UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

HERRAMIENTA PARA DESARROLLO DE MODELOS DE SISTEMAS DE INFORMACION EN LINEA

T E S I S

Que para obtener el Título de:

INGENIERO EN COMPUTACION

P r e s e n t a:

Salvador Alonso Torres Parra

DIRECTOR DE TESIS:
ING. LUIS G. CORDERO BORBOA

MEXICO, D. F. 1988

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

Introducción 1

Capitulo I. USO DE MODELOS DE SISTEMAS DE INFORMACION EN LINEA EN AMBIENTES BANCARIOS.

I.1. IMPORTANCIA DE LOS MODELOS Y LA SIMULACION EN EL DESARROLLO DE SISTEMAS DE INFORMACION EN LINEA. 6

I.2. NECESIDADES DE LOS USUARIOS Y DEFICIENCIAS DE LOS PROGRAMAS EN USO ACTUALMENTE. 13

Capitulo II. ANALISIS DEL SISTEMA Y OBTENCION DEL DIAGRAMA DE FLUJO DE DATOS.

II.1 DESCRIPCION DEL ANALISIS ESTRUCTURADO. 21

II.2. ESTABLECIMIENTO DEL DIAGRAMA DE FLUJO DE DATOS. 40

Capitulo III. DISENO DEL SISTEMA Y OBTENCION DE LA CARTA DE ESTRUCTURA.

III.1. DESCRIPCION DEL DISENO ESTRUCTURADO. 51

III.2. ESTABLECIMIENTO DE LA CARTA DE ESTRUCTURA. 77

Capitulo IV. SEUDOCODIGO Y PROGRAMACION DEL SISTEMA.

IV.1. DESCRIPCION DE LA PROGRAMACION ESTRUCTURADA. 90

IV.2. ESTABLECIMIENTO DEL SEUDOCODIGO. 96

Capitulo V. DESARROLLO DE EJMPLO DE APLICACION.

V.1. CONSIDERACIONES PARA EL USO DE LA HERRAMIENTA DE CONSTRUCCION DE MODELOS DE SISTEMAS EN LINEA 168

V.2. EJEMPLO DE APLICACION DEL SISTEMA DESARROLLADO (HERRAMIENTA) EN UN SISTEMA BANCARIO DE CHEQUES DE CAJA 182

CONCLUSIONES 239

BIBLIOGRAFIA 242

I N T R O D U C C I O N

El presente trabajo, tiene como objetivo, presentar: las bases teóricas y prácticas, así como las diferentes fases de desarrollo que dieron origen a una herramienta que facilita la construcción de modelos , usando una computadora personal, de cualquier sistema en línea cuyo ambiente final sea una máquina grande.

Para elaborar la herramienta, se hizo uso de las técnicas de análisis, diseño y programación estructurada que son aceptadas por la mayoría de las organizaciones e instituciones que requieren de una base teórica formal en sus técnicas de desarrollo. Al hablar de organizaciones e instituciones, no solo se hace referencia a empresas, sino que se incluyen las instituciones de educación superior más importantes del país.

A continuación, se hace una breve descripción del sistema (herramienta) a manera de un bosquejo introductorio y de la estructura de los capítulos que forman el trabajo.

El sistema objeto de éste trabajo, consiste de una serie de programas y archivos, que basados en las características dinámicas en cuanto a creación y modificación de archivos y programas propios del paquete Revelation, permite crear modelos de un sistema en línea con el mínimo de esfuerzo y tiempo invertido por parte del especialista.

Como se mencionó en el párrafo anterior, el sistema se basó en la filosofía de Revelation y por lo tanto, se usarán ciertos módulos de éste paquete que permiten la creación y modificación dinámica de programas y archivos. El sistema está formado por una nueva programación en lenguaje R/BASIC (propio del paquete) para lograr un comportamiento muy similar al que presentan las terminales ADM 12 PLUS, LINK, IMAGO,TD830 y ET1100.

Adicionalmente, el sistema está habilitado para efectuar las tareas de validación de datos; esto es, aprovechando la otra característica importante de Revelation que consiste en declarar por medio de parámetros los patrones de validación, se desarrolló la programación necesaria para que la mencionada validación sea efectuada por el sistema, liberando al especialista de la programación de las rutinas que efectúan estas tareas. Los módulos de validación desarrollados, están programados de tal manera que se comportan como lo hacen sus equivalentes en máquinas grandes.

La organización de los capítulos que integran este trabajo, está definida de la siguiente forma:

En el Capítulo I, se manifiesta la importancia que tienen en el desarrollo de sistemas, los modelos y la simulación y a manera de justificación, las necesidades de usuarios y la deficiencia que presentan algunos programas que se encuentran actualmente en uso.

El Capitulo II, hace una breve descripción de lo que es el Análisis Estructurado y en base a esa técnica, establece un Diagrama de Flujo de Datos (D.F.D.) que se genera para el sistema.

Una breve descripción de la técnica de Diseño Estructurado y el paso del Diagrama de Flujo de Datos generado en el capitulo anterior, a la Carta de Estructura (C.E.) siguiendo la técnica de diseño estructurado caracterizan al Capitulo III.

Una vez obtenida la carta de estructura en el capitulo precedente, en éste Capitulo IV, se exponen los conceptos y teoría de la técnica de Programación Estructurada para poder obtener el seudocódigo de los programas del sistema.

Finalmente, en el Capitulo V, se presentan las consideraciones para uso de la herramienta y un ejemplo de aplicación del sistema desarrollado para sistemas en línea del ambiente bancario. Este ejemplo es una parte del sistema de computarización de sucursales que pretende reducir al mínimo la intervención humana en el mayor número de actividades posible.

C A P I T U L O I

USO DE MODELOS DE SISTEMAS DE INFORMACION EN LINEA EN AMBIENTES BANCARIOS.

I.1 IMPORTANCIA DE LOS MODELOS Y LA SIMULACION EN EL DESARROLLO DE SISTEMAS DE INFORMACION EN LINEA.

I.2 NECESIDADES DE LOS USUARIOS Y DEFICIENCIAS DE LOS PROGRAMAS EN USO ACTUALMENTE.

I.1 IMPORTANCIA DE LOS MODELOS Y LA SIMULACION EN EL DESARROLLO DE SISTEMAS DE INFORMACION EN LINEA.

Actualmente, dada la gran cantidad de información que se genera en cualquier institución o empresa, sin importar su tamaño, (desde un gran banco con un elevado número de sistemas y por consiguiente de programas, hasta en el nivel doméstico, donde se requiere manejar información de, por ejemplo, cuentas bancarias de los miembros de la familia, control de los gastos, agendas, calendarios, etc) así como la rapidéz con que dicha información se genera, se requiere que los especialistas involucrados en el desarrollo de sistemas de información, efectúen su trabajo no solo tomando en cuenta la calidad de sus sistemas, sino que además de ello contemplen que su trabajo se desarrolle de una manera cada ves más rápida.

Como se puede observar, el problema de desarrollo de sistemas, se puede considerar dividido en dos partes que pueden considerarse como las más sobresalientes.

Por un lado lo que se refiere a calidad y por otro rapidéz.

Para resolver el problema de calidad en un sistema de información, se han desarrollado técnicas de análisis, diseño y programación que consisten básicamente en una recopilación y formalización de los métodos de los cuales se han obtenido los mejores resultados, dándoles un enfoque teórico que a su vez se sustenta en los más recientes avances de las ciencias de la computación. Entre tales técnicas podemos mencionar:

- Técnicas de Análisis, Diseño y Programación Yourdon.
- Técnicas de Análisis y Diseño Warnier.
- Otras Técnicas de Programación Estructurada.

En lo que respecta a la solución a los problemas que impiden al especialista mejorar o en su caso optimizar el tiempo de desarrollo de un sistema de información, existen en la actualidad algunos paquetes o programas que reducen el tiempo en cuanto a análisis y diseño dado que dentro de sus algoritmos o programas, tienen escritas rutinas que se encargan de

ejecutar las tareas más tediosas y laboriosas que tienen en algunos de sus pasos estas técnicas, como por ejemplo: las tareas de diagramación y generación de cartas, etc.

Mediante las herramientas mencionadas anteriormente, se han computarizado y por consiguiente acelerado los procesos de análisis, diseño y programación de los sistemas de información, pero existe un factor que ha sido poco considerado y es: la poca o nula visión que puede tener el usuario de un sistema, del producto que estará a su disposición, cuando éste sea liberado. Esto es en otras palabras: Cuando el especialista en el desarrollo de sistemas, efectúa entrevistas con el usuario, se da inicio a una labor de análisis que dará como resultado un diagrama funcional del sistema mediante un diagrama de flujo de datos (D.F.D.), posteriormente, se obtienen las cartas de estructura (C.E.) que representan un diseño del sistema y finalmente, el seudocódigo para dar paso a la programación.

Al concluir ésta última fase de desarrollo, es cuando el usuario puede tener su sistema para ver los resultados que éste genera y hasta entonces, es cuando se puede percatar de los errores de interpretación, diseño, programación, etc; pero frecuentemente ya es demasiado tarde para corregirlos dado que de las primeras entrevistas a ésta última fase, han transcurrido varios meses e incluso años, dándole al trabajo realizado en esta etapa, cierta característica de ineficiencia e ineficacia; aún cuando se hallan empleado las técnicas más recientes de desarrollo de sistemas.

Finalmente, el factor descrito, se traduce también en un problema de bajo nivel de rapidéz de desarrollo ya que en ocasiones, es necesario invertir nuevamente tiempo para volver a rediseñar el sistema.

Una manera de resolver el problema expuesto, es mediante el uso de modelos de sistemas y de los paquetes de simulación.

Respecto de los primeros, existen pocas herramientas que tengan como función facilitar

al especialista el desarrollo de modelos y por consiguiente, hacer más productivo su trabajo, sin invertir demasiado tiempo.

En cuanto a paquetes de simulación, se cuenta en la actualidad con un buen número de paquetes y rutinas que permiten simular incluso ambientes en los que se manejan llegadas y salidas aleatorias como por ejemplo: la clientela de un supermercado. También se pueden encontrar paquetes de simulación de condiciones meteorológicas y muchos otros de diversos grados de sofisticación que combinados con el modelo que un especialista creó, puedan conducir a un sistema de muy buena calidad del cual el usuario pueda sentirse satisfecho en un mínimo de tiempo.

Tanto los modelos como los paquetes de simulación son de gran utilidad en sistemas de información ya sean éstos en lote ('batch') o sistemas en línea ('on-line'), pero los modelos parecen tener más impacto en los sistemas en línea ya que como sabemos, éste tipo de sistemas (en línea), se caracterizan por tener un alto grado de interacción con el usuario final por

medio de menúes y pantallas de captura y/o de consulta que serán objeto de múltiples ajustes y modificaciones antes de liberarse a producción. Esta labor de ajustes y modificaciones se puede ver facilitada e incluso puede llegar a ser bastante agradable cuando se cuenta con la ayuda de un modelo que pueda ser modificado y ajustado hasta en los más mínimos detalles para lograr la completa satisfacción tanto de las necesidades del usuario, como de las necesidades del especialista o especialistas encargados del desarrollo de un producto.

Es necesario establecer con claridad que cuando se mencionan pantallas y menúes (característicos de sistemas en línea), no se hace referencia exclusivamente a esas pantallas y menúes, sino también a las técnicas o métodos de acceso a archivos , definición de los mismos y en general a todos los elementos que toman parte en éste tipo de sistemas.

Considerando que todo lo mencionado tiene como finalidad satisfacer las necesidades de los usuarios, se incluye en éste capítulo el siguiente tópico relacionado con la discusión de dichas necesidades.

A partir de éste momento, al hacer referencia a sistemas, se estará hablando de sistemas en línea, salvo que se haga una indicación en otro sentido.

I.2 NECESIDADES DE LOS USUARIOS Y DEFICIENCIAS DE LOS PROGRAMAS EN USO ACTUALMENTE.

Como se puede observar fácilmente, en el punto I.1, la labor de un especialista en el desarrollo de sistemas, puede ser sometida a una analogía con la labor de un Arquitecto o Ingeniero Civil, pero existe una gran diferencia en lo que a herramientas disponibles se refiere para cada una de las dos disciplinas mencionadas.

Los constructores de casas o edificios tienen como herramientas disponibles, por ejemplo, las maquetas para que el usuario o cliente de sus productos (casas o edificios) pueda tener una idea concreta de lo que va a comprar, pero en el caso de los constructores de sistemas, no se cuenta con alguna herramienta que permita o facilite la construcción de un modelo que pueda mostrar o por lo menos de una idea al usuario del producto que va a comprar (sistemas).

En la actualidad, se hace uso de los documentos que se van generando sucesivamente en las diferentes etapas de desarrollo de un sistema, para explicar o tratar de explicar al usuario como será el producto que se le está construyendo, pero aún cuando los documentos generados tengan muy pocos o ningún término técnico de los usados en el ambiente de la computación, es muy difícil que se logre este objetivo.

Esta situación sería comparable a que un arquitecto o ingeniero de construcción tratara de explicar a su cliente como será la casa que va a construirle proporcionándole exclusivamente un plano y una lista donde se describan los acabados, distribuciones y materiales usados en la elaboración de la casa.

Como resulta evidente, siempre es necesario que el cliente o usuario tenga una visión de lo que va a comprar y que ésta sea presentada antes de iniciar la construcción para que los posibles errores se detecten en las primeras fases de desarrollo, logrando con esto que cualquier cambio que se requiera sea

efectuado sin tener que invertir grandes cantidades (en materiales, tiempo, dinero, etc).

Se ha encontrado que el corregir errores cuando el sistema está en producción, representa un incremento en el costo de un 1000 %.

Continuando con la analogía que se hace respecto de los constructores de casas y de los constructores de sistemas, se puede considerar que así como el cliente de un arquitecto puede construir junto con éste la casa, modificando planos o quitando y poniendo habitaciones, paredes o cambiando la distribución de la casa (obviamente trabajando sobre la maqueta), el usuario de un sistema debe tener la facilidad de trabajar conjuntamente con el especialista encargado del sistema para sugerir y en su momento efectuar las modificaciones que se requieran sobre el sistema en cuestión.

En los últimos cinco años, se han venido desarrollando programas y paquetes que soportan frecuentes alteraciones sobre el diseño, entre los cuales se pueden mencionar:

- Bases de datos relacionales con lenguajes de consulta y generadores de reportes.

- Interfases de pantalla a este tipo de bases de datos.

- Diccionario de datos activo integrado que coordina, valida y monitorea las definiciones de cambio más rápidamente.

- Lenguaje de cuarta generación que con comandos como FIND, SORT y TOTAL, ejecuta una función de base de datos que tomaría varios comando o subrutinas en un lenguaje de tercera generación.

Estos programas y paquetes desarrollados, como se puede observar tienen efecto sobre el sistema y no sobre un modelo del sistema.

De lo anterior se puede concluir que la mayor necesidad del usuario en éste momento radica en la ausencia de modelos de su sistema final.

Por otro lado, dado que el especialista en desarrollo es la persona que atiende a las necesidades del usuario, se requiere entonces que exista una herramienta que le permita la construcción de modelos de una manera rápida para que una ves que se obtenga, se trabaje sobre él hasta llegar a una solución que satisfaga al usuario y hasta entonces iniciar la programación del sistema.

Los programas que actualmente se encuentran en uso, presentan ciertas deficiencias que son originadas por un mal diseño del sistema al que pertenecen o por el desconocimiento por parte del personal de desarrollo y mantenimiento de las técnicas de análisis, diseño y programación estructurada, entre tales deficiencias podemos citar:

- La falta de una estructura modular en los programas de tal manera que permitan ser leidos con claridad y facilidad por una persona que no esté familiarizada con el sistema. Como consecuencia de la no existencia de una estructura modular, se presenta un gran incremento en los costos de mantenimiento de un programa y por consiguiente del sistema.

- La escasa o nula flexibilidad de los programas en operación que implica en ocasiones la necesidad de reescribir el código y de escribir nuevos programas para poder atender algunos requerimientos nuevos y por lo tanto, aumenta el tiempo de respuesta del area de mantenimiento al usuario.

- La falta de uso de nombres mnemónicos en las variables de un programa que hace que aún el programador más experimentado pueda entender las funciones de algunas rutinas.

- El consumo excesivo en ocasiones de los recursos de máquina.

Existen otras muchas deficiencias de los programas que se encuentran en operación actualmente, pero en general, la mayoría de ellas pueden ser solucionadas por medio de los modelos de sistemas y por medio también del uso intensivo de las técnicas de análisis, diseño y programación estructurada disponibles.

Como ya se mencionó previamente, el objeto de este trabajo es presentar una herramienta que

se ha construido para elaborar modelos. A partir del siguiente capítulo, se empieza a trabajar en la construcción de la herramienta.

C A P I T U L O I I

ANALISIS DEL SISTEMA Y OBTENCION DEL DIAGRAMA DE FLUJO DE DATOS.

II.1 DESCRIPCION DEL ANALISIS ESTRUCTURADO.

II.2 ESTABLECIMIENTO DEL DIAGRAMA DE FLUJO DE DATOS.

II.1 DESCRPICION DEL ANALISIS ESTRUCTURADO.

Con el fin de obtener de una manera más clara el concepto de 'Análisis Estructurado' como una herramienta formal para el desarrollo de sistemas, es necesario exponer previamente, algunos de los problemas más sobresalientes a los que los especialistas en el desarrollo de sistemas se enfrentan:

- Dado que los usuarios son personas que hacen sus tareas, en lugar de describirlas, el especialista no puede esperar una explicación lo suficientemente amplia de los requerimientos del sistema por parte del usuario y por lo tanto, se convierte en labor propia del especialista el ayudar al usuario a trabajar con sus necesidades.

- Las personas que conforman la comunidad usuaria no conocen lo suficiente de computadores ni de procesos de datos, como para poder saber lo que es factible y lo que no lo es.

- No existen herramientas que permitan manejar y controlar los detalles de la información que el especialista obtiene.

- El diseño general de un sistema es un documento que hace las veces de contrato entre el usuario y el grupo de desarrollo, pero generalmente es imposible que el usuario pueda entender dicho documento.

- Cuando se especifica un modelo físico antes que un modelo lógico, el diseño resulta de un nivel inferior.

Los problemas mencionados pueden ser resueltos mediante el uso adecuado de la técnica de análisis estructurado que tiene como objetivo fundamental especificar de la manera más precisa posible los requerimientos del usuario para su sistema.

Para lograr su objetivo, el análisis estructurado usa como su principal herramienta al diagrama de flujo de datos que usa a su vez elementos gráficos para mostrar las transformaciones de los datos a medida que

estos fluyen a través de los procesos del sistema. El diagrama de flujo de datos es un instrumento de modelación que permite mostrar a un sistema como una red de subsistemas conectados unos a otros mediante flujos de datos que muestran las relaciones entre subsistemas y las transformaciones que sufren los datos.

El producto del análisis estructurado es una especificación funcional que debe cumplir las siguientes caracteristicas:

- Ser bien entendida y esté de acuerdo con los usuarios.

- Mostrar los requerimientos lógicos sin dictar una implantación física.

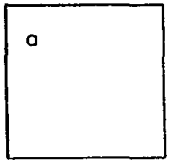
- Expresar preferencias y negociaciones.

Los elementos gráficos de un diagrama de flujo de datos son los siguientes:

- Entidad Externa. Representa cosas o personas que son fuente o destino de datos como por ejemplo: Clientes, empleados, proveedores,

departamentos de una empresa, incluso se consideran entidades externas, a un sistema que proporciona datos al sistema en consideración o al sistema que recibe datos del sistema en consideración.

Una entidad externa se simboliza con un rectángulo con sus lados superior e izquierdo más anchos que los dos lados restantes para diferenciarlos del resto del diagrama. La entidad externa puede ser identificada con una letra minúscula en la esquina superior izquierda. En la figura II.1, se muestra una entidad externa.

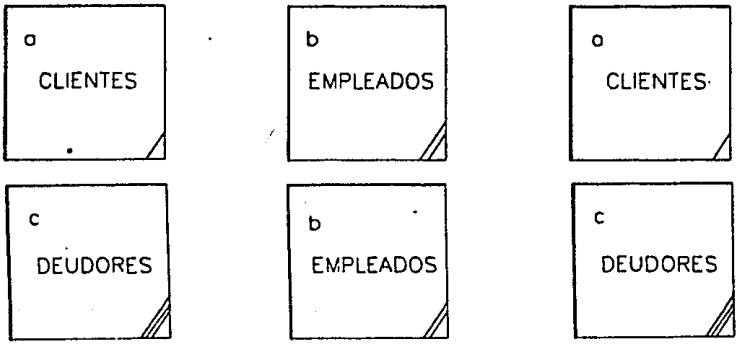


Entidad Externa

fig. II.1

Con el fin de evitar líneas de flujo de datos cruzadas, una entidad puede ser dibujada más de una vés en el mismo diagrama, los dos o mas rectángulos de la misma entidad son identificados trazando en cada uno de ellos una línea opuesta al ángulo de la esquina inferior derecha. Si se requiere duplicar otra entidad, los dos o mas rectángulos de esta segunda entidad, serán identificados trazando dos líneas de las características mencionadas en el párrafo anterior. Para una tercera entidad duplicada, se trazan tres líneas y asi sucesivamente.

El nombre de la entidad se escribe en la parte central del rectángulo (fig. II.2).



Notacion para entidades duplicadas

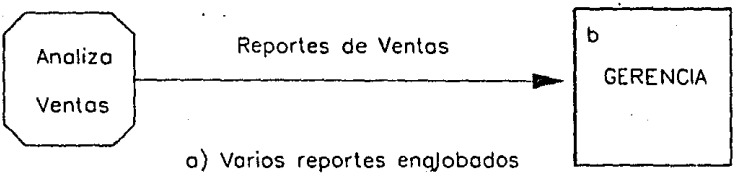
fig. II.2

- Flujo de Datos. Representan la dirección que toman los datos.

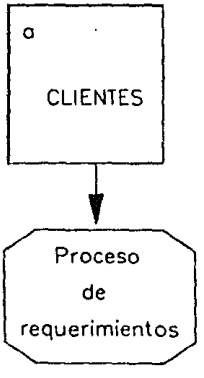
El flujo de datos se simboliza con una flecha, preferentemente horizontal o vertical con una cabeza de flecha señalando la dirección del flujo. Con la intención de ser más claro, se pueden usar flechas con dos cabezas en lugar de dos flechas cuando los datos viajan en ambos sentidos. La parte inicial de la flecha está conectada a donde los datos se originan mientras que su cabeza apunta a donde los datos son enviados. Cada flujo de datos debe tener anotado a su lado una descripción de su contenido.

La descripción debe ser escogida de tal manera que tenga el mayor significado posible para el usuario que revisarán el diagrama de flujo de datos. Esta descripción debe ser escrita en el diagrama con letras mayúsculas y minúsculas en el inicio del análisis, pero en una fase más avanzada del mismo, cuando el diccionario de datos ha sido definido, la descripción puede ser cambiada a letras mayúsculas solamente, con el propósito de mostrar que ya ha sido incluida en el

diccionario de datos. En el caso de que existan varios grupos de datos en un mismo flujo, la descripción del flujo será una que englobe a todas esos grupos y posteriormente, en el diccionario de datos, se pueden listar todas sus componentes (grupos), cada uno de estos componentes es a su ves descrito en el diccionario de datos. Es válido también omitir la descripción si ésta resulta evidente por si misma a la persona que revise el diagrama de flujo de datos. (fig II.3).



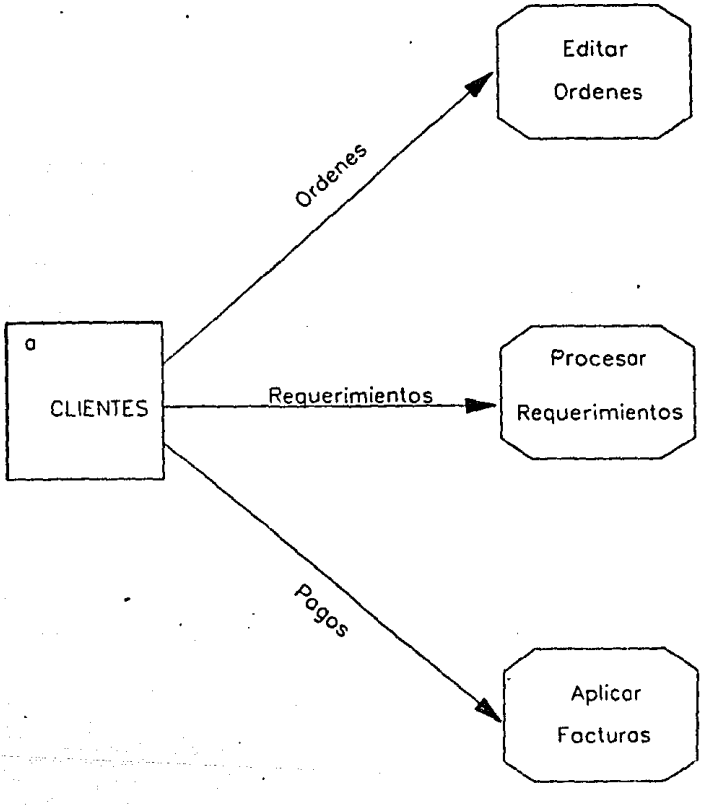
a) Varios reportes englobados



b) Descripcion omitida

fig. II.3

Cuando cada transacción de un conjunto de transacciones se procesará de diferente manera, se puede dibujar una flecha diferente para cada transacción (fig II.4).

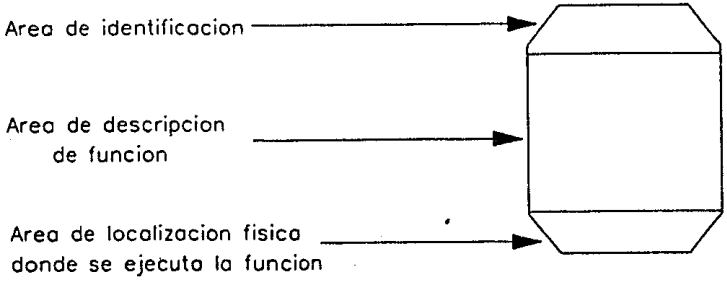


Diferentes procesos para datos

fig. II.4

- Procesos. Representa la función que actúa sobre un dato para modificarlo o generarlo por ejemplo: Extraer, verificar, calcular, etc.

Un proceso se representa con un rectángulo con las esquinas redondeadas y opcionalmente dividido en tres areas (fig II.5).



Areas de un proceso

fig. II.5

En el área de identificación se escribe un número con el único propósito de identificar el proceso. Una vés asignado el número de identificación, éste no debe ser cambiado.

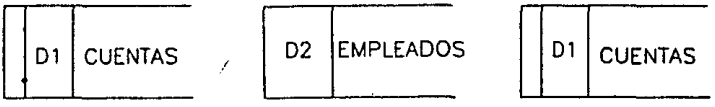
En el área de descripción de la función se escribe de manera no ambigua, la función que realizará el proceso. La descripción de la función del proceso debe ser una expresión imperativa que idealmente consiste de un verbo activo como: extaer, calcular, verificar seguida por una cláusula objeto por ejemplo: Extraer-ventas mensuales, Verificar-crédito, etc.

Finalmente en el área de localización física, se escribe el nombre o clave del departamento o programa que lleva a cabo la función. (fig. II.6).

Ejemplos de uso de areas de un proceso

fig. II.6

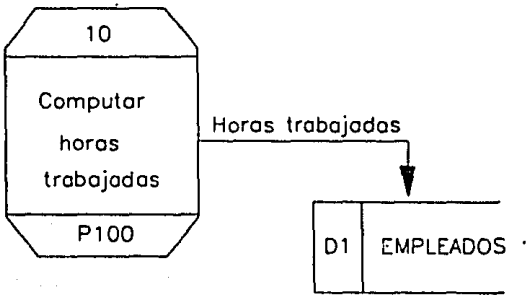
- **Archivos.** Representa los lugares donde se almacenan los datos que usan los procesos. Se simbolizan con un par de líneas paralelas de la longitud que ocupe el nombre de los datos que se almacenan en tal archivo, el par de líneas se cierra en uno de sus extremos (generalmente el izquierdo). El archivo puede ser identificado con una "D" y un número arbitrario en el extremo izquierdo. El nombre del archivo se debe escoger de tal manera que resulte descriptivo para el usuario. De la misma manera que en las entidades externas, para evitar el cruce de flujos de datos, el mismo archivo puede ser dibujado más de una vez en el mismo diagrama identificando los archivos duplicados con líneas adicionales en su extremo izquierdo (fig. II.7).



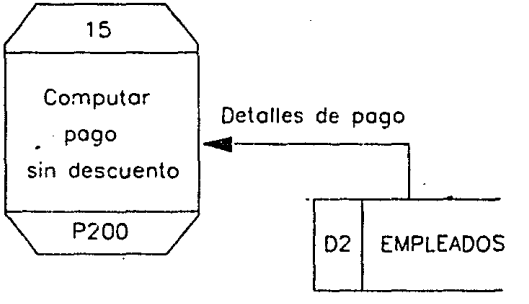
Notacion para archivos duplicados

fig. II.7

Cuando un proceso almacena datos el flujo de datos se dibuja entrando al archivo, en el caso de que el proceso accese datos de un archivo, el flujo de datos se dibuja saliendo del archivo. (fig II.8).



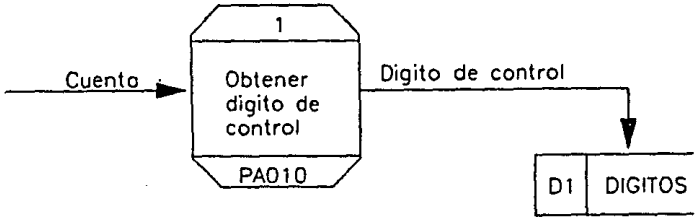
a) Almacen de datos



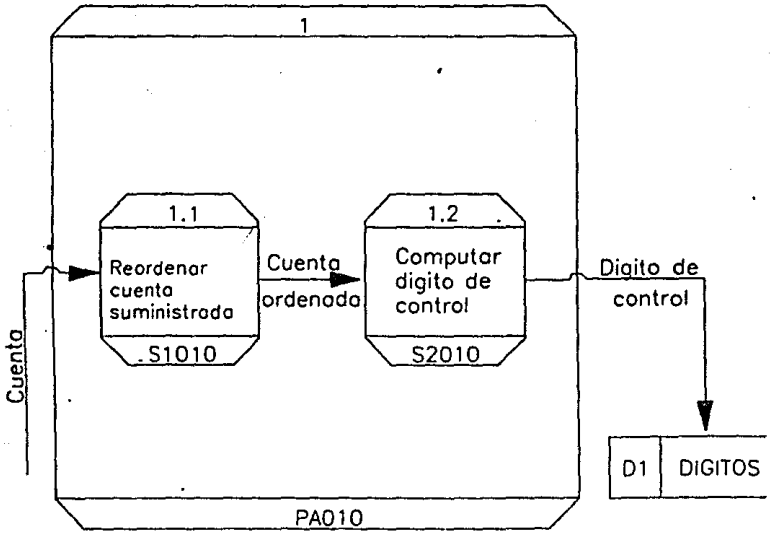
b) Acceso de datos

fig. II.8

Frecuentemente es necesario que un proceso sea desglosado en otros con mayor detalle. Cuando esto ocurre, la identificación de los procesos de menor nivel se efectúa escribiendo decimales del proceso de mayor nivel (fig II.9).



a) Proceso de mayor nivel a detallar



b) Procesos de menor nivel obtenidos

fig. II.9

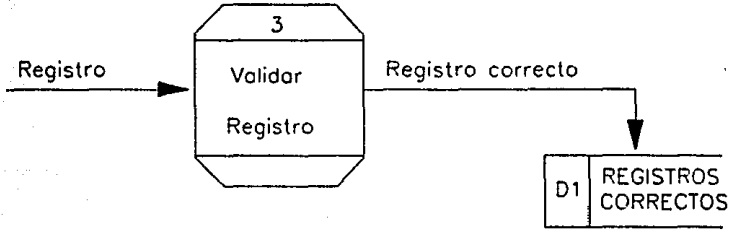
Cuando se representen flujos de datos que se presentan por primera vez en un nivel bajo, como por ejemplo salidas de error, deben rebasar el límite del proceso y para distinguirlos de los demás se marcan con una X en el punto de salida. Los archivos se dibujan dentro del límite del proceso de mayor nivel si son creados y accesados por éste proceso. En caso de que el archivo sea externo al proceso, se dibuja con una mitad dentro del proceso y la otra mitad fuera o completamente fuera, según convenga para la claridad visual del diagrama.

La nomenclatura D4/1 para archivos indica: el primer archivo interno del proceso 4.

Las entidades externas no se muestran dentro de los límites del proceso aún cuando no sean envueltas en algún otro proceso.

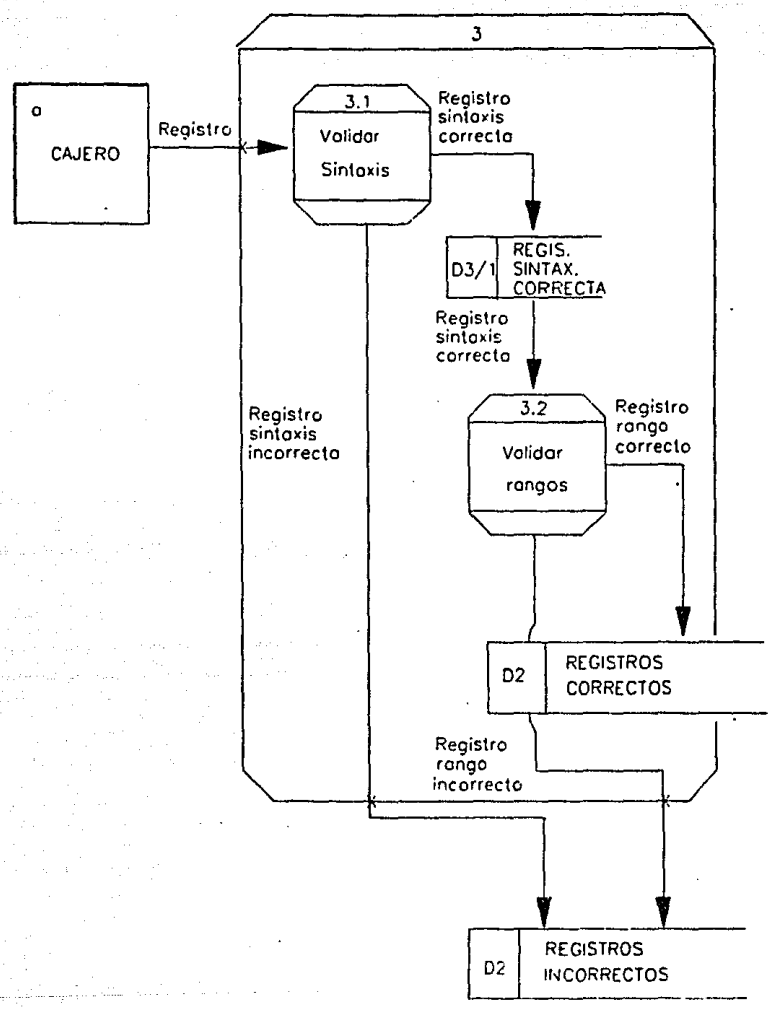
Cuando resulta inevitable que se crucen las líneas de flujo de datos o cuando es necesario cruzar un archivo, se usa la siguiente convención:

En la figura II.10 se muestran las caracteristicas mencionadas en las lineas anteriores.



Proceso de mayor nivel

fig. II.10 a



Notaciones para procesos de menor nivel

fig. II.10 b

Los procesos de error deben manejarse dentro de la los procesos que resulten de 'desmenuzar' el proceso que los genera.

A continuación, se listan los pasos a seguir a manera de guia para dibujar diagramas de flujo de datos:

- **Identificar las entidades externas involucradas.** Esto permitirá definir los límites preliminares del sistema.

- **Identificar las entradas y salidas que se manejarán entre las entidades identificadas en el paso anterior.** Una vés identificadas, tratar de agruparlas en grupos lógicos de entradas y salidas y marcar las entradas y salidas que son generadas por condiciones de error.

- **Identificar los requerimientos de información que pudieran surgir.** Especificar un flujo de datos que defina los datos suministrados al sistema y otro que defina los datos que son requeridos al sistema.

- Dibujar el D.F.D.

Dibujar la entidad externa que se considere sea la principal fuente de datos, dibujar los flujos de datos, los procesos lógicos y los archivos que se consideren requeridos. Es importante no hacer consideraciones de tiempo cuando se está dibujando un D.F.D., y también lo es el dibujar un sistema sin inicio ni fin (es decir que se considere que el sistema siempre ha estado en operación y que nunca terminará para facilitar su dibujo).

Es importante en este momento no considerar condiciones de errores ni decisiones. Una vez terminado éste primer dibujo, verificar nuevamente la lista de entradas y salidas para asegurar que se han considerado todas (excepto errores).

- Duplicar símbolos si es necesario. En caso de ser necesario, duplicar entidades y archivos además de permitir cruces de líneas.

- Desglosar los procesos. Cuando así se

requiera, desglosar los procesos de mayor nivel usando las convenciones especificadas. Repetir este paso hasta que no sea necesario alcanzar mas detalle.

II.2 ESTABLECIMIENTO DEL DIAGRAMA DE FLUJO DE DATOS.

Para iniciar la labor de desarrollo de la herramienta para construcción de modelos, es conveniente situar el lugar preciso que ésta ocupa en el ciclo de desarrollo de sistemas.

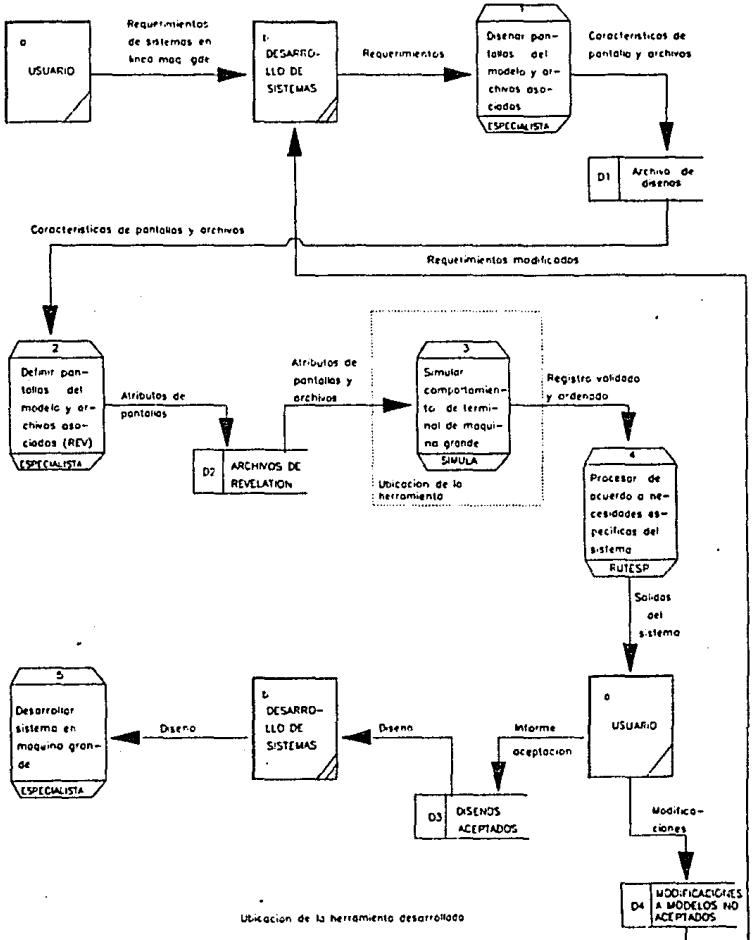
La labor de desarrollo de sistemas, se inicia cuando el usuario del sistema origina un requerimiento de información, la entidad encargada del desarrollo de sistemas inicia a su vez una labor de análisis del problema presentado por el usuario y posteriormente llega a un diseño de pantallas y archivos que manejará el sistema en cuestión. Mediante el uso del paquete Revelation, el cual es un manejador de bases de datos relacional, el especialista encargado del desarrollo del sistema, será capáz de definir pantallas y archivos asociados a esas pantallas y almacenar estas definiciones en los archivos que para ese fin maneja el paquete.

La herramienta para construcción de modelos tomará las definiciones de pantallas y archivos asociados, de los archivos del paquete

correspondientes y simulará en la computadora personal un comportamiento muy similar al que presentan las terminales de máquinas grandes, en lo que se refiere a llenado de campos y validación de los mismos. La salida de la herramienta es un registro validado y ordenado (en cuanto al lugar que ocupa el campo en el archivo que puede ser diferente al que ocupa en la pantalla) que servirá de entrada a una rutina o módulo que el mismo especialista escribirá para efectuar sobre él las operaciones específicas que requiera el sistema que se esté modelando.

Tanto las pantallas en operación como las salidas que produzca el modelo, serán presentadas a la entidad usuaria para que las analize y decida si son las que necesita. En el caso que el usuario decida que las pantallas y salidas son correctas, se continuará con la labor de desarrollo para la máquina grande donde residirá el sistema; en caso contrario, se redefinirán las características de pantallas y archivos para iniciar nuevamente el ciclo de definición con el paquete. Esta última actividad se repetirá cuantas veces sea necesario hasta

que se logre la completa satisfacción del usuario. En la figura II.11, se muestra gráficamente lo expresado en los párrafos precedentes.



Ubicacion de la herramienta desarrollada

fig II.11

Una vez definida la ubicación de la herramienta en el proceso de desarrollo de sistemas, es posible trabajar sobre la construcción de la misma. Es necesario establecer que los límites del problema se considerarán como se ha señalado en la figura II.11 ya que los procesos de diseño y definición de características del modelo son realizados previamente por el especialista y no están involucrados en el proceso de simulación.

A continuación, se listan los pasos señalados en el punto II.1 para dibujar diagramas de flujo de datos y los resultados obtenidos en cada uno de ellos en el proceso de desarrollo de la herramienta.

- Identificar las entidades externas involucradas.

La entidad involucrada es el área de desarrollo de sistemas únicamente, pero no se indicará en los diagramas subsecuentes por la razón descrita anteriormente referente a los límites del sistema.

- Identificar las entradas y salidas que se manejarán entre las entidades.

Entradas: Atributos de pantallas a emular y archivos asociados.

Salidas: Registros validados y ordenados listos a ser procesados mediante las operaciones específicas del sistema que se esté modelando.

- Identificar los requerimientos de información que pudieran surgir.

Datos suministrados al sistema: Atributos de pantallas y archivos asociados.

Datos requeridos al sistema: Registros validados y ordenados y comportamiento de la computadora personal como terminal de máquina grande.

- Dibujar el D.F.D.

El primer diagrama obtenido es el que se presenta en la figura II.12.

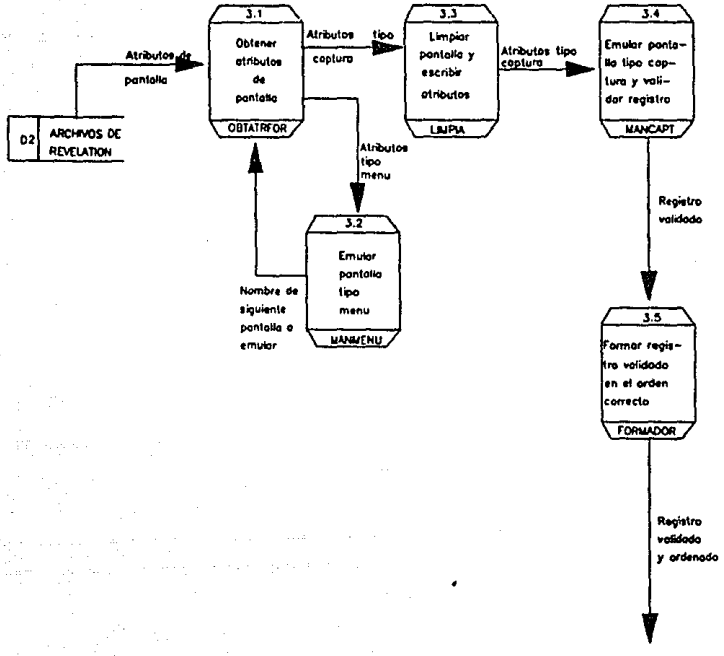


Diagrama de flujo de datos inicial para la herramienta de construccion de modelos

fig. 8.12

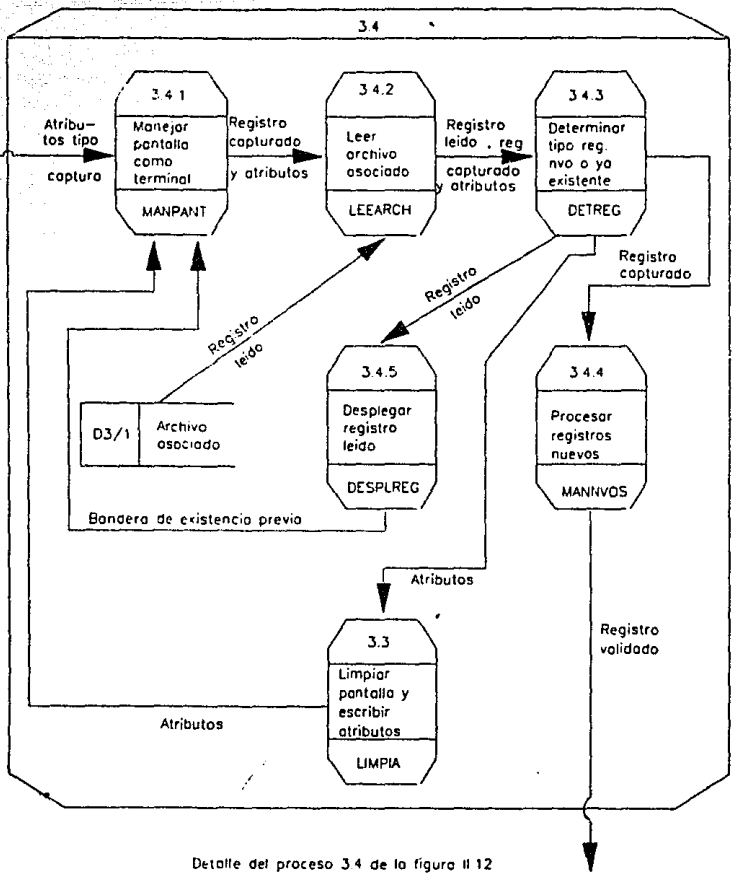
En éste momento, no se consideran condiciones de error.

- Duplicar símbolos si es necesario. Para el problema que se está resolviendo, no es necesario duplicar símbolos.

- Desglosar los procesos. El proceso # 3.4 es el proceso que requiere mayor detalle para que su función resulte clara. En la siguiente figura (fig. II.13) se muestra el resultado de éste paso, dentro de la misma, el desglose del proceso (3.4.4) que a su vez requiere de mayor detalle para comprender su función.

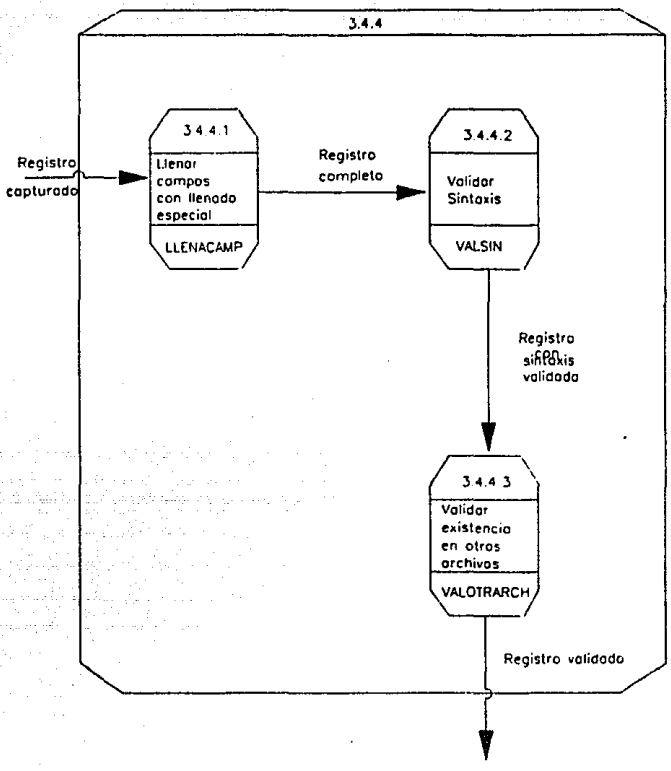
El diagrama de flujo de datos general del sistema es, el diagrama de la figura II.12 con el proceso 3.4 sustituido por la figura II.13. El diagrama general obtenido se muestra finalmente en la figura II.14.

El diagrama de flujo de datos obtenido en éste capítulo, será transformado para obtener su carta de estructura correspondiente en el siguiente capítulo.



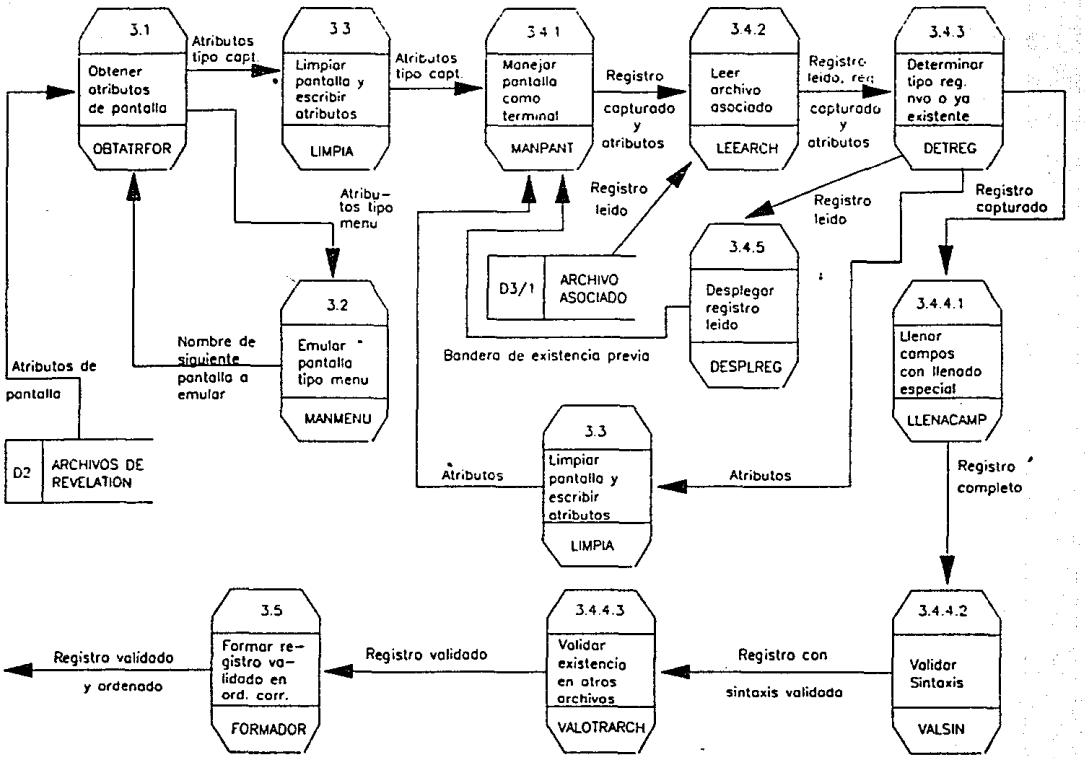
Detalle del proceso 3.4 de la figura II.12

fig II.13 a



Detalle del proceso 3.4.4 de la figura II.13 a

fig II.13 b



D.F.D General para la herramienta desarrollada

fig II.14

C A P I T U L O I I I

DISEÑO DEL SISTEMA Y OBTENCION DE LA CARTA DE ESTRUCTURA.

III.1 DESCRIPCION DEL DISEÑO ESTRUCTURADO.

III.2 ESTABLECIMIENTO DE LA CARTA DE ESTRUCTURA.

III.1 DESCRIPCION DEL DISEÑO ESTRUCTURADO.

El producto final de la labor de análisis es, como se puede observar fácilmente del capítulo anterior, un conjunto de requerimientos funcionales (también conocidos como "especificaciones funcionales" o "especificaciones del sistema") que describen en términos precisos: las entradas que serán suministradas por el usuario, las salidas deseadas por el usuario y los algoritmos involucrados en los cálculos deseados por el usuario.

Tomando como punto de partida el producto final del análisis, el diseño estructurado tendrá como producto final, un documento que consolida, formaliza y hace visibles las actividades del diseño y las decisiones que se suscitan inevitablemente en el curso de un proyecto de desarrollo de un sistema. También se puede definir al diseño estructurado como el proceso de decidir que componentes interconectados en que manera resolverán un problema bien especifico o como el arte de

diseñar los componentes de un sistema y la interrelación entre esos componentes de la mejor manera posible.

El diseño estructurado permite particionar la solución del problema en módulos teóricamente autocontenidos e independientes unos de otros que realizan funciones específicas bien definidas.

A continuación se describen algunas de las definiciones y términos propios del diseño estructurado más usados.

- **Acoplamiento.** Es la medida del grado de interdependencia entre dos módulos. Existen tres tipos de acoplamiento que se mencionan en orden de mayor a menor calidad:

Acoplamiento por datos. Dos módulos están acoplados por datos si se comunican por datos que no sean banderas, ni arreglos, ni registros.

Acoplamiento por estampilla. Dos módulos están acoplados por estampilla si se comunican mediante registros o arreglos.

Acoplamiento por control. Dos módulos están acoplados por control si se comunican al menos por bandera.

- Cohesión. Es la medida del grado de asociación de los elementos dentro del módulo, considerando un elemento como una instrucción, un grupo de instrucciones o una llamada a otro módulo. Existen tres tipos de cohesión que como en el caso anterior, se describen en orden de mayor a menor calidad:

Cohesión funcional. Un módulo con cohesión funcional es aquel en que todos los elementos contribuyen a una y sólo una tarea, por ejemplo:

+ Imprimir una matríz.
+ Leer una matríz.
+ Sumar dos matrices.

Cohesión secuencial. Un módulo con cohesión secuencial es aquel en que todos los datos de salida de un elemento sirven como datos de entrada a otro elemento, por ejemplo:

+ La salida de una llamada a un módulo de validación es una bandera que sirve de entrada a una instrucción de decisión en base a la cual

se determinará si se pueden efectuar un proceso o no.

Cohesión comunicacional. Un módulo con cohesión comunicacional es aquel cuyos elementos contribuyen a tareas diferentes; pero cada tarea tiene los mismos parámetros de entrada y salida, por ejemplo:

+ Imprimir archivo de transacciones.
+ Grabar archivo de transacciones en cinta.

La mejor manera de obtener un buen diseño, es evaluar varios diseños alternativos y escoger el mejor en cuanto a eficiencia, rentabilidad y facilidad para el mantenimiento pero satisfaciendo las restricciones de diseño tales como uso de memoria y tiempo de respuesta. Además sera necesario tener siempre en mente el siguiente concepto:

Para minimizar el costo de implantación, mantenimiento y modificación de un sistema, se requiere que sus componentes cumplan las siguientes caracteristicas:

Corresponder a exactamente una pequeña y bien definida parte del problema y que cada relación entre piezas del sistema corresponda solamente a una relación entre partes del problema.

Las características mencionadas se logran considerando siempre los principios básicos de diseño:

- Las partes del problema que están altamente interrelacionadas, deben estar en la misma pieza del sistema.

- Las partes del sistema que no tienen relación entre si, deben estar en partes no relacionadas del sistema.

- Siempre debe existir una partición adecuada de la aplicación.

- Los juicios que se puedan tener acerca de un sistema computacional, pueden ser expresados haciendo analogías con las organizaciones humanas.

La herramienta principal del diseño estructurado es la Carta de Estructura que como en el caso del análisis estructurado, es una herramienta gráfica que permite mostrar la partición de un sistema en módulos y la relación jerárquica entre éstos además de los flujos de datos y control entre los módulos.

Con el objeto de definir el concepto de módulo, a continuación se define el concepto de órden léxico de un programa haciendo la aclaración de que éste concepto tiene otros significados en linguistica.

El órden léxico de un programa se refiere a como está escrito o a como aparece en un listado, esto es, como se muestra a continuación, la instrucción etiquetada B está incluida léxicamente en el alcance de la rutina o agregado A, y C sigue léxicamente a la instrucción etiquetada como A2.

```
        ----------
A1:     BEGIN A
        ----------
        ----------
B:      ----------
        ----------
        ----------
A2:     END A
C:      ----------
        ----------
        ----------
```

Este órden léxico, es independiente del órden en que las instrucciones aparecerán en memoria y también es independiente del orden en que se ejecutarán.

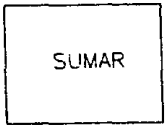
Una vez definido el órden léxico, es posible definir: módulo de un programa o sistema.

Un módulo es una secuencia léxicamente contigua de instrucciones de programas, limitada por elementos de límite y que tiene un identificador de rutina o agregado. Otra definición igualmente válida para un módulo es:

Módulo es un grupo contiguo limitado de instrucciones de un programa que tienen un solo nombre (de rutina o agregado) por medio del cuál pueden ser referenciadas.

Una ves definidos los conceptos de órden léxico y módulo, será necesario describir los elementos de la carta de estructura y las convenciones que se han aceptado para su representación gráfica.

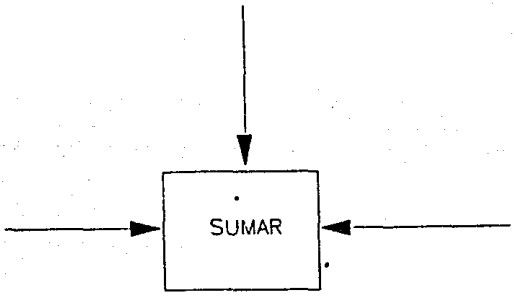
- Módulo. Representa módulos de un programa ó sistema y se simboliza con un rectángulo con un nombre inscrito que indica la función que dicho módulo realiza (fig. III.1)



Modulo

fig. III.1

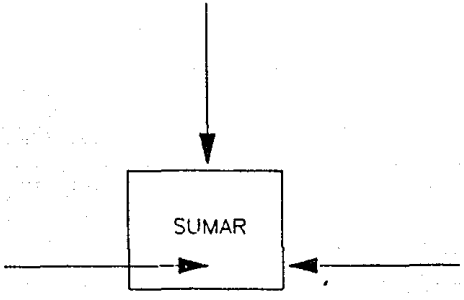
- **Referencia.** Representa la liga entre módulos, en otras palabras, las referencias o llamadas que se hacen a un módulo o agregado. Se representan mediante segmentos dirigidos con cabezas de flechas que terminan en el límite del módulo al cual hacen referencia (fig. III.2).



Notacion para referencias a un modulo

fig. III.2

Cuando la referencia se hace a identificadores definidos dentro de un módulo dado, el segmento dirigido, termina dentro de los límites del módulo al cual hace referencia (fig. III.3).

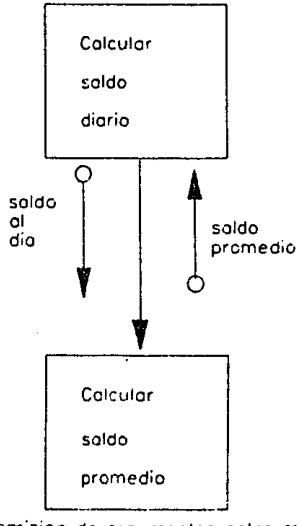


Notacion para referencia a identificadores internos del modulo

fig. III.3

Las referencias a módulos son conocidas como **conexiones normales,** mientras que las referencias a identificadores internos son llamadas **conexiones patológicas.**

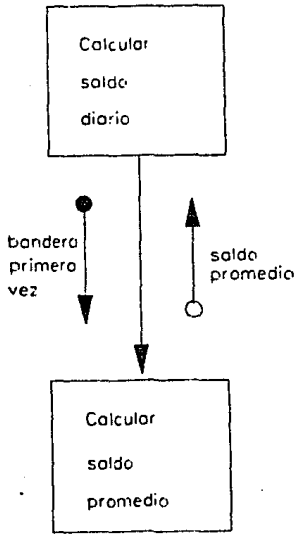
Cuando la existencia y naturaleza de los argumentos de un módulo (paso de parámetros) es de interés, se indica mediante una pequeña flecha con un pequeño círculo en su extremo inicial y en su extremo final la cabeza de flecha para indicar el sentido del paso de los argumentos. Es necesario listar a un costado de



Transmision de argumentos entre modulos

fig. III.4

Como se ha establecido, un pequeño circulo en el extremo inicial de una flecha indica paso de parámetros, pero estos parámetros son exclusivamente datos, cuando se requiere involucrar un elemento de control, la notación cambia a un punto en lugar del pequeño circulo (fig. III.5).

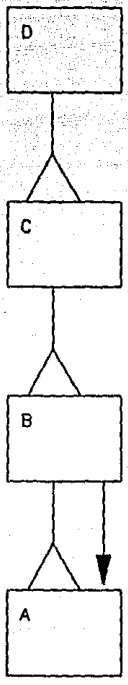


Notacion para control (izquierda) y datos (derecha)

fig. III.5

En la figura III.6, se muestra la convención adoptada para representar la inclusión léxica que se presenta cuando un módulo está incluido dentro de otro, por ejemplo, la figura representa gráficamente que A está incluido en B, B a su ves está incluido en C y finalmente C en D, en un listado del programa aparecería:

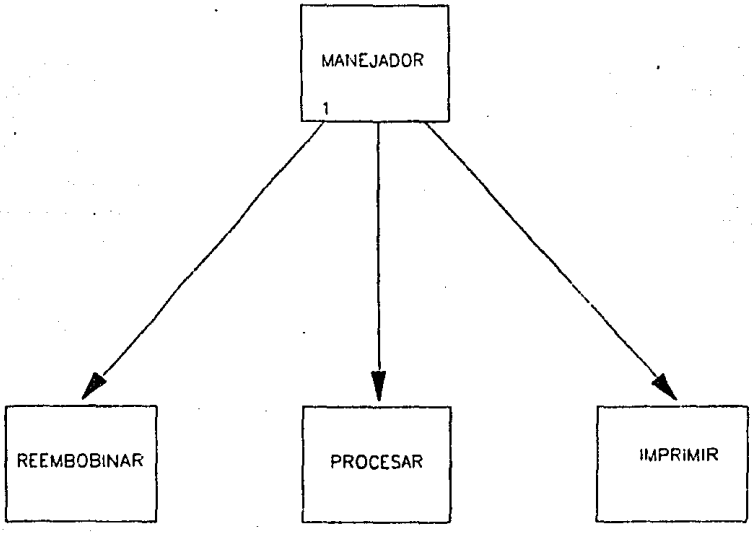
```
D: ----------
   ----------
   ----------
C: ----------
   ----------
B: ----------
   ----------
A: ----------
   ----------
END A:
   ----------
END B:
   ----------
END C:
   ----------
   ----------
END D:
```



Notacion para inclusion lexica

fig. III.6

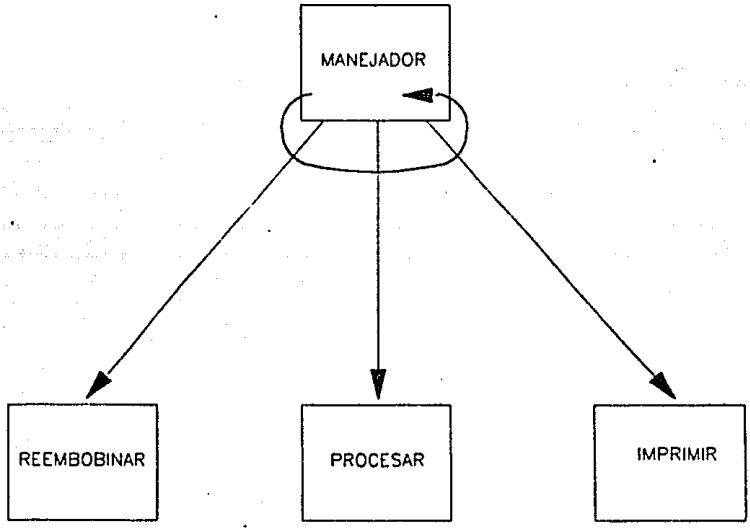
En ocasiones, es necesario indicar que un módulo será ejecutado solo un número determinado de veces, para tal situación, la convención establecida se muestra en la figura III.7 en donde el módulo Reembobinar, se ejecuta una sola ves (que se indica con el Pequeño '1' junto al extremo inicial de la flecha representando la referencia a ese módulo) y no volverá a ejecutarse ni adn en el caso de reiniciar el módulo principal.



Notacion para una sola ejecucion de un modulo

fig. III.7

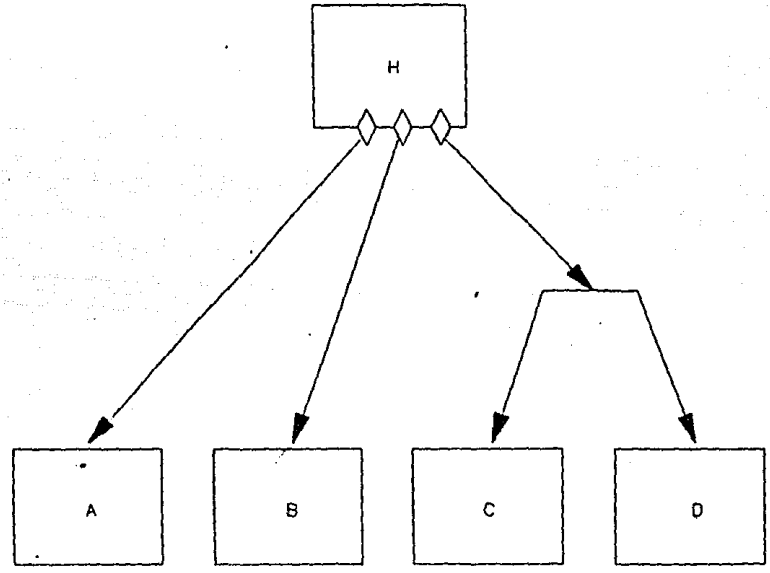
Los módulos subordinados a uno de mayor jerarquía, que son ejecutados de manera repetitiva, se representan gráficamente como se muestra en la figura III.8. Todos los módulos activados dentro de una misma iteración se muestran con sus referencias emergiendo de la misma flecha que indica la ejecución repetitiva.



Notacion para ejecucion repetitiva de modulos

fig. III.8

Para mostrar situaciones en las que intervienen procesos de decisión, se usa un diamante del cual emergen las referencias a subordinados. Es válido incluso tener más de dos referencias saliendo de un diamante para representar el código de una instrucción "CASE" (fig. III.9).

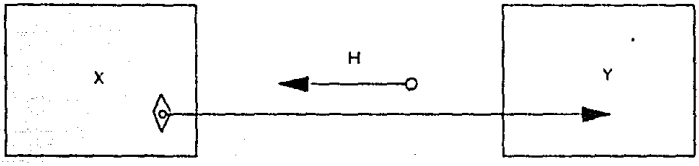


Notacion para salidas de decisiones alternativas

fig. III.9

En la figura III.9, se muestra también el caso especial cuando se hacen varias llamadas a módulos como resultado de una misma condición.

Es permitido también, representar llamadas patológicas condicionadas de datos como se muestra en la figura III.10.



Conexion patologica condicionada

fig. III.10

Una vez que se han presentado todas las notaciones para casos especiales, se requiere dar una lista de los pasos que se deben seguir para lograr un diseño de muy buena calidad. Antes de iniciar esa lista, se considera importante mencionar algunas conclusiones a las que se ha llegado después de varios años de observaciones y que se considera son relevantes para obtener un buen diseño.

Siempre es necesario establecer una relación jerárquica entre los módulos de un programa o sistema y lograr una organización de tipo humano, esto es: En una organización humana, existe por ejemplo un gerente que es el que se encarga de coordinar las actividades de uno o varios subgerentes, cada uno de ellos a su ves coordina las actividades de uno o varios supervisores y así sucesivamente. En el caso de un sistema es necesario tener un módulo principal que coordine la ejecución de sus subordinados y que éstos a su ves coordinen las de otros módulos de menor nivel, etc.

Los módulos de mayor nivel, deben efectuar la

mayoría de las decisiones mientras que los módulos de menor nivel deben ejecutar la mayoría del trabajo detallado.

Como se mencionó al inicio de éste capitulo, el diseño estructurado se inicia a partir de los diagramas de flujos de datos dejados por la fase de análisis estructurado, los pasos que a continuación se describen, tienen como objetivo efectuar el paso o **transformación** del D.F.D. a la carta de estructura. Debido a que se considera una transformación el paso de D.F.D. a la C.E., la estrategia que se usa para éste fin se ha llamado "Diseño Centrado en Transformación".

El diseño centrado en transformación, se puede definir como "una estrategia para derivar diseños estructurales iniciales que generalmente son buenos en lo que se refiere a modularidad y que generalmente requieren una pequeña reestructuración para lograr un diseño final".

Los pasos que se deben seguir para lograr una carta de estructura mediante un análisis de transformación son los siguientes:

- Lograr un D.F.D. con las características mencionadas en el capitulo anterior. A partir de este momento, el nombre de 'proceso' y el de 'transformación' serán equivalentes.

- **Identificar los datos aferentes y eferentes.** Los datos aferentes son aquellos elementos de más alto nivel de abstracción del término 'entradas al sistema' y representan las entradas más procesadas de nivel macro. En otras palabras las entradas aferentes son los datos que tienen la menor semejanza posible de los datos obtenidos de los dispositivos físicos "en bruto", es decir, son datos que han sido despojados del bloqueaje físico, caracteres de control, han sido pasados por convertidores de formatos y todas las tareas de edición, verificación y validación han sido terminadas satisfactoriamente. Lo que queda son datos limpios listos para procesarse.

Otra manera de identificar datos aferentes es comenzar con las entradas físicas al sistema y recorrer el D.F.D. hasta que se encuentren datos que no puedan ser considerados de entrada.

Los datos eferentes son aquellos que fueron producidos por el proceso principal del sistema y que requieren de un mínimo de procesamiento para lograr las salidas lógicas requeridas por el usuario final del sistema.

En este paso, resulta indistinto comenzar a identificar los datos aferentes o los datos eferentes.

Generalmente, este paso deja algunas transformaciones entre los datos aferentes y los datos eferentes. Estas transformaciones se designan **transformaciones centrales.**

- **Factorizar en un primer nivel.** Para cada dato aferente que alimente una transformación central, se especifica un módulo aferente como subordinado inmediato al módulo principal. Este último módulo servirá para enviar el elemento

aferente en cuestión al módulo de mayor jerarquía que lo originó, es decir, al módulo principal. La característica principal del módulo aferente, consiste en que éste tipo de módulos obtiene las entradas de módulos de menor nivel y envía estos datos hacia módulos de mayor nivel.

De manera similar, para cada dato eferente que se origine de cualquier transformación central, se define un módulo eferente subordinado que aceptará el dato eferente y lo transformará en la salida física requerida.

Finalmente, para cada transformación central, se debe especificar un módulo de transformación subordinado que aceptará del módulo principal los datos de entrada apropiados y los transformará en las salidas apropiadas. El módulo principal controla y supervisa todo el proceso. Coordina los módulos aferentes, los módulos de transformación y los módulos eferentes manejando los datos de más alto nivel del sistema.

- **Factorizar los módulos aferentes, eferentes y de transformación.** Antes de describir el proceso de factorización en este paso, es necesario aclarar que no se requiere factorizar completamente un módulo hasta llegar a su nivel mínimo de detalle antes de empezar a trabajar con otro, pero si es necesario identificar todos sus subordinados inmediatos. De la misma manera, no existe una razón que obligue a empezar a trabajar con los módulos aferentes, siendo posible comenzar con cualquier tipo de módulo.

El proceso de transformación es el siguiente:

Identificar la función del módulo aferente en cuestión y encontrar una transformación o cálculos requeridos para lograr esa función.

Para cada entrada a ésta última transformación, se especifica un nuevo módulo aferente inmediatamente subordinado al módulo aferente inicial. Cada uno de estos módulos aferentes de menor nivel, son factorizados recursivamente de la misma manera hasta que se alcanza la última entrada física o bien, se termine el proceso.

Para factorizar módulo eferentes, el proceso es simétrico al descrito previamente, esto es: Para un módulo eferente, se identifica una transformación cuya salida sean los datos más cercanos a su forma física final y se coloca inmediatamente subordinada al módulo eferente de mayor nivel en el sistema. La salida de ésta transformación eferente, sirve de entrada a un nuevo módulo eferente que también estará inmediatamente subordinado al módulo aferente de mayor nivel y como en el caso anterior, recursivamente se agotarán las transformaciones encontradas.

Para las transformaciones centrales no existe una factorización óptima, es suficiente con lograr que los subordinados del módulo principal representen los niveles más altos del proceso y los detalles menos importantes sean concedidos a los subordinados de menor nivel.

- Existen algunos problemas en el mundo real en los que la forma ortodoxa de transformación descrita en las líneas anteriores no se puede aplicar estrictamente. En los casos en que esto suceda, estos cambios deben reflejarse en la

carta de estructura resultante.

Como medio de verificación para saber si un paso ha sido terminado, es necesario cuestionar acerca de si los subordinados son suficientes para implementar la transformación deseada.

III.2 ESTABLECIMIENTO DE LA CARTA DE ESTRUCTURA.

Tomando como punto de partida el diagrama de flujo de datos que se representa en la figura II.12, es posible iniciar la aplicación de los pasos listados en el punto III.1 para lograr la transformación del D.F.D en su carta de estructura correspondiente. Como en el caso de la obtención de diagrama de flujo de datos, la convención utilizada es la siguiente:

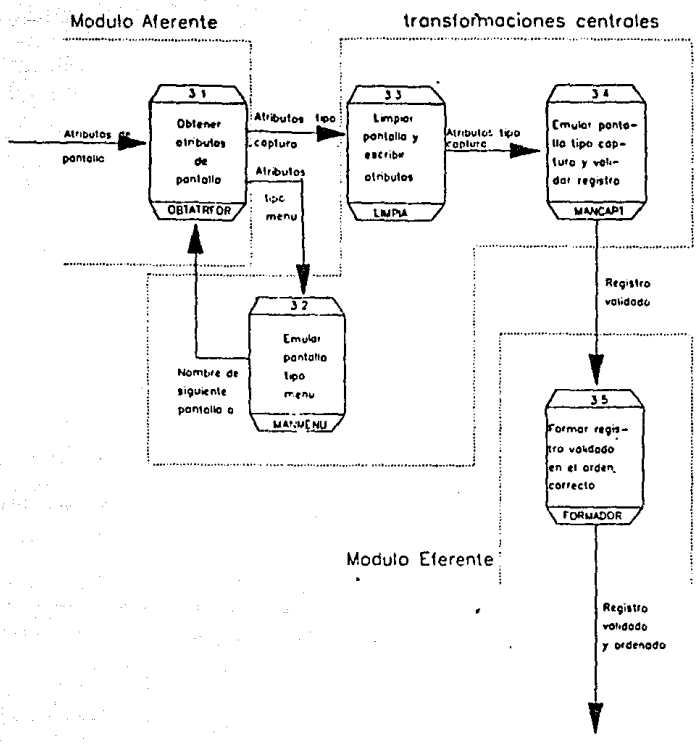
Mencionar el paso aplicado y a continuación el resultado obtenido de la aplicación de ese paso.

- Dibujar un D.F.D. con las características mencionadas en el capítulo anterior. Como ya se ha señalado, el digrama obtenido, es el presentado en la figura II.14

- Identificar los datos aferentes y eferentes. Para el caso en estudio, los datos aferentes consisten únicamente en los atributos de pantallas y archivos asociados que son leidos desde los archivos del paquete. Los datos

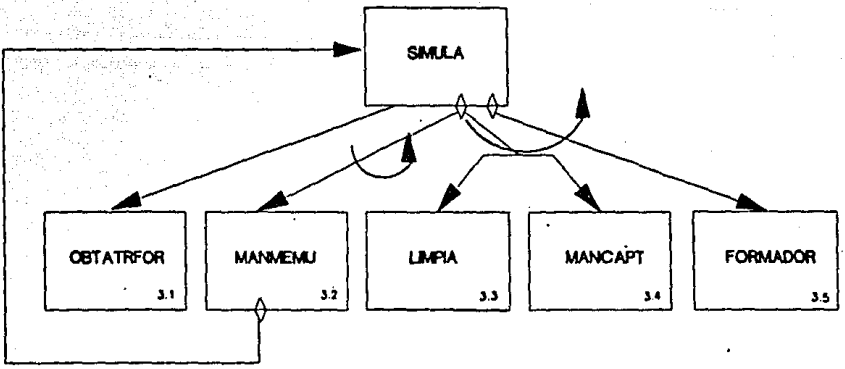
eferentes dadas las características del sistema en desarrollo, consisten en un registro validado y ordenado de acuerdo a la posición de sus campos en el archivo asociado. En base a lo anterior, las transformaciones centrales estarán compuestas por el proceso # 2 (Detectar tipo de pantalla a emular), el proceso #3 (Emular pantalla tipo captura y validar registro) y por el proceso #5 (Emular pantalla tipo menú).

- Factorizar en un primer nivel. En virtud de que el dato aferente es único, solo se especificará un módulo aferente cuya función resulta obvia (leer los atributos de pantalla de los archivos correspondientes del paquete). De manera similar, y dado que el dato eferente resulta ser único, solo se especificará un módulo eferente cuya función también resulta obvia (ordenar el registro según la posición que ocupan sus campos en el archivo). Tanto el módulo aferente como el eferente junto con las transformaciones centrales producen una primera carta de estructura, que es la que se muestra en la figura III.11



Identificacion de modulos aferente y eferente

fig.. III.11 a



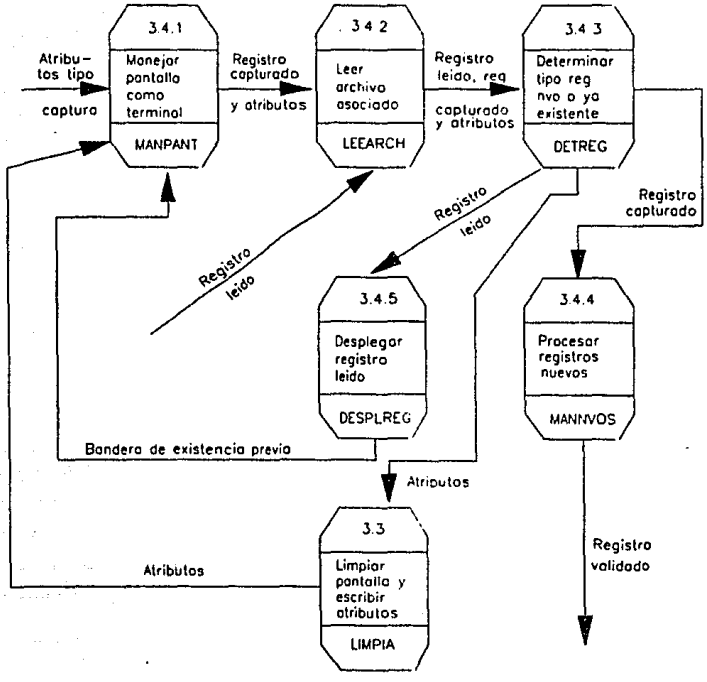
Factorizacion en primer nivel

fig. II.11 b

- Factorizar los módulos aferentes, eferentes y de transformación.

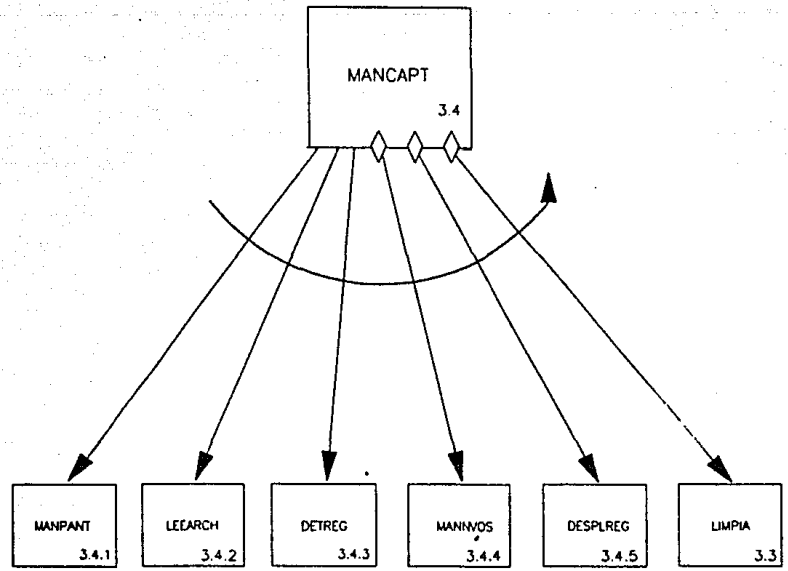
El módulo aferente OBTATRFOR no requiere de factorización dada su función limitada.

Respecto de las transformaciones centrales, el módulo MANMENU cuya función se limita a desplegar formatos de menú y leer opciones, no requiere de una mayor factorización, lo mismo sucede con el módulo LIMPIA y la transformación llamada FORMADOR que tiene como función, limpiar la pantalla, escribir formato y limpiar campos de entrada , y ordenar el registro validado respectivamente. El módulo MANCAPT por el contrario, realiza varias funciones que consisten en emular la pantalla de captura, detectar un registro nuevo (que no existia previamente), desplegar un registro ya existente y limpiar pantalla de captura, por lo que es necesario factorizar su función en 4 módulos subordinados de la manera en que se presentan en la figura III.12. En el capítulo siguiente, puede encontrarse una explicación con mas detalle de las funciones de cada uno de los módulos.



Parte del D.F.D. general correspondiente al modulo MANCAPT

fig. III.12 a



Factorizacion de la transformacion central MANCAPT

fig. III.12 b

Continuando con la factorización, el módulo que está habilitado para manejar los registros que resulten ser nuevos (MANNVOS), requiere a su vez de la colaboración de sus subordinados inmediatos. Los subordinados inmediatos mencionados son los siguientes:

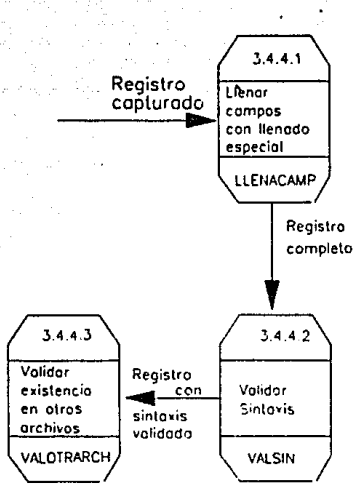
LLENACAMP que efectúa el llenado de los campos que requieran de alguna característica espcial para su llenado.

VALSIN que efectúa la validación de sintaxis de todos y cada uno de los campos capturado y llenados.

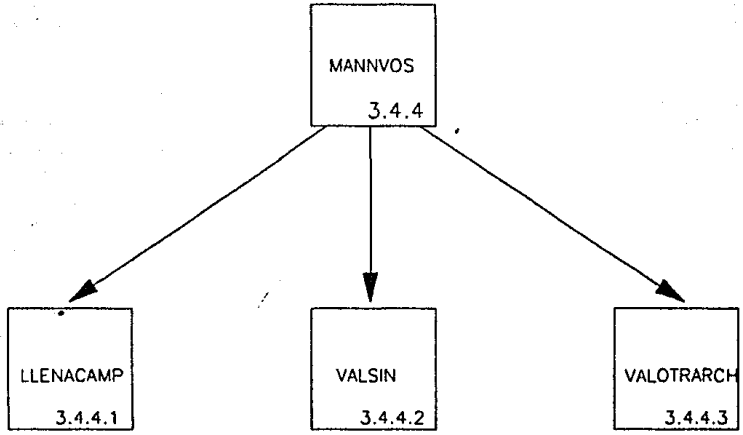
VALOTRARCH que efectúa una verificación de la existencia de un campo del registro capturado como llave de algún otro archivo.

Siguiendo el lineamiento adoptado, en el capítulo siguiente, se proporciona una explicación detallada de las funciones de cada uno de los módulos de la herramienta citados en éste capítulo.

En la figura III.13, se muestra la factorización del módulo MANNVOS.



Parte del D.F.D. general correspondiente al modulo MANNVOS



Factorizacion del modulo MANNVOS

fig. III.13

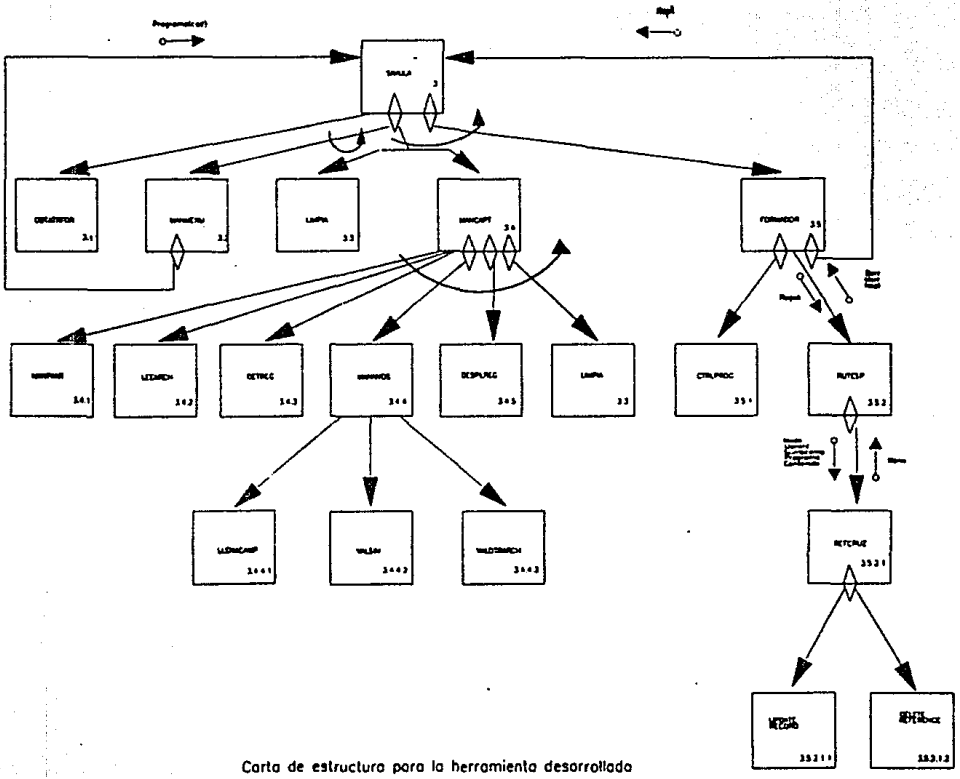
Finalmente, dado que no existen módulos que requieran un mayor factorización, se presenta la carta de estructura del sistema completo (fig. III.14) que tiene las siguientes características:

Se creó un módulo llamado CTRLPROG que permite dar mantenimiento a un archivo que contiene un catálogo de programas que se pueden emular, dado que se ha creado su medio ambiente previo necesario. Este módulo, tiene asociada una pantalla de captura que es emulada por la propia herramienta y por consiguiente resulta ser un usuario de la misma. Para cada pantalla a emular, se requiere tener un módulo o rutina que el propio especialista debe escribir para recibir el registro formado por el sistema y efectuar sobre él las operaciones específicas que requiera el sistema que se esté modelando.

El módulo CTRLPROG es un módulo que cumple con esas características y por lo tanto, en la carta de estructura aparece incluido como subordinado del módulo FORMADOR de la herramienta, del mismo modo que aparece RUTESP

(rutina escrita por especialista).

Los módulos REFCRUZ, UPDATE.RECORD y DELETE.REFERENCE, tienen como función, ejecutar las operaciones de mantenimiento para archivos de referencias cruzadas, cuando alguna rutina escrita por el especialista requiera de éste tipo de operaciones. Como ya se ha mencionado, el detalle de estas funciones se presenta en los capitulos IV y V.



Carta de estructura para la herramienta desarrollada

fig. III.14

C A P I T U L O I V

SEUDOCODIGO Y PROGRAMACION DEL SISTEMA.

IV.1. DESCRIPCION DE LA PROGRAMACION ESTRUCTURADA.

IV.2 ESTABLECIMIENTO DEL SEUDOCODIGO.

IV.1 DESCRIPCION DE LA PROGRAMACION ESTRUCTURADA

Una vez que la carta de estructura para un sistema se ha obtenido, se está en condiciones de programar cada uno de los módulos que formarán el sistema de información que se esté desarrollando.

Con el objeto de producir programas que sean leibles, comprensibles y modificables en un alto grado, la programación estructurada requiere del uso de estructuras de control que solo tengan una entrada y una salida. La técnica clásica usada para implementar este principio, es restringir las estructuras de control a las que a continuación se mencionan y eliminar el uso de la instrucción "GO TO".

Las estructuras de control a que se hace referencia son tres, y mediante la correcta combinación de dichas estructuras, se puede escribir un programa de cualquier nivel de complejidad:

- Secuencia. Representa el código de una o varias instrucciones simples. Su simbolo es:

```
CODIGO X.
```

- Decisión. Representa el código para manejar el flujo condicional del proceso. Tiene tres casos especiales:

a)
```
SI <PREDICADO 1>
     CODIGO A
O SI <PREDICADO 2>
     CODIGO B
O SI <PREDICADO 3>
     CODIGO C
.................
.................
O BIEN
     CODIGO Z.
FIN
```

b)
```
SI <PREDICADO>
     CODIGO A
O BIEN
     CODIGO B
FIN
```

c) SI <PREDICADO>

CODIGO A

FIN

- Iterativas. Representan el codigo para efectuar un proceso repetitivamente. Como en la estructura de control de decisión, tiene tres casos especiales.

a) DESDE I = <EXPRESION 1>

HASTA <EXPRESION 2> PASO N

CODIGO A

FIN

b) EN TANTO <PREDICADO>

CODIGO A

FIN

c) REPETIR

CODIGO A

HASTA <PREDICADO>.

En estas estructuras de control, PREDICADO X se refiere a operaciones lógicas del tipo 'Y', 'O', etc.

EXPRESION X, se refiere a expresiones aritméticas que evaluadas producen un valor a partir del cual se iniciará un proceso repetitivo o un valor que limitará el numero de iteraciones del proceso repetitivo.

El número N indica los incrementos que sufrirá el valor inicial obtenido de la expresión.

Para representar gráficamente el diseño de un programa, la programación estructurada hace uso de varias herramientas entre las cuales podemos mencionar:

- diagramas de Flujo.
- seudocódigo.
- diagramas HIPO.
- diagramas Warnier-Orr.
- y otras.

Es importante mencionar que las herramientas a que se hace referencia pueden ser usadas de acuerdo al tipo de representación que se desee obtener, esto significa que ninguna técnica domina sobre otra en cuanto a utilidad. Más importante que escoger la herramienta que se usará es usar alguna de ellas.

Para el sistema de modelado que se está desarrollando, se ha elegido la técnica de seudocódigo por considerar que es relativamente sencillo y directo el paso del seudocódigo a la codificación de los programas.

El seudocódigo es una notación corta para las estructuras de control y otros elementos de un lenguaje de programación que pueden ser escritos en Español normal o abreviado a solo unas palabras, frases y símbolos matemáticos.

En el seudocódigo, se puede incrementar el nivel de detalle para evitar confusión por medio del uso de comentarios en cualquier parte que sea necesario.

Con la intención de aumentar la facilidad para comprender el funcionamiento de los módulos del sistema, se ha tomado la siguiente convención:

Como encabezado, aparecerá el nombre del módulo, después una breve descripción del funcionamiento del mismo y finalmente su nombre corto y en su caso la lista de parámetros que utiliza. En el caso de existir parámetros se anexa una breve descripción de los parámetros usados.

IV.2. ESTABLECIMIENTO DEL SEUDOCODIGO.

En base a la carta de estructura obtenida en el capitulo anterior, se obtuvo el seudocódigo de los módulos del sistema de simulación de sistemas en linea que a continuación se lista.

De aqui en adelante, al hablar de programa a emular, se estará haciendo referencia al nombre que identifica a los 'atributos de pantalla' que caracterizarán a una pantalla de captura y a su archivo o archivos asociados, y no al código del programa como tal.

Es necesario tener en cuenta que el seudocódigo de los programas y módulos aqui descritos, están basados en su totalidad en la filosofia y caracteristicas (como se indicó en la introducción de este trabajo), del paquete Revelation y que es posible escribirlos en cualquier otro lenguaje diferente a R/BASIC siempre y cuando se implementen rutinas que ejecuten las mismas funciones que las que realizan las funciones intrinsecas de R/BASIC y se creen archivos que sustituyan a los propios del paquete.

Con el objeto de entender de una manera más clara el seudocódigo de los módulos a continuación descritos, es deseable que el lector se refiera previamente a los manuales del paquete Revelation.

MODULO SIMULADOR DE PROGRAMAS EN LINEA.

Este módulo es el más importante del sistema ya que se encarga de:

Leer del archivo de características de programas los atributos de campos y títulos para desplegarlos en pantalla así como las características del archivo o archivos asociados al programa a emular.

En caso de que el registro captado de pantalla exista previamente, se desplegará su contenido en los campos correspondientes para habilitar su posible modificación (excepto del primer campo que es la llave utilizada por el archivo) y posteriormente será validado.

Si por el contrario, el registro no existe, se asume que puede ser dado de alta por lo que se procede a validarlo.

Finalmente el registro validado se envía a la rutina asociada escrita por el especialista para que ésta última le de el tratamiento específico deseado.

En caso de que la rutina asociada requiera de enlazar a otro programa, el nombre del segundo estará en la variable napl y el módulo se llamará a si mismo recursivamente.

SIMULA(Apl). (3)

La variable apl es enviada a éste módulo como un parámetro al momento de su ejecución.

1. Llamar OBTATRFOR
2. Si Tipo = 'ENTRY'
   Llamar LIMPIA
   Yaexis = 0 (bandera para indicar cuando un registro existe en el archivo (1) o no existe (0))
   Repetir
     Mannvos = 0
     Llamar MANCAPT
     Si mannvos = 1 y band = 0
       Llamar FORMADOR
       Si band = 1
         Llamar ENVMSG(Mens)
       O bien
         Llamar LIMPIA
       Fin
     Fin
   Hasta que primer campo de registro sea igual a 'fin'
 O bien
   Llamar MANMENU

Fin

3. Regresar

MODULO PARA OBTENCION DE ATRIBUTOS DE
DE PANTALLA Y CAMPOS CUYO VALOR
DEBE SER EVALUADO (FORMULA)

Función: Obtener los atritbutos de pantalla del programa que se va a emular, es decir, el formato de la pantalla y las caracteristicas de los campos de captura. En el caso de que un campo requiera de la evaluación previa de una expresión para obtener su valor, (esto se indica con uno de los atributos), se generarán las rutinas necesarias para lograr la evaluación mencionada.

OBTATRFOR (3.1)

1. Abrir archivo de caracteristicas de programas a emular.
2. Si la apertura no fué satisfactoria
   Enviar mensaje de error
   Terminar
   Fin
3. Leer archivo de caracteristicas de programas con llave Apl y dejar registro leido en Caracrdes
4. Si la lectura no fué satisfactoria

```
        Enviar  mensaje  'El programa no ha  sido
        dado de alta en archivo de caracteristicas'
        Terminar
      Fin
5. Tipo = Caracrdes(1)
6. Si tipo = 'ENTRY'
        Pantalla = Caracrdes(11)
        Archivo = Caracrdes(15)
        Abrir archivo asociado con nombre Archivo
        Si la apertura no fué satsifactoria
          Enviar mensaje de error
          Terminar
        Fin
        Abrir diccionario de datos de archivo
        Si la apertura no fué satisfactoria
          Enviar mensaje de error
          Terminar
        Fin
        Atrib  = todos los campos desde la  segunda
                 ocurrencia del caracter 247  ASCII
                  hasta fin de caracrdes (atributos
                  de todos los campos)
        Asignar   valor   ''  a   las   variables
        alfanumericas y 0 a las numéricas
        Repetir
          En tanto descamp sea igual a ''
```

```
  S = s + 1
  Descamp = Atrib(s)
Fin
nocampos = nocampos + 1
Si   descamp(1)   diferente   de   ''   o
descamp(7) diferente de ''
  Enviar  mensaje  'No es  valido  usar
  campos multivaluados'
  Terminar
Fin
Posc(nocampos) = descamp(3)
Posr(nocampos) = descamp(4)
Convsal(nocampos) = descamp(6)
Formula(nocampos) = descamp(8)
Si fórmula(nocampos) = diferente de ''
  Abrir archivo SIMFOR
  Si la apertura no fué satisfactoria
    Crear archivo SIMFOR
    Abrir archivo SIMFOR
    Si la apertura no fué satisfactoria
      Enviar mensaje de error
      Terminar
    Fin
  Fin
  Inst = ' '
  Insti    =    'SUBROUTINE PROFOR '
```

```
  concatenado con nocampos
  En tanto Inst sea diferente de ''
    Noinst = noinst + 1
    Inst = fórmula(noinst)
    Si inst diferente de ''
      Inst1 = inst1 concatenado con
      inst
    Fin
  Fin
  Inst1 = Inst1 concatenado con
  'RETURN'
  Escribir en archivo SIMFOR Inst1
  Compilar programa
  Escribir en archivo VOC el programa
  compilado
Fin
Numcampo(nocampos) = descamp(9)
Máscara(nocampos) = decamp(11)
Si máscara(nocampos) es igual a
'MM/DD/AA'
  Máscara(nocampos) = 'DD/MM/AA'
Fin
Opcional(nocampos) = descamp(12)
Patrón(nocampos) = descamp(13)
VPO(nocampos) = descamp(14)
Archver(nocampos) = descamp(15)
```

Archnver(nocampos) = descamp(16)

Hasta que descamp(10) sea igual a 'CHANGE'

Fin

7. Regresar

MODULO PARA PROGRAMAS TIPO MENU

Función: Emular programas cuyo tipo sea 'MENU'.

MANMENU (3.2)

1. Repetir

Titmenú = campo 2 del registro leido del archivo de caracteristicas

Titulos = campo 13 del registro leido del archivo de caracteristicas

Programa = campo 14 del registro leido del archivo de caracteristicas

Limpiar pantalla

Escribir en pantalla Ap1 (posición 0,0)

Ptm = caracteres 2-último de titmenú

Pctm = caracteres 1-caracter antes de ','

Prtm = caracter después de ','-caracter antes de ')'

Escribir el titulo del menú en columna pctm y renglón prtm

Notitulos = 0

L = 0

Repetir

L = 1 + 1

```
  Si Titulos(i) diferente de ''
    Notitulos = notitulos + 1
  Fin
Hasta titulos = ''
Notitulos = notitulos + 1
Desde I = 1 hasta notitulos paso 1
  Titl = subcampo i de titulos
  Tit = caracter después de ')'-último
  Pos = caracteres 1-caracter antes de ')'
  Posc = caracteres 2-caracter antes de ','
  de pos
  Posr  = caracter después de ','-último de
  pos
  Si i < notitulos
    Escribir en pantalla en columna posc -
    3 y renglón posr i
    Escribir  en pantalla en columna posc y
    renglón posr, tit
  O bien
    Escribir en pantalla en columna posc  y
    renglón  posr,  tit concatenado con '['
    concatenado  con  SPACE(longitud  de
    notitulos - 1) concatenado con ']'
  Fin
Fin
Escribir en col 65 ren 22 'Envie página  ['
```

```
concatenado con espacio concatenado con '3'
Cad = ''
Repetir
  Vuelt = 0
  X = longitud de notitulos - 1
  Presp = posc + longitud de tit + 1
  Y = 0
  Repetir
    Anali = 0
    Y = y + 1
    Si y < = x
      Posicionar  cursor en col presp + y y
      ren posr
      Leer de pantalla caracter tecleado  y
      asignar a car
      Si código de car >= 48 y <= 57
        Caracter y de cad = car
      O si car = caracter de corrección
      (<-)
        Y = y - 1
        Si y diferente de 0
          Caracter y de cad = ' '
          Sustituir caracter en la posición
          del cursor por un espacio
          Y = y - 1
        Fin
```

```
    O si car = avance de carácter
      Si longitud de cad = 0
        Y = y - 1
      Fin
    O si car = caracter de borrado (del)
      Caracter y de cad = ''
      Escribir nuevo cad en pantalla
      Y = y - 1
    O si car = avance del tabulador
      Anali = 1
    O si car = retroceso del tabulador
      Anali = 1
    O bien
      Escribir  un espacio en la posición
      del cursor
    Fin
  O bien
    Anali = 1
  Fin
Hasta que anali = 1
Repetir
  Posicionar cursor en clo 78 reng 22
  Leer caracter tecleado y asignar a car
  Si car = avance de caracter o avance de
  tabulador o retroceso de tabulador
    Vueli = 1
```

```
        O si car = regresa (End)
          Si cad diferente de ''
            Si cad < 1 o > notitulos - 1
              Llamar ENVMSG('Opción fuera de
              rango')
              Limpiar campo de entrada de
              opción en pantalla
              Cad = ''
              Vuel1 = 1
            Fin
            Si vuel1 = 0
              Llamar SIMULA(Programa(cad))
            Fin
          Fin
        Fin
      Hasta car = avance de caracter o avance
      de tabulador o retroceso de tabulador
    Hasta vuel1 = 0
  Hasta cad = ''
2. Regresar
```

MODULO PARA LIMPIAR PANTALLA Y ESCRIBIR FORMATO EN PANTALLA

Función: Igualar la variable que contiene los campos que se capturan en pantalla a '' y las banderas numéricas a 0 antes de escribir en pantalla el formato leido en el módulo OBTATRFOR.

LIMPIA (3.3)

1. Cadena = ''
2. Est = ''
3. Escribir en pantalla la variable Pantalla
4. N = s - 1
5. Desde O = 1 hasta N paso 1
   Si máscara(o) diferente de ''
   Escribir en columna Posc(o) - 1 y renglón Posr(o) '[' concatenado con máscara(o) y concatenado con ']'
   Fin
   Fin
6. Escribir en columna 65 renglón 22 'ENVIE PAGINA[' CONCATENADO CON ' ' y con ']'
7. Regresar

MODULO MANEJADOR DE PROGRAMAS TIPO CAPTURA

Función: Procesar los programas a emular tipo 'Entry'

MANCAPT (3.4)

```
1. Llamar MANPANT
2. Band = 0
3. Si registro(1) es diferente de 'fin'
     Si yaexis = 0
       Regis = registro(1)
       Llamar LEEARCH
       Si la lectura no fué satisfactoria
         Vararch = ''
       Fin
     Fin
     Llamar DETREG
     Si detr = 1
```

```
      Llamar MANNVOS
    Fin
    Si  (yaexis  = 0 y vararch  diferente  de
    '')
      yaexis = 1
      Llamar DESPLREG
    Fin
    Si yaexis = 1 y registro igual a ant
      Yaexis = 0
      Llamar LIMPIA
    Fin
4. Regresar
```

MODULO MANEJADOR DE PANTALLAS

Función: Aceptar todos los caracteres que se tecleen en pantalla (caracteres de control, edición, alfanuméricos y símbolos especiales), para interpretarlos. En caso de ser de control o de edición, se ejecuta la función correspondiente y cuando se trata de un caracter alfanumérico o símbolo especial, lo concatena a la cadena de caracteres que formará para cada campo en la variable Registro.

Además de la variable yaexis, éste módulo maneja una bandera (Est) que toma el valor de 1 cuando la cadena en proceso no se esté capturando por primera vez sino que se esté editando, en otras palabras, el campo ya había sido capturado previamente y se le están haciendo modificaciones como por ejemplo en casos de corrección de errores; y tendrá un valor de 0 cuando el campo esté vacio y se comience a capturar.

La bandera bandtrans tendrá un valor de 1 para indicar que el cursor se encuentra en la

zona final de la pantalla y làs cadenas están listas para ser transmitidas; tomará un valor de 0 cuando el cursor se encuentre en otra posición.

MANPANT (3.4.1)

```
1. Asignar valores a variables de edición y
   control
2. Si band = 0
     Si yaexis = 0
       Desde 0 = 1 hasta n paso 1
         Cadena (o) = ''
         Est(o) = ''
       Fin
     Fin
   Fin
3. I = 1
4. Ydel = 0
5. Repetir
     Ydel = 0
     Regform = 0
     Ret1 = 0
     Si (máscara(i) diferente de '') o (i = s)
       Si (yaexis = 0) o i diferente de 1
```

```
Si  (opcional(i) dif de 'P') y  dif  de
'FP'
  Si i = s
    Posc(i) = 78
    Posr(i) = 22
    Máscara(i) = 'XX'
    Bandtrans = 1
  Fin
  L = 0
  En  tanto L <= longitud de máscara(i)
  - 1
    Posicionar  cursor  en  la  columna
    Posc(i) + 1 y renglón posr(i)
    Leer de pantalla caracter tecleado
    Car = caracter leido
    Si bandtrans = 1
      Si car = avance de tabulador
        no operación
      O si car = regresa (return)
        no operación
      O si car = regreso de tabulador
        Repetir
          I = i - 1
        Hasta que (opcional(i) = 'P') o
        a 'FP'
        Bandtrans = 0
```

```
    Ret1 = 1
  O si car = avance de caracter
    I = 1
    Bandtrans = 0
    Ret1 = 1
  Fin
Fin
Si Ret1 = 0
  Si  codigo de caracter >= 32 y <=
  96
    Si Est(i) = 1
      Sustituir    el    caracter
      en la posición del cursor por
      el    caracter    leido    en
      cadena(i) y pantalla
    O bien
      Concatenar  a  la  cadena(i)
      existente, el caracter leido
    Fin
    L = l + 1
  O si car = avance de caracter
    Si longitud de cadena(i) > 0
      L = l + 1
    Fin
    Si i = s
      Ret1 = 1
```

```
  Fin
O si car = caracter de corrección
(<--)
  Si posc(i) + 1 > posc(i)
    Si est(i) = 1
      Sustituir el caracter en la
      posición del cursor por un
      espacio  en la cadena(i)  y
      en pantalla
    O bien
      Eliminar el caracter en  la
      posición del cursor de la
      cadena(i) y de la pantalla
    Fin
    L = 1 - 1
  Fin
O si car = avance de tabulador
  Si longitud de cadena(i) > 0
    Est(i) = 1
  Fin
  I = i + 1
  Bandtrans = 0
  Si i = s
    L = 0
    Posc(i) = 78
    Posr(i) = 22
```

```
    Máscara(i) = 'xx'
    Bandtrans = 1
  O bien
    Si i > s
      I = i - s
    Fin
    Ret1 = 1
  Fin
O si car = retroceso de tabulador
  Si longitud de cadena(i) > 0
    Est(i) = 1
  Fin
  I = i - 1
  Bandtrans = 0
  Si i < 1
    I = s
    L = 0
    Posc(i) = 78
    Posr(i) = 22
    Máscara(i) = 'xx'
    Bandtrans = 1
  O bien
    Ret1 = 1
  Fin
O  si  car = caracter de  borrado
(Del)
```

```
Si longitud cadena(i) > 0
  Si l = 0
    Cadena(i) = cadena(i) del
    segundo al último caracter
  Fin
  Si (l = longitud de
  máscara(i)) y ydel = 0
    Cadena(i) = cadena(i) del
    primer al penáltimo
    caracter
    Ydel = 1
  Fin
  Si (l > 0) y (l < longitud
  de máscara(i))
    Cadena(i) = cadena(i)
    eliminando el caracter en
    la posición del cursor
  Fin
  Si longitud de cadena(i)
  resultante > longitud de
  máscara(i)
    Eliminar solo en pantalla
    los caracteres de la
    derecha que exceden la
    longitud
  Fin
```

```
  Fin
O si car = regresa (End)
  Registro = ''
  Si banditrans = 1
    Desde O = 1 hasta  N paso 1
      Registro     =     Registro
      concatenado  con  cadena(o)
      con   los   espacios   a   la
      izquierda  y  a  la  derecha
      eliminados
      Est(o) = 1
    Fin
  O bien
    Desde O = 1 hasta I paso 1
      Registro     =     registro
      concatenado  con  cadena(o)
      con   los   espacios   a   la
      izquierda  y  a  la  derecha
      eliminados
    Fin
    Desde  O = I + 1 hasta N paso
    1
      Registro(o) = ''
    Fin
  Fin
  Regform = 1
```

```
            Fin
          Fin
          Si reti = 1 o regform = 1
            L = longitud de máscara(i)
          Fin
        Fin
      Fin
    Fin
  Fin
  Si reti = 0
    I = i + 1
  Fin
Hasta que regform = 1
```

6. Regresar

MODULO PARA LEER ARCHIVO ASOCIADO A LA PANTALLA.

Función: Leer archivo asociado a la pantalla que se está emulando.

LEERARCH (3.4.2)

1. Leer archivo asociado a prog con llave regis y dejar reg. leido en vararch
2. Regresar.

MODULO PARA DETECTAR REGISTROS NUEVOS.

Función: Detectar si el registro que se ha capturado existia previamente o se trata de un registro nuevo.

DETREG (3.4.3)

1. Si (yaexis = 1 y registro diferente de ant) o vararch igual a ''
   Detr = 1
   O bien
   Detr = 0
   Fin
2. Regresa

MODULO MANEJADOR DE REGISTROS NUEVOS

Función: Procesar los registros nuevos, es decir, los registros que no existían previamente en el archivo asociado o bien los registros que existiendo previamente en el archivo, hayan sido modificados.

MANNVOS (3.4.4)

```
1. I = 1
2. En tanto I <= nocampos
     Llamar LLENACAMP
     Si band = 0
       Si   patrón  (de   validaciones)
       diferente de ''
         Si  el campo(I) no es opcional  o
         (si es opcional y diferente de '')
           Pat = patrón (I)
           Nopat = 0
           Repetir
             Nopat = nopat + 1
             PatrónI(nopat) = subcampo
             nopat de Pat
           Hasta que patronI(nopat) = ''
           Nopat = nopat - 1
```

```
            Desde p = 1 hasta nopat paso 1
              Llamar VALSIN
              Si band = 1
                Llamar ENVMSG(mens)
              Fin
            Fin
          Fin
        Fin
        Si band = 0
          Llamar VALOTRARCH
          Si band = 1
            Llamar ENVMSG(mens)
          Fin
        Fin
      Fin
      Si band = 1
        I = nocampos + 1
      Fin
    Fin
3. Mannvos = 1
4. Regresar
```

MODULO PARA DESPLEGAR EN PANTALLA EL REGISTRO LEIDO CUANDO YA EXISTIA EN EL ARCHIVO ASOCIADO

Función: Desplegar en los campos y posiciones correspondientes en pantalla, el registro leido del archivo asociado al programa a emular.

DESPLREG. (3.4.5)

```
1. C = 0
2. Registro = ''
3.  Desde F = 1 hasta longitud de registro leido
   paso 1
     Si el caracter F tiene código 254
       C = c + 1
     Fin
   Fin
4. C = c + 1
5. Desde D = 1 hasta N paso 1
     Ant(d) = ''
   Fin
6. Desde D = 2 hasta N paso 1
     Desde I = 1 hasta c paso 1
       Si i = nocampo(i)
         Regleido(i) = OCONV(regleido(i),
         convsal(i))
```

```
        Yaexis = 1
        Si regleido(i) diferente de ''
          Est(d) = 1
        Fin
        Ant(d) = regleido(i)
        Si  longitud de regleido(i) >  longitud
        de máscara(d)
          Eliminar  los caracteres que  exceden
          la  longitud de máscara(i)  (solo  en
          pantalla)
        Fin
        Escribir en pantalla regleido(i)
      Fin
    Fin
  Fin
7. Ant = Regis concatenado con Ant
8. Cadena = ant
9. Bandtrans = 0
10. Regresar
```

MODULO PARA LLENAR CAMPOS CON CONVERSION DE SALIDA, VALOR POR OMISION Y FORMULAS

Función: Revisar los atributos de los campos para determinar cual de ellos requiere de un 'llenado especial' o alguna conversión intrinseca del paquete para obtener su valor.

LLENACAMP (3.4.4.1)

1. Si (yaexis = 1) diferente de 1) o i diferente de 1
   Si convsal(i) diferente de ''
     Si existen varios subacmpos de conversión de salida
       Convsal(i) = primer subcampo de convsal(i)
     Fin
   Fin
   Si (opcional(i) = 'P') o igual a 'FP'
     Llamar ENVMSG('El campo 'concatenado con I concatenado con 'está protegido')
   Fin
   Si (opcional(i) = 'R') y (registro(i)

```
= '')
  Llamar     ENVMSG('El    campo     '
  concatenado  con I  concatenado  con
  'es requerido')
  Band = 1
  Bandtrans = 0
Fin
Si band = 0
  Si fórmula(i) diferente de ''
    Rutina  = 'PROFOR' concatenado con
    I
    Llamar      @Rutina      (compilada
    previamente en OBTATRFOR)
    Resp = valor calculado en @Rutina
    Si  longitud de resp > longitud de
    máscara
      Eliminar  los caracteres  de  la
      derecha  que exceden la longitud
      de máscara (solo en pantalla)
    Fin
    Escribir resp en pantalla
  Fin
  Si  (VPO(i)   diferente   de   '')  y
  (registro(i) = '')
    Si VPO(i) = '''
      Val = OCONV(Ant(i),convsal(i))
```

```
O bien
  Si primer caracter de VPO es
  diferente de '%' y de '{' y de
  '['
    Val = vpo(i)
  O bien
    Si vpo(i) = '%D%'
      Val=OCONV(DATE(),convsal(i))
    O si vpo(i) = '%T%'
      Val=OCONV(TIME(),convsal(i))
    O si vpo(i) = '%TD%'
      Val=TIMEDATE()
    O si primer caracter de vpo(i)
    = '{'
      Nul  = expresión entre los
      caracteres '{' y '}'
      Val = CALCULATE(nul)
      Si @CONV diferente de ''
        Val=OCONV(val,convsal(i))
      Fin
    O si primer caracter de vpo(i)
    = '['
      Val=expresión    entre   los
      caracteres '[' y ']'
      Llamar @Val
      Val = @Ans
```

```
            Fin
          Fin
        Fin
        Registro(i) = val
        Mod(i) = 1
        Si val diferente de ''
          Est(i) = 1
          Cadena(i) = val
          Limpiar campo i en pantalla
          Si  longitud de val  >  longitud
          máscara(i)
            Eliminar  caracteres  de  la
           derecha que exceden la longitud
           de máscara
          Fin
          Escribir  Val en pantalla en  su
          campo correspondiente
        Fin
      Fin
  Fin
2. Regresar
```

MODULO DE VALIDACION SINTACTICA

El módulo de validación sintáctica tiene como función, efectuar sobre los campos que forman el registro captado en pantalla, una validación de tipo sintáctica.

Los parámetros de validación, son leidos del archivo de caracteristicas (RDES) en el módulo OBTATRFOR para su interpretación en éste módulo.

VALSIN (3.4.4.2)

```
1. Exp1 = patróni(p)
2. Mens = ''
3. Si caracter 1 de exp1 = '('
     Si caracter 2 de exp1 = 'D'
       Regfec = registro(i)
       oc = convsal(i)
       Si mod(i) = 1
         Si oc = 'D'
           Dia  =  caracteres  1 y 2 de regfec
           Si caracteres 4-6 de regfec = 'JAN'
             Mes = '01'
           Osi caracteres 4-6 de regfec = 'FEB'
             Mes = '02'
```

```
    Osi caracteres 4-6 de regfec = 'MAR'
      Mes = '03'
    Osi caracteres 4-6 de regfec = 'APR'
      Mes = '04'
    Osi caracteres 4-6 de regfec = 'MAY'
      Mes = '05'
    Osi caracteres 4-6 de regfec = 'JUN'
      Mes = '06'
    Osi caracteres 4-6 de regfec = 'JUL'
      Mes = '07'
    Osi caracteres 4-6 de regfec = 'AGO'
      Mes = '08'
    Osi caracteres 4-6 de regfec = 'SEP'
      Mes = '09'
    Osi caracteres 4-6 de regfec = 'OCT'
      Mes = '10'
    Osi caracteres 4-6 de regfec = 'NOV'
      Mes = '11'
    Osi carcateres 4-6 de regfec = 'DIC'
      Mes = '12'
    Fin
    Ako = caracteres 10-11 de regfec
  Osi oc = 'D2-'
    Mes = caracteres 1-2 de regfec
    Dia = caracteres 4-5 de regfec
    Ako = caracteres 7-8 de regfec
```

```
  Osi oc = 'D4-'
    Mes = caracteres 1-2 de regfec
    Dia = caracteres 4-5 de regfec
    AKo = caracteres 9-10 de regfec
  Osi oc = 'D2-E'
    Dia = caracteres 1-2 de regfec
    Mes = caracteres 4-5 de regfec
    AKo = caracteres 7-8 de regfec
  O si oc = 'D4-E'
    Dia = carcateres 1-2 de regfec
    Mes = caracteres 4-5 de regfec
    AKo = caracteres 9-10 de regfec
  Fin
O bien
  Si  regfec  es  de  seis  caracteres
  numéricos
    Insertar  caracter  '-'  entre
    caracteres 2 y 3 y 4 y 5
  O bien
    Si  caracter  3  o  caracter  6  son
    numéricos
      Mens  = 'Campo ' concatenado con  I
      concatenado con 'Fecha inválida'
      Band = 1
      Regresar
    Fin
```

```
      Dia = caracteres 1-2 de regfec
      Mes = caracteres 4-5 de regfec
      Ako = caracteres 7-8 de regfec
    Fin
  Fin
  Si (mes <= 12) y (mes >= 1)
    Si dia < 1
      Band = 1
      Mens = 'Campo ' concatenado con I
      concatenado con 'Dia fuera de rango'
      Regresar
    Fin
    Si mes = 1 o = 3 o = 5 o = 7
      Si dia > 31
        Mens = 'Campo ' concatenado con i
        concatenado con 'dia fuera de
        rango'
        Band = 1
        Regresar
      Fin
    O si mes = 8 o = 10 o = 12
      Si dia > 31
        Mens = 'Campo 'concatenado con i
        concatenado con 'dia fuera de
        rango'
        Band = 1
```

```
      Regresar
    Fin
  O si mes = 2
    Si Ako es divisible por 4
      Si dia > 29
        Mens = 'Campo ' concatenado con i
        concatenado con 'dia fuera de
        rango'
        Band = 1
        Regresar
      Fin
    O bien
      Si dia > 28
        Mens = 'Campo ' concatenado con i
        concatenado con 'dia fuera de
        rango'
        Band = 1
        Regresar
      Fin
    Fin
  O si mes = 4 o mes = 6 o mes = 9 o = 11
    Si dia > 30
      Mens = 'Campo ' concatenado con i
      concatenado con 'dia fuera de
      rango'
      Band = 1
```

```
        Regresar
      Fin
    Fin
  O bien
    Mens = 'Campo ' concatenado con i
    concatenado con 'mes fuera de
    rango'
    Band = 1
    Regresar
  Fin
  Si mod(i) = 1
    Registro(i) = ICONV(registro(i),oc)
  O bien
    Registro(i) = ICONV(regfec,'D2-E')
  Fin
O si caracter 2 de exp1 = 'H' '
  Si caracter 3 de exp1 = 'T'
    Reghor = registro(i)
    Reghor = ICONV(Reghor,caracteres 2-
    último de exp1)
    Si reghor <= 86399 y > 0
      Registro(i) = reghor
    O bien
      Mens = 'Campo ' concatenado con i
      concatenado con 'hora inválida'
      Band = 1
```

```
        Regresar
      Fin
    O si caracter 3 de exp1 = 'D'
      Regdec = registro(i)
      Regdec  =  ICONV(regdec,caracteres  2-
      último de exp1)
      Si regdec no es numérico
        Mens = 'Campo ' concatenado con i
        concatenado  con  'debe ser numérico'
        Band = 1
        Regresar
      O bien
        Registro(i) = regdec
      Fin
    Fin
  O  si (caracter 2 de exp1 es numérico) o  =
  '-'
    Rang1 = caracteres 2-1 antes de ','
    Rang2  =  caracteres  1 después  de  ','-
    último
    Si algún caracter de exp1 es ','
      Si rang1 y rang2 son numéricos
        Si (registro(i) < rang1) o (registro(i)
        > rang2)
          Mens = 'Campo ' concatenado con i
          concatenado  con  'fuera  de  rango'
```

```
        concatenado con exp1
        Band = 1
        Regresar
      Fin
    Fin
  O bien
    Rang1 = caracteres 2-último de exp1
    Si rang1 es numérico
      Si registro(i) > rang1
        Mens = 'Campo ' concatenado con i
        concatenado con 'fuera de rango'
        concatenado con exp1
        Band = 1
        Regresar
      Fin
    Fin
  Fin
O si caracter 1 de exp1 = '('
  @ANS = registro(i)
  @ANS = CALCULATE(caracteres 2-último de
  exp1)
  Si @ANS = 0
    Mens = 'Campo ' concatenado con i
    concatenado con 'no satisf. reg. dicc'
    concatenado con caracteres 2-último de
    exp1
```

```
    Band = 1
    Regresar
  Fin
O si caracter 1 de exp1 = 'C'
  Regenv = registro(i)
  Ruti = caracteres 2-último de exp1
  Llamar @ruti(Regenv,verif)
  Si verif = 0
    Mens = 'Campo ' concatenado con i
    concatenado con 'no satisf. rutina '
    concatenado  con  ruti
    Band = 1
    Regresar
  Fin
O si caracter 1 de exp1 numérico
  Si registro(i) no satisface exp1
    Mens = 'Campo ' concatenado con i
    concatenado con 'no satisf. patrón'
    concatenado  con  exp1
    Band = 1
    Regresar
  Fin
O si caracter 1 de exp1 = ''' o = '*'
  Si  registro(i) es diferente de  caracteres
  2-último de exp1
    Mens = 'Campo ' concatenado con i
```

```
        concatenado con 'no satisf. patrón'
        concatenado  con  exp1
        Band = 1
        Regresar
      Fin
    O bien
      Mens = 'Campo ' concatenado con i
      concatenado con 'patrón inválido'
      concatenado  con  exp1
      Band = 1
      Regresar
    Fin
4. Regresar
```

MODULO DE VERIFICACION DE EXISTENCIA

O NO EXISTENCIA EN UN ARCHIVO

Función: Dado que Revelation permite verificar que un campo de un registro exista o no exista como llave en otro archivo, la función que realiza el módulo VALOTRARCH es habilitar ésta facilidad.

VALOTRARCH (3.4.4.3)

```
1. Si archver(i) diferente de ''
     Check = 'XXX'
     Av = archver(i)
     Vf = subcampo 1 de av
     Ca = subcampo 2 de av (caracteres
     concatenados a la izquierda del nombre del
     archivo)
     Abrir archivo Vf
     Si apertura no fué satisfactoria
       Mens = 'Arch. ver. ' concatenado con Vf
       concatenado con 'no abierto para campo'
       concatenado con I
       Band = 1
       Bandtrans = 0
       Return
```

```
    Fin
    Si caracter 1 de ca = '('
      Ca = carcteres 1-dltimo de ca
      Ca = CALCULATE(ca)
    Fin
    Concat = ca concatenado con registro(i)
    Leer archivo Vf con llave concat
    Si la lectura no fué satisfactoria
      Check = ''
    Fin
    Si check = ''
      Mens = concat concatenado con 'no
      encontrado en ' concatenado con Vf
      Band = 1
      Bandtrans = 0
      Return
    Fin
  Fin
2. Si archnver(i) diferente de ''
    Check = '%%%'
    Anv = archnver(i)
    Nvf = subcampo 1 de Anv
    Ca = subcampo 2 de anv (misma situación del
    anterior)
    Abrir archivo nvf
    Si apertura no fué satisfactoria
```

```
    Mens = 'Arch no ver. 'concatenado con Nvf
    concatenado con 'no abierto para campo'
    concatenado con I
    Band = 1
    Bandtrans = 0
    Return
  Fin
  Si caracter 1 de ca = '('
    Ca = carcteres 1-dltimo de ca
    Ca = CALCULATE(ca)
  Fin
  Concat = ca concatenado con registro(i)
  Leer archivo nvf con llave concat
  Si la lectura no fué satisfactoria
    Check = ''
  Fin
  Si check diferente de ''
    Mens = concat concatenado con '
    encontrado en ' concatenado con nvf
    Band = 1
    Bandtrans = 0
    Return
  Fin
Fin
```

MODULO FORMADOR DE REGISTRO VALIDADO.

Función: Formar el registro completo que se enviará a la rutina asociada escrita por el especialista. Formar el nombre de la rutina asociada escrita por el especialista tomando para ello algún criterio en la nomenclatura, por ejemplo: dado que los nombres de programas y archivos se forman de 6 caracteres (en PC), insertar entre los 3 primeros caracteres y los 3 últimos de la variable Apl,(que contiene el nombre del programa que se está emulando) una 'R' y tomar este criterio como estandar para formar todos los nombres de rutinas asociadas. En otras palabras, si el programa que se está emulando se llama CHQ123, el especialista debe escribir y compilar una rutina llamada CHQR123.

FORMADOR (3.5)

```
1. Desde w = 2 hasta nocampos paso 1
     Nocampo  =  numcampo(w)
     Si nocampo = ''
       RegoK  =  regoK concatenado con  '&'
       concatenado con registro(w) concatenado
       con '&'
```

```
      Fin
      Regok(nocampo) = registro(w)
    Fin
2. Regok = regis concatenado con regok
3. @RECORD = regok
4. Ant = regok
5.  Aplcom  = caracteres 1-3 de Apl  concatenado
con 'R' concatenado con caracteres 4-6 de apl
6. Llamar @Aplcom(Regok,berr,merr,napl)
7. Si berr = 1
      Mens = merr
      Band = 1
      bandtrans = 0
    O bien
      Si napl diferente de ''
        Llamar SIMULA(Napl)
      Fin
8. Regresar
```

MODULO DE CONTROL DE PROGRAMAS A EMULAR.

Función: Dar mantenimiento a un archivo que contiene un catálogo de los programas a emular así como a un archivo que contiene los programas asociados escritos por el especialista. Un programa se dará de alta solo cuando se haya creado previamente su medio ambiente. Cabe hacer la alcaración de que éste módulo no forma parte del sistema y es por el contrario un usuario del mismo, pero se incluye su seudocódigo porque permite el correcto funcionamiento del resto del sistema.

CTRLPROG(Reg,berr,merr,apl) (3.5.1)

La variable Reg es enviada a éste módulo por el programa SIMULA y contiene un registro de tres campos que son: nombre del programa que se desea dar de alta o modificar, clave de borrado y nombre de la rutina escrita por el especialista asociada.

La variable berr es un bandera que cuando vale 1 indica que se presento un error y en caso contrario vale 0.

La variable merr contiene los mensajes de error que se generen. Las variables berr, merr y apl se envian al módulo SIMULA para que éste las interprete.

La variable apl en éste módulo no se usará y tendrá siempre el valor ''.

Al hacer referencia al archivo de caracteristicas de los programas a emular, se debe entender como el archivo RDES del paquete en lo sucesivo.

1. Apl = '',berr = 0, merr = ''.
2. Abrir archivo de nombres de programas a emular.
3. Si la apertura no fue satisfactoria
   enviar mensaje de error
   terminar
   Fin
4. Llavearch = primer campo registro recibido del módulo SIMULA.
5. Abrir archivo de caracteristicas de programas a emular.
6. Si la apertura no fué satisfactoria
   enviar mensaje de error
   terminar
   Fin

7. Leer archivo de caracteristicas de programas a emular con llavearch.

8. Si la lectura no fué satisfactoria

berr = 1

merr = 'El programa no se encuentra en archivo de caracteristicas'

regresar

Fin

9. Tipo = campo de tipo de programa del registro obtenido de la lectura del archivo de caracteristicas

10. Si Tipo es diferente de 'ENTRY' o 'MENU'

berr = 1

merr = 'El programa no es de tipo válido'

regresar

Fin

11. Clave = segundo campo del registro recibido del módulo SIMULA

12. Si clave es igual a 'X'

Leer archivo de nombres de programas a emular con llavearch y asignar campo a Ya

Si no fue satisfactoria la lectura

Ya = ''

Fin

Si Ya es diferente de ''

```
        desplegar "Introduzca 'S' si desea dar de
                  baja la aplicación"
        Leer de pantalla R
        Si R es igual a 'S'
          Eliminar registro con Llavearch de
          archivo de nombres de programas
          Eliminar del archivo VOC el programa
          escrito por el especialista asociado
          Regresar
        O Bien
          Segundo campo del reg. recibido del
          módulo SIMULA = ''
        Fin
      O Bien
        berr = 1
        merr = "La aplicación no existe"
        regresar
      Fin
    Fin
13. RegoK = último campo del reg. recibido del
          módulo SIMULA
14. Si tipo es "ENTRY"
      Abrir archivo de programas escritos por el
      especialista
      Si la apertura no fué satisfactoria
        berr = 1
```

```
        merr = 'No fue posible abrir el arch  de
                de programas asociados'.
        regresar
      Fin
      Leer  archivo  de programas  escritos  por
      especialista con llave RegoK
      Si la lectura no fue satisfactoria
        berr = 1
        merr  = 'El prog objeto no existe  en
                 archivo de programas asociados'
        regresar
      Fin
      Dar de alta en VOC el programa escrito por
      el especialista asociado
    Fin
15. Escribir  en archivo de nombres de programas
    nuevo programa con llave RegoK
16. Regresar
```

MODULO DE LA RUTINA ESCRITA POR EL ESPECIALISTA

Función: Procesar de una manera particular el registro generado y validado por el sistema simulador. En otras palabras, el especialista que está haciendo uso del sistema deberá escribir éste módulo para realizar las operaciones específicas que debe realizar su modelo, como por ejemplo actualizar archivos, regresar al sistema el nombre del programa lógicamente enlazado al actual para simularlo, efectuar otro tipo de validaciones y en su caso generar nuevos mensajes de error, etc.

RUTESP (3.5.2)

MODULO MANEJADOR DE REFERENCIAS CRUZADAS

Función: Efectuar el mantenimiento necesario para el caso de que la rutina escrita por el especialista haga uso de archivos de referencias cruzadas.

En éste módulo, aparecen algunas variables con sus nombres en inglés debido a que es una copia de la rutina original del paquete con algunas modificaciones para que maneje los mensajes en español y éstos puedan ser devueltos a las rutinas que lo invocaron.

REFCRUZ(Modo,llaverc,nombcamp,programa,contenido,mens)

(3.5.2.1)

Modo. Contiene el tipo de operación que se efectuará (D borrado, U actualización)

Llaverc. Contiene llave para accesar archivo que usa el programa.

Nombcamp. Nombre del campo sobre el que se tiene la referencia cruzada.

Programa. Nombre del programa del que viene el registro.

Contenido. Contenido del campo que se va a grabar en archivo de referencias cruzadas.

Mens. Parámetros para enviar mensajes de error a la rutina que invocó a éste módulo.

1. Mens = ''

2. Abrir archivo de caracteristicas (RDES)

3. Si la apertura no fué satisfactoria

    Mens = 'No fué posible abrir archivo RDES para referencias cruzadas'

    Regresar

  Fin

4. Leer archivo RDES con llave Programa y asignar el campo 15 del reg leido a arcppal

5. Si la lectura no fué satisfactoria

    Mens = 'No hay arch ppal especificado en el programa ' concatenado con programa concatenado con ' para R.C.'

    Regresar

  Fin

6. Abrir diccionario de datos de arcppal

7. Abrir arcppal

8. Llaveusu = 'XR*' concatenado con arcppal concatenado con '*' concatenado con nombcamp

9. Leer RDES con llave llaveusu

10. Si la lectura no fué satisfactoria

    Mens = 'El arch de parámetros' concatenado con llaveusu concatenado con 'no se encuentra en RDES'

    Regresar

  Fin

```
11.Up.fn     =  campo 1 de reg leido de  arch  de
parametros
12.Up.amc    =  campo 2 de reg leido de  arch  de
parametros
13.All.delim=  campo 3 de reg leido de  arch  de
parametros
14.Length    =  campo 4 de reg leido de  arch  de
parametros
15.Sort      =  campo 5 de reg leido de  arch  de
parametros
16.Del       =  campo 6 de reg leido de  arch  de
parametros
17.Delim = ''
18.Repetir
     Car = caracter 1 de all.delim
     Si car diferente de ''
       Si caracteres 1-5 de all.delim = 'SPACE'
         Delim = delim concatenado con ' '
         Caracteres 1-5 de all/delim = ''
       O si caracteres 1-3 de all.delim = 'SVM'
         Delim = delim concatenado con CHAR(252)
         Caracteres 1-3 de all.delim = ''
       O si caracteres 1-2 de all.delim = 'VM'
         Delim = delim concatenado con CHAR(253)
         Caracteres 1-2 de all.delim = ''
       O bien
```

```
          Delim = car
          Caracter 1 de all.delim = ''
        Fin
      Fin
    Hasta car = ''
19.Dict = ''
20.Abrir archivo up.fn
21.Si la apertura no fué satisfactoria
     Mens = 'Archivo' concatenado con up.fn
     concatenado con 'de R.C. no pudo ser
     abierto'
     Regresar
   Fin
22.Delim.length = longitud de delim
23.Si delim.length = 1
     A = 0
     Repetir
       A = a + 1
       Word = Contenido(a) usando como
       delimitador delim
       Si word = ''
         Regresar
       Fin
       Si word diferente de ' '
         word = word con espacios a la
         izquierda y a la derecha eliminados
```

```
    Fin
    Si longitud de word > length
      Si modo = 'D'
        Llamar DELETE.REFERENCE
      O bien
        Llamar UPDATE.RECORD
      Fin
    Fin
  Hasta word = ''
O bien
  Si modo diferente de 'D'
    Leer diccionario de arcppal con llave
    nombcamp y asignar el campo 2 a amc
    Si la lectura no fué satisfactoria
      Mens = Nombcamp concatenado con 'no
      pudo ser leido en dic de 'concatenado
      con arcppal
      Regresar
    Fin
    Leer arcppal con llave llaverc y asignar
    campo amc a old.campo
    Si lectura fué satisfactoria
      Si old.campo = contenido
        Rgeresar
      Fin
      All.data = old.campo
```

```
Modo = 'D'
Word = ''
Repetir
  Otro = 0
  Car = caracter 1 de all.data
  Si car parte de delim o = ''
    Si longitud de word >= length
      Si word diferente de ''
        Si modo diferente de 'D'
          Llamar UPDATE.RECORD
        O bien
          Llamar DELETE.REFERENCE
        Fin
        Llamar DELETE.REFERENCE
        Si modo = 'A'
          Llamar UPDATE.RECORD
        Fin
        Word = ''
        Otro = 1
      O bien
        Otro = 1
      Fin
    O bien
      Otro = 1
    Fin
  Fin
```

```
        Si otro = 1
          Caracter 1 de all.data = ''
        O bien
          Word = word concatenado con car
        Fin
      Hasta car = '' y word = ''
      Modo = 'A'
      Old.campo = ''
    Fin
    All.data = contenido
  O bien
    Word = ''
    Repetir
      Otro = 0
      Car = caracter 1 de all.data
      Si car parte de delim o = ''
        Si longitud de word >= length
          Si word diferente de ''
            Si modo diferente de 'D'
              Llamar UPDATE.RECORD
            O bien
              Llamar DELETE.REFERENCE
            Fin
            Llamar DELETE.REFERENCE
            Si modo = 'A'
              Llamar UPDATE.RECORD
```

```
                Fin
                Word = ''
                Otro = 1
              O bien
                Otro = 1
              Fin
            O bien
              Otro = 1
            Fin
          Fin
          Si otro = 1
            Caracter 1 de all.data = ''
          O bien
            Word = word concatenado con car
          Fin
        Hasta car = '' y word = ''
      Fin
    Fin
24.Regresar
```

MODULO ACTUALIZADOR DE ARCHIVO DE REFERENCIAS CRUZADAS

Función: Actualizar el archivo de referencias cruzadas con llaverc. Si se indica en la variable sort un orden especifico, el módulo es capáz de localizar la posición en el archivo del registro a actualizar de tal manera que el archivo continde ordenado.

De la misma manera que el módulo anterior y el siguiente, el módulo actual, es una copia del módulo original correspondiente del paquete y por lo tanto conserva algunos nombres de variables en su forma original. El resto han sido modificados para lograr congruencia con el módulo que lo llama.

UPDATE.RECORD

(3.5.2.1.1)

1. Leer archivo de referencias cruzadas con llave word y asignar a kw.rec
2. Field.stuff = kw.rec(up.acm)
3. Si sort diferente de ''
   Si delim.length > 1
   Localizar llaverc en field.stuff de

```
    acuerdo a orden indicado en sort usando
    VM
    Insertar llaverc en kw.rec en la posición
    localizada
    Escribir en arch de referencias cruzadas
    kw.rec con llave word
  O bien
    Localizar llaverc en field.stuff de
    acuerdo a orden indicado en sort usando
    VM
    Si la localización no fué satsifactoria
      Insertar llaverc en kw.rec en la
      posición disponible
      Escribir en arch de referencias
      cruzadas kw.rec con llave word
    Fin
  Fin
O bien
  Si delim.length > 1
    Localizar llaverc en field.stuff usando
    VM
    Insertar llaverc en kw.rec en la posición
    localizada
    Escribir en arch de referencias cruzadas
    kw.rec con llave word
  O bien
```

```
        Localizar llaverc en field.stuff usando
        VM
        Si la localización no fué satisfactoria
          Insertar llaverc en kw.rec en la
          posición disponible
          Escribir en arch de referencias
          cruzadas kw.rec con llave word
        Fin
      Fin
    Fin
4. Regresar
```

MODULO ELIMINADOR DE REFERENCIAS CRUZADAS

Función: Eliminar registro con llave word del archivo de referencias cruzadas.

DELETE.REFERENCE

(3.5.2.1.2)

1. Leer arch de referencias cruzadas con llave word y asignar a kw.rec
2. Si la lectura no fué satisfactoria
   Regresar
   Fin
3. Field.stuff = kw.rec(up.acm)
4. Localizar llaverc en filed.stuff usando VM
5. Si la localización no fué satisfactoria
   Regresar
   Fin
6. Eliminar de kw.rec el campo en la posición localizada
7. Si kw.rec(up.acm) = '' y del = 'Y'
   Temp = kw.rec
   Eliminar delimitadores AM concatenado con VM concatenado con SM de temp
   Si temp = ''
   Eliminar de arch de referencias cruzadas registro con llave word

O bien

Escribir en arch de referencias cruzadas

kw.rec con llave word

O bien

Escribir en arch referencias cruzadas

kw.rec con llave word

Fin

8. Regresar

MODULO MANEJADOR DE MENSAJES DE ERROR

Función: Enviar mensaje de error recibido de la rutina que lo llamó, en la parte baja de la pantalla, esperar el tiempo necesario para que el usuario pueda leer el mensaje y borrarlo.

ENVMSG(Mens)

La variable mens, contiene el mensaje de error que la rutina que invocó a éste módulo requiere que sea desplegado.

1. Posicionar el cursor en col 4 ren 22
2. Escribir en pantalla, caracter de alarma (esto es, que se produzca un sonido de alarma)
3. Escribir en pantalla el mensaje recibido
4. Esperar 3 segundos
5. Posicionar el cursor en col4 ren 22
6. Escribir en pantalla 50 espacios
7. Regresar

C A P I T U L O V

DESARROLLO DE EJEMPLO DE APLICACION.

V.1. CONSIDERACIONES PARA EL USO DE LA HERRAMIENTA DE CONSTRUCCION DE MODELOS DE SISTEMAS EN LINEA.

V.2. EJEMPLO DE APLICACION DEL SISTEMA DESARROLLADO (HERRAMIENTA) EN UN SISTEMA BANCARIO DE CHEQUES DE CAJA.

V.1. CONSIDERACIONES PARA EL USO DE LA HERRAMIENTA DE CONSTRUCCION DE MODELOS DE SISTEMAS EN LINEA.

Previamente a la presentación del ejemplo, se expone de una manera breve, la secuencia lógica de pasos que requiere el paquete Revelation para construir una aplicación y una lista de consideraciones que se deben tomar en cuenta para la correcta utilización de la herramienta:

- Definir los archivos que se usarán para la aplicación especifica y construir los diccionarios de datos propios de cada uno considerando atributos tales como justificación, patrones de edición, patrones de validación, etc.

- Definir la pantalla (programa para la terminología del paquete) o pantallas que usarán el o los archivos definidos en cuanto a nombre, tipo (entry o menu), etc.

- Seleccionar los campos de los archivos que aparecerán en una pantalla especifica.

- Generar la pantalla en base a los campos seleccionados en el paso anterior y efectuar las modificaciones en caso necesario.

- El paquete ofrece además, la facilidad de generar un programa fuente para cada pantalla definida escrito en lenguaje R/BASIC y su documentación correspondiente. Para los fines de la herramienta, no es necesario generarlos.

- Construir pantallas tipo menu para enlazar programas.

Cada uno de los pasos mencionados, es ejecutado por un módulo propio del paquete y por lo tanto resulta indispensable que el lector recurra a el manual USER'S GUIDE del paquete para obtener información más detallada dado que a partir de éste momento, se hara referencia a términos propios de Revelation.

Toda la información generada en los pasos expresados anteriormente, es almacenada en el archivo RDES propio del paquete.

Después que se ha establecido la secuencia de

operación del paquete, es necesario enunciar los atributos a los que la herramienta dará un tratamiento especial.

- El atributo de campo multivaluado solo se usa para validar que el campo en cuestión no sea multivaluado ya que ésta característica no es propia de las máquinas grandes.

- El atributo de profundidad de línea es ignorado.

- El atributo de formato se ignora y los valores aparecerán justificados a la izquierda.

- Del atributo de conversión de salida solo se usará el primer subcampo encontrado.

- El atributo 'prompt' se ignora.

- El atributo parte de la llave se ignora.

- No usar la opción %S% para valor por omisión (NULL DEFAULT) ya que es ignorada.

- Los patrones de validación especiales, deberán

estar entre ' o ".

Antes de presentar las consideraciones que se han mencionado al inicio de éste capítulo, es necesario explicar a detalle, el funcionamiento de la herramienta desarrollada, para asegurar su total entendimiento y por consiguiente un correcto uso de la misma.

El principal objetivo de la herramienta, es como se mencionó en la introducción y en el capítulo III, obtener las características de pantallas de captura en cuanto a campos, posición, longitud, patrones de edición y validación, etc. (definidas mediante los módulos del paquete Revelation), para presentarlas al usuario de tal manera que se logre en la computadora personal, un manejo de campos muy similar al manejo que hace de ellos una terminal de máquina grande. De manera complementaria, se efectúan sobre los campos, las validaciones que se hayan definido para cada uno de ellos y se ordenan según su posición en el archivo correspondiente. Como puede observarse, la salida del sistema es un registro validado y ordenado, por lo tanto, se requiere que el

usuario de la herramienta, escriba un programa en el lenguaje propio del paquete, es decir R/BASIC, para que sea el encargado de recibir el registro validado y ordenado para procesarlo. Este programa o rutina (RUTESP) puede ser tan complejo o sencillo como se requiera para el caso que se esté modelando y puede hacer uso de las rutinas de actualizacion de referencias cruzadas (también incluidas de manera complementaria en la herramienta) cuando se requiera tener un archivos de éste tipo. Un ejemplo de uso de este tipo de archivos se presenta cuando se requiere accesar un archivo de clientes cuya llave es " número de cliente", por "nombre del cliente", entonces se crea un archivo de referencias cruzadas que tiene como llave "nombre del cliente" y como campo único, el "número de cliente" que es la llave de acceso al archivo de clientes.

Consideraciones de uso:

- Del mismo modo que un programa escrito por el especialista puede hacer uso de los módulos de actualización de archivos de referencias cruzadas, es posible que se requiera hacer uso

de otros funciones del paquete Revelation tales como la funcion SELECT. Este tipo de funciones, envian mensajes durante su ejecución que pueden ocasionar alteración en las posiciones originales de los campos, por lo que resulta necesario efectuar un manipuleo del mensaje de la manera como se muestra a continuación en las lineas señaladas con asterisco (*).

```
0001 * PRINT @(0,21):
0002 * PERFORM 'SELECT A.AUT.DIS'
0003     READNEXT AUT ELSE CALL ENVMSG('NO FUE
         POSIBLE LEER SGTE AUT'); RETURN
0004     OPEN '','A.AUT.DIS' TO ARC.DIS ELSE
0005        CALL ENVMSG('NO FUE POSIBLE ABRIR
            ARCHIVO AUT.DIS')
0006        RETURN
0007     END
0008     WRITE '' ON ARC.US, AUT
0009     DELETE ARC.DIS, AUT
0010     @ANS = AUT
0011 *   PRINT @(0,23):SPACE(60)
0012     PERFORM 'CLEAR'
```

La instrucción de la línea 2, lee todos los registros del archivo A.AUT.DIS y envía un mensaje que dice "xx Record(s) Selected" o en español "xx Registro(s) Seleccionado(s)", según la versión de Revelation que se esté usando. Con la instrucción 1, se logra que éste mensaje aparezca en la línea 23 (2 abajo que la señalada en la instrucción), permanezca unos segundos y finalmente en la instrucción de la línea 11, se borra el mensaje, sin afectar al formato original de la pantalla.

- En algunos tipos de aplicaciones, existen algunos campos de entrada que no formarán parte del registro que se escribe en disco, por ejemplo, la respuesta S/N al campo DESEA CAPTURAR OTRO DOCUMENTO?, solo es utilizada para decidir si se envía nuevamente la misma pantalla de captura y no se requiere que éste campo se grabe. Este tipo de campos serán identificados para su manejo por la herramienta, entre dos signos "&", por lo que es necesario en el programa escrito por el especialista (RUTESP), analizar el registro completo para detectarlos. Estos campos se agregan a la pantalla correspondiente por medio del módulo SCR del

paquete y se insertan , sus atributos (únicamente posicion de columna y renglón, asegurándose que el atributo de número de campo quede igual a caracter nulo), en RDES, por medio del módulo EDIT, donde se encuentren dos caracteres 247 (separadores de atributos de campo) adyacentes, de la manera como se muestra a continuación: El campo identificado con el número 06, no forma parte del registro.

```
CH0845                    0845 <
                           CAPTURA DE CHEQUES EXPEDIDOS O
                              PAGADOS FUERA DE LINEA

    01  AUTORIZACION    ................

    02  IMPORTE         ..............


    03  POR CUENTA DE LA SUCURSAL     ...


04  BNF  ....................................

05  COM O CON
         ....................................

                          **  06  QUIERES CAPTURAR OTRO DOCUMENTO S/N

                                EDITING:                          MV(position):1
),E,I,D(EL),M,R(E-NBR),S(HIFT),F,L,^,G,W,TOP,END,2N,P ?

        ** Ejemplo de campo que no forma parte del registro (SCR)
```

```
193 W  <---- Caracter 247
194
195 71
196 19
197 L$1
198
199
200
201
202 OTRO.DOCTO
203 .
204 R
205 1A
206
207
208
209
210
211
212
213
214 BANDERA PARA INDICAR SI EXISTEN MAS DOCTOS. A CAPTURAR
215
216
.
217
218
219 W  <---- Caracter 247
220
221
222
223 L$25
224
225
226
227 0
228 CHANGE
229
230 0
*231 1N!2N!"FILE"!"DELETE"
.
```

Ejemplo de campo que no forma parte del registro

(EDIT)

- En cuanto a la definición de pantallas con el módulo SCR, se pueden usar de la columna 0 a la 79 inclusive y del renglón 0 al 21 inclusive, considerando los espacios que ocupan los limitadores de campo '[' y ']'.

- Cuando en la operación del paquete se teclea en el primer campo (que siempre es la llave de acceso al archivo asociado) la llave de un registro que ya existía previamente, se desplegará el contenido del registro y el cursor no podrá accesar el campo llave hasta que suceda una de las siguientes condiciones:

Que el cursor se posicione en el campo de ENVIE PAGINA sin haber efectuado modificaciones al registro y éste se transmita.

Que se modifique alguno o todos los campos no llave del registro, se transmita el registro y los campos pasen las validaciones correspondientes.

En los dos casos anteriores, se limpiarán todos los campos para una nueva captura.

- Al usar las rutinas de mantenimiento a archivos de referencias cruzadas, es necesario pasarles los siguientes parámetros en el orden

en que aparecen:

Modo. Tipo de operación a efectuar (eliminación 'D' o actualización 'U').

Llaverc. Llave para accesar el archivo de R.C.

Nombcamp. Nombre del campo sobre el que se tiene la R.C.

Programa. Nombre de la pantalla de la que viene el registro.

Contenido. Contenido del campo que se va a grabar en el archivo de R.C.

Mens. Parámetro para regresar a la rutina que hizo el llamado los mensajes de error en caso de que se presenten.

Finalmente, antes de pasar al ejemplo, se lista el proceso de creación de modelos que se debe seguir para ese fin y la programación de las teclas para la herramienta.

- Definir y crear archivos y pantallas con los módulos y procedimientos que indica el manual del paquete USER'S GUIDE.

- Escribir el programa asociado a la pantalla

(RUTESP) con las consideraciones mencionadas.

- Dar de alta el código objeto del programa asociado (RUTESP) en el archivo de programas objeto.

- Actualizar el catálogo de programas a emular de la manera como se muestra en el siguiente punto de éste capitulo.

PROGRAMACION DE TECLAS.

Avance del tabulador ..... Flecha hacia abajo.

Retroceso del tabulador .. Flecha hacia arriba.

Eliminar caracter de la izquierda reservando el espacio para que pueda ser usado por otro caracter ..... Flecha hacia la izquierda.

Avance de cursor en un caracter . Flecha hacia la derecha.

Eliminar caracter sin reservar espacio.. Del.

Transmitir registro ..... End.

Para terminar el uso de una pantalla se teclea 'FIN' en el campo llave y para terminar el uso de un menu se envia un nulo.

V.2. EJEMPLO DE APLICACION DEL SISTEMA DESARROLLADO (HERRAMIENTA) EN UN SISTEMA BANCARIO DE CHEQUES DE CAJA.

El modelo de sistema desarrollado para ejemplificar el uso de la herramienta construida, es una parte del proyecto de computarización de la red de sucursales de una institución bancaria, ésta parte corresponde al sistema de cheques de caja que a continuación se describe:

El sistema de cheques de caja es un instrumento que permite elaborar, transmitir y efectuar consultas de cheques de caja. Es conveniente hacer la aclaración de que solo se desarrolló una parte del modelo del sistema, esto es, las partes correspondientes a elaboración del cheque y transmisión de datos, por la razón de que es solo un ejemplo de uso de la herramienta y con las partes desarrolladas es suficiente para dejar clara la manera de usarla; además de que emplearía mucho espacio y tiempo el desarrollo total y no se encuentra entre los objetivos del trabajo; la consulta se efectuará por medio de las mismas pantallas de generación

de cheques y solo por número de autorización.

Después de haber efectuado un análisis y diseño del modelo, se llegó a la siguiente composición en lo que a pantallas se refiere:

Mendes

- CHQ800. Menú principal.

Función: Enlazar a los diferentes sistemas involucrados en el proyecto de computarización de sucursales.

Formato en pantalla.

'CHQ800'

MENU PRINCIPAL

1 ORDENES DE ABONO
2 ORDENES DE PAGO
3 GIROS
4 CHEQUES DE CAJA
5 CHEQUES CERTIFICADOS
6 FUNCIONES COMUNES

OPCION SELECCIONADA [ ]

ENVIE PAGINA[ ]

- CHQ804. Menú de Cheques de Caja.

Función: Enlazar a las diferentes pantallas que componen al sistema de Cheques de Caja.

Formato en pantalla.

'CHQ804'

MENU DE CHEQUES DE CAJA

1 EXPEDICION CH. CLIENTE
2 EXPEDICION CH. INTERNO
3 CORRECCION DATOS GENERALES
4 IMPRESION DE CHEQUE
5 CAPTURA OPER. TRAM. F/LINEA
6 CAMBIO DE DESTRUIDOA ENTREGADO

OPCION SELECCIONADA [ ]

ENVIE PAGINA[ ]

Pantallas

Para la descripción de las pantallas del modelo, se tomará la siguiente convención: Primero aparecerá la descripción de la función de la pantalla y enseguida la descripción de la función de la rutina asociada a esa pantalla, lo que en los capítulos anteriores y en el presente se ha denominado RUTESP. No hay que olvidar que en la función de cada una de las pantallas del modelo va implícita la función de validación de todos y cada uno de los campos que forman el registro cuando así sea definido. Por último aparecerá su formato tal y como aparece en pantalla.

- CHQ841. Pantalla de expedición de cheques de cliente.

Función: Aceptar nombre del beneficiario, importe del cheque, nombre del comprador y opcionalmente el número de autorización correspondiente al movimiento; leer de un archivo un número de operación para el movimiento.

Función de rutina asociada: Verificar que el número de autorización (cuando éste se haya tecleado) no haya sido usado por otro movimiento, asignar números de sucursal, terminal, cuenta de mayor afectada, fecha de contabilización, estado (impreso, no impreso, etc), forma de reembolso, fecha de impresión (todos ellos campos del archivo de cheques vigentes), actualizar archivos de referencias cruzadas para consultas por nombre del beneficiario y fecha de contabilización, actualizar archivo de cheques de caja vigentes con los datos mencionados anteriormente y finalmente imprimir el cheque correspondiente.

Formato en pantalla.

```
'CH0841'

                    EXPEDICION CH. CLIENTE


02  BNF             [.............................]
03  IMPORTE         [NNNNNNNNNN.NN]
04  COM             [.............................]

HAY MAS CHEQUES S/N

01  AUTORIZACION    [..............]

05  NUM.OPERACION   [.....]

                                                    ENVIE PAGINAC ]
```

- CHQ842. Pantalla de expedición de cheques internos.

Función: Aceptar nombre del beneficiario, importe del cheque, nombre del comprador, cuenta de mayor a afectar, fecha de contabilización, bandera de impresión de cheque (si/no), opcionalmente el número de autorización y leer del archivo correspondiente, un número de operación para el movimiento.

Función de rutina asociada: Verificar que el campo 'SE IMPRIME CHEQUE S/N' tenga valor de 'S' o 'N', en caso de haber tecleado un número de autorización ya asignado, verificar que el resto de los campos coincidan con los datos almacenados previamente, asignar valores al resto de los campos (sucursal, terminal, etc.), actualizar archivos de referencias cruzadas, si la respuesta al campo 'SE IMPRIME CHEQUE S/N' es 'S', imprimir el cheque generado; actualizar el archivo de cheques vigentes.

Formato en pantalla.

```
'CH0842'

                    EXPEDICION CH. INTERNO


02  BNF             [.................................]
03  IMPORTE         [NNNNNNNNNN.NN]
04  COM             [.................................]
05  CTA.MAYOR       [....]                          06  FE.CONT [DD/MM/AA]


07  SE IMPRIME CHEQUE S/N  [.]


01  AUTORIZACION    [................]

08  NUM.OPERACION   [.....]



                                                    ENVIE PAGINA[ ]
```

- CHQ843. Pantalla para correción de datos generales.

Función: Aceptar número de autorización de un cheque de caja que se haya generado previamente, y su importe correspondiente, aceptar el nuevo nombre de beneficiario y/o comprador.

Función de la rutina asociada: Leer archivo de cheques vigentes para verificar la existencia del registro a modificar, verificar que el importe sea el mismo que el captado previamente, es decir, solo se permite modificar nombre del beneficiario y/o comprador o concepto, verificar que el cheque haya sido impreso y que la fecha de generación del cheque sea la misma que la fecha en que se trata de modificar; actualizar el archivo de cheques vigentes.

Formato en pantalla

```
'CH0843'

                    CORRECION DE DATOS GENERALES

    01  AUTORIZACION     [.................]

    02  IMPORTE      [NNNNNNNNNN.NN]

03  BNF [................................]


04  COM O CON
        [................................]


                                                ENVIE PAGINA[ ]
```

- CHQ844. Pantalla para impresión de cheques.

Función: Aceptar número de autorización de un cheque que haya sido generado previamente, leer el archivo de cheques vigentes para obtener su importe, aceptar el número o clave de supervisor válido para permitir la impresión del cheque.

Función de la rutina asociada: Buscar entre signos "&" el número de supervisor autorizado y leer archivo de números de supervisor para verificar su existencia, verificar la existencia previa del cheque que se desea imprimir, verificar que el importe del cheque sea el mismo que con el que fué generado, verificar que el cheque no este marcado como robado o extraviado, verificar que el cheque no haya sido impreso previamente, verificar que la fecha de contabilización sea igual a la fecha en que se trata de imprimir, simular la pantalla de cheque en proceso de impresión; asignar los valores propios a los campos de estado de impresión y estado y actualizar archivo de cheques vigentes.

Formato en pantalla

'CHQ844'

IMPRESION DE CHEQUE

01 AUTORIZACION [.....]

02 IMPORTE [NNNNNNNNNN.NN]

03 SUP.[...]

ENVIE PAGINA [ ]

-CH0845. Pantalla de captura de cheques expedidos o pagados fuera de linea.

Función: Aceptar número de autorización asignado a operación fuera de linea, importe del documento, en caso de que el documento pertenezca a otra sucursal, aceptar el número de sucursal originadora, si es de la misma sucursal no es necesario teclear este campo; aceptar el nombre del beneficiario y nombre del comprador o concepto y finalmente la respuesta "S" o "N" para continuar o terminar la captura de documentos.

Función de la rutina asociada: Verificar que el número de operación no haya sido asignado a otro cheque de caja, asignar los valores correspondientes a los campos de sucursal, terminal, etc. y actualizar archivo de cheques vigentes.

Formato en pantalla

'CH0845'

CAPTURA DE CHEQUES EXPEDIDOS O
PAGADOS FUERA DE LINEA

01 AUTORIZACION [.....]

02 IMPORTE [NNNNNNNNNN.NN]

03 POR CUENTA DE LA SUCURSAL [...]

04 BNF [.....]

05 COM O CON
[.....]

06 QUIERES CAPTURAR OTRO DOCUMENTO S/N [.]

ENVIE PAGINA[ ]

- CHQ846. Pantalla de cambio de estado, de destruido a entregado.

Función: Aceptar número de autorización e importe correspondiente de hasta cuatro documentos.

Función de la rutina asociada: Con el número de autorización, leer archivo de cheques vigentes para verificar que el documento al que se hace referencia tenga clave de estado igual a destruido y que el importe del mismo sea igual al tecleado; cambiar el valor de estado al correspondiente a entregado; actualizar archivo de cheques vigentes.

Formato en pantalla

```
CHQ846'

                CAMBIO DE DESTRUIDO A ENTREGADO

     AUTORIZACION                          IMPORTE

01  [................]

                                    02  [NNNNNNNNNN.NN]

                                                    ENVIE PAGINA[ ]
```

Para lograr el correcto funcionamiento de las pantallas y su interrelación, fue necesario crear los siguientes archivos asociados a dichas pantallas.

- A.CHQ.VIG. Archivo de cheques vigentes donde se guardan las características del cheque generado y tiene el siguiente diccionario de datos:

Llave: Autorización. Contiene el número de autorización asignado al documento generado. Este campo llave, puede ser omitido durante la captura, en cuyo caso, deberá ser leido de un archivo de autorizaciones.

Beneficiario. Contiene el nombre de la persona beneficiaria del cheque de caja.

Importe. Contiene el importe del cheque de caja.

Com. Contiene el nombre del comprador del cheque o del concepto del cheque.

Num.Operación. Contiene el número de operación asignada al cheque generado, del mismo modo que en el número de autorización, en caso de no ser tecleado en el tiempo de captura, será leido de un archivo de números de operación.

Sucursal. Contiene el número de sucursal que generó el cheque de caja. Este campo, no aparece en ninguna de las pantallas del sistema, es manejado internamente en las rutinas asociadas a las pantallas y para los fines del modelo, siempre le será asignado el valor de 001.

Terminal. Contiene el número de terminal que generó el cheque de caja, y del mismo modo que en el campo anterior, es manejado internamente en las rutinas asociadas a las pantallas, asignándosele siempre un valor de 01, no aparece en ninguna pantalla.

Cta.Mayor. Contiene el número de cuenta de mayor que afecta el cheque generado, solo para el caso de cheque interno, también es manejado internamente por la rutina asociada a la pantalla expedición de cheque interno.

Fe.cont. Contiene la fecha de contabilización del cheque de caja generado, el valor por omisión es la fecha de captura.

Edo. Contiene una clave para identificar el estado en que se encuentra el cheque, por ejemplo, 00 cheque pagado, 01 cheque extraviado, 02 cheque robado, etc., no aparece en ninguna pantalla.

For.reemb. Contiene una clave para identificar la forma de reembolso del cheque, no aparece en ninguna pantalla.

Fec.imp. Contiene la fecha de impresión del cheque generado, no aparece en ninguna pantalla.

Sta.imp. Contiene una clave para identificar el estado de impresión del cheque, por ejemplo, 1 impreso, 0 no impreso, no aparece en ninguna pantalla.

- A.CHQ.SUP. Archivo de números de supervisor mediante los cuales se permite la impresión de cheques, tiene el siguiente diccionario de datos:

Llave: Sup. Número de supervisor y único campo.

-A.CHQ.NOP.US. Archivo de números de operación usados por algún cheque generado.

Llave: Num.oper. Número de operación usado o ya asignado; único campo.

- A.CHQ.NOP.DIS. Archivo de números de operación disponibles para ser asignados a algún cheque por generar.

Llave: Num.oper. Número de operación disponible o no asignado; único campo.

- A.CHQ.AUT.DIS. Archivo de números de autorización disponibles para ser asignados a un cheque por generar.

Llave: Autorización. Número de autorización disponible para ser asignado; único campo.

-A.CHQ.AUT.US. Archivo de números de autorización usados por algún cheque generado.

Llave: Autorización. Número de autorización usado o ya asignado; único campo.

Los archivos anteriores (A.CHQ.SUP a A.CHQ.AUT.US) fueron llenados por medio del módulo 6 de Revelation.

- RCFEC. Archivo de referencias cruzadas para acceso al archivo de cheques vigentes por fecha de contabilización. Su directorio es manejado por el propio paquete (Revelation).

- RCBNF. Archivo de referencias cruzadas para acceso al archivo de cheques vigentes por nombre del beneficiario. Su directorio es manejado por el propio paquete (Revelation).

A continuación se presenta la manera en que se definieron los campos del archivo A.CHQ.VIG, la manera como se definió la pantalla CHQ842 y el menú del sistema de cheques de caja, así como también, la rutina asociada a ésta pantalla para que el lector pueda ver un ejemplo de estas definiciones, recuérdese que estas definiciones se efectúan exactamente de la manera como se indica en los manuales de Revelation. La razón de exponer la definición de solo un archivo y una pantalla, obedece a que éstas definiciones se efectúan de manera muy semejante, variando únicamente los patrones de validación y posición de los campos. El archivo A.CHQ.VIG usa la mayoría de las opciones que se pueden declarar por lo que resulta un buen ejemplo, lo mismo que la pantalla CHQ842.

En Revelation existe un menú que permite elegir el módulo que se usará para construir una aplicación, normalmente los módulos son usados en el orden en que aparecen en el menú, a menos que se usen para efectuar modificaciones a una aplicación ya existente. En la siguiente página aparece el menú de referencia.

```
                        R/DESIGN  MAIN MENU
                       23:14:33  02 APR 1988

   NEXT ->     (1)  DCF         Define and Create a File
               (2)  BUD         Build a Dictionary for a File

               (3)  PGMR        Set up a New Program
               (4)  SEL         Select Fields to be Displayed
               (5)  SCR.GEN     Generate a Standard Screen
               (6)  SCR         Customize an Existing Entry Screen

               (7)  ENTER       Enter Data Using a Program

               (8)  DOC         Produce Application Documentation
               (9)  GEN         Generate RBASIC Code
               (10) BLD.MENU    Build Menu
               (0)  TCL         Return to REVELATION Command Level


                  Press ENTER key for Selection Number 1 or
                  Enter your Selection Number  ?
```

A partir de éste momento, aparecerán las pantallas mediante las cuales se definen las características necesarias de una aplicación. En la extrema izquierda del formato aparecerá el nombre del módulo seleccionado mediante el menú principal de la página anterior. Para el caso del módulo BUD (constructor de diccionario de datos, aparecerá una página para cada campo del archivo. El módulo 5 "SCR.GEN" (generador de pantallas) no tiene un formato dado que su función es generar los caracteres de control necesarios para la correcta posición de las leyendas definidas. El módulo 6 "SCR" tiene dos componentes, el primero permite la edición de las leyendas y campos en cuanto a posición, el segundo permite adicionar características a los campos que aparecerán en la pantalla que se esta definiendo, por lo tanto se muestra la pantalla de edición una vez, donde es posible observar las posiciones de los campos y leyendas y una pantalla de adición de características por cada campo de la pantalla en definición. Finalmente se muestra la pantalla del módulo 10 "BLD.MENU" (constructor de mendes) con las características del menu del sistema de cheques de caja. El resto de los módulos no son

mostrados dado que no son necesarios para el correcto funcionamiento de la herramienta de construcción de modelos.

```
DEF                    BUILD FILE DEFINITION ITEMS

01  FN          A.CHQ.VIG
02  Q.ACCT
03  OWNER       PROTOTIPOS                          <---------- (Comentario)
04  DESC        ARCHIVO DE CHEQUES DE CAJA VIGENTES <---------- (Comentario)
```

```
CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    FORM        <----- (No forma parte del reg. sirve para
03  SM          S                   leer del archivo de autorizaciones
04  TYPE        S                   dsiponibles un no. de autorizacion)
05  FIELD.NO            06  PART  0
07  CONV


08  FORMULA     01> PRINT @(0,21):
                02> PERFORM 'SELECT A.CHQ.AUT.DIS'
                03> READNEXT AUT ELSE CALL MSG.ERR('@NO FUE POSIBLE LEER SGTE AU
09  JUST         L                              10  LENGTH  16
11  PATRN       01> 3N'/'7N'/'4N
12  PROMPT       FORM
13  DISPLAY      FORM
14  SOURCE       ARCHIVO A.CHQ.AUT.DIS
15  DESC         CAMPO PARA LEER UNA AUTORIZACION DE SU ARCHIVO CORRESPONDIENTE


CHANGE T
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    OPER            <----- (No forma parte del reg. sirve para
03  SM          S                       leer del archivo de num. de operacio-
04  TYPE        S                       nes disponibles un no. de operacion)
05  FIELD.NO              06  PART  0
07  CONV


08  FORMULA     01> PRINT @(0,21):
                02> PERFORM 'SELECT A.CHQ.NOP.DIS'
                03> READNEXT NOP ELSE CALL MSG.ERR('@NO FUE POSIBLE LEER SGTE NO
09  JUST         L                                  10  LENGTH  5
11  PATRN       01> 5N
12  PROMPT       FORM1
13  DISPLAY      FORM1
14  SOURCE       ARCHIVO DE NUMERO DE OPERACIONES
15  DESC         CAMPO SIMBOLICO PARA ASIGNAR UN NUMERO A CADA OPERACION DE
                 CHEQUES DE CAJA


CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    BNF
03  SM          S
04  TYPE        F
05  FIELD.NO    1        06   PART   0
07  CONV


08  FORMULA


09  JUST        L                                  10   LENGTH   33
11  PATRN       01> 0Z
12  PROMPT      BENEFICIARIO
13  DISPLAY     BENEFICIARIO (BNF)
14  SOURCE      CLIENTE
15  DESC        NOMBRE DEL BENEFICIARIO DEL CHEQUE DE CAJA


CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    IMPORTE
03  SM          S
04  TYPE        F
05  FIELD.NO    2          06  PART  0
07  CONV       01> MD2,


08  FORMULA


09  JUST        L                                10  LENGTH  13
11  PATRN      01> (MD2)
12  PROMPT      IMPORTE
13  DISPLAY     IMPORTE
14  SOURCE      CLIENTE
15  DESC        IMPORTE POR EL QUE EL CHEQUE DE CAJA SERA EXPEDIDO


CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    COM
03  SM          S
04  TYPE        F
05  FIELD.NO    3        06  PART  0
07  CONV


08  FORMULA


09  JUST        L                              10  LENGTH  33
11  PATRN       01> OZ
12  PROMPT      COMPRADOR
13  DISPLAY     COMPRADOR
14  SOURCE      CLIENTE
15  DESC        NOMBRE DE LA PERSONA QUE COMPRA EL CHEQUE DE CAJA


CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    NUM.OPERACION
03  SM          S
04  TYPE        F
05  FIELD.NO    4        06  PART  0
07  CONV


08  FORMULA


09  JUST        L                              10  LENGTH  5
11  PATRN       01> 5N
12  PROMPT      NUMERO DE OPERACION
13  DISPLAY     AUT-CENT
14  SOURCE      NUMERO CONSECUTIVO DE OPERACION
15  DESC        NUMERO CON EL QUE SE PUEDE IDENTIFICAR LA OPERACION DE UN CHEQUE
                DE CAJA


CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    SUCURSAL
03  SM          S
04  TYPE        F
05  FIELD.NO    5         06  PART  0
07  CONV


08  FORMULA


09  JUST        L                              10  LENGTH  3
11  PATRN       01> 3N
12  PROMPT      SUCURSAL
13  DISPLAY     SUC:
14  SOURCE      SUCURSAL QUE OPERA EL MOVIMIENTO
15  DESC        NUMERO DE LA SUCURSAL QUE OPERA EL CHEQUE DE CAJA


CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    TERMINAL
03  SM          S
04  TYPE        F
05  FIELD.NO    6        06  PART  0
07  CONV


08  FORMULA


09  JUST        L                              10  LENGTH  2
11  PATRN       01> 2N
12  PROMPT      TERMINAL
13  DISPLAY     TERM
14  SOURCE      SUCURSAL QUE OPERA EL MOVIMIENTO
15  DESC        NUMERO DE LA TERMINAL QUE OPERA EL MOVIMIENTO

.

CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    CTA.MAYOR
03  SM          S
04  TYPE        F
05  FIELD.NO    7        06  PART  0
07  CONV


08  FORMULA
                    &

09  JUST        L                              10  LENGTH  4
11  PATRN     01> 4N                   02> (1000,9999)
12  PROMPT      CTA.MAYOR
13  DISPLAY     CTA.MAYOR
14  SOURCE      OPERADOR DEL CHEQUE DE CAJA
15  DESC        NUMERO DE CUENTA DE MAYOR QUE AFECTARA EL CHEQUE DE CAJA  EN
                CUESTION


CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    FE.CONT
03  SM          S
04  TYPE        F
05  FIELD.NO    8        06  PART  0
07  CONV        01> D2-E


08  FORMULA


09  JUST        L                              10  LENGTH  8
11  PATRN       01> (D)
12  PROMPT      FECHA CONTABLE
13  DISPLAY     FEC. CONTABILIZ
14  SOURCE .    OPERADOR DEL CHEQUE DE CAJA
15  DESC        FECAH DE CONTABILIZACION DEL CHEQUE DE CAJA


CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    EDO
03  SM          S
04  TYPE        F
05  FIELD.NO    9        06  PART  0
07  CONV


08  FORMULA


09  JUST        L                          10  LENGTH  2
11  PATRN       01> (0,8)
12  PROMPT      ESTADO
13  DISPLAY     EDO
14  SOURCE      OPERADOR DEL CHEQUE DE CAJA
15  DESC        ESTADO EN QUE SE ENCUENTRA EL CHEQUE DE CAJA (REVOCADO, PAGADO,
                CANCELADO, ETC)

CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    FOR.REEMB
03  SM          S
04  TYPE        F
05  FIELD.NO    10       06  PART  0
07  CONV

08  FORMULA

09  JUST        L                              10  LENGTH  1
11  PATRN      01> 1N
12  PROMPT      FOR.REEMB
13  DISPLAY     FOR.REEMB
14  SOURCE      OPERADOR DEL CHEQUE DE CAJA
15  DESC        FORMA DE REEMBOLSO DEL CHEQUE DE CAJA (EFECTIVO, DEPOSITO EN
                CUENTA, ETC)

CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN .          A.CHQ.VIG
02  FIELD.NM      FEC.IMP
03  SM            S
04  TYPE          F
05  FIELD.NO      11        06   PART   0
07  CONV          01> D2-E


08  FORMULA


09  JUST          L                                10   LENGTH   8
11  PATRN         01> (D)
12  PROMPT        FEC.IMP
13  DISPLAY       FEC.IMP
14  SOURCE        RUTINAS DE IMPRESION
15  DESC          FECHA EN LA QUE EL CHEQUE DE CAJA FUE IMPRESO


CHANGE ?
```

```
BUD                BUILD DICTIONARY ITEMS

01  FN          A.CHQ.VIG
02  FIELD.NM    STA.IMP
03  SM          S
04  TYPE        F
05  FIELD.NO    12        06   PART   0
07  CONV


08  FORMULA


09  JUST        L                                  10   LENGTH   1
11  PATRN      01> (0,3)
12  PROMPT      STA.IMP
13  DISPLAY     STA.IMP
14  SOURCE      RUTINAS DE IMPRESION DE CHEQUE
15  DESC        INDICADOR DEL ESTADO DE IMPRESION


CHANGE ?
```

```
PGMR                        PROGRAM SET-UP

01   PN           CHQ842
02   TYPE         ENTRY
03   SHORT.NM     Q842
04   LANGUAGE     BASIC
05   MAJ FILE     A.CHQ.VIG
06   READ         01> A.CHQ.VIG
07   WRITE        01> A.CHQ.VIG
08   IN.SCREEN    0
09   DESC         PROGRAMA PARA ELABORAR CHEQUES INTERNOS



CHANGE ?
```

```
SEL          0842 (CHQ842)      FILE = A.CHQ.VIG                    PAGE 1
  .
  01> *O AUTORIZACION    14> FORM
  02> BNF                15> OPER
  03> IMPORTE
  04> COM
  05> NUM.OPERACION
  06> SUCURSAL
  07> TERMINAL
  08> CTA.MAYOR
  09> FE.CONT
  10> EDO
  11> FOR.REEMB
  12> FEC.IMP
  13> STA.IMP



FIELD YOU WANT TO APPEAR ON YOUR SCREEN (# 1) ?
ALWAYS specify record keys first (prompts preceded by '*')
```

```
CHQ842                    Q842 <

                          EXPEDICION CH. INTERNO

02  BNF             ..............................
03  IMPORTE         ............
04  COM             ..............................
05  CTA.MAYOR       ....                          06  FE.CONT  ........


07  SE IMPRIME CHEQUE S/N


01  AUTORIZACION    ................

08  NUM.OPERACION   .....

                                  EDITING:                          MV(position):1
A,E,I,D(EL),M,R(E-NBR),S(HIFT),F,L,^,G,W,TOP,END,2N,P ↑
```

```
SCR                  PROMPT EDITOR

01  PN               CHQB42              19  PART            0
02  FIELD.NM         AUTORIZACION        20  FIELD.NO        0
03  WIND.SZ                              21  USER1
04  DEPTH                                22  USER2
05  YROW             16                  23  MV.MASTER
06  XCOLM            20                  24  KEY.FLAG        KR
07  FORMAT           L$16                25  XREF.FLAG
08  OUT.CONV                             26  DISPLAY.XREF
09  WIND.NO
10  FORMULA
11  PROMPT           AUTORIZACION DE OPERACION
12  MASK             ..........      13  REQ.OPT  0
14  PATRN            01> 3N'/'7N'/'4N
15  NULL.DFLT        <FORM>          16 VFILE              17 NOT.VFILE
18  DESCRIPTION      CAMPO LLAVE PARA ARCHIVO DE CHEQUES DE CAJA VIGENTES


CHANGE ?
```

```
SCR                  PROMPT EDITOR

01  FN               CHQ842                19  PART              0
02  FIELD.NM         BNF                   20  FIELD.NO          1
03  WIND.SZ                                21  USER1
04  DEPTH                                  22  USER2
05  YROW             5                     23  MV.MASTER
06  XCOLM            20                    24  KEY.FLAG
07  FORMAT           L$33                  25  XREF.FLAG
08  OUT.CONV                               26  DISPLAY.XREF
09  WIND.NO
10  FORMULA
11  PROMPT           BENEFICIARIO
12  MASK             ..........      13  REQ.OPT  R
14  PATRN            01> OZ
15  NULL.DFLT                        16 VFILE              17 NOT.VFILE
18  DESCRIPTION      NOMBRE DEL BENEFICIARIO DEL CHEQUE DE CAJA


CHANGE ?
```

```
SCR                  PROMPT EDITOR

01  FN               CHQ842              19  PART            0
02  FIELD.NM         IMPORTE             20  FIELD.NO        2
03  WIND.SZ                              21  USER1
04  DEPTH                                22  USER2
05  YROW             6                   23  MV.MASTER
06  XCOLM            20                  24  KEY.FLAG
07  FORMAT           L$13                25  XREF.FLAG
08  OUT.CONV         MD2,                26  DISPLAY.XREF
09  WIND.NO
10  FORMULA
11  PROMPT           IMPORTE
12  MASK             NNNNNNNNNN      13  REQ.OPT  R
14  PATRN            01> (MD2)
15  NULL.DFLT                        16 VFILE          17 NOT.VFILE
18  DESCRIPTION      IMPORTE POR EL QUE EL CHEQUE DE CAJA SERA EXPEDIDO


CHANGE ?
```

```
SCR                 PROMPT EDITOR

01  PN              CHQB42              19  PART            0
02  FIELD.NM        COM                 20  FIELD.NO        3
03  WIND.SZ                             21  USER1
04  DEPTH                               22  USER2
05  YROW            7                   23  MV.MASTER
06  XCOLM           20                  24  KEY.FLAG
07  FORMAT          L$33                25  XREF.FLAG
08  OUT.CONV                            26  DISPLAY.XREF
09  WIND.NO
10  FORMULA
11  PROMPT          COMPRADOR
12  MASK            ..........      13  REQ.DPT   R
14  PATRN           01> 0Z
15  NULL.DFLT                       16 VFILE              17 NOT.VFILE
18  DESCRIPTION     NOMBRE DE LA PERSONA QUE COMPRA EL CHEQUE DE CAJA


CHANGE ?
```

```
SCR                    PROMPT EDITOR

01  PN                 CHQ842              19  PART            0
02  FIELD.NM           CTA.MAYOR           20  FIELD.NO        7
03  WIND.SZ                                21  USER1
04  DEPTH                                  22  USER2
05  YROW               8                   23  MV.MASTER
06  XCOLM              20                  24  KEY.FLAG
07  FORMAT             L#4                 25  XREF.FLAG
08  OUT.CONV                               26  DISPLAY.XREF
09  WIND.NO
10  FORMULA
11  PROMPT             CTA.MAYOR
12  MASK               ....            13  REQ.OPT  0
14  PATRN              01> 4N              02> (1000,9999)
15  NULL.DFLT                          16 VFILE              17 NOT.VFILE
18  DESCRIPTION        NUMERO DE CUENTA DE MAYOR QUE AFECTARA EL CHEQUE DE CAJA
                       EN CUESTION


CHANGE ?
```

```
SCR                    PROMPT EDITOR

01  PN                 CHQ842                19  PART              0
02  FIELD.NM           FE.CONT               20  FIELD.NO          8
03  WIND.SZ                                  21  USER1
04  DEPTH                                    22  USER2
05  YROW               8                     23  MV.MASTER
06  XCOLM              64                    24  KEY.FLAG
07  FORMAT             L$8                   25  XREF.FLAG         Y
08  OUT.CONV           D2-E                  26  DISPLAY.XREF
09  WIND.NO
10  FORMULA
11  PROMPT             FECHA CONTABLE
12  MASK               MM/DD/YY        13  REQ.OPT  0
14  PATRN              01> (D)
15  NULL.DFLT          %D%             16 VFILE              17 NOT.VFILE
18  DESCRIPTION        FECHA DE CONTABILIZACION DE CHEQUE DE CAJA

CHANGE ?
```

```
SCR                    PROMPT EDITOR

01  FN                 CHQ842                  19  PART            0
02  FIELD.NM           NUM.OPERACION           20  FIELD.NO        4
03  WIND.SZ                                    21  USER1
04  DEPTH                                      22  USER2
05  YROW               18                      23  MV.MASTER
06  XCOLM              20                      24  KEY.FLAG
07  FORMAT             L$5                     25  XREF.FLAG
08  OUT.CONV                                   26  DISPLAY.XREF
09  WIND.NO
10  FORMULA
11  PROMPT             NUMERO DE OPERACION
12  MASK               .....          13  REQ.OPT  FP
14  PATRN              01> 5N
15  NULL.DFLT          <OPER>         16 VFILE              17 NOT.VFILE
18  DESCRIPTION        NUMERO CON EL QUE SE PUEDE IDENTIFICAR LA OPERACION DE UN
                       CHEQUE DE CAJA


CHANGE ?
```

```
V5B                        MENU BUILDER

01  MENU.NM    CHQ804
02  TITLE      (27,3)MENU DE CHEQUES DE CAJA

                              M E N U    OPTIONS
      03
    PROCESS                                          COMMAND

01> (23,6)EXPEDICION CH. CLIENTE                     CHQ841
02> (23,7)EXPEDICION CH. INTERNO                     CHQ842
03> (23,8)CORRECION DATOS GENERALES                  CHQ843
04> (23,9)IMPRESION DE CHEQUE                        CHQ844
05> (23,10)CAPTURA OPER. TRAM. F/LINEA               CHQ845
06> (23,11)CAMBIO DE DESTRUIDO A ENTRE               CHQ846
07> (27,20)OPCION SELECCIONADA



CHANGE ?
```

RUTINA ASOCIADA A LA PANTALLA CHQ842.

```
001  SUBROUTINE CHQR842(REGOK,BERR,MERR,NAPL)
002  EQU AM TO CHAR(254)
003  BERR = 0 ; MERR = '' ; NAPL = ''
004  OPEN '','A.CHQ.VIG' TO ARC.VIG ELSE
005    MERR = 'NO FUE POSIBLE ABRIR EL ARCHIVO A.CHQ.VIG'
006    BERR = 1
007    RETURN
008  END
009  AUTORIZACION = REGOK<1>
010  REGI    =    REGOK[LEN(AUTORIZACION)+2,LEN(REGOK)-
     LEN(AUTORIZACION)+1]
011  INDIC = INDEX(REGI,'&',1)
012  IMP = REGI[INDIC+1,'F&']
013  IF IMP NE 'S' AND IMP NE 'N' THEN
014    BERR = 1
015    MERR = 'EL CAMPO 6 DEBE SER "S" O "N"'
016    RETURN
017  END
018  REGI = REGI[1,INDIC-1]
019  READ VAR.REG FROM ARC.VIG,AUTORIZACION ELSE VAR.REG = ''
020  IF VAR.REG NE '' THEN
021    IF VAR.REG NE REGI THEN
022      BERR = 1
```

```
023       MERR = 'PARA MODIFICAR DATOS USE CH0843'
024       RETURN
025     END
026   END
027   REGI = REGI : AM : '00'
028   REGI = REGI : AM : '00'
029   REGI<5> = '001' ; REGI<6> = '01'
030   CALL REFCRUZ('U',AUTORIZACION,'BNF','CH0842',REGI<1>,MENS)
031   IF MENS NE '' THEN BERR = 1;MERR = MENS;RETURN
032   CALL REFCRUZ('U',AUTORIZACION,'FE.CONT','CH0842',REGI<8>,MENS)
033   IF MENS NE '' THEN BERR = 1;MERR = MENS;RETURN
034   IF IMP = 'N' THEN
035     REGI<12> = '0'
036   END ELSE
037     PRINTER ON
038     PRINT
039     PRINT 'SIMULACION DE IMPRESION DE CHEQUE DE CAJA'
040     PRINT
041     PRINT 'Banco Nacional de Mexico'
042     PRINT;PRINT
043     PRINT '                    MEXICO,D.F. '!OCONV(DATE(),'D')
044     PRINT;PRINT;PRINT
045     PRINT CHAR(27):CHAR(91):CHAR(52):CHAR(119)
046     PRINT 'PAGUESE POR'
047     PRINT 'ESTE CHEQUE A: ':CHAR(27):CHAR(91):CHAR(49):CHAR(119):REGI<1>:
        SPACE(33-LEN(REGI<1>)):OCONV(REGI<2>,'MD2,$')
```

```
048    PRINT;PRINT;PRINT;PRINT;PRINT CHAR(27):CHAR(91):CHAR(52):CHAR(119)
049    PRINT 'ESTE DOCUMENTO PUEDE SER ENDOSADO A UNA INSTITUCION DE
       CREDITO PARA SU COBRO'
050    PRINT CHAR(27):CHAR(91):CHAR(49):CHAR(119)
051    REGI<12> = '1'
052    PRINTER OFF
053  END
054  REGI<11> = DATE()
055  WRITE REGI ON ARC.VIG,AUTORIZACION
056  RETURN
```

Hasta el momento, se han llevado a cabo los pasos de definición de archivos y pantallas y la escritura de la rutina asociada a la pantalla a emular. Al iniciar la captura del programa fuente de la rutina asociada, es posible definir en que archivo residirá tal programa mediante el comando de edición del paquete Revelation TEXT, por ejemplo, si se desea crear el programa fuente PROGRAMA en un archivo que se llame A.ARCHIVO, se teclea TEXT A.ARCHIVO PROGRAMA, al momento de salvar el archivo y su código objeto (que tendrá como nombre $PROGRAMA), éstos serán escritos en el archivo A.ARCHIVO.

El paso final para terminar la definición de un modelo, es dar de alta las pantallas definidas y sus rutinas asociadas al archivo que contiene el catálogo de programas a emular, por medio del módulo CTRLPROG de la herramienta que presenta el siguiente formato:

'CTRLPROG'

MANTENIMIENTO AL ARCHIVO DE APLICACIONES EN USO

01 PROGRAMA [.....]

02 CLAVE [.]

PROG.ASOC [.....]

ENVIE PAGINA[ ]

En el campo 01, se teclea el nombre de la pantalla (solo pantallas, no menúes) definida previamente, de la manera como se indicó en éste punto (V.2), en el campo 02, se teclea una X para el caso en que se desee dar de baja el modelo del catálogo de tal manera que no se podrá emular y en el campo 3 (PROG.ASOC) aparecerá el nombre de la rutina asociada que se

definió de la manera indicada en los párrafos anteriores. Para el caso de dar de baja un modelo, se pedirá confirmar la baja cuando se detecte una X en el campo creado para tal fin.

CONCLUSIONES

Por medio de la herramienta presentada en éste trabajo, es posible detectar y corregir los posibles errores de interpretación por alguna de las partes involucradas (desarrollo de sistemas y usuario) que se pudieran presentar durante las primeras fases del desarrollo de sistemas en línea orientados a trabajar en equipo grande, la base sobre la cuál está construida la herramienta, permite una alta interacción con el usuario de tal manera que es posible mediante etapas sucesivas de refinamiento, llegar a una solución del problema que tiene como característica principal el haber emergido de un acuerdo entre el área usuaria y el área de desarrollo de sistemas.

En base a lo mencionado en el párrafo que antecede, se puede considerar que la herramienta de construcción de modelos de sistemas en línea, objeto de éste trabajo, ayuda de una manera importante al análisis y diseño de sistemas dado que no es necesario esperar a tener algún producto del sistema que se esté analizando para poder empezar efectuar cambios en los casos que así se requiera, por otra parte, el modelo creado mediante la herramienta cumple con las

características requeridas para ser considerado un prototipo del sistema que se encontrará en producción.

Una de las características más importantes de la herramienta consiste en la facilidad y rapidéz con que puede ser construido un modelo, como una medida de lo anterior, baste decir que el sistema de cheques de caja utilizado como ejemplo de aplicación de la herramienta, tomó para su construcción aproximadamente 8 meses desde la fase de análisis hasta la fase de prueba de programas y la construcción del modelo para el mismo sistema tomó para su construcción únicamente 5 dias, después de los cuales fue posible hacer una presentación al usuario para afinar detalles.

Respecto de las limitaciones de la herramienta, podemos considerar por la naturaleza misma de la máquina en que corre, como la más importante, la poca relación que guarda el tiempo de respuesta del modelo con el tiempo de respuesta del sistema cuando éste se encuentre en producción.

BIBLIOGRAFIA

1) REVELATION USERS GUIDE.

DAVE OSTBY.

PAUL OSTBY.

MIKE NOURSE.

1982, 1983, 1984, 1985 By Cosmos Incorporated.

Printed by: Imperial Printing Co.
St. Joseph, MI.

2) REVELATION TECHNICAL REFERENCE.

DAVE OSTBY.

PAUL OSTBY.

MIKE NOURSE.

1982, 1983, 1984, 1985 By Cosmos Incorporated.

Printed by: Imperial Printing Co.
St. Joseph, MI.

3) DISK OPERATING SYSTEM VERSION 3.10 REFERENCE.

IBM CORPORATION.

BOCA RATON FLORIDA 1985.

4) STRUCTURED DESIGN:
Fundamentals of a Discipline of Computer Program and Systems Design.

Edwuard Yourdon
Larry L. Constantine. (1979).

Prentice Hall, Inc. Englewood Cliffs, New Jersey.

5) STRUCTURED SYSTEMS ANALYSIS:
Tools and Techniques.

Chris Gane and Trish Sarson.
Improved Systems Technologies, Inc. (1979).

Prentice Hall, Inc. Englewood Cliffs, New Jersey.

6) Apuntes de Programacion Estructurada.

Raymundo Hugo Rangel Gutierrez.

División de Ingenieria Mecánica y Eléctrica.
Facultad de Ingenieria de la U.N.A.M.
Departamento de Computación.
(1985).
FI/DIME/84-005.

7. Estructura de Datos.

Jorge I. Euán A.
Luis G. Cordero B.

Facultad de Ingenieria U.N.A.M.
Dirección General de Publicaciones U.N.A.M.

1984

8) Software Engineering

Design, Reliability and Management

Martin L. Shooman

International Student Edition

Mc. Graw Hill International Book Company 1985.