UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

Metodología para el Desarrollo de Sistemas Usando Lenguajes de Cuarta Generación.

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N:

G. ADRIANA PATRICIA CABRERA RIOS
RODRIGO ARTURO GOMEZ LAVANDEROS
CARLOS RAMIREZ ZUÑIGA

DIR. ING. GUSTAVO ORIGEL COUTIÑO

CD. UNIVERSITARIA 1990.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

METODOLOGIA PARA EL
DESARROLLO DE SISTEMAS
USANDO LENGUAJES DE CUARTA GENERACION

I N D I C E

I N D I C E

I.- INTRODUCCION

I. INTRODUCCION.

El objetivo del presente trabajo, es proporcionar a la gente de "Desarrollo de Sistemas" un documento, que muestre los medios para desarrollar eficientemente un sistema en ambientes de "Cuarta Generación".

Hemos investigado todos los elementos necesarios para el desarrollo de un sistema con Lenguajes de Cuarta Generación, en todas y cada una de sus etapas o fases. Así mismo, hemos introducido el desarrollo con "Prototipos", que de acuerdo a nuestra metodología, es algo necesario y muy importante, ya que de lo contrario, se caería en el desarrollo de sistemas en forma tradicional, y en este sentido no tendría caso usar Lenguajes de Cuarta Generación que es lo que permite implementar rápidamente el desarrollo con "Prototipos".

Con la incursión de los Prototipos en esta Metodología para el Desarrollo de Sistemas, estamos dando una forma consistente y avanzada para el Desarrollo de Sistemas en ambientes de Cuarta Generación.

Un Prototipo es un producto de nuestro diseño en base a la concepción inicial de las necesidades del usuario, el cual, va a contener una implementación de cada función o subsistema, pero, a nivel representativo de la operación del sistema para que posteriormente, con los ajustes necesarios, pase a ser la función final. El prototipo de sistemas se desarrolla con la intención de obtener la información necesaria para formular el diseño y desarrollar el sistema, implicando el desarrollo de un software, que corra y produzca información; este prototipo contendrá los datos, sus atributos y relaciones del sistema.

Las razones que soportan el uso de prototipos van desde cómo éstos ilustran los formatos de entrada, mensajes, informes y diálogos al usuario; así como, explotar aspectos técnicos del producto propuesto, y también en algunas ocasiones, no es posible definir el producto sin un desarrollo exploratorio y en ocasiones es necesario desarrollarlo completamente.

Proponemos que el prototipo contenga Análisis, Diseno Conceptual, Diseno Detallado, Desarrollo, Implementación y Posimplementación. Estas fases realizan una "trayectoria en espiral", es decir, que cada una de éstas son implementadas, probadas y si no es el producto deseado, se ajustan las fases que sean necesarias y se vuelve a probar, si todavía no es lo que se desea, se repite el proceso las veces que sean necesarias, hasta obtenerse un producto final. A este tipo de trayectoria que describe el proceso se denomina "En Espiral".

Esta metodología está orientada al uso de Lenguajes de Cuarta Generación; por las facilidades que presentan para la implementación de los prototipos, ya que con los elementos con que cuentan estos lenguajes, permiten elaborar rápidamente alguna aplicación.

II.- .QUE ES UN LENGUAJE DE CUARTA GENERACION?

II.1 ANTECEDENTES.

Los lenguajes de tercera generación aparecieron en la década de los 60's y se les llamó lenguajes de alto nivel. Algunos de ellos fueron utilizados para trabajo científico, como el ALGOL Y FORTRAN, otros para trabajo comercial, como COBOL, él cual se convirtió rápidamente en el lenguaje de computación más comúnmente utilizado.

Con la tercera generación, el lenguaje se volvió más independiente del Hardware. Un programador podía codificar programas, sin ningún conocimiento de la máquina.

Una instrucción de 3a. generación es usualmente compilada en varias instrucciones de lenguaje de máquina. La cantidad de trabajo necesaria para escribir programas fué reducida así, sin embargo, necesitaban muchas líneas de código para sistemas comerciales y eran mejor diseñados por profesionales del procesamiento de datos que por el usuario final, consumían mucho tiempo y se dificultaba demasiado la modificación de sistemas complejos.

Estos lenguajes usan principalmente construcciones como las de VON NEUMANN, que expresan una secuencia de operaciones para ser ejecutadas con saltos e iteracciones, sus operaciones básicas son muy similares al conjunto de instrucciones de máquina.

A continuación se muestra la figura 2.1, que ejemplifica las características esenciales de los lenguajes de primera, segunda y tercera generación.

PRIMERA GENERACION	SEGUNDA GENERACION	TERCERA GENERACION
- Lenguajes de Máquina.	- Lenguajes Ensambladores.	- Lenguajes de Alto Nivel.
- Relación uno a uno de las instrucciones de programa y el código generado.	- Permiten el uso de mnemónicos.	- Amplio poder generativo.
- Conocimiento exacto de la ubicación de los datos en memoria.	- Se inicia la utilización de macro instrucciones.	- Lenguajes un tanto independientes del proveedor del Hardware.
- Personal altamente especializado en el hardware y software del proveedor.	- Conocimiento exacto de la ubicación de los datos en memoria.	- Alta flexibilidad de programación.
- Responsabilidad del programador el definir todas las operaciones de lógica con lujo de detalle.	- Personal altamente especializado en el Hardware y software del proveedor.	- La programación al alcance de más gente.
- Un programa para cada función del sistema.	- Responsabilidad del programador el definir todas las operaciones de lógica con lujo de detalle.	- Interfase con lenguajes de aplicación específica.
- Sistemas orientados a procesos batch.	- Un programa para cada función del sistema.	- Crean la necesidad de nuevas metodologías de análisis y diseño.
- Documentación extensa y compleja.	- Sistemas orientados a procesos batch.	- Facilitan el peligro de crear lógica muy compleja.
	- Documentación extensa y compleja.	- Alta supervisión de estándares.
		- Documentación extensa y compleja.

FIGURA 2.1. CARACTERISTICAS DE LA TRES PRIMERAS GENERACIONES DE LOS LENGUAJES DE PROGRAMACION.

II.2 EL TIEMPO DEL CAMBIO.

A finales de los 70's es un punto de grandes cambios en herramientas y técnicas para la elaboración de sistemas por computadora. Las técnicas nuevas y poderosas que surgieron se ilustran en la figura 2.2, dichas técnicas afectan fuertemente la construcción de lenguajes más productivos.

1950 : CODIGO DE MAQUINA.

1960 : LENGUAJE ENSAMBLADOR.

1970 : LENGUAJES DE ALTO NIVEL.

1980 : LENGUAJES DE CUARTA GENERACION.

- Lenguajes No-procedurales.
- Lenguajes de Especificación.
- Desarrollo Automatizado.
- Técnicas de Prototipos.
- Ingeniería de Información.
- Programación para el usuario final.
- Estaciones de Trabajo Poderosas.

1990 : SISTEMAS BASADOS EN EL CONOCIMIENTO.

- Programación basada en reglas.
- Sistemas de Inteligencia Artificial.
- Sistemas Expertos.

FIGURA 2.2. GENERACIONES DE LOS LENGUAJES.

II.3 TIPOS DE LENGUAJES DE PROGRAMACION.

A principios de 1980, la mayoría de los programadores utilizaban lenguajes procedurales, aunque diferentes en sintaxis, eran ampliamente similares en la estructura general, tal como: COBOL, PL/1, FORTRAN, PASCAL, BASIC, ADA y algunos más, que utilizan las construcciones de control que se presentan en la figura 2.3. Hay una variedad de tipos de programación, por ejemplo procedural, non-procedural o ambos, ya sea para ambientes interactivos o no. A continuación se describen diferentes tipos de lenguajes de programación.

II.3.1 LENGUAJES PROCEDURALES.

Un lenguaje que proporciona un conjunto de instrucciones que indican a la computadora lo que debe ejecutar dentro de una secuencia especificada, se le llama lenguaje procedural, esta secuencia puede variar, dependiendo de los grupos de instrucciones que se ejecuten repetidamente. Las construcciones de control gobiernan la secuencia de ejecución de instrucciones. Éstas instrucciones son ilustradas en la figura 2.3, todos los lenguajes de la primera, segunda y tercera generación son de este tipo o se consideran como tales.

II.3.2 LENGUAJES DE PROCEDIMIENTOS ESTRUCTURADOS.

Para que un lenguaje sea considerado como tal, debe de permitir todas las construcciones de la figura 2.3, con excepción de los números ocho y nueve, que no son permitidos. El ejemplo siguiente muestra el código estructurado de un lenguaje de cuarta generación.

```
<<EMP>> FOR EACH EMPLOYEE
        WHERE DEPT = 'D' AND JOB-CODE = 'J'
          DO NOTE-DJ-EMP
  <<DEP>> FOR EACH DEPENDENT
            DO NOTE-DEP
            IF DEP-AGE > 21
               DO TOO-OLD
               PROCESS NEXT DEP
            ENDIF
            DO ANAL-DEP
          ENDFOR
          IF FOUND-ENOUGH-EMP
             QUITE EMP
          ENDIF
        ENDFOR
```

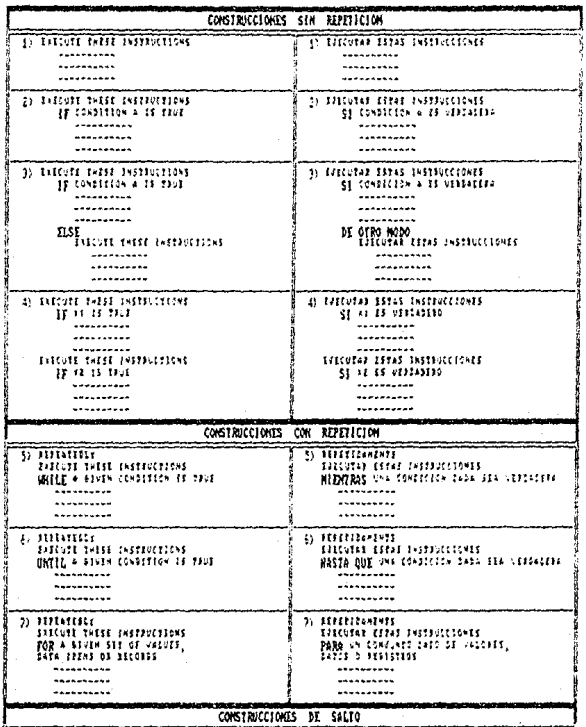


FIGURA 2.3. CONSTRUCCIONES DE CONTROL.

II.3.3 LENGUAJES NON-PROCEDURALES.

Un lenguaje que describe los resultados que serán obtenidos, pero no especifica la secuencia exacta de pasos (procedimientos) para lograrlo, se le llama lenguaje non-procedural.

Ejemplo:

```
GENERATE A MONTHLY PLOT
TITLE "SALES VOLUME BY DISTRICT"
X AXIS LABEL "MONTH OF 1982"
Y AXIS LABEL "MILLIONS"
INPUT DATA
"WEST"
SELECT QUANTITY FROM SALES
WHERE DISTRICT = WEST AND YEAR = 1982
"EAST"
SELECT QUANTITY FROM SALES
WHERE DISTRICT = EAST AND YEAR = 1982
GO
```

Los lenguajes non-procedurales pueden tener una gran variedad de sintaxis.

II.3.4 LENGUAJES FUNCIONALES DE PROGRAMACION.

Es un lenguaje, en el cual las expresiones se componen de varias llamadas a diversas funciones, más que utilizar una secuencia lógica de instrucciones o declaraciones.

Ejemplo:

```
La función : SUB1 N regresa (N - 1)
             FACTORIAL (SUB1 N) regresa (N - 1)!
             TIMES N (FACTORIAL (SUB1 N)) regresa N(N - 1)!
             COND (E1 X) (E2 Y) regresa X si E1 es TRUE,
             de otra forma Y si E2 es TRUE
```

El código mostrado es un ejemplo del lenguaje LISP, considerado una herramienta de quinta generación (SISTEMAS EXPERTOS).

II.3.5 LENGUAJES DECLARATIVOS.

Es un lenguaje que declara un conjunto de realidades o hechos; permite que sean utilizados por instrucciones de pregunta, y cuenta con una secuencia especificada de pasos.

Ejemplo:

```
PRINCIPAL = INSTALLMENT X 100/INTEREST
RATE(1   + INTEREST RATE/100) * * - N) (1 +
INTEREST RATE/100)
```

II.3.6 LENGUAJES BASADOS EN REGLAS.

Es un lenguaje en el cual un conjunto de realidades o hechos y reglas descritas, son usadas como respuesta a interrogantes para la solución de diversos problemas. Prolog es un ejemplo de un lenguaje basado en reglas, es decir:

```
female(mary)
male(john)
like(john,mary)
book(future,shock,toffler,random house)
```

las reglas son expresadas como sigue:

```
sister(x,y): female(x),parent(x,z),parent(y,z)
```

Esto significa que "x" es la hermana de "y" , si "x" es femenino y es pariente de "z", es como el mismo pariente de "y".

II.3.7 LENGUAJES ORIENTADOS A OBJETOS.

Los lenguajes de programación convencionales, especifican una secuencia de operaciones determinada, existen funciones poderosas que son construidas fuera de la secuencia del programa, que posteriormente son llamadas por diferentes módulos, en lugar de utilizar diversas operaciones y operandos dentro del programa. Los lenguajes orientados a objetos utilizan, objetos y mensajes, un objeto es un paquete de información y un mensaje es una descripción de como es manipulado. No sólo contiene datos, sino también un conjunto de funciones que son permitidas para accesar esos datos; cada objeto se comunica con otros objetos, y envía mensajes de acuerdo a la relación que exista entre ellos en un momento dado. SMALLTALK Y LOGO son mejor conocidos como lenguajes orientados a objetos, ambos visualmente son orientados, es sencillo crear ,probar y cambiar prototipos.

II.3.8 LENGUAJES INTERACTIVOS.

Es un lenguaje en el cual un programa es creado como resultado de un diálogo interactivo entre el usuario y el software, es decir:

```
XREPORT
NAME FOR EXCEPTION LIST:> UNDERBUDGET.
CRITERION FOR INCLUSION IN LIST:> DIST.SALES
LT.9*BUDGET
*EXCEPTIONS FOUND.
DATA NAMES TO INCLUDE IN REPORT:DIST.SALES BUDGET
PCT.DIF
```

Una gran variedad de este tipo de lenguajes son incluidos dentro de los lenguajes de cuarta generación.

II.3.9 LENGUAJES INTERACTIVOS DE PANTALLA.

Es un lenguaje en el cual el usuario interactúa con el sistema a través de pantallas: llenando campos, apuntando, moviendo y cambiando valores de "default", de esa manera es más rápida y poderosa la comunicación con la computadora.

II.3.10 LENGUAJES DE PROGRAMACION GRAFICA.

Es un lenguaje el cual se construye interactivamente con: técnicas de gráficas, lógica estructurada, estructuras de control, árboles de decisión y reglas de procesamiento (diagramas). Los diagramas pueden ser ligados hacia un diccionario de datos y es usado para generar código.

II.4 DEFINICION.

Los "lenguajes de cuarta generación" son una evolución de los lenguajes de programación convencionales, como: el COBOL, el FORTRAN etc., que en su tiempo respondieron a las necesidades existentes, sin embargo llegaron a ser tediosos al contar con demasiadas instrucciones para realizar tareas específicas; su sintaxis es compleja y son bastante dependientes de la lógica del programador.

Los lenguajes de cuarta generación son manejados por menú o comandos. Entre ellos se han incluido algunos lenguajes formales de programación, lenguajes informales de consulta, generadores de reportes y generadores de programas específicos, también son llamados "non-procedurales", generadores de aplicaciones, preprocesadores, etc. La tendencia de estos lenguajes es acercarse al lenguaje natural o corriente, orientados hacia el usuario final en el sistema, en éllos se especifica que debe hacerse más que como deba hacerse, permitiéndole al programador usuario describir la aplicación más que programarla. Estos programas generan líneas de código para un lenguaje o sistema de programación ya existente, o bien pueden generar directamente un código de máquina.

Estos lenguajes se definen como "lenguajes de alta productividad"; además de utilizar instrucciones secuenciales, como los de la tercera generación, emplean otros mecanismos como son: pantallas interactivas y gráficas en computadora.

Muchos lenguajes de 4a. Generación son dependientes de un diccionario y de una base de datos. El diccionario, en algunos casos se ha desarrollado por la facilidad con la que puede representar los datos, puede contener formatos de pantalla, formatos de reportes, estructuras de diálogos, relaciones entre muchos datos, validación, controles de seguridad, autorizaciones para leer o modificar datos y relaciones lógicas entre valores de datos. Lo más importante para decidirse por algún lenguaje de 4a. generación es la infraestructura necesaria para soportarlo, lo cual incluye la misma base de datos, bibliotecas y diccionarios de datos.

Los lenguajes de 4a. Generación difieren substancialmente en sus operaciones a las de VON NEWMANN. Los lenguajes de 3a. generación fueron muy variados en su sintaxis, sin embargo ofrecen generalmente un conjunto similar de construcciones.

II.5 OBJETIVOS DE LOS LENGUAJES DE 4a. GENERACION.

Los lenguajes de 4a. generación fueron creados en respuesta a los problemas que se encontraron al manejar los lenguajes convencionales. Entre los objetivos de los lenguajes de cuarta generación figuran los siguientes:

- Acelerar el proceso de desarrollo para diversas aplicaciones.
- Poder hacer cambios fáciles y rápidos, reduciendo así los costos de mantenimiento.
- Minimizar problemas de depuración.
- Generación de código sencillo y de alto nivel para los requerimientos actuales.
- Hacer lenguajes amigables para que el usuario final pueda resolver sus propios problemas y poner trabajos sencillos a la computadora.

II.6 PRINCIPIOS BASICOS PARA EL DISEÑO DE 4GLs.

- Principio de mínimo trabajo. Las computadoras deben trabajar con mínimo esfuerzo.
- Principio de destreza. Deberíamos ser capaces de poner a la computadora a trabajar tan fácil como sea posible, sin necesidad de prácticas complejas.
- Principio del mínimo tiempo. Mínimos errores, mínimo mantenimiento y máximos resultados.

II.7 CARACTERISTICAS.

II.7.1 PROPIEDADES DE LOS LENGUAJES DE 4a. GENERACION.

Para que un lenguaje merezca ser llamado de "Cuarta Generación" debe de contemplar las siguientes características:

1.- Debe de ser amigable al usuario en general.

2.- Un programador sin amplios conocimientos puede obtener resultados con él en poco tiempo.

3.- Emplea un sistema manejador de base de datos directamente.

4.- Los programas para diversas aplicaciones son creados en un orden de magnitud de instrucciones menor que usando el lenguaje Cobol.

5.- Donde sea posible se empleará código non-procedural.

6.- Donde sea posible, tomar decisiones inteligentes acerca de lo que el usuario realmente desea.

7.- Está diseñado para operar en línea.

8.- Fomenta y/o impone código estructurado

9.- Es fácil de entender y mantener para cualquier persona el código.

10.- Los usuarios pueden aprender parte del lenguaje en un curso de entrenamiento de 2 días.

11.- Está diseñado para una fácil depuración.

12.- Facilita el empleo de prototipos, que pueden ser creados y modificados rápida y eficientemente.

13.- Pueden obtenerse resultados, con un orden de magnitud, de tiempo mucho menor que con Cobol para la mayoría de las aplicaciones.

14.- Es un lenguaje seguro para la toma de decisiones.

15.- Se puede utilizar en Mainframes, Minicomputadoras y Microcomputadoras.

16.- Puede accesar Mainframes o bases de datos remotas.

17.- Además de todo esto pueden proporcionar:

- Preguntas y actualizaciones simples.
- Preguntas y actualizaciones complejas.
- Habilidad para crear fácilmente una base de datos.
- Operaciones inteligentes con la base.
- Operaciones de validación y actualización con mínimo esfuerzo.
- Generación de pantallas de entradas de datos.
- Generación de pantallas de actualización de datos.
- Un lenguaje que da capacidad total de programación.
- Técnicas de graficación para el diseño de aplicaciones.
- Manejo de "spreadsheet" (hoja electrónica de cálculo).
- Manejo de matrices multidimensionales.
- Generación de reportes.
- Generación y manejo de gráficas.
- Soporte de decisiones para preguntar "que pasa si...".
- Herramientas de análisis matemático.
- Herramientas de análisis financiero.
- Herramientas para soporte de decisiones.
- Manejo de textos.
- Correo electrónico.

CICLO DE DESARROLLO CONVENCIONAL

Identificación Del Problema

Análisis Del Sistema Actual

Diseño Del Nuevo Sistema

Especificaciones De Los Programas

Desarrollo De Los Programas

Pruebas De Integración De Programas

Implementación Del Sistema

Documentación Del Sistema

Mantenimiento Del Sistema

CICLO DE DESARROLLO CON
METODOLOGIAS DE CUARTA GENERACION

Determinar Las Necesidades Generales Del Usuario Final

Diseño Conceptual Y Construcción De Un Prototipo

Uso Y Pruebas Del Prototipo

Modificación Del Prototipo

Conversión A Un Sistema Final De Trabajo

Mantenimiento, Modificación Del Prototipo

COMPONENTES DE UN LENGUAJE DE 4a. GENERACION.

II.7.2 COMPONENTES DE UN LENGUAJE DE 4a. GENERACION.

Un buen lenguaje de 4a. generación de propósito general, posee varios componentes non-procedurales que a su vez pueden ser ligados a una facilidad procedural.

En la figura 2.4, en lo más alto está una ayuda administrativa para dar un nombre al procedimiento, catalogándolo para manifestar de que versión se trata y quién es responsable de ella .

Lo siguiente, es la facilidad para crear una especificación de los datos utilizados. En un ambiente de archivos, el diseñador de la aplicación puede crear sus propios archivos. En un ambiente de base de datos, el diseñador puede emplear datos hechos por la sección del administrador de los datos.

Lo siguiente, es un generador de reportes. Los reportes pueden ser especificados y la especificación almacenarla en un diccionario. Igualmente un apuntador de pantalla puede ser usado para diseñar pantallas, las cuales serán almacenadas en el diccionario.

Un especificador de diálogo puede ser usado para dar la estructura de interacción persona-computadora.

Un medio puede ser empleado para especificar condiciones o decisiones complejas. Esto puede emplear un árbol de decisión, una tabla o un lenguaje para expresar ciertas reglas. Es deseable que la especificación de decisiones complejas o reglas este separado del cuerpo de la aplicación, porque entonces las reglas o condiciones pueden ser cambiadas, sin alterar el código principal de la aplicación.

El conjunto con los datos especificados, el generador de reportes, generador de pantallas, generador de diálogo y un especificador de reglas, es en general una facilidad procedural.

Esto permite a la estructura de un programa, ser especificada mediante iteraciones, condiciones y rutinas anidadas. Las estructuras procedurales pueden ser diseñadas gráficamente sin la necesidad de recordar la estructura de los comandos del programa.

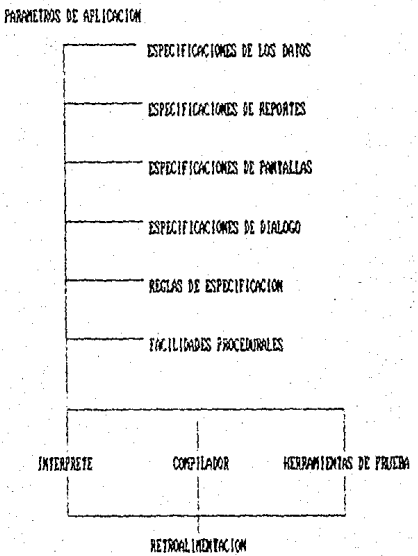


FIGURA 2.4. COMPONENTES DE UN LENGUAJE DE CUARTA GENERACION PARA LA CONSTRUCCION DE APLICACIONES.

II.8 VENTAJAS Y DESVENTAJAS.

VENTAJAS.

- Obtención rápida de resultados. Las cualidades y herramientas que poseen estos lenguajes permiten el desarrollo rápido de sistemas.

- Simplicidad de código. Es fácil de aprender y entender, además de ser poderoso y reducirse el número de líneas de programación.

- Mantenimiento rápido, eficiente y fácil. Las mismas herramientas que traen consigo, proporcionan la capacidad de modificaciones fáciles.

- Documentación automática del sistema.

- Compatibilidad, tanto de software y hardware, dentro de la misma línea.

- Incremento de la productividad.

- Mejor interface con el usuario. En general es amigable, está orientado tanto al usuario final, como al profesional de procesamiento de datos.

- Operación en línea.

- Manejo de menúes y ayudas en línea.

- Acceso directo a una base de datos.

- Diccionario de datos integrado.

- Soportan el empleo de la técnica desarrollo de proyectos de programación empleando prototipos.

DESVENTAJAS.

- Alto consumo de recursos.

- Están orientados principalmente al uso de aplicaciones comerciales y administrativas, por ejemplo : ventas, recibos de caja, facturación, depósitos, retiros, entrada y recepción de mercancía, etc.

- Poca compatibilidad general con los otros lenguajes de 4a. generación, la mayoría de estos mecanismos se aplican a problemas específicos y emplean sistemas de bases de datos o archivos no estándar.

- Las interfases con los sistemas existentes de procesamiento de datos no existen, o son muy complicados para aplicaciones a gran escala en tiempo real.

- Desconfianza de estos productos por la resistencia al cambio.

- Muchos de los lenguajes no pueden crear todos los tipos de aplicaciones. Este es un precio que se tiene que pagar para la gran mejoría en la productividad que traen consigo. En este caso se debe seleccionar el lenguaje que mejor se ajuste a la aplicación específica que se requiera.

III.- DESARROLLO DE UN SISTEMA DE INFORMACION CON LENGUAJES DE CUARTA GENERACION

III.1 GENERALIDADES.

En el desarrollo de sistemas, es de esperarse, que los requerimientos y especificaciones de diseño sean siempre claros y bien definidos. Pero esto no siempre ocurre, en muchas ocasiones no se pueden anticipar todas las características de los sistemas.

Entre diversas razones, por las que no se establecen correctamente las necesidades de información, se pueden mencionar las siguientes:

- Los usuarios sólo saben que tienen que mejorar su sistema de trabajo, o modificar procedimientos existentes, saben que necesitan una mejor información para administrar ciertas actividades, pero no saben con certeza cual es esa información. Esta falta de precisión sobre la verdadera necesidad del usuario, dificulta la formulación de un diseño.

- Por otro lado, los responsables del sistema pueden carecer de experiencia previa en un sistema de este tipo, o bien no cuentan con información de algún sistema similar antes desarrollado.

Un "Prototipo de Sistemas" se desarrolla con la intención de obtener la información necesaria para formular el diseño y desarrollar el sistema. Podemos decir que es un sistema de trabajo que se desarrolla rápidamente, con el fin de probar un conjunto de ideas y esclarecer el entendimiento sobre el nuevo sistema. El prototipo no se limita al trabajo en papel, implica el desarrollo de un software que corra y produzca información.

En otras ramas de la ingeniería, un prototipo se crea antes de construir el producto final. Esto se hace para probar ciertos principios y asegurar que el sistema trabaja al obtener realimentación al diseño que permita ajustarlo antes de gastar mucho dinero. Una planta química se construye en el laboratorio antes de ser finalmente diseñada. La forma del casco de un bote es probado previamente. Un nuevo aeroplano es simulado en muchas formas antes de construirlo.

Uno de los ejemplos más claros en el uso de los prototipos se puede apreciar en la industria automotriz. Las compañías exhiben en las ferias de automóviles sus modelos que pertenecen a programas de investigación y desarrollo, donde muestran los avances tecnológicos que se han de aplicar en futuros vehículos. Estos prototipos no se proyectan con el fin de producirlos de inmediato en serie, su desarrollo contribuye a perfeccionar el diseño y las características de ingeniería de los próximos automóviles.

Los sistemas complejos de procesamiento de datos necesitan del uso de prototipos más que otros sistemas de ingeniería porque hay mucho que aprender de la operación experimental y hay cambios factibles por ser realizados. Los prototipos ayudan a resolver problemas de sistemas que no trabajen de la forma que el usuario realmente necesita, y esto reduce en gran escala las modificaciones que son requeridas eventualmente. En un sentido, los sistemas de procesamiento de datos creados con el ciclo de vida tradicional son un prototipo. Esto no significa que en verdad lo sean o se consideren como tal, pero tienen todas las imperfecciones de ellos. Estas imperfecciones son caras de corregir, por lo que frecuentemente permanecen en el sistema, pesando significativamente en el costo de mantenimiento.

La razón por la que los prototipos de procesamiento de datos no fueron usados comunmente hasta la década de los 80's, fué que el costo de programarlos era tan alto, como el de desarrollar el sistema final de trabajo. El enfoque de prototipos es sólo posible gracias al ciclo de tiempo de desarrollo tan significativamente reducido, y al costo relativamente bajo de implementarlos que ofrecen los lenguajes de cuarta generación. Lo que realmente lo permitió, fueron las facilidades que estos lenguajes nos ofrecen, como son, por mencionar algunas:

- Operación en línea.
- Facilidades para la creación de bases de datos.
- Acceso directo a la base de datos.
- Operaciones inteligentes con la base.

- Operaciones de validación y actualización con mínimo esfuerzo.

- Generación de pantallas de entrada y edición.

- Generación de reportes.

- Generación automática de programas.

- Generación de gráficas.

- Lenguaje tipo "query".

- Editores inteligentes y poderosos.

- Herramientas para generar documentación.

- Procesos eficientes de respaldo y recuperación, etc.

1.1 DEFINICION.

Un prototipo es un producto de nuestro diseno en base a la concepción inicial de las necesidades del usuario, el cual va a contener una implementación de cada función o subsistema, pero a un nivel representativo de la operación del sistema, para que posteriormente, con los ajustes requeridos por el mismo usuario, deje de ser un prototipo y se convierta en un producto final.

1.2 RAZONES PARA EL USO DE PROTOTIPOS.

Existen varias razones para desarrollar prototipos, entre las cuales podemos mencionar las siguientes:

- Ilustrar los formatos de entrada, mensajes, informes y diálogos al usuario. Este es un mecanismo adecuado para explicar opciones de procesamiento y tener un mejor entendimiento de las necesidades de él.

- Explotar aspectos técnicos del producto propuesto. Con frecuencia, una decisión importante del diseno dependerá, por ejemplo, del tiempo de respuesta del controlador de un dispositivo o de la eficiencia de un algoritmo de clasificación; en tales casos, un prototipo puede ser la mejor o única manera de resolver el problema.

- Cuando el ciclo de desarrollo tradicional es inapropiado. Este ciclo de fases se aplica cuando se tiene un conjunto razonablemente completo de especificaciones al inicio del ciclo de vida. Algunas veces no es posible definir el producto sin un desarrollo exploratorio, y en ocasiones no es claro como proceder a la mejora del sistema hasta no evaluarlo en pleno funcionamiento.

1.3 PASOS QUE INTEGRAN EL DESARROLLO DE SISTEMAS CON PROTOTIPOS.

A continuación se muestran los pasos del desarrollo de sistemas en forma tradicional y el desarrollo con prototipos.

El desarrollo de sistemas en forma tradicional consiste en los siguientes pasos:

- Análisis de requerimientos.
- Diseño conceptual.
- Diseño detallado.
- Desarrollo (programación).
- Implementación.
- Post-implementación (mantenimiento).

En éste tipo de desarrollo las fases se encuentran perfectamente definidas, no se avanza de una a la otra hasta no terminar con ella, cada una requiere de un especialista y los requerimientos necesitan estár bien definidos desde el inicio. Si ésto no ocurre se puede caer en un análisis perpetuo, en un ciclo sin fin de análisis-diseño o en un desarrollo no planeado.

En la siguiente figura 3.1 se muestra como se comportan las fases de éste desarrollo:

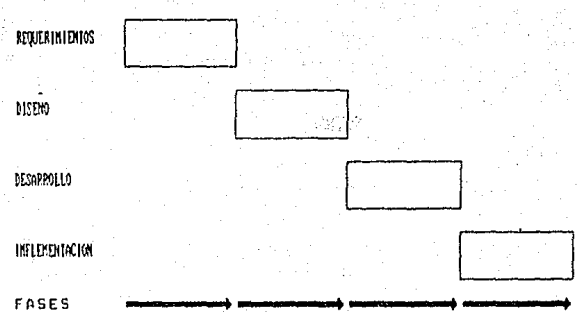


FIGURA 3.1. DESARROLLO POR FASES.

El "Desarrollo con Prototipos" da lugar a un ciclo de vida diferente al tradicional, porque varia sensiblemente en sus caracteristicas y uso. Este ciclo de vida es representado en la figura 3.2 y descrito a continuación:

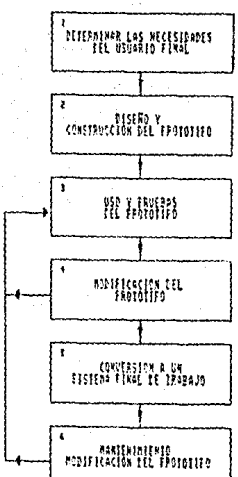


FIGURA 3.2. CICLO DE VIDA DEL DESARROLLO CON PROTOTIPOS.

El primer paso en la figura 3.2 es determinar en forma general lo que el usuario final necesita. Los usuarios hacen un requerimiento los analistas lo estudian y determinan que datos se requieren y cómo se podría construir el producto.

En el segundo paso, un prototipo se realiza en base al diseno conceptualizado del sistema. Es importante que éste sea hecho a la menor brevedad posible para mostrar los principios al usuario. Se pueden mostrar inicialmente sólo las funciones más significativas y no incluir por ejemplo, detalles periféricos, requerimientos de auditoria, o cosas por el estilo.

El tercer paso consiste en una demostración del funcionamiento del prototipo al usuario, él lo debe de evaluar y hacer las observaciones adecuadas.

En el paso número cuatro se realizan los cambios que se hallan requerido. Con la alternativa de regresar al paso tres y así ir afinando el prototipo.

El quinto paso pretende convertir si es necesario el prototipo a un sistema final de trabajo con la posibilidad de ponerlo en producción.

El último paso consiste en dar mantenimiento o modificar el prototipo si es que ha sufrido cambios durante el proceso iterativo, repercutirán en el diseno inicial del sistema, entonces se deben de realizar los ajustes correspondientes para convertirlo en un sistema final de trabajo, el prototipo es conservado para utilizarlo por si se requieren aplicaciones o cambios futuros de mantemnimiento, en dicho caso se realimenta el paso tres, como se muestra en la figura 3.2.

El comportamiento evolutivo de las etapas de desarrollo con prototipos se muestra en la figura 3.3.

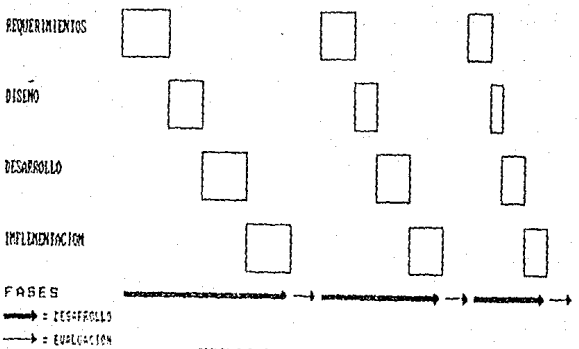


FIGURA 3.3. DESARROLLO EVOLUTIVO.

Las fases en el desarrollo evolutivo con prototipos se van repitiendo de acuerdo como lo vaya requiriendo el desempeño del proyecto, de igual modo, su duración varía según se necesite tendiendo a disminuir hasta que se complete. Cada ciclo consta de una etapa de desarrollo, dónde pueden intervenir algunas o todas las fases, y otra etapa y evaluación de dicho desarrollo, que cómo ya se ha mencionado, realimenta el proyecto hasta alcanzar el estado óptimo de éste.

En los capítulos siguientes se explican a detalle los lineamientos generales de cada fase, así como algunas técnicas propuestas para su desarrollo.

III.2 ANALISIS DE REQUERIMIENTOS.

INTRODUCCION.

El análisis de sistemas es la parte más difícil del desarrollo de un sistema de procesamiento de datos. Los proyectos demandan que el analista tenga profundos conocimientos en el área del negocio y de metodologías en desarrollo de sistemas actuales. Existen dificultades y políticas que surgen especialmente en proyectos extensos donde el nuevo sistema tendrá varios servicios.

Primeramente hay que indicar las especificaciones del modelo, esto significa que se llevará a cabo y cómo será elaborado. Los sistemas que no son aceptados o entendidos por el usuario en etapas preliminares, resultan un conjunto de problemas y malas interpretaciones del análisis es preferible que cuando se detecte un error de inmediato sea corregido y no esperar a etapas subsecuentes donde el sistema es más complicado y muy difícil de adaptar.

La aceptación para todas las partes y la resistencia a la prueba de tiempo es la parte más difícil del desarrollo de sistemas de 4a. Generación; si esto es bien hecho, entonces no importa como se dificulte el diseño y la programación, el sistema sirve a las necesidades del negocio. Si esto no está bien elaborado, entonces no importa que tan excelente sea la implementación y los costos pueden exceder los beneficios.

Las necesidades que requiere el nuevo sistema, deben de estar claras, debemos ser capaces de responder cualquier pregunta relacionada con el desarrollo del sistema llevarlo a cabo en el menor tiempo posible y a un bajo costo, las preguntas a responder son:

1.- ´Qué hace el sistema actual ?

2.- ´Qué necesita hacer el sistema ?

3.- ´Qué es lo que el usuario necesita ?

4.- ´Qué problemas debe de resolver el nuevo sistema ?

5.- ´Qué información debe ser almacenada ?

6.- ´Qué funciones deben ser definidas para los requerimientos de la empresa ?

7.- ´Cómo será el nuevo sistema ?

8.- ´Cómo será afectada la operación del sistema ?

9.- ´Cuáles son los recursos y funcionalidades de la empresa ?

10.- ´Cuál es la organización ?

A continuación se muestra una figura 3.1 que relaciona el costo que representa la detección de los errores cuando se ha avanzado bastante en un proyecto.

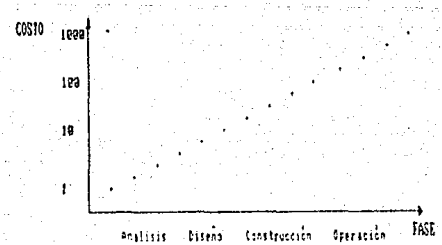


FIGURA 3.1. COSTO DE CORRECION DE ERRORES Y ANALISIS.

2.1 PROBLEMATICA.

El problema de comunicación surge en cualquier situación en donde personas de diferente posición con diferentes puntos de vista y diferentes vocabularios trabajan en forma conjunta. El analista debe equiparar entre qué es generalmente posible en su tecnología (minis, micros, procesamiento distribuido, base de datos, comunicación de datos) y qué es importante hacer para sus negocios.

El analista encuentra difícil el aprendizaje de los negocios para resolver los requerimientos del sistema a través de los ojos del usuario (el usuario no explica claramente sus necesidades).

La gente en la comunidad de usuarios todavía no conoce lo suficiente acerca del procesamiento de datos para saber qué es lo factible y lo irrealizable.

Es muy importante la comunicación entre el departamento de usuarios y el grupo de desarrollo de sistemas, si el documento de especificaciones del nuevo sistema no es muy detallado por el usuario, no será útil para el diseno y para los programadores que tienen que construir el sistema (duplicación del trabajo del analista).

Antes que comencemos a considerar cómo resolver el problema que existe en el área debemos entender las operaciones. Este es un problema típico de análisis donde dicho problema no es entendido y puede traer consecuencias graves, provocando que las correcciones sean muy costosas como anteriormente hemos mencionado.

Tenemos que identificar con claridad que se está haciendo, cómo, por qué y qué se espera de él; una vez entendido el problema u objetivo debemos de confirmar nuevamente nuestro entendimiento, ya que la mayoría de las veces el usuario no se explica del todo, o bien, el analista no interpreta lo que desea el usuario es por eso que surge la necesidad de utilizar un lenguaje común y básico para nuestra metodología.

El problema principal es la falta de un lenguaje común, por citar un ejemplo, dos usuarios de diferentes áreas pueden entender un problema sólo que de diferente manera y así se inician las verdaderas complicaciones de comunicación.

Muchos sistemas fallan porque no se supera el problema de la comunicación, esto no es fácil por la necesidad de satisfacer diversos grupos de gente, que necesitan diferentes cuestiones de análisis, por ejemplo, al jefe de los usuarios le concierne que se lleven a cabo los objetivos, los usuarios por su parte cómo el desarrollo afecta a sus operaciones. El gerente de DP desea conocer cómo el proyecto es procesado, algunos más, que recursos son necesarios, otros (staff), desean conocer que se espera del sistema, con estos problemas surge la necesidad de crear una herramienta de entendimiento general, llamada "Diagrama de Flujo de Datos" (DFD).

2.2 REQUERIMIENTOS DEL USUARIO.

Antes de iniciar cualquier trabajo debemos tener bien definidos los objetivos del proyecto. Para un mejor desarrollo del mismo debe de existir la intervención de ejecutivos que colaboren en el estudio de viabilidad respectivo, el analista debe tener en cuenta los siguientes puntos:

- Requerimientos de usuario.
- Términos de referencia.
- Carta del proyecto.
- Definición del proyecto.

Los requerimientos del usuario no son otra cosa que el estudio del sistema actual y las mejoras que se le hicieron para lograr el máximo beneficio y todo esto surge mediante los DFD, donde el usuario ve e interpreta el movimiento de su información y sus resultados.

2.3 DEFINICION.

El análisis significa:

- Tomar una parte, apartarla o separarla.
- Estudiar un problema.
- Encontrar respuestas acerca de algo.
- Entendimiento de un problema.

El análisis estructurado sirve para resolver problemas de una manera más fácil ya sea tanto problemas pequeños, así como los más complejos y enredados, de hecho una técnica es descomponer el problema en pequeñas partes para encontrar su rápida solución al proceso de reducción de funciones complejas y convertirlas en un conjunto de pequeñas funciones es llamado "Particionamiento" que lo emplearemos en esta metodología y existen varias reglas que son:

1.- El proceso de reducción se puede repetir varias veces hasta alcanzar el nivel de definición y estemos seguros que hemos entendido.

2.- Las piezas que son de pequeño uso tendremos que relacionarlas con el resto del sistema (interfases), debemos de particionar para minimizar el número de interfases, pero también entender la naturaleza de las mismas, ésta es la información que será pasada en las diversas funciones.

3.- Finalmente, tenemos que asegurarnos que los datos necesarios son transmitidos entre los procesos.

Resumiendo se puede decir que el análisis especifica que debe hacerse y cómo debe hacerse.

2.4 TECNICAS ACTUALES.

2.4.1 GRAFICAS.

En general, la gente comprende la información más fácil y rápidamente en dibujos, figuras, etc. en lugar de largas y aburridas pláticas. Es aquí donde entra en acción las representaciones gráficas de la metodología empleada y las técnicas de análisis estructurado, los DFD (Diagrama de Flujo de Datos) manejan procesos y procedimientos, los datos se mueven dentro y fuera de los procesos. Los modelos de relación de entidades contemplan los DFD donde expresan información especifica de los datos, es decir, acerca de los tipos de datos que conforman el registro y las relaciones entre ellos (estructuras de los datos), esto es para evitar que las cosas no se ajusten con lo que se quería.

2.4.2 ARBOL DE DECISION.

El árbol de decisión mostrado en la figura 3.2, las ramas del árbol corresponden a cada una de las posibilidades lógicas; el camino en el cual la cantidad de descuento depende de la combinación de posibilidades que son evidentes. Como herramienta de análisis fuera de la estructura lógica, para iniciar al usuario a confiar que la lógica de la política expresada es correcta, el árbol de decisión es excelente.

FIGURA 3.2. ARBOL DE DECISION.

2.4.3 TABLAS DE DECISION.

Si un árbol de decisión es confuso o no es familiar al usuario para representar cualquier situación donde haya que elegir una opción de un conjunto de soluciones, se pueden emplear dichas tablas, ya que proveen un camino de identificación de todas las combinaciones posibles de condiciones que puedan surgir y pueden ser representadas por más complicadas que éstas sean. La tabla de decisión es una matriz de renglones y columnas que muestra condiciones, acciones y reglas de decisión, y para cada "n" condiciones existen 2 elevado a la "n" combinaciones posibles, sin embargo, debe notarse en que casos no son posibles dichas combinaciones que por lo tanto, no generan ninguna acción por ejemplo:

CONDICIONES	REGLAS DE DECISION			
C1 >= 10 MOTORES	S	N	S	N
C2 < 5 MOTORES	N	S	S	N
ACCIONES				
AL 20 % DE DESCUENTO	X		X	
AL 5 % DE DESCUENTO		X	X	

FIGURA 3.3. TABLA DE DECISION.

Comparando los árboles y las tablas de decisión para indicar cuál es mejor se puede aplicar el siguiente criterio:

> "Usa el árbol de decisión cuando el número de acciones es pequeno y no todas las combinaciones de condiciones son posibles; usa la tabla de decisión cuando el número de acciones es mayor y muchas combinaciones de condiciones pueden ocurrir".

2.4.4 PSEUDOCODIGO.

El pseudocódigo, también conocido como el inglés estructurado o el inglés reducido, etc. sin llegar al grado del análisis sintáctico de cualquier lenguaje de programación, se puede decir que es casi el código del programa. Se forma de varios componentes o instrucciones entre ellos son:

1) Instrucciones secuenciales

```
ejemplo:  "read file"              "lee el archivo"
          "compute pay"            "calcula el pago"
          "update files"           "actualiza archivos"
          "write messages"         "escribe mensajes"
```

2) Instrucciones de decisión

```
ejemplo:   IF condición-1          Si condición-1
             THEN acción-1           ENTONCES acción-1
           ELSE no-condición-1      OTRO no-condición-1
             SO  acción-2            ENTONCES acción-2
```

3) Case (varias posibilidades)

```
ejemplo:   IF caso-1                    Si caso-1
             acción-1                       acción-1
           ELSE IF caso-2                OTRO SI caso-2
                  acción-2                      acción-2
                ELSE IF caso-3                OTRO SI caso-3
                       acción-3                      acción-3
                     ELSE .....                    OTRO .....
```

4) Instrucciones de repetición (loop's)

```
ejemplo:  REPEAT                    REPETIR
            X = A + B                 X = A + B
            Y = H * X                 Y = H * X
            A = A + 10                A = A + 10
          UNTIL <condición>           HASTA QUE <condición>
```

```
DO WHILE <hay registros>          HACER MIENTRAS <haya registros>
    calcula el impuesto               calcula el impuesto
        suma el iva                       suma el iva
ENDDO                             FIN HACER
```

2.5 TECNICAS Y HERRAMIENTAS PROPUESTAS.

Dentro de las técnicas y herramientas propuestas se consideran:

1) El Diagrama de Flujo de Datos (DFD).

2) El Diccionario de Datos.

3) El Modelo de Información (Entidades, Atributos y Relaciones).

Puesto que:

- Nos brindarán una fácil representación de problemas y cómo solucionarlos.

- Permiten una buena comunicación entre el usuario y el Area de Sistemas.

- Nos permite separar el problema en diversos y pequeños módulos, se describe el flujo de los datos y el almacenamiento de los mismos.

- Se reemplazan las largas narrativas eliminando la documentación excesiva.

Es importante tener a continuación un modelo, el cual debe de ser aprobado por el usuario en donde se indicará que será lo que se llevará a cabo.

Los modelos de relación de entidades son representaciones gráficas de varios conjuntos de datos que deben ser manejados en el sistema y las relaciones que existen entre esos conjuntos. Dichas relaciones reflejan los caminos en que los usuarios ven sus datos, es decir, cuentan con un conjunto de mini-especificaciones del modelo.

La idea es que en las especificaciones (pseudocódigo) se documente cada proceso del sistema.

Los DFD son un camino para el modelado de procesos dentro de un sistema y muestran como están interrelacionados dichos procesos entre sí, sin embargo, no es un panorama total del sistema, es decir, necesitamos saber dónde y cómo se almacenan los datos, surgiendo así la necesidad de emplear un diccionario de datos.

A continuación detallaremos estas técnicas y herramientas.

2.6 DIAGRAMA DE FLUJO DE DATOS.

Una de las herramientas más importantes de la construcción de modelos durante las fases de análisis es el DFD, que nos sirve para auxiliarnos de una mejor manera para el entendimiento del sistema, además de permitirnos separar el problema en pequeños y variados módulos para su total comprensión.

Ejemplo:

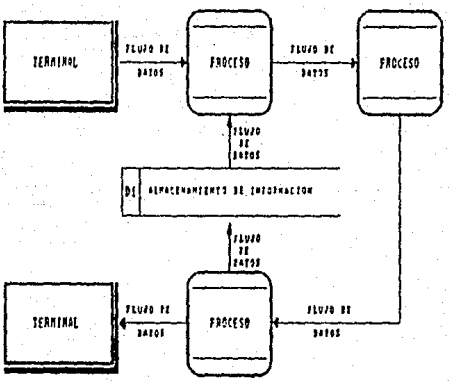


FIGURA 2.4. DIAGRAMA DE FLUJO DE DATOS.

2.6.1 COMPONENTES DEL DFD.

El DFD se auxilia de un simple conjunto de símbolos que permiten el rápido y fácil reflejo de diferentes funciones. Utilizan nombres reconocibles para los usuarios, a continuación describiremos cada una de esas partes que conforman el DFD.

- Flujo de Datos.
- Procesos.
- Almacenamiento de Información.
- Entidad Externa.

2.6.2 FLUJO DE DATOS.

El flujo de datos muestra el movimiento de la información (datos) entre los procesos; dentro y fuera de archivos señalando como es almacenada dicha información y desde luego cómo inicia y cómo termina.

La dirección del flujo se indica por la punta de la flecha. Por definición el flujo describe los datos (secuencia) y como deben ser llamados esos datos, los DFD no son diagramas puramente de control y el flujo de la información no indica en general el control mismo, es decir, todos los flujos deben relacionar alguna(s) pieza(s) de información, pero no necesariamente la descripción específica de cada proceso.

Se expresará el contenido de un flujo de datos definiendo los nombres de las estructuras de datos que pasan a lo largo de ésta se podrá notar:

- La fuente del flujo de datos.
- El destino.
- Los volúmenes de cada estructura de datos o transacciones.
- La presente implementación física del flujo de datos.

El flujo de datos está simbolizado por una flecha preferentemente horizontal y/o vertical, la punta de la flecha muestra la dirección de flujo ver fig 3.5. En la subsecuente escena de análisis cuando el contenido del diccionario de datos ha sido definido, la descripción puede cambiar a letras mayúsculas para mostrar que están entrando al diccionario de datos.

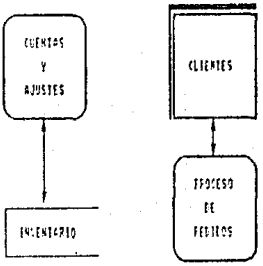


FIGURA 3.5. FLUJO EN AMBAS DIRECCIONES.

2.6.3 EL SIMBOLO "PROCESO".

Es usado para indicar diversas funciones por ejemplo, siempre que los datos son transformados desde una entrada hacia una salida y se obtienen resultados, se dice que existió un proceso éste símbolo también representa un procedimiento donde es transformada la información, es decir, cuando ciertos volúmenes de información son actualizados se dice que la información se procesó.

Ejemplo:

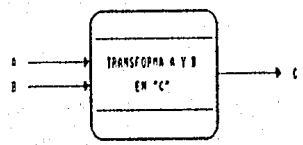


FIGURA 3.6. EL SIMBOLO "PROCESO".

Se necesita describir las funciones de cada proceso para referenciar, y dar a cada proceso un único identificador, posiblemente ligando éste detrás del sistema físico. Los procesos pueden ser simbolizados por un circulo opcionalmente dividido en tres áreas; ver fig 3.7.

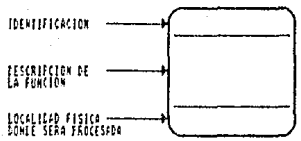


FIGURA 3.7. SIMBOLO PROCESO.

La identificación puede ser un número, inicialmente localizado aproximadamente a la izquierda, sólo él puede identificar un proceso. No hay un punto en el cual se asigne significado al número del proceso, algunos procesos pueden ser divididos en dos o más que puedan ser incorporados a uno durante el trabajo de análisis. Una vez asignada la identificación de los procesos no podrán cambiar excepto para dividirse o unirse, mientras esto sirve como referencia para el flujo de datos y la descomposición de los procesos a niveles bajos.

La referencia física es de gran ayuda cuando se estudia un sistema existente para notar que departamento, o cual programa lleva una función al exterior. Cuando el análisis es completado y el diseno físico de un nuevo sistema está en marcha, es conveniente hacer notar que la función podrá físicamente ser realizada.

Posteriormente se mostrarán pasos para poder dibujar un DFD, que se refinará hasta llegar a un diagrama óptimo.

2.6.4 DATOS ALMACENADOS.

Los datos almacenados es donde se almacenan o se hace referencia a los datos a través de un proceso. Los datos pueden simbolizarce por un par de líneas paralelas horizontales encerradas hasta el final, debe de ser de un ancho en donde entre el nombre. Cada uno puede estar identificado por un número arbitrario localizado a mano izquierda para indicar el número del archivo. El nombre debe de escogerse lo más descriptivo para el usuario como muestra la fig. 3.8.

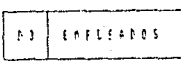


FIGURA 3.8. DATOS ALMACENADOS.

Un dato almacenado es una estructura de datos en reposo, describimos el contenido de cada dato almacenado en términos de estructuras de datos. El contenido lógico de cada dato almacenado está reservado en el diccionario de datos bajo un nombre; lo mismo sucede con el flujo de datos.

El analista especificará los elementos del dato que son reservados para cada almacén, una vez que el dato es extraído por algún flujo puede ser nuevamente leído de la especificación entrante y saliente del flujo de datos.

Si el flujo de datos muestra el movimiento de los mismos, entonces el símbolo de almacenamiento representa dónde o en qué lugar la información es físicamente almacenada para su posterior explotación.

Ejemplo:

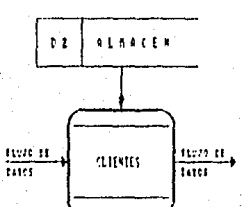


FIGURA 3.9. EXTRACCION DE DATOS ALMACENADOS.

2.6.5 EL SIMBOLO TERMINAL O ENTIDAD EXTERNA.

Es empleado para denotar la fuente y/o destino de la información usada o creada dentro del Area de un proyecto. Un simbolo terminal puede ser una persona, un departamento, una organización o bien otro sistema; los simbolos terminales definen los limites o alcances de una investigación.

Ejemplo:

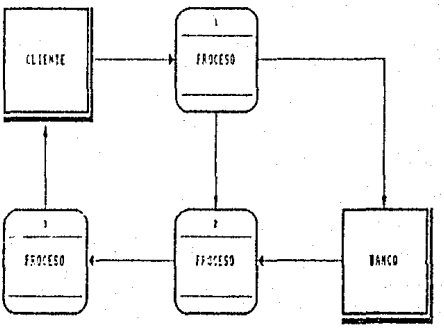


FIGURA 2.10. EL SIMBOLO TERMINAL.

Caracteristicas del simbolo terminal:

a) Define el contexto general del sistema.

b) Origen y recepción de la información.

c) Por lo regular aparece al principio y final del DFD.

Una entidad externa representa una fuente o destino de transacciones, puede ser simbolizada por un cuadro.

La entrada puede estár identificada por una letra minúscula en la esquina superior izquierda para referencia.

Para evitar intersección de las líneas de flujo de datos, la misma entrada puede ser dibujada más de una vez en el mismo diagrama; dos o más cajas por entrada pueden ser identificadas por una línea quebrada en la esquina baja a mano derecha como se muestra en la fig. 3.11 donde otra entrada está siendo duplicada, las instancias de esta tienen dos líneas quebaradas como muestra la fig. 3.12.

FIGURA 3.11. DUPLICACION DE SIMBOLOS DE ENTIDADES EXTERNAS

En los procesos de análisis se aprende más acerca de los objetivos del usuario, se tomarán algunas entidades externas y se conducirán a nuestro sistema del diagrama de flujo de datos, o alternativamente se tomarán de la función del sistema y removerá a éste de condiciones para disenar todas como una entidad externa con flujo de datos.

FIGURA 3.12. DUPLICACION MULTIPLE DE ENTIDADES EXTERNAS

2.6.6 RESUMEN.

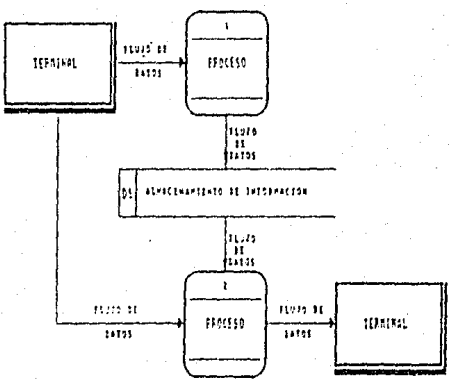


FIGURA 2.13. DIAGRAMA DE FLUJO DE DATOS.

Concluyendo tenemos que:

- Los DFD emplean 4 simbolos, que son: simbolos terminales ó entidades externas, flujo de datos, almacenamiento y procesos o funciones.

- Nombres únicos deben de ser usados en todos los simbolos.

2.6.7 GUIA PARA DIBUJAR EL DIAGRAMA DE FLUJO DE DATOS.

1.- Identificar las entidades externas involucradas.

2.- Identificar las entradas y salidas del sistema.

3.- Especificar un flujo de datos que defina la información otorgada al sistema y un flujo de datos, que nos diga qué es solicitado por el sistema.

4.- Dibujar en una hoja la primer fuente externa de entrada, el flujo de datos que se solicita, los procesos que son necesarios y el almacenamiento de datos que será requerido.

5.- Cuando se tiene el primer bosquejo checar con la lista de entradas y salidas para asegurar que se incluye todo, excepto que se trate de errores y excepciones.

6.- Dichos errores y excepciones serán procesos que pueden ser incluidos dentro del DFD, si es que el sistema así lo requiere, pero deben quedar bien delimitados.

7.- Producir un segundo bosquejo, con procesos únicos y un número mínimo de flujo de datos cruzados. Para minimizar el cruzamiento se considerará lo siguiente:

- Duplicar entidades externas si es necesario.

- Repetir el almacenamiento de los datos si es preciso.

8.- Producir una ramificación a bajo nivel de cada proceso. Trabajar sin errores, excepciones e incorporar cambios a el diagrama si es necesario.

La guía anterior puede ser aplicada a métodos novedosos para elaborar un DFD, como el método de pantallas.

2.7 DICCIONARIO DE DATOS.

Un diccionario de datos es una lista de todos los campos empleados en Archivos, Sistemas etc. y que conforman el DFD para ayudarnos a la determinación de los requerimientos del sistema.

De la misma forma que los procesos son particionados de un proceso complejo y detallado, hasta encontrar el nivel más bajo del DFD, así también el flujo de datos y el almacenamiento de los mismos pueden ser similarmente tratados (particionados), en el nivel más bajo obviamente se encuentran los datos o campos.

Las especificaciones típicas de un sistema, las largas y enrredosas narrativas y las documentaciones excesivas son remplazadas por las especificaciónes estructuradas, éstas comprenden:

- Un conjunto de niveles del DFD que describe los procesos.
- Un modelo de relación de entidades describiendo, como se almacenan los datos.
- Un diccionario de datos, describiendo el flujo de los datos y el almacenamiento de los mismos.
- Las mini-especificaciones, describen el detalle final de alguna parte del proceso.

Cada parte de las especificaciones estructuradas se relacionan y se complementan unas a otras, una combinación de las anteriores minimizará el riesgo de falla.

Se necesita empezar a un nivel lógico, identificando cada uno de los elementos del dato, que están presentes en el flujo de datos, dándoles nombres intencionados, definiendo cada uno y organizándolos para poder estimar una definición fácil.

El diccionario de datos contendrá una definición de cada flujo de datos, el contenido de cada dato almacenado y los elementos del dato por los cuales está compuesto. Dichas definiciones se pueden ordenar alfabéticamente para fácil referencia y así, se tendrá el diccionario de datos para el sistema.

El beneficio más importante para el analista es que puede escribir el flujo de datos, conociendo todos los detalles relacionados a los mismos.

El diccionario de datos está en un lugar conveniente para guardar el glosario de items. Una vez definidos, se puede usar la forma de un elemento del dato, con la extracción sólo del nombre, una descripción pequena y la sección de aliases necesarios.

Los diccionarios de datos surgen como una necesidad para definir los datos, es decir, se necesita una notación para manejarlos y controlarlos, los DFD proporcionan un panorama general del proyecto dentro de un sistema, sin embargo tenemos que tener la seguridad de entender la información ya sea tanto el flujo como el almacenamiento de los mismos y qué contienen ésos datos mas no el formato como se confunde con facilidad.

Algunos de los elementos de interés que deben declararse en el DD/DS son:

1.- Elementos del Dato. Piezas de datos que no son significantes para descomponerse.

2.- Estructura de Datos. Están compuestas de elementos de datos, o de otras estructuras de datos o una mezcla de ambos.

3.- Flujo de Datos y Datos Almacenados. El flujo de datos son las rutas o líneas por las cuales los datos estructurados viajan, los datos almacenados son colocados también donde las estructuras de los datos son almacenadas hasta que se necesiten. El flujo de datos son estructuras de datos en movimiento y los datos almacenados son estructuras de datos en reposo.

La mínima información necesaria para establecer un elemento del dato, es su nombre y descripción de la instancia. Además para el nombre y la instancia es necesario grabar lo siguiente:

1.- "ALIASES". Es un nombre o un símbolo, el cual está en lugar de algo y que no es propiamente su nombre puede surgir por diferentes usuarios de varios departamentos llamando la misma cosa por diferente nombre, la gente del almacén la llama "número de requisición" y la gente de compras "número de orden", por citar unos ejemplos. Los "aliases" a veces pueden surgir, porque la misma cosa es definida en programas escritos en diferentes lenguajes o por diferentes programadores.

2.- RELACIONANDO A LOS ELEMENTOS DEL DATO. Capacidad para poner a los elementos del dato que tienen nombres relacionados, en la entrada del diccionario de datos, se tendrá cuidado al iniciar los nombres con las mismas letras, en una lista alfabética de elementos del dato.

3.- RANGO DE VALORES Y SIGNIFICADO DE VALORES. Para empezar a examinar los valores que un elemento del dato puede tomar se observará que hay dos tipos de elementos del dato.

- Los que para todos los propósitos prácticos pueden tomar cualquier valor en un rango.

- Los cuales sólo pueden tomar hasta un cierto valor.

El primer tipo de elementos del dato pueden llamarse continuo, ya que para este valor es prácticamente continuo sobre su rango. El segundo se puede llamar discreto por que éste sólo toma valores discretos.

Para elementos de datos se necesita anotar el rango que ellos pueden tomar, un valor típico, cualquier información acerca del manejo de valores extremos.

Para los elementos de datos discretos, necesitamos anotar los valores y el significado que es dado por cada valor.

El analista debe de juzgar hasta que punto se debe llegar para considerar los elementos del dato como discretos y tratar un elemento continuo que pueda ser usado como llave a un valor del dato almacenado en la tabla de valores y significados se tabularán los valores en el diccionario de datos, esto es para definir un dato almacenado el cual contiene un significado.

4.- LONGITUD. Especificar la longitud del elemento del dato. La longitud puede ser tomada en binario o en un paquete decimal decodificado. El analista debe de especificar la longitud en la primer pasada a tráves de la creación de un diccionario de datos pero puede ser libre para sumarles en adaptaciones posteriores.

5.- DECODIFICACION. El disenador y el programador necesitan espacio para grabar en el diccionario de datos la forma en la cual el elemento del dato podrá fisicamente ser decodificado en el sistema.

Estas decisiones físicas no son para el analista y no forman parte de la especificación funcional lógica. Sin embargo, en algunas circunstancias no hay decisiones que hacer mientras el sistema propuesto tenga interfase con otro sistema. En éste caso no se tendrá control sobre el formato físico y el analista no anotará la decodificación de este flujo en el diccionario de datos.

Muchos de los valores de un diccionario de datos vienen del hecho de que éste es un almacén central de datos para todos los analistas, disenadores y programadores que trabajan en un solo proyecto o alguna aplicación de un área especifica. Mientras que el diccionario sea un almacén central puede ser controlado por una persona o un grupo. A un nivel de proyecto tal persona debe de llamar al administrador de datos o al administrador del elemento del dato.

El administrador de datos, guarda el control sobre las entradas y cambios a el diccionario de datos.

En general, los paquetes automatizados proveen lo necesario para una edición amplia de entradas y construyen una base de datos con índices y apuntadores para permitir el uso del diccionario para trazar las relaciones alrededor del mismo, usualmente con acceso en línea.

Si nos concierne el mantenimiento de un sistema o si tenemos que desarrollar un sistema con interfase a un número de sistemas existentes, necesitamos ser capaces de pensar en todos los programas relevantes, en el software y generar entradas al diccionario de datos en cualquier formato estándar. Estos nos darán un almacenamiento central de información acerca del sistema que tenemos que mantener y lo podemos explorar usando otras facilidades de búsqueda o el diccionario

La estructura de datos es armada fuera de los elementos del dato u otras estructuras, para describirla es necesario especificar los nombres de las estructuras y elementos que la constituirán proviendo estos componentes que son definidos en otra parte del diccionario de datos, algunos componentes de la estructura son mandatorios y otros son alternativos, algunos son opcionales y varios son repetidos una o más veces.

Se necesita un camino conveniente para especificar los rasgos de una estructura del dato, este camino es el anotar la notación usada en un manual de lenguaje de programación para mostrar la estructura de comandos del lenguaje como sigue:

1.- Estructura Opcional. Una estructura de datos o elementos del dato dentro de un paréntesis cuadrado, significa que éste es un componente opcional de la estructura.

2.- Estructuras Alternativas. Dos o más estructuras de datos o nombres de elementos del dato dentro del paréntesis, significa que uno solo de esos componentes puede estar presente en una instancia de la estructura.

3.- Interacción De Estructuras. Los manuales de lenguajes de programación muestran interacciones tomando lugar tres períodos después del item, esta interacción se marcará con un asterisco y significa que no puede haber uno o más términos; se especifica el rango de posibilidades.

Una forma simple de registrar estructuras de datos, es definiendo cada elemento del dato reelevante que se necesita usando la notación descrita. Cuando la estructura del dato relacione algo físico necesitaremos referenciar a una pequeña descripción.

2.8 MODELO DE INFORMACION.

El objetivo del modelo de información es representar gráficamente toda la información requerida por el usuario y la interrelación de dicha información. El modelo de información es la base para el diseño del almacenamiento de datos lógicos y físicos.

- Una herramienta que facilita el desarrollo continuo cuando el usuario tiene nuevos RSI's.

- Criterios para seleccionar un sistema administrador de la base de datos.

El modelo de informacion contiene una cantidad mínima de datos, es construido utilizando las entidades mínimo se refiere que una llave en particular o atributo ocurra en una sola localidad del modelo. Aunque el modelo permita tener datos no redundantes, se podrán representar múltiples relaciones a los datos. Todas las personas que participen en el estudio de información deberán también verificar que el modelo de información es correcto.

2.8.1 Requerimientos De Entrada/Salida De Información (RSI).

Esta es la parte más importante dentro del modelo de información ya que cualquier sistema que se desee desarrollar por pequeño que éste sea se deben conocer los elementos con los que se va a trabajar (entradas) y los que realmente se van a producir (salidas), surgiendo así los requerimientos de entrada/salida de información (RSI); se documentarán los RSI's usando aquéllos de sistemas relacionados.

- Documentar los RSI's utilizando un formato que nos dará la información necesaria del requerimiento del usuario.

- Verificar con el usuario los RSI's.

- Analizar los diálogos para los procesos.

Un formato propuesto para llevar a cabo la descripción de los RSI's es mostrado en la figura 3.14.

1.- Fecha del sistema.

2.- Número de la página.

3.- Nombre del sistema o subsistema.

4.- Identificación. Identificador del sistema seguido por el número del RSI.

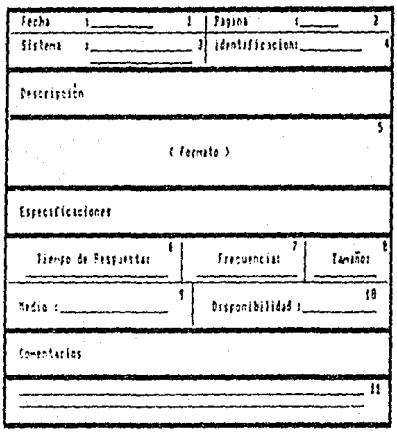


FIGURA 3.14. FORMATO DE ENTRADA/SALIDA.

5.- Descripción. En este espacio se describe en palabras y/o en formatos la terminología que será usada en el reporte si es posible usar una combinación de formatos y narrativas de descripción.

6.- Tiempo de respuesta. Intervalo de tiempo en que el RSI es solicitado y el tiempo en el que éste está disponible al usuario. Este periódo puede variar de segundos a semanas.

7.- Frecuencia. El número de veces que un RSI puede ser desplegado durante un periódo de tiempo especificado.

8.- Tamaño. El volumen aproximado del RSI expresado en número de caracteres.

9.- Medio. Dispositivo en el cual se encontrará almacenado.

10.- Disponibilidad. Periódo durante el cual el usuario puede solicitar el RSI.

11.- Comentarios. Es muy común indicar el orden de distribución para la información que será desplegada, se puede anotar el número total de ocurrencias de un RSI que irá a un proceso en específico.

La verificación de los RSI's consiste en confirmar que el sistema recibe todos los RSI's necesarios, se debe garantizar que cada RSI producido en el sistema contiene suficiente información para que el usuario sea cápaz de realizar todas las funciones subsecuentes cuando sea necesario.

El análisis de los diálogos se hará para los RSI's con un formato de diálogo de pantalla. Para cada función de diálogo se determinará el orden en el cual el usuario trabajará con estas pantallas. Si el usuario no necesita más que un desplegado, el análisis de los diálogos en este punto es terminado. Si el usuario requiere ir a más de un desplegado se debe saber la secuencia en la cual el usuario necesita esta información.

Una vez definidos los RSI's del sistema se analizará la forma en como se relacionan unos con otros, integrando los grupos básicos de información del sistema (Entidades).

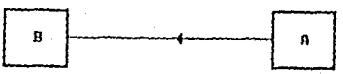
2.8.2 ENTIDADES.

El propósito de definir las entidades es el establecer qué grupos básicos de información constituyen cada uno de los RSI's del sistema, así como determinar sus llaves, atributos y describir todos los elementos de los datos. Los conceptos básicos a usar son :

Entidad : Es una descripción de un concepto u objeto del mundo real.

Relación : Archivo plano o tabla bidimensional; es decir un archivo cuyos registros (llamados tuplos) no pueden tener grupos repetidos. Cada tuplo debe tener un campo identificador único; la relación está compuesta de uno o más campos para cada tuplo las relaciones existentes son:

1 a 1 Representada gráficamente como:

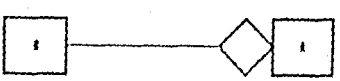


Para cada "A" existe exactamente una "B" y para cada "B" existe cero o una "A".

M a 1 Fuerte. Representada gráficamente como:

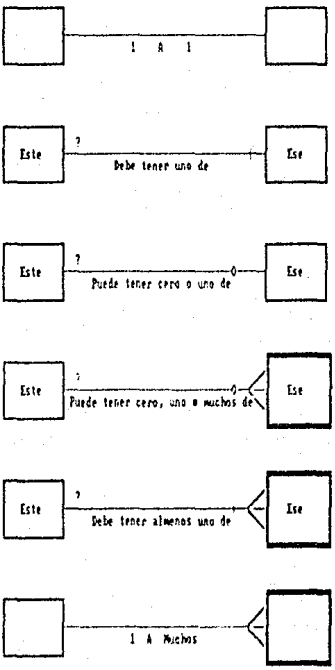
Para cada "A" existe exactamente una "B" y para cada "B" existe al menos una o más "A".

M a 1 Representada gráficamente como:

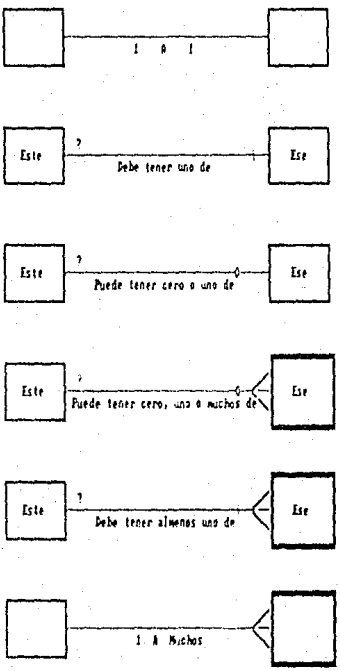


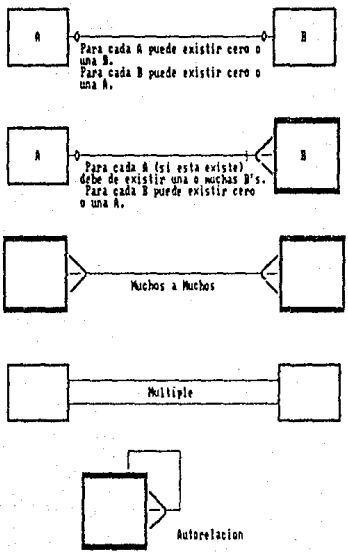
Para cada "A" existe exactamente una "B" y para cada "B" existe cero, una o más "A".

O bien:


1 A 1
Este
?
Debe tener uno de
Ese
Este
?
Puede tener cero o uno de
Ese
Este
?
Puede tener cero, uno o muchos de
Ese
Este
?
Debe tener almenos uno de
Ese
1 A Muchos

O bien:


1 A 1
Este
?
Debe tener uno de
Ese
Este
?
Puede tener cero o uno de
Ese
Este
?
Puede tener cero, uno o muchos de
Ese
Este
?
Debe tener almenos uno de
Ese
1 A Muchos


A
B
Para cada A puede existir cero o una B.
Para cada B puede existir cero o una A.
A
B
Para cada A (si esta existe) debe de existir una o muchas B's.
Para cada B puede existir cero o una A.
Muchos a Muchos
Multiple
Autorelacion

Tuplo : Grupos de valores de campos relacionados específicamente por una relación, cada uno de ellos contienen atributos.

Atributo : Unidad más pequeña de datos llamada también campo de un registro.

Llave : Dato elemental (valor o nombre de un dato o un campo) que se usa para localizar en forma efectiva un registro.

Llave Primaria : Es un atributo o una combinación de atributos los cuales definen en forma única el registro.

Llave Candidata : Más de un conjunto de campos que constituyen una llave, es decir, identifica unívocamenete a cada tuplo y es irredundante.

Dependencia Funcional : El atributo B es completamente dependiente del atributo A si este es capaz de identificar en forma única al atributo mencionado (atributo B).

Estructuras De Datos : Una vez definidos los campos de entrada al sistema (números y/o alfanuméricos) se tienen que analisar entre ellos, es decir, como se relacionan unos con otros, formando así los componentes básicos del sistema. Para enriquecer el estudio se mencionará el proceso de normalización que junto con sus cuatro formas normales, los datos serán manejados de una mejor manera.

Normalización : Es el proceso de poner los datos de la manera más sencilla para que sean fáciles de manejar y mantener. Consiste en cuatro formas de Normalización.

1a Forma Normal : Es una relación en la cual los dominios contienen valores atómicos, es decir si sus dominios son simples, esto es, si es una tabla bidimensional o plana, por otro lado se dice que toda relación normalizada está en primera forma normal

2a Forma Normal : Es una relación que está en 1a. forma normal y cada atributo ó dominio no llave es totalmente dependiente de la llave primaria.

3a Forma Normal : Es una relación que está en 2a. forma normal y cada atributo tiene dependencia no transitiva en toda llave candidata. Suponer que "A" es una llave y "B" y "C" son atributos. Si "C" es funcionalmente dependiente de "A". Se asume que "A" y "B" no son intercambiables como llaves, entonces "C" es transitivamente dependiente de "A".

4a Forma Normal : Una relación está en 4a. forma normal, si todos sus atributos tienen únicamenete una ocurrencia para cada llave; o bien si ésta consiste de un sólo atributo que tiene muchas ocurrencias para cada llave. La 4a forma normal garantiza que el desarrollo del sistema será preciso, verificable y detallado para adecuarse completamente a las bases del diseno.

Para establecer las entidades se debe considerar lo siguiente:

1.- Trabajar con un RSI a la vez.

2.- Listar todos los elementos de los datos que integran el RSI.

3.- Para cada uno de los elementos de los datos, indicar aquellos que lo identifiquen y su llave.

4.- Para cada atributo indicar si ocurre una o varias veces en la entidad.

5.- Verificar las entidades de acuerdo a las cuatro formas de normalización.

Acontinuación se propone un formato para documentar las entidades, mostrado en la figura 3.15.

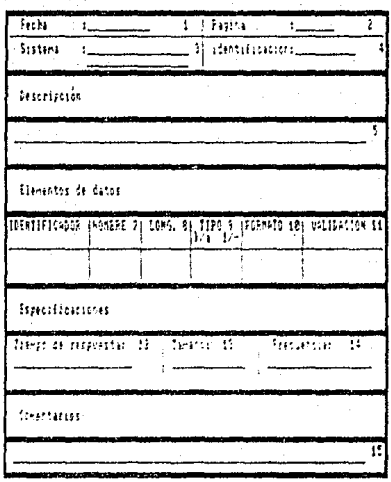


FIGURA 3.17. FORMATO DESCRIPCION DE ENTIDADES.

1.- Fecha del sistema.

2.- Número de la página.

3.- Nombre del sistema o subsistema.

4.- Identificador del sistema. Número del RSI, identificador y número de la entidad.

5.- Descripción. Una pequeña descripción de la entidad.

6.- Identificador y nombre. De cada uno de los elementos de datos del grupo incluyendo sus llaves.

7.- Longitud. De cada uno de los elementos del dato expresado en número de carácteres.

8.- Tipo. K para las llaves y A para los atributos; el número de veces que ocurre la llave en la relación hacia estos; para todos los atributos cuantas veces ocurre en la relación hacia la llave.

9.- Formato. Formato para los datos.

10.- Validaciones. Todas las validaciones que se requieran para los datos.

11.- Tiempo de respuesta. El lapso de tiempo en que la entidad es solicitada y el tiempo en el que está disponible al usuario.El tiempo de respuesta puede ser el mismo que el del RSI.

12.- Tamano. El volumen aproximado del dato expresado en número de caracteres.

13.- Frecuencia. El número de veces que la entidad, será incluida en el RSI.

14.- Comentarios. Descripción de condiciones bajo las cuales la entidad deberá ser desplegada.

Ahora estamos posibilitados para construir el modelo de información una vez que se definieron los RSI's y las entidades.

2.8.3 Costrucción Del Modelo De Información.

Primero hay que ordenar las entidades para facilitar la estructuración del modelo, para esto es necesario clasificar y listar las entidades (alfabeticamente). Identificar la(s) llave(s), así como los atributos que ocurren en las entidades. Los símbolos usados en el modelo de información son:

- Entidad, la cual es didujada como una caja cerrada y el nombre de la entidad tendrá que estar a dentro de esta, figura 3.16.

FIGURA 3.16. ENTIDAD

- Llaves, la llave para cada entidad se localiza en la parte superior indicando el o los atributos que la conformen y para distingirla será subrrayada.

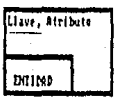


FIGURA 3.17. LLAVE.

- Atributos, se localizan en la parte superior despúes de la llave. Si los atributos que contiene la entidad son muy grandes, se podrán listar en un apéndice con referencia a las llaves en el modelo.

- Relaciones entre las entidades (mostradas con anterioridad).

Por último tendremos que verificar que el modelo de información es correcto, deberán checarse las llaves, atributos y sus relaciones para confirmar que todos los RSI's son correctos.

El modelo no debe confundirse con el almacenamiento de datos lógicos o archivos físicos. Las cajas del modelo representan una extensión de registros físicos.

La fase para determinar las necesidades del usuario en un ambiente de Lenguajes de 4a Generación consta de las siguientes tareas:

2.9 PROGRAMA DE TRABAJO.

1.- Información para entrevistas.

2.- Descripción del problema.

3.- Costo del sistema.

4.- Modelo conceptual.

TAREA NUMERO UNO.

Información Para Entrevistas.

En ésta tarea se pretende obtener una comunicación entre el usuario y el analista de sistemas, a través de entrevistas para la obtención de la información del sistema.

Para esto es necesario tener una constante comunicación con:

- Usuarios.
- Jefes de usuarios.
- Gerentes de DP.
- Otros miembros del proyecto.
- Otros staffs de DP involucrados en etapas posteriores de desarrollo.

TAREA NUMERO DOS.

Descripción Del Sistema.

En ésta tarea deben de quedar muy claras las necesidades del usuario y tener muy en cuenta una amplia definición del proyecto.

Los requerimientos del usuario son el documento llave para el proyecto, idealmente debería contener:

- Declaraciones del problema. Un resumen del problema o problemas que el sistema debe solucionar.
- Definir objetivos. Una o más instrucciones no técnicas que el nuevo sistema debe contemplar.
- Alcance del sistema. Los límites iniciales de la investigación.
- Restricciones o limitaciones en el desarrollo hablando en términos monetarios, tiempo y factor gente involucrados.
- Criterio de operación. Declaraciones que permiten al analista verificar cualquier solución para encontrar los requerimientos del usuario.

TAREA NUMERO TRES.

Costo Del Sistema.

En base a las tareas anteriores tendremos la posibilidad de analizar aún más el sistema. Por otro lado el analista debe tener en cuenta.

- Evaluar Costo/Beneficio.
- Negociar ventajas y desventajas del sistema.
- Mantener claramente el control de que se va hacer en todo momento.

TAREA NUMERO CUATRO.

Modelo Conceptual.

En esta tarea utilizaremos las herramientas propuestas descritas con anterioridad, tales como:

- Diagrama de flujo de datos.
- Modelo de entidades, atributos y relaciones.
- Diccionario de datos.

Por último la información requerida para el sistema debe ser verificada con el jefe de usuarios y gerente de DP (Data Proccessing), para estudiar la forma de cómo se elaborará el proyecto.

PRODUCTOS DE LA FASE.

1.- DEFINICION DEL SISTEMA.

- Definición del problema.
- Justificación del sistema.
- Metas del sistema y del proyecto.
- Funciones que se proporcionarán.
- Estrategias de solución.

2.- CONSIDERACIONES PARA EL USUARIO.

- Panorama y exposición del producto.
- Terminología y características básicas.
- Resumen de informes y despliegues.

3.- ESPECIFICACIONES DE REQUISITOS.

- Requisitos funcionales.
- Modelo de información.
- Interfases externas y flujo de datos.
- Requisistos de operación.
- Manejo de excepciones.
- Subconjuntos iniciales y prioridades para las características del sistema.
- Modificaciones y mejoras.

III.3 METODOLOGIA DEL DISEÑO PROPUESTA.

INTRODUCCION.

El diseño es un proceso de aplicar técnicas y herramientas con el propósito de definir un dispositivo, proceso o sistema con suficiente detalle para permitir su realización.

El objetivo más importante del diseño es entregar y producir las funciones requeridas por el usuario. Hay tres objetivos principales que el diseñador tiene que tener presentes mientras se está desarrollando y evaluando un diseño.

- FUNCIONAMIENTO. Qué tan rápido el diseño será capáz de hacer el trabajo del usuario, dado un recurso particular de hardware.
- CONTROL. La extensión a la cuál el diseño es seguro contra errores humanos, mal funcionamiento de la máquina, o daño deliberado.
- MANTENIBILIDAD. La facilidad con la cuál el diseño permite a el sistema ser cambiado, por ejemplo, al encontrar que las necesidades del usuario tienen diferentes tipos de transacciones procesadas.

Aunque no es siempre cierto, generalmente pasa que éstos tres factores trabajan uno contra otro. Un sistema con controles muy seguros tenderá a tener un funcionamiento degradado, un sistema diseñado para un funcionamiento muy alto, puede ser no muy fácil de cambiar.

El diseño de software es un proceso a través del cuál los requerimientos se traducen a una representación de software.

La fase del diseno inicia cuando la definición de requerimientos ha sido aceptada; una mejor manera de entrar a ésta fase es la realimentación recibida a través del contenido del reporte de definición de requerimientos.

Se debe contemplar lo siguiente:

- Especificaciones funcionales que describan la solución del problema.

- Procesamiento de datos que describan las características técnicas del sistema de aplicación.

La descripción funcional incluye una descripción completa del sistema a implementarse desde la perspectiva del usuario, ésta descripción orientada a documentar soluciones para los problemas identificados.

Los diagramas de las partes del sistema contienen los siguientes tipos de información:

- Descripción de las funciones operacionales y administrativas qué se implementarán en el nuevo sistema.

- Discusión de las entradas del sistema, incluyendo los tipos de transacciones y sus fuentes que serán ejecutadas.

- Discusión de las salidas del sistema, que incluirá la descripción narrativa o formatos actuales de los reportes, pantallas y el propósito de las salidas.

- Descripción de los datos que serán mantenidos por las estructuras de archivos y/o por la base de datos.

- Aplicar los métodos y técnicas de procesamiento, la descripción del procesamiento y los datos que debe contener son los siguientes:

    a) Un estimado de volúmenes de entrada, salida y almacenamiento.

    b) Descripción de las interfases del sistema.

    c) Descripción del software que se usará.

    d) Identificación de objetivos que serán conseguidos.

Los aspectos que deberán resolverse y discutirse dentro de el panorama de procesamiento de los datos son:

- Entrada de datos en lote vs entrada de datos en línea.
- Actualización en lote vs actualización en línea de archivos maestros.
- Reportes en lote vs reportes en línea.
- Mainframe vs minicomputador.
- Proceso propio vs servicio de procesamiento.

El objetivo en ésta etapa de diseño es, especificar el sistema de procesamiento de los datos, el nivel de detalle al cuál serán preparadas las especificaciones del sistema, la destreza de los miembros asignados al proyecto, para implementarlo y las herramientas a ser usadas. El reporte de diseño es utilizado por:

a) Especialistas y técnicos, para preparar las especificaciones de programación.

b) Especialistas en la línea empresarial para, preparar la documentación de procesamiento del usuario.

c) Los directivos del proyecto para guiar las etapas de desarrollo e implementación.

Para llevar a cabo el diseño debemos considerar dos factores muy importantes estos son:

Factores Técnicos: Estos a su vez se subdividen en diseño estructurado y en diseño por estructuras de datos. Para éstos diseños se presentarán técnicas para su elaboración, teniendo en cuenta la independecia de módulos su modularidad, qué tan cohesivo o nivel de acoplamiento se desea que contemple el sistema, además, se presentan las metodologías existentes tales cómo la metodología de Jackson y Warnier. Para llevar a cabo el diseño, dependerá totalmente de la gente que lo tenga a su cargo y el diseñador escogerá las técnicas que él considere más convenientes, esto será en base a su experiencia o bien sobre la cuál tenga más conocimientos.

Factores Humanos: Se subdividen a nivel usuario y a nivel sistema. Estos nos ayudan a desarrollar sistemas con Lenguajes de 4a. Generación en una forma más fácil para el usuario y así motivar a un conjunto más grande de gente para que los use, se expondrá más adelante una serie de consideraciones que nos permitirán evaluar la productividad, la obtención de resultados, el minimizar errores de sintaxis y de lógica.

Se debe incluir en el reporte del diseno:

- Un bosquejo funcional y de proceso de datos para cada aplicación del sistema, incluyendo una descripción narrativa y diagramas de flujo.

- Una discución de las entradas del sistema incluyendo:

    * Descripción de las transacciones.
    * Descripción de los elementos de los datos.

- Descripción de las salidas del sistema (reportes, pantallas, archivos, transacciones a otros sistemas), incluyendo:

    * Formatos de salida.
    * Distribución/Acceso.

- Descripción de los procesos lógicos.

- Descripción de las estructuras de archivos y/o bases de datos.

- Descripción de todas las interfases.

3.1 FACTORES TECNICOS.

Describiremos más apliamente los tres objetivos principales del diseno que se mencionarón con anterioridad.

3.1.1 CONSIDERACIONES DE FUNCIONAMIENTO.

El funcionamiento se expresa usualmente en términos de:

- Transacciones o cálculos por hora, (throughput).
- Tiempo de corrida, (run-time) para un proceso, dónde el mismo monto de trabajo es procesado en cada corrida.
- Tiempo de respuesta, el tiempo que transcurre entre presionar la tecla de "enter" en una terminal y el inicio de la respuesta de la computadora apareciendo en la terminal.

3.1.2 CONSIDERACIONES DE CONTROL.

Dependiendo de la naturaleza del sistema, el diseñador necesitará construir controles de varios tipos. Algunos aspectos de control son:

a) El uso de dígitos verificadores en números predeterminados.

b) El uso de totales o números de control.

c) La creación de auditorías y bitácoras.

d) La limitación del acceso a los archivos.

3.1.3 CONSIDERACIONES DE MANTENIBILIDAD.

El sistema procesa datos del mundo real, cada vez que el mundo real cambia, el sistema puede necesitar cambiar.

Así cómo un usuario puede tener nuevas ideas acerca de los requerimientos de información, la tecnología de procesamiento de datos cambia también tanto en hardware; haciendolo más poderoso y más barato, cómo en el software; introduciendo nuevos sistemas operativos, lenguajes y comunicación de datos.

Por lo tanto la cambiabilidad de un sistema es algo muy importante. Por cambiabilidad se entiende; una medida de tiempo que toma hacer cualquier cambio en el sistema ya sea corregir un error o hacer una actualización. Es importante considerar los estándares así cómo la actualización de rutinas y bases de datos comunes.

Una vez que ha quedado claro lo que es el diseno y lo que debemos obtener de él, entramos al detalle de los factores técnicos, en el cuál proponemos dos formas para llevar a cabo el diseno y son; el diseno estructurado y el diseno por estructuras de datos y por último los factores humanos.

3.2 DISENO ESTRUCTURADO.

La herramienta principal del diseno estructurado es la carta de estructura la cuál muestra la partición del sistema en módulos y la relación jerárquica entre éstos. Además muestra los flujos de datos y control entre los módulos.

Por lo tanto podemos definir el diseno estructurado cómo un conjunto de normas para producir una jerarquía de módulos lógicos que representan un sistema altamente cambiable.

3.2.1 ELEMENTOS DE UNA CARTA ESTRUCTURADA.

Una carta estructurada cuenta con los siguientes elementos:

1.- Un rectángulo con un nombre inscrito para indicar un módulo, el nombre indica la función del mismo.

2.- Líneas qué indican la liga entre módulos (llamadas a módulos).

3.- Flechas qué indican el flujo de datos y de control respectivamente (comunicación entre módulos). Es convencional mostrar cada estructura de datos o elemento de datos con una flecha cómo ésta : oáää>. Dónde un módulo pasa una bandera de control hacia otro módulo, diciéndole al módulo receptor qué pasó, o qué hacer, el elemento de control se ilustra así : oääá>

4.- Un módulo es representado en la figura 3.1.

FIGURA 3.1. REPRESENTACION DE UN MODULO.

5.- El nombre del módulo debe resumir su función y las funciones de sus subordinados inmediatos.

3.2.2 ATRIBUTOS BASICOS.

Un módulo tiene cuatro atributos básicos:

1.- ENTRADA : Los datos qué le pasa quién lo invoca.

SALIDA : Los datos qué regresa a quién lo invoca.

2.- FUNCION : Lo qué hace a sus datos de entrada para producir sus datos de salida.

3.- MECANICA : Cómo realiza su función, es decir, su lógica.

4.- DATOS INTERNOS : Su propio espacio de trabajo, es decir sus variables locales.

Son ejemplos de módulos: PROCEDURE, SUBROUTINE, PROGRAM, SECTION, PARAGRAPH, etc.

Cómo regla general una carta estructurada muestra a su izquierda los módulos de entrada, al centro los módulos qué procesan la información y al lado derecho los módulos de salida.

3.2.3 CARACTERISTICAS DE LA CARTA ESTRUCTURADA.

Una Carta de Estructura muestra:

1.- La partición del programa, es decir, los módulos de qué consta.

2.- La estructura jerárquica, es decir, la relación entre módulos.

3.- Los nombres de módulos y por consiguiente su función.

4.- El grado de acoplamiento entre módulos.

5.- Flujo de datos entre módulos.

6.- Las decisiones e iteraciones qué involucran la llamada a un módulo.

Una Carta de Estructura no muestra:

1.- El número de veces qué se llama un módulo.

2.- La secuencia en qué se llama un módulo.

3.- Cómo realiza su función.

4.- Datos internos del módulo.

La carta de estructura se deriva del diagrama de flujo de datos consideremos el DFD mostrado en la figura 3.2.

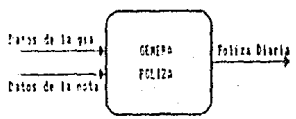


FIGURA 3.2. DIAGRAMA DE FLUJO DE DATOS.

Su correspondiente carta de estructura es mostrada en la figura 3.3.

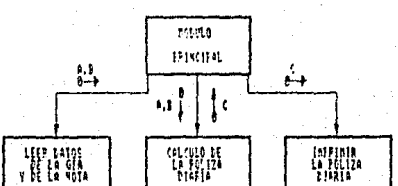


FIGURA 3.3. CARTA ESTRUCTURADA.

A la entrada datos de la gía (A) y a la entrada de la nota (B) en el DFD, le corresponde el módulo "LEER DATOS" en la Carta De Estructura (CDE). Se observa qué éste módulo tiene cómo salida A y B.

Al proceso "GENERA POLIZA" en el DFD, le corresponde el módulo "CALCULA LA POLIZA" en la CDE. Note qué éste módulo tiene cómo entrada A y B, y cómo salida C.

A la salida en el DFD poliza diaria (C) le corresponde el módulo "IMPRIMIR POLIZA". Observe qué éste módulo tiene cómo entrada a C.

3.2.4 M O D U L A R I D A D.

El concepto de modularidad se refiere a qué el software está dividido en elementos separadamente nombrados y direccionados, llamados módulos, qué están integrados para satisfacer los requerimientos del problema.

Se ha manifestado qué "La Modularidad" es el atributo del software qué permite a un programa ser manejable o administrable intelectualmente.

Un argumento de la modularidad dice qué es más fácil resolver un problema complejo cuando lo descomponemos en pequeñas partes manejables.

Un módulo puede ser reentrante, ésto es, un módulo es diseñado para qué en ninguna forma se modifique a sí mismo o las direcciones locales qué lo referencian. Así el módulo puede ser usado por más de una tarea concurrentemente.

Dentro de una estructura de software un módulo puede ser categorizado como:

1.- Un Módulo Secuencial : Es referenciado y ejecutado sin interrupción aparente por el software de aplicación.

2.- Un Módulo Incremental : Puede ser interrumpido, antes de qué se complete, por el software de aplicación y reiniciado posteriormente en el punto de interrupción.

3.- Un Módulo Paralelo : Se ejecuta simultáneamente con otro módulo en ambientes concurrentes de multiproceso.

Los módulos secuenciales son comúnmente encontrados y se caracterizan por compilar macros y subprogramas. Los módulos incrementales, llamados algunas veces corutinas, mantienen un apuntador de entrada qué permite al módulo reiniciar en el punto de interrupción. Tales módulos son extremadamente útiles en sistemas manejadores de interrupciones. Los módulos paralelos, algunas veces llamados corutinas, son encontrados cuando computación de alta velocidad, demanda dos o más cpu's trabajando en paralelo.

3.2.5 INDEPENDENCIA DE MODULOS.

La independencia de módulos se logra desarrollando módulos con funciones "Single-Mined" y una aversión a la interacción excesiva con otros módulos. En otras palabras, queremos diseñar software para qué cada módulo direccione una subfunción específica de requerimientos y tenga una interface simple cuando es vista desde otras partes de la estructura de software.

¿Por qué es importante la independencia de módulos? El software con modularidad efectiva, esto es, módulos independientes, es más fácil de desarrollar porqué las funciones pueden ser divididas y las interfases pueden ser simplificadas.

Los módulos independientes son más fáciles para darles mantenimiento y probar porqué los efectos secundarios causados por el diseño y la modificación de código están limitados, la propagación de errores es reducida.

En resumen, la independencia de módulos es una llave para el buen diseño y el diseño es la llave para el software de calidad. La independencia es medida usando dos criterios cualitativos: Cohesión y Acoplamiento.

3.2.6 C O H E S I O N.

Es una medida de la relativa funcionalidad de un módulo o del grado de asociación de los elementos dentro de un módulo. Se mide en términos de la fuerza de unión de los elementos dentro de un módulo.

Un módulo cohesivo ejecuta una sola tarea dentro de un procedimiento de software, requiriendo poca interacción con procedimientos qué están siendo ejecutados en otras partes de un programa. Un módulo cohesivo debe (idealmente) hacer sólo una cosa.

Se han identificado seis tipos de cohesión de módulos, describiremos los tipos generalmente reconocidos, del peor al mejor.

1.- COHESION COINCIDENTAL (peor)
Los elementos de un módulo están ahí por accidente. Ocurre cuando los elementos dentro de un módulo no tienen relación aparente entre cada uno de ellos. Un ejemplo de ello es la segmentación arbitraria de un programa en varios módulos.

2.- COHESION LOGICA
En éste tipo de módulos varias funciones similares, pero ligeramente diferentes, son combinadas haciendo un módulo más compacto, a diferencia si se hubieran programado separadamente. Los módulos de éste tipo son a veces difícil de cambiar, porque las rutas lógicas a través de ellos son muy complejas. Deben ser reemplazados por módulos de propósito especial, uno por función.

3.- COHESION TEMPORAL
Este tipo de módulos contienen una variedad de funciones cuyo único elemento común es qué son ejecutados al mismo tiempo sin requerir de ningún parámetro o lógica alguna para determinar qué elemento debe ejecutarse, ésto lo vemos, por ejemplo, en la inicialización de un sistema.

4.- COHESION PROCEDITIVA (moderada)
Este tipo de cohesión es encontrada dónde los módulos han sido derivados de una carta de flujo y cada procedimiento de la carta de flujo se ha convertido en un módulo.

5.- COHESION COMUNICACIONAL (de moderada a buena)
Todas las funciones de un módulo cohesivo comunicacionalmente operan en el mismo flujo de datos. Podría entenderse también cómo cuando la salida de un módulo es la entrada para otro.

6.- COHESION FUNCIONAL (la mejor)
Representa un tipo fuerte y deseable de amarre de los elementos de un módulo debido a qué todos los elementos están relacionados con el desempeño de una sola función. Un módulo funcionalmente cohesivo puede usualmente ser descrito por frases sencillas con un verbo activo y un sólo objeto. Por ejemplo imprimir una matriz, calcular la raíz cuadrada.

Es inecesario determinar el nivel preciso de cohesión, es más importante una alta cohesión y reconocer la baja cohesión para qué el diseño de software pueda ser modificado y así lograr una independencia de módulos mayor.

3.2.7 A C O P L A M I E N T O.

Es una medida de interdependencia relativa entre módulos en una estructura de software. Esto significa qué para qué tengamos cambiabilidad, se debe tener el menor acoplamiento posible entre módulos. A continuación se muestran los cinco tipos de acoplamiento qué existen.

1.- ACOPLAMIENTO POR CONTENIDO.
Ocurre cuando un módulo modifica los valores locales o las instrucciones de algún otro módulo. Es la forma de acoplamiento más severa. Por ejemplo, se da en programas en ensamblador.

2.- ACOPLAMIENTO POR ZONAS COMPARTIDAS.
Los módulos son atados en forma conjunta por medio de zonas globales para las estructuras de datos. También es una forma severa de acoplamiento.

3.- ACOPLAMIENTO POR CONTROL.
Se refiere al paso de banderas de control, ya sea cómo parámetros o en forma global, entre los módulos de tal forma qué un módulo controla la secuencia del proceso de otro.

4.- ACOPLAMIENTO POR ESTRUCTURAS DE DATOS.
Es similar al acoplamiento de zonas compartidas excepto qué los elementos globales son compartidos en forma selectiva entre las diversas rutinas qué requieren de los datos.

5. ACOPLAMIENTO POR DATOS.
Es claro qué la forma más deseable de acoplamiento es, dónde un módulo pasa datos a otros módulos cómo parte de una invocación o regresando el control. Es el mejor acoplamiento porque es el menos acoplado.

En general, un diseno en el cuál pocas piezas de datos son pasados entre módulos es más cambiable qué uno en él cuál se pasan muchas piezas de datos. Es más claro y más fácil seguir el acoplamiento cuando los elementos de datos son pasados cómo parámetros en un módulo interfase más qué cuando los datos son parte de uno global qué todos los datos pueden accesar.

A partir de los conceptos de cohesión y acoplamiento podemos concluir que un nivel muy alto de acoplamiento, cómo lo es por el contenido, no es recomendable, el que conviene lograr es el acoplamiento por datos. De igual forma se debe evitar la poca cohesión, cómo la coincidental y se debe procurar la cohesión funcional.

3.3 DISENO ORIENTADO A LA ESTRUCTURA DE DATOS. (METODOLOGIAS)

DISENO Y ESTRUCTURA DE DATOS.

La estructura de los datos influye sobre el diseno en los aspectos estructurales y en procedimientos del software. Los datos repetitivos son siempre procesados con software que tiene fácil control para la repetición; los datos alternativos (información que puede o no estar presente) aceleran el software con elementos de proceso condicional, una organización jerárquica de datos frecuentemente tiene una notable semejanza al software que usa. Esto es, la estructura de la información es un excelente pronosticador de la estructura del software.

CONTRIBUCIONES.

La metodología de Jackson una de las más apliamente usadas en los métodos del diseno de software, toma el punto de vista del paralelismo de las estructuras de entrada y salida de los datos asegurando un diseno de calidad, Jackson enfatiza sobre el desarrollo de técnicas pragmáticas para transformar los datos a estructuras de programas.

La construción lógica de programas (LCP), una metodología desarrollada por Jean Dominique Warnier, proporciona un método más riguroso para el diseno del software. Dibujando los conceptos fundamentales en la ciencia de la computación, Warnier desarrolló un conjunto de técnicas que sostienen un mapeo desde la estructura de los datos de entrada/salida (I/O) hacia una representación detallada de los procedimientos del software.

Una técnica llamada Construcción Lógica de programas está representada por una síntesis del flujo de datos y aproximaciones al diseno orientado a estructuras de datos. Los desarrolladores del método indican que "el diseno lógico puede ser descrito explícitamente si el software está visto cómo un sistema de conjuntos de datos y sus transformaciones". Aunque LCP no es puramente una orientación a la estructura de datos, ésta puede ser vista con estas técnicas de diseno.

3.3.1 LA METODOLOGIA DE JACKSON.

Esta metodología utiliza el diagrama estructurado similar al método del flujo de datos y mapeo de E/S de estructuras de los datos, para producir un programa estructurado. La metodología de Jackson es descrita de la siguiente manera: los problemas se pueden descomponer dentro de estructuras jerárquicas y sus partes pueden ser representadas por tres formas estructurales. Las tres formas estructurales mencionadas son:

SECUENCIA.- Contiene un proceso que puede ser ejecutado uno después de otro, es decir, A suspende a B y C.

SELECCION.- Contiene varios procesos pero sólo uno puede ejecutarse de acuerdo a una condición.

ITERACION.- Contiene un proceso el cuál debe ser ejecutado repetitivamente hasta alcanzar un valor determinado.

Una representación simple de la notación de estructura de los datos de Jackson es mostrada en la figura 3.4. Siguiendo el diagrama jerárquico, una colección de datos A está compuesta por múltiples ocurrencias (denotadas con *) de subestructuras de datos B. La estructura B incluye múltiples ocurrencias de C y otra subestructura D que contiene datos E o F (datos alternativos que son denotados con o). La representación diagramática de Jackson de la información jerárquica puede ser aplicada a entradas, salidas o estructuras de bases de datos con igual facilidad.

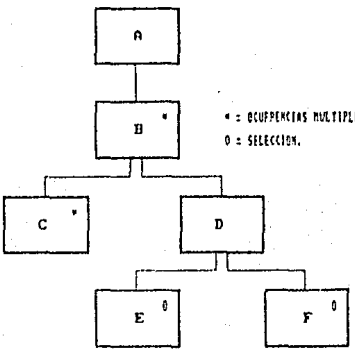


FIGURA 3.4. NOTACION DE ESTRUCTURAS DE DATOS.

Cómo un ejemplo más concreto de esta notación, consideraremos el software a ser desarrollado para el sistema del pago de tarjeta de crédito. Un archivo de pagos contiene los números de clientes (CNO), fecha de pago (FECHA), e importe de pago (AMT), esto es reconciliado con un archivo maestro de clientes que contiene CNO y el balance de lo no pagado. El archivo de pagos está preordenado en grupos de números de cliente (CNO-SROUP) de tal manera que todos los pagos de un individuo son contenidos dentro de un solo registro. La estructura de Jackson para los archivos descritos se muestra en la figura 3.5.

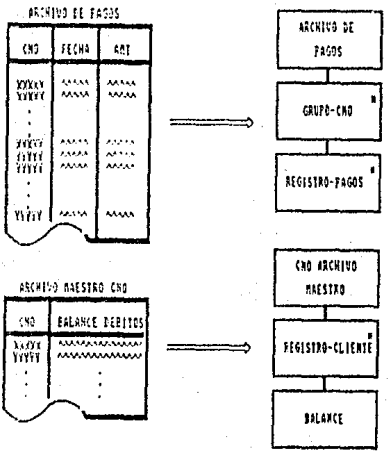


FIGURA 3.5. ESTRUCTURA DE JACKSON PARA LOS ARCHIVOS.

Un reporte de salida para el sistema de pagos de tarjeta de cédito y el resultado del diagrama de la estructura de datos es mostrado en la figura 3.6 de acuerdo a la jerarquización.

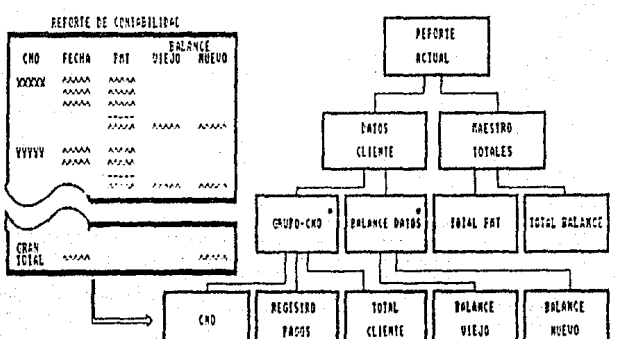


FIGURA 3.6. REPORTE DE SALIDA Y RESULTADO DEL DIAGRAMA DE ESTRUCTURA DE DATOS.

3.3.2 CONSTRUCCION LOGICA DE PROGRAMAS.

La construcción lógica de programas (LCP), da inicio con una representación de la estructura de los datos de entrada y salida con la ayuda de los diagramas Warnier. El siguiente paso de LCP es representar procedimientos de software usando un diagrama de Warnier, el método lleva a una derivación de procedimientos y culmina con métodos sistemáticos para la generación de pseudocódigo, verificación, y optimización. LCP es representado por una serie de reglas que determinan la estructura de la información y la organización resultante del software derivado.

Warnier desarrolló su método bajo la idea de "los programas pueden ser construidos lógicamente" y verificados rigurosamente usando herramientas derivadas de los estudios de informática.

La notación de la estructura de los datos usada en LCP es el diagrama Warnier. Warnier desarrolla una notación para la representación de información jerárquica usando tres construcciones para secuencia, selección y repetición y demuestra que la estructura de software puede ser obtenida directamente de la estructura de datos. Un archivo de datos que tiene tres tipos de registros (1, 2 y 3) que son encontrados cuatro veces, cero, una y n veces, respectivamente es mostrado en la figura 3.7. Los datos del registro uno contienen ítems A, B y C. Los datos del registro dos contienen un ítem F y un ítem G que puede no aparecer (ocurre cero a una vez). Los datos del registro tres siempre contienen el item E y pueden contener cero o n items de D que está contenido de m ocurrencias del elemento I.

El mapeo de las estructuras de entrada/salida de datos producen un programa estructurado. Esta técnica se aproxima para la solución de problemas que requieren un análisis detallado de las estructuras de entrada y salida de datos. El resultado de éste análisis es para desarrollar un diagrama de resultados de la salida y una estructura repetitiva para la entrada. Estos diagramas se forman dentro de estructuras jerárquicas usados para desarrollar programas de estructuras jerárquicos.

La Estructura Lógica de Salida (LOS) es una representación jerárquica de los elementos de datos que componen la salida. El primer paso es aislar todos los elementos de los datos que ya no pueden ser subdivididos, esto se logra revisando la declaración del problema, se usa un reporte prototipo, anotando la frecuencia de ocurrencia para cada elemento del dato que no puede ser subdividido y por último se desarrolla la representación de la Estructura Lógica de Salida usando el diagrama Warnier.

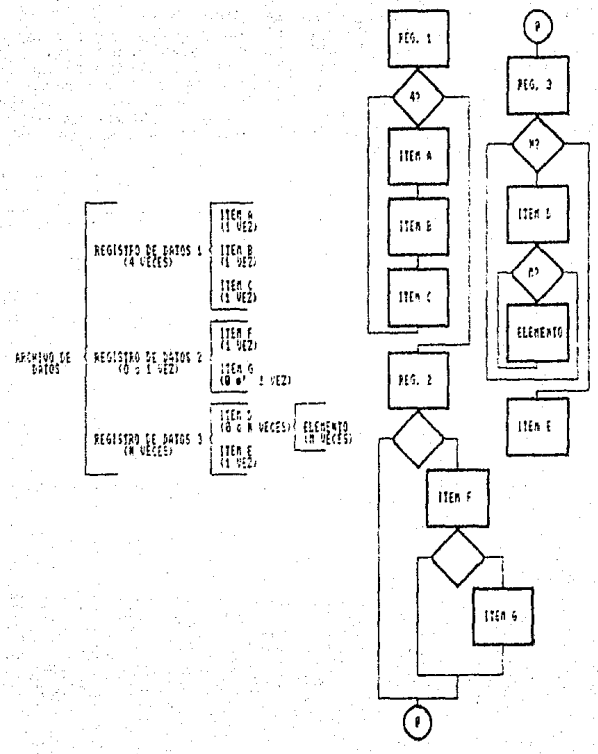


FIGURA 3.7. DIAGRAMA Y CARTA WARNIER.

De la misma forma como se manejo la estructura lógica de salida, la Estructura Lógica de Entrada tendrá el mismo tratamiento sólo que se basa en las características de los archivos o en los requerimientos de la entrada, y se obtendrá su respectivo diagrama de Warnier.

La Estructura Lógica de Procesos (LPS) es una representación de software y procesa sus correspondientes estructuras lógicas de entradas y salidas. El primer paso para obtener la representación de LPS usando el diagrama Warnier es, remover del diagrama los elementos de los datos que no pueden ser subdivididos, adicionar en todas las repeticiones los delimitadores BEGIN y END, paso siguiente definir y especificar el inicio y el fin de las instrucciones así como los procesos de cómputo y los no numéricos. Finalmente se especifican todas las instrucciones de entradas, salidas y sus procesos.

3.3.3 COMPARACION DE LAS METODOLOGIAS DEL DISENO.

1.- Programación Modular (MP).

2.- Diseno Top-Down, basado en la descomposición funcional (TDD).

3.- Diseno Compuesto (CD) desarrollado por Glen Ford J.M.

4.- Diseno estructurado (SD) desarrollado por Larry L. Constantino.

5.- Técnicas de Diseno de Análisis Estructurado (SADT) desarrollado por Douglas Ross.

6.- Construcción lógica de Programas (LCP) y su metodología hermana, la Construcción Lógica de Sistemas (LCS).

7.- Diseno Estructurado de Jackson, metodología desarrollada por él mismo.

Para elegir la mejor metodología en el diseno del software, debemos tomar en cuenta dos factores, los factores estructurales y los periféricos, los primeros se refieren a la base, complejidad, modularidad, escala y la integración estructural todo esto en el desarrollo de sistemas. Los segundos se refieren a los estados o etapas de diseno y su documentación, el control de proyectos, pruebas, portabilidad y compatibilidad externa.

3.3.4 EVALUACION DE LA MEJOR METODOLOGIA.

Dependiendo de la modularidad (estructural), o bien de las estructuras jerárquicas (divisiones, subdivisiones, etc.), en el diseno de sistemas, una u otra metodología va a seleccionarse dependiendo de la tarea o sistema a desarrollar; se habla también del nivel de acoplamiento de los datos, la compatibilidad de código, y el pseudocódigo mismo para una fácil interpretación del sistema.

El análista del sistema tendrá la opción de elegir cualquiera de las metodologías propuestas anteriormente para realizar su diseno y que junto con los prototipos de los lenguajes de 4a. genereación y la elección de la metodología la hará una propuesta más poderosa.

3.4 FACTORES HUMANOS.

INTRODUCCION.

En muchos centros de cómputo en dónde se desarrolla con Lenguajes de 4a. Generación, no se preocupan en la investigación de recursos humanos y sin embargo, no deja de ser un aspecto determinante en lo referente al mantenimiento, operación y desarrollo de sistemas.

El objetivo de los factores humanos es, hacer el desarrollo de sistemas con Lenguajes de 4a. Generación más fáciles para el usuario y motivar a un conjunto más grande de gente para que los use. Se pretende crear sistemas que pueden utilizarse por gente que carece de tiempo y paciencia para aprender aplicaciones automatizadas en computadoras, en otras palabras, manejar sistemas amigables y fáciles de aprender.

En procesos muy complejos el factor humano es determinante y debe tomarse en consideración, porque con él podemos hacer los procesos más rápidos debido a que se enumeran una serie de consideraciones que nos permiten elevar la productividad, obtener buenos resultados, minimizar los errores de sintaxis, de significado y de lógica.

El diseno en cuanto al aspecto humano se puede clasificar cómo sigue:

- El Factor Humano a Nivel de Usuario. Nos proporciona las facilidades que el sistema brindará al usuario para que sea amigable durante su operación.

- El Factor Humano a Nivel de Sistemas. Al igual que en el punto anterior se consideran los factores humanos más importantes para la construcción de un sistema con el fin de que se faciliten las tareas para la realización del mismo.

A continuación se mencionan los factores humanos más importantes que se recomiendan en la fase del diseno, tomando en cuenta tanto el aspecto humano a nivel usuario cómo a nivel sistema.

3.4.1 NOMBRES DE ARCHIVOS Y VARIABLES.

Es muy importante el nombre de los archivos, ya que en el caso de dar mantenimiento a un sistema se puede detectar facilmente si se le asigno un nombre que represente la función que realiza.

El nombre de las variables de trabajo usadas en un sistema es importante porque en función de la operación que realizan es de acuerdo cómo se les debe dar el nombre.

3.4.2 EL DIALOGO.

En la operación de un sistema lo más importante es la estructura del diálogo. ¿Qué va a decir la máquina al usuario? ¿Puede entenderla? ¿Esta confundido en alguna parte? ¿Sabe cómo responder o cómo iniciar un diálogo?. La estructura sicológica del diálogo es un tema complejo, los lineamientos pueden estar establecidos pero "no pueden reducirse a una ciencia exacta". Muchos de los diálogos en terminales requerian al operador recordar nemónicos y secuencias fijas de entrada, esto origino crear diálogos con un reducido número de caracteres.

Es importante que la mayoria de los usuarios finales potenciales no piensen cómo programadores y es notable la dificultad para que recuerden nemónicos, secuencias fijas y formatos. Hoy en día se estan viendo en algunos sistemas mejores diálogos para el usuario final. Es importante comprender que los principios de buenos diálogos serán ampliados a las aplicaciones de procesos avanzados.

Uno de los propósitos que el usuario persigue cundo se enfrenta a la operación de sistemas, es el de tener diálogos amigables. Los profecionales de procesamiento de datos pueden adoptar la técnica de los nemónicos, secuencias y escribir el código con la finalidad de hacer más fácil la implementación de funciones.

Los buenos factores humanos se ven repercitudos en el costo en cuanto a la disminución de ciclos de máquina y memoria necesaría para procesar todo lo que necesiten los usuarios, es decir, una implementación con factores humanos disminuye el costo de procesamiento. Las computadoras personales (P.C.) hacen posible pasar el código aprobado a un mainframe para compilarlo ya que estos tienen muchos más ciclos de máquina. Una computadora personal es baja en costo y se incrementa en poder.

Algunos diálogos estructurados que son faciles de usar consumen mucho tiempo de respuesta, en este caso nos enfrentamos a la siguiente disyuntiva:

Diálogos Faciles. Implican mayor tiempo de respuesta.
Diálogos Dificiles. Implican menor tiempo de respuesta.

La técnica de selección de menus lleva al usuario a mostrar varios menus cada uno con un tiempo de respuesta de 2 segundos, esto preocupa a la gente de procesamiento de datos, así que, el argumento que tienen es generar un nemónico en su lugar para agilizar esta técnica. Sin embargo, si los menus son almacenados en una P.C. rápida, un tiempo de respuesta de décimas de segundo se puede conseguir obteniendo mejores resultados.

Si se requiere seleccionar tres niveles de lista de opciones en una computadora personal sería fácil menejarlos con nemónicos y el usuario esta mucho menos expuesto a cometer errores. Una de las razones más importantes por las que se utilizan listas es por la velocidad con la que se pasa de pantalla a pantalla a través del sistema en ejecución. Los tiempos de respuesta de décimas de segundo de las computadoras personales cambian la técnica elegida de factores humanos, en dónde la opción de usuario amigable toma más tiempo. Es necesario elegir entre sistemas amigables o velocidad de respuesta con nemónicos.

3.4.3 NEMONICOS Y SECUENCIAS FIJAS.

Los nemónicos son comandos que previamente son implementados por software y que se caracterizan por tener una función especifica, de tal forma que al invocar alguno de estos se ejecuta una acción. Esta tarea de aprender nemónicos es difícil para los usuarios en la medida que se incremente el número de nemónicos a ser retenidos. También son usados en combinación con secuencias fijas, para indicar al computador que ejecute alguna acción especifica. Si la secuencia registrada no es reconocida, el computador enviará un mensaje de error.

3.4.4 LOS ERRORES.

Una falla mayor con algunos diálogos es que el usuario pueda llegar a un punto dónde él sepa que es lo que sigue. La terminal puede mostrar algo que posiblemente sucedio cuando el usuario oprimio una tecla accidental o erroneamente. Ahora, ¿que hace él? ¿no hay forma de continuar?. Al azar él registra otra entrada, pero esta, sólo lo mete en más problemas. La forma más adecuada para evitar estos problemas, es dotar al sistema de catálogos de errores para evitar que el usuario se encuentre con esos casos y no sepa que hacer. La realimentación al usuario tiene dos efectos:

1.- El usuario puede corregir rápidamente sus errores.
2.- La rápida realimentación es importante en el aprendizaje.

La realimentación rápida ha hecho que usuarios, particularmente con computadoras personales aprendan a usar rápidamente software complejo, tales cómo hojas electrónicas y también crear notablemente poca basura comparando con la programación tradicional. Un principio de los futuros lenguajes es que sean usados en construcciones hechas por usuarios para ver errores tanto cómo sea posible inmediatamente después de que se cometen, tener una realimentación rápida y poder ayudar al usuario.

La forma de los mensajes de error son importantes. El software puede decir exactamente cuando una operación o una entrada no es valida o cuando puede manejar muchos decimales, etc., esto puede explicarse cuando la operación o la entrada registrada no es valida. Cuando el usuario es detenido en su ejecución por un mensaje de error, tendrá la opción de accesar una pantalla de ayuda que le describa el error en el que ha incurrido. Una mala manera de los mensajes de error es que el computador le envie al usuario un código (pxt1759) al cuál se le pide consultar la descripción del error en un manual.

3.4.5 SINTAXIS Y SEMANTICA.

En los Lenguajes de 4a. Generación este aspecto es importante ya que nos permite hacer chequeos con lo cuál se consigue una más rápida realimentación al usuario acerca de cualquier error detectable. Un poderoso lenguaje de query que usa operaciones relacionales debe hacer unos chequeos sobre lo que esta siendo preguntado para avisar acerca de la semántica. La semántica puede ser de dos tipos:

Semántica Interna : Se refiere a si lo que se esta haciendo se refiere a reglas establecidas en áxiomas básicos.

Semántica Externa : Se refiere a si el sistema esta resolviendo bien los problemas.

La gente de desarrollo de sistemas al igual que los mecanismos generadores, pueden usar pantallas con opciones de selección rápida. Esto hará las herramientas fáciles y rápidas de usar y podrán gradualmente disminuir el número de errores cometidos en sintaxis.

En muchas ocaciones la necesidad de manejar información nos lleva a considerar que herramienta es la que el análista de sistemas necesita para hacer dicho manejo más rápido y fácil. El propósito de este punto es, desarrollar aplicaciones con el mínimo esfuerzo y máxima eficiencia.

3.4.6 LA INCERTIDUMBRE EN PROCESAMIENTO DE DATOS.

El solo hecho de implementar un sistema manejado por un usuario cambia los requerimientos. El analista de un momento a otro provee un servicio que fue sumamente difícil y que los usuarios no podrían resolver. Un operador ve información que previamente se le oculto y hace cambios en el área. Algunos usuarios no desean el sistema e insisten en usar sus propios métodos, otros quieren un nuevo tipo de reporte para cálculos que no estan en los requerimientos del sistema. Después de la implementación los usuarios tienen una base comun para la discusión del sistema. Es frecuente el caso que el usuario no sabe lo que quiere hasta que lo recibe, y cuando lo recibe requiere algo diferente.

El factor humano reta a los lenguajes de hoy, así para los usuarios es posible, aún ayudados por la gente de procesamiento de datos implementar y construir lo que ellos necesitan y ajustarlo continuamente. En el mejor ejemplo de Lenguajes de 4a. Generación usado, esto puede ser encontrado.

3.4.7 LA SATURACION EN COMPUTACION.

Los profesionales de computadoras han prosperado en la creación de nemónicos, puntuación y lo difícil que es recordar secuencias de caracteres. En el mundo de la computación de 4a. Generación es deseable involucrar al usuario tanto cómo sea posible. Para hacerlo es necesario evitar la saturación en computadora. Hay muchas técnicas para evitar la saturación, la sintaxis diferente para no tener la necesidad de recordar nemónicos, extrana puntuación y dificultad para recordar secuencias. A continuación se mencionar las características que evitan la saturación en computación.

- Menus.
- Multinivel de menus.
- Uso de lenguaje humano.
- Diagramas de acción (mostrar una especificación o programa estructurado).
- Un editor de diagramas de acción en el cuál el computador agrega programas de comandos a los diagramas de acción.
- Diagramas de acción combinados con un lenguaje humano que pueda ser interpretado por el software.
- Uso de iconos.
- Inicio de diálogo con el computador (en el que el computador pregunta al operador para dar de alta registros de información).
- Llenar los espacios en blanco del diálogo.
- Pantallas sensibles al tacto.
- Uso de un ratón para mover rápidamente el cursor, desplazamiento por todos los niveles del menú.

- Uso del tiempo de respuesta de décimas de segundo (en P.C.'s) para hacer desplazamientos rápidos de un lado a otro en los niveles de menus o en paneles de selección.

- Abrir paneles de ventana (que aparece sobre la pantalla para ayudar al operador).

3.4.8 FACTORES HUMANOS DESEABLES EN LOS LENGUAJES DE CUARTA GENERACION.

- El usuario podrá requerir aprender tanto cómo sea posible para poder empezar.

- El diálogo evitara forzar al usuario para recordar formatos o registro de secuencias.

- El diálogo nunca pondrá al usuario en situaciones en las que no sepa que hacer.

- El diálogo permitirá que el usuario se recupere de cualquier sorpresa o error, y podrá regresar facilmente al punto antes de la falla ocurrida.

- Todos los mensajes de error podrán ser explicados ampliamente.

- Las técnicas podrán ser seleccionadas para habilitar al usuario a obtener resultados tan rápido cómo sea posible.

- Si una técnica para obtener usuarios amigables es lenta o simple, el usuario podrá dar una técnica alterna, rápida y comprensiva para usarla cuando sea un experto.

- Las técnicas de gráficas podrán ser utilizadas dónde sea posible, cómo una ayuda para aclarar ideas.

- El uso completo de un diccionario de datos, directorio o enciclopedia.

- El lenguaje será designado para minimizar el beneficio de interración con pantallas, ventanas, pantallas divididas y rápido movimiento de apuntadores etc.

- El softaware podrá ser aprendido con buena calidad de computación agregando instrucciones que pueden ser invocadas en cualquier punto durante la construción de un programa.

Cada una de estas propiedades pueden encontrarse en algún Lenguaje de 4a. Generación, sin embargo, la mayoría de estos lenguajes se quedan cortos del rango tan completo de las cualidades de factores humanos aplicables.

3.5 PROGRAMA DE TRABAJO

La fase de diseno en un ambiente de lenguajes de 4a Generación consta de las siguientes tareas:

1.- Describir/disenar archivos y/o base de datos.

2.- Describir las funciones del sistema.

3.- Describir/disenar las salidas del sistema.

4.- Describir/disenar las entradas del sistema.

5.- Describir/disenar las interfases del sistema.

6.- Diseno de la estructura de software.

7.- Diseno por medio de dialogos.

8.- Describir/disenar la lógica del procesamiento del sistema.

9.- Construcción del prototipo.

10.- Preparación de las especificaciones de programas.

11.- Diseno de procesos de control, seguridad y respaldo.

12.- Preparación del plan de conversión de datos.

13.- Preparación del plan de entrenamiento.

TAREA NUMERO UNO.

Diseno De Archivos Y/O Base De Datos.

El objetivo de esta tarea es detallar el modelo de datos, verificar la normalización y describir entidades y atributos, en la forma más óptima posible de las estructuras de archivos y/o base de datos.

Todos los campos conocidos deben ser introducidos, ya sea en un manual o en un diccionario de datos, todos los requerimientos de acceso deben de ser identificados.

Se debe preparar una descripción narrativa para la implementación de los archivos, registros y tablas del sistema, se deben definir las longitudes de registros, factores de bloqueo, elementos de datos y llaves.

Este es el punto en el cuál el análisis de datos es usado, para refinar el agrupamiento lógico del almacenamiento de datos. Los diagramas de flujo de datos nos servirán cómo mecanismo para mover la definición del análisis de datos y su almacenamiento. Las rutas de acceso asociados con el almacenamiento de datos, pueden ser también determinados desde el flujo de datos.

Las tablas del sistema también deben ser definidas. Una tabla puede ser definida cómo un archivo relativamente estático dentro de una base de datos. Los datos contenidos dentro de una tabla son generalmente usados para:

- Control o monitores de proceso.
- Revisión de transacciones.
- Proporcionar información descriptiva a las salidas del sistema.

Deben ser determinados los métodos de acceso así cómo la estrategia apropiada, la estrategia debe de incluir decisiones; cómo se procesa un archivo por medio de llaves o secuencialmente, según las necesidades y facilidades que nos proporciona el Lenguaje de 4a. Generación.

TAREA NUMERO DOS.

Describir Las Funciones Del Sistema.

Para esta tarea es necesario un diagrama de flujo funcional del sistema apoyado por una descripción narrativa de las principales funciones. El diagrama de flujo debera mostrar:

- Las principales funciones que serán incluidas en la nueva aplicación.
- Entradas y salidas de cada función.
- Relaciones entre cada función y otras funciones que estan fuera del alcance del sistema (interfases).

La descripción narrativa debe proporcionar un panorama breve del diagrama de flujo y una discusión de cada función. Se debe identificar:

- Las funciones que serán automatizadas o computarizadas.
- Las funciones que estan actualmente computarizadas y para las cuales deberá proporcionarse una interfase.
- Un recorrido por el diagrama de flujo deberá ser seguido por el usuario, esto servirá para familiarizarlo con la visión del sistema y corregir cualquier falla.

TAREA NUMERO TRES.

Diseno De Las Salidas Del Sistema.

El objetivo de esta tarea es detallar las salidas del sistema. Las salidas del sistema incluyen reportes, pantallas, archivos e información para otros sistemas. El contenido de las salidas debe de estar ya definido y las salidas en forma visual deben tener desplegados sus contenidos.

Generalmente, las salidas visuales son desplegadas cómo reportes o pantallas, para facilitar la comunicación con los usuarios y la administración. Esto es muy reelevante si el sistema está siendo implementado para proporcionar información que nunca antes ha estado disponible.

Al disenar reportes y pantallas debe tomarse en cuenta el monto de información que puede ser desplegada en una pantalla o en una hoja de papel de computadora.

El criterio de selección y ordenamiento para todos los reportes debe ser bien definido.

Se debe considerar lo siguiente para los reportes y pantallas:

- Descripción del contenido de reportes y pantallas, incluyendo todos los datos fuente.
- Formatos detallados de reportes a pantallas en una forma apropiada de codificación.
- Explicación de todos los cálculos a ejecutarse.
- Descripción de los propósitos de cada salida, incluyendo una identificación de sus usuarios primarios.
- Descripción de restricciones de acceso.

Lo importante en esta etapa es la realimentación del usuario.

TAREA NUMERO CUATRO.

Diseno De Las Entradas Del Sistema.

Los objetivos de esta tarea son: ampliar la descripción y el nivel de detalle de las entradas del sistema, debemos considerar la frecuencia y que se establezca claramente lo siguiente:

- Volúmen anticipado de información.

- Medio de almacenamiento.

- Elementos de datos principales.

La descripción de las entradas del sistema deben incluir una discusión de los tipos de revisión que serán ejecutados, deberá de prepararse a nivel de elementos de los datos y describir que es lo que la revisión va a lograr en lugar de cómo va a lograrse. Por citar un ejemplo, se puede especificar que cierto campo de un archivo será validado contra una tabla, pero no especificará el medio físico para almacenar y accesar la tabla. Para el diseñador de sistemas es importante, indicar la frecuencia con la cuál la tabla cambiará.

Los elementos de datos en las transacciones, incluyendo aquéllos utilizados cómo apoyo en las soluciones de proceso de datos deben ser definidos en el diccionario de datos.

Las longitudes y tipos de todos los elementos de datos de entrada deben ser especificados. Se deben preparar los detalles de transacciones y formatos de pantallas.

El establecer una estructura de clasificación de transacciones en esta fase, puede facilitar el proceso de definición de los requerimientos de entrada. Mientras que la estructura de clasificación sea única para cada usuario y sistema, una estructura de transacción general puede incluir:

- Mantenimiento de tablas y archivos.

- Corección de errores.

- Interfases de sistema automatizados.

- Entradas al sistema generado.

- Peticiones de generaciones de salidas.

Subtareas.

a) Describir las transacciones con detalle, de las funciones a efectuarse.

b) Actualizar frecuencias y volúmenes esperados.

c) Identificación de las medias de las entradas.

d) Definir todas las transacciones y formatos de pantallas.

e) Diseñar las formas de entrada (documentos fuente).

TAREA NUMERO CINCO.

Diseño De Las Interfases Del Sistema.

Describir y diseñar las interfases requeridas para implementarse al sistema. Una interfase del sistema puede ser una entrada, una salida, un archivo compartido o bien una estructura de base de datos. La información requerida por la interfase debe ser identificada.

- Describir y diseñar las interfases existentes para sistemas (internas y externas).
- Describir y diseñar las interfases entre los sistemas de aplicación.

Para que una interfase quede bien identificada es necesario; definiciones detalladas, descripción de los datos y protocolos de información de dichas interfases.

TAREA NUMERO SEIS.

Diseño De La Estructura Del Software.

El objetivo de esta tarea es diseñar la estructura del software a un nivel de detalle para identificar los programas individualmente y para validar que los objetivos a realizarse sean llevados a cabo. Se presentan 4 niveles de diseño:

Nivel Sistema.	Por ejemplo Sistema Modular Financiero.
Nivel De Aplicación.	Por ejemplo libro mayor, cuentas por pagar cuentas por cobrar.
Nivel De Subsistema.	Subsistema de Entrada/Salida.
Nivel De Programa.	Asentamiento en el libro mayor.

Si las aplicaciones son integradas, no es necesario una distinción entre el sistema y los niveles de aplicación.

La estructura de software es una representación jerárquica, que nos muestran la interrelación entre diversos elementos (módulos) de una solución de software a un cierto problema.

El software procedural explica en forma detallada los procesos a ejecutarse en un módulo. Se deberá identificar en dicha estructura que tipos de módulos la componen dependiendo de la función de los módulos, y pueden ser de los siguientes tipos:

Control o Interfase Lógica.	Selecciona la parte del Proceso requerida.
Proceso.	Realiza el trabajo de una función determinada.
Rutinas Auxiliares.	Son los módulos comunes.
Interface De Archivo.	Entrada y salida a periféricos.
Errores y Excepciones.	Realizan manejos de la información no requerida.

Es necesario también considerar los módulos de acuerdo a lo que van a ejecutar, se pueden tener los siguientes:

- SECUENCIALES.	Son ejecutados completamente sin interrupcion del software aplicativo.
- INCREMENTALES.	Se puede tener una interrupción de software y reiniciarse posteriormente en el punto de interrupción.

- PARALELOS. Dos o más módulos pueden ejecutarse simultáneamente.

Para identificar la independencia entre los módulos es necesario tener presente que tanto acoplamiento o cohesión se desea tener entre los módulos.

Es preciso aclarar que en la estructura de software, los módulos no realizan actividades de tipo operativo, también es conveniente que se muestre el ancho así cómo la profundidad de la estructura del software en dicha estructura, debe de quedar bien especificado el número de salidas (Fan-Out), así cómo el número de entradas (Fan-In) a los módulos.

TAREA NUMERO SIETE.

Diálogos.

El diseno por medio de los diálogos, puede ser de gran ayuda para los analistas y los disenadores de el sistema o para usuarios finales.

La ventaja de disenar por medio de menus o panels de instrucciones es que éstos son muy concisos y relativamente simples. Para el disenador es muy rápido hacer cambios y se facilitan las modificaciones al sistema. Los panels de instrucciones son de gran ayuda para el disenador si es que algunos aspectos del sistema fueron olvidados tales como; efecto de fallas, teclas que no serán validas o requerimientos de seguridad por citar algunos de éstos.

Los diálogos van indicando cómo el software procede, se despliega un submodelo de los datos de entrada mostrando las acciones que pueden ocurrir por medio de un menú.

Si el analista cuenta con una herramienta computalizada que realize el diseno, la máquina le formulará preguntas reelevantes hasta llegar a completar la ejecución del código que describa el diccionario de datos que contiene, la definición de los campos y parámetros, así cómo formatos de reportes. De esta forma los lenguajes de 4a Generación minimizan el trabajo de crear la documentación necesaria.

TAREA NUMERO OCHO.

Diseno De La Lógica De Procesamiento Del Sistema.

El objetivo de esta tarea es describir y disenar la arquitectura de los procesos lógicos de los datos que serán ejecutados por el sistema, cómo apoyo para la clasificación y codificación de las estructuras, así cómo las reglas del sistema.

Los tipos de procesamiento de datos que generalmente requieren solución por citar algunos ejemplos pero no son los únicos son los siguientes:

- Reglas para ejecutar un chequeo de crédito en una aplicación de órdenes de entrada.
- Algoritmos para calcular el interés en cuentas de ahorro para una aplicación bancaria.
- Lógica para el cierre de fin de mes y/o fin de ano para cualquier tipo de aplicación.

En el ambiente de Lenguajes de 4a. Generación, dónde la lógica está usualmente contenida en las entradas y salidas del sistema puede servir cómo punto de revisión por si hubiese algo que no se había contemplado.

La complejidad de la lógica de procesamiento dictará la técnica óptima para su documentación y la comunicación con el usuario.

Generalmente, el texto narrativo sin ilustraciones, no es adecuado ya que es ambiguo y largo. Los diagramas de flujo o las cartas estructuradas son apropiadas para la mayoria de las descripciones de procesamiento.

Las tablas de decisión pueden ser requeridas para una lógica de procesamiento muy compleja.

Esta tarea es muy significativa para un ambiente de 4a. Generación, el disenador de sistemas determina cómo serán realizados los requerimientos del Proceso de Datos. Algunos de los factores a considerar para esta determinación son:

- Utilización de Tablas.Usadas para proceso de control, o para el manejo de información descriptiva.
- Lógica Descriptiva. Para sistemas de contabilidad o financieros que deben proporcionar reportes mensuales, deben contener los archivos acumulados mensuales o a una fecha determinada.
- Corrección Lógica de Errores. A que grado debe el sistema dar apoyo para corregir una transformación identificada cómo inválida.
- Actualización Lógica. Secuencia de actualización de archivos o base de datos, si dicha actualización debe ser en línea o en batch.

TAREA NUMERO NUEVE.

CONSTRUCCION DEL PROTOTIPO.

Cómo antes se ha mencionado, un prototipo es una representación del producto formal de programación. Por lo regular tiene funcionamiento limitado en cuanto a capacidades, confiabilidad o eficiencia. El objeto en si es que el usuario tenga participación directa en el desarrollo del sistema.

Del mismo modo que el sistema general, el prototipo necesita de un trabajo de análisis, diseno, desarrollo, implementación y pruebas, desde luego que a su nivel.

La construcción del prototipo se realiza después de la conceptualización inicial del sistema en una forma general. Los prototipos son sistemas que se desarrollan por medio de un proceso iterativo y se retroalimentan conforme se van probando, por lo que involucra los bloques 2, 3 y 4 del ciclo de vida de desarrollo con prototipos, los cuales son mostrados en la figura 3.8.

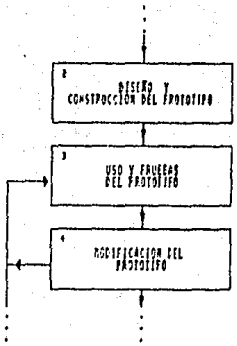


FIGURA 3.8. BLOQUES 2,3 Y 4 DEL CICLO DE VIDA DE DESARROLLO CON PROTOTIPOS.

Cómo se expuso en el capítulo de "Generalidades", este desarrollo se comporta de una forma evolutiva, que implica varias fases, o en ocasiones todas ellas, en cada iteración. Algunas veces será preciso hacer ajustes al diseño del prototipo, e incluso habrá ocasiones en que será necesario volver al punto de análisis de requerimientos.

Podemos considerar al primer producto de la construcción del prototipo, cómo la versión uno, a medida que este se va modificando y sofisticando, el número de versión irá aumentando, hasta que la versión "n" se pruebe cómo la correcta.

De acuerdo a todo lo antes mencionado, se expone la figura 3.9 que ilustra el flujograma de la forma en que puede iterar la construcción del prototipo.

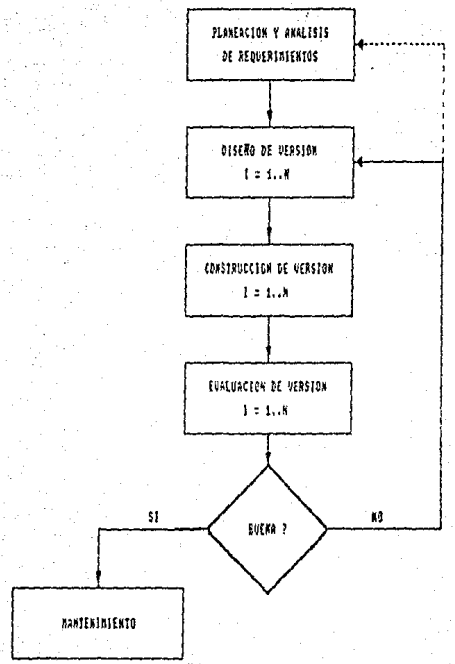


FIGURA 3.5. ITERACIONES EN LA CONSTRUCCION DEL PROTOTIPO.

DISEÑO DEL PROTOTIPO.

Para diseñar el prototipo, se deben tomar en cuenta todas aquéllas funciones que se consideren cómo más representativas del sistema. Puede iniciar con unas cuantas e irse agrandando al agregarsele algunas otras que se vayan detectando sobre la marcha. También se puede iniciar con un conjunto completo de funciones, y expandirse o contraerse a través de su uso y la experiencia. Este prototipo creado tendrá la característica principal de mostrar al usuario la forma en que será automatizada una determinada función, que fue previamente implementada y probada por la gente de sistemas, para someterla a juicio del propio usuario.

El objetivo que se persigue con esto, no es más que tener un punto de evaluación hacia el prototipo, de tal forma que se vayan haciendo los ajustes a la función en cuestión.

Se sugiere que en este proceso se consideren las funciones por nivel de jerarquía, de tal modo que se vayan desarrollando, primero, aquéllas que presenten el nivel superior, y una vez concluidas éstas, continuar a los niveles inferiores de una función determinada siguiendo el mismo proceso hasta completarla, entendiendose por completar una función, al hecho de haber concluido también, con todas las subfunciones que dependen de ésta. Se prosigue con las funciones siguientes del mismo modo, aplicando a cada una el ciclo de vida propuesto anteriormente.

Es importante no continuar con otra función si no se ha terminado con la que se esta trabajando, para no acumular actividades pendientes.

En base al diseño preliminar se tomarán las consideraciones pertinentes para elaborar el prototipo, sus características y extensión dependen directamente de la naturaleza del sistema, de tal modo, que los componentes del diseño que deben integrar el prototipo, estan sujetos a cada proyecto en particular. Para ésto podemos tomar en cuenta los siguientes criterios:

- Mostrar lo más representativo, aquéllo que de alguna manera identifique los objetivos básicos del sistema.
- Utilizar lo que este mejor definido, aquéllo que sea más claro sobre los requerimientos del usuario.

- Tratar de usar lo más sencillo, evitar los componentes muy complejos para desarrollar el prototipo en el menor tiempo posible.

- Pensar en la versatilidad del prototipo, se necesita la capacidad de efectuar modificaciones rápidas y fáciles.

- Presentar aquéllo que pueda tener mayor interacción con el usuario, para que éste tenga un marco de referencia en base a que juzgar. Cuando un usuario ve algo tangible en la pantalla o en papel, es más fácil que precise que información le hace falta o cuales serán sus cuestionamientos sobre el funcionamiento de su sistema.

- Manejar la información más relevante. Incluir los datos que realmente sean indispensables para mostrar la idea fundamental del sistema.

- Considerar los componentes que puedan tener más alta prioridad para el usuario, los que necesite a la mayor brevedad posible.

El prototipo generalmente no contiene todas las características de un sistema terminado; por ejemplo, la organización de archivos puede ser temporal y las estructuras de los registros incompletas, pueden faltar controles de entradas y validaciones, los formatos de reportes y pantallas pueden ser provisionales, etc. Normalmente falta toda la documentación.El diseno del prototipo se realiza en una forma similar al diseno del sistema, por lo que implica el manejo de las mismas tareas:

1.- Funciones del prototipo.
2.- Salidas del prototipo.
3.- Entradas del prototipo.
4.- Interfases del prototipo.
5.- Archivos/bases de datos del prototipo.
6.- Estructura del software del prototipo.
7.- Diálogos prototipo.
8.- Lógica de Procesamiento del prototipo.

Sin embargo, de acuerdo a los criterios anteriormente expuestos, se decide cuales tareas deben integrar el diseno del prototipo y a que nivel quedarán las que se hayan considerado. En una etapa posterior de diseno, al iterar en el ciclo de realimentación, las mismas tareas pueden sufrir cambios o ser incluidas algunas que no fueron antes tomadas en cuenta, para que constituyan la nueva versión del prototipo.

EL PROCESO DE DESARROLLO Y EVALUACION DE PROTOTIPOS.

El sistema prototipo se crea con rapidez, es de bajo costo pero no necesariamente es tan eficiente cómo un sistema que se desarrolla con más detalle. Se debe hacer con la idea de una fácil modificación. Para conseguir ésto, hay que aprovechar las facilidades y herramientas de los "Lenguajes de Cuarta Generación", cómo los generadores de pantallas, reportes y aplicaciones, las facilidades de creación y modificación de bases de datos, y muchas otras que ya han sido antes expuestas con mayor amplitud.

Una vez que queda armado el prototipo, se procede a su evaluación, de acuerdo al avance de éste, puede progresar, desde una demostración inicial a un grupo reducido de usuarios, hasta una más completa y refinada prueba con bastantes usuarios. Algunos cambios requeridos por ellos pueden efectuarse inmediatamente en la pantalla, otras observaciones son anotadas para modificarse posteriormente.

Algunas veces los prototipos iniciales pueden ser inapropiados, tienen detalles que afinar y/o cosas que crear o suprimir. Cuando uno alcanza un estado aceptable, varios usuarios pueden ser entrenados para usarlo y experimentar con él. Para demostrar su uso y manejo, se pueden plantear cuestiones que se consideren que el sistema debe ser capaz de responder, los responsables de su creación deben demostrar esta habilidad, o en su defecto, tomarlo en cuenta para la adecuación del prototipo.

La información que se obtiene a través de su uso, se aplica a un diseno modificado. La versión modificada, puede usarse a su vez, cómo el prototipo para obtener aún más información del diseno. Cada versión incorpora las capacidades anteriores, con los ajustes correspondientes, así cómo nuevas funciones que puedan agregársele. El proceso se repite tantas veces cómo sea necesario para revelar los requerimientos escenciales del diseno del sistema.

Las tareas previas a ésta, deben ser ajustadas según los últimos requerimientos que se hallan encontrado durante el manejo del prototipo, es importante no seguir adelante con las siguientes tareas mientras dichos requerimientos no queden perfectamente definidos.

Finalmente, cuando un prototipo es aceptado en plenitud por el usuario, todas aquéllas características y componentes que hayan sido manifestados, repercutirán en el diseno e implementación del sistema sobre una base mucho más firme.

TAREA NUMERO DIEZ.

Preparación De Las Especificaciones De Programas.

El objetivo será armar módulos independientes, de esta forma se produce un sistema confiable que facilite su implementación y mantenimiento en el ambiente de Lenguajes de 4a. Generación el detalle dependerá de la experiencia del equipo de trabajo. Se prepararán las especificaciones que proporcionarán toda la información necesaria para programar y probar:

- Rutinas y módulos comunes.
- Programas de aplicación.
- Programas de conversión de datos.

Hay tres partes para las especificaciones de módulos comunes:

- Descripción de la función que realiza el módulo, cómo se ejecuta y sus consideraciones de procesamiento de datos.
- Descripción de los requerimientos para el uso del módulo.
- Descripción de la lógica interna del módulo y los parámetros de entrada/salida. Esta descripción puede ser en una serie de maneras, dependiendo del tamaño y complejidad del módulo. Generalmente, sería en una o más de las siguientes formas:
    * Narrativa detallada.
    * Pseudocódigo.
    * Diagrama de flujo.
    * Tablas de decisión.

Para asegurar uniformidad y calidad en el software, deben de estar muy claros los estándares de técnicas de diseño y métodos de control. Hay una serie de secciones para las especificaciones de programación:

- Se describe el programa y se expresa su propósito.
- El diagrama de flujo contiene el número de programas, títulos, funciones de entrada y salida, pueden ser representados por:

* Diagrama de bloques.
* Carta estructurada.
* Diagramas HIPO.

- Los formatos de registro, base de datos, transacciones, reportes y pantallas debieron quedar descritos con anterioridad.

- Se realiza una descripción de todos los métodos y técnicas a ser usadas. Esto puede ser un compendio de:

  * Código y mensajes.
  * Glosario de banderas.
  * Rutinas y módulos comunes.

Cualquier cambio, por muy urgente que sea, debe de ser comunicado al líder del proyecto, para que se lleve a cabo un análisis adecuado de costo-beneficio.

El diseñador será responsable de:

- Rutinas y módulos comunes.

- Programas de aplicación incluyendo interfaces del sistema.

- Programa de conversión de datos.

Un beneficio del diseño del software es la factibilidad de identificar módulos y rutinas comunes, estos módulos pueden ser implementados cómo subprogramas o subrutinas para ser usados por varios programas. La identificación y desarrollo de módulos comunes no sólo reducen el esfuerzo de implementación del sistema, además de ser confiable y fácil de mantener y modificar. Algunas funciones de los sistemas son comunes para casi todas las aplicaciones. El sistema debe contener módulos comunes, tipo de utilerías para los siguientes tipos de funciones:

- Rutinas principales.

- Rutinas de error.

- Rutinas de lectura escritura.

- Rutinas de actualización.

En suma, cada aplicación del sistema contendrán funciones de uso común que son únicas para las aplicaciones. Los módulos comunes pueden ser derivados de aplicaciones que generalmente son similares.

TAREA NUMERO DOCE.

Diseno Del Proceso De Control, Seguridad Y Respaldo.

Uno de los objetivos de esta tarea es asegurar que los suministros efectivos de control sean incluidos en el diseno del sistema. Algunos suministros de control típicos, para lenguajes de 4a. Generación, incluyen:

- Verificar las entradas y salidas del sistema descritas en fases anteriores.

- Diseno de tablas que faciliten el control cómo son tablas del sistema.

- Adecuar archivos históricos, de errores y de procedimientos.

Otro de los objetivos de esta tarea es asegurar que los procedimiento apropiados de seguridad sean incluidos en el diseno del sistema. Los suministros adecuados de seguridad frecuentemente dependerán de las características de las instalaciones de hardware y software. Es responsabilidad del disenador del sistema utilizar apropiadamente los componentes de seguridad de la instalación para proporcionar una aplicación correcta.

Por otro lado esta tarea se encarga también de asegurar que los procedimientos adecuados de respaldo y recuperación sean considerados en el diseno del sistema. Los requerimientos del proyecto y la importación del sistema bajo el desarrollo, influirán considerablemente para el alcance de esta tarea. En un ambiente de actualización en línea, la carga de transacciones antes y/o después de la carga de archivos imagen debe ser considerada. Los procedimientos de respaldo y recuperación pueden responder a los siguientes tipos de preguntas:

- Se incluyen programas para copiado periódico (dumping) de archivos.

- El tiempo de procesamiento para archivos "DUMP" va acorde con los requerimientos de procesamiento de archivos (frecuencia, tamano de archivos, etc.) para asegurar una recuperación efectiva.

- Los archivos son actualizados con alta prioridad.

- Existe duplicidad en archivos de datos significativos creados simultáneamente para proporcionar la recuperación.

- Sería posible una rapidez razonable de recuperación para operaciones normales después de que una falla ha sido resuelta.

El departamento interno de auditoría del usuario o su equivalente debe involucrarse en la reversión de los suministros de control del sistema que debe ser requerido para la autorización de los controles o documentación de todos los procedimientos de seguridad. Se debe tener en cuenta la revisión de los procedimientos de respaldo y recuperación.

TAREA NUMERO TRECE.

Preparación Del Plan De Conversión De Datos.

El objetivo es delinear todas las actividades manuales y automáticas, requeridas para una conversión correcta de datos al nuevo sistema. Estas actividades deben de ser calendarizadas e identificadas sus responsabilidades.

Las primeras actividades de ésta tarea deben de empezar antes de la conversión real. Generalmente la primera actividad es para identificar la naturaleza de los datos del sistema viejo. El proceso probablemente sea lento, y antes de la conversión, el usuario necesitará dedicar una cantidad considerable de tiempo para depurar la información. Es posible que los programas de conversión sean requeridos para descargar los archivos viejos y para reformatear y cargar a los nuevos formatos de archivos. El nuevo sistema generalmente incluye tablas que tendrán que ser cargadas por primera vez. Otros aspectos de conversión que requerirán planeación y preparación incluyen:

- Procedimientos de recompilación de archivos.

- Ordenar las nuevas formas.

- Entrega e instalación del nuevo equipo.

- Impresión y distribución de nuevos manuales.

- Conversión en la actualización de la lógica.

- Procedimientos de recuperación en caso de una conversión defectuosa o falla del nuevo sistema.

Se debe nombrar un coordinador de conversión, preferentemente un usuario, para controlar ésta parte del proyecto.

TAREA NUMERO CATORCE.

Preparación De el Plan De Entrenamiento.

El objetivo de esta tarea es asegurar que exista una estrategia efectiva de entrenamiento y que haya sido incluida en el calendario del proyecto. La gente representa el problema más crítico de proceso en un sistema de información basado en computadora. Por esto, un entrenamiento adecuado del usuario es un asunto primordial cuando se traduce un nuevo sistema de 4a Generación. La estrategia de entrenamiento debe de ser proyectada a varios niveles como, usuarios, directivos, técnicos y operadores incluyendo a todo el personal, sea experto o inexperto.

III.4 IMPLEMENTACION.

INTRODUCCION.

En su intento de poner al alcance y facilitar el uso de los Lenguajes de Cuarta Generación hacia un mayor número de usuarios, éstos han sido diseñados considerando las construcciones de control de la programación estructurada; ejemplo de ésto lo representan los lenguajes de programación LINC, NATURAL e INFORMIX.

Como ya se mencionó en el capítulo de generalidades, el ciclo de vida del desarrollo con prototipos es un proceso iterativo, en el cual después de implementar un sistema y operarlo, empiezan a surgir detalles que no estaban previstos o de interpretación, como se menciona en el capítulo de factores humanos; es por esta razon, que de acuerdo a la metodología propuesta, podemos vernos obligados a caer varias veces en la fase del ciclo de vida con prototipos de implementación, como lo muestra el diagrama del ciclo mencionado anteriormente.

El objetivo de esta fase está enfocada a instalar el sistema de 4a. Generación probado y aceptado por el usuario en un ambiente de producción. Al ejecutarse dicha fase se desarrolla el código fuente, verificación de las tablas y archivos así como el volumen de los datos históricos, pruebas para el nuevo sistema que será liberado, preparación de programas, sistemas y operación, así como la documentación para el usuario, entrenamientos para la gerencia y sus usuarios, desarrollo y prueba de programas y/o procedimientos de conversión de datos.

Esta fase es muy compleja desde la prespectiva de administración de proyectos por el número y tipo de profesionales requeridos y la estrecha coordinación que se necesita entre:

- Los miembros del equipo del proyecto.

- La gerencia del usuario.

- Los usuarios.

- Los operadores del computador.

La habilidad empresarial del gerente del proyecto es importante para el éxito de esta fase. El gerente del proyecto debe ser capaz de identificar los problemas antes de que se hagan criticos y actualizar las fases anteriores de acuerdo como sean requeridas por las circunstancias. Un problema critico que puede ocurrir es que el equipo encargado del desarrollo determine que el diseno, como fue especificado no cumpla con los requerimientos, funciones y ni con los objetivos. El programa de trabajo que se presenta, consta de una serie de tareas en las cuales se mencionan técnicas recomendadas y documentación para el control del sistema.

4.1 CARACTERISTICAS DE LA PROGRAMACION ESTRUCTURADA QUE DEBEN CONTENER LOS 4GL'S.

En general, todas las estructuras de programación estructurada son usadas por los lenguajes, ya que muchas de estas son cotidianamente implementadas en base a las estructuras tradicionales.

La programación estructurada es una continuación de la metodologia "DE ARRIBA HACIA ABAJO", que usan la carta estructurada y el pseudocódigo. Una de las razones por las que se usa la programación estructurada es debido a su sentido práctico, ver figura 3.1.

La Programación Estructurada es:

1.- Consistente con la metodologia "DE ARRIBA HACIA ABAJO".
2.- Fácil de depurar, probar y mantener que otra no estructurada.
3.- Fácil de revisar.
4.- Facilmente leible.

FIGURA 3.1. BENEFICIOS DEL CODIGO ESTRUCTURADO.

4.2 PROGRAMACION ESTRUCTURADA.

Otra razón para utilizar programación estructurada, es que ayuda a mantener una estructura de programación modular. El codigo de programación generalmente sigue la misma secuencia física y lógica. La programación no estructurada rompe la secuencia de ejecución y brinca de un lugar a otro. La programacion estructurada fluye física y lógicamente desde arriba hasta el fondo, o desde el inicio hasta el final. Por supuesto, hay excepciones de este flujo, pero si las técnicas de diseno son rigurosamente aplicadas, el programa se apegará a ellas.

La programación estructurada estimula y simplifica la revisión que contribuye a la calidad del programa. La programación estructurada es una técnica para asegurar la precisión o exactitud de un diseno o programa implicando una segunda parte en la evaluación del producto.

Un beneficio final de la programación estructurada es el mejoramiento en la redacción del programa.

4.2.1 ESTRUCTURAS BASICAS DE CONTROL.

La estructuras básicas de control son: Secuencia, Selección, Iteracción y Case, estas cuatro construcciones son disenadas sin la necesidad de que el programador use la instrucción GO TO, sin embargo algunos lenguajes pueden requerir de su uso para implementar una o más estructuras de control.

Cada una de las construciones son consideradas como "cajas negras" donde una cierta actividad ocurre. La importancia de la caja negra es que hay sólo una entrada y una salida.

La función de la caja es representada por un bloque de código del programa que consta de una de las cuatro estructuras de control o una combinación de ellas. Permitiendo múltiples entradas y salidas se complica el código y alienta una programación sucia, que en consecuencia hace difícil su depuración.

4.2.2 ESTRUCTURAS DE CONTROL Y SECUENCIA.

La estructura básica consta de una secuencia de dos o más instrucciones de lenguaje que son ejecutadas en el orden que aparecen. No hay decisión o iteración en esta estructura.

4.2.3 ESTRUCTURAS DE CONTROL Y SELECCION.

La selección o estructura IF THEN ELSE elige desde dos posibles rutas basado en el resultado de evaluar el predicado; que puede ser cualquier expresión booleana válida.

4.2.4 ESTRUCTURAS DE CONTROL E ITERACION.

Esta estructura es conocida como DO WHILE o DO UNTIL, similar a un comando de repetición. Hay una significativa diferencia entre las anteriores estructuras. En el DO WHILE el predicado se evalua primero y si es verdadero la acción definida es ejecutada. El predicado se evalua nuevamente, hasta que la operación es falsa, tiempo en el cual el proceso terminará y entrará al punto de salida.

4.2.5 ESTRUCTURAS DE CONTROL CASE.

Esta estructura es condiderada opcional, a través de su lógica puede ser implementada con una estructura de control y selección. Algunos lenguajes implementan esta estructura directamente con una estructura de lenguaje equivalente, pero la mayoría requiere alguna adaptación de comandos disponibles.

4.3 ´ QUE CONSTITUYE UN BUEN PROGRAMA ?

Un buen programa es aquel que realiza una función que otro programa intentó hacer y que trabaja de acuerdo a las especificaciones dadas. El considerar el tamano de la memoria y la eficiencia podrían ser importantes, especialmente si se usa una computadora pequena.

Si un programa es usado o leído rara vez por una sola persona, ésto no existe en la vida real, principalmente porque se dan muchos movimientos de programadores, y se origina que los que desarrollaron un proyecto no esten cuando se necesite darles mantenimiento.

Los puntos más importantes para que un programa sea considerado bueno son: fácil de leer y entendible. Estas cualidades contribuirán para hacer pruebas de depuración con bastante facilidad con un mínimo de tiempo y dinero. En síntesis, un buen programa debe contener lo siguiente:

1.- Trabaja de acuerdo a la especificaciones.

2.- No hace uso excesivo de memoria.

3.- Corre eficientemente

4.- Fácil de leer.

5.- Fácil de entender.

6.- Fácil de depurar y probar.

7.- Puede ser mantenido con un mínimo de esfuerzo.

4.4 ESTANDARES.

Los estándares deben de mantenerse y respetarse mientras se consideren vigentes, a menos que se efectue un consenso, que certifique que los estándares son obsoletos y unos nuevos deben remplazar los viejos.

Los estándares adaptados para el desarrollo de un sistema deben cumplirse para la totalidad de éste. Si surge la necesidad de efectuar cambios, deben ser aplicados hasta el siguiente desarrollo, a menos que sea imperativo realizar las modificaciónes inmediatamente, en este caso, los estándares viejos deberán ser sustituidos en todos los programas para mantener la consistencia del sistema.

4.5 EFICIENCIA.

La eficiencia es una tendencia para usar recursos críticos, en lo que se refiere a la eficiencia del código, eficiencia de la memoria y en las entradas/salidas.

Algunos lenguajes permiten al programador escribir una línea de código conteniendo más de una instrucción, otra manera de escribir el código sería una instrucción por línea, esto toma más espacio pero los beneficios son considerables y proporciona una gran facilidad para leer el código. Un segundo beneficio es el caso de insertar unas instrucciones adicionales como resultado de las pruebas o mantenimiento del programa, de esta manera es fácil de modificar.

Hay veces que las instrucciones exceden la logitud de línea, la mayoría de los compiladores aceptan continuar con la siguiente línea con alguna marca, una mejor solución sería mandar a la siguiente línea una frase completa en lugar de dividir alguna palabra.

El uso de constantes dentro de los programas tiende a crear dificultades cuando los cambios son hechos.

Es usual que los programadores usen comentarios dentro de los programas, pero lo que no es usal es que se utilizen de una forma métodica. Estos comentarios son de mucha utilidad ya que ahorran tiempo y dinero. Hay tres tipos generales de comentarios que son: comentarios de prólogo, de módulo y de instrucción.

4.6 EJEMPLOS DE LENGUAJES DE 4a. GENERACION.

4.6.1 I N F O R M I X .

Informix-4GL, es un lenguaje de cuarta generación muy poderoso. Consta de dos módulos que son: I4GL y SQL en el primero se maneja la base de datos, se generan pantallas y menus así como reportes por medio de su lenguaje de programación y SQL es el administrador de la base de datos y tablas, en el cual se puede crear la base de datos y tablas, diseno de pantallas y menus.

I N F O R M I X - I 4 G L .

Este lenguaje de cuarta generación, provee todas las herramientas necesarias para crear un sistema administrador de base de datos relacional. Informix-I4GL es:

- Un lenguaje de base de datos en la cual se puede almacenar, recobrar, actualizar y borrar información de una base de datos que ha sido creada.

- Es un lenguaje de programación.

- Sirve para construir pantallas.

- Sirve para construir menus.

- Escritor de reportes.

Informix-I4GL está disponible en sistema operativo UNIX y DOS (PC-DOS y MS-DOS).

Es una herramienta desarrollada para crear bases de datos relacionales y provee el uso para manejar con gran facilidad los datos almacenados en ella.

Informix-I4GL es creado por RDSQL, RD es sólo una extensión y SQL (lenguaje estructurado para la manipulación de información en una base de datos) desarrollado por IBM. SQL ha llegado rápidamente a convertirse en un estándar de lenguajes para sistemas administradores de datos. RDSQL ofrece especificaciones como son:

1.- Creación de la base de datos.

2.- Insertar información en la base de datos.

3.- Seleccionar información de la base de datos.

4.- Actualizar información de la base de datos.

5.- Borrar información en la base de datos.

Así se puede llevar a cabo programas que permiten el uso para anadir, recobrar, actualizar y borrar información de la base de datos.

Es diseñado para aplicaciones en la base de datos incluye especificaciones que pueden encontrarse en lenguajes de propósito general. Las especificaciones básicas para asignar valores, construcción de ciclos e instrucciones condicionales son provistas por informix-I4GL.

Se pueden asignar valores usando especificaciones como LET, WHILE y FOR para ciclos en los programas. El IF y el CASE realizan la misma función que corresponde en C y en PASCAL.

Provee estructuras de datos como son, registros y arreglos que permiten manipular muchos valores simultaneamente.

Los programas pueden llegar a ser muy largos y complejos, usando la especificación FUNCTION, que permite crear subrutinas.

Informix-I4GL incluye una utileria llamada FORMBUILD que permite crear pantallas con un mínimo de esfuerzo, esta utileria permite automáticamente generar una pantalla para cualquier tabla en una base de datos.

Asi mismo provee una especificación MENU que simplifica el proceso de crear menus, con una mínima cantidad de código se pueden ir creando submenus.

Consta de una serie de menus que guian a el usuario a través de pasos para el desarrollo de una aplicación. Se puede trabajar en módulos de programas, crear y compilar pantallas y además el compilar y ligar módulos para así crear programas multi-módulos y se puede hacer uso de informix-SQL.

Informix-I4GL incluye funciones de librerias y diversas utilerias, programas que checan y almacenan la integridad de los indices de archivos, carga datos de diferentes fuentes.

I N F O R M I X - S Q L .

Informix-SQL, es un sistema administrador de base de datos que consiste de programas o módulos disenados para ejecutar tareas del administrador de base de datos. Un buen sistema administrador de base de datos puede reducir sustancialmente la cantidad de tiempo requerido para organizar, guardar y recuperar la información.

Con informix-SQL se puede llevar a cabo las siguientes tareas:

- Creación base de datos y tablas.
- Diseno de pantallas.
- Modificación e introducción de datos usando las pantallas.
- Cargar datos de archivos del sistema operativo.
- Correr aplicaciones por pantalla o por medio de un lenguaje interactivo.
- Producir reportes.
- Diseno de menus que pueden incluir elementos, utilerias del sistema y otros programas de informix-SQL.

Los identificadores incluyen tablas y nombres de columnas, entre otros. Cada identificador puede tener una longitud hasta de 18 caracteres y el primer caracter de un identificador debe ser una letra. Se puede utilizar letras, caracteres y lineas de subrayado para el resto del identificador.

Cada nombre de la columna de una tabla dederá identificar únicamente una columna, pero se pueden duplicar nombres de columnas en una base de datos. Cuando se utilizan diversas tablas en aplicaciones, formas o reportes y el mismo nombre de la columna aparece en más de una tabla seguido de un punto y por último el nombre de la columna (nombre tabla - nombre columna) para así identificar la columna correcta.

Informix-SQL crea un directorio del sistema operativo para cada base de datos. El directorio contiene archivos para las tablas e indices en esa base de datos. El directorio y los archivos estan marcados por tres letras para la extensión.

- Database.dbs : Nombre del directorio de la base de datos.
- Tabla-Identificador.dat : Datos de una tabla.
- Tabla -Identificador.idx : Indice de una tabla.

Las formas, reportes y comandos de archivo asociados con la base de datos son almacenados en el directorio actual:

- formfile.per : Contiene especificaciones para formar una pantalla, este archivo se puede crear usando informix-SQL o un editor. Se debe compilar este archivo antes de que se use para una pantalla.

- formfile.frm : Contiene la especificación de la forma compilada. Informix-SQL usa estos archivos, no se debe nunca trabajar directamente con ellos.

- report.ace : Contiene especificaciones para un reporte. Estos archivos se crean por medio de informix-SQL o por un editor este archivo se debe compilar antes de correrlo.

- report.arc : Contiene especificaciones de reportes compilados. Informix-SQL usa este archivo nunca se deberá trabajar directamente con él.

- command.sql : Contiene una o más declaraciones de RDSQL, se puede crear dicho archivo por medio de informix-SQL, o por un editor.

Los cátalogos del sistema son los archivos que mantienen datos de tablas e información de índices, cada directorio de la base de datos contiene 18 archivos para el catálogo del sistema. El catálogo almacena la ruta de las tablas, columnas índices, ventanas, sinónimos y permisos en cada base de datos.

4.6.2 NATURAL.

Natural es uno de los lenguajes de cuarta generación más poderoso y que es usado en muchas instalaciones de cómputo por la gran cantidad de herramientas que posee para generar sistemas que administren bases de datos y técnicas de manejo de base de datos de tipo relacional. En conjunto con el soporte del diccionario de datos y técnicas de manejo de bases de datos es conocido como lenguaje no procedural y provee todas las funciones necesarias para el desarrollo de aplicaciones en un ambiente sencillo integrado, interactivo y de usuario amigable.

Natural con su completa integración del diccionario de datos, manejo de librería, lenguaje de programación, procesamiento de archivos de la base de datos, implementación de mapas, editores, interfase de comunicación de datos en forma de paquete (batch), creación de reportes y documentación en línea, uso de menús, permite el ciclo de vida de una aplicación desde el diseño hasta la producción y el mantenimiento.

Base de Datos.- Natural soporta aplicaciones en ambientes de bases de datos relacional tal como ADABAS y DL1/IMS, así como en ambientes de archivo estándar como VSAM.

Prototipos.- En este caso se realiza la definición de mapas prototipos (pantallas) vía el easy-to-user, definición de mapas orientados a usuarios finales y fáciles de probar.

Implementación de Mapas.- Las pantallas de salida son mostradas directamenta sobre la pantalla utilizando un mínimo de código de programación usando el editor de mapas inteligentes de natural llamado MAPPING.

Documentación en Línea.- Esta función tiene como objetivo la documentación en línea de la definición del problema. Inclusión de mapas de salida desde la librería de mapas y descripción de los campos desde el diccionario de datos de Natural.

Uso de Menus.- Con objeto de proporcionar al usuario las funciones con que cuenta el sistema. Estos menus se crean con la utilería de Natural llamada MAPPING.

El ambiente de programación tiene las siguientes características:

- Poca cantidad de código a dar mantenimiento.
- Código claramente estructurado.
- Uso de computadoras personales como estaciones de trabajo mostrando las funciones de estas y las facilidades de la conexión/Natural que habilita acceso a mainframes (macrocomputadores) y archivos de datos, así como el flujo de las funciónes del sistema operativo vía menus para que el usuario nunca tenga problemas con comandos del sistema MS-DOS.

Natural es un sistema completamente integrado cuando las siguientes funciones de procesamiento son adheridas.

- Predict.
- Adabas.
- Con-nect.
- Com-plete.
- Net-work.

PREDICT es un completo diccionario de datos activo que provee soporte de documentación para el ambiente de procesamiento de datos. Es usado también para guardar parámetros de seguridad para datos y programas, realiza las siguientes funciones:

- Creación y mantenimiento de datos, PREDICT contiene toda la definición de datos físicamente almacenada en la base de datos.

- Estas definiciones están disponibles para usuarios en la forma de visión de datos que representa campos seleccionados de uno o más archivos.

- Consistencia en la implementación y el diseno, los campos pueden ser definidos por PREDICT en la fase del diseno de una aplicación. Estos describen los datos y los programas a usarse son desplegados en la pantalla para verificar que la implementación del programa sea consistente con el diseno original.

- Autorización de acceso de datos y reglas de verificación durante la verificación del programa, el compilador NATURAL usa las definiciones de PREDICT para verificar que el programa compilado ha sido autorizado para usar el campo requerido en una pantalla de datos.

- Una o más reglas de verificación pueden ser definidas en una pantalla de datos. Las reglas de verificación son automáticamente aplicadas en el programa NATURAL cuando los campos son referidos durante ejecución de funciones como mapas de pantalla, actualización, mantenimiento y borrado. Esa verificación y reglas de procesamiento permiten la creación de estructuras de actualización compleja que pueden ser mantenidas automáticamente sin usar programación.

ADABAS es una moderna base de datos relacional que permite crear un fondo de almacenamiento central de datos, en donde, puede ser usado por diferentes aplicaciones y usuarios. También es un manejador de datos que relaciona los diferentes campos dinámicamente, basados en valores que establecen el ligado lógico entre varios componentes y que proveerán la flexibilidad para obtener todos los datos relacionados disponibles, además para manejo masivo de información.

Como un manejador de base de datos central, ADABAS coordina la operación de usuarios en línea y lote (batch) usando los mismos datos y provee facilidades para un reinicio automático después de cualquier interrupción sin ninguna intervención manual, también provee un sistema que usa la técnica de manejo del almacenamiento de datos llamadas tablas bidimensionales. ADABAS provee las siguientes funciones:

- Funciones de manipulación de datos relacionales vía operaciones SELECT, PROYECT1 y JOIN.

- Facilidades de procesamiento de datos orientado a campos que hacen los programas de aplicación completamente independientes del manejo de almacenamiento de datos físicos.

- Evaluación de complejos criterios de selección donde el calificativo conjunto de entidades son evaluados basados en campos principales sin tener que accesar cualquier otro registro.

ADABAS provee un mecanismo de control de transacción lógica que asegura consistencia y confianza en el procesamiento de transacciones.

El procesamiento de transacciones lógicas es totalmente independiente de los eventos físicos parecido a los límites de transacciones de la terminal. El aislamiento del usuario ocurre a nivel registro asegurando que todos los registros ocupados por un usuario han terminado la transacción.

CON-NECT es un sistema de automatización de oficina basado en un macrocomputador que puede ser usado para establecer una aplicación de automatización de oficina que está completamente integrada con otras aplicaciones de procesamiento de datos. Esta solución integrada no puede ser realizada usando aisladamente sistemas procesador de texto o igualar redes de área local. La principal ventaja de un sistema basado en un macrocomputador es la integración de datos y textos dentro de la misma base de datos y la disponibilidad de funciones de procesamiento de texto para todas las aplicaciones. Para la integración de las funciones de oficina dentro de la red general terminal, el costo para suplir a varios usuarios con el procesador de texto y funciones de comunicación es mínima. Incremento en la capacidad requerida de almacenamiento en disco como un resultado de procesamiento de texto extendido y recuperación de funciones pueden ser suplidas por la base de datos integrada por el bajo costo y casi con un potencial de expansión sin límite. Con-nect provee las siguientes funciones:

- Correo electrónico.
- Formateo de texto.
- Edición de texto.
- Clasificación de documentos.
- Recuperación de texto.
- Listas de recursos.

COMPLETE/COM-POSE es un sistema de comunicación de terminales que establece un ambiente que permite al usuario facilmente definir transacciones conversacionales sin que sea concerniente acerca del manejo de recursos a través de funciones de comunicación de terminales. COMPLETE, cuando se usa con NATURAL, y/o SUPER NATURAL, provee un completo ambiente interactivo conversacional que no requiere al usuario obtener un conocimiento especializado.

Como un monitor de teleproceso, la operación de multitareas de COMPLETE puede operar en un ambiente de multi-proceso donde múltiples sistemas de tareas son dinámicamente despachados a través de múltiples CPU'S.

La funcionalidad de COMPLETE no termina con procesamiento de terminal en línea avanzado. Esto también cubre un amplio rango de sistemas de utilerías requeridas para interactivamente controlar y monitorear operaciones de computadora en ambos ambientes de procesamiento en línea y en paquete (batch).

COMPLETE puede ser usado para controlar una red de terminales en ambientes OS/MVS y DOS/VSE usando VTAM o CTAM (COMPLETE tiene su propio método de acceso a la terminal).

COMPLETE/COMPOSE ofrece las siguiente funciones compatibles en ambientes OS/MVS y DOS/VSE.

- Procesamiento de transacción conversacional de alta ejecución.
- Sistema de almacenamiento (spooling) en línea.
- Extenso monitoreo de actividades en línea.
- Protección de almacenamiento.
- Pantalla de texto y programa fuente editor.
- Facilidades para reasignación de trabajos (job's).
- Control de actividades en modo paquete (batch).
- Reporte de utilización de espacio en disco.
- Prueba interactiva.
- Consola de operación operador remoto.

NET-WORK es una red distribuida de datos y procesamiento de transacciones, la distridución de datos y procesamiento de transacciones sobre una red interconectada de nodos de computadora representa el siguiente paso lógico en establecer una información común y recurso de comunicación dentro de grandes organizaciones. NETWORK es el conjunto que maneja todos los problemas técnicos para manejar cualquier dato en un ambiente distribuido disponible para cualquier usuario. Esto permite la transferencia de ejecución de un programa a el punto de máximo acceso de datos.

El sistema NETWORK consta de los siguientes componentes que pueden ser configurados de acuerdo a los requerimientos especificos del usuario:

- ADANET, operación de base de datos distribuida.
- ACCESS, procesamiento distribuido en línea.

- VTAM, comunicación con conexiones remotas.
- CTCS, comunicación de software directa canal a canal.
- VM/CMS comunicación de software.
- COMUNICACION DE HIPERCANAL, computadores, IBM y DEC/VAX.

4.6.3 LINC II.

LINC II significa Logic and Information Network Compiler, es un generador de sistemas de información, que permite a las organizaciones especificar sus necesidades del procesamiento de información en términos del negocio.

LINC II creará, implementará y dará mantenimiento a sistemas de información pequeños y muy grandes incluyendo:

- Especificación de la red para comunicación de datos.
- Proceso distribuido con estaciones de trabajo de LINC II.
- Base de Datos DMS II.
- Formatos de pantalla y de reportes.
- Lógica de procesamiento de transacciones en línea y en tiempo real.
- Sofisticadas estrategias de integridad y recuperación de datos.
- Capacidad para prototipos, asegurando el involucramiento del usuario y aprobación de todas la etapas del desarrollo.
- Manejo de otros idiomas además de inglés.

LINC II corre en computadoras UNISYS, y usa el software estándar de UNISYS para implementar los sistemas de información generados con LINC II. Los sistemas de información generados con LINC II tendrán un rendimiento, al menos, tan eficiente como los sistemas desarrollados con medios convencionales.

LINC II ofrece facilidades para el proceso distribuido. Hace que el desarrollo de complejos sistemas de información sean rápidos, baratos y manejables aumentando la productividad.

Proporciona las ventajas disponibles de la nueva tecnología. No se necesita esperar a que el sistema sea rediseñado, una simple generación con una nueva versión de LINC II abrio su sistema de información existente a todos los beneficios disponibles con la última tecnología de computadoras.

Es una nueva clase de generadores de sistemas de información, y es tal ves el único con las habilidades requeridas para desarrollarse por si mismo.

Ofrece los mejores medios disponibles para reducir el tiempo de desarrollo e implementación para permitir a una organización disfrutar de beneficios lo antes posible.

LINC II ha sido escrito en LINC II, proporcionar un ambiente de desarrollo en línea y en tiempo real para especificar los requerimientos de información. LINC II entonces generará un sistema de información en línea y en tiempo real para capturar los eventos del negocio conforme ocurran y para extraer información cuando se necesite.

Con un sistema de información de LINC II, la entrada de datos es colectada y almacenada en la base de datos y es analizada durante la salida para conocer los requerimientos de la organización. Cuando la organización quiere cambiar, sólo los reportes de salida necesitan ser cambiados.

Permite el desarrollo de un sistema en el modo de prototipo. Un prototipo es creado apartir de un sistema de información básica con las suficientes funciones y lógica para que el usuario verifique el diseño inicial, sin necesidad de proporcionar el sistema total, las revisiones interactivas de usuarios y diseñadores convierten al prototipo en un sistema completo, listo para la aceptación final como un sistema en producción. El resultado es satisfactorio porque el usuario participa desde el principio estrechamente en la definición del sistema, los beneficios incluyen un mínimo de capacitación hacia el usuario.

LINC II cuenta con las siguientes características :

- Checa la especificación para el sistema, la integridad y consistencia de datos.
- Crea los programas que accesan la base de datos.
- Crea el software para la red de terminales.
- Proporciona el ambiente para el software de estaciones de trabajo y la interfase donde sea requerida.
- Crea el software para el manejo de transacciones.

El resultado es todo el software requerido para producir una completa implementación de un sistema de información.

LINC II desarrollará un sistema de información como una evolución del primer prototipo del sistema completo.

Los sistemas generados con LINC II, cuando son comparados con sistemas equivalentes tienen un rendimiento tan bueno o mejor que los otros sistemas.

LINC II proporciona un simple pero completo ambiente de procesamiento en tiempo real para transacciones en línea o en procesos batch, además el software necesario para la completa operación de un sistema de información en línea y en tiempo real.

LINC II reconoce la importancia de la seguridad de la información y proporciona las herramientas necesaria para asegurar la integridad de los datos almacenados en la base de datos. Existen dos tipos de integridad para la información :

- Integridad de Procesamiento. Es requerida para permitir el uso de procesamiento concurrente, esto es, tener más de una transacción siendo procesada a un tiempo.
- Integridad de recuperación. Es la seguridad de que después de una falla de cualquier tipo, los datos contenidos en el sistema de información en LINC II serán reestablecidos a la condición que había inmediatamente antes de que ocurriera la falla.

La comunicación entre sistemas generados con LINC II es proporcionada por LINC II a través de un programa llamado LINC HUB el cual permite la comunicación con hasta otros cuarenta sistemas generados con LINC II.

También proporciona la comunicación de un sistema generado con LINC II con otro sistema en tres formas diferentes :

- LINC/OFFLINE.
- LINC/GLI.
- LINC/USER.

Los sistemas de información generados con LINC II son desarrollados usando el compilador interactivo LINC II. El compilador interactivo alamacena todos las especificaciones de LINC II en una base de datos, las esfecificaciones incluyen todas las opciones formatos de pantalla reportes y comandos de lógica usados para describir el sistema de información requerido. Este compilador interactivo está diseñado para soportar un ambiente de multiusuario y multiespecificación. Esto proporciona:

- Control central del desarrollo.
- Diccionario de datos para mantener la consistencia en la definición de los datos usados en reportes y pantallas.
- Muchos desarrolladores pueden trabajar en la especificación de un sistema concurrente.
- El control operacional de todo el código fuente del sistema es muy simple ya que proporciona utilerías de respaldo y recuperación.

CICLO COMPLETO DE LINC

0.- El operador recibe pantalla de bienvenida de su sistema o solicita otra pantalla distinta.

1.- Se ejecuta lógica prepantalla. Se utiliza para llenar valores establecidos (de default) en lugar de que tenga que hacerlo el operador.

2.- LINC manda la pantalla a la terminal.

3.- El operador teclea los datos y oprime transmit (XMIT).

4.- LINC recibe los datos.

5.- LINC edita los campos numéricos y valida condiciones propias de cada campo, como si se requiere información dentro de un campo, etc.

6.- Se ejecuta lógica pre-LINC. Se usa para procesar o crear datos antes que LINC verifique los datos con lecturas automáticas.

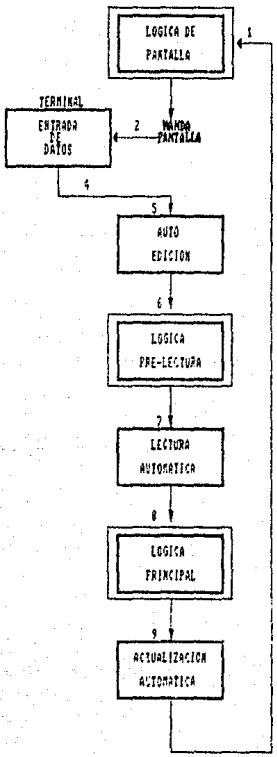
7.- LINC hace las lecturas automáticas para verificar la existencia de la información fija del sistema, como catálogos de clientes.

8.- Se ejecuta la lógica principal del usuario. Se usa para actualizar otras estructuras independientes de la que se est* accesando.

9.- LINC actualiza la base de datos si no encuentra condiciones de error.

A continuación se muestra una representación gráfica del ciclo completo de LINC:

CICLO COMPLETO DE LINC



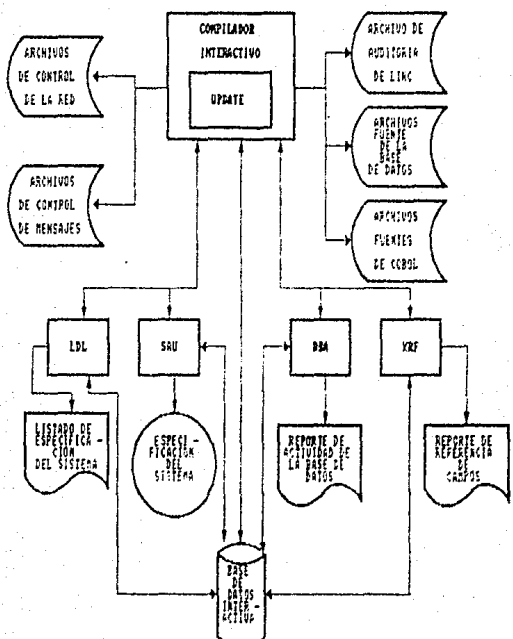
DICCIONARIO DE DATOS

El diccionario de datos provee una definición centralizada de campos de datos, incluyendo características de datos (longitud, tipo, etc.) y documentación descriptiva. Los campos de datos se pueden declarar como globales, disponibles a todas las especificaciones del sistema almacenadas en la base de datos interactiva de LINC, o locales a una especificación de un sistema en particular.

Los campos de datos definidos en el diccionario se pueden invocar por el nombre durante el pintado de la pantalla o del reporte. Todas las características definidas para el campo de datos automáticamente se aplicarán al momento de la invocación. Cualquier intento por modificar las características definidas en el diccionario será rechazado por LINC.

Cuando un cambio en la definición del campo de datos llegue a ser necesario, el diccionario de datos automáticamente cambia la especificación del sistema en cualquier lugar dónde se encuentre el campo de datos. La especificación completa se checa para encontrar cualquier lugar dónde el cambio no pudo ser efectuado (p.e. dónde la longitud del campo de datos resulta en un truncamiento de video en la pantalla). Se notifica al programador de cualquier problema potencial antes de llevar a cabo el cambio.

MEDIO AMBIENTE DE DESARROLLO DE LINC II



ESTRUCTURAS PARA EL ALMACENAMIENTO DE DATOS

COMPONENTE

- Describen la información estática y representan la razón de ser del sistema.

- El nombre de cada componente debe tener como máximo 5 caracteres y 2 como mínimo y deberá ser único.

- Existen 3 tipos de componentes:

+ Estándar: Es un conjunto de datos con un campo como llave natural de acceso. A esta llave se le llama "ORDINATE".

Cada acceso al Componente LINC trae a memoria un bloque de información (un registro).

+ TABLA: Es un conjunto de datos con un campo como llave natural de acceso (ORDINATE). Se utiliza para volúmenes pequeños de información.

La diferencia con el Estándar, es que al primer acceso, LINC carga a memoria todo el data-set del Componente

+ MEMO: Es un conjunto de datos sin llave natural de acceso. Por tanto, en este Componente se permiten registros duplicados, se utiliza principalmente con histórico.

EVENTO

- Describen las transacciones que afectan a los componentes.

- El nombre debe ser único y constar de 5 caracteres como máximo.

- No tiene llave natural de acceso.

- Todos los campos de los Eventos de un sistema son guardados en un mismo Data-set, por lo que es recomendable dar el mismo nombre a los campos de los eventos mientras sea posible.

PROFILES

- Diferentes vistas a la base de datos.

- El nombre del profile puede ser hasta de 9 caracteres.

- Puede manejar hasta 1 componente o hasta 20 eventos.

- Puede tener hasta 20 llaves (ordinates).

- Cada llave puede ser ascendente (default) o descendente.

- Puede seleccionar registros para inclusión por condiciones de DO WHEN como lo muestra el siguiente ejemplo:

```
PROFILE BYRATE

   COMP.NAME = ROOM
   OR; RATE DSC;
   OR; ROM-NO
   DO.WHEN ; ROOM.MAINT NOT = (D)

PROFILE PRODBAL
   EVENT.NAME = SALE
   EVENT.NAME = STKIN
   OR; PROD - NO

PROFILE WATCHOUT
   EVENT.NAME = BUY
   OR; CUST-NO
   DW;AMT > (10000) AND
   DW;DAYS < (30)
```

4.7 PROGRAMA DE TRABAJO.

1.- Preparación para el desarrollo.

2.- Revisar la estructura de los archivos y base de datos.

3.- Conducir la programación.

4.- Consideración del ambiente en el cual el sistema será desarrollado.

5.- Preparación de la documentación.

6.- Ejecución de pruebas internas.

7.- Conducción del entrenamiento.

8.- Conducción de pruebas de aceptación.

9.- Preparación del plan de imprevistos.

10.- Instalación del software de aplicación.

11.- Conducir la instalación del sistema.

12.- Lleva a cabo la operación inicial del sistema.

TAREA NUMERO UNO.

Preparación Para El Desarrollo.

Los objetivos de esta tarea son:

- Establecer un ambiente que conduzca a buenas relaciones entre el centro de cómputo y el equipo de desarrollo.
- Contar con un ambiente en el cual el equipo de desarrollo, mantenga la productividad y además sea rápida.
- Asegurar que el equipo de desarrollo entiende como será implementado el sistema.

Es muy importante el contacto entre el gerente del proyecto y el personal apropiado del centro de cómputo. Al administrador del centro de cómputo se le dará un panorama de la metodología con la cual el sistema será desarrollado e implementado. Los recuros típicos a considerar son :

- Terminales.
- Horas de operación (disponibilidad del sistema).
- Espacio en disco.
- Cintas.

El gerente del proyecto debe también establecer una relación con el personal del centro de datos tales como, programadores, supervisores de operación y el administrador de la base de datos.

Es necesario establecer el ambiente de programación y pruebas. El propósito de la organización es resolver detalles asociados con el trabajo en el nuevo ambiente antes de la llegada del equipo completo de desarrollo. Algunas actividades típicas son:

- Obtener passwords.
- Distribuir librerías y datos de prueba.
- Obtener los compiladores.
- Establecer procedimientos de respaldo y recuperación.

La última actividad de la lista anterior se debe desarrollar para:

- Archivos fuente y objeto.
- Archivos de prueba de datos.
- Archivos de documentación.
- Archivos de lenguaje de control.

TAREA NUMERO DOS

Revisar Las Estructuras De Los Archivos y Base De Datos.

El objetivo de esta tarea es tomar la estructura de los archivos y la base de datos de las etapas anteriores, ya que se tiene la certeza que los datos son confiables y actuales, para así asegurar que éstos no ocasionarán problemas al sistema.

TAREA NUMERO TRES

Conducir La Programación.

El objetivo de ésta tarea es producir el código fuente del programa de aplicación, lenguajes de control o de trabajo y la interfase para la comunicación de datos tan rápida y segura como sea poible, a pesar de la herramienta de programación que sea usada. El código debe:

- Ser auto-documentable para minimizar el mantenimiento y las modificaciones.
- Debe de ser apropiado a los estándares.

La eficiencia significa la velocidad con la cual el código es producido también como las técnicas utilizadas para minimizar el esfuerzo. Algunas técnicas efectivas para procucción de código son:

- Uso de un paquete generador de código.
- Uso de subrutinas comunes.
- Modificación de una copia de un programa, existente para formar un nuevo programa.
- Codificación de secciones estándares de código esqueleto y completar el código único para cada programa.
- Uso de un diccionario de datos automatizado.

Las actividades de programación y pruebas generalmente se traslapan y desde la prespectiva de administración de proyectos, son las tareas más complejas debido al número de profesionales involucrados.

Algunos de los resultados de manejo que enfrenta el gerente del proyecto al efectuar esta tarea incluyen:

- SECUENCIA DE LAS ACTIVIDADES DE PROGRAMACION.
  La programación debe de ser ejecutada con una secuencia para que facilite la agrupación de las pruebas del sistema y subsistemas.

- MODULOS Y RUTINAS COMUNES.
  Primero programar todos los módulos y rutinas comunes. Conducir caminos estructurados de todas las rutinas en cuanto se complete su programación. Si es posible tener programas y grupo de datos de prueba disponibles.

- ASIGNACION DE TRABAJO.
  Antes de desarrollar asignaciones especificas para el equipo de programación, el gerente del proyecto debe determinar la complejidad de cada subsistema y si se encuentra en la ruta crítica. Los programadores más experimentados deben de ser asignados responsablemente para los programas más complejos, librerías comunes o rutinas que estan en la ruta crítica.

- ENTRENAMIENTO ANTES DE CODIFICAR.
  Los programadores deben tener un entrenamiento antes de codificar el programa o rutina. Se requiere un alto nivel de interacción entre el diseñador de software y el programador (asumiendo que no son la misma persona).

El gerente del proyecto tendrá la responsabilidad de monitorear y controlar las actividades de programación y pruebas, las principales son:

- Completar las tareas de programación y pruebas a tiempo.

- Completar las asignaciones de trabajo individuales a tiempo.

- Monitorear la calidad de los programas codificados.

Para estar completo un programa debe de estar codificado, compilado y probado.

Las pruebas examinarán si un programa es capaz o no de comunicarse con otro programa, ya sea proporcionando o recibiendo datos de él, errores de programación o especificaciones pueden ser encontrados en etapas de pruebas posteriores, un programa no trabaja hasta que procesa datos exitosamente. El primer programa completado por un programador debe tener las características de la programación estructurada y se debe asegurar que:

- Se ha programado de acuerdo a los estándares.
- Se han usado módulos y rutinas comunes.
- El código no se desvia de las especificaciones.

TAREA NUMERO CUATRO.

Consideración Del Ambiente En El Cual El Sistema Será Desarrollado.

En esta tarea es necesario que todo lo concerniente al hardware esté completamente instalado, se considera el lugar en el cual se encuentra el equipo eléctrico, sistema de seguridad, líneas de comunicaciones, todo lo anterior se sujeta al tamaño y tipo de hardware a instalar.

TAREA NUMERO CINCO.

Preparación De La Documentación.

El objetivo de esta tarea, es producir una documentación de alta calidad para el entrenamiento del usuario y operación del nuevo sistema. Dederá ser diseñada una estrategia de operación a diversos niveles para el usuario, el gerente del proyecto, el técnico, el operador y el personal experimentado o sin experiencia.

Un sistema bien diseñado no puede ser totalmente usado o apreciado por el usuario a menos que la documentación sea de similar calidad. La documentación debe ser diseñada de la forma en que será diseñado el sistema. Hay cuatro elementos principales de la documentación.

- Manual de documentación del sistema.

- Manual de documentación de programas.

- Manual de documentación de operación.

- Manual de documentación del usuario.

Cada manual debe de ser el producto de una sola persona, o un grupo pequeño de personas, por tres razones:

1.- Ya que se puede proporcionar una unidad conceptual para el documento. Así el manual del usuario puede llegar a tener un estilo de escritura ordenado y consistente.

2.- Para que el manual sea considerado como un buen trabajo es necesario que éste sea individual, ya que si es hecho por un grupo y en conjunción con otras tareas llega a ser arduo.

3.- El manual proporciona un ambiente en el cual, todo el equipo es productivo ya que se concentran ideas muy buenas concernientes a la programación y a las pruebas.

Una buena documentación incluye lo siguiente:

- Propósitos del sistema.

- Significado para el usuario.

- Bosquejo del diseño del sistema.

- Sugerencias para mejor uso.

- Un procedimiento paso a paso, para guiar al usuario a través de la aplicación.

TAREA NUMERO SEIS.

Ejecución De Pruebas Internas.

Esta tarea tiene como objetivo asegurar que el sistema se ejecuta u opera satisfactoriamente antes de ser liberado al usuario, es por esto que se somete a pruebas de aceptación. Se realizan dichas pruebas para encontrar posibles errores, evaluar productividad y todo lo anterior proporciona confianza en el software.

Deberá de llevarse acabo la ejecución del plan de pruebas a nivel programa, subsistemas, interfases y sistema.

Las siguientes consideraciones deben haberse ya determinado para el plan de pruebas:

- ESTRATEGIA DE PRUEBAS. De arriba-abajo, de abajo-arriba, como caja negra.
- Preparar los datos de pruebas.
- Alcance de la prueba.
- Papel de los miembros del proyecto.
- Papel del usuario.
- Criterio de aceptación de la prueba.

El usuario desempeña un papel activo en esta tarea, particularmente en la preparación de los datos de prueba, así se facilita el entrenamiento y la ejecución de las pruebas de aceptación de calidad. Esta tarea podrá conjuntarse con las tareas, tres (conducir la programación), cinco (preparar la documentación) y siete (conducir el entrenamiento).

Para completar la función de las pruebas, un miembro del equipo del proyecto de tiempo completo seleccionado debe:

a) Preparar los datos de prueba basados en las especificaciones del programa. Los datos de prueba deben de incluir las transacciones que serán procesadas por los programas, también el contenido de los archivos y tablas que soportan el sistema.

b) Preparar los programas y/o utilerías para vaciar el contenido de los archivos antes y después de la ejecución de las pruebas.

c) Preparar los procedimientos que ejecutan las pruebas.

d) El grupo que lleva a cabo las pruebas, no es responsable para determinar el éxito o fracaso de una prueba.

Procedimientos similares son requeridos para apoyar las pruebas en los niveles de sistemas y subsistemas, sin embargo, en esos niveles, las transacciones de prueba o especificaciones para el contenido de los archivos, deben de ser preparados por especialistas en conjunción con el usuario.

Existen dos métodos que proveen el desarollo de pruebas; pruebas de caja blanca y pruebas de caja negra.

Las pruebas de caja blanca estan enfocadas a la estructura de control del programa, la prueba debe conducir a afirmar que todas las declaraciones del programa y sus condiciones lógicas se ejecutan de acuerdo a lo especificado. La prueba de ruta base es una técnica de caja blanca, hace uso de las gráficas de programas (o matrices gráficas) para proveer un grupo de rutas linealmente independientes. Otra prueba de caja blanca es la de loop esta prueba descubre la inicialización, indexación o incremento de errores que pueden ocurrir en un loop.

Las pruebas de caja negra son disenadas para validar requerimientos funcionales sin considerar el tarbajo interno del programa la prueba debe conducir a demostrar que cada función es operacional, se debe tener en cuenta lo siguiente: la definición de la salida esperada o del resultado a obtener; condiciones de entradas válidas e inválidas; confirmar que el sistema y el programa hace lo que se espera y no pensar que hay cosas inesperadas. La gráfica causa efecto es una técnica de caja negra que valida grupos complejos de acciones y condiciones. La prueba de validación de datos otra técnica, permite asegurar que datos interactivos son procesados apropiadamente.

El ambiente de pruebas, debe de ser usado por un grupo de programación, para probar futuros cambios antes de hacerlos en producción.

TAREA NUMERO SIETE.

Conducción Del Entrenamiento.

El objetivo de esta tarea es, educar al usuario en la operación, uso y mantenimiento del nuevo sistema. Hay tres áreas principales de entrenamiento.

- Entrenamiento para la gerencia.
- Entrenamiento para el departamento de usuarios.
- Entrenamiento para procesamiento de datos.

Una buena documentación facilita un buen entrenamiento, pero sola no es suficiente. Como en la documentación, una estrategia de entrenamiento también debe tomar en cuenta los diferentes tipos y niveles de usuarios.

El entrenamiento del usuario, para ser benéfico, debe de estar estrictamente limitado a tópicos que son verdaderamente significativos para la operación y uso del sistema implementado. Los usuarios generalmente no se interesan en una discusión sobre la metodología utilizada, ellos quieren invertir el mínimo tiempo necesario en aprender a operar y usar el nuevo sistema.

Cada sesión de entrenamiento debe contener:

- Propósitos del sistema.
- Significado para el usuario.
- Un procedimiento paso a paso para guiar al usuario a través de la sesión.

Los usuarios deben de ser estimulados para preguntar cualquier duda que tengan, así como los analistas deben de estar perfectamente capacitados para resolverlas.

Esta tarea generalmente se traslapa con las actividades de programación y pruebas.

El entrenamiento es una prueba efectiva del sistema y de la documentación, éstos deben ser totalmente integrados entre sí.

Subtareas:

a) Revisar el plan de entrenamiento.

b) Preparar y finalizar el material del entrenamiento.

c) Ejecutar el plan de entrenamiento incluyendo:

- Clases formales.
- Material de entrenamiento de auto-estudio.
- Seminario.

d) Obtener y documentar la realización del usuario a pesar de la efectividad del entrenamiento.

TAREA NUMERO OCHO.

Conducción De Pruebas De Aceptación.

Los objetivos de esta tarea son:

- Asegurar que el sistema ha sido desarrollado de acuerdo a las especificaciones.

- Asegurar que el sistema está operando a la satisfacción del usuario.

- Documentar la aceptación del sistema.

Las siguientes consideraciones del plan debieron haber sido ya determinadas en el plan de:

- Procedimientos de pruebas.
- Preparación de los datos de prueba.
- Papel de los miembros del proyecto.
- Papel del usuario.
- Criterio para una prueba de aceptación éxitosa.

Esta tarea proporciona la integración del entrenamiento del usuario y la prueba del sistema. Los datos de prueba, así como sus resultados deben prepararse en conjunción con el usuario, antes de la ejecución de la prueba. Este esfuerzo, proporcionará un entrenamiento completo en todas las funciones que ejecuta el sistema.

En la ejecución de las pruebas de aceptación, el usuario debe ejecutar las funciones clave que serán llevadas a cabo cuando el sistema sea puesto en producción. Esto debe ser ejecutado y guiado por la documentación contenida en los manuales del sistema.

Los factores que afectan la calidad del software se categorizan en dos grupos: factores que se pueden medir indirectamenete y los que se pueden medir directamente. Los factores que afectan la calidad del software enfocan tres aspectos importantes de un producto de software; características operacionales, capacidad para sufrir cambios y adaptabilidad a nuevos ambientes. Los factores que afectan la calidad del software son los siguientes:

- Correcciones.
- Confiabilidad.
- Eficiencia.
- Integridad.
- Usable.
- Mantenibilidad.
- Flexibilidad.
- Comprobabilidad.
- Portabilidad.
- Reusabilidad.
- interoperatividad.

Un grupo de métricas son definidas y usadas para cada uno de los factores, muchas de las métricas definidas por Mc Call pueden ser medidas subjetivamente, las métricas son usadas para clasificar atributos específicos de software con una escala de 0 (menor) a 10 (mayor), las siguientes métricas son usadas en el esquema de clasificación:

- Auditabilidad.
- Exactitud.
- Comunicación común.
- Perfección.
- Brevedad.
- Consistencia.
- Datos comúnes.
- Tolerancia de errores.
- Expandible.
- Independencia de hardware.
- Instrumentación.
- Modularidad.
- Operatividad.
- Autodocumentable.
- Seguridad.
- Simplicidad.
- Independencia de software.

La complicación de los factores es la precisión de la relación entre la variable que es medida y la calidad del software.

Mc Cabe define una técnica que es usada para calcular la métrica de complejidad de un programa V(G), se determinan el número de regiones de una gráfica plana esta métrica provee un indicador cuantitativo.

A diferencia de la tarea 6 (ejecutar pruebas internas), lo más importante de las pruebas de aceptación, desde el punto de vista del equipo del proyecto, es demostrar que el sistema trabaja, y no necesariamente en la identificación de errores. Desde el punto de vista del usuario las pruebas de aceptación son el procedimiento final de control de calidad, que puede ser hecho para determinar si el sistema funciona como se esperaba o no.

Para llevar a cabo esta tarea se debe contemplar lo siguiente:

- Conducir las pruebas del usuario del sistema.

- Obtener la verificación del usuario de los resultados de las pruebas.

- Obtener la verificación del usuario de los nuevos procedimientos.

- Obtener la aceptación del usuario y su firma.

- Documentar los resultados de las pruebas.

- Preparar un reporte escrito de la aceptación del sistema.

- Conducir la representación oral.

TAREA NUMERO NUEVE.

Preparación Del Plan De Imprevistos
En El Sistema.

La planeación está dirigida hacia la posibilidad de que algún evento imprevisto pudiera, por un periodo de tiempo extenso, interrumpir la entrega del sistema. La naturaleza de tales eventos es:

- Ocurren súbitamente y sin advertencia.

- Son imprevisibles y tienen efectos que pueden ser catastróficos.

- Pueden ocurrir en el curso normal de operación.

El resultado de un desastre es tal, que el centro de datos, podría estar inoperable por un periodo largo de tiempo. Las aplicaciones que soportan las funciones críticas, no pueden ser procesadas normalmente. Pueden ser requeridas capacidades adicionales de procesamiento debido a demandas "poco" normales. Dos objetivos muy importantes que debe contener el plan son:

- Reducir la probabilidad de ocurrencia de un desastre.

- Minimizar el impacto de un desastre, antes de que ocurra.

El programa de trabajo se debe diseñar para:

- Determinar la vulnerabilidad del usuario en interrupciones de servicio significativas y definir medidas preventivas que pueden ser tomadas para minimizar la probabilidad e impacto de tales interrupciones.

- Identificar y analizar la existencia de una interrupción larga en el procesamiento de datos, así como alternativas de procesamiento.

- Determinar las necesidades de recuperación y requerimientos de recursos inmediatos, intermedios y a largo plazo.

- Identificar las alternativas y seleccionar la aproximación más viable (costo-eficaz) para proporcionar capacidades de respaldo en procesamiento de datos y la restauración oportuna del servicio.

TAREA NUMERO DIEZ.

Instalación Del Software De Aplicación.

El objetivo de esta tarea es preparar al sistema de 4a. generación, ya aceptado, para la producción. Al concluir esta tarea, toda evidencia de pruebas del medio ambiente del sistema deberá ser removida con excepción de aquella que sirva como una utilería permanente para pruebas del sistema. Para llevar a cabo esta tarea se debe considerar:

- Salvar el medio ambiente de pruebas (respaldar archivos de prueba, librerías, datos, etc.)

- Desactivar elementos de monitoreo y pruebas (rastreo, desplegados. etc.)

- Recopilar programas para la producción de librerías.

- Modificar las pruebas para los requerimientos de producción (incrementar áreas en disco, cambiar nombres de archivos, etc.)

- Preparar los módulos de soporte del sistema (base de datos, TCL; terminal control tables, etc.)

- Establecer códigos de producción de usuarios y claves de acceso.

- Instalar programas y proporcionar documentación de operación y guías del usuario.

- Establecer un procedimiento para reporte de problemas en los programas.

TAREA NUMERO ONCE.

Conducir La Instalación Del Sistema.

El objetivo de esta tarea es completar la ejecución y asegurar que todo esté en su lugar para la operación inicial del sistema. La instalación de hardware, software y la conversión de archivos debe haber terminado o puede estar ocurriendo conjuntamente con el inicio de esta tarea. La instalación del sistema es un esfuerzo manual que involucra operaciones con el personal y los usuarios directos del sistema. Deben ser destruidas nuevas formas, manuales, procedimientos y las versiones anteriores removidas. Típicamente se escoge un fin de semana para esta actividad.

Debe hacerse un chequeo final con operaciones reales, revisando:

- Relaciones del flujo de tareas.

- Calendarios y prioridades.

- Tareas.

- Procedimientos de la red.

- Archivos catalogados.

- Respaldos y recuperaciones.

Se debe realizar una revisión final con producción y control de datos, revisando:

- Balance de procedimientos.

- Instrucciones de ingreso de datos.

- Procedimientos de peticiones de ejecución.

- Reglas de retención.

- Opciones de parámetros de ejecución.

Existen cuatro técnicas básicas de remplazo:

- Paralela Limitada.

- Implementación en Fase.

- Operación Paralela.

- Directa.

La operación Paralela a Escala Completa, es difícil para propósitos de probar el nuevo sistema, porque generalmente el sistema existente y el nuevo sistema no producirán resultados comparables. La operación Paralela a Escala Completa es también muy costosa. Una técnica que facilita la recopiliación de los viejos y los nuevos sistemas, y reduce el costo, es la operación en Paralelo Limitada. En este modo de operación los nuevos sistemas y los ya existentes, son ejecutados en paralelo, sin embargo, el sistema nuevo procesa sólo un conjunto seleccionado de los datos, esta operación en paralelo reduce el nivel de esfuerzos requeridos para ingresar dos conjuntos de transacciones y recopilar los resultados obtenidos. Esto también reduce significativamente el costo de operación en modo paralelo. La operación Paralelo Limitada producirá los mismos resultados que la operación a Escala Completa, si el subconjuto de datos que se proporciona es representativo de todos los datos a ser procesados en el modo de producción.

La Implementacion por Fases es el proceso de instalar porciones del sistema en períodos de tiempo, antes de instalar completamente el sistema de una sola vez. Se utiliza también para describir el proceso de implementar completamente el sistema en sitios seleccionados antes de hacerlo en todos los lugares a un mismo tiempo. La Implementación por Fases, puede o no ser conducida en conjunción con operaciones limitadas en paralelo.

Este tipo de operacion es recomendada, cuando la tarea de instalar completamente el nuevo sistema de una sola vez no es práctico por dar un número limitado de recursos.

La operación en Paralelo a Escala Completa incluye el procesamiento de todas las transacciones en ambos sistemas y la recopilación de resultados diarios, semanales y mensuales, hasta que todas las partes sean convincentes de que el nuevo sistema trabaja. La operación en Paralelo a Escala Completa generalmente requiere empleados eventuales para operar el sistema viejo, mientras los empleados permanentes son asignados al nuevo sistema y da resultado la recopilación de procesos. La primera ventaja de la operación en paralelo es que el sistema viejo puede ser usado sin falla en el nuevo sistema sin perder tiempo, la principal desventaja es el costo asociado a éste método.

Aunque existe un riesgo con el reemplazo completo al nuevo sistema, éste puede ser minimizado a través de pruebas adecuadas y asegurar que el viejo sistema esté en disposición de usarse si es necesario.

Las técnicas de reemplazo varían de situación en situación. En la decisión de las técnicas que se usarán, se consideran los siguienetes factores:

- Naturaleza de los sistemas de aplicación.
- Comparación de las salidas del sistema.
- Recursos disponibles (personal y harware).
- Riesgos.
- Costos.

La técnica, reemplazo completo del sistema, es la más común. Como una prevensión es aconsejable tener un plan para regresar al sistema viejo en caso de problemas insuperables con el nuevo sistema, si es posible. También, estas técnicas deben ser precedidas por pruebas del sistema

TAREA NUMERO DOCE.

Llevar A Cabo La Operación Inicial Del Sistema.

Los objetivos de esta tarea son:

- Asegurar que los problemas del sistema sean resueltos rápidamente y profesionalmente.
- Asegurar que el usuario opera el sistema correctamente.

Las técnicas de asistencia, son requeridas para alcanzar estos objetivos. La magnitud y duración de la asistencia, variará por el tamano, complejidad del sistema, y las necesidades del usuario.

Para llevar a cabo esta tarea se debe considerar lo siguiente:

a. Proporcionar una guia en la operación del sistema.

b. Proporcionar el soporte para definición/corrección de errores.

c. Proporcionar entrenamiento de trabajo al usuario.

d. Realizar afinaciones al sistema.

III.5 POSTIMPLEMENTACION.

INTRODUCCION.

En un ambiente de 4a. Generación, la fase de Postimplementación contiene una descripción del programa de trabajo y liberación; se debe revisar la calidad del sistema, puede ser realizada por un grupo externo (inclusive gente de desarrollo pero ajena al proyecto) de preferencia, los puntos que se revisarán son: Funcionalidad, Eficiencia y Mantenibilidad.

Después de que el sistema ha sido implementado, es importante que los integrantes del equipo de desarrollo estén dispuestos a proporcionar asistencia a solución de problemas y generalmente a usuarios del nuevo sistema (incluyendo a usuarios de procesamiento de datos). Generalmente, este periodo pasa por uno o dos ciclos operacionales del nuevo sistema.

Esta fase presenta una oportunidad para revisar e incrementar los beneficios y características del sistema actual y revisar el cambio de requerimientos observados en la fase de implementación y para identificar servicios adicionales de gran beneficio al usuario.

5.1 PROGRAMA DE TRABAJO.

La fase de soporte posterior a la implementación consta de cinco tareas:

1.- Supervisión de la operación del sistema.

2.- Supervisión de las prácticas del medio ambiente del usuario.

3.- Supervisión de la documentación.

4.- Preparación del reporte de postimplementación.

5.- Seguimiento del usuario.

Las tareas anteriores pueden ser consideradas opcionales dependiendo de los requerimientos del usuario y recursos técnicos.

Las actividades que incluyen estas tareas pueden ser vistas como propósitos o actividades de desarrollo a futuro.

La implementación completa presenta una excelente oportunidad para recibir servicios adicionales de un usuario satisfecho.

TAREA NUMERO UNO.

Supervisión De La Operación Del Sistema.

Los objetivos de esta tarea son:

- Evaluar ejecución del sistema contra la ejecución de los objetivos.
- Identificar áreas de mejoramiento.

Una vez que el nuevo sistema esta operando, se pueden o no presentar varios cuellos de botella. Durante la Postimplemetación todos los programas y job's (ya sea en Línea o en Batch) podrán ser revisados desde una perspectiva de ejecución. Un ejemplo potencial puede ocurrir cuando un trabajo en Batch ejecutó un gran número de accesos a la Base de Datos y seriamente degradó el tiempo de respuesta en línea. La solución de este problema pudo incluir el ajuste de los parámetros de la Base de Datos, restructurar el programa en Batch o bien, la capacidad o prioridad en ese momento era bastante mala.

Para poder llevar a cabo esta tarea, se debe tener presente lo siguiente:

a) Revisar si los objetivos se cumplieron.

b) Revisar si el sistema ejecuta lo que el usuario necesita; llevar un balance de los resultados del sistema.

c) Pruebas de software para el mejoramiento del sistema.

d) Entrevistas a usuarios y a operadores concerniente a la ejecución y satisfacción del sistema.

e) Certificación de la calidad del sistema (Quality assurance).

Es importante contar con una sección de documentación que contenga:

1.- Notas de entrevistas y documentos de soporte.

2.- Análisis y conclusiones de la ejecución del sistema.

3.- Ejecución del software, salida de los reportes.

TAREA NUMERO DOS.

Supervisión Del Medio Ambiente Del Usuario.

Los objetivos para esta tarea son:

- Determinar si el sistema está siendo usado efectivamente.
- Determinar si hay prácticas organizacionales o de procedimientos que impidan el uso efectivo y eficiente del sistema.
- Determinar si hay oportunidad para improvisar prácticas organizacionales y de procedimientos.
- Dificultad en distribución oportuna de reportes.

A menudo problemas periféricos a el sistema son descubiertos, por ejemplo el tiempo de un ciclo (vida del software) de usuario, puede afectar la eficiencia y la efectividad del sistema.

Para llevar a cabo esta tarea se debe considerar:

a) Evaluación de procedimientos y colección de datos.

b) Evaluación de procedimientos en distribución de salidas.

c) Ciclo de análisis en empresas.

d) Análisis de formas y flujo de documentos.

e) Análisis organizacional de Responsabilidades.

f) Pruebas estructuradas del sistema.

La sección de documentos de trabajo debe contener:

- Entrevistas, notas y documentos de soporte.
- Análisis y conclusiones de prácticas organizacionales y procedimientos.

TAREA NUMERO TRES.

Supervisión De La Documentación.

El objetivo de esta tarea es, determinar si la documentación es usada, exacta y adecuada. Existen tres niveles en las herramientas de documentación:

1.- Sistemas.
2.- Usuarios.
3.- Operadores

Para satisfacer el objetivo de esta tarea tres preguntas pueden ser contestadas para cada tipo de documentación, éstas son:

- ¿Hace usted uso de la documentación? si no, ¿por qué? (note que esta contestación negativa puede ser señal de que un sistema es muy fácil para usar).
- ¿Cuándo usted usa la documentación, es exacta?
- ¿La documentación contesta todas sus pregutas acerca del sistema? Todas las respuestas negativas podrán ser documentadas con ejemplos específicos.

Para tener éxito en esta tarea, se debe de evaluar la siguiente documentación:

a) Documentación del sistema.

b) Documentación de usuarios.

c) Documentación de operadores.

La sección de documentos de trabajo debe contener:

- Entrevistas, notas y documentos de reporte.

- Análisis y conclusiones de la documentación.

TAREA NUMERO CUATRO.

Preparación Del Reporte De La Postimplementación .

La revisión de la postimplementación puede conducir totalmente al éxito del sistema y al estar bien documentada pueden sugerir cambios con el mínimo esfuerzo. Se pueden realizar entrevistas con usuarios, operadores y personal de soporte del sistema. Se puede hacer una comparación de los resultados esperados en la fase de definición de requerimientos y de los resultados obtenidos actualmente, ocurrencias no usuales y desviaciones de lineamientos de diseno podrán ser anotadas.

Los equipos de soporte en producción de sistemas podrán entrevistarse para determinar los tipos de excepciones encontradas que sugieren mejoras para evaluar la documentación. Datos estadísticos en cuanto a costos y beneficios podrán darnos una idea en cuanto a gastos de personal involucrado en el proyecto.

Una vez revisado el proyecto se evaluará la estructura del programa de trabajo, estimando lineamientos, estructuras de archivos, pruebas etc.; todo con el fin de evitar futuros problemas que se presenten en el desarrollo de sistemas.

El objetivo de esta tarea, es preparar un reporte de salida conciso y comprensible que nos muestre:

- Los resultados obtenidos en cada área con respecto a:

    * Entrenamiento a usuarios.
    * Capacidad de planeación.
    * Servicios de red y procesamiento.

- Recomendaciones (propósitos) en cada área.

- El plan de acción (si es necesario).

El reporte podrá convencer al usuario de que el proyecto terminó y las recomendaciones surgirán en ese momento si el usuario no es capaz de manejar dicho sistema. La organización del reporte seguirá la secuencia del programa de trabajo. Para llevar a cabo esta tarea se debe considerar:

a) Revisión de problemas y objetivos en el sistema.

b) Preparar y conducir el reporte de postimplementación.

c) Verificación de las funciones del sistema.

d) Revisión del reporte con el usuario.

e) Presentación oral con el usuario.

f) Preparación final del reporte escrito.

La sección de documentación debe contener:

- Presentación de reportes.

- Reporte escrito.

TAREA NUMERO CINCO.

Seguimiento Del Usuario .

El objetivo es buscar empresas en servicio de software adicionales. A menudo existe una tendencia para buscar nuevos usuarios nuestro mejor prospecto es un usuario, a quien nosotros hemos probado nuestras habilidades profesionales.

Después de un largo y sucesivo tiempo dedicado a la elaboración de un proyecto, las relaciones afectivas de trabajo que han sido establecidas entre usuarios y el departamento de sistemas, han desarrollado un buen entrenamiento de las necesidades organizacionales del usuario y es posible hacer recomendaciones constructivas en cuanto a futuros requerimientos de sistemas. El usuario entiende los beneficios del camino profesional y nos permitirá presentar propósitos para desarrollo de Software y/o emprender estudios de factibilidad para nuevas aplicaciones, así como un plazo para la instalación de sistemas.

Para llevar a cabo esta tarea es importante lo siguiente:

a) Revisión en el avance del usuario.

b) Verificación del desarrollo de Software.

c) Revisión de futuros requerimientos de Hardware.

d) Preparación en servicios de Software.

e) Presentación en servicios a usuarios.

La documentación necesaria es:

- La presentación del Software (manuales) para usuarios.

IV.- ADMINISTRACION DE PROYECTOS EN UN AMBIENTE DE CUARTA GENERACION

IV. ADMINISTRACION DE PROYECTOS EN UN AMBIENTE DE CUARTA GENERACION.

INTRODUCCION

Una mala planeación es la principal causa de los retrasos en un proyecto de programación, del incremento en costos, poca calidad en los productos y altos costos de mantenimiento. Para evitar estas complicaciones se requiere de una planeación cuidadosa.

Las actividades gerenciales son tan importantes como las técnicas, si se desea un buen control técnico para el desarrollo de un producto en programación, se requiere de un excelente control administrativo. Los recursos y el ambiente en que suceden las actividades técnicas, la elaboración del plan de trabajo, el desarrollo de estrategias, la contratación y capacitación del personal, son; responsabilidades de la gerencia, así como asegurar que los productos se entregen a tiempo y dentro del presupuesto estimado, y que exhiban la funcionalidad y calidad que el usuario requiere.

Las actividades de administración en un proyecto comprenden: los métodos para organizarlo y seguirle el curso, estimación de costos, definición de logros, determinación de avances en el proyecto, y ajustes al calendario.

IV.1 ESTIMACION DE LOS REQUIMIENTOS EN TIEMPO.

Los proyectos de programación deben considerar la naturaleza crítica de algunas actividades, por lo que se debe contar con una adecuada planeación del tiempo de desarrollo, ya que esto depende en gran medida, de su éxito o fracaso. Los proyectos que se desarrollan a tiempo cuentan con características comunes:

1.- Una estimación cuidadosamente formulada de los requerimientos en tiempo.

2.- La posibilidad de que la gerencia verifique el progreso.

3.- Un instrumento para la comparación del desempeno real respecto a lo planeado.

4.- Información suficiente para atender los problemas cuando se presentan.

La formulación de las estimaciones del tiempo que se requiere para desarrollar un proyecto de programación, es uno de los aspectos más difíciles e importantes. Esto consiste en determinar las aproximaciones de tiempo, horas, días o meses, que se necesitan de esfuerzo para el desarrollo en un producto de programación.

Existen diversos factores que intervienen en ésto como son: las habilidades individuales del equipo involucrado en el proyecto, la complejidad del mismo, así como elementos externos que no se relacionan directamente con él, y por tanto, no se encuentran bajo el control directo de los responsables del proyecto. La precisión depende en gran medida de las habilidades, conocimientos y experiencia del personal que prepara las estimaciones. Existen tres métodos para realizar éstas funciones, se describen a continuación:

IV.2 METODO HISTORICO.

Consiste en comparar el plan del proyecto con los registros que se hallan llevado de trabajos previos. En estos registros se indican las características del programa o proyecto, la asignación de tareas, los requerimientos en tiempo del personal y los acontecimientos extraordinarios. Cuando se proponen nuevos proyectos, se comparan con los registros llevados para obtener la estimación del tiempo esperado en el desarrollo. Este método consume mucho tiempo y es tan efectivo como los registros que se tengan, además de ser útil sólo si el proyecto nuevo posee características similares a algún desarrollo anterior. Es de vital importancia este método ya que, el método de la fórmula estándar considera diversos factores que si se cuenta con datos históricos la estimación del tiempo será más exacta.

IV.3 METODO INTUITIVO.

Se basa en las estimaciones del personal con más experiencia para determinar el tiempo esperado del proyecto. El método es descrito como predicción aprendida, ya que el individuo lo efectúa de acuerdo al aprendizaje de sus vivencias en el desarrollo de proyectos de programación en que halla participado. No utiliza casos documentados o registros históricos como el método anterior, y aunque son pocos los que pueden efectuar una estimación con un alto grado de precisión, es un método rápido y conveniente; ejemplo, método DELFHI.

IV.4 METODO DE LA FORMULA ESTANDAR.

A pesar de que se han sugerido muchas fórmulas para lograr una estimación acertada del tiempo de desarrollo, no se ha logrado una fórmula verdaderamente estándar para su evaluación, debido a que los proyectos poseen características de lo más diversas. Sin embargo, todas ellas intentan detectar los factores individuales que afectan el tiempo de desarrollo para su posterior cuantificación. Estos factores, como las características personales, los detalles del sistema y la complejidad del proyecto, reciben puntuaciones por separado. Una fórmula aritmética nos muestra cómo relacionar los elementos individuales para producir una estimación del tiempo de desarrollo en horas, días o semanas. Para estimar el tiempo de un proyecto con gran precisión los métodos más conocidos y usados son: "Método COCOMO" y "Análisis por Puntos de Función (FPA)", detallado a continuación.

IV.5 ANALISIS POR PUNTOS DE FUNCION.

Es una forma de estimar los requerimientos de tiempo en un proyecto de programación, para llevar a cabo la estimación es necesario evaluar lo siguiente: entradas, salidas, archivos, interfases, consulta externa de entradas y salidas. Para cada una se determinará considerando sus características, el grado de complejidad que presenta, puede ser simple, complejo o un promedio. Existen una tablas que muestran el grado de complejidad para cada función evaluada, es necesario registrar la evaluación para obtener los puntos de función.

IV.5.1 ENTRADAS.

Se analizan todos los campos de las entradas que se tienen, por sus características pueden ser:

SIMPLES
- Si hay pocos datos.
- Si existen pocas referencias a archivos internos.
- Si no se consideran los factores humanos.

COMPLEJOS
- Si son muchos datos.
- Si hay muchas referencias a archivos internos.
- Si los factores humanos afectan al diseño.

Otros factores que se deben considerar son:
- Si se desea movimiento de cursores automáticos.
- Si hay que hacer conversión de datos.
- El funcionamiento de la aplicación.

IV.5.2 SALIDAS.

Los campos de las salidas pueden ser :

SIMPLES
- Si tiene pocas columnas.
- Si la transformación de los datos es simple.

PROMEDIO
- Si tiene múltiples columnas.
- Si se manejan subtotales.
- Si hay múltiples transformaciones de los datos.

COMPLEJA
- Intrincadas transformaciones de los datos.
- Referencias complejas a archivos.
- Consideraciones significantes de funcionamiento.

Factores adicionales a considerar:
- El número de subtotales que contiene.
- Las transformaciones de los datos que maneja.
- El funcionamiento de la aplicación.

Para la evaluación de las entradas y salidas se recurre a la tabla mostrada en la figura 4.1, una S para las simples, C para las complejas y A cuando es un promedio, la evaluación esta en función de los campos que las constituyen, se registrará la evaluación y se obtiene un total de todas aquellas que fueron simples, complejas o promedio. El formato propuesto para llevar a cabo lo anterior se muestra en la figura 4.2 diferenciando solamente el tipo de función que se esta evaluando.

ENTRADA

CAMPOS	1 - 4	5 - 15	) 15
0 - 1	S	S	A
2	S	A	C
) 2	A	C	C

SALIDA

CAMPOS	1 - 5	6 - 19	) 19
0 - 1	S	S	A
2 - 3	S	A	C
) = 4	A	C	C

FIGURA 4.1. TABLAS DE EVALUACION FPA.

TIPO DE FUNCION : _____

DESCRIPCIONES	S	A	C
TOTAL			

FIGURA 4.2. FORMATO DE EVALUACION FPA.

IV.5.3 ARCHIVOS.

Los archivos por sus caracteristicas pueden ser :

SIMPLES
- Si cuenta con pocos tipos de registros.
- Si tiene pocos elementos de datos.
- No es significativo el funcionamiento y la recuperación.
- Pocos niveles jerárquicos.

COMPLEJO
- Si tiene muchos tipos de registros.
- Si tiene muchos elementos de datos.
- Si es significativo el funcionamiento y la recuperación.
- Contiene muchos niveles jerárquicos.

Factores adicionales a considerar
- El funcionamiento.
- La recuperación/respaldo.
- Criterios de búsqueda.

IV.5.4 INTERFASES.

Igual a la evaluación de archivos sólo que para el grado complejo además se debe considerar lo siguiente:

- Si un archivo es accesado por otra aplicación entonces, dicho archivo se contará en ambas funciones (archivos, interfases).

Factores adicionales a considerar
- El funcionamiento de la aplicación.
- Criterios de búsqueda.
- La recuperación/respaldo.
- Si la distribución es múltiple.
- La conversión.

Para la evaluación de los archivos y las interfases se recurre a la tabla mostrada en la figura 4.3, una S si son simples, C si son complejos y A cuando es un promedio, la evaluación está en función de los campos del archivo o de la interfase y los tipos de registros que los constituyen. Se registrará la evaluación como en las funciones anteriores en el formato propuesto sólo diferenciar el tipo de función en cuestión.

ARCHIVO

ARCHIVOS / TIPO REG.	1 - 19	20 - 50	> 50
1	S	S	A
2 - 5	S	A	C
> = 6	A	C	C

INTERFASE

ARCHIVOS / TIPOS REG.	1 - 19	20 - 50	> 50
1	S	S	A
2 - 5	S	A	C
> = 6	A	C	C

FIGURA 4.3. TABLAS DE EVALUACION FPA.

IV.5.5 CONSULTAS EXTERNAS.

Tipos de consultas a considerar:

- Uso de consultas sin actualizar archivos.
- Mensajes de ayuda.
- Menues de selección.

Para evaluar la complejidad de las consultas considerar en forma independiente la consulta de las entradas y las de las salidas, recurir a la tabla de la figura 4.1 para las entradas y para las salidas la de la figura 4.4. Registrar la evaluación para cada consulta en el formato propuesto.

SALIDA

CAMPOS	1 - 5	6 - 19	> 19
0 - 1	S	S	A
2 - 3	S	A	C
> 3	A	C	C

FIGURA 4.4. TABLAS DE EVALUACION EPA.

Después de haber evaluado las funciones, se realiza un resumen para obtener los puntos de función que consiste en lo siguiente:

Por cada función se toma el total de las que se evaluaron como simples, promedio y complejas, se multiplicarán por una constante que se muestra en la tabla 4.5, obteniendo subtotales por cada función y conjuntandolos obtendremos un total que será llamado Total de Puntos de Función no Ajustados.

ENTRADAS		Simples	*	3	=
		Promedio	*	4	=
		Complejos	*	6	=
	Subtotal				
SALIDAS		Simples	*	4	=
		Promedio	*	5	=
		Complejos	*	7	=
	Subtotal				
ARCHIVOS		Simples	*	7	=
		Promedio	*	10	=
		Complejos	*	15	=
	Subtotal				
INTERFASES		Simples	*	5	=
		Promedio	*	7	=
		Complejos	*	10	=
	Subtotal				
CONSULTAS		Simples	*	3	=
		Promedio	*	4	=
		Complejos	*	6	=
	Subtotal				
PROCESOS DE RESPALDO		Simples	*	10	=
		Promedio	*	20	=
		Complejos	*	30	=
	Subtotal				
TOTAL PUNTOS DE FUNCION NO AJUSTADOS					=

FIGURA 4.5. CONTABILIZANDO LAS FUNCIONES.

El paso siguiente será evaluar la complejidad del procesamiento considerando las características que se muestran en la figura 4.6, los valores que pueden tomar van de cero (simple) a cinco (complejo), se obtiene un total que será llamado Grado Total de Influencia (ID). Con éste dato se calcula el Factor de Ajuste utilizando la siguiente fórmula.

$$\text{Factor de Ajuste} = 0.65 + (0.01 * DI)$$

CARACTERISTICAS	DI	CARACTERISTICAS	DI
1.- Comunicación de datos		8.- Actualización en linea	
2.- Funciones distribuidas		9.- Procesamiento complejo	
3.- Funcionamiento		10.- Reusabilidad	
4.- Muy usado		11.- Fácil de instalar	
5.- Transacción		12.- Fácil de operar	
6.- Entrada en linea		13.- A nivel red	
7.- Efectivo al usuario		14.- Fácil de cambiar	

FIGURA 4.6. COMPLEJIDAD DEL PROCESAMIENTO.

Utilizando la siguiente formula se obtienen los puntos de función.

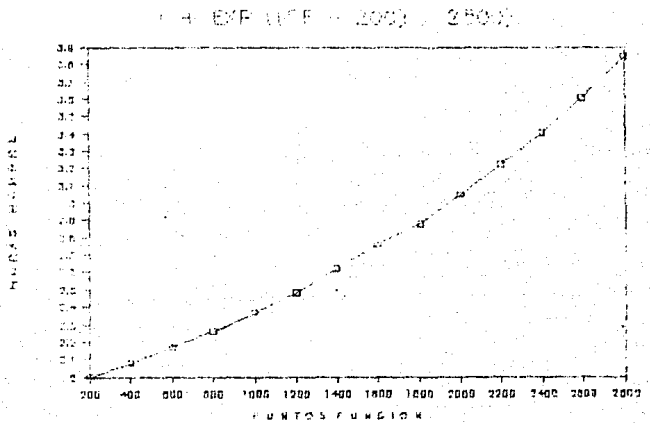
$$\text{Puntos de Función} = \text{Total de Puntos de Función no Ajustados} * \text{Factor de Ajuste}$$

Para obtener las horas/hombre requeridas se emplea la siguiente formula.

$$\text{Horas/Hombre} = (PFA / a) ** b$$

Donde:
a y b son factores de productividad (experiencia del staff, uso de lenguajes, facilidades de software y hardware, etc).

Para nuestros casos evaluados tenemos que, por cada punto de función obtenido corresponde a dos horas/hombre (ver figura 4.7), así mismo, se puede obtener el total de días y recursos que se requieren dependiendo de las horas que se labore, estando en función de la gente que participará.



IV.6 REQUERIMIENTOS DE TIEMPO CALENDARIO.

Existen otras actividades que consumen tiempo adicional al identificado en los requerimientos del tiempo en un proyecto de programación. Las reuniones de la gerencia, revisiones del proyecto, capacitación, interacción con los usuarios, incapacidades, vacaciones, días festivos, etc. extienden el programa más allá de lo estimado. A continuación se describen dos métodos para planear los requerimientos de tiempo calendario.

IV.6.1 GRAFICA DE BARRAS (GANTT).

Este método consiste en identificar primero cada tarea y estimar el tiempo que es necesario para desempeñarla. Cuando ésta información se transfiere a la gráfica de barras, las tareas se listan de arriba a abajo en la parte izquierda de la gráfica, en el orden que se llevarán a cabo. El tiempo de calendario se ilustra de izquierda a derecha. Una horizontal se marca en la gráfica para cada tarea, indicando cuando dá principio y cuando se espera su terminación. La ausencia de una barra significa que no se asocia ninguna tarea con la actividad durante un periodo específico. Debajo de cada barra se coloca una segunda indicando el tiempo real que vaya consumiendo la actividad. La figura 4.8 muestra un ejemplo del desarrollo de un proyecto de programación ilustrado por una Gráfica de Gantt.

Cuando el proyecto consiste de un número limitado de actividades o tareas, las gráficas de barras son perfectamente manejables, en caso contrario, el tamaño de la gráfica se vuelve desproporcionado al incluir demasiadas barras, dificultando el manejo de la información.

Las gráficas de Gantt pueden manejarse por niveles comunicando la información relativa a la planeación. Una gráfica general de planeación muestra las principales tareas. Las gráficas se pueden realizar al nivel de detalle que se necesite, subdividiendo las actividades en otras más pequeñas para su manejo y control individual, dependiendo de su utilización.

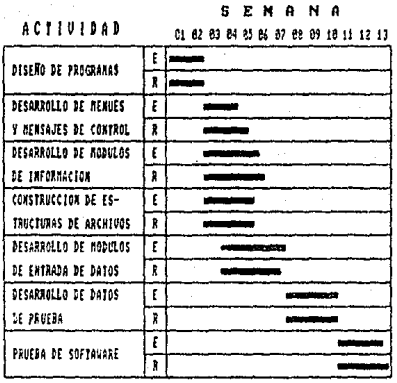


FIGURA 4.8. GRAFICA DE BARRAS (GANTT).

IV.6.2 GRAFICAS PERT.

Es el método más complejo para la planeación de proyectos. Los proyectos consisten en acontecimientos y actividades. En este tipo de gráficas, a diferencia de las de barras, se muestra la interdependencia entre actividades y las tareas críticas que deben complementarse a tiempo y en una secuencia específica.

En la gráfica se utilizan círculos (nodos) y caminos (rectas o arcos) para representar la interrelación de las actividades del proyecto (figura 4.9). Los nodos representan acontecimientos y los caminos muestran actividades que se requieren para adelantarse de un acontecimiento a otro. Los números indican el tiempo necesario para llevar a cabo cada actividad.

Al terminar la gráfica de relaciones se estudia para determinar la ruta crítica, es decir, el camino que es necesario llevar desde el principio hasta el fin y por el cual el tiempo total requerido será mayor que por cualquier otro camino (se muestra con línea gruesa en la figura 4.9). Si las actividades a lo largo de este camino no se terminan a tiempo, todo el proyecto se retrasará, por tanto, se debe prestar especial atención a estas actividades.

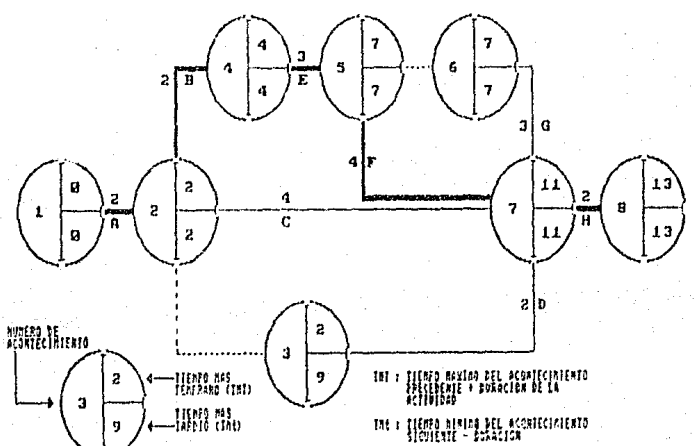


FIGURA 4.9. GRAFICA PERT.

IV.6.2 GRAFICAS PERT.

Es el método más complejo para la planeación de proyectos. Los proyectos consisten en acontecimientos y actividades. En este tipo de gráficas, a diferencia de las de barras, se muestra la interdependencia entre actividades y las tareas criticas que deben complementarse a tiempo y en una secuencia especifica.

En la gráfica se utilizan circulos (nodos) y caminos (rectas o arcos) para representar la interrelación de las actividades del proyecto (figura 4.9). Los nodos representan acontecimientos y los caminos muestran actividades que se requieren para adelantarse de un acontecimiento a otro. Los números indican el tiempo necesario para llevar a cabo cada actividad.

Al terminar la gráfica de relaciones se estudia para determinar la ruta critica, es decir, el camino que es necesario llevar desde el principio hasta el fin y por el cual el tiempo total requerido será mayor que por cualquier otro camino (se muestra con linea gruesa en la figura 4.9). Si las actividades a lo largo de este camino no se terminan a tiempo, todo el proyecto se retrasará, por tanto, se debe prestar especial atención a estas actividades.

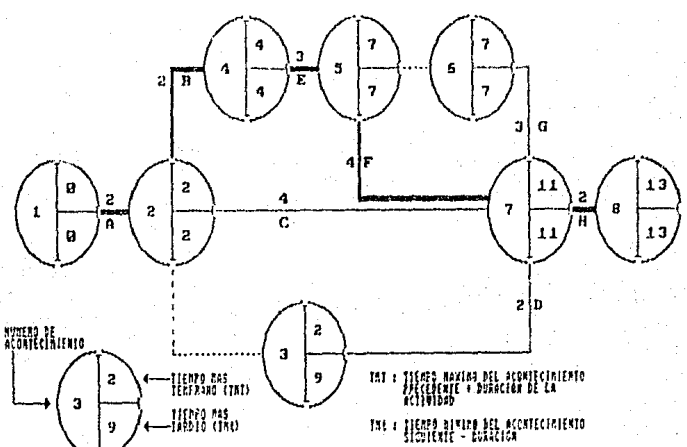


FIGURA 4.9. GRAFICA PERT.

Para el desarrollo de una red PERT en un proyecto de programación deben identificarse primeramente las tareas y las relaciones a cada una de ellas. Posteriormente se debe determinar la secuencia de actividades y los lugares donde tareas específicas deben preceder a otras y donde ciertas actividades ocurren de manera simultanea con otra.

El siguiente paso consiste en analizar el programa de tiempos de proyecto. Lo que se debe determinar es:

1) ¿Qué tan pronto puede iniciarse un acontecimiento?

2) ¿Cuánto puede tardarse en iniciar éste, sin afectar el programa general del proyecto?

El tiempo más temprano (TMT) es cero para el primer acontecimiento, y para las demás actividades, la suma de la duración más alta de la actividad y el TMT de cualquier acontecimiento inmediatamente precedente. El TMT se computa mediate el análisis del TMT y el tiempo de duración para cada acontecimiento que le sucede. La figura 4.9 muestra el TMT para toda la red. Una notación utilizada como referencia muestra al TMT en la parte derecha superior del nodo y el TMt (tiempo más tardio) en la parte derecha inferior.

El tiempo más tardio del acontecimiento es el tiempo limite en el cual puede dar inicio sin retrasar el proyecto. Para determinar este tiempo es necesario trabajar retrospectivamente en la red iniciando desde la derecha. El tiempo más tardio constituye la diferencia menor entre el TMt del acontecimiento terminal menos el tiempo de duración de la actividad.

La ruta critica es el conjunto de actividades que se deba supervisar más cercanamente. Identifica los acontecimientos que se deben iniciar y terminar a tiempo y que requieren no más que el tiempo de duración estimado; de otro modo, el proyecto sufrirá un retraso.

En la figura 4.9 se ilustra con la línea gruesa, la ruta crítica del proyecto. Se determinó al vincular todos los nodos en donde los TMT y TMt son iguales, lo que significa que no hay posibilidad de cambio o desviación, es decir, no hay tiempo de tolerancia u holgura. El tiempo de holgura que se relaciona con un acontecimiento se puede ilustrar formalmente restando el tiempo de duración y el TMT del nodo inicial del TMt del nodo terminal.

En un momento dado puede usarse una combinación de gráficas, PERT para planear el desarrollo e ilustrar las interdependencias, y gráficas de barras para mostrar los programas de calendarios.

IV.7 COSTOS Y BENEFICIOS.

El análisis de costos y beneficios es uno de los aspectos que determinan si un sistema es aceptado, se debe de asegurar de que se identifiquen y estimen apropiadamente todos los costos. Los costos varían según el tipo, y consisten en varios elementos distintos. Los beneficios también varían de acuerdo con su tipo y se clasifican según las ventajas que proporcionan.

Los costos asociados con el sistema del negocio son los gastos, salidas o pérdidas resultantes del desarrollo y del uso del sistema. Los beneficios son las ventajas que se obtienen de la instalación y utilización del mismo. Las tres clasificaciones principales de costos y beneficios son: tangibles o intangibles, fijos o variables y directos o indirectos. Estas categorías no se excluyen; por tanto, un aspecto del costo o beneficio puede clasificarse dentro de más de una categoría al mismo tiempo.

IV.7.1 COSTOS Y BENEFICIOS TANGIBLES O INTANGIBLES.

El término "costo" se iguala a menudo con dinero o finanzas, sin embargo las salidas de efectivo sólo son un tipo de costos, llamados en este caso costos tangibles. Se saben que existen algunos costos, como el valor de la pérdida de un cliente, o un descenso en la imagen de la compañía, pero su monto financiero no puede determinarse con exactitud. Estos son costos intangibles. El estimado es una aproximación. No es factible fijar costos intangibles exactos. La mayor parte de los costos son tangibles y se pueden identificar por los analistas.

Los beneficios también clasificados como tangibles o intangibles, a menudo son más difíciles de especificar en forma exacta que los costos. El valor del beneficio es una ventaja que se gana a través de la utilización del sistema. Los beneficios tangibles como reducción de gastos o menores tasas de error son cuantificables. Los beneficios intangibles, como el valor de un mejor servicio al cliente, una respuesta más rápida a las solicitudes de los usuarios o mejores condiciones de trabajo, a menudo se pueden cuantificar. Los proyectos de sistemas no se deben desarrollar sólo sobre la base de beneficios intangibles.

IV.7.2 COSTOS Y BENEFICIOS FIJOS O VARIABLES.

Algunos costos y beneficios de sistemas son constantes y no cambian, sin importar cuánto se utilice un sistema de información. En contraste, los costos y beneficios variables son aquellos en los que se incurre en proporción a la actividad o al tiempo. Los costos de suministro de computadora varian en proporción con el monto del proceso que se lleva a cabo.

IV.7.3 COSTOS Y BENEFICIOS DIRECTOS O INDIRECTOS

Si los costos y beneficios son atribuibles a un sistema de negocios, al sistema de información o actividad de trabajo se denominan directos. En otras palabras, utilizar el sistema o hacer el trabajo directamente produce costos y beneficios. Reducir el costo de los errores es un beneficio directo.

Los costos indirectos son gastos de apoyo de tipo extra que, mientras son reales o sustanciales, no están especificamente asociados con el sistema de información. Son el resultado de operar otros sistemas o llevar a cabo actividades necesarias dentro de la empresa, que apoyan al sistema investigado. Los beneficios indirectos se consiguen como un subproducto de otro sistema.

Para cada proyecto se puede calcular una estimación de la relación Costo/Beneficio que nos puede representar un sistema, mediante la siguiente fórmula ROI (Return On Investment).

$$ROI = \frac{\text{Beneficios Estimados}}{\text{Costos Estimados}}$$

Para que sea factible : ROI > 1

IV.8 CONTROL DE CALIDAD.

El control de calidad se basa en los estándares del proyecto y supervisa que se respeten, se llevan a cabo auditorías de los proyectos y productos y se realizan las pruebas de aceptación, ésto se puede hacer con la participación del usuario.

El control de calidad es un modelo planeado y sistemático de todas las acciones necesarias para proporcionar la confianza de que el artículo o producto se ajusta a los requerimientos técnicos establecidos. El propósito de un grupo en el control de calidad del software es proporcionar la garantía de que los procedimientos, las herramientas y las técnicas utilizadas durante el desarrollo y la modificación del producto son adecuados para alcanzar el nivel de confianza deseado en los productos de trabajo.

La preparación de un Plan de Control de Calidad del Software para cada proyecto de programación debe tocar los siguientes temas:

1.- Propósito y alcance del plan.

2.- Documentos referidos en el plan.

3.- Estructura organizacional, tareas que se realizarán y responsabilidades específicas relacionadas con la calidad del producto.

4.- Documentos que se deben preparar y revisiones que deben efectuarse para la adecuación de la documentación.

5.- Estándares, prácticas y conversiones que se utilizarán.

6.- Revisiones y auditorías que deben llevarse a cabo.

7.- Un plan de: la administración de configuración que identifique los elementos del producto de software, control e implantación de los cambios y que se registre e informe de los estados modificados.

8.- Prácticas y procedimientos que se deben seguir para informar, rastrear y resolver los problemas del software.

9.- Herramientas y técnicas específicas que se usarán para apoyar las actividades del control de calidad.

10.- Métodos y facilidades que se usarán para reunir, mantener y conservar los registros del control de calidad.

Otras tareas desarrolladas por el personal del control de calidad son:

1.- Generación de políticas y procedimientos estándar.

2.- Desarrollo de herramientas de prueba y otros auxiliares para el control de calidad.

3.- Ejecución de las funciones del control de calidad descritas en el Plan del Control de Calidad del Software para cada proyecto.

4.- Ejecución y documentación de las pruebas de aceptación del producto final para cada producto del software.

No hay que olvidar, como una consideración importante para la planeación del proceso de desarrollo, el modelo del ciclo de vida del producto de programación.

La metodología aquí propuesta, en base al desarrollo con prototipos, permite un manejo fácil y eficiente mediate los métodos planteados para la administración de proyectos. Las tareas básicas que se han descrito en los cápitulos correspondientes a cada etapa del desarrollo del sistema, dan lugar a una buena adecuación para el manejo y control del proyecto.

Cada proyecto especifico contiene sus características particulares, que deben contemplarse, cuando se realize la planeación del proyecto, de tal modo que las actividades a realizarse, queden perfectamente identificadas y se consiga la administración apropiada del proyecto de programación.

V.- ESTRUCTURA ORGANIZACIONAL DE UNA UNIDAD DE SERVICIOS DEL MANEJO DE INFORMACION EN UN AMBIENTE DE 4a. GENERACION

V. ESTRUCTURA ORGANIZACIONAL DE UNA UNIDAD DE SERVICIOS DEL MANEJO DE INFORMACION (MIS) EN UN AMBIENTE DE CUARTA GENERACION.

INTRODUCCION.

Durante las fases del ciclo de vida en el desarrollo de sistemas, se deben realizar diversas actividades que comprenden la planeación, la implementación y mantenimiento, así como otras actividades implícitas a las ya mencionadas, como pueden ser servicios, publicaciones, control de calidad, apoyo, etc., los métodos para organizar estas tareas pueden ser los formatos de proyecto, el funcional y el matricial.

Cada equipo de trabajo debe contar, además, con una estructura interna, que varía de acuerdo a cada proyecto y a cada producto. Las estructuras básicas son: el grupo democrático, en el que todos los miembros participan en todas las decisiones; el grupo con jefe de programación, en el que otros miembros del equipo apoyan y auxilian al jefe y por último, el grupo jerárquico que combina aspectos de los dos anteriores. Pueden existir variaciones y combinaciones de las tres estructuras mencionadas.

El hecho de utilizar productos de 4a. Generación implica también, el tomar en consideración cambios en la estructura organizacional en la Unidad MIS, para que se adapte a las situaciones y necesidades requeridas.

A continuación se detalla la propuesta para la estructura organizacional de una unidad de servicios del manejo de información en un ambiente de 4a. Generación.

V.1 ESTRUCTURA ORGANIZACIONAL.

La estructura de las Unidades de Servicios de Manejo de Información (MIS), durante la década de los 70's, ha variado sensiblemente de acuerdo a la naturaleza de las actividades que éstas desarrollan, sin embargo, cumplen con un patrón básico similar al que se presenta en la figura 5.1.

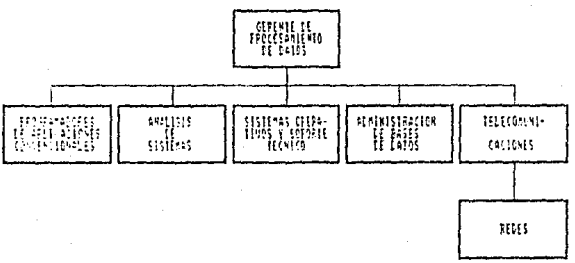


FIGURA 5.1 ESTRUCTURA ORGANIZACIONAL TIPICA DE LA DECADA DE LOS 70'S.

Como se ha visto en capítulos anteriores, los Lenguajes de 4a. Generación, han traído cambios importantes en el desarrollo de proyectos de programación, con el fin de optimizar la productividad. Es lógico pensar que también las estructuras básicas organizacionales de las Unidades MIS, deban sufrir las modificaciones necesarias para adecuarse y cumplir con los requisitos de un ambiente de 4a. Generación.

Las soluciones que se sugieren giran, precisamente, alrededor de este ambiente, específicamente en sus herramientas y técnicas. Por otro lado, se pretende ligar lo más estrechamente posible, el conocimiento de procesamiento de datos, al de la línea de la organización a la que pertenezca la Unidad MIS en cuestión.

El objetivo de muchas de las Unidades MIS es crear verdaderos centros de información y ésto se puede lograr combinando la experiencia del personal de Desarrollo de Sistemas, con la capacidad de cómputo orientado al usuario final. Estos entienden que información, reportes y soporte a las decisiones necesitan y los expertos en sistemas saben como organizarlos y como puede ser obtenida. Balanceando estos conocimientos se puede lograr el máximo de productividad.

En la figura 5.2. se presenta una aproximación de la estructura básica organizacional recomendada para un ambiente de 4a. Generación, y a continuación la descripción de sus módulos.

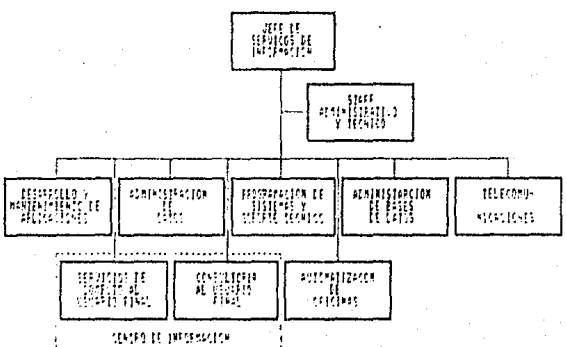


FIGURA 5.2 ESTRUCTURA ORGANIZACIONAL EN UN AMBIENTE DE CUARTA GENERACION.

V.1.1 JEFE DE LA UNIDAD MIS Y DEL "STAFF" ADMINISTRATIVO Y TECNICO.

El jefe de la Unidad MIS es el responsable del establecimiento y funcionamiento de ésta, de manera que satisfaga las necesidades, en lo referente a la información, requerida por la organización para que pueda alcanzar sus objetivos.

El jefe de la Unidad MIS cuenta con un grupo "STAFF" integrado por personal, tanto a nivel administrativo como técnico, a fin de asesorarlo en cualquiera de estos ramos, en la dirección de la Unidad MIS. Este grupo suple al jefe cuando es necesario, además de supervisar y coordinar las funciones de tipo "STAFF" de la Unidad MIS, tales como: planeación, control de proyectos, medidas de seguridad, servicios administrativos y técnicos, etc.

V.1.2 DESARROLLO Y MANTENIMIENTO DE APLICACIONES.

Esta área debe soportar los sistemas desarrollados con la metodología tradicional. Se encarga de realizar, de la mejor manera, el desarrollo de sistemas, contando con analistas y programadores de lenguajes convencionales, para asegurar el funcionamiento y el mantenimiento a los sistemas existentes, de acuerdo a los requerimientos de mantenimiento u optimización a los mismos, y que gradualmente se incorporarán al Centro de Información bajo nuevas metodologías.

V.1.3 CENTRO DE INFORMACION.

Esta es un área dedicada al cómputo orientado al usuario final y se integra, en la figura 5.2, por los bloques SERVICIOS DE COMPUTO AL USUARIO FINAL, y CONSULTORIA AL USUARIO FINAL. Cuenta con especialistas técnicos en Lenguajes de 4a. Generación y consultores por línea de operación de la organización, que entrenan y asisten a los usuarios finales, tanto en el uso de herramientas de productividad, como en el desarrollo de nuevas aplicaciones usando los Lenguajes de 4a. Generación. Le deben proporcionar herramientas, asistencia y entrenamiento. Algunos productos como los Lenguajes de Consulta, los Generadores de Reportes y los Sistemas de Planificación Financiera, son ejemplos de los paquetes de Software que puede proporcionar el Centro de Información. Para apoyar el análisis de los datos efectuado por el usuario, los asesores del Centro de Información trabajan junto con ellos para poner a su alcance los archivos y bases de datos necesarios, provenientes de la computadora principal.

V.1.4 ADMINISTRACION DE DATOS.

Esta área será la encargada de administrar y controlar la creación y ubicación de cada uno de los datos que sean definidos en los sistemas. En otras palabras es el manejo, como recurso, de los datos y la información de una organización.

El flujo de los datos e información en una organización puede ser complejo. Conforme el mismo dato pasa de un departamento a otro puede ser visto de diferente manera. Por ejemplo, si un cliente realiza un pedido se vuelve una orden de venta y parte del volumen total de ventas; una estadística demográfica para estudios de mercado; para el departamento de procesamiento de pedidos, un pedido que hay que vigilar; para la oficina de finanzas es una variable en la proyección de flujo de efectivo; para el almacén es una orden de abastecimiento y para el fabricante es una orden de producción. Los usuarios tienen su propio punto de vista sobre estos datos y tienen diferentes necesidades de información derivada de ellos, así como de actualización. El personal de operación requiere detalles, y la gerencia resúmenes.

Los Administradores de Datos son los que se encargan de organizar todo ésto, se ocupan del análisis de las necesidades del usuario en lo referente a datos e información a través de las líneas departamentales, así como del desarrollo de los puntos de vista del usuario en relación con estos datos. Sus funciones son clasificar grupos de datos relacionados, desarrollando modelos que describen a los datos en sí y sus relaciones con otros de ellos y con los usuarios. Los modelos especifican que datos cruzan fronteras departamentales dentro de una organización. Para desarrollar el marco adecuado en el diseno de bases de datos deberán visualizarse los datos y la información incluyendo a toda la organización. El desarrollo de un Diccionario de Datos, documenta el análisis realizado por el Administrador de Datos. Los Modelos y el Diccionario de Datos, combinados con el volumen de transacciones (realizado por el Analista de Sistemas), son la materia prima para el diseno de Bases de Datos.

V.1.5 SOPORTE TECNICO Y PROGRAMACION DE SISTEMAS.

Esta área es la encargada de proporcionar Soporte Técnico en Software para que los usuarios y áreas de producción exploten racional y eficientemente los recursos de cómputo disponibles, operan e integran la red de teleproceso adecuandola a las necesidades y a los mismos recursos. También debe proporcionar servicios de mantenimiento preventivo y correctivo a las terminales de los equipos de cómputo, además de realizar auditorias en cómputo y los elementos que lo conforman para proporcionar medidas preventivas y/o correctivas a los mismos.

V.1.6 TELECOMUNICACIONES.

Tradicionalmente, el Departamento de Telecomunicaciones, efectuaba sus operaciones de control de tráfico telefónico de una organización de manera separada e independiente. Con la aparición de los conmutadores telefónicos modernos que trabajan con senales digitales, la frontera entre la computación y las comunicaciones se ha reducido. Como resultado, las líneas de control y de mando tradicionales de las comunicaciones y de las computadoras han empezado a utilizarse de manera sistemática en muchas companias.

Con la tendencia del procesamiento de datos a transformarse en un proceso distribuido y en el factor principal en la AUTOMATIZACION DE OFICINAS, cada vez necesita más Redes de Comunicaciones eficientes para su cometido. Es por esto que el área de TELECOMUNICACIONES es considerada dentro de la estructura organizacional.

V.1.7 ADMINISTRACION DE BASES DE DATOS.

Esta área se encarga de diseñar y controlar las Bases de Datos. Es la responsable del diseño físico de las estructuras de datos en una Base de Datos, además de la evaluación, selección e implementación del Sistema de Manejo de Bases de Datos. En organizaciones pequeñas, el Administrador de Datos y el Administrador de Bases de Datos casi siempre son la misma persona; sin embargo, la función del Administrador de Datos es el manejo de los datos y la del Administrador de Bases de Datos es la Programación de Sistemas. El Administrador de Bases de Datos implementa el Software de la Base de Datos, la cual cumple con las necesidades planteadas por el Administrador de Datos y los Analistas de Sistemas de la Organización.

V.1.8 AUTOMATIZACION DE OFICINAS.

Debido al gran auge que está teniendo, se ha incluido un área exclusiva para el desarrollo de la Automatización de Oficinas.

La Automatización de Oficinas constituye la integración de todas las funciones posibles de información de una oficina; es la incorporación de varias formas de procesamiento de información, incluyendo el procesamiento de datos, procesamiento de palabras, correo electrónico, y el graficado, así como la voz humana. El eje principal de la automatización de oficinas es una red de servicio local (dentro del mismo edificio o planta) que se constituye en la trayectoria de comunicaciones entre todos los usuarios y las computadoras. Los usuarios pueden crear, almacenar y recuperar cualquier forma de información (mensaje, correo, datos, voz, etc.) y transmitirla a cualquier otro usuario dentro de la organización.

Todas las funciones tradicionales de una oficina, como el dictado, la mecanografía, el archivo, el copiado, la operación del TWX y del TELEX, el manejo de microfilmes y de registros, las operaciones telefónicas y de comunicación telefónica, son elementos suceptibles de ser incluidos en un Sistema de Automatización de Oficinas.

Se mencionaron las áreas que se han considerado como las básicas en la integración de una estructura organizacional, incluyendo aquellas que se han ido manifestando como la tendencia en un ambiente de 4a. Generación, sin embargo, esto sigue siendo suceptible a modificaciones y/o adecuaciones al perfil, características y necesidades de cada organización particular.

VI.- EJEMPLOS DE DESARROLLOS DE SISTEMAS CON LA METODOLOGIA PROPUESTA

VI. EJEMPLOS DE DESARROLLOS DE SISTEMAS CON LA METODOLOGIA PROPUESTA.

INTRODUCCION.

Para mostrar la forma en que la metodología se aplica, se desarrollaron dos ejemplos que utilizan la metodología propuesta. Ambos ejemplos se apegaron a la metodología propuesta pero al desarrollarlos notamos que no todas las tareas de las fases de Análisis y Diseño aplicaban debido a las características propias de los mismos.

El primer ejemplo muestra el desarrollo del Sistema Subastas Múltiples de Certificados de la Tesorería de la Federación, el objetivo del sistema es desarrollar un mercado de tasas reales en donde podrán participar con igualdad de condiciones tanto Banco de México como las Casas de Bolsa, para llevar a cabo el objetivo del sistema se requiere de un microcomputador TOWER 32/600 y la herramienta de software a utilizar es el manejador de bases de datos relacional ISQL y el lenguaje de 4a. Generación INFORMIX.

Como segundo ejemplo se desarrollo un Sistema de Aduanas, cuyo objetivo es poder tener un estricto control sobre los pedimentos de importación que se manejan en las aduanas del país. Para realizar el sistema se requiere de un computador de UNISYS Serie A y del lenguaje de 4a. Generación LINC II.

A continuación se presentan de manera amplia los dos sistemas que ejemplifican el desarrollo de sistemas usando Lenguajes de 4a. Generación.

VI.1 A N A L I S I S.

VI.1.1 TAREA NUMERO UNO.

Información Para Entrevistas.

Para llevar a cabo el análisis del sistema "Subastas Múltiples De Certificados De La Tesorería De La Federación". Se entrevistarón los gerentes del área de Mercado de Dinero y el área de Automatización Operativa para obtener toda la información del sistema, especificando que necesita hacer el nuevo sistema así como los problemas que este resolverá, la información que será almacenada, los recursos y funciones de la empresa, todo lo anterior deberá quedar acentado en un documento de especificaciones del nuevo sistema, presentado a continuación.

En este sistema podrán participar en igualdad de condiciones tanto Banco de México como las casas de bolsa, buscando con esto el perfeccionamiento del mercado.

OBJETIVO

El objetivo principal del sistema es desarrollar un Mercado de Dinero más transparente, que se apoye en un mercado de tasas reales, para así poder establecer un grado mayor de competencia entre los diversos participantes.

POLITICAS GENERALES

La casa de bolsa que convoque a la subasta deberá especificar el método de convocatoria y asignación a seguir. En caso de que existan otras posturas del mismo lado de la subasta, deberán adecuarse al tipo de subasta convocada, bajo las políticas de operación que se presentan a continuación.

Banco de México tendrá prioridad en tiempo para efectuar subastas en el piso de remates. En caso de que Banco de México sea de venta, no podran adherirse a ella ninguna casa de bolsa.

Politicas De Operación Del Método De La Bolsa Mexicana De Valores.

CONVOCATORIA

La casa de bolsa subastadora a través de su operador de piso, entregará por escrito al personal de mercado de dinero las características de la subasta a realizar, indicando el tipo de operación de que se trate (contado, contado valor mismo día, reporto, reporto valor mismo dia). En el intervalo de tiempo que dure la subasta la información del monto, tasas y postores será confidencial.

POSTURAS

Las casas de bolsa interesadas en participar, deberán presentar para cada plazo o emisión subastada, ordenes en firme de compra o venta al personal del corro de mercado de dinero destinado para la subasta. En cada postura deberá indicarse el valor nominal en millones de pesos, la emisión y la tasa de descuento solicitada, en caso de reporto deberá de indicarse los días de reporto y la tasa premio.

ASIGNACION

Las tasas de descuento o premio que se indiquen en las posturas de venta, reflejarán las mayores a las que el oferente esté dispuesto a vender, en el caso de las posturas de compra las tasas serán las mínimas a las que el demandante esté dispuesto a comprar. La asignación se hara en función de la postura que presente mejores condiciones, siendo para la compra, el descuento o premio menor y para la venta el mayor.

HORARIO

El horario para la realización de las subastas será en "valor mismo día" de 10 am A 13 hrs y en liquidación 24 horas de 10 am A 13:45 hrs.

RESULTADOS

La subdirección de mercado de dinero dará a conocer los resultados de la asignación.

Características De La Convocatoria.

El operador de piso que realice la subasta deberá indicar en la orden en firme, las siguientes características:

a) Subasta de venta o compra en directo ("24 horas" y "valor miso día").

- Emisión.
- Valor nominal.
- Tasa de descuento.

b) Subasta de venta en reporto ("24 horas y "valor mismo día").

- Emisión.
- Valor nominal.
- Tasa de descuento.
- Tasa premio.
- Días de reporto.

c) Subasta de compra en reporto ("24 horas y "valor mismo día").

- Valor nominal.
- Tasa premio.
- Días de reporto.

Características De Las Posturas.

a) Subasta de venta o compra en directo ("24 horas" y "valor miso día").

- Emisión.
- Valor nominal.
- Tasa de descuento.

b) Subasta de venta en reporto ("24 horas y "valor mismo día").

- Valor nominal.
- Tasa de descuento.
- Tasa premio.
- Días de reporto.

c) Subasta de compra en reporto ("24 horas y "valor mismo día").

- Emisión.
- Valor nominal.
- Tasa premio.
- Días de reporto.

ASIGNACION.

TASA MULTIPLE.

PASOS A DESARROLLAR

1.- Utilizar las convocatorias y posturas de la subasta.

2.- Ordenar las convocatorias y las posturas.

3.- Asignación de montos.

ESPECIFICACION Y VALIDACIONES

1.- Ordenar las convocatorias y las posturas.
    1.1. Se ordenan las posturas de los convocadores y los postores de la siguiente manera:
        1.1.1. Para contado en base a la tasa de descuento y para reporto en base a la tasa premio.
        1.1.2. Si la subasta es de venta en orden descendente. Si la subasta es de compra en orden ascendente.

2.- Asignación de montos.
    2.1. Se toma la mejor postura de los postores y se compara contra las posturas de los convocadores. La asignación existirá siempre y cuando:

        * Si la subasta es de venta. La tasa de los postores tiene que ser menor a la de los convocadores y competitiva.
        * Si la subasta es de compra. La tasa de los postores tiene que ser mayor a la de los convocadores y competitiva.

    2.2. De las convocatorias que cumplan con la condición anterior se realizará una sumatoria de montos, determinando un monto total de convocatorias y posturas se obtiene un porcentaje aritmético de cada una de los postores ajustado a centésimas.

    2.3. El monto de las posturas será distribuido entre los convocadores de acuerdo al punto 2.2.

    2.4. Se determinan remanentes de las posturas y se toma la siguiente mejor postura y se efectua nuevamente el proceso del punto 2.3.

2.5. La asignación termina cuando ya no se cumple la condición del punto 2.1 o bien cuando se terminan los montos de los convocadores o de los solicitantes.

2.6. La tasa de asignación es la que presenten los postores.

3.- Para la prorrata de los montos se aplica la siguiente fórmula:

$$A = \frac{B}{C} * D$$

Donde:

A = Monto asignado a la casa de bolsa.
B = Monto que compra o vende la casa de bolsa.
C = Suma de montos que se van a prorratear.
D = Monto que falta de asignar.

TASA UNICA.

PASOS A DESARROLLAR

1.- Utilizar las convocatorias y posturas de la subasta.

2.- Todas las posturas que reciben asignación se celebran a la misma tasa.

3.- Obtención de la tasa única.

4.- Se ordenan las convocatorias y posturas por:

* Tasa Descuento (contado).
* Tasa Premio (reporto).
* Ventas de mayor a menor.
* Compras de menor a mayor.

5.- Asignación.

ESPECIFICACION Y VALIDACIONES

1.- Obtención de la tasa única.
1.1. Se evaluan las posturas de los convocadores y los postores para obtener la tasa única hasta que:

* Tasa del convocador sea mayor a la tasa del postor o
* El monto del convocador o postor sea cero.

2.- El valor de la tasa única será:

* Si la tasa del convocador es igual a la tasa del postor entonces la tasa única es igual a tasa del convocador.
* Si la tasa de convocadores es mayor a la tasa de postores o los montos de convocadores y postores es cero entonces, la tasa única es igual a la suma de la tasa anterior del convocador y del postor dividiendo el resultado entre dos.
* Si el monto de la convocatoria es cero y el monto del postor es diferente de cero la tasa única es igual a la tasa del postor anterior.
* Si el monto del convocador es diferente de cero y el monto del postor es cero la tasa única es igual a la tasa del convocador anterior.

3.- Asignación.

3.1. Una vez ordenadas las convocatorias y posturas se obtiene la sumatoria de montos de las convocatorias con tasa mayor o igual a la tasa única.

3.2. Para posturas obtener la sumatoria de montos con tasa menor o igual a la tasa única.

3.3. Si el monto del convocador es mayor que el monto de posturas entonces:

* Se prorratea a la menor tasa del convocador que alcance asignación.

3.4. Si el monto del postor es mayor que el monto del convocador entonces:

* Se prorratea a la mayor tasa del postor que alcance asignación.

4.- El prorrateo.

4.1. Se utiliza la misma fórmula que en tasa múltiple.

TASA EXCLUSIVA

PASOS A DESARROLLAR

1.- Ordenar las posturas de la contraparte.

2.- Obtener el monto total de las convocatorias y las posturas.

3.- Asignar por prioridad de tasas descuento (contado), premio (reporto).

* La subasta termina cuando se agota el monto del convocador o bien de las posturas.
* Si en las posturas existen varias casas de bolsa a la misma tasa y el monto del convocador no fuera sificiente se asignará a prorrata de los montos.

ESPECIFICACION Y VALIDACIONES

1.- Para contado.
1.1. Ordenar las posturas.

* Si son de venta tasa de descuento de mayor a menor.
* Si son de compra tasa de descuento de menor a mayor.

2.- Obtener el monto total de los convocadores y postores.

3.- Asignar por prioridad de tasas.
3.1. Si la sumatoria de montos del convocador es mayor a la sumatoria de montos de los postores se asigna todo a la tasa de cada postor.

3.2. De lo contrario se asignan montos iguales a la tasa de la postura y efectuando prorrata si exsisten dos o más tasas iguales, o si el monto no alcanza a cubrirlas.

3.3. La subasta termina hasta que se acabe el monto del convocador o bien de las posturas.

4.- Para reporto.
4.1. Si la convocatoria es venta.
4.1.1. Ordenar posturas.
* Se ordenan de menor a mayor tasa premio.
4.1.2. Asignar tomando en cuenta lo siguiente.
* Si el comprador puso emisión, se asigna de la emisión pedida.
* Si el comprador no puso emisión, se le asigna cualquier emisión, tomado en cuenta el orden en que entraron las ventas.
* Si existieran condiciones iguales y el monto no alcanzará a cubrirlas se asignará a prorrata.
* La asignación termina cuando se acabe el monto de las posturas o de la convocatoria.

4.2. Si la convocatoria es compra.

4.2.1 Ordenar posturas.

* Se ordenan de mayor a menor tasa premio.

4.2.2 Asignar tomando en cuenta lo siguiente:

* Si existe convocatoria para la emisión que se vende, esta tendrá prioridad de asignación, siempre y cuando exista convocatoria sin emisión.
* En caso de no existir esa emisión o no ser suficiente, el monto se asignará del monto sin emisión en caso de existir éste.
* Si existen condiciones iguales y el monto no alcanza a cubrirlas se asignará a prorrata.
* La asignación termina cuando se acabe el monto de las posturas o de las convocatorias.

5.- El prorrateo.

5.1. Se utiliza la misma fórmula que en tasa múltiple.

VI.1.2 TAREA NUMERO DOS.

Descripción Del Problema.

Requerimientos del usuario para el nuevo sistema basados en las entrevistas con los gerentes.

- DECLARACION DEL PROBLEMA.

Enriquecer el mercado de dinero con nuevas alternativas de operación que nos permitan contribuir de manera eficiente al desarrollo que requiere al mercado para su consolidación.

- OBJETIVO.

Desarrollo de un mercado de dinero más transparente que se apoye en un mercado de tasas reales, mediante la implementación de mecanismos de operación que permitan establecer un grado mayor de competencia entre los diversos participantes. Deberá cumplir con la mecánica operativa y las políticas de operación, así como con el marco normativo que regirá esta nueva modalidad de operación, en donde tanto Banco de México como las Casas de Bolsa podrán participar en igualdad de condiciones buscando con esto el perfeccionamiento del mercado.

- SOLUCION PROPUESTA.

Para llevar a cabo el objetivo del sistema, se propone la siguiente solución:

* Convocatorias.

Se tendrán Altas, Bajas, Cierre y Reportes de todas las convocatorias para llevar a cabo la asignación de montos.

* Posturas.

Se manejarán Altas, Bajas y Reportes de todas las posturas que participarán en la subasta, para así poder hacer la asignación de montos.

- DESCRIPCION DEL SISTEMA PROPUESTO.

El sistema propuesto se piensa realizar en un minicomputador TOWER 32/600 de NCR. El software se compone básicamente del Lenguaje de 4a. Generación INFORMIX y el manejador de la base de datos ISQL.

El sistema debe ser en línea por la dinámica que se requiere en Mercado De Dinero y con esto poder brindar un servicio adecuado y eficiente a los Operadores de las Casas De Bolsa.

Se pretende que Mercado De Dinero tenga un amplio control sobre los vendedores y compradores que participen en la subasta para conseguirlo se tendrá la información siguiente:

- Información de las Casas De Bolsa que cotizan en la Bolsa Mexicana De Valores.
- Información de los convocadores y de los Postores.
- Información de las políticas de operación de la subasta que se está asignando.

Esta información podra ser dada de alta o baja por el personal de mercado de dinero.

Deberan existir reportes de Convocatorias, Posturas y montos asignados asi como los resultados finales de la subasta.

VI.1.3 TAREA NUMERO TRES.

Costo Del Sistema.

El siguiente formato determina el costo del sistema puede variar dependiendo de la persona que sea la encargada de realizarlo, en este caso utilizamos el Análisis Por Puntos De Función.

ENTRADAS	Simples	0	*	3	= 0	
	Promedio	1	*	4	= 4	
	Complejos	1	*	6	= 6	
	Subtotal					10
SALIDAS	Simples	1	*	4	= 4	
	Promedio	2	*	5	= 10	
	Complejos	1	*	7	= 7	
	Subtotal					21
ARCHIVOS	Simples	1	*	7	= 7	
	Promedio	3	*	10	= 30	
	Complejos	1	*	15	= 15	
	Subtotal					52
CONSULTAS	Simples	0	*	3	= 0	
	Promedio	2	*	4	= 8	
	Complejos	2	*	6	= 12	
	Subtotal					20

TOTAL PUNTOS DE FUNCION NO AJUSTADOS = 103

CARACTERISTICAS	DI	CARACTERISTICAS	DI
1.- Comunicación de datos	3	8.- Actualización en línea	3
2.- Funciones distribuidas	3	9.- Procesamiento complejo	3
3.- Funcionamiento	3	10.- Reusabilidad	3
4.- Configuración de muchos usos	2	11.- Fácil de instalar	1
5.- Tasa de Transacciones	3	12.- Fácil de operar	1
6.- Entradas en línea	3	13.- Sites múltiples	5
7.- Eficiencia al usuario	3	14.- Fácil de cambiar	3

DI TOTAL = 39.

Factor de Ajuste = 0.65 + (0.01 * 39) = 1.04

Puntos de Función = 103 * 1.04 = 107.12

Considerando que un punto de función es igual a dos hora hombre tenemos:

214.24 Horas/Hombre

Y si el día está constituido por ocho horas/hombre.

26.78 Días/Hombre

Si el mes está constituido por 20 horas/hombre.

1.389 Meses/Hombre

Que equivalen a 2 Meses/Hombre.

Considerando un costo de 30 dólares por hora/hombre, el costo total del sistema será 1,713.92 dólares y se termirará en 2 meses.

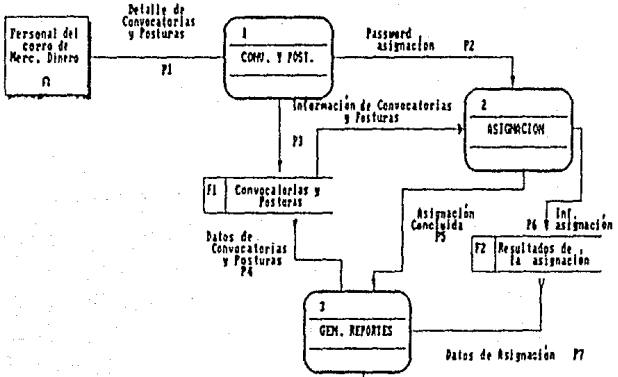
VI.1.4 TAREA NUMERO CUATRO.

Modelo Conceptual.

Entidades Eternas.

ENTIDAD	ENTRADA QUE RECIBE	SALIDA QUE PROPORCIONA
Personal del correo.	Datos de la(s) Convocatoria(s) y la(s) Postura(s).	Reporte de la(s) Convocatoria(s) y la(s) Postura(s) que participan en la subasta. Así como los resultados de la asignación y los resultados finales de la subasta.

Diagrama De Flujo De Datos.



Descripción Del Flujo De Datos.

CLAVE	NOMBRE	ORIGEN	DESTINO
P1	Detalles de Convocatorias y Posturas	A	1
P2	Password para Realizar la Asignación	1	2
P3	Información de Convocatorias y Posturas	1 F1	F1 2
P4	Datos de Convocatorias y Posturas	F1	3
P5	Asignación Concluida	2	3
P6	Información de la Asignación	2	F2
P7	Datos de la Asignación	F2	3
P8	Reporte	3	A

CLAVE	DESCRIPCION
P1	Datos completos de convocatorias y posturas que intervendán en la subasta.
P2	Password para poder llevar a cabo la asignación de montos.
P3	Información detallada de todas las convocatorias y posturas que intervienen en la subasta para poder asignar los montos respectivos.
P4	Reporte de todas las convocarorias y posturas que entraron en la subasta.
P5	Asignación concluida para poder obtener sus reportes.
P6	Información detallada de la asignación de montos.
P7	Reporte de los resultados de la asignación de montos.
P8	Reportes finales de la asignación de montos para ser anunciados por el área de mercado de dinero.

Descripción De Los Procesos.

1 Convocatorias y Posturas.
Proceso dedicado a incluir y dar de baja en el archivo de convocatorias y posturas los datos de las solicitudes y la hora de cierre para la subasta. Se entra a esta rutina cuando se desea dar de alta, baja o un reporte de convocatorias y posturas. Esta rutina se encarga de pasar el requerimiento de la asignación de montos al proceso asignación.

2 Asignación.
Este módulo recibe el password para llevar a cabo la asignación de acuerdo al método de la subasta. Este proceso permite accesar la información del archivo convocatorias y posturas que a su vez le sirve para crear uno que contiene los resultados de la asignación.

3 Genera reportes.
Módulo para presentar los resultados finales de la asignación en base a los datos contenidos en el archivo de resultados de la asignacion y el de convocatorias y posturas.

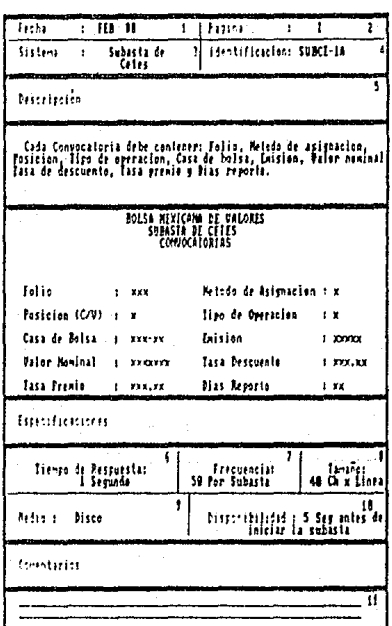
Normalización De Estructuras.

Más adelante se presenta la sección de estructura de datos, con la descripción de cada uno de los campos de los distintos archivos a usar en el sistema. Se observará que los archivos se encuentran ya normalizados por lo que no es necesario profundizar más al respecto.

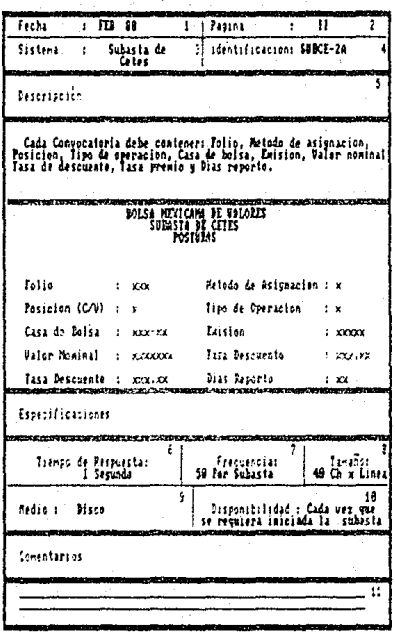
Requerimientos De Entrada/Salida De Información (RSI).

Descripción De Entradas.

Convocatorias.



Posturas.



Fecha : FEB 88 1	Pagina : 11 2
Sistema : Subasta de Cetes	Identificacion: SUBCE-2A 4

Descripción 5

Cada Convocatoria debe contener: Folio, Metodo de asignacion, Posicion, Tipo de operacion, Casa de bolsa, Emision, Valor nominal Tasa de descuento, Tasa premio y Dias reporto.

BOLSA MEXICANA DE VALORES
SUBASTA DE CETES
POSTURAS

Folio	: xxx	Metodo de Asignacion	: x
Posicion (C/V)	: x	Tipo de Operacion	: x
Casa de Bolsa	: xxx-xx	Emision	: xxxxx
Valor Nominal	: x,xxx,xxx	Tasa Descuento	: xxx.xx
Tasa Descuento	: xxx.xx	Dias Reporto	: xx

Especificaciones

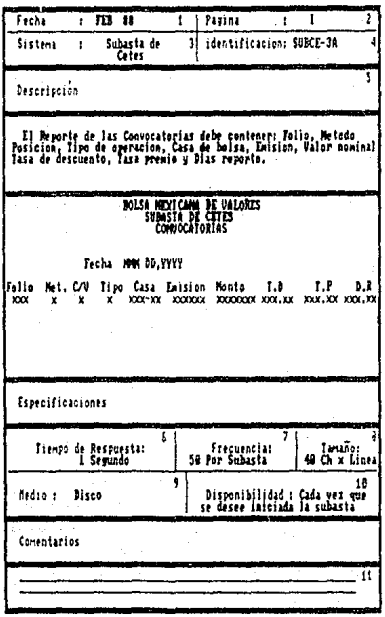
6	7	8
Tiempo de Respuesta: 1 Segundo	Frecuencias: 50 Por Subasta	Tamaños: 40 Ch x Linea

9	10
Medio : Disco	Disponibilidad : Cada vez que se requiera iniciada la subasta

Comentarios 11

Descripción De Salidas.

Convocatorias.



Fecha : FEB 88 1	Pagina : 1 2
Sistema : Subasta de Cetes 3	identificacion: SUBCE-3A 4

5

Descripción

El Reporte de las Convocatorias debe contener: Folio, Metodo Posicion, Tipo de operacion, Casa de bolsa, Emision, Valor nominal Tasa de descuento, Tasa premio y Dias reporto.

BOLSA MEXICANA DE VALORES
SUBASTA DE CETES
CONVOCATORIAS

Fecha MMM DD,YYYY

Folio	Met.	C/V	Tipo	Casa	Emision	Monto	T.D	T.P	D.R
XXX	X	X	X	XXX-XX	XXXXXX	XXXXXXX	XXX.XX	XXX.XX	XXX.XX

Especificaciones

6 Tiempo de Respuesta: 1 Segundo	7 Frecuencia: 50 Por Subasta	8 Tamaño: 40 Ch x Linea
9 Medio : Disco	10 Disponibilidad : Cada vez que se desee iniciada la subasta	

Comentarios

11

Posturas.

Fecha : FEB 88 1	Pagina : 1 2
Sistema : Subasta de Cetes 3	identificacion: SUBCE-3B 4

Descripción 5

El Reporte de las Posturas debe contener: Folio, Metodo asigna Posicion, Tipo de operacion, Casa de bolsa, Emision, Valor nominal Tasa de descuento, Tasa precio y Dias reporto.

BOLSA MEXICANA DE VALORES
SUBASTA DE CETES
POSTURAS

Fecha MMM DD,YYYY

Folio	Met.	C/V	Tipo	Casa	Emision	Monto	T.D	I.P	B.R
XXX	X	(	X	XXX-XX	XXXXXX	XXXXXXX	XX.XX	XX.XX	XX.XX

Especificaciones

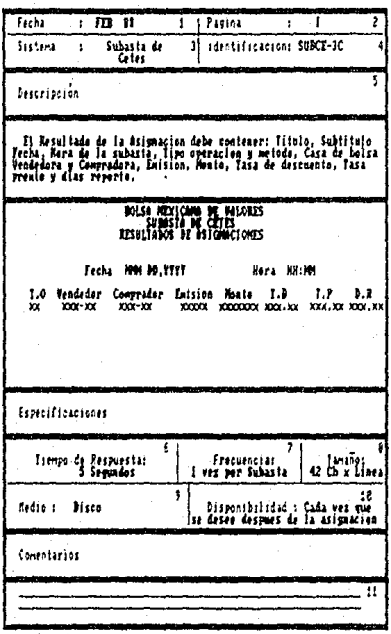
Tiempo de Respuesta: 1 Segundo 6	Frecuencia: 50 Por Subasta 7	Tamaño: 8
		60 Ch x Linea

Medio : Disco 9	Disponibilidad : Cada vez que se desee iniciada la subasta 10

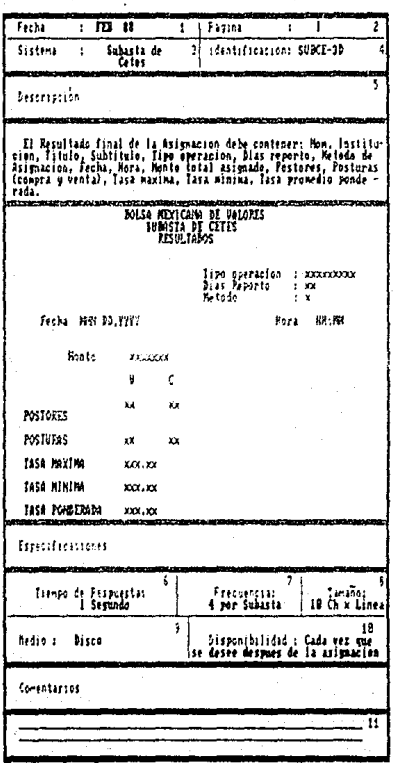
Comentarios

11

Asignación.


Fecha : FEB 88 1
Pagina : 1 2
Sistema : Subasta de Cetes 3
Identificacion: SUBCE-IC 4
Descripcion 5
El Resultado de la Asignacion debe contener: Titulo, Subtitulo Fecha, Hora de la subasta, Tipo operacion y metodo, Casa de Bolsa Vendedora y Compradora, Emision, Monto, Tasa de descuento, Tasa precio y dias reporto.
BOLSA MEXICANA DE VALORES
SUBASTA DE CETES
RESULTADOS DE ASIGNACIONES
Fecha MMM DD,YYYY Hora HH:MM
T.O Vendedor Comprador Emision Monto I.D I.P D.R
XX XXX-XX XXX-XX XXXXX XXXXXXX XXX.XX XXX.XX XXX.XX
Especificaciones
6
Tiempo de Respuesta: 5 Segundos
7
Frecuencia: 1 vez por Subasta
8
Lenght: 42 Ch x Linea
9
Medio : Disco
10
Disponibilidad : Cada vez que se desee despues de la asignacion
Comentarios
11

Resultados Finales.



Fecha : FEB 88 1	Pagina : 1 2
Sistema : Subasta de Cetes 3	Identificacion: SUBCE-3D 4

Descripción 5

El Resultado final de la Asignacion debe contener: Hora, Institucion, Titulo, Subtitulo, Tipo operacion, Dias reporto, Metoda de Asignacion, Fecha, Hora, Monto total asignado, Postores, Posturas (compra y venta), Tasa maxima, Tasa minima, Tasa promedio ponderada.

BOLSA MEXICANA DE VALORES
SUBASTA DE CETES
RESULTADOS

Tipo operacion : xxxxxxxx
Dias Reporto : xx
Metodo : x

Fecha MMM DD,YYYY Hora HH:MM

Monto	xx,xxx,xxx	
	V	C
POSTORES	xx	xx
POSTURAS	xx	xx
TASA MAXIMA	xxx.xx	
TASA MINIMA	xxx.xx	
TASA PONDERADA	xxx.xx	

Especificaciones

Tiempo de Respuestas 6	Frecuencia: 7	Tamaño: 8
1 Segundo	4 por Subasta	18 Ch x Linea

Medio : Disco 9	Disponibilidad : Cada vez que se desee despues de la asignacion 10

Comentarios

11

Descripción De Entidades.

Fecha : FEB 88	1	Pagina : 1	2
Sistema : Subasta de Cetes	3	identificacion:	ICONV-1 4

Descripción **convocatorias** 5

Contiene descripcion de la convocatoria para efectuar la subasta.

Elementos de datos

IDENTIFICADOR 7	NOMBRE 7	LONG. 8	TIPO 9 A/N	1/M	FORMATO 10	VALIDACION 11
Folio	Folio	3	N	1	xxx	unico 1-100 Contado 401-500 Con RD 201-300 Repor. 601-700 Rp. RD
Metodo	Met-asig	1	N	1	x	M Multiple U Unica E Exclusiva
Posicion	Pos	1	A	M	x	C Compra V Venta
Tipo opera - cion	Tip-oper	1	A	M	x	1 Contado 2 Cont. RD 3 Reporto 4 Reporto RD
Casa Bolsa Emision	C-bolsa Emis	6 5	A A	M M	xxx-xx xxxxx	Debe existir En Con. debe existir y ser igual. En Vta. Rep. debe existir y puede ser di - ferente. En Compra Rep. puede no existir.
Valor Nominal	Val-nom	7	A	M	xxxxxxx	No puede ser 0
Tasa descuento	T-desc	6	A	M	xxx.xx	En Cont () 0 En Vta Rep) 0 En Com Rep = 0
Tasa Premio	T-premio	6	A	M	xxx.xx	En Cont. = 0 En Rep. () 0
Dias reporto	Dias-rep	2	A	M	xx	En Cont. = 0 En Rep. = al convocador.

Especificaciones

Tiempo de respuesta: 12	Tamano: 13	Frecuencia: 14
1 Segundo	41 caracteres	Cada vez que el usuario lo requiera.

Comentarios 15.

Despues de haber capturado la rep. convocatoria se verificara que el metodo y la posicion siempre sea igual a la de la primera, esto para cada tipo de operacion.

Fecha : FEB 88 1		Pagina : 1 2	
Sistema : Subasta de Cetes 3		identificacion: IPOST-1 4	

Descripcion **posturas**

Contiene descripcion de las posturas para efectuar la subasta. 5

Elementos de datos

IDENTIFICADOR	NOMBRE 7	LONG. 8	TIPO 9 1/a	TIPO 9 1/m	FORMATO 10	VALIDACION 11
Folio	Folio	3	N	1	xxx	unico 101-200 Cont. 501-600 Con MD 301-400 Repor. 701-800 Re. MD
Metodo	Met-asig	1	N	1	x	M Multiple U Unica E Exclusiva
Posicion	Pos	1	A	M	x	C Compra V Venta
Tipo opera - cion	Tip-oper	1	A	M	x	1 Contado 2 Cont. MD 3 Reporto 4 Reporto MD
Casa Bolsa	C-bolsa	6	A	M	xxx-xx	Debe existir
Emision	Emis	5	A	M	xxxxx	En Con. debe existir y ser igual. En Vta. Rep. puede no existir. En Compra Rep. debe de existir, puede ser diferente.
Valor Nominal	Val-nom	7	A	M	xxxxxxx	No puede ser 0
Tasa descuento	T-desc	6	A	M	xxx.xx	En Cont () 0 En Vta Rep) 0 En Com Rep = 0
Tasa Premio	T-premio	6	A	M	xxx.xx	En Cont. = 0 En Rep. () 0
Dias reporto	Dias-rep	2	A	M	xx	En Cont. = 0 En Rep. = al convocador.

Especificaciones

Tiempo de respuesta: 12 1 Segundo	Tamano: 13 41 caracteres	Frecuencia: 14 Cada vez que el usuario lo requiera.

Comentarios

Despues de haber capturado la 1cr. postura verificara que el metodo y la posicion siempre sea igual a la de la primera, esto para cada tipo de operacion. 15

Fecha	:	FEB 88	1	Pagina	:	1	2
Sistema	:	Subasta de Cetes	3	Identificacion:		ICASA-1	4

Descripción: **casas de bolsa**

5

Contiene la descripcion de todas las casas de bolsa autorizadas a operar en Bolsa Mexicana de Valores.

Elementos de datos

IDENTIFICACION	NOMBRE 7	LONG. 8	TIPO 9 (A/N S/N)	FORMATO 10	VALIDACION 11
CLAVE NOMBRE CASA	CLAVE NOM-CASA	6 35	X 1 A 1	xxx-xx xxxxxxxx	

Especificaciones

Tiempo de respuesta: 12	Tamaño: 13	Frecuencia: 14
1 Segundo	41 caracteres	Cada vez que el usuario lo requiera.

Comentarios

15

Despues de haber capturado la casa de bolsa en las convocatorias se verifica que efectivamente dicha casa exista.

Fecha : FEB 88	1	Pagina : 1	2
Sistema : Subasta de Cetes	3	Identificacion: IPOLI-1	4

Descripción

politicas

5

Contiene la descripcion de las politicas para captura de las convocatorias y posturas.

Elementos de datos

IDENTIFICADOR	NOMBRE 7	LONG. 8	TIPO 9 I/a	I/m	FORMATO 10	VALIDACION 11
TIPO OPERACION	TPO-OPER	1	X	1	x	1 Contado 2 Contado M.D. 3 Reporto. 4 Reporto M.D.
Metodo	Met-asig	1	X	1	x	M Multiple. U Unica. E Exclusiva.
Posicion	Pos	1	X	1	x	C Compra. V Venta.
Emision	Emis	5	A	M	xxxxx	
Dias Reporto	Dias-rep	2	A	M	xx	
Folio Convoca	Verifol	3	A	1	xxx	
Casa de Bolsa	C-bolsa	5	A	M	xxxxx	

Especificaciones

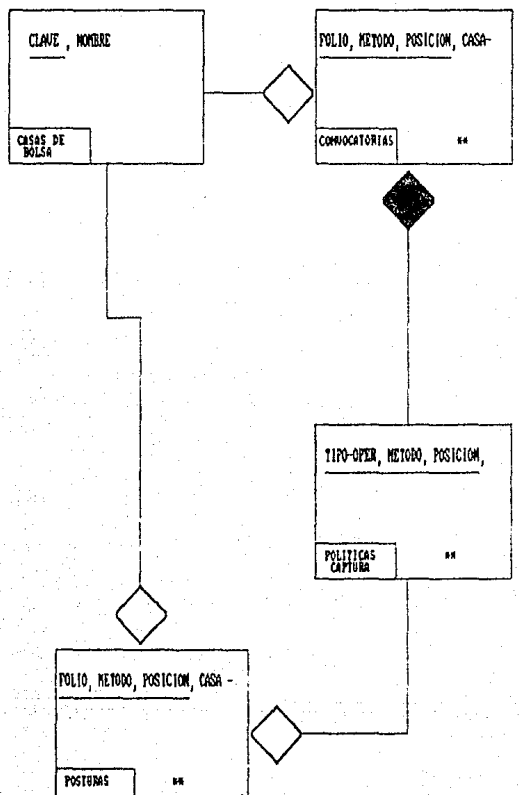
Tiempo de respuesta: 12	Tamano: 13	Frecuencia: 14
1 Segundo	18 caracteres	Cada vez que se inicie una subasta.

Comentarios

15

Contendra como maximo 4 registros uno para cada tipo de operacion todos los datos son creados durante la captura de la convocatoria.

Modelo De Información.



NOTA:

** Los atributos faltantes en la entidad se encuentran detallados a continuación.

CONVOCATORIAS

(FOLIO, METODO, POSICION, CASA BOLSA, EMISION, VALOR NOMINAL, TASA DESCUENTO, TASA PREMIO, DIAS REPORTO)

POSTURAS

(FOLIO, METODO, POSICION, CASA BOLSA, EMISION, VALOR NOMINAL, TASA DESCUENTO, TASA PREMIO, DIAS REPORTO)

POLITICAS

(TIPO-OPER, METODO, POSICION, EMISION, DIAS REPORTO, FOLIO CONVOCA, CASA BOLSA)

CASAS DE BOLSA

(CLAVE, NOMBRE)

Diccionario De Datos.

Fecha : FEB 88 Pagina : 1
Sistema : Subasta de Cetes

Nombre del Dato	Alias	Rango	Longitud
Casa de bolsa	C-bolsa		5
Dias reporto	Dias-rep		2
Emision	Emis		2
Folio	Folio	Continuo	3
Metodo de asignacion	Met-asig		1
Posicion	Pos		1
Postores	Postores		2
Posturas	Posturas		2
Tasa descuento	T-desc		6
Tasa maxima	T-max		6
Tasa minima	T-min		6
Tasa premio	T-premio		6
Tasa promedio	T-prom		6
Tipo de operacion	Tip-oper		1
Valor nominal	Val-nom		7

DESCRIPCION DE DIALOGOS Y MENUS.

Para los diálogos que el usuario necesita algunos se desplegarán una o más veces, los menus servirán para navegar por las pantallas.

Cuando es requerido iniciar una nueva subasta y/o una nueva asignación se desplegarán los mensajes:

Password para iniciar una nueva subasta:
Password para hacer una nueva asignación:

Cuando esté todo bien se desplegará el mensaje de L I S T O sino, se despliega Password erroneo.

En las convocatorias se podrá elegir entre darlas de Alta, Baja, Reportes (por Pantalla o por Impresora), Hora de cierre. En las posturas se darán de Alta, Baja, Reportes (por Pantalla o por Impresora).

Para llevar a cabo el proceso de asignación se debe dar un password si éste es correcto se despliega el mensaje de que se está llevando a cabo la asignación o bien se despliega password equivocado. Los resultados serán, de la asignación o de los resultados finales ambos por Impresora o Pantalla.

VI.2 D I S E N O.

VI.2.1 TAREA NUMERO UNO.

Diseno De Archivo Y/O Base De Datos.

Basandonos en el modelo de información la base de datos esta constituida por las siguientes tablas:

* DATCONV
* DATPOST
* ASI
* RESULT
* CLVSUB
*LIMITES

La tabla "DATCONV" será accesada por el módulo de convocatorias y se podrán realizar operaciones de Altas y Bajas de registros. Esta tabla contendrá tres campos llave, serán el número de folio, el método de asignación y la posición.

La tabla "DATPOST" será accesada por el módulo de posturas y se podran llevar a cabo operaciones de Altas y Bajas de registros. Los campos llave de ésta tabla serán, el número de folio, el método de asignación y la posición.

La tabla "ASI" será accesada por el módulo de asignación, sus campos llave serán el número de folio y el método de asignación.

La tabla "RESULT" será accesada por el módulo de reportes, al igual que las tablas anteriores sus campos llave serán el número de folio y el método de asignación.

La tabla "CLVSUB" será accesada por el módulo de convocatorias, en el momento que se de la primer alta de la convocatoria.

la tabla "LIMITES" será accesada por el módulo de convocatorias, cuando se de la hora de cierre de la subasta.

Tanto la tabla de "CLVSUB" como la de "LIMITES" y "CASAS" son auxiliares.

A continuación se decribirán cada uno de los registros que constituyen las tablas que serán utilizadas en el sistema.

BOLSA MEXICANA DE VALORES, S.A DE C.V.

GERENCIA DE AUTOMATIZACION OPERATIVA
DESCRIPCION DE ARCHIVOS

SISTEMA: SUBASTA DE CETES HOJA:1
ARCHIVO: SATCONV

NOMBRE EXT.	NOMBRE INT.	LONG.	TIPO	FORMATO	VALIDACIONES
Folio	Folio	3	Entero	999	Unico 1 - 100 Contado 201 - 300 Reporto 401 - 500 Contado M.D. 601 - 700 Reporto M.D.
Metodo	Met_asig	1	Char	A	M Tasa Multiple U Tasa Unica E Tasa Exclusiva
Posicion	Pos	1	Char	A	C Compra V Venta
Tipo de operacion	Tip_oper	1	Char	9	1 Contado 2 Contado M.D. 3 Reporto 4 Reporto M.D.
Casa de bolsa	C_bolsa	6	Char	999-99	DEBE EXISTIR
Emision	Emis	5	Entero	99999	En Contado es igual y debe de existir. En Venta Reporto debe de existir y puede ser diferente En Compra Reporto puede no existir.
Valor nominal	Monto	7	Entero	9999999	No debe ser menor a cero.
Tasa de descuento	T_desc	6	Decimal	999.99	Tasa_desc) 0 En Contado () 0 En Venta Reporto () 0 En Compra Reporto = 0
Tasa premio	T_premio	6	Decimal	999.99	Tasa_premio) 0 En Contado = 0 En Venta Reporto () 0 En Compra Reporto () 0
Dias reporto	Dias_rep	2	Entero	99	Dias Reporto) 0 En Contado = 0 En Reporto igual conversador

BOLSA MEXICANA DE VALORES, S.A DE C.V.

GERENCIA DE AUTOMATIZACION OPERATIVA
DESCRIPCION DE ARCHIVOS

SISTEMA: SUBASTA DE CETES
ARCHIVO: SATPOST

HOJA:1

NOMBRE EXT.	NOMBRE INT.	LONG.	TIPO	FORMATO	VALIDACIONES
Folio	Folio	3	Entero	999	Unico 1 - 100 Contado 201 - 300 Reporto 401 - 500 Contado M.D. 601 - 700 Reporto M.D.
Metodo	Met_asig	1	Char	A	M Tasa Multiple U Tasa Unica I Tasa Exclusiva
Posicion	Pos	1	Char	A	C Compra V Venta
Tipo de operacion	Tip_oper	1	Char	9	1 Contado 2 Contado M.D. 3 Reporto 4 Reporto M.D.
Casa de bolsa	C_bolsa	6	Char	AAA-AA	DEBE EXISTIR
Emision	Emis	5	Entero	99999	En Contado es igual y debe de existir. En Compra Reporto debe de existir y puede ser diferente En Venta Reporto puede no existir.
Valor nominal	Val-nom	7	Entero	9999999	No debe ser menor a cero.
Tasa de descuento	T_desc	6	Decimal	999.99	Tasa_desc) 0 En Contado () 0 En Venta Reporto () 0 En Compra Reporto = 0
Tasa premio	T_premio	6	Decimal	999.99	Tasa_premio) 0 En Contado = 0 En Venta Reporto () 0 En Compra Reporto () 0
Dias reporto	Dias_rep	2	Entero	99	Dias Reporto) 0 En Contado = 0 En Reporto () 0

BOLSA MEXICANA DE VALORES, S.A DE C.V.

GERENCIA DE AUTOMATIZACION OPERATIVA

DESCRIPCION DE ARCHIVOS

SISTEMA: SUBASTA DE CETES HOJA:1
ARCHIVO: ASI

NOMBRE EXT.	NOMBRE INT.	LONG.	TIPO	FORMATO	VALIDACIONES
Folio	Folio	3	Entero	999	Unico 1 - 199 Contado 201 - 399 Reporto 401 - 599 Contado M.D. 601 - 799 Reporto M.D.
Metodo	Met_asis	1	Char	X	M Tasa Multiple U Tasa Unica E Tasa Exclusiva
Vendedor	Vendedor	6	Char	XXX-XX	
Comprador	Comprador	6	Char	XXX-XX	
Emision	Emision	5	Entero	99999	En Contado es igual y debe de existir. En Venta Reporto debe de existir y puede ser diferente En Compra Reporto puede no existir.
Valor nominal	Monto	7	Entero	9999999	No debe ser menor a cero.
Tasa de descuento	Tasa_desc	6	Decimal	999.99	Tasa_desc) 0 En Contado () 0 En Venta Reporto () 0 En Compra Reporto = 0
Tasa premio	Tasa_premio	6	Decimal	999.99	Tasa_premio) 0 En Contado = 0 En Venta Reporto () 0 En Compra Reporto () 0
Dias reporto	Dias_rep	2	Entero	99	Dias Reporto) 0 En Contado = 0 En Reporto () 0

BOLSA MEXICANA DE VALORES, S.A DE C.V.

GERENCIA DE AUTOMATIZACION OPERATIVA
DESCRIPCION DE ARCHIVOS

SISTEMA: SUBASTA DE CETES HOJA:1
ARCHIVO: RESULT

NOMBRE EXT.	NOMBRE INT.	LONG.	TIPO	FORMATO	VALIDACIONES
Postores	Postores	3	Entero	000	
Metodo	Metodo_asig	1	Char	X	
Posturas	Posturas	3	Entero	000	
Tipo de operacion	Tipo_oper	1	Char	X	
Tasa maxima	Tasa_max	6	Decimal	999.99	
Tasa minima	Tasa_min	6	Decimal	999.99	
Monto	Monto	7	Entero	9999999	
Tasa de promedio	Tasa_pond	6	Decimal	999.99	
Dias reporto	Dias_rep	2	Entero	99	

BOLSA MEXICANA DE VALORES, S.A DE C.V.

GERENCIA DE AUTOMATIZACION OPERATIVA
DESCRIPCION DE ARCHIVOS

SISTEMA: SUBASTA DE CETES HOJA:1
ARCHIVO: CLAVSUB

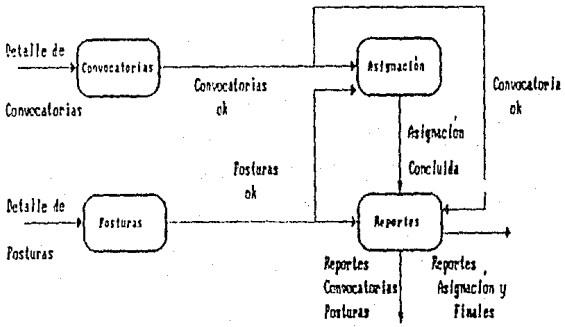
NOMBRE EXT.	NOMBRE INT.	LONG.	TIPO	FORMATO	VALIDACIONES
Tipo de operacion	Tip-oper	1	Char	###	
Metodo	Met_asig	1	Char	A	
Posicion	Pos	1	Char	A	
Emision	Emis	5	Entero	#####	
Dias reporto	Dias-rep	2	Entero	##	
Folio convocador	Folio	3	Entero	###	
Casa de bolsa	C_bolsa	6	Char	###-###	
Tasa de descuento	T_desc	6	Decimal	##9.##	

VI.2.2 TAREA NUMERO DOS.

Describir Las Funciones Del Sistema.

El diseno del sistema propuesto para dar solución a la problematica existente, se base en cuatro funciones:

- Convocatorias.
- Posturas.
- Asignación.
- Reportes.



CONVOCATORIAS. Captura de convocatorias se validará cada uno de los campos de la mascarilla de captura dependiendo del tipo de operación y valor del que se trate. Una convocatoria podrá ser dada de baja dando como datos llave el número de folio, el método de asignación, la posición y el tipo de operación.

POSTURAS. Captura de posturas se validará cada uno de los campos de la mascarilla de captura dependiendo del tipo de operación y valor del que se trate. Una postura podrá ser dada de baja dando como datos llave el número de folio, el método de asignación, la posición y el tipo de operación.

ASIGNACION. La asignación se llevará a cabo dependiendo del método que se halla seleccionado.

REPORTES. Se contarán con reportes de convocatorias, posturas, asignación , resultados de asignación y resultados finales dichos reportes se podrán consultar por medio de impresora o bien por pantalla.

VI.2.3 TAREA NUMERO TRES.

Diseno De Las Salidas Del Sistema.

Los reportes de convocatorias y posturas deberán mostrar:

Folio	Folio correspondiente de la convocatoria.
Met.	Método por el cual se llevará a cabo la asignación
C/V	Posición de la convocatoria o postura. C.- Compra. V.- Venta.
Tipo	Tipo de operación. 1.- Contado. 2.- Contado Mismo Dia. 3.- Reporto. 4.- Reporto Mismo Dia.
Casa	Número de la casa de bolsa convocadora o de la contra parte.
Emision	Número de emisión del cete.
Monto	Monto de los cetes sobre los cuales se hará la asignación.
T.D.	Tasa de descuento.
T.P.	Tasa premio.
D.R.	Dias reporto.

Los reportes de asignación deberán mostrar:

Fecha	Fecha de la subasta.
Hora	Hora límite de la subasta.
T.O.	Tipo de operación y método de la subasta.
Vendedor	Casa de bolsa vendedora.
Comprador	Casa de bolsa compradora.
Emisión	Número de la emisión del cete asignado.
Monto	Monto de la asignación en miles de pesos.

S U B A S T A D E C E T E S

CONVOCATORIAS

Folio	Met.	C/V	Tipo Casa	Emision	Monto	T.D	T.P	D.R

S U B A S T A D E C E T E S

POSTURAS

Folio	Met.	C/V	Tipo Casa	Emision	Monto	T.D	T.P	D.R

S U B A S T A D E C E T E S

RESULTADOS DE ASIGNACIONES

Fecha: Hora:

T.O	Vendedor	Comprador	Emision	Monto	T.D	T.P	D.R

B O L S A M E X I C A N A D E V A L O R E S

S U B A S T A D E C E T E S

RESULTADOS

Tipo Operacion:
Dias Reporto:
Metodo:

Hora:

Fecha:

MONTO	U	C
POSTORES		
POSTURAS		
TASA MAXIMA		
TASA MINIMA		
TASA PONDERADA		
TASA UNICA		

T.D.	Tasa de descuento a la que se asigno.
T.P.	Tasa premio a la que se asigno.
D.R.	Dias reporto.

El reporte de resultados finales deberá mostrar:

Tipo	Tipo de operación de la asignación.
Dias	Dias reporto a los cuales se asigno.
Met	Método de la asignación.
Fecha	Fecha de la subasta efectuada.
Hora	Hora límite de la subasta.
Monto	Monto total de la asignación.
Postores	Total de postores asignados.
Posturas	Total de posturas asignadas.
T.Max.	Tasa máxima de la asignación.
T.Min.	Tasa mínima de la asignación.
T.Pon.	Tasa promedio ponderada de la asignación.

VI.2.4 TAREA NUMERO CUATRO.

Diseno De Las Entradas Del Sistema.

Las entradas para convocatorias y posturas serán:

Folio	Folio correspondiente de la convocatoria o postura.
Met.	Método por el cual se llevará a cabo la asignación.
POS.	Posición de la convocatoria o postura. C.- Compra. V.- Venta.
Tipo	Tipo de operación. 1.- Contado. 2.- Contado Mismo Dia. 3.- Reporto. 4.- Reporto Mismo Dia.
Casa	Número de la casa de bolsa convocadora o de la contra parte.
Emision	Número de emisión del cete.
Valor	Valor nominal de los cetes sobre los cuales se hará la asignación.
T.D.	Tasa de descuento.
T.P.	Tasa premio.
D.R.	Dias reporto.

B O L S A M E X I C A N A D E V A L O R E S

CONVOCATORIA SUBASTA DE CETES

Folio	: [f00]	Metodo asignacion	: [a]
Posicion (C/V)	: [b]	Tipo operacion	: [c]
Casa de bolsa	: [f001]	Emision	: [f002]
Valor nominal	: [f003]	Tasa descuento	: [f004]
Tasa premio	: [f005]	Dias reporto	: [f6]

B O L S A M E X I C A N A D E V A L O R E S

POSTURAS SUBASTA DE CETES

Folio	: [f00]	Metodo asignacion	: [a]
Posicion (C/V)	: [b]	Tipo operacion	: [c]
Casa de bolsa	: [f001]	Emision	: [f002]
Valor nominal	: [f003]	Tasa descuento	: [f004]
Tasa premio	: [f005]	Dias reporto	: [f6]

VI.2.5 TAREA NUMERO CINCO.

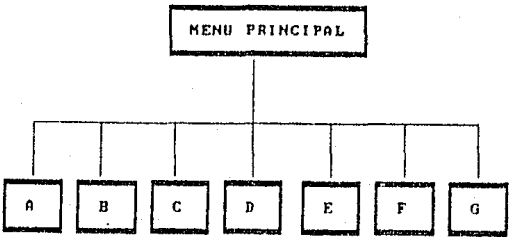
Diseno De Las Interfases Del Sistema.

Por las caracteristicas del sistema esta tarea no se aplica.

VI.2.6 TAREA NUMERO SEIS.

Diseno De La Estructura Del Software.

Para tener una visión más amplia del sistema a continuación se muestra la Carta Estructurada.



EL MODULO DE NUEVA SUBASTA, es un proceso que se encarga de inicializar la subasta, su "Fanout" y el "Fanin" es de uno.

EL MODULO DE POLITICAS, es una rutina auxiliar para el módulo de convocatorias y posturas, contiene las características de la subasta su "Fanout" y el "Fanin" es de uno.

EL MODULO DE CASAS DE BOLSA, es una rutina auxiliar para los módulos de convocatorias y posturas contiene información de las casas de bolsa autorizadas a operar en la Bolsa Mexicana De Valores, su "Fanout" y el "Fanin" son de uno.

EL MODULO DE CONVOCATORIAS, es el proceso que realiza, el alta, baja y reporte de convocatorias, así como la hora de cierre de la subasta, el "Fanout" es de dos y el "Fanin" es de uno.

EL MODULO DE POSTURAS, es el proceso que realiza el alta, baja y reporte de posturas, el "Fanout" es de dos y el "Fanin" es de uno.

La cohesión de los módulos de Convocatorias y Posturas es lógica, cada uno de estos dos módulos está constituido de submódulos de propósito especial.

EL MODULO DE ASIGNACION, es el proceso que lleva a cabo la asignación de montos para los métodos de asignación existentes, su "Fanout" es de uno y el "Fanin" es de dos cuenta con una cohesión funcional.

EL MODULO DE RESULTADOS, es el proceso que realiza los reportes requeridos de la asignación de montos, así como los resultados finales de la subasta, su "Fanout" es de dos y el "Fanin" es de uno, tiene una cohesión funcional.

Todos los módulos de la subasta de cetes son secuenciales y cuentan con independencia de módulos.

Con respecto al acoplamiento; el módulo de Convocatorias y Posturas tienen un acoplamiento por zonas de datos que estan contenidas en el módulo de Políticas. El módulo de Asignación y el Resultados tienen un acoplamiento por datos.

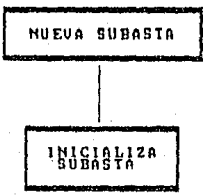
VI.2.7 TAREA NUMERO SIETE.

Diálogos.

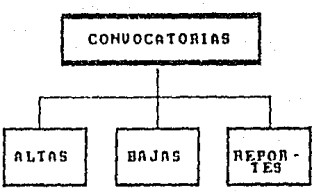
Por las caracteristicas del sistema esta tarea no se aplica.

VI.2.8 TAREA NUMERO OCHO.

Diseno De La Lógica De Procesamiento Del Sistema.



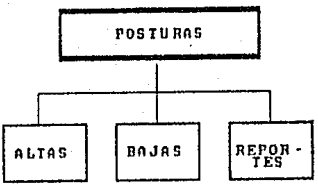
INICIALIZA SUBASTA .- Proceso que limpia las tablas de "datconv", "datpost", "clvsub" "result' y "asi" para poder iniciar la subasta una vez que el password fue correcto.



ALTAS .- Proceso para dar de alta una convocatoria, con todas las validaciones necesarias además almacena la hora de cierre de la subasta. Para la primera convocatoria del mismo tipo de operación se guardan las políticas de la subasta en la tabla "clvsub", para cada convocatoria que se capture se almacenará en la tabla de "datconv".

BAJAS .- Proceso para dar de baja una convocatoria.

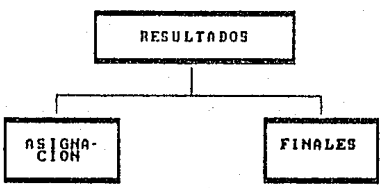
REPORTES .- Proceso que genera los reportes de las convocatorias por pantalla y/o impresora.



ALTAS .- Proceso para dar de alta posturas, considerando todas las validaciones necesarias, para cada postura que se capture se almacenará en la tabla de "datpost".

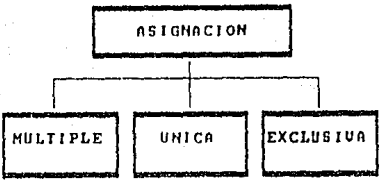
BAJAS .- Proceso para dar de baja una postura.

REPORTES .- Proceso que genera los reportes de las posturas por pantalla y/o por impresora.



RESULTADOS ASIGNACION .- Proceso que genera el reporte de los resultados de todas las asignaciones concluidas, por pantalla y/o por impresora, extre la información de la tabla de "asi".

FINALES .- Proceso que genera los resultados finales de la subasta, dichos resultados se almacenan en la tabla de "result".



MULTIPLE .- Una vez que se dio el password para asignar se checa si procede la asignación, se ordenan las tablas de "datconv" y " datpost" para proceder con la asignación de montos, los datos de dicha asignación son almacenados en la tabla de "asi".

UNICA .- Si es correcto el password para la asignación checar si la asignación procede, se obtiene la tasa única utilizando las tablas de "datconv" y "datpost" para proceder con la asignación de montos, los datos de la asignación son almacenados en la tabla de "asi".

EXCLUSIVA .- Si el password es correcto verificar si se lleva a cabo la asignación, ordenar las tablas de "datconv" y "datpost", se obtiene el monto total de las convocatorias y posturas para poder llevar a cabo la asignación de montos que se almacenan en la tabla de "asi".

VI.2.9 TAREA NUMERO NUEVE.

Construcción Del Prototipo.

Para iniciar con la construcción del prototipo se tienen nuevamente presentes los requerimientos del usuario, así como el objetivo del sistema.

Lo que se le presenta al usuario como, la primera versión del prototipo son las pantallas de captura para los datos de entrada, así como los reportes de las convocatorias y las posturas, ya que son los elementos principales para llevar a cabo la subasta. Todos los datos son incluidos en el prototipo y se considera que la base de datos ya se ha generado.

La característica principal del prototipo es, para mostrar al usuario la forma en que serán automatizadas las principales funciones, que con anterioridad fueron probadas por la gente de sistemas. La primer versión del prototipo fue pensada para que al usuario precise que información le hace falta y los cuestionamientos que le pudiesen surgir sobre el funcionamiento del sistema.

El desarrollo del prototipo fue rápido debido a que se utilizaron las facilidades y herramientas de los lenguajes de cuarta generación. Una vez que el prototipo fue evaluado por el usuario y modificado por la gente de sistemas, se presentó la segunda versión del prototipo que contempla la asignación de montos y reportes de dicha asignación así como los resultados finales de la subasta.

Obtenida la aprobación por el usuario se manejará la asignación de montos para el método de tasa exclusiva y única como otro prototipo que será desarrollado posteriormente. La asignación de montos para el método de tasa múltiple es el sistema que queda concluido.

VI.2.10 TAREA NUMERO DIEZ.

Preparación De Las Especificaciones De Programas.

Módulo Nueva Subasta.

La función que desempeña este módulo es, borrar todos los datos existentes de subastas anteriores de las tablas datconv, datpost, clvsub, así y result, para que lo anterior proceda es necesario un password.

PSEUDOCODIGO

```
PROCEDIMIENTO PRINCIPAL
        Begin
         Despliega mensaje para password
           Si password ok
             Borra datos de las tablas
                Datpost
                Datconv
                Asi
                Result
                Clvsub
                Limites
           Si password erroneo
             Despliega mensaje que es erroneo
        End
```

Módulo Convocatorias.

Su función es dar de alta, baja y reportes de las convocatorias de las subastas por impresora y/o pantalla.

PSEUDOCODIGO

```
PROCEDIMIENTO PRINCIPAL
  Begin
   Despliege de menu
  End

ALTAS
  Begin
   Despliega forma de captura
   Valida datos
   Si es la 1er. llamada módulo de politicas
   Guarda datos en datconv
  End
BAJAS
  Begin
   Despliega forma de captura
   Valida existencia del registro
   Corrobora baja
   Borra registro de datconv
    Si folio que marcaba politicas fue borrado
       Llamada módulo de politicas
   Despliega mensaje folio dado de baja
  End
```

```
REPORTES
  Begin
   Despliega menu
   PANTALLA
     Limpia pantalla
     Despliega encabezado
     Despliega datos de datconv
     Mensaje si hay mas información o si es toda
   IMPRESORA
     Inicializa reporte
     Manda a impresora todos los datos de datconv
     Despliega mensaje reporte impreso
  End
```

Módulo Posturas.

Su función es dar de alta, baja y reportes por pantalla y/o impresora de las posturas de la subasta.

PSEUDOCODIGO

```
PROCEDIMIENTO PRINCIPAL
  Begin
   Despliege de menu
  End
ALTAS
  Begin
   Despliega forma de captura
   Valida datos
   Lamada a checar politicas de la subasta
   Guarda datos en datpost
  End
BAJAS
  Begin
   Despliega forma de captura
   Valida existencia del registro
   Corrobora baja
   Borra registro de datpost
   Despliega mensaje folio dado de baja
  End
```

```
REPORTES
  Begin
   Despliega menu
   PANTALLA
     Limpia pantalla
     Despliega encabezado
     Despliega datos de datpost
     Mensaje si hay mas información o si es toda
   IMPRESORA
     Inicializa reporte
     Manda a impresora todos los datos de datpost
     Despliega mensaje reporte impreso
  End
```

Módulo De Casas De Bolsa.

Su función es verificar que la casa de bolsa que desee entrar a la subasta este autorizada a operar.

Módulo De Políticas.

La función que desempeña es almacenar en la tabla de clvsub las características de la subasta una vez que la primer convocatoria ha sido capturada y valida la contra parte de los convocadores.

Los módulos de políticas y casas de bolsa se implementarán como subfunciones ya que son usados por varios módulos.

Módulo De Asignación.

Su función es procesar los datos de los convocadores y postores para realizar la asignación de montos estos datos son almacenados en la tabla "asi" y una vez terminada la asignación contabiliza lo asignado para que sea almacenado en la tabla de "result".

PSEUDOCODIGO

```
PROCEDIMIENTO PRINCIPAL
  Begin
   Ordena tablas de datconv y datpost
   Inicia proceso de asignación
   Guarda datos en asi
   Contabiliza datos de asi para obtener finales
   Guarda datos en result
  End
```

Módulo De Resultados.

La función que realiza es la presentación de los resultados de los montos asignados así como, los resultados finales por pantalla y/o por impresora.

```
PSEUDOCODIGO

     PROCEDIMIENTO PRINCIPAL
       Begin
         Despliega menu
       End

     RESULTADOS ASIGNACION
       Begin
         PANTALLA
          Limpia pantalla
          Despliega encabezado
          Despliega datos de asi
          Mensaje si hay mas información o si es toda
         IMPRESORA
          Inicializa reporte
          Manda a impresora todos los datos de asi
          Despliega mensaje reporte impreso
       End
   RESULTADOS FINALES
       Begin
         PANTALLA
          Limpia pantalla
          Despliega encabezado
          Despliega datos de result
          Mensaje si hay mas información o si es toda
         IMPRESORA
          Inicializa reporte
          Manda a impresora todos los datos de result
          Despliega mensaje reporte impreso
        End
```

VI.2.11 TAREA NUMERO ONCE.

Diseno Del Proceso De Control, Seguridad y Respaldo.

La seguridad del sistema está manejada por password's, que deben ser confidenciales solamente algunos usuarios los conocerán y así se podrán deslindar las responsabilidades.

Dentro de las politicas de respaldo se determino realizar un respaldo total de la base de datos de producción dos veces por semana así como los discos en los cuales resida el sistema de producción y el software de desarrollo. El respaldo se lleva a cabo en cintas.

Durante la etapa de desarrollo y a nivel programación se incluyen instrucciones que nos ayudan a conservar la integridad de la base de datos si se llega a presentar una falla en el sistema, estas instrucciónes son: BEGIN WORK, COMMIT WORK y ROLLBACK WORK.

Si el sistema es aceptado por el usuario se hace un respaldo integro de todo el sistema por si es necesaria su recuperación para poderla llevar a cabo.

VI.2.12 TAREA NUMERO DOCE.

Preparación Del Plan De Entrenamiento.

El entrenamiento del sistema tiene dos fases una, inmediatamente después de que el prototipo fue aceptado y la última cuando el sistema ha sido instalado, la capacitación es enfocada a usuarios y a operadores del sistema.

Una vez que se finalizo el prototipo se podrán efectuar pruebas para así determinar el grado de avance del sistema. Cuando culmina el desarrollo se realizan las pruebas finales o la prueba en paralelo del sistema.

A continuación se presenta una parte de lo que constituye la programación del sistema "Subasta Múltiples de Certificados de la Tesoreria de la Federación".

Los módulos de convocatorias y posturas tienen un principio de funcionamiento muy similar sólo en lo que cambian es en el archivo que usan, es por ello que solamente se presenta el módulo de convocatorias.

Este módulo es el encargado de desplegar el menu para el manejo de las convocatorias así como la pantalla de captura de los datos de las convocatorias ya sea para dar una alta o bien si se desea una baja, contemplando sus respectivas validaciónes, también se incluye en este módulo el manejo de los reportes por impresora o bien por pantalla.

El segundo programa presentado es el que lleva a cabo la asignación de montos por el método de tasa múltiple el cual trabaja principalmente con los archivo de las convocatorias y posturas.

Los módulos restantes son muy similares en cuanto a su estructura se refiere.

```
database sub_ce

globals

      define dummy,answer        char (1),
             p_result             array(4) of record
                                    monto,
                                    postor,
                                    postura   integer,
                                    tunica    decimal (6,2)
                                  end record
end globals

# Rutina principal del sistema
# Convoca a las modulos del sistema de cetes.

main

   defer interrupt

   options message line 22,
           prompt line 21

   call display_menu ()

   menu "CETES"
         command "Nueva subasta"
                 "Entrada al modulo de nuevas subastas."
                  call nuev_sub ()
         command "Convocatorias"
                 "Entrada al modulo de convocatorias."
                  call conv01 ()
                  clear screen
         command "Posturas"
                 "Entrada al modulo de posturas."
                  call post01 ()
                  clear screen
         command "Asignacion"
                 "Entrada al modulo de asignacion."
                  call elegir_met ()
         command "Reportes"
                 "Entrada al modulo de reportes."
                  call imp_rep()
         command "Salir"
                 "Salir del sistema."
                  exit menu
    end menu

end main

function display_menu ()

        display "BOLSA MEXICANA DE VALORES"      at 6,22
        display "SUBASTA DE CETES"             at 8,27

end function
```

```
database sub_ce

function nuev_sub()

#       Funcion para limpiar archivos de convocatoria, posturas y resultados
#       , para iniciar una nueva subasta.

        define i        ,       integer,
               p                char(5),
               a                char(6)

        prompt "Deme el password para iniciar una nueva subasta : " for p

        if p = "cconv" then
          delete from datconv
          delete from datpost
          delete from clvaut
        else
         clear screen
         display "password erroneo "
         sleep 2
         clear screen
        end if

        prompt "Deme el password para hacer una nueva asignacion : " for a

        if (a = "cnueva") and (p = "cconv") then
           drop table asi
           delete from result
           delete from resconv
           create table asi
                    (vendedor      char(6) not null,
                     comprador     char(6) not null,
                     emision       integer,
                     monto         integer not null,
                     tasa_desc     decimal (6,2),
                     tasa_premio   decimal (6,2),
                     tipo_oper     char (1) not null,
                     dias_rep      integer);
        clear screen
        display "l i s t o "
        sleep 3
        clear screen
      else
         clear screen
         display "password erroneo "
         sleep 2
         clear screen
      end if

end function
```

```
        Modulo para el manejo de convocatorias

database [illegible]

globals

      define p_datconv record like datconv.*,
             p_clvasub record like clvasub.*,
             [illegible] char (1)
             [illegible] char (1)
end globals

function conv01 ()

#       Funcion principal para el manejo de convocatorias

      open [illegible] convoca from "datconv"
      menu "CONVOCATORIAS"
            command "Altas"
                  "Altas de convocatorias."
                  call add_conv ()
                  clear screen
            command "Bajas"
                  "Bajas de convocatorias."
                  call bajas_conv ()
                  clear screen
            command "Reportes"
                  "Reportes de convocatorias."
                  call repor_conv ()
                  clear screen
            command "Cierre"
                  "Dar hora de cierre de la subasta."
                  call cierre ()
                  clear screen
            command "Salir"
                  exit menu
      end menu
end function

function add_conv ()

#       Funcion [illegible] alta una convocatoria con todas las [illegible]
#       necesarias.

        define [illegible]
               first time  char (1),
               t           decimal(4,2)

        display form convoca

        let p_datconv.metodo_asig = "O"
        let p_datconv.posicion = "V"
        let p_datconv.[illegible] = "1"

        while (1 > 0)
              input by name p_datconv.[illegible]
```

```
                    p_datconv.metodo_pago,
                    p_datconv.posicion,
                    p_datconv.tipo_oper
         without defaults

          after field folio
                 if p_datconv.folio < 0 then
                    return
                 end if
                 select * from datconv
                    where p_datconv.folio = folio
                 if status != notfound then
                    error "Folio duplicado "
                    next field folio
                 end if
          after field tipo_oper
                 select * into p_clvsub.* from clvsub
                    where p_datconv.tipo_oper = clvsub.tipo_oper
                 if status = notfound then
                    let first_time = "s"
                    call verify_folio ()
                    if v = false then
                       next field tipo_oper
                    end if
                 else
                    let first_time = "n"
                    call verify_folio ()
                    if v = false then
                       next field tipo_oper
                    end if
                    if p_clvsub.metodo != p_datconv.metodo_pago then
                         error "Este tipo no checa con el metodo dado."
                         next field tipo_oper
                    end if
                    if p_clvsub.posicion != p_datconv.posicion then
                         error "Este tipo no checa con la posicion dada."
                         next field tipo_oper
                    end if
                 end if
          end input

          input by name p_datconv.casa_bolsa,
                        p_datconv.emision,
                        p_datconv.monto,
                        p_datconv.tasa_desc,
                        p_datconv.tasa_premio,
                        p_datconv.dias_reporto

          after field casa_bolsa
                 call verify_cb (...)
                 if v = false then
                    let p_datconv.casa_bolsa = ""
                    next field casa_bolsa
                 end if
                 if (p_clvsub.metodo = "E") and
                    (first_time = "s") and
```

```
                    (p_datconv.tasa_colsa != p_clvsub.tasa_colsa) then
                    error "Solo puede a [illegible] en convertidor"
                    let p_datconv.tasa_colsa = " "
                    next field tasa_colsa
                 end if
           after field emision
                 if (p_datconv.tipo_oper matches "[12]") and
                    (p_datconv.emision != p_clvsub.emision) and
                    (first_time = "n") then
                    error "La emision es erronea."
                    next field emision
                 end if
                 if (p_datconv.tipo_oper matches "[12]") and
                    (p_datconv.emision = 0) and
                    (first_time = "s") then
                    error "Debe llevar emision."
                    next field emision
                 end if
                 if (p_datconv.posicion = "V") and
                    (p_datconv.tipo_oper matches "[34]") and
                    ((p_datconv.emision = 0) or
                    (p_datconv.emision is null)) then
                    error "Falta la emision."
                    next field emision
                 end if
                 if (p_datconv.posicion = "C") and
                    (p_datconv.tipo_oper matches "[34]") and
                    (p_datconv.emision != 0) then
                    error "No debe llevar EMISION"
                 end if

           after field tasa_desc
                 case
                 when p_clvsub.metodo matches "[U,M]"
                    if (p_datconv.tipo_oper matches "[12]") and
                       (p_datconv.tasa_desc = 0) then
                       error "La tasa de descuento no puede ser 0."
                       next field tasa_desc
                    end if
                    if (p_datconv.tipo_oper matches "[34]") and
                       (p_datconv.posicion = "V")and
                       (p_datconv.tasa_desc = 0) then
                       error "La tasa de descuento no puede ser 0. "
                       next field tasa_desc
                    end if
                    if (p_datconv.tipo_oper matches "[34]") and
                       (p_datconv.posicion = "C")and
                       (p_datconv.tasa_desc != 0) and
                       (p_clvsub.tasa_desc != 0) then
                       error "La tasa de descuento debe ser 0. "
                       next field tasa_desc
                    end if
                 when p_clvsub.metodo matches "[E]"
                    if (p_datconv.tipo_oper matches "[12]") and
                       (p_datconv.tasa_desc != p_clvsub.tasa_desc) and
                       (first_time = "n") then
```

```
                error "La tasa de descuento debe ser 1)a0 "
                next field tasa_desc
              end if
              if p_datconv.tipo_oper matches "[34]" then,
                if p_datconv.emision = "Y" then
                  if p_datconv.tasa_desc = 0 then
                    error "La tasa de descuento no puede ser 0."
                    next field tasa_desc
                  else
                    select tasa_desc into t from datconv
                      where emision = p_datconv.emision
                      if t > 0 then
                        if p_datconv.tasa_desc != t then
                          error "La tasa debe ser la misma para ese
                          next field tasa_desc
                        end if
                      end if
                  end if
                else
                  if p_datconv.tasa_desc != 0 then
                    error "La tasa de descuento debe ser 0 "
                    next field tasa_desc
                  end if
                end if
              end if
            end case

        after field tasa_premio
            if (p_datconv.tipo_oper matches "[12]") and
               (p_datconv.tasa_premio != 0) then
               error "La tasa premio debe ser 0."
               next field tasa_premio
            end if
            case
            when p_clvsub.metodo matches "[U,M]"
               if (p_datconv.tipo_oper matches "[34]") and
                  (p_datconv.tasa_premio = 0) then
                  error "La tasa premio no puede ser 0."
                  next field tasa_premio
               end if
            end case
        after field dias_reporto
            if (p_datconv.tipo_oper matches "[12]") and
               (p_datconv.dias_reporto != 0) then
               error "Los dias de reporto deben ser 0."
               next field dias_reporto
            end if
            if (p_datconv.tipo_oper matches "[34]") and
               (p_datconv.dias_reporto != p_clvsub.dias_reporto)and
               (first_time = "n") then
               error "Los dias de reporto son erroneos"
               next field dias_reporto
            end if
            if (p_datconv.tipo_oper matches "[12]") and
               (p_datconv.dias_reporto = 0) and
               (first_time = "n") then
```

```
[illegible]

                              [illegible]
                              [illegible]
                         end if
            end input

#           Si es la primera convocatoria [illegible]

            if int_flag = false then
              if first_time = "S" then
               insert into cl_suc values (p_datconv.tipo_oper,
                                          p_datconv.[illegible],
                                          p_datconv.[illegible],
                                          p_datconv.[illegible],
                                          p_datconv.[illegible],
                                          p_datconv.folio,
                                          p_datconv.casa_bolsa,
                                          p_datconv.[illegible])
              end if

#             Guarda la convocatoria que capture

               insert into datconv  values (p_datconv.*)
            else
              let int_flag = false
            end if
            clear form
        end while
end function

function verify_cb ()

#       Funcion para verificar que exista la casa de bolsa dada

         select * from clavesco
                where p_datconv.casa_bolsa = claves.clave
         if status = notfound then
            error "No existe la casa de bolsa "
            let v = false
            return
         end if
         let v = true
 end function

 function verify_folio ()

#       Funcion que verifica si es correcto el rango de folio dado
#       1 --- contado.
#       2 --- contado mismo dia
#       3 --- reporto
        4 --- reporto mismo dia

        let v = false
        case (p_datconv.tipo_oper)
          when "1"
               if (p_datconv.folio [illegible] (p_datconv.folio [illegible]) then
                  let v = [illegible]
```

```
                end if
          when "2"
                if (p_datconv.folio ... ) then
                   let v = true
                end if
          when "3"
                if (p_datconv.folio >= 201) and (p_datconv.folio <= 300) then
                   let v = true
                end if
          when "4"
                if (p_datconv.folio >= 601) and (p_datconv.folio ... ) then
                   let v = true
                end if
        end case
        if v = false then
          error "El tipo de operacion no corresponde al rango de folio dado."
        end if
 end function

function bajas_conv ()

#       Funcion para dar de baja una convocatoria.

        define p_folio like datconv.folio,
               respuesta2 char(1)
        display form convoca
        while 1 > 0
              input by name p_datconv.folio
              if p_datconv.folio = -1 then
                 return
              end if
              let p_folio = p_datconv.folio
              select * into p_datconv.*
                    from datconv where folio = p_folio
              if status = notfound then
                    message "E s e   f o l i o   n o   e x i s t e"
                    sleep 2
                    message ""
              else
                    display by name p_datconv.*
                    prompt "Seguro desea dar de baja este folio (s/n) ?"
                           for char respuesta2
                    if respuesta2 matches "[sS]" then

                     Si voy a borrar el folio que me dieron

                     select * into p_clvsub.*
                           from clvsub where verif = p_folio
                     if status != notfound then

                       Borra el folio que ... las ...

                       delete from clvsub where verif = p_folio
                       delete from datconv where folio = p_folio
                       select ... (folio) into p_datconv.folio from datconv
                          where ...
```

```
Jul 13 1:34 1998  convOI.4gl Page 7


                        if p_datconv.folio <> 0 then

                        Obtengo las nuevas politicas para ese tipo

                        select * into p_datconv.* from datconv
                          where p_datconv.folio = folio

                          Ya no se toma en cuenta la casa de bolsa.

                          let p_datconv.casa_bolsa = "      "

                          Borro las nuevas politicas.

                          insert into divaux values (p_datconv.tipo_oper,
                                                     p_datconv.metodo_asig,
                                                     p_datconv.posicion,
                                                     p_datconv.emision,
                                                     p_datconv.dias_reporto,
                                                     p_datconv.folio,
                                                     p_datconv.casa_bolsa,
                                                     p_datconv.tasa_dese )
                        end if
                        message "Folio dado de baja"
                        sleep 2
                        message ""
                      else

                        Borro un folio que no marca politicas.

                        delete from datconv where folio = p_folio
                        message "Folio dado de baja"
                        sleep 2
                        message ""
                      end if
                    end if
            end if
            clear form
        end while
end function

function repor_conv ()

#       Funcion principal para el manejo de reportes de convocatorias.

        menu "REPORTES"
                command "Todas"
                        "Reporte de todas las convocatorias"
                        clear screen:0:0
                        call todas_conv3()
                command "Resultados"
                        "Resultados de las convocatorias por detalle"
                        clear screen
                        call result_con
                command "Impresora"
                        "Resultados de las convocatorias por impresora"
                        clear screen
```

```
                        call imp_mus_conv ()
                command "Salir"
                        "Regresar al menu anterior"
                        exit menu
        end menu
end function

function todas_conv ()

#       Funcion para el manejo del menu donde se decide si se despliegan todas
#       las convocatorias por pantalla o por impresora.

        menu "Todas"
                command "Pantalla"
                        "Reporte de convocatorias por pantalla"
                        call todas_pan ()
                command "Impresora"
                        "Reporte de convocatorias por impresora"
                        call todas_imp()
                command "Salir"
                        "Regresar al menu anterior"
                        exit menu
        end menu
end function

function todas_imp ()

#       Funcion que manda el despliegue de todas las convocatorias a impresora.

        define f char(12)

        declare r_cursor cursor for
          select * from datconv
          order by tipo_oper,folio

        start report conv_list to printer
          call dia() returning f
          foreach r_cursor into p_datconv.*
            output to report conv_list (p_datconv.*,f)
          end foreach
        finish report   conv_list
        display "Reporte impreso"
        sleep 3
end function

report conv_list (r_datconv,f)

#       Reporte de todas las convocatorias por impresora.

  define r_datconv record like datconv.*,
         f         char(12)

  output
        page length 66
        left margin 0
  format
```

```
        page header
         skip 1 lines
         print [illegible]
         print [illegible] convocatorias
         skip 1 line
         print [illegible]
         skip 1 lines
         print "folio met [illegible] tipo casa emision monto t d [illegible]
         print "--------------------------------------------------------------------------------

        on every row
          print column 2, r_datconv.folio using "###",
                column 9, r_datconv.metodo_asig
                column 15, r_datconv.posicion,
                column 20, r_datconv.tipo_oper,
                column 24, r_datconv.casa_bolsa,
                column 33, r_datconv.emision using "#####",
                column 40, r_datconv.monto using "#,###,###",
                column 51, r_datconv.tasa_desc using "###.##",
                column 60, r_datconv.tasa_premio using "###.##",
                column 68, r_datconv.dias_reporte using "###"
         after group of r_datconv.tipo_oper
                skip 2 lines
end report

function todas_pan ( )

#       Funcion para desplegar todas las convocatorias por pantalla

         define reng smallint
         clear screen
         call display_headers ()
         declare pointer1 cursor for
                select * into p_datconv.* from datconv
                order by tipo_oper,folio
         let reng=10
         foreach pointer1
                display p_datconv.folio                             at reng,1
                display p_datconv.metodo_asig                       at reng,10
                display p_datconv.posicion                          at reng,16
                display p_datconv.tipo_oper                         at reng,25
                display p_datconv.casa_bolsa                        at reng,30
                display p_datconv.emision      using "#####"        at reng,42
                display p_datconv.monto        using "#,###,###"    at reng,4[illegible]
                display p_datconv.tasa_desc    using "###.##"       at reng,5[illegible]
                display p_datconv.tasa_premio  using "###.##"       at reng,67
                display p_datconv.dias_reporte using "###"          at reng,7[illegible]

                [illegible]
                if reng >= 20 then
                 [illegible]
                 [illegible] return [illegible] para cancelar [illegible]
                      for [illegible]
                 [illegible]
                 [illegible]
                     [illegible]
```

```
                end if
                call display_headers ()
                end if
        end foreach
        error "no hay mas convocatorias..."
        clear screen
end function

function display_headers ()

        display
                "B O L S A   M E X I C A N A   D E   V A L O R E S"
                at 3,17
        display "SUBASTA DE CETES " at 4,2
        display "CONVOCATORIAS"    at 5,30
        display
          "Folio  Metodo  Posc.  Tipo  Casa Bolsa  Emision    Monto"
                at 8,1
        display "T.D.     T.P.     D.R." at 8,59
end function

function result_conv ()

#       Funcion que despliega en pantalla los resultados de las convocatorias.

         define suma,reng,aux1 integer,
                tasa,tasa2     decimal(6,2),
                p_datconv1     record like datconv.*,
                hf             char(5),
                f              char(12)
         clear screen
         call dia() returning f
         select hora_cierre into hf from limites
         let reng=12
         display "BOLSA MEXICANA DE VALORES" at 2,16
         display "    SUBASTA DE CETES    "  at 3,18
         display "       CONVOCATORIAS      " at 4,16
         display "FECHA: ",f                  at 5,12
         display "HR. CIERRE: ",hf            at 6,40
         display "TO C/V MONTO   EMIS.   T.D.   T.P.   D.R."
                at 8,11
         declare pointer2 cursor for
                select * into p_datconv1.* from datconv
                order by tipo_trar
         foreach pointer2
            #  call ayuda() returning suma,tasa,tasa2
                display "                                             "  at reng,11
                display p_datconv1.tipo_oper                            at reng,11
                display p_datconv1.metodo_ast,                          at reng,12
                display p_datconv1.posicion                             at reng,14
                display p_datconv1.monto           using "#,###,###"    at reng,15
                display p_datconv1.emision         using "######"       at reng,25
                display p_datconv1.tasa_desc       using "##.##"        at reng,32
                display p_datconv1.tasa_prom       using "##.##"        at reng,38
                display p_datconv1.dias_tr         using "###"          at reng,45
```

```
            let cong = cong + 1
            if cong >= 20 then
               let cong = 10
               prompt "Oprima <RETURN> para continuar o 'X' para cancelar.."
                  for char answer
               clear screen
               if answer matches "[Xx]" then
                  return
               end if
            end if
         end foreach
         prompt "Ya no hay mas convocatorias, oprima <RETURN>" for cong
         clear screen
end function

function ayuda()

#        Funcion para obtener monto total , mejores tasas de las convocatorias.

#        suma  --- monto total de convocatorias segun un tipo.
#        tasa  --- tasa de descuento mejor
#        tasa2 --- tasa pronto mejor

         define suma        integer,
                tasa,tasa2  decimal (6,2)

                  let tasa = 0
                  let tasa2 = 0
                  case (p_clvsub.tipo_oper)

#                      Contado

                       when "1"
                       if p_clvsub.posicion = "C" then
                        select sum(monto),min(tasa_desc)
                               into suma,tasa from datconv
                               where datconv.tipo_oper = "1"
                       else
                        select sum(monto),max(tasa_desc)
                               into suma,tasa from datconv
                               where datconv.tipo_oper = "1"
                       end if

#                      Contado mismo dia

                       when "2"
                       if p_clvsub.posicion = "C" then
                        select sum(monto),min(tasa_desc)
                               into suma,tasa from datconv

                               where datconv.tipo_oper = "2"
                       else
                        select sum(monto),max(tasa_desc)
                               into suma,tasa from datconv
                               where datconv.tipo_oper = "2"
                       end if
```

```
                        Reporto.

                        when "3"
                        if p_clvsub.posicion = "C" then
                         select sum(monto),min(tasa_premio)
                                into suma,tasa2 from datconv
                                where datconv.tipo_oper = "3"
                        else
                         select sum(monto),max(tasa_premio)
                                into suma,tasa2 from datconv
                                where datconv.tipo_oper = "3"
                        end if

                        Reporto mismo dia.

                        when "4"
                        if p_clvsub.posicion = "C" then
                         select sum(monto),min(tasa_premio)
                                into suma,tasa2 from datconv
                                where datconv.tipo_oper = "4"
                        else
                         select sum(monto),max(tasa_premio)
                                into suma,tasa2 from datconv
                                where datconv.tipo_oper = "4"
                        end if
                    end case

#                   Si el tipo fue rporto o reporto mismo dia llamo a una funcion
#                   que busca la tasa de descuento acertada.

                    if p_clvsub.tipo_oper matches "[34]" then
                        call find_tasa (tasa2) returning tasa
                    end if
                   return suma,tasa,tasa2
end function

function find_tasa (t2)

#        Funcion para encontrar la tasa de descuento correspondiente a la mejor
#        tasa premio de un reporto.

#        t2 --- tasa premio de la cual se obtine su tasa de descuento.

          define t,t2 decimal(6,2),
                 minimo integer
          select min(folio) into minimo from datconv
                 where (tasa_premio = t2) and
                       (tipo_oper = p_clvsub.tipo_oper)
          select tasa_desc into t from datconv where folio = minimo

         En "t" regreso la tasa de descuento correspondiente a "t2".

          return t
end function
```

[illegible]

```
function cierre

# [illegible]

    clear screen
    select hora_cierre into hcierre from limites
    prompt "Dar hora de cierre (hh:mm) : "
           for hcierre
    update limites set
           hora_cierre = hcierre
end function
```

[illegible] conv[illegible] Page 1

```
database [illegible]

globals
    define
        p_[illegible]     [illegible]
        hclerr[illegible]     char(5)
end globals

function imp_res_conv()

#       Funcion para imprimir el resultado de convocatorias

        define  suma            integer,
                f               char(12),
                tasa,tasa2      decimal(6,2)

        clear screen
        declare pointer3 cursor for
                select * into p_[illegible] from [illegible]
                order by tipo_oper
        start report listado_conv to printer
            call [illegible]() returning f
            foreach pointer3
#               call ayuda() returning suma,tasa,tasa2
                output to report listado_conv(suma,tasa,tasa2,f)
            end foreach
        finish report listado_conv
        display "Reporte impreso"
        sleep 3
end function

report listado_conv(suma,tasa,tasa2,f)

#       Reporte para el resultado de convocatorias

        define suma         integer,
               f            char(12),
               tasa,tasa2   decimal(6,2)

        output
           page length 66
           left margin 0
        format
           page header

             skip 3 lines
             print 15 spaces, "BOLSA MEXICANA DE VALORES"
             print 15 spaces, "   [illegible] DE CETES      "
             print 14 spaces, "RESULTADOS DE CONVOCATORIAS"
             skip 1 lines
             print [illegible] spaces "
                   [illegible]
             [illegible]
             print "[illegible]        [illegible]        [illegible]        D.R "
             print "------------------------------------------------------------------",
                   [illegible]
```

Jue 9 12:21 1993 conv02.4gl Page 2

```
        on every row

          skip 1 lines
          print column 2, p_datconv1.tipo_oper,
                column 3, p_datconv1.metodo_asig,
                column 9, p_datconv1.posicion,
                column 13, p_datconv1.monto using "$#,###,##&",
                column 28, p_datconv1.emision using "-####",
                column 37, p_datconv1.tasa_dscto using "##&.##",
                column 45, p_datconv1.tasa_premio using "##& &",
                column 55, p_datconv1.dias_reporto using "###
end report
```

```
Database sub_ce

globals "glo_ex.4gl"

function principio ()

      define r_auxconv1          record like datconv.*

#       Funcion para iniciar el metodo de tasas multiples.

#       Inicializo la variable que cuenta las posturas, durante la asignacion.

        let tot_par    = 0
        let total_conv = 0

#       Creo las tablas temporales para convocatorias "conv" y posturas "post".

        call crea_cb ()

#       Ordeno las tablas de convocatorias y posturas segun la posicion.

        call ordenar_cb (co_ve,oper)
        call cambiar ()

#       Guardo la hora de cierre para esta asignacion.

        update limites set hora_final = hora_cierre

#       Obtengo el monto total de las convocatorias.

        declare point cursor for
            select * into r_auxconv1.* from conv
                   where tipo_oper = oper

        foreach point
             let total_conv = total_conv + r_auxconv1.monto
        end foreach

        declare p_conv cursor for
           select * into a_conv.* from conv
           for update

        declare p_post cursor for
           select * into a_post.* from post
           for update

end function


function tasas_m ()

#       Funcion para obtener las siguientes tasas y emisiones de compra y venta
#       asi como para guardar las actuales.

        let tasac_ant = tasac
        let tasav_ant = tasav
```

```
        call dif_tasas ()

        if co_ve = "C" then
           let tasav = tasa_post
           let tasac = tasa_conv
           while (tasav = tasav_ant) and (tasav != -1)
                 fetch p_post
                 if status != notfound then
                    call dif_tasas ()
                    let tasav = tasa_post
                    let tasac = tasa_conv
                 else
                    let tasac = -1
                 end if
           end while
        else
           let tasav = tasa_conv
           let tasac = tasa_post
           while (tasac = tasac_ant) and (tasac != -1)
                 fetch p_post
                 if status != notfound then
                    call dif_tasas ()
                    let tasav = tasa_conv
                    let tasac = tasa_post
                 else
                    let tasac = -1
                 end if
           end while
        end if

        if tasav = 0 then
           let tasav = 501
        end if

end function

function dif_tasas ()

# Obtener tasas de convocatorias y de posturas.

        if oper matches "[12]" then
           let tasa_conv = a_conv.tasa_desc
           let tasa_post = a_post.tasa_desc
        else
           let tasa_conv = a_conv.tasa_premio
           let tasa_post = a_post.tasa_premio
        end if

end function

function cambiar ()

# Funcion para indicar el nombre de la tasa que se va a usar segun el tipo
```

```
# de operacion.

        if oper matches "[12]" then
           rename column conv.tasa_desc to tasa
           rename column post.tasa_desc to tasa
        else
           rename column conv.tasa_premio to tasa
           rename column post.tasa_premio to tasa
        end if

end function


function parcial ()

      define r_auxconv          record like datconv.*

#       Obtengo la suma de montos con igual tasa de posturas. Siempre y cuando
#       se trate de una tasa diferente a la anterior.

        let parcial_post = 0

        if (co_ve matches "C") and
           (tasav_ant != tasav) then
                declare point1  cursor for
                   select * into r_auxconv.* from post
                      where (tipo_oper = oper) and
                            (tasa      = tasav)

                foreach point1
                   let parcial_post = parcial_post + r_auxconv.monto
                end foreach
        end if

        if (co_ve matches  "V") and
           (tasac_ant != tasac) then
                declare point2  cursor for
                   select * into r_auxconv.* from post
                      where (tipo_oper = oper) and
                            (tasa      = tasac)

                foreach point2
                   let parcial_post = parcial_post + r_auxconv.monto
                end foreach
        end if

end function


function mat_mul (pos,tipo)

#       Funcion para efectuar la asignacion por el metodo tasas multiples.

#       pos   -- Posicion de la convocatoria (Compra o Venta)
#       tipo  -- Tipo de operacion de la convocatoria (cod, cmd, reg.cmd )
```

```
        define pos,tipo             char(1)

          let co_ve = pos
          let foper  = tipo

          call principio ()

          open  p_conv
          open  p_post
          fetch p_conv
          fetch p_post

          let tasav = -1
          let tasac = -1

          call tasas_m ()
          call parcial ()

#         Ciclo de prorateos.

          call prorat ()

          close p_conv
          close p_post

          call valores_mul ()
          call tirar_cb ()
          call tirar_mul ()

end function


function crea_mulc ()

#         Funcion que crea tablas temporales durante el proceso de asignacion
#         de montos. Metodo Multiple.

#         mulconv - para guardar unicamente las convocatorias ordenadas del
#                   mulconv actual

create table mulconv (folio integer not null, metodo_asig char(1) not null
        , posicion char(1) not null, tipo_oper char(1) not null,
          casa_bolsa char(6) not null, emision integer, monto integer not
          null, tasa_desc decimal(6,2), tasa_premio decimal(6,2),
          dias_reporto integer);

end function


function tirar_mul ()

          drop table mulconv
          drop table mulpas

end function
```

```
function prorat ()

#       Funcion que lleva a cabo el prorateo de el metodo de tasas multiples.

      define r_datconv              record like datconv.*,
             r_mulc                 record like conv.*,
             r_mulpas,aux_conv      record like datconv.*,
             m_post                 record like datpost.*,
             suma,suma1,
             cont2,control2,
             sum_parcial,
             aux_cont,nuev_mon,
             asignar,monto,
             monto_t,paso_monto,
             c_pos,cont_post,
             control_pos,
             aux33,aux44            integer,
             tasa_aux1,
             tasa_m                 decimal(6,2),
             aux55                  decimal(11,4),
             band,band1,
             ultimo,aux_band,
             cont_tasas,
             mul_band,met,
             flag_fin,postu         char(1)

        call crea_mulc ()

        # Verificamos la tasa maxima de convocatorias.

        if co_ve matches "C" then
           select max (tasa) into tasa_lim from conv
                where (tipo_oper = oper)
        else
           select min (tasa) into tasa_lim from conv
                where (tipo_oper = oper)
        end if

        # Verificamos la tasa maxima de asignacion.

        if co_ve matches "C" then
           select min (tasa) into tasa_lasig from conv
                where (tipo_oper = oper)
        else
           select max (tasa) into tasa_lasig from conv
                where (tipo_oper = oper)
        end if

        # Vaciamos los valores de datconv en mulpas.

        if oper matches "[12]" then
          if co_ve matches "C" then
             declare api cursor for
                    select * into r_datconv.* from datconv
                           where tipo_oper = oper
```

```
                        order by tasa_desc,folio

        foreach ap1
           if (r_datconv.tasa_desc <= t_post) then
              insert into mulpas
                     values ( r_datconv.folio,r_datconv.metodo_asig,
                              r_datconv.posicion,r_datconv.tipo_oper,
                              r_datconv.casa_bolsa,r_datconv.emision,
                              0,r_datconv.tasa_desc,r_datconv.tasa_premio,
                              r_datconv.dias_reporto)
           end if
        end foreach
        close ap1
     else
        declare ap2 cursor for
                select * into r_datconv.* from datconv
                        where tipo_oper = oper
                        order by tasa_desc desc,folio

        foreach ap2
           if (r_datconv.tasa_desc >= t_post) then
              insert into mulpas
                     values ( r_datconv.folio,r_datconv.metodo_asig,
                              r_datconv.posicion,r_datconv.tipo_oper,
                              r_datconv.casa_bolsa,r_datconv.emision,
                              0,r_datconv.tasa_desc,r_datconv.tasa_premio,
                              r_datconv.dias_reporto)
           end if
        end foreach
        close ap2
     end if
  else
     if co_ve matches "C" then
        declare ap3 cursor for
                select * into r_datconv.* from datconv
                        where tipo_oper = oper
                        order by tasa_premio,folio

        foreach ap3
           if (r_datconv.tasa_premio <= t_post) then
              insert into mulpas
                     values ( r_datconv.folio,r_datconv.metodo_asig,
                              r_datconv.posicion,r_datconv.tipo_oper,
                              r_datconv.casa_bolsa,r_datconv.emision,
                              0,r_datconv.tasa_desc,r_datconv.tasa_premio,
                              r_datconv.dias_reporto)
           end if
        end foreach
        close ap3
     else
        declare ap4 cursor for
                select * into r_datconv.* from datconv
                        where tipo_oper = oper
                        order by tasa_premio desc,folio

        foreach ap4
```

```
                if (r_datconv.tasa_premio >= t_post) then
                   insert into mulpas
                          values ( r_datconv.folio,r_datconv.metodo_asig,
                                   r_datconv.posicion,r_datconv.tipo_oper,
                                   r_datconv.casa_bolsa,r_datconv.emision,
                                   0,r_datconv.tasa_desc,r_datconv.tasa_premio,
                                   r_datconv.dias_reporto)
                end if
             end foreach
             close ap4
          end if
       end if

       # Se selecciona el metodo que se esta usando.

       select metodo into met from clvsub
            where tipo_oper = oper

       if met matches "M" then
          let aux33 = parcial_post
       end if

       # Inicializacion de variables de control.

       let band1         = true
       let mul_band      = true
       let flag_fin      = true
       let ultimo        = false
       let cont_tasas    = false
       let postu         = false
       let control_pos   = 0

#      Se asigna el nuevo monto por medio de la siguiente formula:

#      nuevo monto = (a/b) X c

#      a = Monto del colocador.
#      b = Monto total de las convocaciones.
#      c = Monto total de los postores.

       call dif_tasas ()

#      Verificamos el total de posturas.

       let c_pos     = 0
       let cont_post = parcial_post

       declare pcin cursor for
           select X into m_post1 from datpst
                where tipo_oper = oper

       foreach pcin
           let c_pos = c_pos + m_post.monto
       end foreach
```



```
      while band1
```

```
        # Verificamos si se trata del ultimo monto a asignar.

        if flag_fin = false then
           let band1 = false
        end if

        # Verificamos el total de convocatorias.

        let cont2     = 0
        select count (*) into cont2 from conv
              where tipo_oper = oper
        let control2 = 0

        while (status != notfound) and ( cont2 > control2 )
                let aux55 = (a_conv.monto/total_conv) * parcial_post
                let aux44 = aux55
                update conv set monto = aux44
                       where current of p_conv
                if oper matches "[12]" then
                   insert into falta_mul
                          values (a_conv.folio,aux44,aux55,a_post.tasa_desc)
                else
                   insert into falta_mul
                          values (a_conv.folio,aux44,aux55,a_post.tasa_premio)
                end if
                fetch p_conv
                call dif_tasas ()
                let control2 = control2 + 1
                let aux44    = 0
                let aux55    = 0
        end while

        if oper matches "[12]" then
           select sum(cifra1) into suma from falta_mul
                  where tasaf = a_post.tasa_desc
        else
           select sum(cifra1) into suma from falta_mul
                  where tasaf = a_post.tasa_premio
        end if

        let aux44 = parcial_post - suma

        Si hubo decimales se ajustan a enteros

        let band     = true
        let aux_cont = 0
        close p_conv
        open  p_conv
        fetch p_conv

        if flag_fin  = false then
           let aux44 = 0
           let cont2 = 0
        end if
```

```
        while band
              if aux_cont != aux44 then
                    let nuev_mon = a_conv.monto + 1
                    update conv set monto = nuev_mon
                           where current of p_conv
                    fetch p_conv
                    let aux_cont = aux_cont + 1
              else
                    # Actualizamos la tabla temporal de tasas multiples

                    let band  = false
                    let sumal = 0
                    close p_conv
                    declare p_mulc cursor for
                            select I into r_mulcI from conv
                                   where tipo_oper = oper

                    foreach p_mulc
                       if co_ve matches "C" then
                          if tasav < r_mulc.tasa then
                             let r_mulc.monto = 0
                          end if
                       else
                            if tasac > r_mulc.tasa then
                               let r_mulc.monto = 0
                            end if
                       end if
                       insert into mulconv
                              values (r_mulcI)
                       let sumal = sumal + r_mulc.monto
                    end foreach

                    if sumal = parcial_post then
                       let sum_parcial  = 0
                     end if
              end if
        end while

        # Actualizar los totales de la tabla conv.

        close p_conv
        open  p_conv
        fetch p_conv

        let control2 = 0

        if oper matches '[12]' then
           declare p_mulpas1 cursor for
                   select I into r_mulpasI from mulpas
                   for update
        else
           declare p_mulpas2 cursor for
                   select I into r_mulpasI from mulpas
                   for update
        end if
```

```
        if oper matches "[12]" then
           open  p_mulpas1
           fetch p_mulpas1
           if co_ve matches "C" then
              open  ap1
              fetch ap1
           else
              open  ap2
              fetch ap2
           end if
        else
           open  p_mulpas2
           fetch p_mulpas2
           if co_ve matches "C" then
              open  ap3
              fetch ap3
           else
              open  ap4
              fetch ap4
           end if
        end if

        while cont2 > control2
              if oper matches "[12]" then
                 if (r_mulpas.monto != 0) or (mul_band = true) then
                   let nuev_mon = r_mulpas.monto + a_conv.monto
                   update mulpas set monto = nuev_mon
                          where current of p_mulpas1
                   if a_conv.monto != 0 then
                      let monto    = r_datconv.monto - nuev_mon
                      update conv set monto = monto
                             where current of p_conv
                   end if
                 else
                     let mul_band = true
                   update conv set monto = 0
                          where current of p_conv
                   update mulpas set monto = 0
                          where current of p_mulpas1
                 end if
                   if co_ve matches "C" then
                     fetch ap1
                   else
                      fetch ap2
                   end if
                   fetch p_conv
                   fetch p_mulpas1
                   let control2 = control2 + 1
              else
                 if (r_mulpas.monto != 0) or (mul_band = true) then
                   let nuev_mon = a_conv.monto + r_mulpas.monto
                   update mulpas set monto = nuev_mon
                          where current of p_mulpas2
                   let monto    = r_datconv.monto - nuev_mon
                   if a_conv.monto != 0 then
                      update conv set monto = monto
```

```
                        where current of p_conv
                end if
              else
                update conv set monto = 0
                        where current of p_conv
                update mulpas set monto = 0
                        where current of p_mulpas2
                  let mul_band = true
              end if
                if co_ve matches "C" then
                   fetch ap3
                else
                   fetch ap4
                end if
                fetch p_conv
                fetch p_mulpas2
                let control2 = control2 + 1
        end if
    end while

    if oper matches '122' then
       if co_ve matches "C" then
          close ap1
       else
          close ap2
       end if
       close p_mulpas1
       close p_conv
       open  p_conv
       fetch p_conv
    else
       if co_ve matches "C" then
          close ap3
       else
          close ap4
       end if
       close p_mulpas2
       close p_conv
       open  p_conv
       fetch p_conv
    end if

    # Actualizamos el monto por asignar.

    let asignar = total_conv - parcial_post
    fetch p_post

    # Verificamos si se trata de un ultimo monto por asignar.

    if ultimo then
       let flag_fin   = false
       let band1      = false
       let cont_tbsas = true
    end if
                                    273

    if control_pos = 1 then
```

```
            let flag_fin    = false
            let band1       = false
            let cont_tasas = true
        else

        let paso_monto = asignar - parcial_post

            if paso_monto =  0 then
                let parcial_post = total_conv
                let flag_fin     = false
                let band1        = false
            else
                if paso_monto < 0 then
                    let total_conv = asignar
                    let ultimo     = true
                end if
                if paso_monto > 0 then
                    let total_conv = asignar
                end if
            end if

        if (asignar < 1) or (asignar = 0) then
            let flag_fin = false
            let band1    = false
        end if

        if co_ve matches "C" then
            let tasa_m = tasav
        else
            let tasa_m = tasac
        end if

        if (tasa_lim = tasa_m) and (paso_monto > 0) and (asignar != 0) then
            call tasas_m ()
            call parcial ()
            let cont_tasas = true
            call verifica_monto () returning aux_band
            if aux_band = false then
                let flag_fin = false
                let band1    = false
            end if
        else
            if (co_ve matches "C") and (tasa_lim > tasa_m) and
               (ultimo = false) and (asignar != 0) then
               call tasas_m ()
               call parcial ()
               let cont_tasas = true
               call verifica_monto () returning aux_band
               if aux_band = false then
                  let flag_fin = false
                  let band1    = false
               end if
            end if
            if (co_ve matches "V") and (tasa_lim < tasa_m)
               and (asignar != 0) and (sum_parcial != 0) then
                  call tasas_m ()
```

```
                call parcial ()
                let cont_tasas = true
                call verifica_monto () returning aux_band
                if aux_band = false then
                   let flag_fin = false
                   let band1    = false
                end if
             end if
          end if

          if tasa_lasig = tasa_m then
             let flag_fin = false
             let band1    = false
          end if

          if cont_tasas = false then
             call tasas_m ()
             call parcial ()
          end if

          let cont_tasas = false

          # Verifico si las tasas de convocatorias son asignables.

          if (co_ve matches "C") and (tasav < tasa_lim) then
             let control2 = 0
             let tasa_aux1 = 0.00
             while cont2 > control2
                   if a_conv.tipo_oper matches "[12]" then
                      let tasa_aux1 = a_conv.tasa_desc
                   else
                      let tasa_aux1 = a_conv.tasa_premio
                   end if
                   if tasa_aux1 > tasav then
                      let total_conv = total_conv - a_conv.monto
                      update conv set monto = 0
                           where current of p_conv
                      update mulpas set monto = 0
                           where (folio = a_conv.folio)
                      let mul_band = false
                   else
                        let tasa_lim = tasa_aux1
                   end if
                   fetch p_conv
                   let control2 = control2 + 1
             end while
          else
              if (co_ve matches "V") and (tasac > tasa_lim) then
              let control2 = 0
              let tasa_aux1 = 0.00
              while cont2 > control2
                    if a_conv.tipo_oper matches "[12]" then
                       let tasa_aux1 = a_conv.tasa_desc
                    else
                       let tasa_aux1 = a_conv.tasa_premio
                    end if
```

```
            if tasa_aux1 < tasac then
               let total_conv = total_conv - a_conv.monto
               update conv set monto = 0
                     where current of p_conv
               update mulpas set monto = 0
                     where (folio = a_conv.folio)
               let mul_band = false
            end if
            fetch p_conv
            let control2 = control2 + 1
        end while
        end if
    end if

    close p_conv
    open p_conv
    fetch p_conv
    let control2 = 0

    if (tasa_lasig < tasac) and (co_ve matches "V") then
       let asignar  = 0
       let flag_fin = false
       let band1    = false
    end if

    if (tasa_lasig > tasav) and (co_ve matches "C") then
       let asignar  = 0
       let flag_fin = false
       let band1    = false
    end if

    if asignar <= parcial_post then
       let parcial_post = asignar
    end if

    # verificamos si se puede seguir asignando.

    if ( co_ve matches "V") and (paso_monto < 0) then
       if tasac > tasa_lasig then
          let flag_fin   = false
          let band1      = false
       end if
    else
       if tasav < tasa_lasig then
          let flag_fin   = false
          let band1      = false
       end if
    end if

    # Verificamos el total de posturas.

    let cont_post = cont_post + parcial_post

    if cont_post = c_pos then
       let postu      = true
    end if
```

```
        if (postu = true ) and (control_pos = 0) then
           let flag_fin    = true
           let band1       = true
           let control_pos = 1
        end if

        end if

       end while

    close p_mulc
    close p_conv
    close p_post

    # lamada a la rutina de segundo prorateo.

    if oper matches "[12]" then
       call prorat1 ()
    end if

    if oper matches "[34]" then
       call prorat4 ()
    end if

end function


function verifica_monto ()

# Rutina que verifica si el total del monto ha sido asignado completamente
# a la tasa competitiva.

      define estado               char (1),
             b_actp,b_summ        record like conv.*,
             monto_pro            integer

   # Inicializamos las variables de control.

   let estado  = false

   # declaramos el apuntador que inicializara los montos de tasa mayor o
   # igual a la tasa limite.

   declare act_pr cursor for
           select * into b_actp.* from conv
                  where (tasa >= tasa_lim)
           for update

   foreach act_pr
           update conv set monto = 0
                  where (tasa >= tasa_lim)
   end foreach

   # Obtenemos el monto que falta por asignar.
```

```
    let monto_pro = 0

    declare sum_m cursor for
            select * into b_summ.* from conv
                    where (tipo_oper = oper)

    foreach sum_m
            let monto_pro = monto_pro + b_summ.monto
    end foreach

    # Verificamos si el monto que falta por asignar lo cubre la tasa siguiente

    if monto_pro < parcial_post then
       open  p_conv
       fetch p_conv
       while (status != notfound)
             insert into mulconv values (a_conv.*)
             if co_ve matches "C" then
               if oper matches '[12]' then
                  insert into falta_mul
                         values (a_conv.folio,a_conv.monto,0,a_post.tasa_desc)
               else
                  insert into falta_mul
                         values (a_conv.folio,a_conv.monto,0,a_post.tasa_premio)
               end if
             else
               if oper matches '[12]' then
                  insert into falta_mul
                         values (a_conv.folio,a_conv.monto,0,a_post.tasa_desc)
               else
                  insert into falta_mul
                         values (a_conv.folio,a_conv.monto,0,a_post.tasa_premio)
               end if
             end if
             fetch p_conv
       end while
    end if

    if monto_pro >= parcial_post then
       let estado      = true
       let total_conv = monto_pro
    end if

    if (monto_pro = 0) and (parcial_post = 0) then
       let estado = false
    end if

    # Regresamos el valor de estado.

    return estado

end function
```

A N A L I S I S.

TAREA NUMERO UNO.

INFORMACION PARA ENTREVISTAS.

Para realizar el análisis para el sistema de aduanas, se realizaron entrevistas con las siguientes personas:

- Director General de Aduanas.
- Director de Pedimentos.
- Director de Administracion de Aduanas.
- Subdirector de Aduanas.
- Subdirector de Importaciones.
- Jefe del Departamento de Aranceles.
- Jefe del Departamento de Impuestos.

A partir de las entrevistas se obtuvo la información necesaria para la realización del Sistema de Aduanas. Esto es, se realizó un diagnóstico para conocer la situación actual, la forma en que el sistema les ayudaría, la información que se almacenaría y los recursos que se necesitarían, tanto técnicos como humanos.

Como resultado de las entrevistas se definieron los requerimientos para el Sistema de Aduanas, todo quedo acentado en el documento que se presenta a continuación.

Objetivo:

El objetivo principal del sistema es ayudar a la Dirección General de Aduanas de la S.H.C.P. a ejercer un control absoluto de todos los pedimentos de importación que se manejan en todas las aduanas del país.

Politicas Generales.

Todas las aduanas deberán registrar todos los pedimentos de importación que se les presenten, para hacerles el trámite correspondiente y darles el seguimiento adecuado.

La Dirección General de Aduanas se encargará de proporcionar oportunamente la información relativa a fracciones arancelarias, así como su tasa advalorem y todo tipo de disposiciones fiscales que se necesiten.

Politicas de Operación.

Para una correcta operación del sistema, se establece como responsabilidad de la Dirección General de Aduanas, el mantenimiento de todos los catálogos y tablas con que cuente el sistema, ya que las aduanas no podrían utilizar, ni los catálogos ni las tablas, solo las utilizarán para consultas.

Pedimentos.

Las aduanas procederán a dar trámite a los pedimentos de importación que les sean presentados.

Se establece como responsabilidad de la aduana, a través del vista aduanal, la comprobación de la autenticidad y legitimidad de la información declarada por el importador.

Las aduanas podrán cancelar los pedimentos que no tengan movimiento alguno en los últimos 30 días naturales, emitiendose un reporte de éstos para el jefe de la aduana.

Características del Pedimento.

El agente aduanal al que se le turne el pedimento de importación se encargará de completar junto con el importador la siguiente informacion:

a) Datos del importador.

- Importador.
- RFC.
- Número de facturas y fecha.
- Proveedor.

b) Registros.

- Registro de entrada.
- Fecha y número de pedimento.
- Registro de ingreso.
- Vista autorizado.
- Certificación de caja.

c) Declaraciones.

- Medio transporte.
- País de origen.
- País de procedencia.
- Tipo de cambio.
- Valor factura en moneda extranjera.

d) Mercancía.

- Bultos.
- Cantidad.
- Peso bruto.
- Descripción comercial.
- Fracción arancelaria.
- Valor de las mercancías.
- Valor advalorem.
- Impuesto general de importación.

Especificaciones.

1. Para el cálculo de todos los impuestos se tomará como base el mayor valor entre el valor normal y el valor comercial, al cual se le llamará valor base.

1.1 La tasa advalorem se obtiene de la fracción arancelaria correspondiente y se multiplica por el valor base.

1.2 Se cálcula un 5% sobre el valor base.

1.3 Se cálcula un 4% al millar sobre el valor base.

1.4 Se cálcula el IVA sobre el valor base.

1.5 Se obtiene el total sumando los 4 impuestos anteriores.

2. Los posibles status que puede tomar un pedimento serán:

Status	Significado
10	Recibido
20	Comprobado
30	Aprobado
40	Rechazado
50	Registró ingreso en caja
60	Impreso
99	Cancelado

TAREA NUMERO DOS.

DESCRIPCION DEL PROBLEMA.

La definición de requerimientos para el sistema se obtuvo a partir del levantamiento de información a través de las entrevistas con la Dirección General de Aduanas.

- Declaración del Problema.

Actualmente la Dirección no tiene la información acerca de todos los pedimentos que se presentan en las aduanas del pais, ademas de que no les puede proporcionar oportunamente , a las aduanas, las tasas advalorem de las fracciones arancelarias vigentes.

- Objetivo.

Contar, a nivel aduana, con una herramienta que permita controlar eficazmente los pedimentos que se reciban, implementando un sistema de información que ayude a llevar ese control y que proporcione oportunamente información para la toma de decisiones, que sea seguro, paramétrico y que sea flexible para que permita adicionar otros módulos.

DESCRIPCION DEL SISTEMA PROPUESTO.

El sistema propuesto se piensa realizar con un computador Unisys serie A.

El software con que se desarrollará es LINC II version 13.1 que usa la base de datos DMS II.

El sistema debe ser en linea por la dinámica que se requiere para que así una aduana pueda dar un servicio eficiente a los importadores y exportadores.

Se pretende que en una aduana se lleve un control estricto sobre todos los pedimentos de importación, por ello se piensa en un archivo maestro de pedimentos, dicho archivo necesitará auxiliarse de un conjunto de catálogos y tablas que le proporcionarán la información necesaria para poder darle un seguimiento adecuado a los pedimentos.

La información que tendrán los catálogos y tablas será la misma para todas las aduanas, y se dará de alta y se actualizará únicamente por la Dirección General de Aduanas, dichas actualizaciones serán realizadas en linea para que sean oportunas.

Los pedimentos de importación irán cambiando de status en el sistema, en la medida en que el importador vaya cumpliendo con los requisitos necesarios para poder introducir al país los artículos de su interés.

Deberá existir un reporte que esté en "mezcla" todo el tiempo, preguntando cuando un pedimento esté en status OK y en ese instante imprimirlo y entregar una copia al importador como constancia de que ha cumplido todos los trámites correspondientes e introducir sus articulos.

La idea de los catálogos y tablas auxiliares es con el fin de que el sistema sea paramétrico, facilitando con ello el mantenimiento.

TAREA NUMERO TRES

COSTO DEL SISTEMA

Haciendo uso del análisis de puntos de funcion (FPA) se tiene:

Hoja de trabajo para el resumén de puntos de función.

Funciones:

Entradas	0 simples * 3	= 0
	1 rueda * 4	= 4
	1 compleja * 6	= 6
	Total	10
Salidas	3 simples * 4	= 12
	1 promedio * 5	= 5
	1 compleja * 7	= 7
	Total	24
Archivos	9 simple * 7	= 63
	2 promedio * 10	= 2
	1 complejo * 15	= 15
	Total	98
Consulta	0 simple * 3	= 0
	0 promedio * 4	= 0
	1 compleja * 6	= 6
	Total	6
Procesos batch	0 simple * 10	= 0
	3 promedio * 20	= 60
	1 complejo * 30	= 30
	Total	90

Total de puntos de función sin ajustar 228

Complejidad del proceso (0 = simple 5 = complejo)

Característica	GI
1. Comunicación de datos	0
2. Funciones distribuidas	2
3. Rendimiento	3
4. Configuración de mucho uso	2
5. Tasa de transacciones	2
6. Entradas On Line	3
7. Eficiencia al usuario final	3
8. Actualización On Line	4
9. Procesamiento complejo	3
10. Reusable	1
11. Fácil de instalar	3
12. Fácil de operar	3
13. Sites multiples	4
14. Facilidad de cambios	5
Total	23

Total del grado de influencia (GI) = 38

Factor de ajuste : 0.65 + (0.01 * (38/GI)) = 1.03

Puntos de función: (228/tot de puntos de función sin ajustar) * (1.03/factor de ajuste)=234.84

234.84 puntos de función

1 punto - 2 horas hombres

469.68 horas/hombre 1 dia/hombre = 8 hrs/hombre

58.71 dias/hombre 1 mes/hombre = 20 dias/hombre

2.9355 meses/hombre - 3 meses hombre

Si se considera un costo de 60 dolares por hora hombre, el costo total del sistema será de $28,180.80 dolares y se terminará dentro de 3 meses hombre.

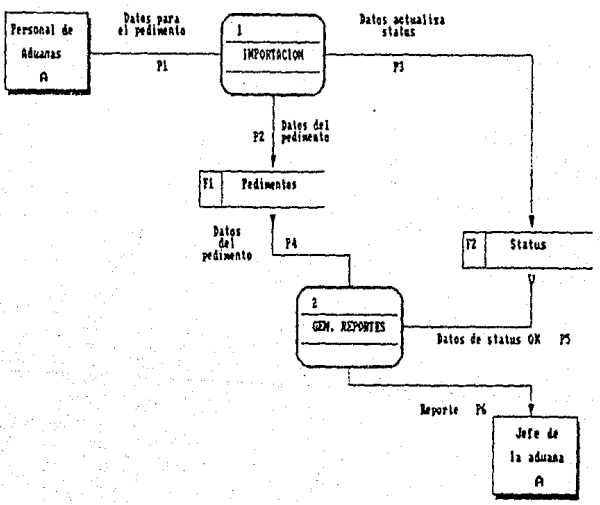
TAREA NUMERO CUATRO.

MODELO CONCEPTUAL.

ENTIDADES EXTERNAS.

ENTIDAD	ENTRADA QUE RECIBE	SALIDA QUE PROPORCIONA
Personal de la aduana	Datos para la solicitud del pedimento de importación	Reporte de pedimentos aprobados y de los cancelados.

DIAGRAMA DE FLUJO DE DATOS.



Descripción del flujo de datos.

Clave	Nombre	Origen	Destino
P1	Datos para el pedimento	A	1
P2	Datos del pedimento	1	F1
P3	Datos actualiza status	1	F2
P4	Datos del pedimento	F1	2
P5	Datos status ok	F2	2
P6	Reporte	2	A

Descripción Detallada del Flujo de Datos

Clave	Descripción
P1	Datos proporcionados que intervienen en la solicitud de un pedimento de importación.
P2	Información que actualiza todas las solicitudes de pedimentos para poder aprobarlas.
P3	Datos que actualizan el status de un pedimento de importacion.
P4	Datos de todas las solicitudes de pedimentos que fueron aprobadas y/o cancceladas.
P5	Status para poder imprimir el reporte del pedimento aprobado.
P6	Reporte de los pedimentos aprobados y/o cancelados.

Descripción de Procesos.

1. El proceso de pedimentos de importación permitirá las acciones de altas, bajas, cambios y consultas a los pedimentos de importación que se almacenarán en el archivo correspondiente.

2. El proceso de generación de reportes para los pedimentos de importación, se encarga de generar el reporte de los pedimentos de importación, tanto de los que han sido aprobados, como de los que han sido cancelados.

Normalización de Estructuras.

A continuación se describen algunas de las estructuras de datos empleadas y la especificación de cada campo en los archivos que se generaron para crear el sistema que resolverá el problema.

Fecha : EXE 90	1	Pagina : 1	2
Sistema : Aduanas	3	Identificacion: CAADU	4

Descripción 5

El Catálogo de aduanas deberá contener los siguientes datos Clave de la aduana, nombre de la aduana, nombre del administrador de la aduana y el RFC del administrador.

S. H. C. P.
SISTEMA DE ADUANAS
CATÁLOGO ADUANAS

CLAVE DE LA ADUANA [ ]
NOMBRE DE LA ADUANA [ ]
ADMINISTRADOR [ ]
R.F.C. ADMINISTRADOR [ ]

Especificaciones

Tiempo de Respuesta: 2 Segundos 6	Frecuencia: A solicitud usuario 7	Tamaño: 500 registros 8
Medio : Disco 9	Disponibilidad : A solicitud del usuario 10	

Comentarios

Aquí se registraran los datos principales de las aduanas 11

Fecha : ENE 90	1	Pagina : 2	2
Sistema : Aduanas	3	Identificacion: CAAGE	4

Descripción 5

El Catálogo de agentes aduanales deberá contener los siguientes datos: clave del agente aduanal, nombre del agente aduanal, indicador de si es representante legal o no.

S. H. C. P.
SISTEMA DE ADUANAS
CATALOGO DE AGENTES ADUANALES

CLAVE AGENTE ADUANAL [ ]

NOMBRE AGENTE ADUANAL [ ]

REPRESENTANTE LEGAL [ ] (SI/NO)

Especificaciones

Tiempo de Respuesta: 2 Segundos	6	Frecuencia: A solicitud usuario	7	Tamaño: 4000 registros	8
Medio : Disco	9	Disponibilidad : A solicitud del usuario			10

Comentarios

Aqui se registraran los datos principales de los agentes 11

Fecha :	DIC 90	1	Página :	1	2
Sistema :	Aduanas	3	Identificación:	CADU	4

Descripción

catalogo de aduanas

5

Contiene la información de las aduanas en cuanto a sus datos generales.

Elementos de datos

IDENTIFICADOR	NOMBRE 7	LONG. 8	TIPO 9 I/A	I/M	FORMATO 10	VALIDACION 11
CVEADUANA	Clave de aduana	4	X	1	xxxx	Diferente de cero.
NOMADUANA	Nombre aduana	40	A	M	xxxx	Diferente de espacios.
ADMADUANA	Adminis-trador	40	A	M	xxxx	Diferente de cero.
RFCADMADU	RFC del adminis.	13	A	M	xxxx	Diferente de espacios.
FECHACAMB	Fecha cambio	6	A	M	xxxxxx	
HORACAMB	Hora de cambio	8	A	M	xxxxxxxx	
BAJA	Baja	1	A	M	x	

Especificaciones

Tiempo de respuesta: 12	Tamaño: 13	Frecuencia: 14
3 Segundo	112 caracteres	Cada vez que el usuario lo requiera.

Comentarios

15

Todos los campos son requeridos para dar de alta o modificar un registro excepto los campos FECHACAMB, HORACAMB y BAJA.

Fecha :	DIC 90	1	Pagina :	2	2
Sistema :	Aduanas	3	identificacion:	CARGE	4

Descripción **catalogo de agentes**

5

Contiene la información de los ayentes aduanales en lo referente a sus datos generales.

Elementos de datos

IDENTIFICADOR	NOMBRE 7	LONG. 8	TIPO 9		FORMATO 10	VALIDACION 11
			A/N	I/M		
CVEAGEADU	Clave de agente aduanal	5	N	I	xxxxx	Diferente de cero.
NOMAGEADU	Nombre agente	40	A	M	xxxx	Diferente de espacios.
INDREPLEG	Indica repte. legal	2	A	M	xx	Solo SI o NO
FECHACAMB	Fecha cambio	6	A	M	xxxxxx	
HORACAMB	Hora de cambio	8	A	M	xxxxxxxx	
BAJA	Baja	1	A	M	x	

Especificaciones

Tiempo de respuesta: 12	Tamaño: 13	Frecuencia: 14
3 Segundo	62 caracteres	Cada vez que el usuario lo requiera.

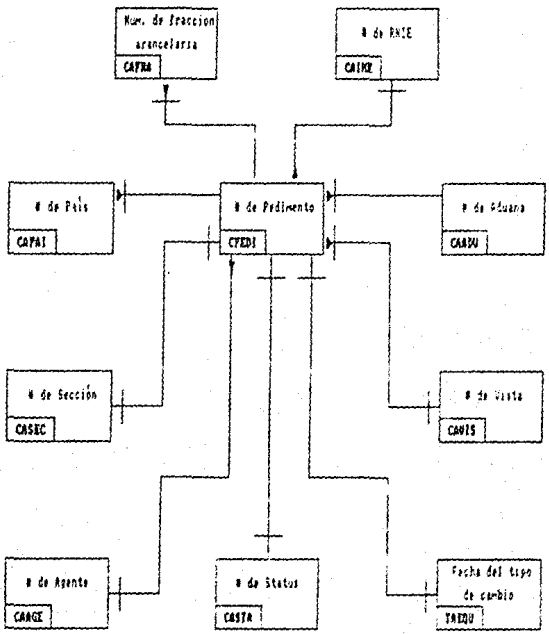
Comentarios

15

Todos los campos son requeridos para dar de alta o modificar un registro excepto los campos FECHACAMB, HORACAMB y BAJA.

MODELO DE INFORMACION

MODELO ENTIDAD RELACION



DISEÑO

TAREA NUMERO UNO

Diseño de Archivos y/o Base de Datos.

A partir del modelo de información, la base de datos la integran los siguientes catálogos y tablas:

1.- CAADU
2.- CAAGE
3.- CAFRA
4.- CAIME
5.- CAPAI
6.- CASEC
7.- CASTA
8.- CAVIS
9.- CPEDI
10.- DEPED
11.- TAEQU
12.- TASIS

El catálogo CAADU acepta : altas, bajas cambios y consultas, su llave de acceso será la clave de la aduana.

El catálogo CAAGE acepta: altas, bajas, cambios y consultas, su llave de acceso será la clave del agente aduanal.

El catálogo CAFRA acepta: altas, bajas, cambios y consultas utilizará una llave de acceso compuesta: número de fracción y número de subfracción.

El catálogo CAIME acepta: altas, bajas, cambios y consultas, su llave de acceso será el registro nacional de importador/exportador.

El catálogo CAPAI acepta las acciones de altas, bajas, cambios y consultas, su llave de acceso será la clave del pais.

El catálogo CASEC acepta las acciones de altas, bajas, cambios y consultas, utilizará una llave de acceso compuesta: clave de la aduana y número de sección.

El catálogo CASTA acepta las acciones de altas, bajas, cambios y consultas, su llave de acceso será la clave del status.

El catálogo CAVIS acepta las acciones de altas, bajas, cambios y consultas, su llave de acceso será compuesta formada por: clave de la aduana y clave del vista aduanal.

El archivo CPEDI acepta las acciones de altas y consulta, su llave de acceso será compuesta, formada por: número de pedimento, clave de la aduana y número de sección.

El archivo DEPED acepta las acciones de altas y consultas, su llave de acceso será compuesta, formada por: el número de pedimento, clave de la aduana, número de sección y secuencia.

La tabla TAEQU acepta las acciones de altas, bajas, cambios y consultas, su llave de acceso será la fecha del tipo de cambio.

La tabla TASIS acepta las acciones de altas, bajas, cambios y consultas, su llave de acceso será un uno ya que es un registro único.

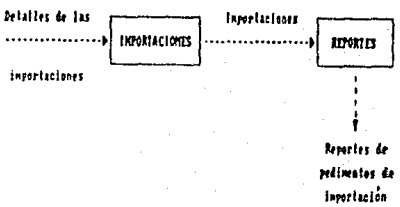
NOTA: Para ver la descripción de los archivos referirse a la Tarea Cuatro del Análisis.

TAREA NUMERO DOS

DESCRIBIR LAS FUNCIONES DEL SISTEMA

El diseno del sistema propuesto para dar solución a la problemática detectada, se basa en 2 funciones:

- Importaciones
- Reportes



IMPORTACIONES.

En esta función se dará de alta la carátula del pedimento de importación de que se trate, teniendo cuidado de llenar todos los datos de la pantalla en forma correcta.

Cuando este dado de alta el pedimento de importación en forma correcta, se deberá proceder a capturar, para dar de alta, los detalles de ese pedimento de importación, en esta pantalla se deberá indicar cuando ya no existan más detalles para ese pedimento, en ese instante se dará por concluida la captura de un pedimento de importación.

REPORTES.

Se contará con un módulo de reportes para los pedimentos de importación los cuales se podrán mandar a imprimir o sólo verlos en la pantalla.

TAREA NUMERO TRES

DISENO DE LAS SALIDAS DEL SISTEMA.

El reporte del pedimento de importación, deberá mostrar la siguiente información.

Pedimento	Número de pedimento y fecha
Agente	Clave y nombre del agente aduanal
Aduana	Clave y nombre de la aduana
País origen	Clave y nombre del país de origen
País procedencia	Clave y nombre del país de procedencia
Régimen	Régimen aduanero
	T Temporal
	D Definitivo
Importador	Nombre del importador
Datos	Datos generales del importador
	Domicilio
	RFC
	RNIE
	Facturas
	Proveedor
	Registro de entrada
	Número de registro de entrada
Nivel comercial	P, O, C, X
Registro de caja	Número de registro de ingreso a caja
Bultos	Cantidad bultos, descripción y peso bruto
Mercancía	Cantidad, unidad de medida, peso neto
Valor comercial	Valor comercial
Valor normal	Valor normal
Impsto advalorem	Impuesto advalorem
Impuesto del 5%	Valor base
4% al millar	4% al millar
IVA	IVA
Total	Total
Fracción arancel	Número de fracción arancelaria

El reporte de los agentes aduanales mostrará la siguente información:

Clave	Clave del agente aduanal
Tipo	Si representante legal
	No agente aduanal
Nombre	Nombre del agente o representante legal

El reporte de importadores/exportadores mostrará la siguiente información:

RNIE	Clave del registro nacional de importador/exportador
Nombre	Nombre del importador
RFC	RFC del importador
Nivel comercial	Nivel comercial p, c, o, x
Domicilio	Domicilio del importador

El reporte de vistas aduanales mostrará la siguiente información:

Aduana	Clave de la aduana
Vista	Calve del vista
Nombre	Nombre del vista
RFC	RFC del vista

TAREA NUMERO CUATRO

DISENO DE LAS ENTRADAS DEL SISTEMA

La información que se requiere para las entradas de los pedimentos de importación será:

Agente	Clave del agente
Aduana	Clave de la aduana
País origen	Clave del país de origen
País procedencia	Clave del país de procednecia
Régimen	Tipo de régimen aduanero
	T Tempooral
	D Definitivo
Importador	Datos del importador
	Nombre
	Domicilio
	RFC
	RNIE
	Facturas
	Proveedor
Nivel comercial	p, o, c, x
Registro de caja	Registro de ingreso a caja
Bultos	Cantidad
	Descripción
	Peso bruto
Mercancía	Cantidad
	Unidad de medida
	Peso neto
Valor comercial	Valor comercial
Valor normal	Valor normal
Fracción arancel	Número de fracc. arancelaria

La información requerida como entrada para el catalogo de agentes aduanales será:

Clave	Clave del agente aduanal
Nombre	Nombre del agente aduanal
Repte legal	SI Indica que es rep legal
	NO Indica que es un agente aduanal

La información requerida como entrada para el catalogo de importadores/exportadores será:

RNIE	Cladel reg. nal. de imp/exp
Nombre	Nombre del importador
RFC	RFC del importador
Dirección	Domicilio del Importador
Nivel comercial	p,c,x,o

La información requerida como entrada para el catalogo de vista será:

Aduana	Clave de la aduana
Vista	Clave del vista
RFC	RFC del vista

TAREA NUMERO CINCO

DISENO DE LAS INTERFASES DEL SISTEMA

No hay interfases por lo tanto esta tarea no aplica.

TAREA NUMERO SEIS

DISENO DE LA ESTRUCTURA DEL SOFTWARE

Para tener una idea clara del sistema, se muestra a continuacion la carta de estructuras.

A.- El módulo de catalogos, es el módulo en el cual se encuentrán todos los catalogos y tablas auxiliares que utilizan los pedimentos de importación.

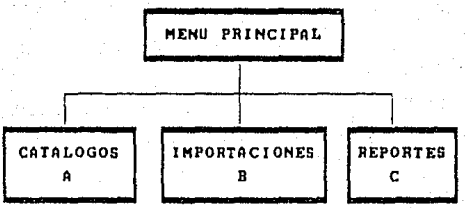
El "Fanin" es de uno al igual que su "Fanout".
La cohesion de este módulo es comunicacional.

B.- El módulo de importaciones, es el módulo en el cual se registrán todos los pedimentos de importación.

El "Fanin" es de uno y su "Fanout" es de 2.

La cohesion de este módulo es comunicacional.

C.- El módulo de reportes es en el que se realizán los reportes requeridos por los pedimentos de importación. Su "Fanin" es de uno y su "Fanout" es de dos.


MENU PRINCIPAL
CATALOGOS
A
IMPORTACIONES
B
REPORTES
C

La cohesion de éste módulo es funcional.

El acoplamiento de los módulos de importación y reportes es por control.

El acoplamiento entre los módulos de cátalogos es por datos.

El acoplamiento entre los modulos de importaciones y de

La cohesion de éste módulo es funcional.

El acoplamiento de los módulos de importación y reportes es por control.

El acoplamiento entre los módulos de cátalogos es por datos.

El acoplamiento entre los módulos de importaciones y de cátalogos es por datos.

Por lo que en general hay un excelente acoplamiento y la cohesion de los módulos es muy buena.

TAREA NUMERO SIETE

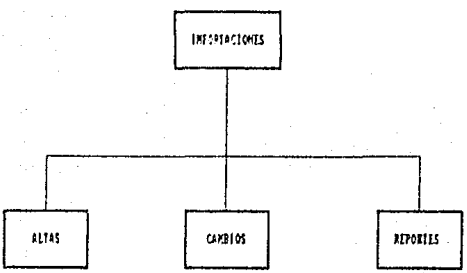
DIALOGOS

No hay diálogos por tanto esta tarea no aplica.

TAREA NUMERO OCHO

Diseño de la lógica de procesamiento del sistema

Para el módulo de Importaciones se tiene:

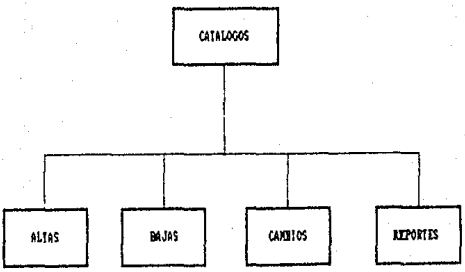


ALTAS — Proceso para dar de alta un pedimento de importación con todas las validaciones especificadas, se desplegaran como validación adicional los nombres o descripciones que se tengan en el módulo de catalogos, en los catálogos correspondientes, CAIME, CAPAI, CAVIS CASIA, CASEC, CAAGE, CAADU, CAFRA, TASIS, TAEQU

CAMBIOS — Servirá para modificar, actualizar la información de los pedimentos de importación.

REPORTES — Proceso que genera los reportes para los pedimentos de importación por video o impresor.

Para el módulo de Catálogos se tiene:



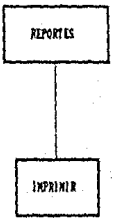
ALTAS — Proceso para dar de alta registros en cualquier catálogo, con las validaciones especificadas.

BAJAS — Proceso para dar de baja registros en cualquier catálogo.

CAMBIOS — Proceso para actualizar los datos de cualquier catálogo.

CONSULTAS — Consiste en desplegar los datos de un registro específico.

Para el módulo de Reportes se tiene:



IMPRIMIR Proceso que imprime los pedimentos de importación y muestra los resultados por video o impresora.

TAREA NUMERO NUEVE

Construccion del prototipo

A partir de la definicion de requerimientos, del analisis y con lo que hasta el momento se ha disenado, se puede iniciar el desarrollo del prototipo.

La version inicial del prototipo es aquella en la que se han pintado las pantallas que sirven para dar mantenimiento a los catalogos y para capturar los datos de entrada, asi como una muestra del reporte de pedimentos de importacion y las pantallas de captura de los pedimentos de importacion.

Esta presentacion del prototipo al usuario, sirve para ilustrar la forma, el procedimiento que se seguira y como se hara para automatizar las principales funciones que fueron definidas.

El prototipo sirve ademas para que el usuario haga sus comentarios acerca de como se estan haciendo las cosas por parte del departamento de sistemas y sirve de retroalimentacion para que el sistema tome el enfoque que el usuario desea.

El desarrollo del prototipo fue rapido debido a que el pintado de pantallas y reportes es muy eficiente en LINC II y se pudieron enfocar rapidamente al punto de vista del usuario y porque diario se tenian las modificaciones que se solicitaban.

Una vez que el prototipo se evaluo totalmente y que se le hicieron las modificaciones solicitadas, se obtuvo la aprobacio del usuario y este se comprometio a seguir de cerca el desarrollo del sistema con visitas periodicas, una ves a la semana, para revisar el sistema, ademas de disponobilidad para consultas y aclaraciones por parte del equipo de desarrollo.

En este prototipo que se desarrollo, ya se incluyeron todos los datos de las estructuras que forman la base de datos, tal y como se muestra en el modelo de atributos y relaciones y en la descripcion de la estructura de datos.

Es fundamental que desde el principio del prototipo, se incluyan todos los datos y sus relaciones tal como se disenaron para poder evaluar completamente el prototipo.

TAREA NUMERO DIEZ

Preparacion de las especificaciones de programas

MODULO CATALOGOS

Las acciones que se efectuan en este modulo son:

Altas
Bajas
Cambios
Consultas

mismas que son utilizadas en los catalogos:
CAADU, CAAGE, CAFRA, CAIME, CAPAI, CASEC, CASTA, CAVIS, TAEQU y TASIS.

PSEUDOCODIGO

```
Pre-Screen
    Arma encabezados

Pre-Linc
    Efectua validaciones

Main-Logic
    Si Accion = Alta
       Crear registro

    Si Accion = Baja
       Borrar registro

    Si Accion = Cambio
       Actualizar registro

    Si Accion = Consulta
       Desplegar registro
```

MODULO IMPORTACIONES

Su funcion es dar mantenimiento Altas y Consultas a los pedimentos de importacion a traves de las pantallas CPEDI y DEPED.

PSEUDOCODIGO para CPEDI

```
Pre-Screen
    Arma encabezados

Pre-Linc
    Efectua validaciones

Main-Logic
    Si Accion = Alta
       Leer siguiente numero pedimento disponible
       Actualizar registro


   Si Accion = Consulta
      Desplegar registro


   Si Confirmacion = Si
      Pasar datos a DEPED
      LLamar pantalla DEPED

   Si Confirmacion = No
      LLamar a la misma pantalla con los mismos datos
```

PSEUDOCODIGO para DEPED

```
Pre-Screen
    Tomar datos de CPEDI
    Armar encabezados

Pre-Linc
    Efectuar validaciones
```

```
Main-Logic
    Si Accion = Alta
       Crear registro

    Si Accion = Consulta
       Despliega registro

    Si Ultimo Detalle = Si
       Actualizar registro en CPEDI
       Actualizar registro en DEPED

    Si Ultimo Detalle = No
       Actualizar registro en DEPED
       LLamar a la misma pantalla pero sin informacion
```

MODULO REPORTES

Su funcion es ejecutar el reporte seleccionado y mostrarlo en pantalla o imprimirlo en papel.

PSEUDOCODIGO

```
Pre-Screen
    Arma encabezados

Pre-Linc
    Efectua validaciones

Main-Logic
    Ejecuta reporte
    Si Opcion = VD
       Desplegar en video

    Si Opcion = LP
       Imprimir en papel
```

PSEUDOCODIGO REPORTE1

```
Leer      CAAGE
Imprimir  CAAGE
Hasta     EOF
```

```
PSEUDOCODIGO REPORTE2

 Leer      CAIME
 Imprimir  CAIME
 Hasta     EOF


PSEUDOCODIGO REPORTE3

 Leer      CAVIS
 Imprimir  CAVIS
 Hasta     EOF


PSEUDOCODIGO REPORTE4

 Leer      CASTA
 Si Status Pedimento = 50
    Leer CPEDI
         Imprimir encabezado
    Leer DEPED
         Imprimir detalle
         Hasta deped-ultimodetalle = Si
```

TAREA NUMERO ONCE

Diseno del Proceso de Control, Seguridad y Respaldo

Para Control y Seguridad, se declaran las terminales en la red de LINC II donde se les definen niveles de acceso hacia el sistema.

El usuario al asignarse al equipo, inmediatamente tendra acceso a la ventana del sistema, de este modo se determina a traves de la bitacora, que usuario estuvo en que terminal, en que horario y que transacciones realizo.

Se sugiere conservar en secreto el Password de cada usuario y no prestarselo entre ellos mismos para poder deslindar responsabilidades posteriormente.

Dentro de las politicas de respaldo se determino hacer un respaldo total de la base de datos a traves de la utileria SYSTEM/DMUTILITY con las opciones OFFLINE DUMP, para la base de datos de produccion.

Ademas respaldar, despues de correr el SYSTEM/DMUTILITY, todo el software del sistema en cinta magnetica.

Durante la etapa de desarrollo dejar durante la noche , despues de la generacion de la base de datos, un SAU (System Audit Utility) con las opciones UNLOAD TAPE para proteger lo que se haya desarrollado durante el dia.

Se sugiere tambien, respaldar todos los discos donde resida el sistema de produccion y el software de desarrollo (LINC II) durante el fin de semana y cada vez que sea necesario.

Antes de reorganizar la base de datos de produccion, hacer un respaldo total mediante dos metodos:

- SYSTEM/DMUTILITY
- COPY

Esto se hara con el fin de estar protegidos en caso de que la reorganizacion fallara, para poder trabajar normalmente si recuperamos la base de datos del respaldo.

En caso de falla del equipo durante la operacion normal de la base de datos es muy importante tener respaldados los archivos de auditoria de la base de datos asi como los archivos de bitacora que genera LINC II pues aunque es altamente confiable LINC II, y tiene la capacidad de recuperarse solo, corriendo automaticamente un software llamado SYSTEM/DMRECOVERY el cual se encarga de llevar al sistema hasta el punto inmediato anterior a la caida del equipo, si se cuenta con las auditorias y la bitacora, es posible poder recuperar cualquier transaccion que LINC II no pudiera recuperar por si mismo.

Por lo mismo, es recomendable respaldar tanto las auditorias de la base de datos como los archivos de bitacora de LINC II, o sea el LINCLOG, a medida que estos se vayan generando, lo cual es muy sencillo, pues LINC II se encarga de avisar cada vez que requiere una cinta para respaldo.

TAREA NUMERO DOCE

Preparacion del Plan de Entrenamiento

Se considera que el plan de entrenamiento se dara los dias martes de cada semana, empezando la semana siguiente a partir de la finalizacion del prototipo, ya que ademas servira para efectuar pruebas y determinar el grado de avance del sistema.

Una vez que se ha terminado el desarrollo y cuando empiezan las pruebas formales o el paralelo, del sistema, se realizara simultaneamente la capacitacion a todas las personas que operan el sistema.

Por lo tanto habra dos niveles de capacitacion:

A) Detallada Donde sa dara una capacitacion exahustiva a la gente que trabajar con el sistema directamente.

B) Conceptual Enfocada a niveles de direccion con el fin de que conozcan el sistema y sepan la informacion que les proporcionara para tomar decisiones.

Esta capacitacion se dara diariamente durante cuatro horas diarias una vez que se haya terminado el desarrollo.

DESARROLLO

TAREA NUMERO UNO

Desarrollo del sistema

Durante esta fase de desarrollo, se tomo como base el prototipo que se empezo a construir en la fase de Diseno, el cual ya contenia todos los datos y sus atributos tal como se mostraron en el Modelo de Entidad-Relacion.

Al tener un prototipo que de algun modo representaba lo que el sistema a desarrollar tendria que hacer, fue sencillo empezar con el desarrollo del sistema, partiendo del prototipo.

Lo que se hizo fue aplicar todos los comentarios que el usuario habia emitido durante la etapa de desarrollo del prototipo, por lo que el sistema estaba enfocado totalmente hacia el usuario, el cual revisaba periodicamente el avance del sistema y hacia los comentarios convenientes.

Asi mismo, durante esta fase de desarrollo, se realizo la capacitacion que se considero necesaria que el usuario necesitaba para el menejo eficiente del sistema, y fue aqui donde surgieron la mayoria de los comentarios, por parte del usuario, para hacer del sistema una herramienta para facilitar el trabajo de un grupo de gentes y que se ajustara realmente a sus necesidades, que fuera de facil operacion, claro y sencillo de entender en cuanto a los mensajes que emitia.

Para el desarrollo del sistema o sea la programacion se conto con dos gentes que se dedicaron de tiempo completo a la misma.

El desarrollo del sistema se pudo completar a tiempo y para ello diario se mandaban a generar y a veces hasta dos veces al dia, por lo que los cambios se podian presentar al usuario con acelerada prontitud, esto gusto mucho al usuario y hacia que el mismo participara mas activamente y haciendo comentarios enfocados a eficientar la operacion del mismo.

A continuacion se muestra dos de las pantallas que se usan en el sistema tal como quedaron programadas finalmente en LINC II, solo se incluyen dos pantallas pues solo se quiere mostrar que es lo que se puede hacer con LINC II, estas dos pantallas son representativas de sus respectivos modulos.

La primer pantalla se llama CAADU, catalogo de agentes aduanales, y pertenece al Modulo de Catalogos, es importante mencionar que todas las pantallas del Modulo de Catalogos tienen su correspondiente estructura dentro de la base de datos y que las operaciones de alta, baja, cambio y consulta las realiza LINC II sin necesidad de logica especial para los campos que tienen representacion en la pantalla.

Las pantallas del Modulo de Catalogos, realizan las mismas operaciones, alta, baja, cambio y consulta.

La segunda pantalla se llama CPEDI, Caratula de Pedimentos Aduanales, que pertenece al modulo de Pedimentos de Importacion y es importante mencionar que durante la fase de construccion del prototipo y al inicio del desarrollo, se vio que seria mas conveniente hacer dos pantallas para controlar lo que son los pedimentos de importacion ya que en el Modelo de Entidad-Relacion solo se habia contemplado una estructura de datos, pero posteriormente se vio que era mejor tener dos estructuras, una que fueran las Caratulas de Pedimentos de Importacion CPEDI y otra que fueran los Detalles de los Pedimentos de Importacion.

Esta pantalla tambien tiene su representacion en la base de datos y realiza las operaciones de altas, cambios y consultas.

```
REPORT: [illegible]          [illegible] 16 MAY [illegible]
          SINGLE SPECIFICATION LISTING FOR DATABASE [illegible] FULL [illegible]

 ISPEC CAADU - COMPONENT

** SCREEN IMAGE FOR CAADU COMPONENT **


+------------------------------------------------------------------------------+
?ATCAADUCC[illegible]    03C    ]                                              ?
???SHCP ** SISTEMA DE ADUANAS                  CATALOGO DE ADUANAS   **   [illegible] ?
?                                                                              ?
?                                                   PROD. PANTALLA            ?
?                                                                              ?
?                                                                              ?
?                                                                              ?
?       CLAVE DE ADUANA            #?    []                                    ?
?                                                                              ?
?                                                                              ?
?                                                                              ?
?       NOMBRE DE LA ADUANA        [                                      ]    ?
?                                                                              ?
?       ADMINISTRADOR              [                                      ]    ?
?                                                                              ?
?       R.F.C. DEL ADMINISTRADOR   [            ]                              ?
?                                                                              ?
?                                                                              ?
?                                                                              ?
?                                                                              ?
?                                                                              ?
?                                                                              ?
?                         FECHA CAMBIO *     []     HORA CAMBIO *       []     ?
? *                                                                       ]    ?
+------------------------------------------------------------------------------+
```

```
REPORTE: [illegible]          [illegible]
          [illegible] SPECIFICATION LISTING [illegible]

ISPEC CARDU - COMPONENT

** TEACH SCREEN FOR [illegible] **

+------------------------------------------------------------------------+
?                                                                        ?
?     ESTE ES EL CATALOGO DE ADUANAS CUYA LLAVE DE ACCESO                ?
?     ES LA CLAVE DE LA ADUANA.                                          ?
?     EL "R.F.C." QUE SE SOLICITA ES EL ASOCIADO AL ADMI-                ?
?     NISTRADOR DE LA ADUANA Y DEBE PROPORCIONARSE CON --                ?
?     EL SIGUIENTE FORMATO [illegible] ES DECIR, SIN                     ?
?     GUIONES INTERMEDIOS ENTRE LETRAS Y NUMEROS.                        ?
?                                                                        ?
+------------------------------------------------------------------------+
```

```
REPORT: [illegible]          GENERATED 1[illegible] MAY [illegible] 19[illegible]
            LINC II SPECIFICATION LISTING FOR [illegible]

 ISPEC CAADU - COMPONENT

BRIEF DESCRIPTION: CATALOGO DE ADUANAS
ISPEC NUMBER IS 10
USAGE I-O
LINC SUBSYSTEM IS PRIMARY
EXPECTED NUMBER OF ENTRIES IS 100
AUTOMATIC RECALL

NO INTEGRITY
PRIVILEGE LEVEL IS 1
TRACE NOT SET
ACCOUNT MONTH NOT ENTERED
LAST CHANGE DATE 210E[illegible]3, TIME 11:3[illegible]:14.[illegible]


DI; (??SHCH ** SISTEMA DE ADUANAS)
                          LI; 2 POS; 1 - 23
DI; (CATALOGO DE ADUANAS      )
                          LI; 2 POS;47 - 65
DI; (**)                  LI; 2 POS;70 - 71
DI; (CAADU)               LI; 2 POS;75 - 79
DI; (PROG. PANTALLA)      LI; 4 POS;57 - 72
DA; PROGIPO               LI; 4 POS;74 - 80 LE; 5 EDA; USIINPUT
DI; (CLAVE DE ADUANA)     LI; 5 POS; 9 - 23
DA; CVEADUANA             LI; 5 POS;35 - 41 LE; 4 EDA; RV;
DI; (NOMBRE DE LA ADUANA  )
                          LI;13 POS; 9 - 27
DA; NOMADUANA             LI;13 POS;35 - 76 LE;40 EDA; DAD;NOMBRE ADUANA
DI; (ADMINISTRADOR)       LI;14 POS; 9 - 21
DA; ADMADUANA             LI;14 POS;35 - 76 LE;40 EDA;
DI; (R.F.C. DEL ADMINISTRADOR)
                          LI;16 POS; 9 - 32
DA; RFCADMADU             LI;16 POS;35 - 47 LE;13 EDA;
DI; (FECHA CAMBIO)        LI;23 POS;33 - 44
DA; FECHACAMB             LI;23 POS;46 - 53 LE; 6 EDA; NOE; DAD;FECHA CAMBIO
DI; (HORA CAMBIO)         LI;23 POS;57 - 67
DA; HORACAMB              LI;23 POS;69 - 7[illegible] LE; [illegible] EDA; NOE; DAD;HORA CAMBIO
DA; TEXTO                 LI;24 POS; 2 - 77 LE;74 EDA; NOE; USIINPUT DAD;MENSAJ
DA; BAJA                  LE; 1 EDA; USIOUTPUT


AYUDA          000100      ?
AYUDA          000200      ? ?
AYUDA          000300      ? ??
AYUDA          000400      ? ?
AYUDA          000500      ? ??     ESTE ES EL CATALOGO DE ADUANAS CUYA LLAVE DE ACC
AYUDA          000600      ? ??     ES LA CLAVE DE LA ADUANA.
AYUDA          000700      ? ?      EL "R.F.C." QUE SE SOLICITA ES EL ASOCIADO AL AD
AYUDA          000800      ? ?      MINISTRADOR DE LA ADUANA Y DEBE PROPORCIONARSE CON
AYUDA          000900      ? ?      EL SIGUIENTE FORMATO  AAAA999999, ES DECIR, SIN
AYUDA          001000      ? ?      GUIONES INTERMEDIOS ENTRE LETRAS Y NUMEROS.
AYUDA          001100      ? ?
AYUDA          001200      ? ?
AYUDA          001300      ?
AYUDA          001400
```

ISPEC CAADU - COMPONENT

```
PL   000100  MV; GLB.ACRN                           [illegible]
PL   000200  DW; PROXIMO        NOT =               GLB.SPACES
PL   000300      RC; PROXIMO
PL   000400  END;
PL   000500  DW; TC.DAYNUMBER                       [illegible]
PL   000600  DW; TC.DATE                            GSD-FDMA
PL   000700  MV; GSD-FECHAO                         GSD-FECDMA
PL   000800  MV; GSD-DIADMA                         GSD-DIAAMD
PL   000900  MV; GSD-MESDMA                         GSD-MESAMD
PL   001000  MV; GSD-ANODMA                         GSD-ANOAMD
PL   001100  MV; GSD-FECAMD                         GSD-FECHAA
PL   001200  MV; GSD-FECAMD                         FECHACAMB
PL   001300  MV; GLB.TIME                           GSD-HORA
PL   001400  MV; GLB.TIME                           HORACAMB
PL   001500  DW; MAINT                  =           (ADD)
PL   001600      LU; FROM  CVEADUANA                (CAADU)
PL   001700          EK;
PL   001800      END;
PL   001900      DW; GLB.STATUS         NOT =       GSD-ASTER        AND
PL   002000      DW; CAADU.CVEADUANA        =       CVEADUANA
PL   002100          MV; GD-MSGN060                 GSD-NUMER
PL   002200          MV; GD-MSG060                  GSD-TEXTO
PL   002300          MV; GSD-MEN                    TEXTO
PL   002400          RECALL; (CAADU)
PL   002500      END.EXIT;
PL   002600  END;
PL   002700  DW; MAINT                  =           (CHG)
PL   002800      LU; FROM CVEADUANA                 (CAADU)
PL   002900          EK;
PL   003000      END;
PL   003100      DW; GLB.STATUS         =           GSD-ASTER        OR
PL   003200      DW; CAADU.CVEADUANA    NOT =       CVEADUANA
PL   003300          MV; GD-MSGN061                 GSD-NUMER
PL   003400          MV; GD-MSG061                  GSD-TEXTO
PL   003500          MV; GSD-MEN                    TEXTO
PL   003600          RECALL; (CAADU)
PL   003700      END.EXIT;
PL   003800  END;
PL   003900  DW; MAINT                  =           (DEL)
PL   004000      LU; FROM CVEADUANA                 (CAADU)
PL   004100          EK;
PL   004200      END;
PL   004300      DW; GLB.STATUS         =           GSD-ASTER        OR
PL   004400      DW; CAADU.CVEADUANA    NOT =       CVEADUANA
PL   004500          MV; GD-MSGN061                 GSD-NUMER
PL   004600          MV; GD-MSG061                  GSD-TEXTO
PL   004700          MV; GSD-MEN                    TEXTO
PL   004800          RECALL; (CAADU)
PL   004900      END.EXIT;
PL   005000  END;
PL   005100  DW; MAINT                  =           (INQ)
PL   005200      LU; FROM CVEADUANA                 (CAADU)
PL   005300          EK;
PL   005400      END;
PL   005500      DW; GLB.STATUS         =           GSD-ASTER        OR
```

ISPEC CAADU - COMPONENT

```
PL  005600      DW; CAADL.CVEADUANA   NOT =     [illegible]
PL  005700          MV; GD-MSGN001              GSD-NUMER
PL  005800          MV; GD-MSG001               GSD-TEXTO
PL  005900          MV; GSD-MEN                 TEXTO
PL  006000          RECALL; (CAADL)
PL  006100      END.EXIT;
PL  006200      MV; CAADL.CVEADUANA             CVEADUANA
PL  006300      MV; CAADL.NOMADUANA             NOMADUANA
PL  006400      MV; CAADL.NOMADUANA             NOMADUANA
PL  006500      MV; CAADL.RECADUANA             RECADUANA
PL  006600      DW; CAADL.BAJA         =        GSD-UNO
PL  006700          MV; GD-MSGN005              GSD-NUMER
PL  006800          MV; GD-MSG005               GSD-TEXTO
PL  006900          MV; GSD-MEN                 TEXTO
PL  007000          RECALL; (CAADL)
PL  007100      END;
PL  007200      DW; CAADL.BAJA        NOT =     GSD-UNO
PL  007300          MV; GD-MSGN004              GSD-NUMER
PL  007400          MV; GD-MSG004               GSD-TEXTO
PL  007500          MV; GSD-MEN                 TEXTO
PL  007600          RECALL; (CAADL)
PL  007700      END;
PL  007800    END;

LG  000100    DW; MAINT                =        (ADD)
LG  000200        MV; GD-MSGN001                GSD-NUMER
LG  000300        MV; GD-MSG001                 GSD-TEXTO
LG  000400        MV; GSD-MEN                   TEXTO
LG  000500        RECALL; (CAADL)
LG  000600        MV; GLB.SPACES                GLB.ERROR
LG  000700    END;
LG  000800    DW; MAINT                =        (CHG)
LG  000900        FL; GLB.ZEROS                 CAADL.BAJA
LG  001000        FL; GSD-FAMD                  CAADL.FECHACAMB
LG  001100        FL; GSD-HORA                  CAADL.HORACAMB
LG  001200        MV; GD-MSGN002                GSD-NUMER
LG  001300        MV; GD-MSG002                 GSD-TEXTO
LG  001400        MV; GSD-MEN                   TEXTO
LG  001500        RECALL; (CAADL)
LG  001600        MV; GLB.SPACES                GLB.ERROR
LG  001700    END;
LG  001800    DW; MAINT                =        (DEL)
LG  001900        FL; GSD-UNO                   CAADL.BAJA
LG  002000        FL; GSD-C                     CAADL.MAINT
LG  002100        FL; GSD-FAMD                  CAADL.FECHACAMB
LG  002200        FL; GSD-HORA                  CAADL.HORACAMB
LG  002300        MV; GD-MSGN003                GSD-NUMER
LG  002400        MV; GD-MSG003                 GSD-TEXTO
LG  002500        MV; GSD-MEN                   TEXTO
LG  002600        RECALL; (CAADL)
LG  002700    END.EXIT;
```

[illegible]

ISPEC CPEDI - COMPONENT

** [illegible] CPEDI [illegible] **

```
+------------------------------------------------------------------------------+
?[illegible]                                          [illegible] FORMATO [illegible] ??
???SHOP [illegible] **  SISTEMA DE ADUANAS           CARATULA DE PEDIMENTOS ** CPEDI ?
?# DE PEDIMENTO:[illegible] ADUANA:[illegible] SECCION:[illegible]                  ?
??                                                                             ?
? CVE ADUANA      ?    []       NOMBRE  ........................................  ?
? MEDIO TRANSP. [                    ]        VALOR FACTURA      ?          []   ?
? BANDERA TRANS [                  ]          FECHA TIPO CAMBIO ?     []         ?
? CVE PAIS ORI  ?  [] ......................  TIPO CAMBIO        .........       ?
???VE PAIS PRO  ?  []?......................  NUM FACTURAS       ?   []?         ?
? CVE [illegible]  [          ] NOMBRE  ..........................................  ?
? TRAN SIMPLE   ?    []        PROVEEDO [                                    ]  ?
??NIV COMERCIAL [ ]?       DOM PROVEED [                                    ]? ?
? FOLIO REG ENT *       []  FECHA    *     [] PER ADUANERO      [ ]            ?
? CVE VISTA     ?    []        NOMBRE  ........................................  ?
? REG ING CAJA  *       []  [illegible] [                          ]                 ?
??FLETES        ?        []?                                                   ?
?                                  ?    T O T A L E S   A D U A N A L E S   ??
? STATUS DE PEDIMENTO * []         ?   VALOR BASE           *           [] ??
? ................................ ?   ADVALOREM            *           [] ??
??SI PARA CONFIRMAR   [ ]          ?   [illegible] VALOR BASE *          [] ??
? FECHA CAMBIO        *     []     ?   4% AL MILLAR         *           [] ??
? HORA CAMBIO         *       []   ?   I.V.A.               *           [] ??
??                                 ?   TOTAL                *           []???
?*                                                                      ]  ?
+------------------------------------------------------------------------------+
```

ISPEC CREDI - COMPONENT

** TEACH SCREEN FOR CREDI AYUDA **

```
+------------------------------------------------------------------------------+
?                                                                              ?
?                                                                              ?
?       ESTA ES LA CARATULA DE PEDIMENTO. .                                    ?
?       SU CLAVE DE ACCESO ESTA COMPUESTA DE LA CLAVE DE LA                    ?
?       ADUANA, LA CLAVE DE LA SECCION Y EL NUMERO DE PEDI-                    ?
?       MENTO. ESTE ULTIMO LO PROPORCIONA EL SISTEMA AUTO-                     ?
?       MATICAMENTE.                                                           ?
?                                                                              ?
?                                                                              ?
+------------------------------------------------------------------------------+
```

```
ISPEC CPEDI - COMPONENT

BRIEF DESCRIPTION: CARATULA DE PEDIMENTO DOS
ISPEC NUMBER IS 13
USAGE I-C
LINC SUBSYSTEM IS PRIMARY
EXPECTED NUMBER OF ENTRIES IS 100

NO INTEGRITY
PRIVILEGE LEVEL IS 1
TRACE NOT SET
ACCOUNT MONTH NOT ENTERED
LAST CHANGE DATE ASSIGNED, TIME 15:57:10.42


DI; (PAG. FORMATO)                LI; 1 POS;60 - 72
DA; PAGINAS                       LI; 1 POS;74 - 80 LE; 5 EDA; USING-PAG
DI; (??CHIP  **  SISTEMA DE ADUANAS)
                                  LI; 2 POS; 1 - 30
DI; (CARATULA DE PEDIMENTOS ** CPEDI)
                                  LI; 2 POS;49 - 79
DA; NOMPEDPAS                     LI; 3 POS; 1 - 53 LE;17 EDA; ANS
DI; (?)                           LI; 4 POS; 1 -  1
DI; (CVE AGENTE)                  LI; 5 POS; 2 - 10
DA; CVE-CE-AGE                    LI; 5 POS;16 - 22 LE; 5 EDA; DAD;CVE AGENTE ADUA LNAI
DI; (NOMBRE)                      LI; 5 POS;23 - 34
DA; NOMBRE1                       LI; 5 POS;37 - 72 LE;40 EDA; USING-INQUIRY
DI; (MEDIO TRANSF.)               LI; 6 POS; 2 - 14
DA; MEDTRANS                      LI; 6 POS;16 - 37 LE;20 EDA;
DI; (VALOR FACTURA)               LI; 6 POS;46 - 58
DA; VALFACTU                      LI; 6 POS;64 - 77 LE;12 EDA; DAD;VALOR FACTURA
DI; (BANDERA TRANS)               LI; 7 POS; 2 - 14
DA; BANTRANSP                     LI; 7 POS;16 - 35 LE;16 EDA; DAD;BANDERA TRANSPOR
DI; (FECHA TIPO CAMBIO)           LI; 7 POS;42 - 52
DA; FECTIPCAMB                    LI; 7 POS;64 - 71 LE; 6 EDA; DAD;FECHA TIPO CAMBI
DI; (CVE PAIS ORI)                LI; 8 POS; 2 - 13
DA; CVEPAISOR                     LI; 8 POS;16 - 20 LE; 3 EDA; DAD;CVE PAIS ORIGEN
DA; NOMBRE2                       LI; 8 POS;22 - 43 LE;22 EDA; USING-INQUIRY
DI; (TIPO CAMBIO)                 LI; 8 POS;46 - 56
DA; TIPOCAMBIO                    LI; 8 POS;64 - 72 LE; 9 EDA; DI; 2 USING-INQUIRY DAD;TIP
DI; (CVE PAIS PRO)                LI; 9 POS; 1 - 13
DA; CVEPAISPR                     LI; 9 POS;16 - 20 LE; 3 EDA; DAD;CVE PAIS PROCEDE
DI; (?)                           LI; 9 POS;21 - 21
DA; NOMBRE3                       LI; 9 POS;22 - 43 LE;22 EDA; USING-INQUIRY
DI; (NUM FACTURAS)                LI; 9 POS;46 - 57
DA; NUMFACTUS                     LI; 9 POS;64 - 65 LE; 4 EDA; DAD;NUM FACTURAS
DI; (?)                           LI; 9 POS;70 - 75
DI; (CVE PAIS)                    LI;10 POS; 2 - 14
DA; CVEPAIS                       LI;10 POS;16 - 27 LE;10 EDA; DAD;CVE PAIS EMISOR
DI; (NOMBRE)                      LI;10 POS;23 - 34
DA; NOMBRE4                       LI;10 POS;37 - 72 LE;42 EDA; USING-INQUIRY
DI; (TRAM SIMPLE)                 LI;11 POS; 2 - 12
DA; NLAUTTPSIM                    LI;11 POS;16 - 21 LE; 5 EDA; DAD;NUM AUTOR TRAM S
DI; (PROVEED)                     LI;11 POS;29 - 35
DA; NOMPROVE                      LI;11 POS;37 - 72 LE;40 EDA; DAD;NOMBRE PROVEEDOR
DI; (NIV COMERCIAL)               LI;12 POS; 1 - 14
DA; NIVCOM                        LI;12 POS;16 - 16 LE; 1 EDA;
```

```
ISPEC CPEDI - COMPONENT

DI# (?)                          LI#1_ POS#19 - 11
DI# (DOM PROVEED)                LI#1_ POS#2_ - 1_
DA# DOMPROV                      LI#1_ POS#37 - 7_ LD#_C [illegible]
DI# (?)                          LI#12 POS#7_ - 7_
DI# (FOLIO PED INT)              LI#13 POS# 1 - 14
DA# FOLPEDINT                    LI#13 POS#1_ - 2_ LD# _ EDIT NO## D# FOLIO PED INTER
DI# (FECHA)                      LI#13 POS#2_ - 33
DA# FECPEDIM                     LI#13 POS#27 - 44 LD# _ EDIT NO## D##FECHA PEDIMENTO
DI# (REG ADUANERO)               LI#1_ POS#45 - 57
DA# REGADUA                      LI#13 POS#6_ - 6_ LD# 7 EDIT DAD#REGIMEN ADUANERO
DI# (CVE VISTA)                  LI#14 POS# 2 - 1_
DA# CVEVISTA                     LI#1_ POS#1_ - 2_ LD# _ _D_ _ D#CLAVE VISTA
DI# (NOMBRES)                    LI#14 POS#2_ - 34
DA# NOM-RES                      LI#14 POS#37 - 7_ LD#A_ EDIT _____IRY
DI# (REG ING CAJA)               LI#15 POS# 2 - 13
DA# REGINGCAJA                   LI#15 POS#16 - 25 LD# _ EDIT NO## D#REG INGRESO CAJ
DI# (LOC FIS)                    LI#15 POS#2_ - 3_
DA# LOCFISMERC                   LI#15 POS#37 - 6_ LD#3C ED#_ __#LOC FISICA MERCA
DI# (?FLETES)                    LI#16 POS# 1 -  7
DA# FLETES                       LI#16 POS#1_ - 27 LD#1_ EDIT
DI# (?)                          LI#16 POS#2_ - 3_
DI# (?)                          LI#17 POS#_7 - 37
DI# (T O T A L E S)              LI#17 POS#43 - 5_
DI# (P E D I M E N T O)          LI#17 POS#6C - 7_
DI# (?)                          LI#17 POS#8C - 8_
DI# (STATUS DE PEDIMENTO        )
                                 LI#18 POS# 2 - 1_
DA# ESTATUS                      LI#18 POS#22 - 25 LD# 2 EDIT NO## D##STATUS PEDIMENT
DI# (?)                          LI#18 POS#27 - 37
DI# (VALOR BASE)                 LI#1_ POS#41 - 5_
DA# VALBASE                      LI#1_ POS#_2 - 7_ LD#1_ EDIT NO## _A##VALOR BASE
DI# (?)                          LI#1_ POS#8C - 8C
DA# DESCSTA                      LI#1_ POS# 2 - 34 LD#3_ EDIT _S_____IRY
DI# (?)                          LI#19 POS#27 - 37
DI# (ADVALOREM)                  LI#19 POS#41 - 4_
DA# ADVALORIM                    LI#19 POS#_ - 7_ LD#1_ EDIT NO##
DI# (?)                          LI#19 POS#8C - 8_
DI# (?)                          LI#2_ POS# 1 -  1
DI# (SI PARA CONFIRMAR)          LI#2C POS# 2 - 1_
DA# CONFIRMA                     LI#2C POS#22 - 2_ LD# _ EDIT DAD#CONFIRMACION
DI# (?)                          LI#2_ POS#37 - 37
DI# (5% SOBRE VALOR BASE        )
                                 LI#2C POS#41 - 59
DA# VALBASES                     LI#2C POS#6_ - 7_ LD#1_ EDIT NO## DAD#VALOR BASE 5%
DI# (?)                          LI#2_ POS#_C - _
DI# (FECHA CAMBIO)               LI#21 POS# 2 - 13
DA# FECHACAMB                    LI#21 POS#22 - 2_ LD# 6 EDIT NO## DAD#FECHA CAMBIO
DI# (?)                          LI#21 POS#_7 - 37
DI# (4% AL MILLAR)               LI#21 POS#41 - 5_
DA# MILLAR                       LI#21 POS#62 - 7_ LD#15 EDIT NO##
DI# (?)                          LI#21 POS#8C - 8_
DI# (HORA CAMBIO)                LI#22 POS# 2 - 1_
DA# HORACAM_                     LI#22 POS#22 - 31 LD# _ EDIT NO## DAD#HORA CAMBIO
DI# (?)                          LI#2_ POS#_7 - 37
DI# (I.V.A.)                     LI#2_ POS#41 - 4_
```

```
REPORTE L L                  [illegible] 1 [illegible]
              LINCOE SPECIFICATION LISTING FOR [illegible]

 ISPEC CREDI - COMPONENT

DAF I[illegible]                          LINEA [illegible] - 7, [illegible]
DIF (?)                               LINEA POSIC - 1[illegible]
DIF (?)                               LINEA POS 1 - [illegible]
DIF (?)                               LINEA POSIC? - ??
DIF (TOTAL)                           LINEA POS[illegible]1 - 4[illegible]
DAF TOT-L                             LINEA POSIC - 7[illegible] [illegible]
DIF (?)                               LINEA POSI7 - 7?
DIF (?)                               LINEA POSIC - 5?
DAF TEXTO                             LINEA POS 1 - 7[illegible] [illegible]
DAF BAJA                              LIN 1 [illegible]

AYUDA          000100      ?
AYUDA          000200      ? ?
AYUDA          000300      ? ??
AYUDA          000400      ? ?
AYUDA          000500      ? ??    ESTA ES LA CARATULA DE PEDIMENTOS .
AYUDA          000600      ? ??    SU CLAVE DE ACCESO ESTA COMPUESTA DE LA CLAVE DE
AYUDA          000700      ? ?     ADUANA, LA CLAVE DE LA SECCION, Y EL NUMERO DE AS
AYUDA          000800      ? ?     IENTO. ESTE ULTIMO LO PROPORCIONA EL SISTEMA AUT
AYUDA          000900      ? ?     OMATICAMENTE.
AYUDA          001000      ? ?
AYUDA          001100      ? ?
AYUDA          001200      ? ?
AYUDA          001300      ?
AYUDA          001400

PL   000100  MVF CLB.WORK                             SDD-WORK
PL   000200  SDF SD-DAVIS                             GAF
PL   000300       SDF SD-[illegible]                  E[illegible]   L[illegible]
PL   000400       SDF SD-UVISTA                       [illegible]   L[illegible]
PL   000500  ESF
PL   000600  SDF SD-KEY                               G?F
PL   000700       SDF SD-PEDI[illegible]              E[illegible]   LE [illegible]
PL   000800       SDF SD-ADU[illegible]               [illegible]   L[illegible]
PL   000900       SDF SD-SECCI                        [illegible]   LE [illegible]
PL   001000  ESF
PL   001100  SDF SD-FETICA                            EQ [illegible]   LE [illegible]
PL   001200  DWF PROXIMO            NOT *             CLB.SPACES
PL   001300       ROF PROXIMO
PL   001400  ENDF
PL   001500  DCF TC.DAYNUMBER                         INPUT-DATE
PL   001600  DCF TC.DATE                              SSD-FDMA
PL   001700  MVF GSD-FECHAD                           [illegible]
PL   001800  MVF GSD-DIADMA                           GSD-DIA[illegible]
PL   001900  MVF GSD-MESDMA                           [illegible]
PL   002000  MVF GSD-ANODMA                           GSD-ANOAMD
PL   002100  MVF GSD-FECAMD                           GSD-FECHA[illegible]
PL   002200  MVF GSD-FECAMD                           FECHACA[illegible]
PL   002300  MVF CLB.TIME                             GSD-[illegible]
PL   002400  MVF CLB.TIME                             [illegible]
PL   002500  [illegible]                 =            [illegible]
PL   002600       MVF [illegible]                     [illegible]
PL   002700       MVF [illegible]                     [illegible]
```

ISPEC CPEDI - COMPONENT

```
PL        MV; GLB.ZEROS                       VALBASE
PL        MV; GLB.ZEROS                       VALBASES
PL        MV; GLB.ZEROS                       TOTAL
PL        MV; GLB.ZEROS                       ADVALOREM
PL        MV; GLB.ZEROS                       PORCEGENT
PL        MV; GLB.ZEROS                       REGINGCAJA
PL        MV; GLB.SPACES                      TEXTO
PL        LU; FROM  KCAFEDPAS                 (CPEDI)
PL            EX;
PL        END;
PL        DW; GLB.STATUS           NOT =      GSD-ASTER          AND
PL        DW; CPEDI.KCAFEDPAS          =      KCAFEDPAS
PL            MV; GD-MSG-NE               GSD-NUMER
PL            MV; GD-MSG...               GSD-TEXTO
PL            MV; GSD-YES                 TEXTO
PL            RECALL; (CPEDI)
PL        END.EXIT;
PL    END;
PL    DW; MAINT                        =      (INS)
PL        LU; FROM KCAFEDPAS                  (CPEDI)
PL            EX;
PL        END;
PL        DW; GLB.STATUS               =      GSD-ASTER          OR
PL        DW; CPEDI.KCAFEDPAS      NOT =      KCAFEDPAS
PL            MV; GD-MSG-061              GSD-NUMER
PL            MV; GD-MSG061               GSD-TEXTO
PL            MV; GSD-YES                 TEXTO
PL            RECALL; (CPEDI)
PL        END.EXIT;
PL        MV; CPEDI.KCAFEDPAS                 KCAFEDPAS
PL        MV; CPEDI.CVEAGEADU                 CVEAGEADU
PL        MV; CPEDI.MEDTRANS                  MEDTRANS
PL        MV; CPEDI.BANTRANSP                 BANTRANSP
PL        MV; CPEDI.CVEPAISCR                 CVEPAISCR
PL        MV; CPEDI.CVEPAISPR                 CVEPAISPR
PL        MV; CPEDI.VALFACTL                  VALFACTL
PL        MV; CPEDI.FECTIPCAMB                FECTIPCAMB
PL        MV; CPEDI.NUMFACTUS                 NUMFACTUS
PL        MV; CPEDI.NIVCOM                    NIVCOM
PL        MV; CPEDI.CVERNIE                   CVERNIE
PL        MV; CPEDI.NOMPROVE                  NOMPROVE
PL        MV; CPEDI.NUALTTFSIM                NUALTTFSIM
PL        MV; CPEDI.DOMPROV                   DOMPROV
PL        MV; CPEDI.FECFEDIM                  FECFEDIM
PL        MV; CPEDI.CVEVISTA                  CVEVISTA
PL        MV; CPEDI.REGINGCAJA                REGINGCAJA
PL        MV; CPEDI.FLETES                    FLETES
PL        MV; CPEDI.LCCFISMERC                LCCFISMERC
PL        MV; CPEDI.REGADUA                   REGADUA
PL        MV; CPEDI.VALBASE                   VALBASE
PL        MV; CPEDI.ADVALOREM                 ADVALOREM
PL        MV; CPEDI.VALBASES                  VALBASES
PL        MV; CPEDI.PILLAR                    PILLAR
PL        MV; CPEDI.TOTAL                     TOTAL
PL        MV; CPEDI.IVA                       IVA
$
```

ISPEC CPEDI - COMPONENT

```
PL  008300     MV; CPEDI.ESTATUS                    ESTATUS
PL  008400     LU; CPEDI.ESTATUS                    (CASTA)
PL  008500     OW; GLB.STATUS          NOT =        GSD-ALTER
PL  008600         MV; CASTA.DESCSTATUS             DESCSTA
PL  008700     END;
PL  008800     LU; CPEDI.CVEAGENADU                 (CAAGE)
PL  008900     OW; CPEDI.CVEAGENADU         =       CAAGE.CVEAGENADU
PL  009000         MV; CAAGE.NOMAGEADU              NOMBRE1
PL  009100     END;
PL  009200     LU; CPEDI.CVEPAISCR                  (CAPAI)
PL  009300     OW; CPEDI.CVEPAISCR          =       CAPAI.CVEPAIS
PL  009400         MV; CAPAI.NOMPAIS                NOMBRE2
PL  009500     END;
PL  009600     LU; CPEDI.CVEPAISAP                  (CAPAI)
PL  009700     OW; CPEDI.CVEPAISAP          =       CAPAI.CVEPAIS
PL  009800         MV; CAPAI.NOMPAIS                NOMBRE3
PL  009900     END;
PL  010000     DO; TC.DAYNUMBER                     CPEDI.FECTIPCAMB           ::
PL  010100     MV; GLB.TOTAL                        SD-FETICA                  ::
PL  010200     LU; SD-FETICA                        (TABGU)                    ::
PL  010300     OW; SD-FETICA                =       TABGL.FECTIPOC             ::
PL  010400         MV; TABGL.TIPOCAMBIO             TIPOCAMBIO
PL  010500     END;
PL  010600     LU; CPEDI.CVEMAIS                    (CAIME)
PL  010700     OW; CPEDI.CVEMAIS            =       CAIME.CVEMAIS
PL  010800         MV; CAIME.NOMIE                  NOMBRE4
PL  010900     END;
PL  011000     MV; KCAPEDPAS                        SD-KEY
PL  011100     MV; GD-ADUAN                         SD-ADUANA
PL  011200     MV; CPEDI.CVEVISTA                   SD-CVISTA
PL  011300     LU; SD-CAVIS                         (CAVIS)
PL  011400     OW; SD-CAVIS                 =       CAVIS.KVISTA
PL  011500         MV; CAVIS.NOMVISTA               NOMBRE5
PL  011600     END;
PL  011700     OW; CPEDI.BAJA               =       GSD-UNO
PL  011800         MV; GD-MSGNO05                   GSD-NUMER
PL  011900         MV; GD-MSG005                    GSD-TEXTO
PL  012000         MV; GSD-MEN                      TEXTO
PL  012100         RECALL; (CPEDI)
PL  012200     END;
PL  012300     OW; CPEDI.BAJA          NOT =        GSD-UNO
PL  012400         MV; GD-MSGNO04                   GSD-NUMER
PL  012500         MV; GD-MSG004                    GSD-TEXTO
PL  012600         MV; GSD-MEN                      TEXTO
PL  012700         RECALL; (CPEDI)
PL  012800     END;
PL  012900  END;
PL  013000  OW; MAINT           =         (CHG)     OR
PL  013100  OW; MAINT           =         (DEL)
PL  013200     ME; ERROR (SOLO SE PERMITE ADD E INQ)
PL  013300  END.EXIT;


LG  000100  OW; MAINT                    =          (ADD)
LG  000200      DO; SD-KCAPED    OR;
```

ISPEC CFEDI - COMPONENT

```
LG  CC0300          SD; SCREDIT            [illegible]   [illegible]
LG  CC0400          SD; SCADUAN            [illegible]   [illegible]
LG  CC0500          SD; SCUECI             [illegible]   [illegible]
LG  CC0600      ES;
LG  CC0700      SD; SD-ACCUSEC    CW;
LG  CC0800          SD; SCPEDI             [illegible]   [illegible]
LG  CC0900          SD; SCADSE             [illegible]   [illegible]
LG  CC1000      ES;
LG  CC1100      SD; SD-ACCUSEC1   CW;
LG  CC1200          SD; SCADSE1            [illegible]   [illegible]
LG  CC1300      ES;
LG  CC1400      SD; SD-NLVPED              [illegible]   [illegible]
LG  CC1500      SD; SD-REGLNT              [illegible]   [illegible]
LG  CC1600      SD; SD-UNA                 [illegible]   [illegible]
LG  CC1700      DW; CVEAGEADU    =   GLB.SPACES
LG  CC1800          MES ERROR (?SE REQUIERE LA CLAVE DEL AGENTE ADUANAL)
LG  CC1900      END.EXIT;
LG  CC2000      DW; MEDTRANS     =   GLB.SPACES
LG  CC2100          MES ERROR (?SE REQUIERE EL MEDIO DE TRANSPORTE)
LG  CC2200      END.EXIT;
LG  CC2300      DW; BANTRANSP    =   GLB.SPACES
LG  CC2400          MES ERROR (?SE REQUIERE LA BANDERA DEL TRANSPORTE)
LG  CC2500      END.EXIT;
LG  CC2600      DW; CVEPAISOR    =   GLB.SPACES
LG  CC2700          MES ERROR (?SE REQUIERE LA CLAVE DEL PAIS DE ORIGEN)
LG  CC2800      END.EXIT;
LG  CC2900      DW; CVEPAISPR    =   GLB.SPACES
LG  CC3000          MES ERROR (?SE REQUIERE LA CLAVE DEL PAIS DE PROCEDENCIA)
LG  CC3100      END.EXIT;
LG  CC3200      DW; FECTIPCAMB   =   GLB.ZEROS
LG  CC3300          MES ERROR (?SE REQUIERE LA FECHA DEL TIPO DE CAMBIO)
LG  CC3400      END.EXIT;
LG  CC3500      DW; CVERNIE      =   GLB.SPACES
LG  CC3600          MES ERROR (?SE REQUIERE LA CLAVE DEL A.L.A.D.I.)
LG  CC3700      END.EXIT;
LG  CC3800      DW; NIVCOM       =   GLB.SPACES
LG  CC3900          MES ERROR (?SE REQUIERE EL NIVEL COMERCIAL)
LG  CC4000      END.EXIT;
LG  CC4100      DW; NIVCOM  NOT =   GLB.SPACES
LG  CC4200          DW; NIVCOM        NOT =   GSD-P   AND
LG  CC4300          DW; NIVCOM        NOT =   GSD-C   AND
LG  CC4400          DW; NIVCOM        NOT =   GSD-F   AND
LG  CC4500          DW; NIVCOM        NOT =   GSD-O
LG  CC4600              MES ERROR (?NIVEL COMERCIAL DEBE SER "P","C","F","O")
LG  CC4700          ES;
LG  CC4800      END;
LG  CC4900      DW; NIVCOM        =              GSD-F
LG  CC5000          LU; CVERNIE                  (CNIRED)
LG  CC5100          DW; CNIRE.NIVCOMF NOT = GSD-SI
LG  CC5200              MES ERROR (?INDICADOR DE NIVEL COMERCIAL IGUAL A NO)
LG  CC5300          ES;
LG  CC5400      END;
LG  CC5500      DW; NIVCOM        =              GSD-O
LG  CC5600          LU; CVERNIE                  (CNIRED)
LG  CC5700          DW; CNIRE.NIVCOMO NOT = GSD-SI
```

ISPEC CFED1 - COMPONENT

```
            MES ERROR (INDICADOR DE NIVEL [illegible] IGUAL A NO)
        EE;
    END;
    CA; NIVCOM     =              [illegible]
        LU; CVENIE                (CAIPE)
        CA; CAIPE.NIVCOM NOT = [illegible]
            MES ERROR (INDICADOR DE NIVEL COMERCIAL IGUAL A NO)
        EE;
    END;
    CA; NIVCOM     =              [illegible]
        LU; CVERNIE               (CAIPE)
        CA; CAIPE.NIVCOM NOT = SSD-[illegible]
            MES ERROR (INDICADOR DE NIVEL COMERCIAL IGUAL A NO)
        EE;
    END;
    CA; VALFACTU     =   GLB.ZERO
        MES ERROR (SE REQUIERE EL VALOR DE LA FACTURA)
    END.EXIT;
    CA; NUMFACTUS    =   GLB.ZERO
        MES ERROR (SE REQUIERE EL NUMERO DE FACTURAS)
    END.EXIT;
    CA; NOMPROVE     =   GLB.SPACES
        MES ERROR (SE REQUIERE EL NOMBRE DEL PROVEEDOR)
    END.EXIT;
    CA; DOMPROV      =   GLB.SPACES
        MES ERROR (SE REQUIERE EL DOMICILIO DEL PROVEEDOR)
    END.EXIT;
    CA; CVEVISTA     =   GLB.ZERO
        MES ERROR (SE REQUIERE LA CLAVE DEL VISTA ADUANAL)
    END.EXIT;
    CA; LOCFISMERC   =   GLB.SPACES
        MES ERROR (SE REQUIERE LA LOCALIZACION FISICA)
    END.EXIT;
    CA; REGADUA      =   GLB.SPACES
        MES ERROR (SE REQUIERE EL REGIMEN ADUANERO)
    END.EXIT;
    CA; CONFIRMA  NOT = (SI)     AND
    CA; CONFIRMA  NOT = (NO)
        MES ERROR (EL INDICADOR DE ACTUALIZACION DEBE SER SI/NO)
    EE;

    MV; KCAPEDPAS                 SD-KCAPED
    MV; SD-KCAPED                 SD-ADLSEC
    MV; SCADSE                    SD-ADLSEC1
    LU; SD-ADLSEC1                (CASEC)
    MV; CASEC.NUMPEDIPEN          SD-NUMPED
    AD; SSD-UNO                   SD-NUMPED
    MV; SD-NUMPED                 SD-PEDIM
    MV; SD-KCAPED                 KCAPEDPAS
    MV; CASEC.FOLPSGENT           SD-PSGENT
    AD; SSD-UNO                   SD-PSGENT
    MV; SD-PSGENT                 FOLPSGENT
    MV; CAAGE.NOMAGEADU           NOMAGE
    LU; PROA CVEPAISOR            (CAPAIS)
        EX;
```

ISPEC CPEDI - COMPONENT

```
          ENDJ
          CMJ GLB.STATUS       = GSD-ASTOK       OR
          CMJ CVEPAISOR   NOT = CAPAI.CVEPAIS
              MEJ ERROR (?CLAVE DE PAIS ORIGEN INVALIDA)
          EEJ
          MVJ CAPAI.NOMPAIS               NOMBRE2
          LUJ FROM CVEPAISPR              (CAPAI)
              EXJ
          ENDJ
          CMJ GLB.STATUS       = GSD-ASTOK       OR
          CMJ CVEPAISPR   NOT = CAPAI.CVEPAIS
              MEJ ERROR (?CLAVE DE PAIS PROCEDENCIA INVALIDA)
          EEJ
          MVJ CAPAI.NOMPAIS               NOMBRE3
          DCJ TC.DAYNUMBER                FECTIPCAMB
          CMJ GLB.STATUS       =          GSD-ASTOK
              MEJ ERROR (?FECHA TIPO DE CAMBIO INVALIDA)
          EEJ
          MVJ GLB.TOTAL                   SD-FETICA
          LUJ SD-FETICA                   (TABGL)
          CMJ SD-FETICA        =          TABGL.FECTIPCC
              MVJ TABGL.TIPOCAMBIO        TIPOCAMBIO
          ENDJ
          MVJ KCAPEDPAS                   SD-KEY
          MVJ SD-ADUAN                    SD-ADUANA
          MVJ CVEVISTA                    SD-CVISTA
          LUJ SD-CAVIS                    (CAVIS)
          MVJ CAVIS.NOMVISTA              NOMBRE5
          MVJ CAIME.NOMIE                 NOMBRE6
          DCJ TC.DAYNUMBER                INPUT-DATE
          DCJ TC.DATE                     SD-DMA
          MVJ SD-DMA                      FECPEDIM
          RCJ (CPEDI)
          CMJ CONFIRMA         =          GSD-SI
              MVJ (10)                    ESTATUS
              FLJ SD-NUMPED               CASEC.NUMPEDIMEN
              FLJ SD-REGENT               CASEC.FOLREGENT
              MVJ SD-KCAPED               GSD-KCAPED
              MVJ VALFACTU                GSD-VALFAC
              MVJ SD-FETICA               GSD-FTIPCAM
              MVJ GSD-WORK                GLB.WORK
              RCJ GC-DEPED
              MVJ GLB.SPACES              GLB.ERROR
          ENDJ
      ENDJ
```

LOCAL ITEMS

DATA: ACCION (3)   EDS A LES 3 DADS ACCION

SYNONYMS   ACCION

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CUTKE DEPED

DATA: ADMONADUA (5)   EDS A LES 40 DADS ADMON DE ADUANA

SYNONYMS   ADMINISTRADOR DE ADUANA

DATA: ADVALOREM (7)   EDS A LES 15 DADS ADVALOREM

SYNONYMS   ADVALOREM

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: BAJA (8)   EDS N LES 1 DADS BAJA

SYNONYMS   BAJA DE UN REGISTRO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAADU CAAGE CAFRA CAIME CAPAI CASEC CASTA CAVIS CPEDI DEPED
TAEDU TASIS

DATA: BANTRANSP (9)   EDS A LES 13 DADS BANDERA TRANSPOR

SYNONYMS   BANDERA DE TRANSPORTE

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: CANTBULTOS (14)   EDS N LES 6 DADS CANTIDAD BULTOS

SYNONYMS   CANTIDAD DE BULTOS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA: CANTMERC (15)   EDS N LES 6 DADS CANT MERCANCIA

SYNONYMS   CANTIDAD DE MERCANCIA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED   NT MS

DATA: CLASE (20)   EDS A LES 30 DADS CLASE DESCRIP

LOCAL ITEMS

SYNONYMS [illegible]

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA; CONFIRMA (22) EDS A LEN 2 DADS CONFIRMACION

SYNONYMS CONFIRMACION CAPEL

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA; CVEADUANA (25) EDS N LEN 4 DADS CLAVE ADUANA

SYNONYMS CLAVE DE LA ADUANA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAADU

DATA; CVEAGEADU (25) EDS N LEN 5 DADS CVE AGENTE ADUA

SYNONYMS CLAVE AGENTE ADUANA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAAGE CPEDI

DATA; CVEPAIS (27) EDS N LEN 3 DADS CLAVE DEL PAIS

SYNONYMS CLAVE DEL PAIS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAPAI

DATA; CVEPAISOR (23) EDS N LEN 3 DADS CVE PAIS ORIGEN

SYNONYMS CLAVE DEL PAIS ORIGEN

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA; CVEPAISPR (27) EDS N LEN 3 DADS CVE PAIS PROCEDE

SYNONYMS CLAVE DE PAIS PROCEDENCIA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

LOCAL ITEMS

DATA: CVE-RNE (30) EDS A LES 13 DADS CVE RNIE

SYNONYMS CVE REGISTRO NACIONAL IMP/EXP

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAIMI CPEDI

DATA: CVESECCION (31) EDS N LES 4 DADS CLAVE SECCION

SYNONYMS CLAVE DE SECCION

DATA: CVESTATUS (32) EDS N LES 2 DADS CLAVE STATUS

SYNONYMS CLAVE STATUS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CASTA

DATA: CVEVISTA (33) EDS N LES 5 DADS CLAVE VISTA

SYNONYMS CLAVE DE LA VISTA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: DESCORTA (35) EDS A LES 20 DADS DESCRIP. CORTA

SYNONYMS DESCRIPCION CORTA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAFRA DEPED

DATA: DESC-SECC (36) EDS A LES 40 DADS DESCRIPCION SECC

SYNONYMS DESCRIPCION SECCION

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CASEC

DATA: DESCSTATUS (37) EDS A LES 40 DADS DESCRIP STATUS

SYNONYMS DESCRIPCION DEL STATUS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CASTA

340

DATA: DESLARGA1 (38) EDS A LES 40 DADS DESCRIP LARGA 1

LOCAL ITEMS

SYNONYMS DESCRIPCION LARGA 1ER. RENGLON

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAFRA

DATA: DESLARGA2 (39) EDT: A LEF 40 DAD: DESCRIP LARGA 2

SYNONYMS DESCRIPCION LARGA 2DO.RENGLON

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAFRA

DATA: DESLARGA3 (40) EDT: A LEF 40 DAD: DESCRIP LARGA 3

SYNONYMS DESCRIPCION LARGA 3ER. RENGLON

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAFRA

DATA: DESLARGA4 (41) EDT: A LEF 40 DAD: DESCRIP LARGA 4

SYNONYMS DESCRIPCION LARGA 4TO.RENGLON

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAFRA

DATA: DESLARGA5 (42) EDT: A LEF 40 DAD: DESCRIP LARGA 5

SYNONYMS DESCRIPCION LARGA 5TO. RENGLON

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAFRA

DATA: DIRIE (44) EDT: A LEF 50 DAD: DIRECC IMP/EXP

SYNONYMS DIRECCION IMPORTADOR/EXPORTAD

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAIME

DATA: ESTATUS (46) EDT: A LEF 2 DAD: STATUS PEDIMENTO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDA CSTAT

DATA: FECFACT (47) EDT: A LEF 6 DAD: FECHA FACTURA

SYNONYMS FECHA DE LA FACTURA

LOCAL ITEMS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA; FECHACAMB (46) ED; N LE; 6 DAD; FECHA CAMBIO

SYNONYMS FECHA CAMBIO DEL REGISTRO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAADU CAAGE CAFR: CAIME CAPAI CASEC CESTA CAMIS CMEDI DSTAT
DEPED TAEDU TADIT

DATA; FECPARSIS (49) ED; 6 LE; 6 DAD; FECHA PARA SIS

SYNONYMS FECHA PARAMETRO DE SISTEMA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
TASIS

DATA; FECPEDIM (50) ED; N LE; 6 DAD; FECHA PEDIMENTO

SYNONYMS FECHA DEL PEDIMENTO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CMEDI

DATA; FECTIPCAMB (51) ED; N LE; 6 DAD; FECHA TIPO CAMBI

SYNONYMS FECHA TIPO DE CAMBIO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CMEDI

DATA; FECTIPOC (52) ED; N LE; 6 DAD; FECHA DE CAMBIO

SYNONYMS FECHA TIPO DE CAMBIO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
TAEDU

DATA; FLAGREPED (53) ED; N LE; 1 DAD; FLAG REPORTE PED

SYNONYMS BANDERA REPORTE PEDIMENTO

DATA; FLETES (54) ED; N LE; 10 DAD; FLETES

SYNONYMS IMPORTE POR FLETE

LOCAL ITEMS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEUI

DATA; FOLREGENT (55) EDS N LEN 9 DADS FOLIO REG ENTRADA

SYNONYMS FOLIO REGISTRO DE ENTRADA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CASEC CPEUI

DATA; FRACARANCE (56) EDS A LEN 12 DADS FRACC ARANCELARIA

SYNONYMS FRACCION ARANCELARIA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA; HORACAMB (59) EDS N LEN 4 DADS HORA CAMBIO

SYNONYMS HORA DE CAMBIO DEL REGISTRO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAADU CAAGE CAFRA CAIME CAPAI CALEC CASTA CAVIC CPEUI CSTAT
DEPED TAEQU TASIS

DATA; IMPUESTO1 (60) EDS N LEN 15 DADS IMPUESTO 1

SYNONYMS IMPUESTO "1"

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA; IMPUESTO2 (61) EDS N LEN 15 DADS IMPUESTO 2

SYNONYMS IMPUESTO "2"

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA; INDREPLEG (62) EDS A LEN 2 DADS INDIC REPRE LEGA

SYNONYMS INDICADOR REPRESENTANTE LEGAL

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAAGE

DATA; IVA (63) EDS N LEN 15 DADS IVA PAGIMENTO

LOCAL ITEMS

SYNONYMS IVA DEL PEDIMENTO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: LOCFISMERC (66) EDT A LEN 20 DADS LOC FISICA MERCA

SYNONYMS LOCALIZACION FISICA MERCANCIA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: MARCAS (67) EDT A LEN 10 DADS MARCAS

SYNONYMS MARCAS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: MEDTRANSP (68) EDT A LEN 20 DADS MEDIO TRANSPORTE

SYNONYMS MEDIO DE TRANSPORTE

DATA: MILLAR (71) EDT N LEN 15 DADS MILLAR

SYNONYMS MILLAR

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: NIVCOMC (72) EDT A LEN 2 DADS NIVEL COMER. C

SYNONYMS NIVEL COMERCIAL C

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAIME

DATA: NIVCOMO (73) EDT A LEN 2 DADS NIVEL COMER. O

SYNONYMS NIVEL COMERCIAL O

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAIME

DATA: NIVCOMP (74) EDT A LEN 2 DADS NIVEL COMER. P

SYNONYMS NIVEL COMERCIAL P

LOCAL ITEMS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAIME

DATA: NIVCOME (75) EDT A LEN 6 DADS NIVEL COMERC. X

SYNONYMS NIVEL COMERCIAL X

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAIME

DATA: NOMADUANA (76) EDT A LEN 40 DADS NOMBRE ADUANA

SYNONYMS NOMBRE DE LA ADUANA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAADU

DATA: NOMAGEADU (77) EDT A LEN 40 DADS NOM AGENTE ADUAN

SYNONYMS NOMBRE AGENTE ADUANAL

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAAGE

DATA: NOMIE (78) EDT A LEN 40 DADS NOMBRE IMP/EXP

SYNONYMS NOMBRE IMPORTADOR/EXPORTADOR

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAIME

DATA: NOMPAIS (79) EDT A LEN 40 DADS NOMBRE DEL PAIS

SYNONYMS NOMBRE DEL PAIS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAPAI

DATA: NOMPROVE (80) EDT A LEN 40 DADS NOMBRE PROVEEDOR

SYNONYMS NOMBRE DEL PROVEEDOR

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: NOMVISTA (81) EDT A LEN 40 DADS NOMBRE VISTA

SYNONYMS NOMBRE DEL VISTA

LOCAL ITEMS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAVIS

DATA: N-AUTTRSIM (33) ED: A LE: 5 DAD: NUM AUTOR TRAM

SYNONYMS NUM AUTORIZACION TRAMITE SIMPL

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: NUMERO (33) ED: A LE: 5 DAD: NUMERO

SYNONYMS NUMERO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA: NUMFACT (34) ED: N LE: 8 DAD: NUM DE FACTURA

SYNONYMS NUMERO DE LA FACTURA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DUPED

DATA: NUMFACTUS (35) ED: N LE: 4 DAD: NUM FACTURAS

SYNONYMS NUMERO DE FACTURAS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: NUMORDEN (36) ED: N LE: 4 DAD: NUMERO ORDEN

SYNONYMS NUMERO DE ORDEN

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA: NUMPEDIMEN (37) ED: N LE: 9 DAD: NUM PEDIMENTO

SYNONYMS NUMERO DE PEDIMENTO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CASIC

DATA: NUMPERSECO (38) ED: A LE: 15 DAD: NUM PERMISO SECO

LOCAL ITEMS

SYNONYMS [illegible]

THIS ITEM IS USED IN THE FOLLOWING [illegible]/REPORTS
[illegible] DEPED

DATA[illegible] PESO [illegible] (90) [illegible]

SYNONYMS [illegible]

THIS ITEM IS USED IN THE FOLLOWING [illegible]/REPORTS
DEPED

DATA[illegible] PESONETO (91) [illegible] PESO NETO

SYNONYMS PESO NETO

THIS ITEM IS USED IN THE FOLLOWING [illegible]/REPORTS
DEPED

DATA[illegible] PRECIODET (92) [illegible]

SYNONYMS PRECIO DETALLE

THIS ITEM IS USED IN THE FOLLOWING [illegible]/REPORTS
[illegible]

DATA[illegible] PROAIMO (93) [illegible]

SYNONYMS CAMBIOS DE PANTALLA

THIS ITEM IS USED IN THE FOLLOWING [illegible]/REPORTS
[illegible]
[illegible]

DATA[illegible] REGADUA (97) [illegible] 1 [illegible] REGIMEN ADUANERO

SYNONYMS REGIMEN ADUANERO

THIS ITEM IS USED IN THE FOLLOWING [illegible]/REPORTS
[illegible]

DATA[illegible] REGINSCAJA (98) [illegible] REG INSPECC CAJA

SYNONYMS REGISTRO DE INSPECC DE CAJA

THIS ITEM IS USED IN THE FOLLOWING [illegible]/REPORTS
[illegible]

LOCAL ITEMS

DATA: PFCAD[illegible]NAD (99) EDA [illegible]

SYNONYMS [illegible]

DATA: RFCIE (100) [illegible]

SYNONYMS REGISTRO FEDERAL CAUS IMP/EXP

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAIME

DATA: RFCVISTA (101) EDA A LEA 13 DADA RFC VISTA

SYNONYMS RFC DEL VISTA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAVIS

DATA: SECUENCIA (103) EDA N LEA 2 DADA NUMERO SECUENCIA

SYNONYMS NUMERO DE SECUENCIA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPEL

DATA: SUBFRACC (105) EDA A LEA 4 DADA SUBFRACCION

SYNONYMS SUBFRACCION

DATA: TADVALOREM (106) EDA N LEA 5 DEA 2 DADA TASA ADVALOREM

SYNONYMS TASA ADVALOREM

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAFPA DEPEL

DATA: TEXTO (109) EDA A LEA 74 DADA MENSAJE DE ERROR

SYNONYMS MENSAJE DE ERROR

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CAADU CAASE CAFPA CAIME CAPAI CAUEC CASTA CAVIS CFPUI DSTAT
DEPED MENU MENUP TADGU TADIS

DATA: TIPOCAMBIO (110) EDA N LEA 8 DEA 2 DADA TIPO CAMBIO

SYNONYMS TIPO DE CAMBIO

LOCAL ITEMS

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI TASAS

DATA: TOTAL (111)   EDT N LEN 15 DADS TOTAL PEDIMENTO

SYNONYMS   TOTAL DEL PEDIMENTO

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: UNIDMEDID (112)   EDT N LEN 17 DADS UNIDAD DE MEDIDA

SYNONYMS   UNIDAD DE MEDIDA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA: VALBASE (113)   EDT N LEN 15 DADS VALOR BASE

SYNONYMS   VALOR BASE

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: VALBASE5 (114)   EDT N LEN 15 DADS VALOR BASE 5%

SYNONYMS   VALOR BASE DEL 5%

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: VALCOMER (115)   EDT N LEN 15 DADS VALOR COMERCIAL

SYNONYMS   VALOR COMERCIAL

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
DEPED

DATA: VALFACTU (116)   EDT N LEN 15 DADS VALOR FACTURA

SYNONYMS   VALOR DE LA FACTURA

THIS ITEM IS USED IN THE FOLLOWING ISPECS/REPORTS
CPEDI

DATA: VALNORMAL (117)   EDT N LEN 15 DADS VALOR NORMAL

SYNONYMS   VALOR NORMAL

REPORT: LDL GENERATED: 16 MAY 83 16:22

LINCOLN SPECIFICATION LISTING FOR DATABASE ADVENTURE DICTIONARY ITEMS

LOCAL ITEMS

THIS ITEM IS USED IN THE FOLLOWING SPECS/REPORTS
DEPOT

VII.- IMPACTO EN EL DESARROLLO DE SISTEMAS USANDO LENGUAJES DE CUARTA GENERACION

VII. IMPACTO EN EL DESARROLLO DE SISTEMAS CON LENGUAJES DE 4a. GENERACION.

INTRODUCCION.

La razón principal para la utilización de los Lenguajes de Cuarta Generación, es incrementar la productividad en la construcción de aplicaciones. Los resultados experimentados para alcanzar este objetivo han sido de lo más variado, de acuerdo al tipo de sistema implementado, y las técnicas utilizadas en el manejo del proyecto y a la experiencia y capacidad del personal involucrado. En algunos casos el uso de los Lenguajes de 4a. Generación no han reportado incremento alguno en la productividad, o bien éste ha sido muy pequeno debido a que las técnicas de manejo aplicadas han sido inapropiadas o porque los usuarios de los lenguajes de 4a. Generación carecen de una metodología adecuada de diseno.

Es común encontrar un programa de un Lenguaje de 4a. Generación para un sistema moderadamente complejo que tenga un rago de 1 a 50 o 1 a 10 de número de líneas de código equivalentes a las que un programa en COBOL pudiera tener. Un programador en COBOL escribe en promedio, alrededor de 20 líneas de código por día (esto es, el número total de líneas de COBOL en el sistema final, divido entre el número total de días de programación, es alrededor de 20, incluyendo depuración y reescritura).

Un programador típico de Lenguaje de 4a. Generación obtiene mejores resultados, frecuentemente alcanza de 40 a 100 líneas de código cuando la unidad de medida es la misma (número de líneas en COBOL). Muchas veces es sumamente complicado usar la misma unidad de medida, porque el Lenguaje de 4a. Generación puede usar técnicas diferentes, como el llenado de paneles en pantalla.

VII.1 PROPORCION ENTRE COBOL - 4a. GENERACION.

La proporción COBOL 4a. Generación varía considerablemente con la naturaleza de la aplicación. Si el programa de aplicación consiste principalmente en generación de reportes, uso de menús y pantallas de captura, y acceso a bases de datos, el Lenguaje de 4a. Generación lo hará mucho mejor que el COBOL. Si la aplicación está basada principalmente en estructuras lógicas, como rutinas anidadas, iteraciones, estructuras "CASE", etc., con poco uso de reportes, pantallas y accesos a bases de datos, disminuirá la proporción COBOL - 4a. Generación.

En el uso de los Lenguajes de 4a. Generación, es de suma importancia el contar con una metodología para el desarrollo, adecuada en especial para aquellos sistemas que incluyen programas muy largos (alrededor del equivalente a 200,000 líneas de COBOL), ya que estas metologías pueden ser completamente diferentes a las del desarrollo tradicional de COBOL o PL/1.

La construcción de sistemas con el ciclo de vida tradicional de la figura 7.1, muestra la relación del número de personas necesarias a través de las etapas del proyecto de programación.

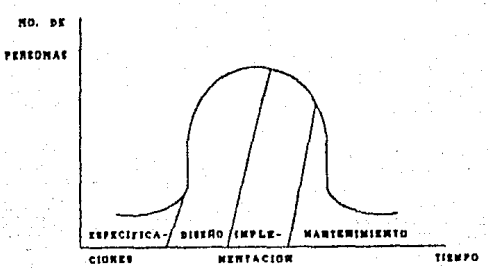


FIGURA 7.1 RELACION PERSONAS/TIEMPO EN EL CICLO DE VIDA TRADICIONAL.

El uso de un lenguaje diferente de programación incluso los de 4a. Generación utilizando la misma metodología tradicional tan solo afecta a las etapas de implementación (con sus correspondientes pruebas y mantenimiento), dentro del ciclo de vida. Este efecto es mostrado en la figura 7.2. Si el lenguaje genera reportes, pantallas, diálogos y acceso eficientes a una base de datos, el progreso es mayor, sin embargo, aún no se consume mucho tiempo y se utiliza demasiado personal para cada una de las etapas.

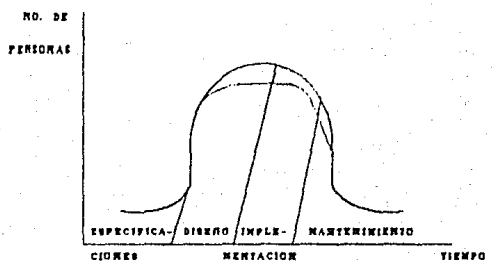


FIGURA 7.2 EFECTO DEL CICLO DE VIDA TRADICIONAL UTILIZANDO UN LENGUAJE DE CUARTA GENERACION.

Algunas herramientas de los Lenguajes de 4a. Generación se concentran en el inicio del ciclo de vida, siendo capaces de generar código desde el mismo diseño del sistema. Para esto, tanto las especificaciones de requerimientos, como el diseño deben ser sumamente claros y precisos, surgiendo así, la necesidad de una metodología enfocada a esto. El "Diseño con Prototipos" permite poner la atención que necesitan estas etapas, concentrando en ellas el esfuerzo principal tal y como se indica en la figura 7.3. donde se aprecia que el área bajo la curva es menor que el mostrado en la figura 7.2, osea que requiere menos personal y tiempo que utilizando el ciclo de vida tradicional. Pero existe otra ventaja todavía mayor el tiempo utilizado en las etapas subsecuentes (implementación y mantenimiento), disminuye considerablemente.

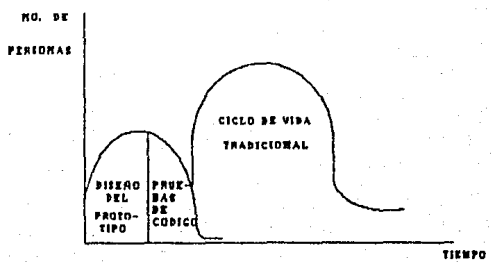


FIGURA 7.3 EFECTO DEL CICLO DE VIDA UTILIZANDO PROTOTIPOS.

Hay algunas instalaciones en las cuales han sido reportadas malas experiencias con Lenguajes de 4a. Generación, o muy pocos beneficios con respecto a su productividad. Las razones incluyen lo siguiente :

- Se necesita mucho aprendizaje para manejar algunos Lenguajes de 4a. Generación con suficiente habilidad. Las organizaciones no han invertido el dinero y tiempo necesarios para formar un grupo capaz y con práctica.

- Algunos generadores de aplicaciones están limitados en lo que pueden generar. Cuando son aplicados a un sistema especifico, pueden causar problemas o fallas en los resultados requeridos.

- Para alcanzar la mayor reducción en el tiempo de desarrollo es necesario cambiar las técnicas de manejo y control. Algunos controles apropiados para COBOL son usados con generadores de aplicaciones, por lo que se pierde mucho de su capacidad para un desarrollo rápido.

- Las caracteristicas de prototipos interactivos de la herramienta no son usados; se persiste en utilizar las especificaciones tradicionales (las cuales generalmente son inadecuadas), precindiendo de las ventajas de la herramienta.

- Algunos Lenguajes de 4a. Generación son limitados y no tienen la capacidad necesaria para desarrollar sistemas complejos.

La figura 7.4 muestra una experiencia en el desarrollo de un sistema con el Lenguaje de 4a. Generación FOCUS en un banco. En este caso se insistió en utilizar una metodología de desarrollo para un sistema formal con documentación escrita a mano.

Es interesante observar que en muchos ejemplos de utilización de Lenguajes de 4a. Generación, la proporción COBOL-4ª Generación es substancialmente mayor que el que se muestra en la figura. Una razón de 10 a 1 es un factor razonable para sistemas comerciales de tamano pequeno y mediano. La razón baja, relativamente, de la figura 7.4 pudo haber sido causada por la insistencia en utilizar una metodología más apropiada para el desarrollo con COBOL. Algunas personas con un alto nivel de habilidad en Lenguajes de 4a. Generación pueden obtener mejores resultados con mucha más rapidez.

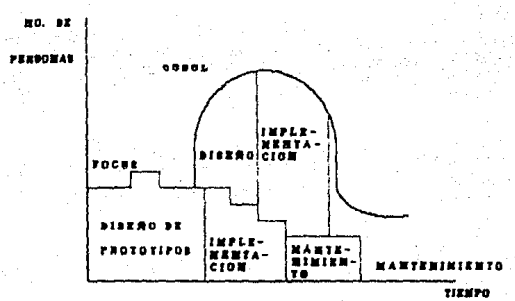


FIGURA 7.4 COMPARACION DEL DESARROLLO DE COBOL Y FOCUS EMPLEANDO METODOLOGIA FORMAL DE DISEÑO

VII.2 COMPARACION ENTRE LOS LENGUAJES DE 3a. Y 4a. GENERACION.

Es difícil encontrar una medida adecuada para evaluar la productividad en el desarrollo de sistemas.

Una medida común de evaluación en el mundo de COBOL, es el número de líneas de código por persona-día. Con esta medida se dificulta la comparación de COBOL con Lenguajes de 4a. Generación, ya que estos lenguajes presentan frecuentemente herramientas como los generadores de pantallas de captura, reportes etc., o bien se manejan por medio de menús o paneles de control en pantalla. Tan solo podría efectuarse la comparación por medio de la estimación del número de líneas de código de COBOL o personas-días requeridas con este lenguaje de 4a. Generación. Se ha intentado medir el tamaño o complejidad de las aplicaciones independientemente del lenguaje usado. La medida más común es contar las funciones liberadas por un programa, más que el volumen o complejidad del código. El hecho consiste en aislar las variables críticas que determinan la productividad. En base a esto se realizo un méto llamado "Evaluación de Puntos de Función", que consiste en contar ciertos elementos de un programa y clasificarlos por tres niveles de complejidad. Un factor de peso es aplicado al total de cada elemento. Esta técnica mide el funcionamiento de un programa más que su forma, y se utiliza para la comparación entre los programas con Lenguajes de 3a. Generación contra los Lenguajes de 4a. Generación.

En un estudio utilizando puntos de función para comparar programas de 1a. 3a. y 4a. Generación. Un punto de función fué equivalente a 114 líneas de COBOL en promedio y 14 líneas de LINC. Usando LINC, el desarrollo toma al rededor de una hora por punto de función. Con COBOL y PL/1 el promedio es alrededor de 20 horas por punto de función.

VII.3 PRODUCTIVIDAD, HERRAMIENTAS Y TECNICAS.

Para alcanzar los mejores resultados en el mejoramiento de la productividad, se necesita más que un Lenguaje de 4a. Generación. Las herramientas de diseño son importantes y el manejo del ciclo de vida debe ser adaptado al manejo de prototipos y técnicas de ingeniería de información. A continuación se listan los factores más importantes para un desarrollo de alta productividad con Lenguajes de 4a. Generación:

- Un Lenguaje estructurado de 4a. Generación que refuerce plenamente el código estructurado.
- Herramientas gráficas para diseno automatizado.
- Un diccionario en linea.
- Un modelo de datos completamente normalizado.
- Estaciones de trabajo dedicadas para el desarrollo, ligadas a un diccionario de datos central o enciclopedias.
- Desarrollo rápido de Prototipos.
- Un ciclo de vida de desarrollo iterativo.
- Auxilio de conversión computarizada de prototipos en el código final, si es necesario.
- Documentación automatizada.
- Una libreria de módulos reusables.
- Entrenamiento completo y práctica con las herramientas.
- División de programas largos en módulos pequenos, autónomos ligados a un modelo de datos (y si es posible a una enciclopedia).

En conclusión podemos decir que los Lenguajes de 4a. Generación deben utilizarse con una metodologia apropiada a sus caracteristicas. La metodologia del desarrollo con prototipos permite la óptima utilización de las herramientas y cualidades que estos lenguaje presentan. Esto permite obtener los beneficios que se han mencionado anteriormente, y asi alcanzar un veradadero incremento en la productividad.

VIII.- CONSIDERACIONES PARA LA EVALUACION DE UN LENGUAJE DE CUARTA GENERACION

VIII CONSIDERACIONES PARA LA EVALUACION DE UN LENGUAJE DE CUARTA GENERACION.

VIII.1 CRITERIOS DE SELECCION PARA LENGUAJES DE 4A. GENERACION.

Diferentes lenguajes de 4a. generación son diseñados para diversas aplicaciones.

A continuación se presenta una lista de interrogantes acerca de las aplicaciones de los lenguajes de 4a. generación.

1.- ¿Será para usuarios finales o para profesionales de procesamiento de datos?

2.- ¿Medio ambiente con especificaciones meticulosas?

3.- ¿Sistema batch o en línea?

4.- ¿Trabajo comercial o científico?

5.- ¿Soporte técnico?

6.- ¿Trabajos pesados?

7.- ¿Qué clase de computador es el adecuado?

8.- ¿Qué espacio se necesita para la base de datos?

9.- ¿Base de datos o archivos existentes?

10.- ¿Especificaciones complejas?

11.- ¿Prototipo o aplicación final?

12.- ¿Acceso por una red de cómputo?

13.- ¿Ligado a paquetes de aplicación?

14.- ¿Ligado a automatización de oficinas?

VIII.2 CATEGORIAS DE LAS FUNCIONES DE LOS LENGUAJES DE CUARTA GENERACION.

Cualquier lenguaje de 4a. generación puede ocupar más de una categoría, en la figura 8.1 los usuarios pueden compararlos y escogerlos de acuerdo a sus necesidades, estas características cambian rapidamente.

El término (Full-Function), que significa "para todo tipo de funciones", de un lenguaje de 4a. generación podrá hacer cualquier cosa a diferencia de los lenguajes de 3a. generación, y son menos poderosos, aunque algunos pueden generar reportes, dirigir preguntas a una base de datos, pero con la limitante de no poder dar ciertas facilidades que los lenguajes de 4a. generación podrían hacer.

Algunos lenguajes de 4a. generación fueron diseñados para ciertas aplicaciones, otros fueron demasiado elegantes y nada fácil de manejar. Por eso que el comprador deberá juzgar la integridad de la arquitectura de los lenguajes y la simplicidad de su sintaxis

Ahora bien, ¿puede un lenguaje de 4a. generación remplazar completamente a Cobol?. Existen dos aspectos a esta pregunta:

1.- ¿ Los lenguajes tienen funcionalidad suficiente para reemplazar a Cobol?

2.- ¿ Los lenguajes proporcionan a la máquina buena ejecución para reemplazar a Cobol?

Varios lenguajes de 4a. generación no intentan reemplazar a Cobol o lenguajes similares. Ciertos generadores de aplicaciones trabajan bien y otros no, así, de esta manera, no se puede afirmar que los lenguajes de 4a. generación reemplazan cualquier lenguaje de 3a. generación.

Ciertos lenguajes de 4a. generación tienen la funcionalidad para reemplazar a Cobol, proporcionandole a la máquina un trabajo fuerte con poca ejecución de cómputo.

UN LENGUAJE DE 4a. GENERACION IDEAL DEBERIA TENER TODAS LAS CARACTERISTICAS DE ESTA TABLA. TRES LENGUAJES SON EVALUADOS CON ESTA LISTA. (NOTE QUE LA EVALUACION PUEDE ESTAR FUERA DE TIEMPO YA QUE NUEVAS CARACTERISTICAS SON AGREGADAS A ESTOS LENGUAJES CONTINUAMENTE.	INFORMIX	LINC	NATURAL
CANCELACION DEL USO DE GO TO'S	x	x	x
LLAMADAS A SUBRUTINAS	x	x	x
BLOQUES DE SIMPLE EJECUCION	x	x	x
DO	x	x	x
IF	x	x	x
IF-ELSE	x	x	-
CASE	x	x	-
BLOQUES DE REPETICION	x	x	-
DO WHILE	x	x	x
DO UNTIL	x	x	-
FOR (RELACIONADO CON VARIABLES)	x	x	-
FOR (RELACIONADO CON ARCHIVOS O BD)	x	x	x
ESCAPE (DE UN BLOQUE)	x	x	-
ESCAPE (DE MAS DE UN BLOQUE)	x	x	-
END (PARA TODOS LOS BLOQUES)	x	x	x
BLOQUES DE LOS LOOP'S DE INICIAL.-TERMIN.	x	x	x
VARIABLE DE CONTROLACION DE LOOP'S	x	x	x
AMIGABLE AL USUARIO FINAL Y PROFESIONAL	x	x	x
MANEJADOR DE BASE DE DATOS	x	x	x
CANTIDAD REDUCIDA DE CODIGO	x	x	x
CODIGO NON-PROCEDURAL DONDE ES POSIBLE	x	x	x
OPERACION EN LINEA	x	x	x
SOPORTA TECNICA DE PROTOTIPOS	x	x	x
GENERADOR DE PANTALLAS	x	x	x
GRAFICACION PARA APLICACIONES	-	-	x
GENERADOR DE REPORTES	x	x	x
DOCUMENTACION AUTOMATICA	-	x	-
MANEJO DE MENUES	x	x	x
DICCIONARIO DE DATOS INTEGRADO	-	x	x

FIGURA 8.1. CARACTERISTICAS GENERALES DE LOS LENGUAJES DE CUARTA GENERACION.

Algunos lenguajes de 4a. generación pueden y deben ser reemplazados por los de 3a. generación, si no cumplen con los requisitos básicos de cada aplicación. Los propósitos de los lenguajes de 4a. generación son, que fueron creados para evitar el exceso de trabajo y lograr el manejo de éstos de la forma más rápida y fácil posible, por citar un ejemplo, si en un programa en Cobol se generan mil líneas de código, un lenguaje de 4a. generación simplemente generaría no más de cincuenta líneas.

VIII.3 NIVEL DE EVOLUCION DE LA SINTAXIS.

Se observan diversas características de la evolución de estos lenguajes de 4a. generación para incrementar sus dialogos, estas son:

1.- Operación en línea.

2.- Diálogos unidimensionales, que pueden ser usados con terminales parecidas a una máquina de escribir.

3.- Diálogos bidimensionales, que pueden ser usados en computadoras personales con despliege visual.

4.- Diálogos orientados a tiempos de respuesta de computadores personales, estaciones de trabajo o redes locales.

5.- Diálogos usando gráficas para diseño de sistemas y su lógica.

6.- Herramientas, usando técnicas y sistemas expertos.

VIII.4 SISTEMAS AMIGABLES AL USUARIO.

Muchos vendedores proclaman que su lenguaje de 4a. generación es "diseñado para usuarios finales". Algunos son más apropiados para programadores profesionales que para el análisis de negocios. Esto es claramente conveniente para seleccionar productos que son adecuados para usuarios finales o analistas de negocios.

El comprador tendrá que preguntar lo siguiente:

1.- Si el lenguaje es apropiado para usuarios finales, analistas de sistemas o programadores.

2.- ¿ Cuanto tiempo es conveniente que un usuario final, tome para empezar a obtener resultados (horas, días, semanas)?

3.- ¿ Está diseñado para que un usuario pueda instalarlo aprenderlo por él mismo ?

4.- ¿ La sintaxis es fácil de aprender y recordar ?

5.- ¿ Se forza al usuario a recordar mnemónicos y formatos ?

6.- ¿ Es procedural, no-procedural o ambos ?

7.- Si es procedural, ¿ se ejecuta o se crea código estructurado ?

8.- ¿ La sintaxis del lenguaje es buena para el mantenimiento?

9.- ¿ Puede operar a diferentes niveles en sus diálogos ?

10.- ¿ Puede almacenar y catalogar procedimientos de usuarios al instante ?

11.- ¿ Puede el usuario switchear al instante para realizar una operación o espacio de trabajo ?

12.- ¿ Tiene documentación clara, buena y en línea como en un manual ?

13.- ¿ Cómo es la categoría, copia y apariencia de los caracteres, reportes y gráficas generadas ?

14.- ¿ La pantalla dividida tiene capacidad de revisar datos procedimientos, reportes y mensajes de error ?

	ORIENTADO PARA USUARIOS FINALES			ORIENTADO PARA ANALISTAS DE SISTEMAS			ORIENTADO PARA PROGRAMADORES PROFESIONALES		
	LINC	INFORMIX	NATURAL	LINC	INFORMIX	NATURAL	LINC	INFORMIX	NAT.PAL
LENGUAJE DE PREGUNTAS SIMPLES	X	X	X	X	X	X	X	X	X
LENGUAJES COMPLEJOS DE PREGUNTAS Y ACT.	-	-	-	X	X	-	X	X	-
GENERADORES DE REPORTES	X	X	X	X	X	X	X	X	X
LENGUAJES GRAFICOS	-	-	X	-	-	X	-	-	X
LENGUAJES DE APOYO DE DECISIONES	-	-	X	-	-	X	-	-	X
GENERADORES DE APLICACIONES	X	X	-	X	X	X	X	X	X
LENGUAJES DE ESPECIFICACIONES	X	X	-	X	X	X	X	X	X
LENGUAJES DE PROGRAMACION DE MUY ALTO NIVEL	X	X	-	X	X	X	X	X	X
PAQUETES DE APLICACIONES PARAMETRICAS	X	X	X	X	X	X	X	X	X
LENGUAJES DE APLICACION	X	X	-	X	X	X	X	X	X

FIGURA 8.2. CATEGORIZACION POR COMPARACION DE LOS LENGUAJES DE CUARTA GENERACION. MUCHOS LENGUAJES INCLUYEN FACILIDADES EN MAS DE UNA CATEGORIA.

IX.- CONCLUSIONES

IX. C O N C L U S I O N E S

PANORAMA GENERAL.

El objetivo de nuestra tesis, como hemos expuesto a lo largo de ésta, tiene como fin la implementación de una Metodología para el Desarrollo de Sistemas que trabajen en ambientes de Cuarta Generación.

Hemos buscado en la literatura existente las técnicas y elementos que sustenten dicha metodología, obteniendo como resultado un producto final que le muestre a la gente de sistemas, las tareas a seguir para desarrollar correcta y eficientemente un sistema.

A continuación mencionamos algunas generalidades hacia las cuales está orientado el presente documento.

En términos generales, el uso de los Lenguajes de Cuarta Generación tiene como consecuencia fundamental en el desarrollo de sistemas, la obtención de resultados rápidos y en un periodo de tiempo que se considera ampliamente productivo, aunque con un mayor consumo de recursos en HARDWARE.

Dentro de los lenguajes de cuarta generación que existen, es necesario distinguir los procedurales de los que no lo son, ya que esto es determinante para saber que tanto se va a involucrar la gente, ya que, mientras más procedural sea estará más orientado a la gente de sistemas, como es el caso de LINC e INFORMIX, y si es menos procedural, podrá participar y utilizarlo el usuario, como sucede con MAPER y NATURAL, aunque tal vez convenga limitar al usuario a sólo consultas.

Los lenguajes de cuarta generación están hechos fundamentalmente para el manejo de Bases de Datos por la forma en que éstos trabajan a través de pantallas, realizando mantenimiento en la forma general del sistema (altas, bajas, etc.) así como, el pintador de pantallas, validación de ciertos campos, accesos rápidos a la información a través de la elaboración de fáciles programas de aplicación y que en un momento dado el usuario puede elaborar, como para consultas o tal vez para realizar un reporte, son características para determinar que este tipo de lenguajes están orientados a procesos en línea.

Sin embargo, también se utilizan para procesos en lote, pero por las características que presentan y que antes se mencionaron definitivamente no es eficiente usarlos para este tipo de procesos.

Así mismo, también podemos decir que no es muy recomendable usar este tipo de lenguajes para cálculos matemáticos muy complejos (operaciones con matrices, vectores etc).

Los alcances y limitaciones de los lenguajes de cuarta generación en comparación con los de tercera generación plantean grandes diferencias como hemos mencionado a lo largo de nuestro trabajo.

Mientras los lenguajes de tercera generación, necesitan muchas líneas de código para alguna aplicación específica, la elaboración del diseño la realizaba gente especializada de sistemas, consumían mucho tiempo y se dificultaba su mantenimiento; por lo contrario los lenguajes de cuarta generación necesitan pocas líneas de código para ciertas aplicaciones, a diferencia de los de tercera generación en los cuales no tenía participación directa el usuario, en los de cuarta generación si la tiene, y su mantenimiento es más fácil de hacer.

Los lenguajes de cuarta generación cuentan con la facilidad de desarrollar sistemas empleando el manejo de prototipos esto, constituye a realizar un sistema más preciso respecto a las necesidades del usuario en comparación con un desarrollo de los lenguajes de tercera generación.

El uso de lenguajes de cuarta generación exige una mayor capacitación tanto, como por parte del usuario como por parte del equipo de trabajo de sistemas, ya que por un lado necesitan conocer la filosofía del lenguaje, así como, la forma en que se debe diseñar u orientar el diseño. Es muy importante la capacitación, puesto que de otro modo se puede caer en el error de desarrollar sistemas con lenguajes de cuarta generación usándolos como si fueran de tercera generación.

La metodología propuesta, es consistente para desarrollar sistemas basándose en las técnicas actuales más usadas, esto la convierte en que sea fácil de implementar por las características propias de los lenguajes de cuarta generación. La metodología plantea y propone el uso del prototipo para que apartir de ahí, se evolucione hasta llegar finalmente al sistema deseado, el uso del prototipo y la metodología hacen que los sistemas sean dinámicos y que se ajusten rápidamente a los cambios porque está orientada primero a los datos y luego a los procesos, ya que los datos no son tan variables. Los lenguajes de cuarta generación no son nada sin una metodología, que oriente en su forma de usarlos. El objetivo principal de esta metodología es el evitar en su gran mayoría todos los problemas que se pudiesen presentar durante el desarrollo del sistema; la ventaja más importante con el uso de nuestra metodología, es que el usuario a través del desarrollo del sistema determinará exactamente cuáles son sus requerimientos a diferencia del desarrollo de un sistema en la forma tradicional, donde generalmente al concluir el sistema resultó no ser lo que el usuario necesitaba.

X. B I B L I O G R A F I A.

FOURTH-GENERATION LANGUAGES
VOLUMEN I. PRINCIPLES
JAMES MARTIN
PRENTICE HALL.

INFORMACION POR COMPUTADORAS
ROGER L. SISSON
RICHARD G. CANNING
LIMUSA-WILEY SA.

LA ADMINISTRACION DEL PROCESAMIENTO DE DATOS
ROGER L. SISSON
RICHARD G. CANNING
LIMUSA-WILEY SA.

SPL'S 4TH GENERATION SYSTEMS DESIGN.

PROCEDIMIENTOS INFORMATICOS EN SISTEMAS EMPRESARIALES
FRANK J. CLARK
RONALD GALE
ROBERT GRAY
PRENTICE HALL/INTERNATIONAL.

CONCEPTOS DE LOS SISTEMAS DE INFORMACION
PARA LA ADMINISTRACION
HENRY C.LUCAS, JR.
MC-GRAW-HILL.

STRUCTURED SYSTEMS ANALYSIS:
TOOLS AND TECHNIQUES
CHRIS GANE AND TRISH SARSON
PRENTICE-HALL, INC.(ENGLEWOOD CLIFFS, NEW JERSEY 07632)

SOFTWARE IMPLEMENTATION METHODOLOGY
BURROUGHS, CORPORATION 1986.

NATURAL 4TH GENERATION APPLICATION DEVELOPMENT SOFTWARE
SOFTWARE A.G.

ANALISIS AND DESIGN OF INFORMATION SYSTEMS
JAMES A. SENN
MC-GRAW-HILL.

LENGUAJES DE CUARTA GENERACION
JAMES MARTIN
MC-GRAW-HILL.

SIM PROFESSIONAL PROYECT PRACTICES
BURROUGHS.

PROJECT PLANNING AND CONTROL
BURROUGHS.
KEITH LONDON ASSOCIATES-1986

NATURAL
PROVEN 4TH GENERATION TECHNOLOGY
CONCEPTS & FACILITIES.

SOFTWARE ENGINEERING
AND PRACTITIONER'S APPROACH
ROGER S. PRESSMAN.
MC-GRAW-HILL

NATURAL ADVANCED TECHNIQUES
WORKSHOP SEPTEMBER 82

SQL/DATA SYSTEM
GENERAL INFORMATION FOR
VM/SYSTEM PRODUCT
IBM

LINC II CAPABILITIES OVERVIEW
RELATIVE TO RELEASE 13.0
UNISYS CO.

INFORMIX 4TH GENERATION AND INFORMIX ISQL
REFERENCE MANUAL