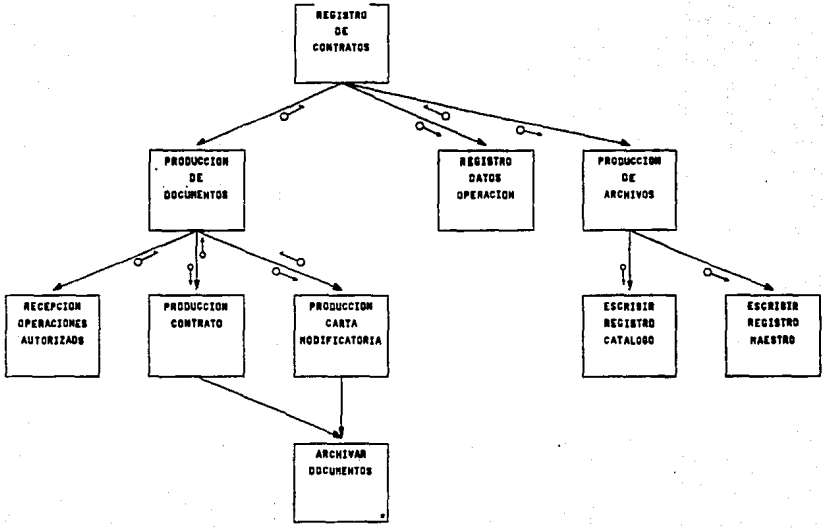

REGISTRO DE CONTRATOS
PRODUCCION DE DOCUMENTOS
REGISTRO DATOS OPERACION
PRODUCCION DE ARCHIVOS
RECEPCION OPERACIONES AUTORIZADS
PRODUCCION CONTRATO
PRODUCCION CARTA MODIFICATORIA
ESCRIBIR REGISTRO CATALOGO
ESCRIBIR REGISTRO MAESTRO
ARCHIVAR DOCUMENTOS

Figura 4.3.1

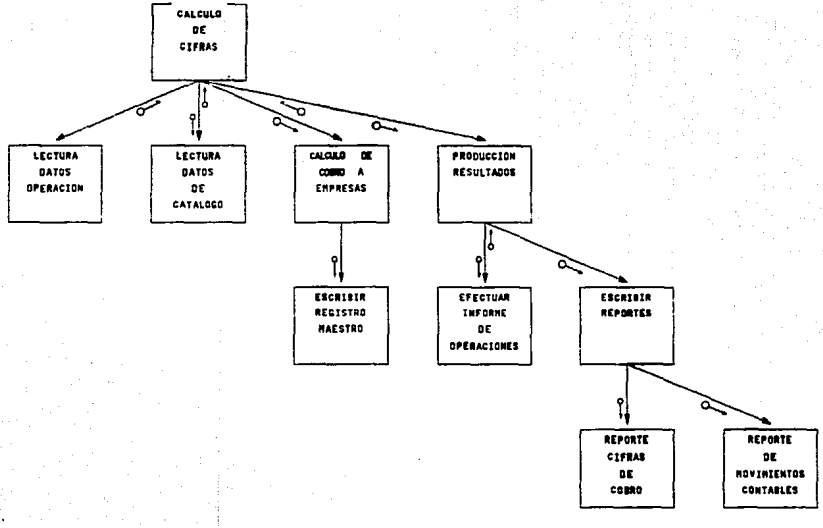

CALCULO
DE
CIFRAS
LECTURA
DATOS
OPERACION
LECTURA
DATOS
DE
CATALOGO
CALCULO DE
COBRO A
EMPRESAS
PRODUCCION
RESULTADOS
ESCRIBIR
REGISTRO
MAESTRO
EFECTUAR
INFORME
DE
OPERACIONES
ESCRIBIR
REPORTES
REPORTE
CIFRAS
DE
COBRO
REPORTE
DE
MOVIMIENTOS
CONTABLES

Figura 4.3.2

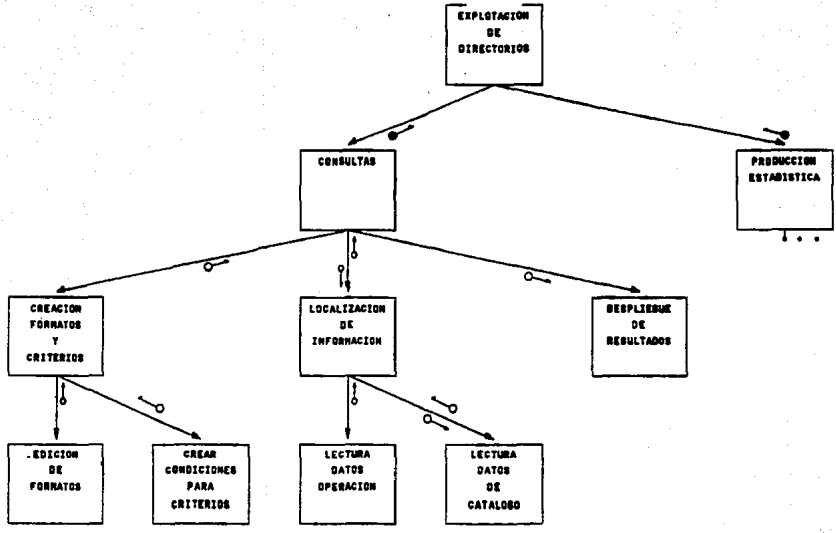

EXPLOTACION DE DIRECTORIOS
CONSULTAS
PRODUCCION ESTADISTICA
CREACION FORMATOS Y CRITERIOS
LOCALIZACION DE INFORMACION
DESPLIEGUE DE RESULTADOS
EDICION DE FORMATOS
CREAR CONDICIONES PARA CRITERIOS
LECTURA DATOS OPERACION
LECTURA DATOS DE CATALOGO

Figura 4.3.3

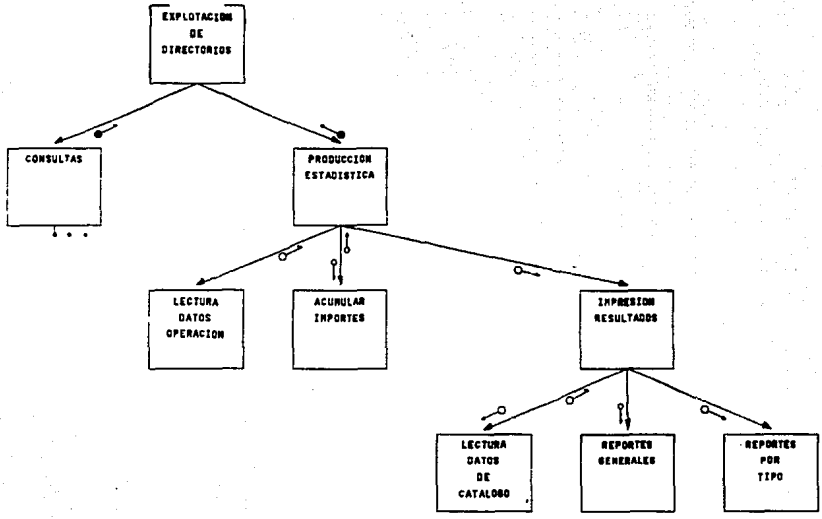

EXPLOTACION
DE
DIRECTORIOS
CONSULTAS
PRODUCCION
ESTADISTICA
LECTURA
DATOS
OPERACION
ACUMULAR
IMPORTES
IMPRESION
RESULTADOS
LECTURA
DATOS
DE
CATALOGO
REPORTES
GENERALES
REPORTES
POR
TIPO

Figura 4.3.4

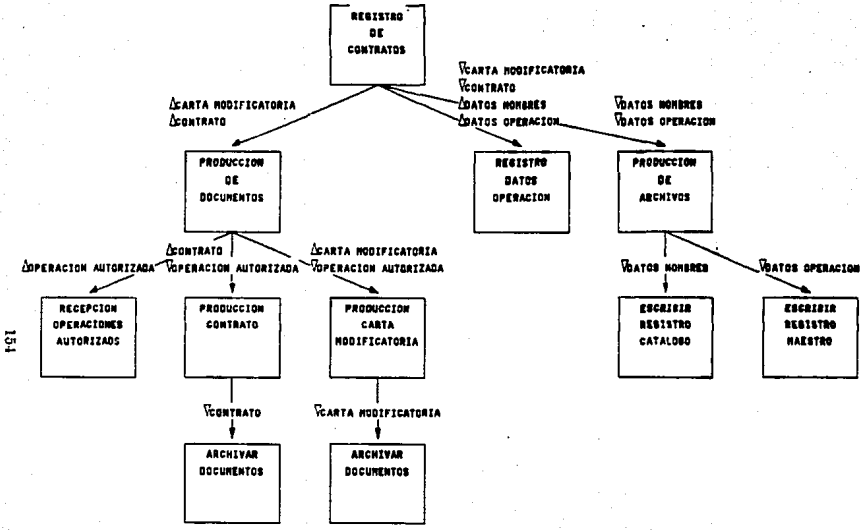
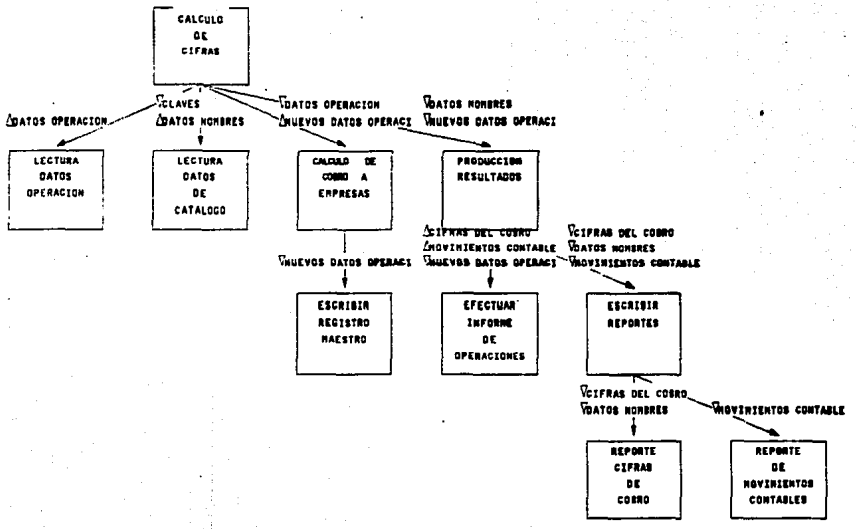

REGISTRO DE CONTRATOS
ΔCARTA MODIFICATORIA
ΔCONTRATO
∇CARTA MODIFICATORIA
∇CONTRATO
ΔDATOS HOMBRES
ΔDATOS OPERACION
∇DATOS HOMBRES
∇DATOS OPERACION
PRODUCCION DE DOCUMENTOS
REGISTRO DATOS OPERACION
PRODUCCION DE ARCHIVOS
ΔCONTRATO
ΔOPERACION AUTORIZADA
ΔCARTA MODIFICATORIA
∇OPERACION AUTORIZADA
∇OPERACION AUTORIZADA
∇DATOS HOMBRES
∇DATOS OPERACION
RECEPCION OPERACIONES AUTORIZADOS
PRODUCCION CONTRATO
PRODUCCION CARTA MODIFICATORIA
ESCRIBIR REGISTRO CATALOGO
ESCRIBIR REGISTRO MAESTRO
∇CONTRATO
∇CARTA MODIFICATORIA
ARCHIVAR DOCUMENTOS
ARCHIVAR DOCUMENTOS

Figura 4.3.5



CALCULO DE CIFRAS

December 4, 1992 10:56:29

Figura 4.3.6

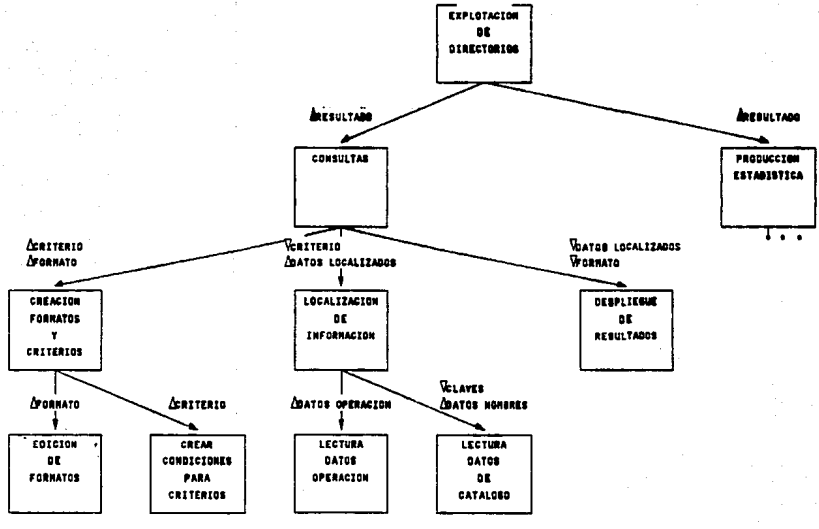

EXPLOTACION DE DIRECTORIOS
ΔRESULTADO
ΔRESULTADO
CONSULTAS
PRODUCCION ESTADISTICA
ΔCRITERIO
ΔFORMATO
∇CRITERIO
ΔDATOS LOCALIZADOS
∇DATOS LOCALIZADOS
∇FORMATO
CREACION FORMATOS Y CRITERIOS
LOCALIZACION DE INFORMACION
DESPLIEGUE DE RESULTADOS
ΔFORMATO
ΔCRITERIO
ΔDATOS OPERACION
∇CLAVES
ΔDATOS NOMBRES
EDICION DE FORMATOS
CREAR CONDICIONES PARA CRITERIOS
LECTURA DATOS OPERACION
LECTURA DATOS DE CATALOGO

Figura 4.3.7

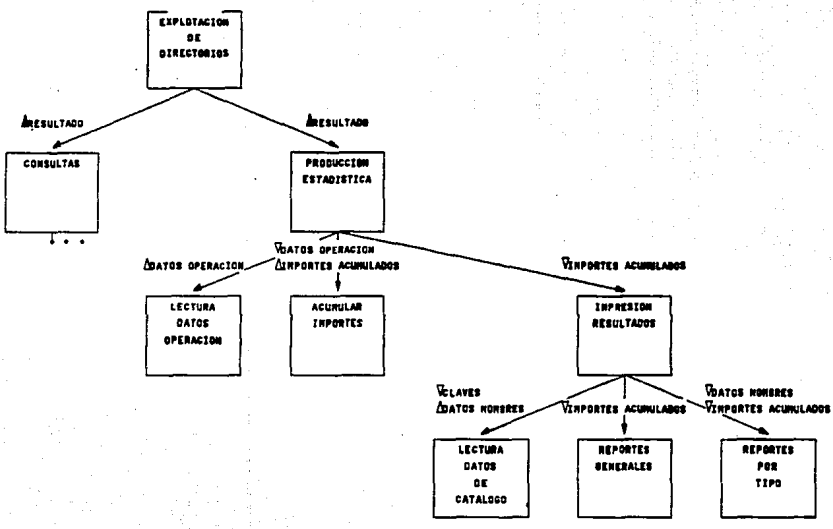

EXPLOTACION DE DIRECTORIOS
ΔRESULTADO
ΔRESULTADO
CONSULTAS
PRODUCCION ESTADISTICA
ΔDATOS OPERACION
∇DATOS OPERACION
ΔIMPORTES ACUMULADOS
∇IMPORTES ACUMULADOS
LECTURA DATOS OPERACION
ACUMULAR IMPORTES
IMPRESION RESULTADOS
∇CLAVES
ΔDATOS NOMBRES
∇IMPORTES ACUMULADOS
∇DATOS NOMBRES
∇IMPORTES ACUMULADOS
LECTURA DATOS DE CATALOGO
REPORTES GENERALES
REPORTES POR TIPO

Figura 4.3.8

4.4 HERRAMIENTAS DE IEW.

Todas las figuras presentadas en este capítulo, fueron elaboradas con las diferentes estaciones del ambiente CASE de IEW. Como puede apreciarse, cada estación proporciona los diagramas y reportes necesarios para el desarrollo de una etapa del ciclo de vida del sistema. Dentro de cada estación los diagramas y reportes se encuentran relacionados a través de la Enciclopedia, lo que permite que un cambio en cualquiera de ellos se refleje inmediatamente en todos los demás.

Adicionalmente, existen reportes auxiliares para todas las estaciones que amplían la información de los objetos incluidos en los diferentes diagramas, dando a conocer algunas características no contempladas en las gráficas, o bien comentarios generales relevantes.

A continuación se incluyen algunos comentarios relativos a los productos de cada estación de IEW.

Estación de Planeación. Para establecer el marco de referencia de un sistema, es necesario conocer el funcionamiento de la organización, determinando sus necesidades de información e identificando los problemas que la aquejan. En el segundo libro de su obra *Information Engineering*, James Martin propone que en la fase de planeación se deben precisar entre otras características de la organización las siguientes:

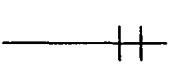
- Objetivos.
- Unidades Organizacionales que la integran.
- Funciones.
- Procesos.
- Problemas que la afectan.
- Factores de éxito críticos (Critical success factors).

- Entidades y relaciones.

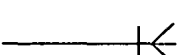
Todas ellas son conocidas como Objetos de la fase de Planeación y por lo general forman asociaciones entre sí (por ejemplo, ciertos procesos están a cargo de determinadas Unidades Organizacionales). Dado que estas asociaciones son de cardinalidad múltiple en ambos sentidos, una forma apropiada de representarlas es a través de matrices de asociación, en las cuales las columnas representan las instancias de un tipo de objeto y los renglones las de otro, y en las entradas de la matriz se aprecia una señal cuando existe una relación entre los objetos correspondientes.

La estación de Planeación permite incluir todos los objetos mencionados por Martin en su teoría, y una de sus principales herramientas gráficas está constituída por las matrices de asociación como las que se muestran en la sección 4.1.

Para completar el modelo de la situación de la empresa, la otra herramienta gráfica relevante de esta estación son los diagramas Entidad-Relación, que permiten representar las relaciones existentes entre las diferentes fuentes de información, indicando además la cardinalidad mínima y máxima de dichas relaciones de la siguiente manera: una relación uno a uno se indica con dos barras paralelas sobre los extremos de la línea que representa a la relación, mientras que una relación de uno a muchos se indica dividiendo en tres el extremo de dicha línea. (ver diagrama 4.1).



uno a uno



uno a muchos

Para las entidades puede generarse un reporte descriptivo que incluye todos sus atributos y las relaciones en que está involucrada. Aquellos atributos que identifican a la entidad de manera única se señalan con las letras ID en la columna izquierda (ver figuras 4.7 a 4.9).

Estación de Análisis. Los Diagramas de flujo de datos y las miniespecificaciones empleados en la construcción de la Especificación Estructurada de sistema no requieren mayores comentarios ya que siguen las pautas establecidas por la teoría de Análisis Estructurado de Tom De Marco. Por lo tanto solo se describirán los reportes auxiliares con los que cuenta esta sección.

Como se mencionó en el tercer capítulo, el Reporte de Conservación de datos señala aquellos nodos que violan la regla de conservación de los datos dentro de los diagramas de flujo. En la figura 4.4.1, localizada al final de esta sección puede observarse un reporte de este tipo obtenido para el proceso Cálculo de Cifras Nuevo.

El reporte de Análisis de Trayectorias de Datos indica cuál es el camino seguido por un flujo de datos dado, identificando todos los nodos por los que pasa y resaltando su origen y destino final. La figura 4.4.2 es un reporte de este tipo obtenido para el flujo Datos Nombres.

Los reportes que se muestran en esta sección fueron generados restringiendo los objetos sobre los que actúan al mínimo, pero es posible producirlos de manera que contemplen todos los objetos de la enciclopedia, o todos los objetos presentes en un diagrama en particular.

Estación de Diseño. Como puede apreciarse, la estación de diseño de IEW soporta la creación de Gráficas de Estructura para el diseño de procesos. Una vez definida la estructura general, se dispone de una herramienta para la especificación del pseudocódigo de los módulos básicos, de manera completamente similar a la utilizada en la estación de análisis para escribir las miniespecificaciones.

Para auxiliar en la definición de las gráficas de estructura, la estación de diseño de IEW proporciona dos reportes principales: el primero, llamado *Call Analysis* compara los parámetros de llamada de un módulo con sus parámetros formales, reportando todas las incongruencias u omisiones. El segundo reporte es conocido como *Module Analysis* se encarga de listar aquellas características que no han sido definidas

para los módulos examinados. En las siguientes páginas puede observarse una muestra de cada uno de estos reportes, obtenidas del ejemplo desarrollado en este capítulo. (Figuras 4.4.3 y 4.4.4).

Por otro lado, una parte importante de la estación de diseño está dedicada al diseño de bases de datos. En ella pueden generarse Diagramas de Estructura de Datos, Diagramas de Definición de Bases de Datos, Diagramas para Descripción de Archivos y Diagramas de Definición de Bases de Datos Relacionales entre otros. Con esto se busca que el trabajo efectuado a lo largo de las diferentes estaciones culmine tanto con la creación de los procesos necesarios para llevar a cabo las funciones del sistema, como de las estructuras de datos requeridas para sustentar dichos procesos.

En el presente capítulo no se profundiza en el diseño de estructuras de datos. dado que el desarrollo de la teoría y la práctica de este aspecto del desarrollo de sistemas ameritan por sí mismos un trabajo aparte, por lo tanto no se describen los diagramas mencionados anteriormente.

Opciones Generales. Entre los reportes comunes a todas las estaciones destacan los siguientes: el Reporte Resumen de Objetos que describe todas las propiedades y asociaciones de cada instancia de un tipo de objeto seleccionado, y el Reporte de Excepciones que indica la información que no ha sido registrada para los objetos seleccionados.

Del primer tipo se incluye la figura 4.4.5 con la descripción integral de un proceso. Del segundo tipo se generó el Reporte de Excepciones para los objetos del tipo entidad. verificando si fue registrada la información relativa a su definición y comentarios y si se encuentran asociados a algún proceso (Figura 4.4.6).

El reporte llamado Lista de Objetos, despliega todos los objetos de un mismo tipo contenidos en la Enciclopedia. con el fin de permitir una consulta rápida. La figura 4.4.7 es una lista de los objetos del tipo Atributo.-

En la figura 4.4.8 puede observarse el Reporte Resumen de la Enciclopedia, que nos muestra el contenido de ésta en un momento dado. Dado que la enciclopedia sirve de conexión entre los módulos de Planeación, Análisis y Diseño, este reporte muestra objetos contenidos en las tres estaciones.

Un último punto que merece destacarse es el control de actualizaciones que establece IEW, registrando la fecha, la hora y la clave del usuario que efectuó la modificación al objeto, así como la fecha y hora de producción de cada impresión, con lo que el control de proyectos puede hacerse tan rígido como se desee, ya que además cada usuario puede contar con una clave de acceso única.

BIBLIOGRAFIA DEL CAPITULO

- James Martin, "Information Engineering Book II Planning and Analysis", Prentice Hall, 1990.

- KnowledgeWare, "Information Engineering Workbench/Workstation Planning Workstation User Guide", Release 5.0, Septiembre 1988.

- KnowledgeWare, "Information Engineering Workbench/Workstation Analysis Workstation User Guide", Release 5.0, Septiembre 1988.

- KnowledgeWare, "Information Engineering Workbench/Workstation Design Workstation User Guide", Release 5.0, Septiembre 1988.

Data Conservation Analysis

Process Calculo de cifras nuevo
in context of Process OPERACION DE CONTRATOS NUEVO

Incoming data flows:
Data Flow Datos operacion
Data Flow Datos nombres
Data Flow Aviso de movimiento

Outgoing data flows:
Data Flow Nuevos datos operacion
Data Flow Cifras definitivas

Data Conservation Failure
Process Calculo de cifras nuevo involves
Relationship Type
Entity Type EMPRESA
suscriptora de
Entity Type CONTRATO
not involved in any Incoming, Outgoing, or Internal Data Flow

Data Conservation Failure
Process Calculo de cifras nuevo involves
Relationship Type
Entity Type BANCO INTERMEDIARIO
operador de
Entity Type CONTRATO
not involved in any Incoming, Outgoing, or Internal Data Flow

Figura 4.4.1

Flow Course Analysis Report

Data Flow Datos nombres

Location
Process OPERACION DE CONTRATOS NUEVO

Immediate Source
Junction
in context of Process OPERACION DE CONTRATOS NUEVO

Immediate Destination
Process Explotacion de directorios
in context of Process OPERACION DE CONTRATOS NUEVO

Ultimate Source:
Data Store Catalogo unico
in context of Process Dominio sistema de directorios

Ultimate Destination:
Sequential Process Produccion estadistica
in context of Process Explotacion de directorios

Data Flow Course:

Process Dominio sistema de directorios
Process OPERACION DE CONTRATOS NUEVO
Process Explotacion de directorios

Ultimate Source:
Data Store Catalogo unico
in context of Process Dominio sistema de directorios

Ultimate Destination:
Sequential Process Produccion formatos y criterios
in context of Process Explotacion de directorios

Data Flow Course:

Process Dominio sistema de directorios
Process OPERACION DE CONTRATOS NUEVO
Process Explotacion de directorios

Figura 4.4.2

Call Analysis

Report will show all formal and calling parameters

Modules THAT CALL Module REGISTRO DE CONTRATOS

Modules CALLED BY Module REGISTRO DE CONTRATOS

Module PRODUCCION DE ARCHIVOS

Call Number 1, call type: EXTERNAL SYNCHRONOUS

Formal Parameters	Calling Parameters
DATOS NOMBRES - DATOS OPERACION	DATOS NOMBRES
DATOS OPERACION	DATOS OPERACION

Module PRODUCCION DE DOCUMENTOS

Call Number 1, call type: EXTERNAL SYNCHRONOUS

Formal Parameters	Calling Parameters
CARTA - CONTRATO	CARTA MODIFICATORIA
CONTRATO	CONTRATO

Module REGISTRO DATOS OPERACION

Call Number 1, call type: EXTERNAL SYNCHRONOUS

Formal Parameters	Calling Parameters
CARTA - CONTRATO	CARTA MODIFICATORIA
DATOS NOMBRES - DATOS OPERACION	DATOS NOMBRES
DATOS OPERACION	DATOS OPERACION
CONTRATO	CONTRATO

Figura 4.4.3

Module ESCRIBIR REGISTRO CATALOGO

Action Diagram is empty except for formal parameters.

Module ESCRIBIR REGISTRO MAESTRO

Action Diagram is empty except for formal parameters.

Module PRODUCCION DE ARCHIVOS

These calls were added since the module was last saved:
Call to Module ESCRIBIR REGISTRO MAESTRO
Call to Module ESCRIBIR REGISTRO CATALOGO

Module REGISTRO DATOS OPERACION

Action Diagram is empty except for formal parameters.

Module REGISTRO DE CONTRATOS

These calls were added since the module was last saved:
Call to Module PRODUCCION DE DOCUMENTOS
Call to Module REGISTRO DATOS OPERACION
Call to Module PRODUCCION DE ARCHIVOS

Figura 4.4.4

Object Summary Report

Process: Calculo de cobro a empresas

Definition
Calculo del cobro periodico que se hara a las empresas

Comments
Este calculo es efectuado en paralelo por el banco intermediario

PROPERTY	VALUE
Last Update	1992/05/27 18:37 ANTONIO
Created	1992/05/22 19:56:42 ANTONIO
Importance	H
System Support Type	IN

ASSOCIATION	TYPE	NAME
Involves	Entity Type	EMPRESA
		BANCO INTERMEDIARIO
Is Part Of	Process	Procesos de credito
Is Source of	Data Flow	Nuevos datos operacion
		Cifras pago
Is Destination of	Data Flow	Datos operacion
		Datos nombres
Is Responsibility of	Organizational Unit	OPERACION DE CONTRATOS
Is Affected by	Problem	Elevado num. de casos especiales
		Falta coordinacion en oficinas
		Falta de capacitacion
		Gran volumen de operaciones
		Incongruencia entre directorios
		Registro separado de operaciones

Figura 4.4.5

Exception Report

Report for Entity Type object types.

Required?	Property Type
Yes	Purpose
Yes	Definition
Yes	Comments

Min	Max	Association Type
0	M	Entity Type causes Problem
0	M	Entity Type is available at Location
0	M	Entity Type supports Critical Assumption
0	M	Entity Type supports Critical Success Factor
0	M	Entity Type supports Information Need
0	M	Entity Type supports Goal
0	M	Entity Type is affected by Problem
0	M	Entity Type is included in Project
1	M	Entity Type is involved in Process
0	M	Entity Type is involved in Subject Area
0	M	Entity Type is involved in Data Collection
0	M	Entity Type is responsibility of Organizational Unit
0	M	Entity Type source is Modeling Source

Object Instance = ACREEDOR

Object Instance = BANCO INTERMEDIARIO

Object Instance = CONTRATO
Missing Property: Comments
Min: 1 Act: 0 For: Entity Type is involved in Process

Object Instance = CREDITO
Min: 1 Act: 0 For: Entity Type is involved in Process

Figura 4.4.6

Object Instance = CUENTA BANCARIA
Missing Property: Definition
Missing Property: Comments
Min: 1 Act: 0 For: Entity Type is involved in Process

Object Instance = CUENTA BANCO DE MEXICO
Missing Property: Definition
Missing Property: Comments
Min: 1 Act: 0 For: Entity Type is involved in Process

Object Instance = EMPRESA

Figura 4.4.6 (Cont.)

Object List

Attribute Type ACREEDOR.Atencion 1992/06/01 18:46 ANTONIO
Attribute Type ACREEDOR.Clave 1992/06/01 18:43 ANTONIO
Attribute Type ACREEDOR.Direccion 1992/06/01 18:47 ANTONIO
Attribute Type ACREEDOR.Fax 1992/06/01 18:48 ANTONIO
Attribute Type ACREEDOR.Nombre 1992/06/01 18:43 ANTONIO
Attribute Type ACREEDOR.Telefono 1992/06/01 18:47 ANTONIO
Attribute Type BANCO INTERMEDIARIO.Atencion 1992/06/01 19:01 ANTONIO
Attribute Type BANCO INTERMEDIARIO.Clave 1992/06/01 18:59 ANTONIO
Attribute Type BANCO INTERMEDIARIO.Direccion 1992/06/01 19:01 ANTONIO
Attribute Type BANCO INTERMEDIARIO.Fax 1992/06/01 19:03 ANTONIO
Attribute Type BANCO INTERMEDIARIO.Nombre 1992/06/01 19:00 ANTONIO
Attribute Type BANCO INTERMEDIARIO.Telefono 1992/06/01 19:02 ANTONIO
Attribute Type CONTRATO.firma 1992/07/09 18:06 ANTONIO
Attribute Type CONTRATO.Folio 1992/06/01 18:52 ANTONIO
Attribute Type CONTRATO.Monto 1992/06/01 18:53 ANTONIO
Attribute Type CONTRATO.Plazo 1992/06/01 18:54 ANTONIO
Attribute Type CONTRATO.Tipo 1992/06/01 18:54 ANTONIO
Attribute Type CREDITO.Divisa 1992/06/01 18:35 ANTONIO
Attribute Type CREDITO.Monto 1992/06/01 18:33 ANTONIO
Attribute Type CREDITO.Registro 1992/06/01 18:29 ANTONIO
Attribute Type CREDITO.Vencimiento 1992/06/01 18:37 ANTONIO
Attribute Type CUENTA BANCARIA.Divisa 1992/06/01 19:05 ANTONIO
Attribute Type CUENTA BANCARIA.Numero 1992/06/01 19:05 ANTONIO
Attribute Type CUENTA BANCARIA.Saldo 1992/06/01 19:06 ANTONIO
Attribute Type CUENTA BANCO DE MEXICO.Divisa 1992/06/01 19:09 ANTONIO
Attribute Type CUENTA BANCO DE MEXICO.Numero 1992/06/01 19:08 ANTONIO
Attribute Type CUENTA BANCO DE MEXICO.Saldo 1992/06/01 19:10 ANTONIO
Attribute Type EMPRESA.Atencion 1992/06/01 18:23 ANTONIO
Attribute Type EMPRESA.Clave 1992/06/01 18:20 ANTONIO
Attribute Type EMPRESA.Direccion 1992/06/01 18:25 ANTONIO
Attribute Type EMPRESA.Fax 1992/06/01 18:27 ANTONIO
Attribute Type EMPRESA.Nombre 1992/06/01 18:22 ANTONIO
Attribute Type EMPRESA.Telefono 1992/06/01 18:26 ANTONIO

Figura 4.4.7

Encyclopedia Summary Report

Object counts:

33	attribute types
0	critical assumptions
0	critical success factors
0	data collections
33	data flows
12	data stores
7	data structure templates
1	data types
0	DBDs
7	entity types
4	external agents
0	flat file databases
0	functions
0	goals
0	information needs
68	junctions
0	libraries
0	locations
0	mechanisms
0	modeling sources
0	modules
9	organizational units
0	problems

Figura 4.4.8

33 processes

0 projects

0 PSBs

0 records

27 relations

1 relational databases

11 relationship types

0 screens

0 segments

19 sequential processes

0 subject areas

Planning association counts:

17 Process consists of Process

41 Process involves Entity Type

8 Organizational Unit manages Organizational Unit

Figura 4.4.8 (Cont.)

CONCLUSIONES Y EXPECTATIVAS.

Trabajar con una herramienta CASE del tipo de IEW representa aceptar una serie de cambios en la forma en que tradicionalmente se desarrollan los sistemas en gran número de organizaciones. Algunos de estos cambios tienen un radio de acción relativamente pequeño, pero otros son cambios de fondo que implican una mentalidad diferente.

Para comenzar, el analista o grupo de analistas, se obligan a realizar un estudio detallado del medio ambiente donde operarán los sistemas antes de comenzar el desarrollo de éstos. Es decir, la planeación y el análisis adquieren la importancia que generalmente se les ha negado. Debido a lo anterior, los resultados prácticos del desarrollo del sistema, representados para una mayoría por el código de los programas, parecen tardar más en aparecer, haciendo indispensable la comprensión de los supervisores hacia el proyecto.

La capacitación es uno de los factores principales que deben ser tomados en cuenta al planear la adopción de una herramienta CASE. El estudio de la Ingeniería de Sistemas, de las metodologías estructuradas para el análisis y diseño de sistemas, de las diferentes técnicas de diagramación e incluso de métodos auxiliares para comunicarse con los usuarios mediante entrevistas o cuestionarios, debe efectuarse con anterioridad al empleo de las herramientas contenidas en cualquier ambiente integrado CASE, pues de lo contrario éste no podría explotarse por completo.

Como puede verse, la adopción de una herramienta CASE para el desarrollo de sistemas dentro de una institución, es un plan que debe contemplarse a mediano plazo, ya que la obtención de resultados inmediatos es prácticamente nula. Para que los beneficios comiencen a ser tangibles, es necesaria una continuidad en el proyecto, seleccionando un equipo entusiasta que comprenda que su esfuerzo dará frutos importantes en el transcurso de uno o dos años dependiendo del tamaño y complejidad de la institución.

Adicionalmente, debido a sus características, las herramientas CASE requieren de recursos de cómputo considerables, entre los que pueden mencionarse: un procesador razonablemente veloz, una cantidad considerable de espacio en disco y de memoria disponible y facilidades gráficas en el video y la impresora.

En vista de los razonamientos anteriores es válido preguntarse si vale la pena el esfuerzo de emplear herramientas CASE. Regresemos a la problemática de los sistemas de información comentada en el primer capítulo de este trabajo y veamos cuáles de los problemas allí mencionados pueden ser atacados con herramientas CASE.

Con el soporte del ciclo de vida y soporte de metodologías estructuradas de un ambiente CASE se reducen los problemas de Planeación y estimación inadecuadas, análisis insuficiente y ausencia de diseño. Con el repositorio central de información desaparecen la comunicación deficiente entre equipos de trabajo y la disparidad de criterios al desarrollar, así como la documentación desactualizada. Con la capacidad gráfica de estas herramientas mejora sensiblemente la comunicación entre analistas y usuarios, obteniéndose documentos tan importantes como el mismo código que a la vez facilitan el mantenimiento posterior del sistema.

En general es posible afirmar que el empleo de herramientas CASE incrementa la calidad general de los sistemas al facilitar la aplicación automática de conceptos que hasta ahora sólo era posible llevar a la práctica con un esfuerzo considerable.

En lo referente a requisitos de hardware, este punto no debe considerarse un obstáculo, ya que la industria los está proporcionando en un volumen cada vez mayor a un costo cada vez menor.

Por lo tanto puede decirse que el uso de herramientas CASE por parte de los grupos de desarrollo de sistemas es ampliamente recomendable, sobre todo en aquellas organizaciones con un alto grado de complejidad y con equipos de trabajo numerosos.

EXTENSIONES A ESTE TRABAJO.

Las extensiones inmediatas a este trabajo se encuentran dentro del mismo ambiente integrado CASE de IEW. Como se mencionó en su oportunidad. algunas herramientas no fueron incluidas para evitar que el texto se extendiera en demasía. Las más interesantes de entre ellas son las que permiten el diseño de bases de datos y las que soportan la definición de prototipos.

Por otro lado, existe una metodología estructurada que está tomando gran impulso actualmente por lo que sería justo su desarrollo detallado. Me refiero a la Ingeniería de Información de James Martin que permite establecer una visión más amplia del desarrollo de sistemas, al comenzar desde un nivel mucho más general que cualquiera de las otras.

Finalmente debe destacarse que la fuerte corriente representada por el análisis y diseño orientado a objetos, está influyendo en las herramientas CASE, de manera que se maneja ya el soporte de esta corriente en las versiones más recientes de algunas de ellas.