44

UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

Facultad de Ingeniería

DISEÑO Y DESARROLLO DEL SISTEMA INTEGRAL ADMINISTRATIVO Y DE CONTROL DE OPERACIONES

T E S I S

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACION

P R E S E N T A N :

ARMANDO HERNANDEZ DEL CASTILLO
JOSE ALFONSO MORA BELTRAN

Director de Tesis Ing. Adolfo Millan Nájera

México, D.F. 1994

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

A MIS PADRES:

SR. JOSE C. MORA ZAPATA.
SRA. FILIPINA BELTRAN GREEN.

Les dedico este trabajo por el gran esfuerzo y sacrificio que realizaron al darme los medios, su apoyo y comprensión, para que yo pudiera realizarme en la vida como una persona de bien y provecho. Gracias por mostrarme el camino de la vida y del saber.

A MIS HERMANOS:

Por su apoyo y comprensión que me brindaron a lo largo del camino.

JOSE ALFONSO MORA BELTRAN

A MI ESPOSA CHAYITO:

A ella que ha sido la principal razón de que yo pueda continuar por este largo y hermoso camino de la vida. Gracias te doy por tu amor, comprensión y consejo, que para mi han sido de mucho valor y la razón de mi existir.

JOSE ALFONSO MORA BELTRAN

A MIS PADRES :

PROF. ARMANDO HERNANDEZ MARQUEZ.
SRA. JUANA DEL CASTILLO CERON.

A ustedes que me dieron la vida, y que con sus consejos y ejemplo me formaron como persona. Mi agradecimiento profundo al esfuerzo y sacrificio para que pudiera tener una preparación profesional para enfrentarme a la vida.

A MI ESPOSA:

GUADALUPE REYES SOTELO.
Y A MIS HIJOS:
RAUL ARMANDO Y GERARDO.

Por la motivación para concluir este trabajo, y por toda la felicidad que me brindan en mi hogar, Les dedico este trabajo porque son la fuerza y la razón de mi vida

A MIS HERMANOS:

Por su apoyo y comprensión.

A TODA MI FAMILIA:

Por ser una familia tan fuertemente unida.

ARMANDO HERNANDEZ DEL CASTILLO

A LA LIC. ROSA MABEL CANDELARIA CERON.

AL ING. ENRIQUE ORTIZ FUENTES.

Con todo respeto, por su valioso apoyo y consejos durante mi vida de estudiante, que siempre recordaré como parte fundamental para poder concluir mis estudios.

A MIGUEL ANGEL MENDOZA PEREZ :

"No hay imposibles cuando realmente se anhela algo en la vida"

A LA SRA. MODESTA CORTES GOMEZ:

Como un humilde homenaje a su vida y obra.

ARMANDO HERNANDEZ DEL CASTILLO

A NUESTRO MAESTRO Y DIRECTOR DE TESIS:

ING. ADOLFO MILLAN NAJERA

Nuestro agradecimiento profundo a la confianza y orientación que nos brindó en nuestra carrera y para la realización de esta tesis.

AL C.P. FRANSCISCO JAVIER MORALES GOMEZ:
Director Administrativo de Qualli.

Con admiración y respeto, agradecemos el apoyo y motivación que tanto nos brindó para la realización de este trabajo.

ARMANDO HERNADEZ DEL CASTILLO
JOSE ALFONSO MORA BELTRAN

DISEÑO Y DESARROLLO DEL SISTEMA INTEGRAL ADMINISTRATIVO Y DE CONTROL DE OPERACIONES

CONTENIDO: *Página*

INTRODUCCION

En el presente trabajo se describe el análisis, diseño y desarrollo del "Sistema Integral Administrativo y de Control de Operaciones (SIACO)", elaborado para la empresa Promociones y Filmaciones para Televisión S.A. de C.V., cuyo nombre comercial es Qualli, que significa en Náhuatl: "Lo Bueno".

Qualli es una empresa líder en la producción y post-producción de video clips, documentales y comerciales para televisión. Qualli cuenta con la más alta tecnología en equipo de procesamiento de video y audio, dando servicio a las más importantes casas productoras y agencias de publicidad, nacionales e internacionales; ofreciendo la renta de salas de edición, cabinas de audio, foro de grabaciones, equipo adicional y personal técnico altamente calificado para la elaboración de productos de video de alta calidad.

Dada la importancia de la empresa en su ramo, era necesario tener un estricto control de la información generada en los distintos departamentos de la misma; por lo que el objetivo del presente trabajo es exponer la manera en que resolvimos dichas necesidades a través del diseño y desarrollo de un sistema de cómputo que automatiza los procesos administrativos y operativos de la empresa, facilitando su operación y proporcionando información oportuna y veraz para la toma de decisiones gerenciales.

Las operaciones de Qualli se basan principalmente en la renta de salas de post-producción, de producción y de audio. SIACO controla desde que el cliente inicia sus actividades con la empresa al solicitar una reservación de renta de equipo y/o salas que requiere en la realización de su proyecto, generando una orden de servicio para posteriormente facturar todos aquellos recursos que utilizó, dando inicio entonces al proceso administrativo, el cual incluye el manejo contable que de ello se deriva, además de controlar procesos adicionales propios a la operación interna de la empresa como: control de personal, tesorería, cobranzas, presupuestos, almacén, relaciones públicas, etc..

Pretendemos con esta tesis exponer de una forma sencilla, como se elaboró el sistema y para ello citamos la metodología empleada (combinación de paradigmas) y detallamos el sub-sistema de reservaciones de equipo que sirve para ejemplificar esta metodología.
Este trabajo esta estructurado de la siguiente manera:

En el capítulo 1 se describen los antecedentes y las necesidades que dieron origen al desarrollo del sistema en estudio.

En el capítulo 2 se abordan los principios de la Ingeniería del Software y su método tradicional o clásico con sus principales características, se analizan también los métodos de cuarta generación y de construcción de prototipos, y finalmente, como mediante una combinación de éstos, se llega a un método alterno que resulta muy útil y práctico.

En el capítulo 3 se mencionan las distintas herramientas que se desarrollaron para la generación de los prototipos que se utilizarón en el desarrollo del sistema SIACO, como son:

El modelo para capturar "ABC", los modelos para consultas, el menú de acceso al sistema, reportes y el manejo de estadísticas.

En el capítulo 4 se describe la evolución del sistema a través de las fases de definición, desarrollo y mantenimiento, así como de la aplicación de los prototipos en dichas fases.

Finalmente se concluye en el capítulo 5 lo práctico que resulta este método en el desarrollo de sistemas de este tipo, ya que el software desarrollado ha sido capaz de adaptarse a los cambios y actualizaciones continuas que se han presentado.

CAPITULO 1 ANTECEDENTES Y MARCO DE REFERENCIA

1.1 LA INFORMATICA EN LAS ORGANIZACIONES EMPRESARIALES

El desarrollo alcanzado por las organizaciones empresariales de México, demanda en la actualidad un incremento en el control de los requerimientos y volúmenes de información, así como de la necesidad de su manejo adecuado para la toma de decisiones cada vez más precisas y oportunas.

Los medios con que se contaba anteriormente eran insuficientes para cubrir las nuevas necesidades, no existía gran inversión de recursos, los tiempos de proceso eran muy prolongados y en algunos casos más complejos, debido a esto se empezaron a incluir dentro de las empresas públicas y privadas las funciones de la informática, una de estas necesidades es enfrentar estos problemas y relacionarlos, estudiando la forma adecuada de resolverlos.

El incremento en la demanda de la automatización es directamente proporcional al crecimiento de las empresas, es por esto que las áreas de desarrollo de sistemas han llegado a ser el cuello de botella para la obtención oportuna de la información.

La falta de una definición clara en los requerimientos del usuario y de una planeación adecuada en el desarrollo de los sistemas (normalmente se deben a la necesidad de reducir tiempos para obtener la información al menor plazo posible), provocan que existan sistemas de baja calidad, con reportes que no se usan, datos importantes que no se conservan o información innecesaria, así como procedimientos que se desvirtúan o documentación desactualizada. Además como punto muy importante, no se siguen teorías de programación establecidas por lo que el sistema obtenido es de baja calidad y muy

difícil de darle mantenimiento, cuando se requieren efectuar cambios o adiciones es necesario dedicarle mucho tiempo adicional porque no lo soportan infraestructuras teóricas y en la mayoría de los casos es preferible rehacer todo nuevamente.

Por lo anterior, los sistemas requieren de un mantenimiento extra constante, ocasionando un crecimiento desmedido en los mismos ya que es necesario aumentar el número de líneas de código para cubrir los requerimientos del usuario, haciendo éstos más complejos y menos eficientes.

Otro problema que se presenta es la falta de personal técnico especializado en el ramo. En muchos casos no existen líderes de proyectos, ni analistas de sistemas en los puestos adecuados y en el peor de los casos, también nos encontramos con personal que teniendo la preparación necesaria, por muchos factores no aplican la Ingeniería del Software adecuadamente.

En lo que se refiere a la documentación, frecuentemente es insuficiente, obsoleta, sin estándares, o en el peor de los casos no existe, originando que se requieran analizar más profundamente los sistemas para su mantenimiento y explotación, lo que casi siempre se realiza con retraso.

Otro factor importante es el gran tiempo invertido en resolver un problema o requerimiento de los usuarios sin usar técnicas apropiadas de programación que permitan por si solas soportar cambios o adiciones que se presenten.
Por lo anterior se puede apreciar que el analista o programador normalmente está más ocupado en la construcción y mantenimiento de los sistemas, que en actividades más creativas para optimizar y lograr resultados eficientes y productivos.

En la actualidad es imprescindible la informática dentro de todas las organizaciones. Nos ocuparemos principalmente de su aplicación en la organización empresarial. Esta área presenta ciertas características y necesidades que son de nuestro interés para el desarrollo del presente trabajo:

- Es necesaria la informática para el control administrativo, contable y operativo (producción).

- Se requieren de sistemas de software completamente integrales que sean considerados como un ambiente único, en donde la información fluya de una forma natural entre sus departamentos. El flujo de la información representa la manera en que los datos cambian conforme pasan a través del sistema, la entrada se transforma en datos intermedios y más adelante se transforma en la salida.

- La operación de los sistemas de software no sea tan compleja, ni se tengan que efectuar demasiados manejos para actualizar subsistemas con características comunes. Es decir, que conformen un ambiente de información único y uniforme; la comunicación y conexión entre ellos sea robusta y consistente. Para que las operaciones realizadas por los usuarios sean lo más simple posible y no tengan que efectuar operaciones adicionales en un mismo proceso.

-Que los sistemas de software tengan flexibilidad a cambios. No se deben afectar en demasía los subsistemas relacionados al realizar una modificación a alguno de ellos, es decir, los subsistemas deben tener una medida adecuada de modularidad y que no generen una interdependencia perjudicial entre los mismos para que se facilite el mantenimiento y/o la actualización.

- La actualización o adición de nuevos subsistemas se efectúa en forma consecuente o natural.

Trataremos pues en este estudio un caso de una empresa con necesidades inmediatas de informática y la manera como se solucionó a través del "Sistema Integral Administrativo y de Control de Operaciones (SIACO)".

1.2 ANTECEDENTES Y NECESIDADES QUE DIERON ORIGEN AL DESARROLLO DEL SISTEMA.

El caso a tratar es el de una empresa líder en el área de producción y post-producción de audio y video: "Qualli"; que nace bajo la perspectiva de ser un centro de servicios integrados que conjunte la tecnología de vanguardia y los mejores editores expertos del medio. Ofreciendo esto, se pretendía tener una empresa líder al mejor nivel de Latinoamérica y los Estados Unidos.

Con todo esto, y de una forma natural, se esperaba tener a la mayoría de los clientes del mercado nacional e internacional. Bajo la visión de sus directivos se puso mucho énfasis en tener tecnología de vanguardia y al personal más calificado, iniciando así sus actividades con el éxito esperado y más. Pero dentro de la organización se empezaron a detectar fallas en el manejo de la información en el área de sistemas, ya que esperando estar en el nivel más alto en tecnología se planteó el manejo de toda la información y procesos de la organización a través de un sistema de cómputo que no rindió los resultados esperados.

En primer lugar, se determinó que el desarrollo del sistema lo realizara una compañía externa dedicada al servicio y al desarrollo de sistemas de cómputo, la cual inició su trabajo de investigación antes de que Qualli iniciara sus actividades, esto repercutió posteriormente, ya que al paso del tiempo (muy corto tiempo) este sistema no fue capaz de soportar los cambios que se presentaron en el funcionamiento de la empresa volviéndose este software obsoleto, siendo muy costosas las continuas actualizaciones que se requirieron y muy lenta la respuesta de la compañía para enfrentar las nuevas necesidades.

Finalmente ésto creó conflictos entre la dirección general de Qualli y la citada compañía de software rompiéndose las relaciones e incrementando el problema de la empresa, ya que las operaciones debían de continuar y no se contaba con el apoyo de un sistema de cómputo adecuado que manejara los procesos administrativos y operativos, por lo tanto, toda la empresa estaba siendo afectada.

Estos problemas provocaron desconfianza en los directivos que estaban convencidos de que el sistema de cómputo en lugar de ayudar estaba entorpeciendo las operaciones de la empresa, por lo tanto era la última oportunidad que se daba al área de sistemas para demostrar que trabajando sólo con los recursos con que se contaba (no se quería invertir en vano) se deberían de solucionar estos problemas. Se requería entonces dar soluciones reales y en poco tiempo.

Este es el panorama general con el que nos encontramos en Qualli, y con el cual inicia nuestro caso de estudio, ya que nuestra intención es plantear un caso real y de cómo se solucionó, a través del desarrollo del sistema SIACO el cual tiene bases sólidas en la Ingeniería del Software con una variante al método clásico y que nos resultó muy útil y práctico.

Los principales propósitos de SIACO son: ser un sistema modular e integral, que sea flexible y soporte cambios futuros demandados por la dinámica de la empresa y su constante crecimiento, dar apoyo a los procesos administrativos derivados del control de la empresa; atender a las necesidades de servir mejor al cliente que viene a trabajar a Qualli, a través de un control del proceso operativo, desde que el cliente solicita un servicio mediante un reservación hasta que termina su trabajo y se le factura.

SIACO como un sistema modular. El sistema principal está construido a base de módulos menores o subsistemas que abarcan cada uno de los principales procesos que integran la administración y operación de la empresa, estos subsistemas operan de manera "independiente" ya que su mantenimiento afecta lo menos posible a otros, sin embargo existe cierto flujo de datos entre uno y otro, esto principalmente sienta las bases de la integración de los subsistemas en uno solo: SIACO.
La motivación que tuvimos en diseñar SIACO de manera modular es que la modularidad se ha convertido en un enfoque aceptado en todas las disciplinas de ingeniería. Un diseño modular reduce la complejidad, facilita los cambios (un aspecto crítico en el momento de darle mantenimiento al software) y dá como resultado una más fácil implementación, posibilitando el desarrollo paralelo de diferentes partes del sistema.

SIACO permite, dentro de una de sus principales características, obtener información estadística y reportes que dan una imagen clara de los procesos que aquí se manejan, como un apoyo a la toma de decisiones a nivel ejecutivo y gerencial.

SIACO como un sistema integral. Debido a que muchos de los procesos que realiza y para poder tener un estricto control de la información, es necesario que haya una "comunicación interna" entre dichos procesos, es decir, que los datos que se almacenen en la base de datos sean checados constántemente para asegurar que son lo más preciso

posibles y que poseen las características con las que fueron demandados, a su vez el sistema realiza múltiples intercambios de información con los subsistemas involucrados en el proceso. Con esto podemos ver que la principal característica de SIACO es la de conjuntar muchos módulos o subsistemas que se comunican entre sí.

SIACO como una base de datos. Los sistemas actuales utilizan bases de datos como una forma de tener la información agrupada y seleccionada con ciertas características que nos permiten obtenerla rápidamente. Además nos facilita incorporar o eliminar archivos y/o relaciones a la base de datos sin afectar al ambiente del sistema.

Podemos decir que la base de datos es una colección de datos interrelacionados en conjunto sin redundancias perjudiciales o innecesarias, su finalidad es la de servir a una aplicación o más, de la mejor manera posible; los datos se almacenan de modo que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir datos nuevos y para modificar o extraer los datos almacenados. Cuando hablamos de SIACO como un sistema de bases de datos es porque éstas son totalmente independientes desde el punto de vista estructural. Con todo lo anterior podemos asegurar un buen funcionamiento si tomamos en cuenta los siguientes puntos:

- *No redundancia (perjudicial)*
- *Independencia de los datos (lógica y físicamente de las aplicaciones)*
- *Interconectividad (ligas o vinculaciones entre las bases de datos)*
- *Protección y seguridad*
- *Accesabilidad en tiempo real (multiusuario)*
- *Versatilidad para la representación de las bases de datos.*
- *Interfase con el pasado y con el futuro (evolución del sistema a nuevas versiones)*
- *Migración de datos (a equipos diferentes y/o sistemas operativos)*
- *Simplicidad*

SIACO como una base de datos auxilar. Para la generación de estadisticas resulta poco conveniente el manejo de la información en linea (sobre todo en información histórica que no cambia) ya que el consumo de recursos en horas pico afecta drásticamente el rendimiento del sistema. Por lo que creamos una base de datos auxiliar donde el diseño de las relaciones no está tan estrictamente apegada a la teoria, esto es, se permite cierta redundancia controlada (generalmente fechas) y no está totalmente normalizada (generalmente campos totalizadores de otras relaciones). Con lo anterior conseguimos obtener las siguientes ventajas:

. Seguridad al programador y al usuario (caracteristicas multi-usuario).
. Velocidad, disminución de overhead, disminución de I/O, búsqueda inmediata.
. Flexibilidad, programación sencilla, no requiere búsquedas complejas.

SIACO como un sistema a base de prototipos. La constante necesidad de elaborar sistemas que realicen una tarea especifica, requiere que haya una comunicación estrecha con los usuarios que lo solicitan. Prototipos nos permite desarrollar sistemas con la vigilancia y supervisión de los mismos usuarios involucrados, esto nos permite tener una concepción más exacta de las necesidades de ellos y de la empresa. Prototipos nos facilita resolver proyectos de caracteristicas medianas y nos da flexibilidad para adaptarlos a cambios del medio o necesidades percibidas por los usuarios. Los productos realizados con prototipos proporcionalmente son menos funcionales y menos robustos, pero mayores en el caso de utilización y en el caso de aprendizaje, asi como de una respuesta rápida en su elaboración.

CAPITULO 2 FUNDAMENTO TEORICO

2.1 INTRODUCCION A LA INGENIERIA DEL SOFTWARE.

Antes de empezar se estudió la metodología más conveniente para diseñar un sistema de software de alto rendimiento (productivo, durable y de fácil operación) y que pudiera ser desarrollado en corto tiempo. Para ello recurrimos a la Ingeniería del Software.

La aplicación de los sistemas de software demandan diversas habilidades y técnicas para el reconocimiento de los problemas y su solución. El desarrollo del software para la solución de un problema, sin embargo, puede ser realizado con el uso de un conjunto de técnicas que son independientes de la aplicación. Estas técnicas forman la base de la metodología de la Ingeniería del Software.

La Ingeniería del Software está modelada con técnicas, métodos y controles realmente probados, asociados con el desarrollo del hardware. Aunque existan diferencias fundamentales entre el hardware y el software, los conceptos asociados con la definición, desarrollo y mantenimiento son similares para ambos elementos de los sistemas.

Los objetivos clave de la Ingeniería del Software son:

1) Una bien definida metodología que maneje las fases del ciclo de vida del software como son definición, desarrollo y mantenimiento.

2) Un conjunto de normas de los componentes del software que documenten cada paso del ciclo de vida y muestren los lineamientos seguidos de un paso a otro, y

3) Un conjunto de puntos predecibles que puedan ser probados a intervalos regulares a través del ciclo de vida del software.

La Ingeniería del Software abarca un conjunto de tres elementos que son: métodos, herramientas y procedimientos, que nos facilitan controlar el proceso de construir software de alta calidad de una forma productiva.

Los métodos suministran el "cómo" construir el software y abarcan un amplio espectro de tareas que incluyen: planificación y estimación de proyectos, análisis de los requerimientos del sistema y del software, diseño arquitectónico de la estructura de datos, codificación, pruebas y mantenimiento.

Las herramientas de la Ingeniería del Software suministran un soporte automático o semiautomático para los métodos.

Y finalmente los procedimientos son quienes unen a los métodos con las herramientas, definen la secuencia en la que se aplican los métodos.

La Ingeniería del Software está compuesta de pasos que abarcan los métodos, herramientas y procedimientos tratados anteriormente. Estos pasos se denominan paradigmas o modelos de la Ingeniería del Software. Un paradigma para la Ingeniería del Software se elige basándose en la naturaleza del proyecto y de la aplicación. A continuación se enuncian tres de los paradigmas más comunes: paradigma del ciclo de vida clásico, paradigma de construcción de prototipos y el paradigma de técnicas de cuarta generación. El método de cada paso puede variar de un paradigma a otro, pero el enfoque global que exige la definición, desarrollo y mantenimiento permanece invariable.

El proceso de desarrollo del software contiene tres fases genéricas, independientemente del paradigma de ingeniería elegido. Las tres fases, definición, desarrollo y mantenimiento, se encuentran en todos los desarrollos de software, independientemente del área de aplicación, tamaño del proyecto o complejidad.

La fase de la definición se enfoca sobre el "qué", esto es, identificar que información ha de ser procesada, que función y rendimiento se desea, que interfaces han de establecerse, que ligaduras de diseño existen y que criterios de validación se necesitan, independientemente del paradigma utilizado de alguna forma se producirán los siguientes tres pasos:

- Análisis de sistemas: Define el papel de cada elemento de un sistema informático.

- Planificación del proyecto de software: Una vez que está asignado finalmente el ámbito del software, se asignan los recursos, se asignan costos y se definen las tareas y planificación del trabajo.

- Análisis de Requerimientos: Antes de comenzar a trabajar es necesario disponer de una información más detallada del dominio de la información y de la función del software.

La fase de desarrollo se enfoca sobre el "cómo", esto es, cómo han de diseñarse las estructuras de datos y la arquitectura del software, cómo han de implementarse los detalles procedimentales, cómo ha de trasladarse el diseño a un lenguaje de programación, y cómo ha de realizarse la prueba. Los métodos aplicados durante esta fase variarán dependiendo del paradigma (o combinación de paradigmas) aplicado. Sin embargo, se producirán tres pasos concretos :

- Diseño del software : El diseño traslada los requerimientos del software a un conjunto de representaciones (algunas gráficas, otras tabulares o basadas en lenguajes) que describen la estructura de datos, la arquitectura y procedimiento algoritmico.

- Codificación: La representación del diseño debe de trasladarse a un lenguaje de programación convencional o a un lenguaje no procedimental, que dá como resultado unas instrucciones ejecutables por la computadora.

- Pruebas del Software: Una vez que el software se ha implementado en una forma ejecutable por la máquina debe ser probado para descubrir los defectos que puedan existir en la función, lógica e implementación.

La fase de mantenimiento se enfoca sobre el cambio que va asociado con una correcccción de errores, adaptaciones requeridas por la evolución del entorno y modificaciones debidas a los cambios de los requerimientos del usuario para reformar o aumentar el sistema. Se describen a continuación los tres tipos de cambios más comunes :

- Correción: Incluso en el software de alta calidad es probable que el usuario descubra defectos. El mantenimiento correctivo cambia el software para corregir los defectos.

- Adaptación: Con el paso del tiempo es probable que cambie el entorno original (por ejemplo, procesador, sistema operativo, periféricos, etc.) para el cual se desarrolló. El mantenimiento de adaptación se traduce en modificaciones del software para acomodarlo a los cambios de su entorno externo.

- Aumento: Conforme se utiliza el software el usuario identificará funciones adicionales que podria resultar beneficioso añadirlas. El manteniento perfectivo aumenta el software

más allá de sus requerimientos funcionales originales.

En las siguientes páginas enlistaremos las principales ventajas y desventajas de los paradigmas mencionados para poder evaluar el más conveniente para nuestro sistema SIACO en estudio, dadas sus caracteristicas.

2.2 CARACTERISTICAS DEL METODO TRADICIONAL PARA EL DESARROLLO DE SISTEMAS.

El paradigma del ciclo de vida clásico o tradicional, exige un enfoque sistemático y secuencial del desarrollo de software, que comienza en el nivel de sistema y progresa a través del análisis, diseño, codificación, prueba y mantenimiento.

- Análisis de los requerimientos: Este paso es muy importante ya que dependiendo de los resultados del análisis nos basaremos para elaborar el producto y si no tenemos una visión clara del sistema se podria tener un producto terminado distinto al deseado por el usuario, por esto, es necesaria una interacción continua con el usuario. En este paso se debe comprender el dominio de la información, así como la función, rendimiento e interfases requeridas.

- Diseño: se compone de tres pasos que son: estructura de datos, arquitectura del software y detalle procedimental, el proceso de diseño traduce los requerimientos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación.

- Codificación: El diseño debe traducirse en una forma legible para la máquina, esta tarea la ejecuta la codificación.

- Prueba: Al terminar la generación del código, comienza la prueba del programa que se enfoca sobre su lógica, para asegurar que la entrada definida producirá los resultados que realmente se esperan.

El ciclo de vida clásico es el más viejo y más ampliamente usado, sin embargo con el paso del tiempo se han presentado problemas algunas veces como lo son:

1) Los proyectos reales raramente siguen el flujo secuencial que propone el modelo, siempre ocurren iteraciones y se crean problemas en la aplicación del paradigma.

2) Normalmente es difícil para el usuario establecer explícitamente al principio todos los requerimientos, el ciclo de vida clásico requiere esto y tiene dificultades en acomodar posibles incertidumbres que pueden existir al comienzo de muchos proyectos.

3) El usuario debe tener paciencia, una versión funcionando del programa no estará disponible sino hasta las etapas finales del desarrollo del proyecto, un error importante no detectado hasta que el programa este funcionando puede ser desastroso.

2.3 CARACTERISTICAS DEL METODO DE CONSTRUCCION DE PROTOTIPOS.

Normalmente el usuario definirá un conjunto de objetivos generales, pero no identificará los requerimientos detallados de entrada, procesamiento y salida. Es difícil tratar aspectos específicos cuando hablamos con usuarios que no están muy seguros de lo que realmente quieren o necesitan pero sabrán que és tan pronto como lo vean. En otros casos el programador puede no estar seguro de la eficiencia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma en que debe realizarse la interacción hombre-máquina. Por lo que puede ser mejor método de Ingeniería del Software el realizar un prototipo.

La construcción del prototipo es un proceso que facilita al programador la creación de un modelo del software a construir, este modelo puede ser:

- Un prototipo en papel que describa la interacción hombre-máquina de forma que facilite al usuario la comprensión de cómo se producirá tal interacción.

- Un prototipo que funcione y que implementa algunos sub-conjuntos de la función requerida, software deseado, o un programa existente que ejecute parte o toda la función deseada, pero que tenga otras características que deban ser mejoradas en el nuevo trabajo de desarrollo (modelos de esto son el ABC, el modelo de consulta, generación de estadísticas, reportes, etc., tratados posteriormente).

La secuencia de sucesos para este paradigma, comienza con la recolección de los requerimientos; el técnico y el usuario se reúnen y definen los objetivos globales y perfilan las áreas en dónde será necesario una mayor definición.

Luego se produce un "diseño rápido" que se enfoca sobre la representación de los aspectos del software visibles al usuario, por ejemplo: métodos de entrada y formatos de salida, este diseño rápido conduce a la construcción de un prototipo el cual es evaluado por el usuario en forma interactiva, el prototipo sirve para identificar los requerimientos del software, pero puede ser problemático por las siguientes razones:

1) El usuario ve un prototipo funcionando y lo confunde con un producto terminado e ignora que por las prisas no se han considerado los aspectos de calidad y mantenimiento a largo plazo y que será reconstruido, el usuario exige que se le apliquen "cuantas mejoras" sean necesarias para hacer del prototipo un producto final que funcione. Demasiadas veces el desarrollador accede.

2.- El técnico realiza frecuentemente ciertos compromisos de implementación en orden a obtener un prototipo que funcione rápidamente. Puede utilizarse un sistema operativo o lenguaje inapropiado sólo porque estaba disponible, las cuales pueden formar parte integral del sistema.

En general este paradigma puede ser efectivo para la Ingeniería del Software, pero se debe aclarar que el prototipo que se construya debe servir sólo como un mecanismo de definición de los requerimientos, posteriormente ha de ser descartado (al menos parte de él) y debe construirse software real con los ojos puestos en la calidad y mantenimiento.

2.4 CARACTERISTICAS DEL METODO DE TECNICAS DE CUARTA GENERACION.

El término técnicas de cuarta generación, abarca un amplio espectro de herramientas de software, que tienen una cosa en común: todas facilitan al que desarrolla software especificar algunas características del software a alto nivel, luego, la herramienta genera automáticamente el código fuente basándose en la especificación del técnico.

El paradigma de técnicas de cuarta generación se orienta hacia la habilidad de especificar software a un nivel que sea más próximo al lenguaje natural o en una nota que proporcione funciones significativas.

Actualmente, un entorno para el desarrollo del software que soporte este paradigma, incluye algunas o todas de las siguientes herramientas : lenguajes no procedimentales para consulta a la base de datos, generación de informes, manipulación de datos, interacción y definición de pantallas y generación de código, capacidades gráficas de alto nivel y capacidad de hoja de cálculo, todas estas herramientas existen pero sólo para dominios de aplicación muy específicos.

Este paradigma comienza con el paso de recolección de requerimientos, idealmente el usuario debe describir los requerimientos y estos deben traducirse directamente en un prototipo operacional, pero este no funciona, el usuario puede no estar seguro de lo que necesita, puede ser ambiguo en la especificación de los hechos que son conocidos, y puede ser incapaz o no desear especificar la información en la forma que una herramienta de cuarta generación pueda requerirla, además las herramientas actuales no son lo suficientemente sofisticadas para acomodarse al lenguaje natural.

Para aplicaciones pequeñas puede ser posible, ir directamente desde el paso de establecimiento de los requerimientos a la implementación, usando un lenguaje de cuarta generación no procedimental, sin embargo es necesario un mayor esfuerzo para desarrollar una estrategia de diseño para el sistema. Incluso si se utiliza un lenguaje de cuarta generación el uso de técnicas de cuarta generación sin diseño, causará las mismas dificultades (poca calidad, pobre mantenimiento, mala aceptación por el usuario) que se encuentra usando los métodos convencionales.

La implementación usando un lenguaje de cuarta generación facilita al que desarrolla el software, la descripción de los resultados deseados, los cuales se traducen en código fuente para producir dichos resultados, debe existir una estructura de datos con información relevante y debe estar rápidamente accesible al lenguaje de cuarta generación. Resumiendo :

1) Con muy pocas excepciones, el dominio de aplicación actual de las técnicas de cuarta generación está limitado a las aplicaciones de sistemas de información comercial específicamente, el análisis de información y la obtención de informes en grandes bases de datos hasta la fecha se ha usado muy poco en productos de ingeniería y áreas de aplicación de sistemas.

2) La recolección de datos preliminares que acompaña al uso de esta técnica, parece indicar que el tiempo requerido para producir software, se reduce mucho para aplicaciones pequeñas y de tamaño medio, y que la cantidad de análisis y diseño para las aplicaciones pequeñas también se reduce.

3) Sin embargo el uso de estas técnicas para grandes trabajos de desarrollo de software, exige el mismo o más tiempo de análisis, diseño y prueba (actividades de Ingeniería del

Software) perdiendo así un tiempo sustancial que se ahorra mediante la eliminación de la codificación.

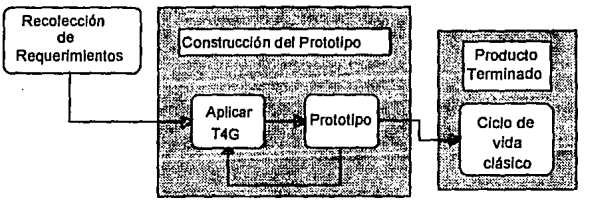
Para resumir, las técnicas de cuarta generación se convertirán probablemente en una parte importante del desarrollo de software durante el transcurso de la presente década, los métodos y paradigmas convencionales contribuirán cada vez menos al desarrollo de software y las técnicas de cuarta generación rellenarán el hueco que exista en el desarrollo de sistemas.

2.5 COMBINACION DEL METODO TRADICIONAL CON LOS METODOS DE CONSTRUCCION DE PROTOTIPOS Y TECNICAS DE CUARTA GENERACION.

El análisis debe ser conducido independientemente del paradigma de Ingeniería del Software aplicado. Sin embargo, la forma que ese análisis tomará puede variar. En algunos casos es posible aplicar los principios de análisis fundamental y derivar a una especificación en papel del software desde el cual puede desarrollarse un diseño. En otras situaciones, se va a una recolección de los requerimientos, se aplican los principios de análisis y se construye un modelo de software, llamado prototipo, según las apreciaciones del usuario y del que lo desarrolla. Finalmente, hay circunstancias que requieren la construcción de un prototipo al comienzo del análisis, puesto que el modelo es el único medio mediante el que los requerimientos pueden ser derivados efectivamente.

En la figura que a continuación presentamos se muestra como pueden combinarse los tres paradigmas mencionados anteriormente, el trabajo comienza con la recolección de requerimientos; posteriormente se construye el prototipo (auxiliándonos con herramientas de cuarta generación) aplicando sus criterios de construcción hasta que se llega a un

producto terminado, pasando a la fase de mantenimiento (correcciones, adaptaciones y aumentos) que es el paso final del paradigma del ciclo de vida clásico.



Combinación de paradigmas

La combinación de paradigmas nos permite dar soluciones más acordes a nuestras necesidades, ya que la aplicación del método clásico exige una mayor dedicación, planeación y tiempo de desarrollo del software; aunque es el camino ideal, no se presta para dar soluciones rápidas, Por otra parte, la aplicación directa del diseño e implementación de sistemas a base de técnicas de prototipos únicamente, no nos dá buenos resultados a largo plazo y los sistemas quedan muy vulnerables a la hora de dar mantenimiento, esta técnica es recomendable para sistemas pequeños. Las técnicas de cuarta generación en la actualidad por si solas no suministran los elementos suficientes para el desarrollo de sistemas. Por esta razón, si combinamos algunos aspectos escenciales de cada uno de los métodos anteriormente señalados, podemos garantizar un mejor diseño y desarrollo rápido, eficiente, durable y fácil de mantener.

2.6 MODELO PROPUESTO PARA EL DESARROLLO DEL SISTEMA.

En conclusión y con los planteamientos anteriores, las características de la empresa y la premura de tiempo, para resolver el problema se propone combinar los tres paradigmas para obtener un método más acorde a nuestras necesidades y solucionar los problemas que se nos presentan con rapidez, y así tener un sistema con calidad, fácil de mantener y en corto tiempo. Sin embargo, no aplicamos el método de técnicas de cuarta generación en un sentido estricto, ya que aquí desarrollamos algunos modelos de prototipos muy específicos a manera de plantilla que vamos a utilizar para el desarrollo posterior del sistema. Estos modelos los utilizamos en lugar de las herramientas de cuarta generación, ya que nos van a servir para generar el código de las partes esenciales del sistema. Dichas herramientas están programadas con lenguaje de cuarta generación y son: el ABC como modelo de captura, los modelos de consultas, de reportes y de estadísticas, que detallaremos en el siguiente capítulo.

Características del método propuesto:

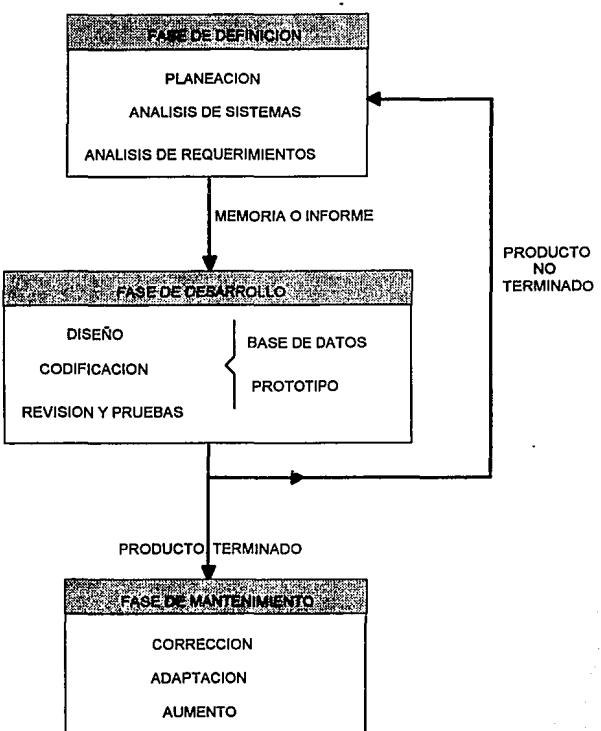
- *Se aprovechan las características del método clásico, construcción de prototipos y técnicas de cuarta generación.*
- *Los modelos propuestos sirven de herramientas reusables que nos van a permitir generar código más flexible y acorde a nuestras necesidades.*
- *Los sistemas generados de esta forma nos facilitan el mantenimiento a los mismos.*
- *Los prototipos generados por este método se convierten fácilmente en el sistema terminado, ya que en algunos casos no se requiere realizar ninguna modificación adicional.*
- *Facilita la programación del sistema, ya que es más fácil ensamblar que construir.*

Desventajas:

- *Un error no detectado oportunamente en alguno de los modelos utilizados para el diseño puede ser desastroso si se detecta en alguna fase tardía del diseño.*
- *La actualización del diseño de alguno de los modelos se puede volver problemática en la medida que éste se propague durante el crecimiento de todo el sistema.*
- *Los modelos propuestos únicamente se pueden utilizar en diseños similares en donde se requiera este tipo de operaciones. En aplicaciones muy específicas no se pueden usar, ya que no son adaptables a cualquier tipo de entorno .*
- *Los productos elaborados con esta técnica pueden en muchos casos quedar sujetos a la manera de ver y hacer las cosas de un usuario (originado por la propia técnica de prototipos y en ocaciones por lo pequeño del departamento para el cual se está desarrollando) y después al ya no estar dicho usuario tener la necesidad de adaptarlo a los nuevos requerimientos del usuario sustituto. Sin embargo, esto se puede sortear si el departamento en cuestión tiene bien definido su proceso y sus respectivos procedimientos de trabajo y de interacción con otras áreas de trabajo.*

Como mencionamos anteriormente, la Ingeniería del Software se compone en forma genérica de tres fases, las cuales son: Definición, Desarrollo y Mantenimiento. La metodología que nosotros proponemos está basada en estas fases. A continuación se describe de una manera gráfica la forma en que aplicamos dicha metodología :

MODELO PROPUESTO



Fase de Definición:

La fase de definición se compone de tres pasos que son: planificación, análisis de sistemas y análisis de requerimientos del proyecto de software.

Dentro de la fase de definición, la planificación del proyecto de software combina dos tareas: investigación y estimación. La investigación nos permite definir el alcance del elemento software de un sistema informático. La estimación nos permite definir los recursos requeridos para desarrollar el software.

El objetivo de la planificación es el de suministrar una área de trabajo que permita al director del proyecto hacer razonables estimaciones de recursos, costos y métodos. Estas estimaciones se hacen sin un marco limitado de tiempo al principio de un proyecto de software y deben de ser actualizadas regularmente al progresar el proyecto.

En la fase de definición, dadas las características y tamaño de la empresa, no requerimos de efectuar una planeación muy detallada (investigación y estimación) y sólo consideramos los siguientes puntos:

- Alcances del sistema: Se investiga si es necesario elaborar un sistema de software para solucionar las necesidades del usuario y si este sistema está dentro de la capacidad de cómputo instalada

- Requerimientos de equipo: dependiendo del tamaño de las operaciones que se efectuarán con el sistema y volúmenes de información a manejar se define la cantidad de equipo necesario. Normalmente un equipo personal por usuario más un servidor central que albergue al sistema con su base de datos, por lo que no se tiene un impacto importante de inversión.

- Requerimientos de personal: está dividido en tres partes, personal requerido para el desarrollo del sistema, personal que lo operará y la gente a la que irá dirigido nuestro sistema. Generalmente Directivos, Gerentes y/o Jefes de Departamento o aquellas personas que por las características de sus puestos necesitan del sistema, es importante evaluar esto, ya que normalmente se tiene que trabajar sólo con el personal de planta.

- Area de trabajo: para nuestro caso es necesario ubicar el área en donde se encontrarán las estaciones de trabajo, ya que estamos trabajando bajo un ambiente de red de área local y se requieren de lineas de comunicación hasta estas áreas, y en algunos casos equipo de respaldo de energía.

- Planeación de actividades y tiempos: Se establecen junto con el usuario las actividades y se les asigna el tiempo estimado para realizarlas y para ello nos apoyamos en el formato QI el cual nos permite controlar el avance del proyecto. En primer lugar se listan en desorden todas las actividades que se deben realizar, después en esta lista se les asigna un número consecutivo de acuerdo a su prioridad, con esto se reordena la lista en forma consecutiva y se llena el formato QI, en donde se sombrea la fecha estimada de realización de cada actividad y conforme se avanza se marca la actividad realizada; con esto se puede tener una visión objetiva de las actividades terminadas y los atrazos que se puedan tener, los cuales se pueden reprogramar al final de la lista reasignándoles nuevas fechas.

El siguiente paso de la fase de definición es el análisis de sistemas, en este paso se ubica al nuevo subsistema dentro del ambiente general de todo el sistema, recordemos que estamos tratando un sistema integral en dónde interactúan varios subsistemas, por lo que es necesario ubicar al nuevo subsistema y sus relaciones con los demás y el impacto que él mismo pueda tener dentro de todo el ambiente.

Finalmente tenemos el paso de análisis de requerimientos, en donde se tienen entrevistas más detalladas con el usuario, definiendo objetivos y alcances del nuevo software, dominios de la información, relaciones, etc. y elaborando un informe escrito lo más completo posible de lo anterior, el cual nos sirve como fuente de información para la siguiente fase.

Fase de desarrollo:

Una vez que se han establecido los requerimientos del software, se inicia la fase de desarrollo, la cual comprende tres pasos distintos: diseño, generación de código (manual o automáticamente) y prueba. Cada paso transforma la información de forma que finalmente se obtiene un software para computadora validado.

En la fase de Desarrollo el diseño es el primer paso de cualquier producto o sistema de ingeniería. Puede ser definido como: " ... el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física".

El objetivo del diseñador es producir un modelo o representación de una entidad que será construida más adelante. El proceso por el cual se desarrolla el modelo combina: intuición y criterios basándose en la experiencia de construir entidades similares, un conjunto de principios y/o heurísticas que guían la forma en la que se desarrolla el modelo, un conjunto de criterios que facilitan discernir sobre la calidad y un proceso de iteración que conduce finalmente a una representación del diseño final.

El diseño se divide en: diseño de datos, diseño arquitectónico y diseño procedimental. El diseño de datos se enfoca sobre la definición de estructura de datos. El diseño

arquitectónico define las relaciones entre los principales elementos estructurales del programa. El diseño procedimental transforma los elementos estructurales en una descripción procedimental del software.

El diseño de datos es la primera y de alguna forma podríamos decir la más importante de las tres actividades del diseño. El impacto de la estructura de datos sobre la estructura de programas y la complejidad procedimental, hace que el diseño de datos tenga una profunda influencia en la calidad del software.

En nuestro caso de estudio (sistema SIACO) en esta parte construimos una base de datos temporal aplicando los correspondientes criterios de normalización para cubrir la etapa del diseño de datos. Decimos base de datos temporal porque no es definitiva, ya que en el transcurso del análisis, el diseño puede sufrir modificaciones al aplicar las técnicas de prototipos al mismo diseño.

El objetivo principal del diseño arquitectónico es desarrollar una estructura modular del programa y representar las relaciones de control entre los módulos.

Para nosotros este paso del diseño está implícito al trabajar con una plantilla pre-elaborada (inicio del prototipo) porque es una arquitectura pre-definida y que en la mayoría de los casos nos evita tener que desarrollar este paso.

El diseño procedimental se realiza una vez que se han establecido la estructura del programa y de los datos. En un mundo ideal, la especificación procedimental requiere definir los detalles algorítmicos que deben establecerse en un lenguaje natural.

Para el método propuesto consideramos que este paso está implícito también al trabajar

con el prototipo y su respectiva codificación. Ya que las relaciones entre los módulos de un programa y sus respectivas transacciones hacia otros, quedan bien definidas con el lenguaje estructurado.

Resumiendo, en el paso del diseño lo más importante para nuestro método es definir la base de datos, aunque sea en forma temporal, para posteriormente aplicarle nuestro prototipo.

Todos los pasos de la Ingeniería del Software que se han presentado hasta ahora van dirigidos hacia un objetivo final: traducir las representaciones del software a una forma que pueda ser comprendida por la computadora. La codificación es un proceso que transforma el diseño en un lenguaje de programación.

Para el método propuesto esta etapa la dividimos en dos partes: la primera que es donde usamos una de nuestras principales herramientas que denominamos ABC (descrita en detalle posteriormente) como prototipo inicial de la codificación del sistema y que nos va ayudar para que nuestro usuario vea con más claridad como va quedando su sistema y en caso necesario realizamos un proceso iterativo a las fases anteriores para redefinir los planteamientos del sistema hasta que el usuario esté de acuerdo con lo que realmente desea, asegurando así que estamos trabajando sobre las necesidades reales para que el producto final sea lo que realmente se espera; esto marca una diferencia importante contra el método clásico puro, ya que en éste el usuario puede no haber definido correctamente sus requerimientos y no verá su producto sino hasta que esté completamente codificado, con los resultados obvios, un producto distinto al que esperaba. Posteriormente en la segunda parte de la codificación vamos añadiendo y/o revisando junto con el usuario el prototipo, las partes funcionales y sus respectivas validaciones que le irán dando forma al mismo, hasta llegar a un producto terminado.

La etapa de pruebas dentro del desarrollo de sistemas de software establece lo siguiente:

- La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

- Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

- Aquí la etapa de revisión y pruebas se realiza continuamente y de manera iterativa hacia las fases iniciales del modelo propuesto, ya que al aplicar la técnica de prototipos automáticamente nos damos cuenta de posibles errores de diseño y/o de codificación del proyecto. Con lo cual de manera consecuente se va actualizando al mismo hasta tener un producto terminado.

Fase de mantenimiento:

Los programas de computadora siempre están cambiando. Hay que solucionar errores, añadir mejoras y llevar a cabo optimizaciones. No sólo hay que cambiar la versión actual, si no también la versión del año pasado (que todavía está siendo usada) y la versión del año que viene (que ya casi funciona). Además de los problemas que requieren la realización de cambios para solucionarlos, el hecho mismo del cambio lleva a problemas adicionales.

Podemos describir el mantenimiento en tres actividades principales que se llevan a cabo al terminar un programa.

La primera actividad se da debido a que no es razonable asumir que la prueba del software

haya descubierto todos los errores. El proceso que incluye el diagnóstico y la corrección de uno o más errores se denomina mantenimiento correctivo.
La segunda actividad se da debido al rápido cambio inherente a todo aspecto de la informática, nuevas generaciones de hardware, nuevos sistemas operativos, y frecuentemente se mejoran o modifican los equipos periféricos. Por tanto, el mantenimiento adaptativo es una actividad que modifica el software para que interaccione adecuadamente con su entorno cambiante.

La tercera actividad se da cuando un paquete de software tiene éxito. A medida que se usa el software se reciben de los usuarios recomendaciones sobre nuevas posibilidades, sobre modificaciones de funciones ya existentes y sobre mejoras en general.
Para satisfacer estas peticiones, se lleva a cabo el mantenimiento perfectivo o aumentativo. Esta actividad contabiliza la mayor cantidad de esfuerzo gastado en el mantenimiento del software.

A continuación y para reforzar el tema de los prototipos ampliamente usado aquí, describiremos los pasos para su elaboración :

Algunos autores justifican la técnica de construcción de prototipos de esta forma:

" La mayoría de los métodos recomendados para definir los requerimientos de sistemas comerciales están diseñados para establecer un conjunto de requerimientos finales, completos, consistentes y correctos, antes de que el sistema sea diseñado, construido, visto o experimentado por el usuario. La experiencia industrial indica que a pesar del uso de técnicas rigurosas, en muchos casos los usuarios rechazarán las aplicaciones por no ser correctas o por ser incompletas cuando éstas se terminan. Consecuentemente, para armonizar la especificación original con la prueba definitiva de las necesidades

operacionales reales, es necesario un nuevo trabajo, que es caro, divisorio y que consume mucho tiempo. En el peor caso, en vez de aprovechar el sistema construido, éste es abandonado. Los desarrolladores pueden construir y probar de nuevo las especificaciones, pero los usuarios aceptan o rechazan de nuevo las realidades operacionales actuales. "

Aunque lo anterior representa una visión extrema, su argumento fundamental es robusto. En muchos casos (aunque no todos), la construcción de un prototipo, acoplado posiblemente con métodos de análisis sistemáticos, es una forma efectiva de Ingeniería del Software.

El paradigma de construcción de prototipos para la Ingeniería del Software se introdujo en el subíndice 2.3 de este capítulo. Ampliaremos este paradigma para indicar los pasos individuales de un proceso de construcción de prototipos y los puntos de decisión que dictan cómo se aplican estos pasos.

Como mencionamos anteriormente, todos los proyectos de Ingeniería del Software comienzan con una petición del usuario. La petición puede estar en la forma de una memoria que describe un problema, un informe que define un conjunto de objetivos comerciales o del producto, una petición de propuesta formal de una agencia o compañía exterior, o una especificación del sistema que ha asignado una función y comportamiento al software, como un elemento de un sistema mayor basado en computadora. Suponiendo que existe una petición para un programa de una de las formas dichas anteriormente, para construir un prototipo del software se aplican los siguientes pasos:

***PASO 1:** Evaluar la petición del software y determinar si el programa a desarrollar es un buen candidato para construir un prototipo. No todos los programas son adecuados para la construcción de prototipos. Varios factores de candidatura de construcción de*

prototipos pueden ser definidos: área de aplicación, complejidad de la aplicación, características del usuario y características del proyecto.

En general cualquier aplicación que cree una presentación visual dinámica, interactúe fuertemente con un usuario humano, o demande algoritmos o procesos combinatorios que deben ser desarrollados de una forma evaluatoria, son candidatos para la construcción de prototipos. Sin embargo, estas áreas de aplicación deben sopesarse con respecto a la complejidad de la aplicación. Si una aplicación candidato (una que tiene características descritas anteriormente) requiere el desarrollo de decenas de miles de líneas de código, antes de que pueda realizarse cualquier función demostrable, será normalmente, demasiado compleja para realizar un prototipo. Sin embargo, si la complejidad puede ser subdividida, puede ser aún posible realizar el prototipo de porciones del programa.

Debido a que el usuario debe interactuar con el prototipo en los últimos pasos, es esencial que: 1) el usuario participe en la evaluación y refinamiento del prototipo, y 2) el usuario sea capaz de tomar decisiones de requerimientos de una forma oportuna. Finalmente, la naturaleza del proyecto de desarrollo tendrá una fuerte influencia en la eficacia del prototipo. ¿Quiere y puede el desarrollador del proyecto trabajar con el método de la construcción de prototipos? ¿Están disponibles las herramientas de construcción de prototipos?, ¿Tienen los desarrolladores experiencias con los métodos de construcción de prototipos?

PASO 2: *Dado un proyecto candidato aceptable, el analista desarrolla una representación abreviada de los requerimientos. Antes de que pueda comenzar la construcción de un prototipo, el analista debe representar los dominios funcionales y de información del programa y desarrollar un método razonable de partición. La aplicación de estos principios de análisis fundamentales, pueden realizarse mediante los métodos de análisis de requerimientos.*

PASO 3: *Después de que se haya revisado la representación de los requerimientos, se crea un conjunto de especificaciones de diseño abreviadas para el prototipo. El diseño debe ocurrir antes de que comience la construcción del prototipo. Sin embargo, el diseño de un prototipo se enfoca normalmente hacia la arquitectura a nivel superior y a los aspectos de diseño de datos, en vez de hacia el diseño procedimental detallado.*

PASO 4: *El software del prototipo se crea, prueba y refina. Idealmente, los bloques de construcción de software preexistentes se utilizan para crear el prototipo de una forma rápida. Desafortunadamente, tales bloques construidos raramente existen. Alternativamente, pueden usarse herramientas de construcción de prototipos especializadas para asistir al analista/diseñador en la representación del diseño y traducirlo a una forma ejecutable. Incluso si la implementación de un prototipo que funcione es impracticable, el escenario de construcción de prototipos puede aún aplicarse. Para las aplicaciones interactivas con el hombre, es posible frecuentemente crear un prototipo en papel que describa la interacción hombre-máquina (preguntas, presentaciones, decisiones, etc.) usando una serie de hojas de historia: Cada hoja de historia contiene una representación de una imagen de la pantalla con texto, que describe la interacción entre la máquina y el usuario. El usuario revisa las hojas de historia, obteniendo una perspectiva de usuario de la operación del software.*

PASO 5: *Una vez que el prototipo ha sido probado, se presenta al usuario, el cual "conduce la prueba" de la aplicación y sugiere modificaciones. Este paso es el núcleo del método de construcción de prototipos. Es aquí donde el usuario puede examinar una representación implementada de los requerimientos del programa, sugerir modificaciones que harán al programa cumplir mejor las necesidades reales.*

PASO 6: *Los pasos 4 y 5 se repiten iterativamente hasta que todos los requerimientos estén formalizados o hasta que el prototipo haya evolucionado hacia un sistema de producción. El paradigma de construcción del prototipo puede ser conducido con uno o dos objetivos en mente: 1) el propósito del prototipo es establecer un conjunto de requerimientos formales que pueden luego ser traducidos en la producción de programas mediante el uso de métodos y técnicas de ingeniería de programación, o 2) el propósito de la construcción del prototipo es suministrar continuidad que pueda conducir al desarrollo evolutivo de la producción del software. Ambos métodos tienen sus méritos y ambos crean problemas.*

CAPITULO 3 HERRAMIENTAS REUTILIZABLES PARA LA GENERACION DE PROTOTIPOS

3.1 INTRODUCCION. Métodos y herramientas para la construcción de prototipos

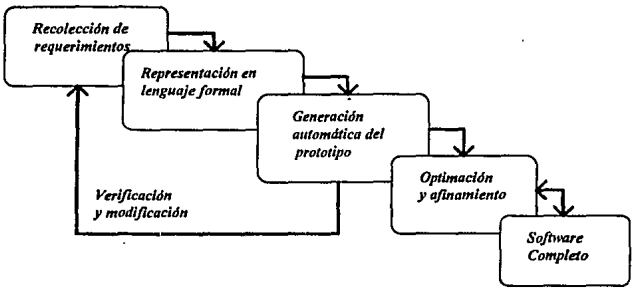
Para que la construcción de prototipos de software sea efectiva, un prototipo debe desarrollarse rápidamente, de forma que el usuario pueda comprobar los resultados y recomendar cambios. Para conseguir una construcción rápida de prototipos, existen tres clases genéricas de métodos y herramientas: técnicas de la cuarta generación, componentes de software reusables, especificación formal y entornos de construcción de prototipos.

*- **Técnicas de la cuarta generación.** Las técnicas de la cuarta generación comprenden un amplio repertorio de lenguajes de informes y preguntas a la base de datos, generadores de programas y aplicaciones y otros lenguajes no procedimentales de muy alto nivel. Debido a que estas técnicas facilitan al ingeniero de programación generar código ejecutable rápidamente, son ideales para la construcción rápida de prototipos. Desafortunadamente, el dominio de aplicación de estas técnicas está actualmente limitado a sistemas de información comerciales.*

*- **Componentes de software reusables.** Otro método para la construcción rápida de prototipos es ensamblar, en vez de construir el prototipo, usando un conjunto de componentes de software existente. Un componente de software puede ser una estructura de datos (o base de datos) o un componente arquitectónico de software (p.ej., un programa) o un componente procedimental (es decir, un módulo). En cada caso, el*

componente de software debe ser diseñado de forma que facilite el ser reusado sin conocer los detalles de su funcionamiento interno. Debe observarse que un producto de software existente puede ser usado como un prototipo para un producto nuevo, mejorado y competitivo. De alguna manera, esto es una forma de reusabilidad para la construcción de prototipos del software.

- ***Especificación formal y entornos para la construcción de prototipos.*** *En las pasadas dos décadas se han desarrollado varios lenguajes de especificación formal para reemplazar las técnicas de especificación en lenguaje natural. Hoy, los desarrolladores de estos lenguajes formales están en el proceso de desarrollar entornos interactivos que: 1) Facilite al analista crear interactivamente una especificación basada en el lenguaje de un sistema o software; 2) Llame a herramientas automáticas que traduzcan las especificaciones basadas en lenguaje en código ejecutable, y 3) Faciliten al usuario utilizar el código ejecutable del prototipo para refinar los requerimientos formales. Algunos lenguajes de especificación, están siendo acompañados de entornos interactivos en un esfuerzo para conseguir un paradigma de Ingeniería del Software automatizado (como se muestra en la siguiente figura). Aunque aún están en las primeras etapas de desarrollo y aplicación, tales entornos ofrecen una esperanza sustancial para mejorar la construcción de prototipos y la productividad en el desarrollo de software.*



Un paradigma de Ingeniería del Software automatizada.

En nuestro caso en estudio, el lenguaje de programación Dataflex proporciona algunas herramientas basadas en técnicas de cuarta generación como lo son: generador de programas y bases de datos (DFAUTO) y generador de reportes (DFQUERY), sin embargo estas herramientas generan productos todavía muy limitados e informales por lo que las utilizamos sólo en casos muy sencillos y para el segundo método (componentes de software reusable) no tenemos una librería de componentes de software totalmente reusable sino que desarrollamos algunos componentes arquitectónicos de software (programas) y algunos componentes procedimentales (modulos) que incorporamos en los programas que así lo requieran, es decir, la librería que desarrollamos comprende módulos que se pueden incorporar sin ninguna modificación en los programas y otros módulos pueden servir de plantilla para adaptarlos a una aplicación en particular. Esto es debido a las limitaciones de Dataflex para manejar librerías, principalmente en lo que se refiere al paso de parámetros.

A continuación describiremos nuestras herramientas reutilizables que empleamos para la generación de prototipos.

3.2 EL "ABC " COMO MODELO DE CAPTURA.

Dentro de la metodología para apoyar el desarrollo del sistema a través de prototipos, elaboramos algunas herramientas que nos sirven de plantillas o esqueletos para facilitarnos el desarrollo, una de estas herramientas es el "ABC" que es un modelo o prototipo para manejar la captura o introducción de información a la base de datos del sistema. Este modelo nos permite (junto con el usuario) definir las estructuras de datos que se capturarán en el sistema en desarrollo, además de conocer las operaciones y validaciones que se tendrán con la información.

Se manejan en este software las operaciones básicas que puede requerir un usuario al ingresar y actualizar sus datos, como lo son: Altas de nuevos registros, Bajas a registros ya existentes, Cambios y/o edición de registros ya existentes y Consulta de todos los registros en el archivo elegidos por selección del usuario.

Al terminar de armar un "ABC" el usuario puede visualizar los principales datos que ingresarán a su base de datos, esto ayudará de una manera más objetiva a aclarar si el sistema que desea obtener es como el prototipo que está viendo. Esto es muy importante porque como ya se describió anteriormente en la fase de análisis pueden surgir conflictos al no "entenderse" el usuario y el programador, se puede caer en el error de trabajar sobre un sistema como lo comprendió el desarrollador, pero que es distinto al que desea el usuario.

El "ABC" se maneja bajo un estándar de operación, para ayudar a los diferentes usuarios a aprender a utilizar sus sistemas más fácilmente.

Por ejemplo: el manejo de las pantalla de captura del departamento de Cobranza, en esencia es similar al manejo de las pantallas de captura del departamento de Tesoreria, aunque la información y procedimientos que ahí se operan sean distintos.

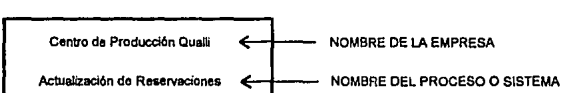
Para mayor facilidad en la comprensión del software, dividimos en dos partes su descripción; la vista que tiene el usuario y la vista que tiene el desarrollador.

Vista del Usuario:

Detallamos a continuación la vista externa que tiene el usuario de una pantalla de un "ABC". Básicamente esta vista se compone de dos secciones la del encabezado y la de la captura.

- Sección del encabezado : Esta sección aparece centrada en la parte superior y en ella se indica el nombre de la empresa seguida por la descripción del proceso.

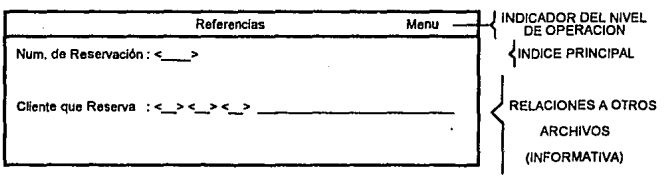
SECCION DEL ENCABEZADO



- Sección de captura: En esta sección se encuentra el cuerpo del "ABC" que está dividido en dos partes principales: referencias y datos generales.

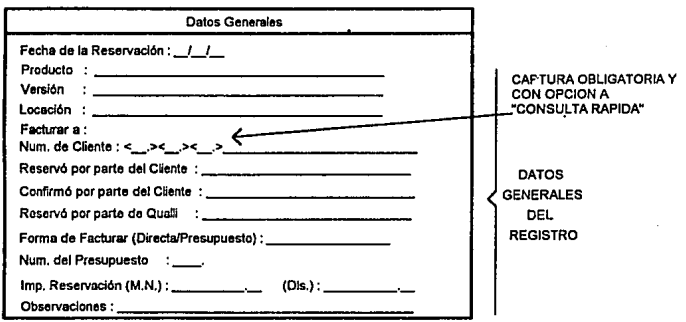
Referencias: En esta parte se capturan las llaves principales del archivo y se presentan algunas referencias a los archivos relacionados o relaciones. En la parte superior derecha se encuentra el indicador del nivel donde nos encontramos dentro del "ABC" (Menú, Altas, Bajas, Cambios y/o Consultas) como se muestra a continuación.

SECCION DE REFERENCIAS



Datos Generales: Esta parte es la sección de captura en general y en donde se realizan las validaciones necesarias de los datos que van a ingresar a los campos restantes del registro. Un campo encerrado entre llaves angulares "<>" indica que la captura es obligatoria y que la información que se suministre debe existir previamente en la base de datos, para ello se proporciona una consulta rápida a la base de datos mediante una tecla preprogramada (Tab) con la cual podemos elegir la clave del registro deseado sin necesidad de recurrir a fuentes externas.

SECCION DE DATOS GENERALES



- Sección de mensajes: Esta área nos permite visualizar mensajes de error o indicaciones especiales del proceso, además de la acción a seguir para poder continuar. Estos mensajes o indicaciones están clasificados de la siguiente manera:

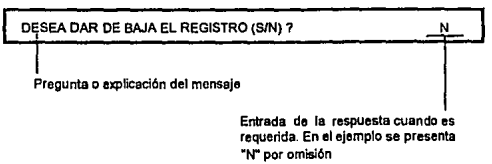
Mensajes de advertencia: Se presentan cuando para completar la operación que se pretende realizar es bajo criterio del usuario y requiere una observación especial.

Mensaje de error: Se presenta cuando el sistema realiza algún tipo de validación interna y la operación que se pretende realizar no procede.

Mensajes en general: Básicamente se emplean cuando algún proceso requiere de más tiempo del normal para realizar alguna operación y/o validación compleja, se indica con un mensaje de que se está efectuando el proceso.

Mensajes interactivos: Mensajes donde se espera respuesta del usuario para completar la operación. Existen procesos en los que es recomendable confirmar la operación antes de efectuarla, por ejemplo para dar de baja un registro se solicita la confirmación antes de borrarlo.

SECCION DE MENSAJES



Sección del Menú: Esta sección nos permite manipular las opciones básicas de la actualización de datos, como son Altas, Bajas, Cambios y Consultas, ya mencionadas. Existen varias formas de operar este menú:

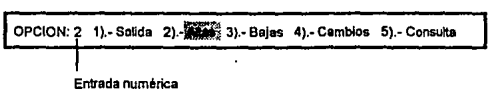
-Seleccionando la opción mediante el suministro del número indicado a la izquierda de la misma y presionando la tecla "Enter".

-Seleccionando la opción mediante el suministro del primer caracter o letra con el que empieza el nombre de la opción y posteriormente la tecla "Enter".

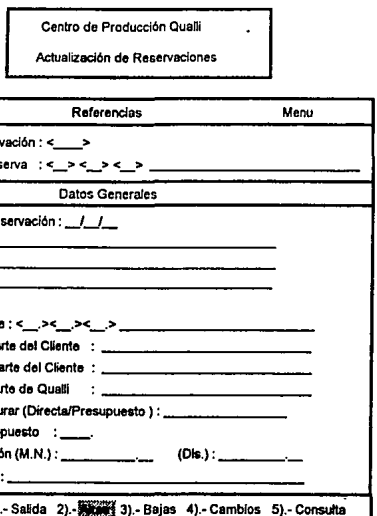
-Seleccionando la opción mediante la utilización de las teclas flecha izquierda y flecha derecha del teclado y posicionando el cursor en la opción deseada y presionando la tecla "Enter".

-Seleccionando la opción mediante la barra espaciadora y presionando la tecla "Enter".

SECCION DEL MENU



A continuación se muestra una vista completa de un "ABC" con todas sus secciones.



Vista del programador:

Trataremos ahora la vista interna del "ABC" que es con la que trabaja el desarrollador. De igual manera que la anterior y para facilitar su comprensión dividiremos el programa en secciones, para esto se recomienda seguir el listado del programa que se detalla en el capítulo 4.2.2.

- Sección del encabezado: Se tiene en primer término la creación de las pantallas que vé el usuario, la primer pantalla que se modifica es la del encabezado en donde se deberá de especificar la descripción del programa en cuestión.

-. Sección de captura: Al igual que la vista del usuario esta sección se divide en dos partes:

Referencias: Posteriormente en el cuerpo de la pantalla de captura se actualiza la sección de referencias, en donde se acomodan los campos que sean índices principales y también aquellos campos que nos muestren información adicional de otras relaciones y que puedan servir de referencia para el que captura.

Los campos entre llaves angulares "< >" indican que la captura de la información en estos campos, no se puede omitir y además de que tenemos la opción de realizar una consulta rápida presionando la tecla preprogramada para consultas (Tab).

Datos generales: En esta sección se pondrán los campos restantes del registro, inclusive índices secundarios del archivo o relación. Aquí el usuario podrá capturar toda la información que desea almacenar y manejar.

- Sección de declaración de archivos, variables locales y ventanas de la pantalla de captura: En esta sección se declaran todos los archivos que se utilizarán, así como las variables locales necesarias y se les asigna un nombre o identificador a todas las ventanas de las pantallas construidas.

- Sección de manejo del menú: Este es un módulo permanente, conformado por rutinas que no requieren ningún cambio ya que son rutinas para el manejo y control del menú para seleccionar la opción deseada (salida, altas, bajas, cambios y consultas), posteriormente este módulo cederá el control a la sección del "ABC" correspondiente.

- Sección de Altas: Si en el menú se seleccionó la opción de altas, se ejecuta esta sección. Esta rutina servirá para la introducción de registros nuevos al archivo, es necesario capturar primero el índice principal, acontinuación se asigna automáticamente el número consecutivo que formará parte de la llave del registro, para posteriormente permitir capturar los demás campos del nuevo registro, validándolo en caso de ser necesario.

Al final se actualizan las demás relaciones, si es que existen, y se salva el registro indicando el número consecutivo asignado (el sistema maneja estos números consecutivos mediante un archivo maestro) y se regresa el control al inicio de la sección de "altas" para iniciar una nueva captura.

Si la aplicación tiene más niveles de captura, se enlaza a un segundo programa "ABC" para capturar en el siguiente nivel de las relaciones, pasando como parámetro el índice principal para que se actualice el próximo nivel y así se establezca la relación con el primer nivel (padre a hijo). Y así sucesivamente en caso de existir más niveles de captura.

- Sección de Bajas: El objetivo de esta rutina es la de borrar o la de "marcar" un registro existente siempre y cuando no existan relaciones que se afecten y queden sin referencia (sin padre).
De igual manera se captura el índice principal para poder buscar y traer el registro a borrar, en caso de no existir, nuevamente se maneja la rutina de error y se manda el control al inicio de la sección de "bajas", en caso contrario (búsqueda verdadera), se presenta en la pantalla la información que contiene el registro a borrar para que el usuario visualice si el registro elegido es el correcto, a continuación se presenta un mensaje en donde se pregunta al usuario si está seguro de querer borrar el registro que aparece en pantalla y se espera que teclee las letras "S" en caso afirmativo o "N" en caso de retractarse. Existen dos tipos de bajas: la baja física que implica borrar físicamente el

registro y la baja lógica que implica marcar el registro mediante la modificación de un estado lógico (ACTIVO, INACTIVO, CANCELADO, IMPRESO, etc.). En caso de negación se libera el registro simplemente y no ocurre nada.

- Sección de Cambios: Además de las dos operaciones anteriores es necesario la edición para cambiar sólo algunos campos de algún registro. En esta sección se captura el índice principal para buscar y presentar el registro completo, de igual manera se valida la existencia del registro para que en caso contrario se active la rutina de error y se indique que ese registro no existe retornando a la sección de "cambios" para reintentar nuevamente. Si la búsqueda fue verdadera se presenta todo el registro para permitir posicionarse en el campo deseado y editarlo, al terminar la operación de igual manera que en la sección de "altas" se trabajará con los segundos niveles (y con los subsecuentes niveles si es que existen).
En la sección de bajas y cambios es importante cuidar la condición de multiusuario del sistema para brindar protección a la integridad de la base de datos, utilizando para ello los comandos de REREAD y UNLOCK.

- Sección de consultas: Cuando en la sección del menú se elige la opción de consultas se cede el control a la rutina de consulta, en donde en algunos casos se pedirá capturar los parámetros iniciales que condicionan la información a presentar y se encadena a algún programa o módulo de consulta que describiremos posteriormente.

- Sección de salida: En está sección se dá por terminada la sesión de captura cerrándose los archivos, abandonando el programa y regresando el control al menú principal.

- Sección de rutinas generales: Aquí se programan las rutinas específicas de cada aplicación que sirven para la validación de datos y la búsqueda de información adicional

en las relaciones, así como la posible actualización de otras relaciones ligadas al proceso (transacciones) o cálculos especiales.

- Teclas de control: El lenguaje Dataflex de cuarta generación (ver apéndice A) maneja algunas teclas preprogramadas para la operación de los procesos de captura a la base de datos, nos permite además programar otras teclas para funciones adicionales de uso específico de la aplicación, por ejemplo:

Tecla	*Función*
F1	*Ayuda en línea*
F2	*Regresar un campo*
Escape	*Aborta la operación en curso*
F9	*Limpia la pantalla y regresa el cursor al inicio*
Tab	*Llama al módulo de consultas*
F7	*Consultas adicionales en el módulo de consultas*
F8	*Consultas adicionales en el módulo de consultas*

Aunque el software obtenido se maneja como un prototipo hasta que el usuario tiene la claridad de sus requerimientos, se tiene hasta este momento, un modelo representativo pero que no trabaja completamente, ya que la plantilla se arma sólo con los campos que intervendrán (como resultado del análisis de requerimientos), así como de algunas de las funciones que se realizarán, como también de las validaciones y relaciones involucradas en el proceso, con esta información el desarrollador trabaja sobre el diseño de la base de datos considerando las relaciones necesarias y creándolas en caso de ser requeridas de tal manera que se puedan normalizar, además de adicionar las rutinas específicas del sistema en desarrollo, posteriormente se tienen más entrevistas con el usuario y se revisan y actualizan todos los cambios necesarios, haciendo un proceso iterativo desde la fase de

análisis de requerimientos como se mencionó en el método propuesto, hasta tener un producto terminado.

3.3 MODELOS DE CONSULTAS.

Una de las operaciones descritas dentro de las opciones del "ABC" es la de consultar a la base de datos, por lo que desarrollamos también un prototipo de consultas, para apoyar al usuario en la manipulación y visualización rápida de la información contenida en los archivos. Dentro de las consultas tenemos dos tipos de acuerdo a su manejo, el modelo bidireccional y el modelo multidireccional que permiten consultas rápidas (a catálogos y directorios principalmente) mientras se captura. Esto es, cuando se llega a un campo en que se necesita un conocimiento previo de alguna clave (número de cliente, número de parte, etc.) utilizamos la tecla preprogramada (Tab) para activar la consulta y es posible consultar estos catálogos para obtener la clave deseada rápidamente.

Estos prototipos aunque un poco más elaborados que los del "ABC" (con la diferencia de que el "ABC" nos sirve para ayudar al usuario a definir su sistema), este prototipo es de apoyo, para no tener que consultar manualmente catálogos o tener que imprimir reportes en papel para conocer los resultados, además de que proporcionan mayor detalle de la información que los reportes, ya que se pueden manejar varios niveles de detalle y de consulta según se requiera.

Las consultas se elaboran en la fase en la que los archivos y sus índices están completamente definidos y normalizados, ya que es necesario un total conocimiento de los índices para manipular los archivos y presentar la información en la forma deseada.

Al igual que la descripción del "ABC", manejaremos la vista que tiene el usuario y la vista que tiene el desarrollador.

3.3.1 MODELO BIDIRECCIONAL

Tenemos dos modelos de consulta de acuerdo a su operación, en primer término describiremos las consultas bidireccionales o de manejo solo vertical, llamadas así porque presentan la información a consultar en forma de lista vertical y hacen movimientos de los registros solo hacia arriba y/o hacia abajo (scrolling). La aplicación principal de estas consultas es a catálogos o para establecer parámetros de selección en las consultas de varios niveles.

Clientes Ordenados por Facturación

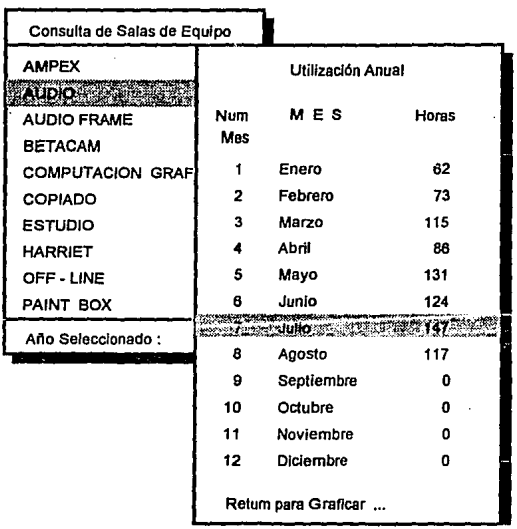
Orden	cli	Razon Social	Total Facturado
1	1	TELEVISA, S. A. DE C. V.	10
2	124	DOS PRODUCCIONES, S. A. DE C. V.	9
3	66	FILM CORE, S. C.	8
4	486	DANIEL J. EDELMAN, INC.	7
5	406	EDITORIAL ERES, S. A. DE C. V.	6
6	32	CINE POR ACTO, S. A. DE C. V.	5
7	13	QUITTANI PRODUCCIONES, S. A. DE C. V.	4
8	61	EDITORIAL OMGSA, S. A. DE C. V.	3
9	208	ALEJANDRO GAVIRA DE LA ROSA	2
10	29	IMAX PRODUCCIONES, S. A. DE C. V.	1

Seleccione y Presione <Return> para Detalle

Ejemplo de una consulta bidireccional al catálogo de clientes ordenados por mayor facturación.

Vista del Usuario:

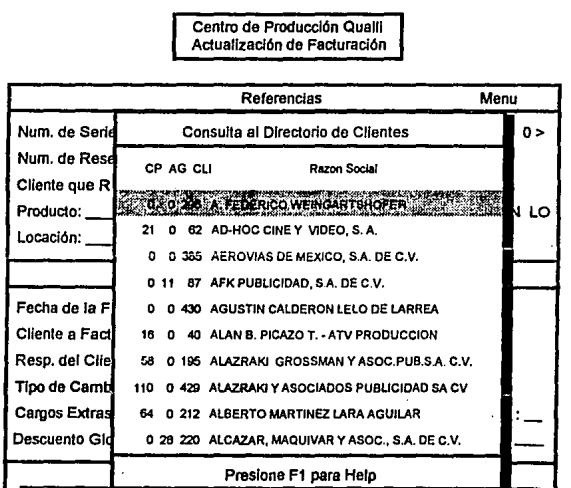
Las consultas pueden tener varios niveles de encadenamiento de acuerdo al detalle de la operación, esto es, al seleccionar un parámetro en una consulta de un nivel inicial, este parámetro puede servir de condicionante para llamar a la siguiente consulta y asi sucesivamente, por ejemplo :



Ejemplo de consulta bidireccional encadenada de dos niveles.

En esta consulta la selección tomada del primer nivel (audio) pasa como condición inicial para la segunda consulta, la selección tomada de la segunda consulta (Julio) pasa como condición para presentar la información deseada.

Otra aplicación de estas consultas son las llamadas "Consultas Rápidas", en donde se apoya al usuario cuando está capturando algún campo que requiere de un conocimiento previo (conocer una clave de un producto o de un cliente por ejemplo) y desea consultar al catálogo correspondiente, entonces, al estar posicionado sobre el campo en cuestión se presiona una tecla programada previamente (Tab) para llamar a la consulta y aparecerá una pantalla con los registros del catálogo deseado, se podrá entonces seleccionar la clave que se requiere sin necesidad de consultar manualmente el catálogo, y al presionar la tecla <Return> la clave seleccionada se trasladará hacia el campo que requirió la clave.



Ejemplo de una "Consulta Rápida" en donde la primer pantalla es un ABC que solicita una clave del cliente y se activa la consulta rápida con la tecla de (Tab).

El manejo de estas consultas sigue la misma filosofía de "apunta y dispara" que se maneja comunmente, es decir, se puede realizar una navegación (hacia arriba o hacia abajo) en la pantalla, estos movimientos se realizan con las teclas flecha arriba (↑) o flecha abajo (↓) para un movimiento entre cada uno de los registros o se puede obtener un movimiento de página completa con las teclas de página arriba (pg-up) o página abajo (pg-dw) según se requiera, además se puede ir al inicio de la consulta, con la tecla de inicio (Home,Inicio) o al final, con la tecla de (End, Fin) y una vez ya seleccionado el registro se presiona la tecla <Return>.

Existe una manera de ir rápidamente al registro deseado y es a través de una búsqueda rápida (F10), ésta sólo se puede efectuar si dentro de los campos presentados se tiene alguno en forma de string, fecha o entero (por ejemplo razón social del cliente, nombre del banco, nombre del empleado, etc.) y se obtiene al teclear las primeras letras de la cadena deseada en el campo indicado por la pantalla, ésta servirá de condicionante para buscar dentro del archivo a todos aquellos registros que contenga la cadena deseada.

Por ejemplo: se requiere buscar al cliente "PROMOCIONES Y FILMACIONES PARA T.V." podríamos teclear algún subconjunto de la cadena a buscar por ejemplo, la subcadena "PRO" y la consulta buscará entonces en el campo de razón social de todos los registros, de aquellos clientes que contengan esta subcadena y los desplegará, ejemplo :

"PRODUCTORA DE TELEPROGRAMAS S.A."
"PROMOCIONES Y FILMACIONES PARA T.V."
"PROMOCIONES ESPECIALES PARA CINE S.A."

Una vez posicionado en el registro deseado, se presiona la tecla <Return> para aceptar la selección, en este momento dependiendo de la aplicación, puede terminar ahí la consulta

trasladando la clave al campo de captura que lo solicitó. Si la consulta es de varios niveles, entonces en este momento se encadenará a la consulta del siguiente nivel y así sucesivamente hasta el último nivel en donde se efectuará el traslado de las claves.
En algunas consultas se puede obtener información adicional del registro seleccionado al presionar la tecla <F7>, (que como se mencionó anteriormente, es una tecla preprogramada) en donde el sistema presenta una pantalla adicional con información a detalle.

Vista del Desarrollador :

El programa de consulta está dividido en: presentación de pantallas, acción a ejecutar y manejo de cursor.

- Presentación de pantallas. En esta sección se diseña la presentación de los campos en la pantalla, se calcula el número de renglones y de columnas que se manejarán, se les asigna nombres o identificadores a las ventanas del primero y último renglón, Dataflex maneja pantallas o imágenes de captura, a cada campo de captura de datos en estas pantallas se les conoce como ventanas de captura, se les identifica por el nombre de la pantalla y el orden que guardan, tomando la referencia de izquierda a derecha y de arriba a abajo, por ejemplo, si la pantalla se llama catálogo, a la primera ventana se le identifica como catálogo.1, a la segunda como catálogo.2 y así sucesivamente o también se les puede asignar un identificador o nombre.
El cuerpo principal del programa inicia presentando una primer pantalla (rutina "desp_pant_ini") que se forma mediante un ciclo de búsqueda con las condiciones iniciales y despliega los 10 primeros registros (este número puede ser variable y depende del tamaño de la pantalla que se quiera manejar) que las cumplan, se ilumina con video inverso el primer renglón indicando de este modo el renglón al que apunta el cursor.

Acción a ejecutar. Aquí se tiene la rutina "entracar" la cual censa el teclado, esperando que se presione una tecla (de navegación o de selección) para qué, dependiendo de la tecla presionada, se ejecute la rutina correspondiente, éstas rutinas son:

mueve_cursor_arriba : Investiga la posición del cursor dentro de la pantalla de consulta, si se encuentra en el inicio de la pantalla (primer renglón) llama a la rutina "busca_arriba" que efectúa una búsqueda en dirección al inicio del archivo analizando las condiciones de búsqueda requeridas, si ya se está en el inicio del archivo o no se cumplen las condiciones establecidas, nos lo indica con una campanada y dá por terminada la consulta sin modificar la posición del cursor; si no, recorre todos los renglones una posición hacia abajo para dejar libre el primer renglón en dónde inserta el nuevo registro encontrado en el archivo (esta acción la efectúa la rutina llamada "scrolling"). En el segundo caso sólo se mueve el cursor al renglón anterior al que se encuentra.

mueve_cursor_abajo : Investiga la posición del cursor dentro de la pantalla de consulta, si se encuentra en el final de la pantalla (último renglón) llama a la rutina "busca_abajo" que efectúa una búsqueda en dirección del final del archivo analizando las condiciones de búsqueda requeridas ,si ya se está en el final del archivo o no se cumplen las condiciones establecidas, nos lo indica con una campanada y dá por terminada la consulta sin modificar la posición del cursor; si no, recorre todos los renglones una posición hacia arriba para dejar libre el último renglón en dónde inserta el nuevo registro encontrado en el archivo (esta acción la efectúa la rutina llamada "scrolling"). En el segundo caso solo se mueve el cursor al renglón posterior al que se encuentra.

mueve_pag_arriba : Ejecuta la rutina "mueve_cursor_arriba" en forma iterativa 10 veces o el número de renglones asignados a la consulta, ya que se trata de mover la pantalla por página completa hacia arriba.

mueve_pag_abajo : Ejecuta la rutina "mueve_cursor_abajo" en forma iterativa 10 veces, de igual manera que la anterior, sólo que con movimiento hacia abajo..

desp_pant_fin : Despliega la pantalla a partir de los parámetros que se pasaron inicialmente, pero partiendo del final del archivo, de manera que, la búsqueda la inicia de atrás hacia adelante, esta operación resulta de mucha utilidad cuando tenemos archivos demasiado grandes y únicamente queremos revisar la última captura.

Dentro de la sección de acción a ejecutar tenemos el siguiente grupo de teclas :

Key.return : Al presionar la tecla de retorno debemos estar posicionados en la pantalla en el registro deseado, si es una consulta de varios niveles la rutina se enlaza al siguiente nivel de consulta pasándole los parámetros tomados del registro sobre el que se presionó <Return> ; por otra parte y en caso de que la consulta sea de un solo nivel, se trasladará la clave principal al campo de captura que requirió a la consulta y se abandona la consulta también .

key.user (F7) : Con este comando podemos realizar una consulta a detalle del registro seleccionado, ya que en la mayoría de los casos, no es posible presentar toda la información en una sola pantalla. Aquí también, podemos tener enlaces de varias pantallas presionando cualquier tecla, y al llegar a la última nos lleva de regreso a la consulta principal.

key.user2 (F8) : Funciona igual que F7, sin embargo la información que nos presenta puede estar ligada con otros archivos y nos sirve para completarla únicamente.

key.find (F10) : Esta función nos ayuda a localizar a un conjunto de registros (dentro del mismo archivo), que posean caracteristicas similares, como pueden ser, un rango de fechas, un rango de números, una subcadena del nombre o razón social, etc. Para ello nos presenta una pantalla especial, en donde nosotros pondremos dichas condiciones, las cuales pueden ser nulas también.

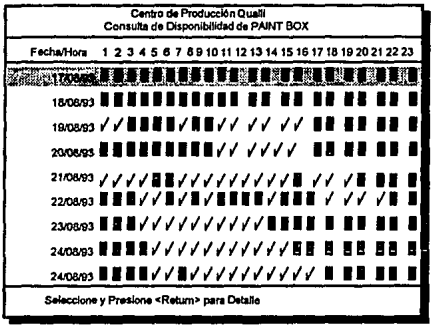
key.escape (ESC) : Esta función da por terminada la consulta regresando el control al programa que la llamó, sin trasladar ninguna clave a dicho programa.

3.3.2 MODELO MULTIDIRECCIONAL:

Esta es una consulta que cumple todas las caracteristicas del modelo bidireccional, la única diferencia es que nos permite realizar además una consulta con movimiento de izquierda a derecha dentro de un mismo registro; como por ejemplo, la consulta de una agenda de trabajo, la consulta de un rango de horas, etc. Para lo cual desarrollamos las siguientes rutinas que complementan al modelo bidireccional :

key.right : mediante esta función, el usuario puede oprimir la tecla flecha derecha, con lo cual la consulta efectuará un movimiento del cursor a la derecha, es decir, se posicionará en el siguiente campo del registro, siempre y cuando exista.

key.left : mediante esta función, el usuario puede oprimir la tecla flecha izquierda, con lo cual la consulta efectuará un movimiento del cursor a la izquierda, es decir, se posicionará en el siguiente campo del registro, siempre y cuando exista.



Ejemplo de una consulta multidireccional, en donde se permiten movimientos del cursor de izquierda a derecha y de arriba hacia abajo.
Estas consultas son muy útiles cuando tenemos un archivo con muchos campos o que éstos son muy grandes para desplegarlos en una sola pantalla.

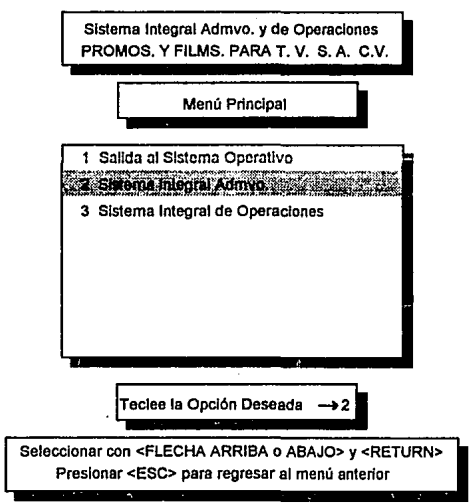
3.4 EL MENU COMO MODELO DE ACCESO AL SISTEMA.

El menú es un módulo que nos sirve para accesar todos y cada uno de los programas que conforman los subsistemas. El módulo del menú está dividido en dos partes esenciales:

- Presentación y control de la pantalla principal: Esta sección está conformada por un programa que maneja las siguientes partes: encabezado, donde se indican los datos generales del sistema, como lo son el nombre del sistema y el nombre de la compañía. El sub-encabezado, donde ubicamos el lugar en el cual se encuentra el usuario, nombre del sistema y proceso (menú origen y menú actual). El menú, es donde se presentan cada una de las opciones que se pueden elegir dentro de un sistema, y al seleccionarla el menú

cederá el control al programa o subsistema que aquí se tenga indicado, y cuando termina recupera el control automáticamente, permitiéndonos navegar de nuevo por el menú. En este punto, podemos adicionar una clave de acceso a las opciones que queramos que estén restringidas para algunos usuarios (ver apéndice A).

- Control de la información del menú. La columna vertebral del menú está realizada sobre un archivo de bases de datos. Este archivo es alimentado, a través de un programa suministrado por Dataflex llamado MENUDEF, el cual nos sirve para capturar cada una de las pantallas del menú con todas sus características y sus respectivas opciones, también, nos permite construir las "ligas" internas de cada menú, como lo son : menú origen , menú destino y acción a ejecutar.



Ejemplo de una pantalla terminada del menú

En el apéndice A describiremos la forma en que se puede manipular y actualizar los diferentes menús a través de la utilería MENUDEF.

3.5 LOS REPORTES COMO MODELO DE SALIDA DE DATOS.

Una forma adicional de tener vistas de la información de la base de datos es la de obtener la salida de los archivos de la base de datos para reportes a papel, etiquetas, disco y/o formas preimpresas; para esto también desarrollamos plantillas de los reportes más comúnes, y tratamos con esto de tener un solo formato de los reportes que se manejan en la empresa, principalmente aquellos que van dirigidos a los ejecutivos en forma de estadísticas e informes especiales.

Dataflex tiene integrado un generador de reportes el cual es una herramienta de cuarta generación llamada DFQUERY, que puede generar un programa de reporte, sin embargo, no es todavía lo suficientemente poderosa para realizar diseños complejos. Por otra parte Dataflex tiene una macroinstrucción llamada REPORT la cual maneja varias secciones y con las que armamos nuestra plantilla de reportes. Esta macroinstrucción nos brinda un mayor y mejor servicio para la creación de diseños más completos.

El comando REPORT consiste de un conjunto de rutinas predefinidas de salida que pueden combinarse según las necesidades. Un programa desarrollado con el comando REPORT está dividido en dos áreas principales: líneas de imágenes y líneas de comando. Las líneas de imágenes especifican los encabezados, ventanas y su formato. Las líneas de comandos contienen comandos que le indican al comando REPORT de como "llenar" con datos las áreas de imágenes para su salida.

Las áreas de imágenes y comandos contienen secciones etiquetadas con diferentes

direcciones de requerimientos funcionales de un reporte, por ejemplo.:

Sección HEADER: En esta sección se define como será el encabezado en cada hoja, por ejemplo : el nombre de la compañía, el nombre del reporte, la fecha de emisión, el rango de fechas que abarca el reporte, el número de página, etc.

las imágenes etiquetadas de un reporte son:

/HEADER

/SUBHEADER#

/BODY

/SUBTOTAL#

/TOTAL

El caracter diagonal (/) es parte de la sintáxis para nombrar a una imagen, el símbolo (#) después de las palabras SUBHEADER y SUBTOTAL, pueden ser un digito entre el 1 y el 9, para indicar en que nivel de corte para subtotal se encuentra el programa.

La sección BODY es la sección principal de un reporte ya que se puede usar solo esta sección sin utilizar las secciones de header, subheader, subtotal y total.

En la sección de BODY se utiliza el comando PRINT el cual tiene dos funciones: (a) mover los datos desde la base de datos hasta su respectiva posición en la sección de imágenes para una subsecuente presentación y (b) la actualización para subtotalizar o totalizar algún registro, esto es, almacenar acumulados por cada dato que se envia a alguna ventana .

El comando REPORT soporta hasta 9 niveles de cortes de subtotales, las secciones de SUBHEADER# y SUBTOTAL# están ligadas en los procesos de cortes. La sección de TOTAL acumula los totales de todos los campos que se envian a la salida presentando un total general.

A continuación se muestra un ejemplo de las secciones posibles de un reporte utilizando el comando REPORT.

Secciones de Imágenes:

/HEADER

/SUBHEADER1

/SUBHEADER2

/BODY

/SUBTOTAL2

/SUBTOTAL1

/TOTAL

/SELECTION

*/**

Secciones de Comandos:

OUTFILE

OPEN FILENAME1 BY INDEX.#

OPEN FILENAME2

MOVE # TO PAGEEND

REPORT filename1 BY INDEX.# BREAK filename1.field filename2.field

INDICATE SELECT AS data_element LT\LE\EQ\GE\GT SELECTION.#

SECTION HEADER

comandos

SECTION SUBHEADER1 ###

comandos

SECTION SUBHEADER2 ###

comandos

SECTION BODY

comandos

SECTION SUBTOTAL1

comandos

SECTION SUBTOTAL2

comandos

SECTION TOTAL

comandos

REPORTEND

Sección Header. En esta sección se manejan los encabezados que aparecen al inicio de cada hoja, la plantilla maneja rutinas para que aparezcan los rangos de fechas que abarca el reporte, además se debe de indicar el nombre de la empresa, el nombre del programa y la descripción de los datos que aparecerán inmediatamente después. También se puede manejar un indicador de número de página.

Sección SubheaderX. En esta sección se efectúan cortes parciales y va asociada con el manejo de subtotales parciales por lo tanto a cada sección de subheaderX le corresponde una sección de subtotalX.

Sección Body. En esta sección se maneja la información contenida en el registro en turno y se pueden imprimir los campos que se deseen del registro y se pueden implementar búsquedas a otros archivos relacionados e imprimirlas también.

Sección SubtotalX. Como se mencionó en la parte de subheaderX, se pueden manejar acumulados de los campos numéricos del Body y presentarlos en esta sección cuando ocurre un corte.

Sección Total. Esta es la última sección del REPORT y maneja los acumulados de las secciones de subtotalX presentando un total general de las operaciones.

Rutinas adicionales : Se maneja en esta plantilla rutinas para el manejo de fechas para cambiar de formato DD/MM/AA al formato día del mes de año .

Se habilitan además las teclas de :

Key.escape (Esc) : Para dar fin al proceso en cualquier momento.
key.Backfield (F2) : Para retroceder al campo anterior en la captura de los datos de condición.
key.clear (F9) : Para borrar todos los parámetros de selección.

3.6 MODELOS DE ESTADISTICAS.

El manejo de estadísticas en la empresa es de mucha importancia ya que pueden indicar el comportamiento administrativo, operativo o financiero y ayudar a la toma de decisiones, las estadísticas que se manejan están orientadas a los directivos de cada área y se dividen en : Administrativas y Operativas :

y como ejemplos se citan las siguientes:

- *Ventas acumuladas*
- *Cobranza acumulada*
- *Compras acumuladas*
- *Pagos acumulados*

- *Utilización del equipo*
- *Utilización de las Salas*
- *Comportamiento de los Clientes*

Estas estadísticas se manejan con el propósito de presentar la información acumulada al día, semana, mes y en el año seleccionado (se detallan posteriormente).
La información estadística que se maneja no es en línea ya que se requerían de muchos recursos de cómputo, por lo que se optó por manejar archivos de base de datos que tuvieran sólo la información estadística (diaria, semanal, mensual y anual) y que se generan diariamente. Para lograr esto se desarrolló un sistema de base de datos auxiliar (descrito en el capítulo 1.2) que básicamente se utiliza en este tipo de aplicaciones estadísticas.

3.6.1 GENERACION DE ESTADISTICAS.

Para generar los archivos de estadísticas se ejecutan programas que trabajan con los archivos de datos reales y en base a éstos se generan los archivos de estadísticas que acumulan la información diaria, semanal, mensual y anual.

Vista del Usuario :

Estos programas los ejecuta un solo usuario al inicio del día y solo ingresa la fecha inicial y la fecha final que se procesará y el programa indicará la fecha que se está procesando.

Vista del desarrollador :

La plantilla que maneja la generación de estadísticas es muy fácil de actualizar, se requiere en primer lugar que esté definido el archivo de estadísticas; el manejo y presentación de la pantalla de selección es similar a la que se maneja en los reportes, primero aparece el logotipo indicando la descripción de la estadística de que se trata, después la pantalla de selección en donde se permite ingresar por ejemplo la fecha inicial y la fecha final. Posteriormente se maneja un ciclo en donde de manera iterativa se realiza una búsqueda selectiva en todo el archivo (por rango de fechas) el cual se ordena por fecha de acuerdo al índice establecido y se manejan rutinas que actualizan el archivo de estadísticas generando registros para la información diaria, semanal, mensual y anual.

3.6.2 REPORTE DE ESTADISTICAS.

Estos reportes normalmente están dirigidos a los directivos. Condensan la información presentando sólo las cifras más relevantes para que de una manera rápida se conozcan la situación administrativa, financiera y/o operativa de la empresa.

Vista del Usuario :

Se maneja un encabezado en donde se indica el nombre de las estadísticas y la fecha a la que corresponden, a continuación se detallan los movimientos del día con indicaciones de a quién corresponde y sus importes, también se totaliza al final las operaciones del día. A continuación se presenta un ejemplo de un reporte de estadísticas de ventas en donde se puede observar los acumulados del año y del mes, en moneda nacional y dólares, hasta el día anterior, en el siguiente renglón los importes del día y al final los acumulados hasta el

día, de esta manera y en una forma rápida y concisa se obtienen las cifras más importantes de las ventas de la empresa, en estas aplicaciones es donde nos resulta de gran utilidad el uso de los archivos de bases de datos auxiliares.

Promociones y Filmaciones para T.V. S.A. de C.V.

Reporte del diario de Ventas al 02 de Enero de 1994

Núm Factura	*Núm. Reserva.*	*Fecha Factura*	*Razón social del cliente*	*Importe Facturado (M.N.)*	*(Dls)*
1-23	*234*	*02/01/94*	*Productora MOBA S.A*	*$30.00*	*$10.00*
1-24	*235*	*02/01/94*	*Publicidad HECA, S.A. DE C.V.*	*$90.00*	*$30.00*
			TOTAL FACTURADO DEL DIA:	*$120.00*	*$40.00*

	<— IMPORTE EN M.N. —>		*<— IMPORTE EN DLS. —>*	
	MES PRESENTE	*AÑO PRESENTE*	*MES PRESENTE*	*AÑO PRESENTE*
Hasta Ayer:	*$30.00*	*$60.00*	*$10.00*	*$20.00*
Hoy	*$120.00*	*$120.00*	*$40.00*	*$40.00*
Hasta Hoy :	*$150.00*	*$180.00*	*$50.00*	*$60.00*

Ejemplo de un reporte de estadísticas de ventas

Vista del desarrollador:

La plantilla para este reporte, es de las más fáciles de actualizar ya que es un formato único y con sólo remplazar el nombre del archivo y sus campos se puede obtener un nueva versión. Esta plantilla es la que menos modificaciones necesita.

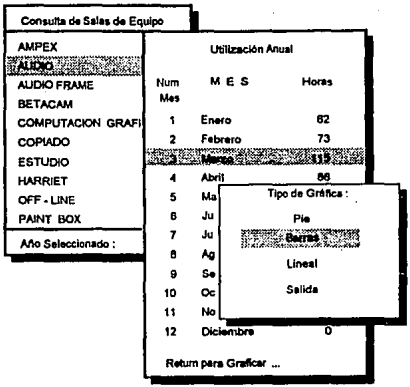
3.6.3 REPRESENTACION GRAFICA

El sistema puede manejar la información en una representación gráfica para tener una visualización más objetiva de los resultados. Normalmente las gráficas presentan información financiera de las operaciones acumuladas (diarias, mensuales y/o anuales) y se maneja a través de consultas de varios niveles para tener los parámetros que servirán de condicionante a la información que se presentará.

Se pueden tener gráficas tipo pastel, barras o lineales. El usuario puede seleccionar el tipo que desee.

Para utilizar el modelo de representaciones gráficas el programador necesita tener los parámetros de condición para seleccionar la información deseada, para lo cual puede utilizar los modelos de consulta antes descritos y armar una pantalla con una lista del tipo "X, Y", es decir, una lista vertical con elementos uno a uno, después utilizar una sola rutina para dibujar la gráfica, esta rutina está basada en las macroinstrucciones de Dataflex para el manejo de gráficas.

A continuación se muestra un ejemplo de manejo de estadísticas a través de gráficas, en este ejemplo se encadenan dos consultas bidireccionales para seleccionar los parámetros iniciales de la gráfica,en la primera consulta se selecciona el tipo de sala de edición, en la segunda se selecciona el mes que se desea consultar y finalmente en la tercera consulta el tipo de gráfica que se presentará, (pastel, barras o lineal).



Ejemplo de consultas para seleccionar parámetros

Ejemplo de Gráfica de tipo de Barras

CAPITULO 4 APLICACION DEL MODELO PROPUESTO EN EL DESARROLLO DEL SISTEMA "SIACO"

4.1 FASE DE DEFINICION DEL SISTEMA.

En el presente capítulo expondremos la aplicación del modelo propuesto para el desarrollo de sistemas, a un solo proceso del sistema "SIACO", ya que el analizar todo el sistema resultaría un documento de gran volumen, por lo cual sólo analizaremos globalmente a las demás partes que lo conforman y para conocer en forma muy somera todas aquellas áreas que se automatizaron sus procesos. Analizaremos en detalle el área de Programación de Servicios de las Salas de Producción y Post-Producción, por ser la más representativa dentro del contexto general de la empresa, es en esta área donde se inicia el proceso operativo, ya que es el primer contacto que tenemos con el cliente (ver "Introducción").

A continuación se describen los subsistemas que conforman al sistema SIACO.

Area Administrativa:

En el área administrativa están automatizados los siguientes departamentos:

- *Contabilidad*
- *Cobranzas*
- *Personal*
- *Tesorería*
- *Facturación*

y en el Area Operativa:

- *Almacén*
- *Reservaciones*
- *Relaciones Públicas*
- *Asesores de Post-Producción*
- *Paquetes de Servicios*
- *Plan Francés*

Detallaremos a continuación en forma breve las principales características y objetivos de cada uno de estos procesos.

PROCESO: CONTABILIDAD

OBJETIVO: *Control y manejo de las cuentas contables para determinar la situación financiera y contable de la empresa.*

DESCRIPCION: *Este sistema maneja la contabilidad en su forma tradicional, es decir, la afectación de cuentas contables a través de cargos y abonos en pólizas. Una característica importante que tiene este subsistema, es su amplia conectividad con otros procesos de la empresa, por lo que este subsistema se convierte en el eje donde van a caer todos los sistemas relacionados, por ejemplo:*

El subsistema de Personal a través de la nómina de empleados, al emitir los cheques genera pólizas de egresos con los importes pagados, afectando a su vez las diversas cuentas contables involucradas, como lo son las de bancos y las de gastos de las gerencias y direcciones.

El subsistema de Facturación al imprimirse la facturación del día genera un diario de ventas que a su vez genera una póliza de diario que afecta a las cuentas de los clientes e ingresos.

En el subsistema de Cobranza, al efectuar cobros, se actualiza las cuentas de bancos y de clientes.

En el subsistema de Tesorería, con los egresos por pagos varios, se actualizan además de los bancos, las cuentas contables correspondientes. Todo lo anterior repercute en un mayor apoyo y reducción de la carga de trabajo en los departamentos antes mencionados, y principalmente al de Contabilidad.

Los procesos principales con los que cuenta actualmente este subsistema, se enlistan a continuación :

ACTUALIZACION:

- *Actualización de catálogos de cuentas (cuentas de mayor, sub-cuentas y sub-sub-cuentas)*
- *Actualización de pólizas*
- *Conciliación mensual de clientes*
- *Conciliación mensual de cobranza*
- *Conciliación mensual de tesorería*
- *Conciliación mensual de nómina*
- *Conciliación mensual de bancos*
- *Actualización de documentos*
- *Actualización del catálogo de estados financieros*
- *Actualización de estados financieros*

REPORTES Y CONSULTAS:

- *Catálogo de cuentas*
- *Pólizas en general.*
- *Balanza de comprobación*
- *Auxiliar de cuentas*
- *Estados financieros*
- *Pólizas de ventas del día*
- *Pólizas de cobranza del día*
- *Pólizas de cheques de tesorería*
- *Pólizas de cheques de nómina*
- *Consultas al sistema de facturación*

- *Consultas al sistema de cobranzas*
- *Consultas al sistema de nómina*
- *Consultas al sistema de tesorería*

PROCESOS ESPECIALES:

- *Traspaso de saldos*
- *Actualización de saldos*
- *Inicializar saldos*
- *Comprobación de pólizas y consistencia de cuentas*
- *Conversión de pólizas a saldos iniciales*

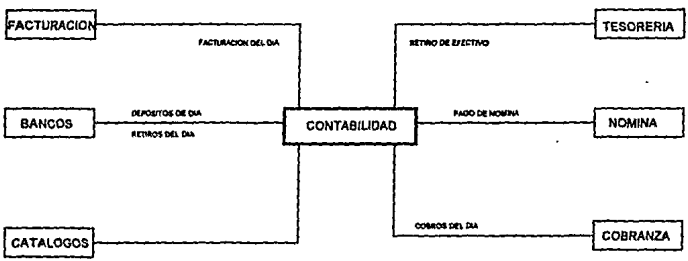


Diagrama de bloques del proceso de contabilidad
y sus relación con otros procesos.

PROCESO: COBRANZAS

OBJETIVO: *El control adecuado de créditos a los clientes y la cobranza de las facturas, así como de la cartera vencida de los clientes.*

DESCRIPCION: *Siguiendo una secuencia cronológica del flujo de la información. Cobranzas es el siguiente paso de la facturación ya que cuando se genera una factura se actualiza en las cuentas por cobrar y se tiene que enviar a revisión del cliente y de acuerdo a los días de crédito, se deberá de realizar el cobro.*

Mediante este departamento, se canalizan los problemas o dudas de los clientes que pudieran surgir en las facturas, hacia los departamentos involucrados, como lo son: facturación, programación de servicios y contabilidad.

Una vez realizado el cobro se actualiza en el sistema de cobranzas, y en ese momento se descarga de las cuentas por cobrar, se actualiza en los bancos el depósito y en las cuentas contables correspondientes (ingresos y clientes).

Los principales módulos que maneja son:

ACTUALIZACIONES:

- *Contrarecibos a revisión*
- *Cobranza de facturas*
- *Comentarios de estado de clientes*
- *Actualización de anticipos*

CONSULTAS Y REPORTES:

- *Cobranza del dia*
- *Antigüedad de saldos*
- *Estados de cuenta de clientes*
- *Acumulado de cobranza mensual y anual*
- *Comentarios de estados de clientes*
- *Kardex de clientes.*
- *Anticipos*

ESTADISTICAS:

- *Comparativo de ventas contra cobranza*
- *Gráfica de cobranza semanal, mensual y anual.*
- *Gráfica de saldos mensuales*
- *Gráfica de acumulados de cobranza comparativa*
- *Estadísticas de clientes*

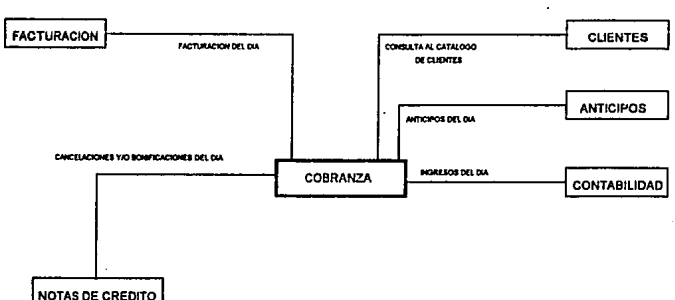


Diagrama de bloques del proceso de cobranzas
y su relación con otros procesos.

PROCESO: <u>DEPARTAMENTO DE PERSONAL</u>

OBJETIVO: *Automatizar todos los procesos del departamento de personal para un mejor control de los recursos humanos.*

DESCRIPCION: *En este departamento se manejan procesos estándares como lo son: el manejo de nóminas, kardex, impuestos, declaraciones, ya que éstos son lo mismo en muchas empresas con la única diferencia que se tienen que manejar algunos procesos adicionales para enviar información a la empresa matriz.*

El sistema de Personal cuenta con algunas características que lo hacen flexible y práctico, por ejemplo: Se pueden manejar varias empresas a la vez, del cálculo de la nómina se pueden obtener reportes contables y administrativos, al imprimir los cheques de la nómina existe comunicación con el sistema de contabilidad y se actualizan las cuentas contables correspondientes, se tiene un conjunto de reportes muy variados de datos acerca de los empleados.

Se comunica también con el banco para el manejo del SAR de los empleados y lo mejor aún, es que su diseño a soportado los cambios fiscales que a marcado la Secretaría de Hacienda.

ACTUALIZACIONES:

- *Conceptos de nómina (percepciones y deducciones)*
- *Tablas de: IMSS, ISPT (mensual y anual), crédito al salario, vacaciones y días festivos.*
- *Inicialización del periodo de nómina*
- *Actualización de nómina por periodo*

- *Cálculo de la nómina por período*
- *Actualización del formato HISR-90*
- *Actualización de empleados*
- *Actualización de direcciones*
- *Actualización de gerencias*
- *Actualización de departamentos*
- *Actualización de empresas*

Y LAS PRINCIPALES CONSULTAS Y REPORTES:

- *Recibos de nómina*
- *Cheques de empleados (nómina, aguinaldos y PTU)*
- *Declaraciones anuales de impuestos retenidos*
- *Distribución de vales de comida y de despensa para empleados*
- *Reportes de nómina en múltiples modalidades*
- *Reportes de empleados en múltiples modalidades*
- *Proyección de salarios*
- *Reporte de puestos, áreas, departamentos, tablas de impuestos, conceptos de nómina e IMSS.*
- *Impresión del formato de ingresos y egresos*

Y EN OTROS PROCESOS:

- *Generación de archivo para SAR/INFONAVIT*
- *Generación del PTU*
- *Generación de datos HISR-90*
- *Manejo de fondo de ahorro*
- *Manejo de aguinaldos*

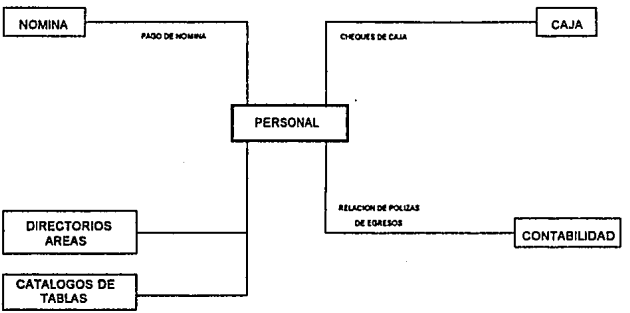


Diagrama de bloques del proceso de personal
y su relación con otros procesos.

PROCESO: TESORERIA

OBJETIVO: *Controlar los movimientos bancarios (depósitos, retiros y saldos) de las cuentas bancarias de la empresa y controlar el manejo de las cuentas por pagar a proveedores.*

DESCRIPCION: *Dentro de las operaciones más delicadas de la empresa, está el manejo de los ingresos y egresos en el departamento de Tesorería, por lo que fue necesario automatizar sus operaciones a través de un sistema de cómputo. Por la naturaleza de sus operaciones se divide en dos partes su procedimiento:*

La primera es el control de los movimientos bancarios, como depósitos en general, principalmente de la cobranza de las facturas por servicios de producción y/o post-producción a clientes, y el manejo de retiros destinados a pagar proveedores, nómina, etc.

El sistema desarrollado para estas necesidades, permite además, imprimir cheques y actualizar las cuentas contables al mismo tiempo, ahorrando con esto, tiempo a los auxiliares de contabilidad y a la cajera.
Este sistema controla además los saldos de cada cuenta y es posible manejar varios bancos con varias cuentas.
Con esto el director administrativo puede consultar el cierre diario de Tesorería y los saldos bancarios.

El segundo procedimiento que se opera en Tesorería es el control de las cuentas por pagar, esto es, a los proveedores de insumos y servicios de la empresa, después de realizar el servicio o entregar los insumos. el proveedor presenta una factura a revisión que al ingresar al sistema de cómputo se convierte en una cuenta por pagar, en ese momento se

imprime un contrarecibo que se le entrega al proveedor con el cual se presenta después a cobrar sus facturas.

Las facturas que por cualquier razón no se paguen, se pueden ir controlando a través de un reporte de antigüedad de adeudos, además de programar pagos y emitir reportes de adeudos y pagos por día.

ACTUALIZACIONES:

- *Actualización del directorio de proveedores*
- *Contrarecibos de proveedores*
- *Pagos a proveedores*
- *Catálogo de proveedores*
- *Gastos por comprobar*
- *Catálogo de Bancos*
- *Catálogo de Cuentas Bancarias*
- *Actualización de movimientos bancarios (depósitos y retiros)*

CONSULTAS Y REPORTES:

- *Pagos a proveedores del dia*
- *Antiguedad de adeudos*
- *Impresión de contrarecibos*
- *Programación de pagos*
- *Estados de cuenta por proveedor*
- *Impresión de cheques*
- *Informe general de cuentas bancarias*

- *Catalógos de proveedores, bancos, cuentas y empresas*
- *Informe bancario por empresa o cuenta*
- *Informe bancario por banco*
- *Reporte de gastos por comprobar*
- *Manejo de caja chica*
- *Manejo de honorarios*
- *Corte del dia*

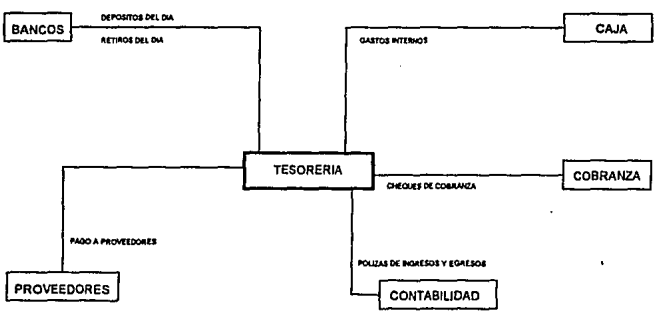


Diagrama de bloques del procesos de tesorería
y su relación con otros procesos.

PROCESO: FACTURACION

OBJETIVO: *Facturar todos los servicios utilizados por los clientes, como son la renta de salas de edición, salas de audio, foros de grabación, operadores y equipo adicional.*

DESCRIPCION: *El área de facturación es el punto de comunicación entre operaciones y administración y su principal fuente de información es la orden de trabajo que se genera en el departamento de Programación de Servicios de acuerdo a lo reportado por las salas, por lo que resulta muy importante que exista una buena comunicación entre estas dos áreas, ya que de lo contrario, se generarán errores en las facturas retrasando el proceso de Cobranza también.*

Existen dos formas de facturar: La facturación directa (que desglosa toda la orden de trabajo) y la facturación de paquetes de servicios, en donde se elabora una sola factura que abarca todo un proyecto completo con varias ordenes de trabajo.

Se maneja además la elaboración (previa autorización) de notas de crédito, que son documentos que bonifican o cancelan a una factura.

Por último, este proceso está relacionado Conontabilidad ya que al elaborar una factura se carga automáticamente a las cuentas contables que corresponda y en Cobranzas se actualizan las cuentas por cobrar.

ACTUALIZACIONES:

- *Actualización de catálogos de equipo, material, servicios*
- *Actualización de los directorios de clientes, agencias y casas productoras*
- *Actualización de facturas*
- *Actualización de notas de crédito*

REPORTES Y CONSULTAS DE:

- *Catálogos de equipo, material (cintas y cassettes) y servicios (tarifas)*
- *Directorios de clientes, agencias y casas productoras*
- *Impresión de facturas*
- *Impresión de notas de crédito*
- *Estados de cuenta por cliente*
- *Ventas acumuladas del día, semanas, mes y año*
- *Consultas al histórico de movimientos de clientes*

ESTADISTICAS:

- *Gráfica de ventas acumuladas*
- *Gráfica de ventas por cliente*
- *Análisis de facturación por mes*
- *Análisis de facturación por sala*
- *Equipo facturado por mes*
- *Clientes por mayor facturación*

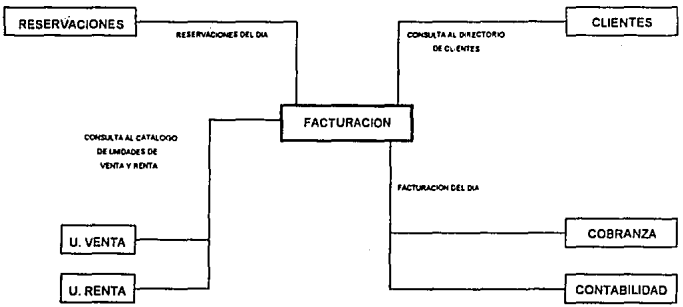


Diagrama de bloques del proceso de facturación
y su relación con otros procesos.

PROCESO: ALMACEN

OBJETIVO: Controlar los movimientos de entrada y salida del material del almacén (cintas y cassettes) y su aplicación a la facturación.

DESCRIPCION: Es necesario para el manejo de grabaciones de video en dispositivos de almacenamiento para que los clientes se puedan llevar sus trabajos de producción y post-producción, para ello se manejan diversos tipos de formatos, medidas de cintas y cassettes, tanto para video como para audio; así como bobinas para su respectiva transportación.
En el almacén se maneja únicamente material virgen; además se controlan las entradas de los dispositivos mencionados hacia el almacén por parte de los proveedores y la salida del material por parte de los clientes y su respectiva aplicación a través de la facturación; y por lo tanto, se indican además las cantidades máximas y mínimas de stock, reorden, etc. El material cuando ingresa al almacén es etiquetado con una clasificación especial que sirve para identificarlo y así poder tener un mejor control del mismo.

ACTUALIZACIONES:

- *Entradas al Almacén*
- *Salidas del Almacén*
- *Aplicación del material en la facturación*

CONSULTAS Y REPORTES:

- *Ordenes de salida*
- *Relación de entradas al almacén*
- *Listado de reorden de material*
- *Listado de stock de material*
- *Material por factura*
- *Compras del mes y año*

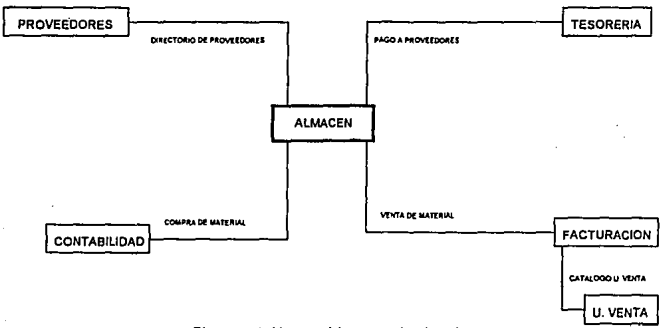


Diagrama de bloques del proceso de almacén y su relación con otros procesos.

PROCESO: RESERVACIONES

OBJETIVO: *Asignar, controlar y administrar los horarios de las salas, la renta de equipo de producción y post-producción, así como de sus respectivos operadores.*

DESCRIPCION: *Con este sistema, se inician las actividades de la empresa ya que el principal ingreso de la empresa es la venta de tiempo de salas y la renta de equipo, para esto el cliente llama telefónicamente (con varios días de anticipación generalmente) y de acuerdo a las necesidades de su trabajo, reserva la sala y el equipo adicional adecuado para su proyecto.*

En este punto, la sala queda como reservada y no podrá ocuparse por otro cliente en esa fecha y hora.

Cuando se realiza el servicio se imprime una orden de trabajo con los datos generales del cliente y el producto que se ha realizado. Esta orden la llena el operador de acuerdo a los recursos utilizados.

Finalmente, existe una coordinadora de salas que recoge estas órdenes, las revisa y las entrega al departamento de facturación para su respectiva facturación.

Este sistema, está conectado sólo con Facturación para continuar con la secuencia del proceso, pero lo interesante resulta cuando desde cualquier otra área (involucrada en el proceso) se puede consultar la disponibilidad de las salas (horarios ocupados y desocupados), además de que con esta consulta se puede visualizar el tiempo sin utilizar y permitir un mejor aprovechamiento de éste.

ACTUALIZACIONES:

- *Actualización de reservaciones*
- *Actualización de disponibilidad de equipo*

REPORTES Y CONSULTAS:

- *Consultas reservaciones del dia*
- *Consultas reservaciones por cliente*
- *Consultas reservaciones y disponibilidad por fecha*
- *Disponibilidad por sala (Rank, Ursa, Paint-Box, Audio, Ampex, Sony, Copiado, Henry, Harriet y Venice)*
- *Impresión de reservaciones*
- *Impresión de la programación diaria del equipo y las salas*
- *Impresión de ordenes de trabajo*
- *Reporte histórico de reservaciones*

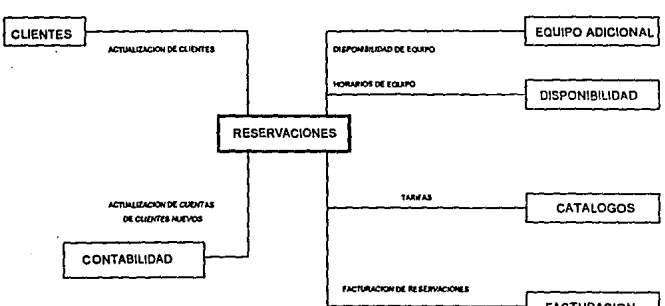


Diagrama de bloques del proceso de reservaciones y su relación con otros procesos.

PROCESO: RELACIONES PUBLICAS

OBJETIVO: *Relaciones públicas se encarga de "enganchar" nuevos clientes, básicamente realizando promocionales y anuncios en revistas del ramo de la producción y post-producción de videos comerciales.*

DESCRIPCION: *Su función principal es promover las ventas a clientes, tanto para clientes que ya han trabajado con la empresa, como para clientes nuevos. Además, de darles un seguimiento cuando ellos están en algún servicio. Se manejan directorios de clientes para enviarles correspondencia o invitaciones a la empresa, o para realizarles visitas de negocios. Además se tienen consultas a los servicios que se proporcionan en las salas para darle el seguimiento requerido. Nosotros, a través de SIACO, apoyamos a esta área proporcionándole los elementos necesarios para ayudarle en su labor de ventas.*

ACTUALIZACIONES:

- *Directorio de clientes*
- *Directorio de ejecutivos de agencias y casas productoras*
- *Directorio de clientes clasificados*
- *Catálogo de líneas de productos que manejan los clientes*
- *Catálogo de productos que manejan los clientes*

CONSULTAS Y REPORTES:

- *Reservaciones de equipo*
- *Paquetes de Servicios*
- *Directorio de clientes*
- *Disponibilidad de equipo a reservar*
- *Impresión de etiquetas para correspondencia a clientes*

- *Productos de clientes*
- *Líneas de producción de los clientes*

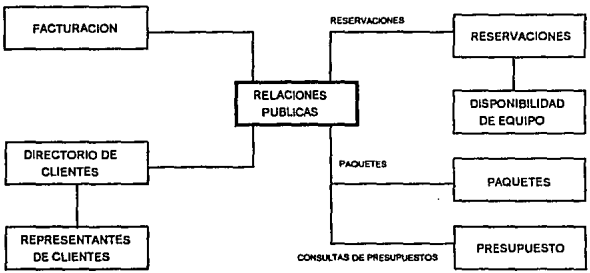


Diagrama de bloques del proceso de relaciones públicas y su relación con otros procesos.

PROCESO: ***ASESORES DE POST-PRODUCCION***

OBJETIVO: *Apoyo a las ventas a través de presupuestar paquetes de servicios atractivos para el cliente.*

DESCRIPCION: *Cuando un cliente tiene un proyecto completo, una forma atractiva para él, puede ser el trabajar a través de los asesores de post-producción de la empresa. Ellos se encargan de estimar un precio final a cada proyecto, en algunos casos se pueden negociar descuentos o mejoras a los precios por proyecto y se tienen que apegar a este (aunque resulte mayor o menor al terminar el proyecto), redundando generalmente, en un mayor beneficio para ambas partes.*

Su labor inicia entrevistando al cliente para conocer las características del proyecto a realizar, éste se captura en el sistema y se emite una carta-presupuesto con todos los detalles del proyecto, incluyendo el precio final. A partir de aquí, se encargan de resolver todos los puntos relacionados con el servicio como lo son: Reservar tiempos, indicaciones a los editores y coordinarse con Facturación para que se contabilicen todas las órdenes de trabajo dentro de un solo paquete, y con ésto, se puede establecer si existe pérdida o ganancia de lo presupuestado.

ACTUALIZACION:

- *Actualización de cotizaciones*
- *Actualización de clientes de cotizaciones*
- *Actualización de representantes de clientes*
- *Actualización de presupuestos*

REPORTES Y CONSULTAS;

- *Carta a los clientes con su presupuesto*
- *Carta a los clientes con su cotización*
- *Consulta a presupuesto*
- *Consulta a disponibilidad de salas*
- *Consulta a paquetes*
- *Consulta estado de cuenta de clientes*
- *Reporte de utilización de equipo*
- *Consulta a reservaciones*
- *Consulta a facturación*

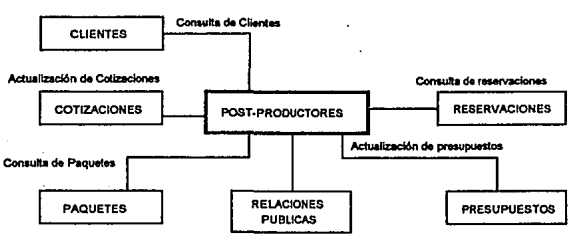


Diagrama de bloques del proceso de post-productores
y su relación con otros procesos

PROCESO: ***PAQUETES DE SERVICIOS***

OBJETIVO: *Control de las órdenes de trabajo que correspondan a un presupuesto para su facturación posterior como paquete.*

DESCRIPCION: *Dentro de los incentivos de ventas que se les ofrecen a los clientes existe el de trabajar todo su proyecto (producción, post-producción y audio) con la empresa, esto es, no realizar por partes su proyecto, para lo cual se les presupuesta un precio total por la elaboración de su proyecto y se les trabaja como paquete, es decir, todos los servicios que utilizan estarán incluidos en el presupuesto con el riesgo de utilizar más o menos del precio por paquete, además de que sólo saldrá una factura por concepto de "Servicios de Post-Producción" y por el importe del precio del paquete, por lo anterior es importante controlar el importe acumulado de lo que se va utilizando para compararlo contra lo presupuestado y poder obtener el margen de ganancia o pérdida.*

El subsistema de Paquetes de Servicios recibe información del subsistema de Reservaciones (órdenes de trabajo ya procesadas) y la envía al subsistema de Facturación.

ACTUALIZACION:

- *Actualización de Paquetes*

CONSULTAS Y REPORTES:

- *Utilizado por paquetes*
- *Consulta a presupuestos*
- *Relación de lo utilizado en paquetes contra importes facturados*
- *Utilización por salas*
- *Utilización por clientes*

- *Utilización por producto*
- *Utilización por orden de trabajo*
- *Estado de cuenta de paquetes*

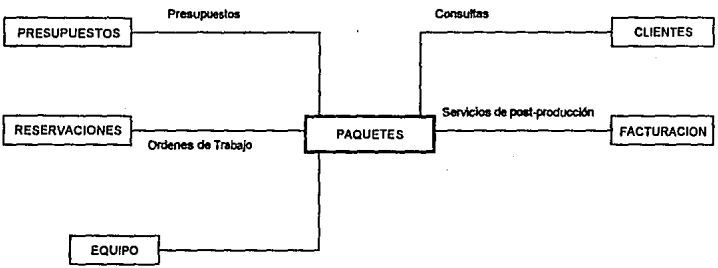


Diagrama de bloques del proceso de paquetes
y su relación con otros procesos.

PROCESO: PLAN FRANCES

OBJETIVO: *Controlar los paquetes que trabajan dentro del plan de financiamiento llamado "Plan Francés".*

DESCRIPCION: *Existe una variante dentro del manejo de paquetes que es el de "Plan Francés", el cual es un plan de financiamiento con las siguientes caracteristicas:*

En este plan, se maneja un depósito monetario por parte del cliente equivalente al monto que trabajará en un periodo determinado de tiempo, por consecuencia obtendrá beneficios al realizar sus trabajos dentro de la empresa, como lo son: precios especiales, preferencias al reservar sus tiempos, descuentos y asesorías entre otros.
La comunicación de este subsistema se realiza con el subsistema de Paquetes y con el subsistema de Facturación y puede elaborar desde un solo paquete a muchos, lo que alcance a cubrir con lo presupuestado en el tiempo fijado.

De igual manera que el subsistema de paquetes, es importante controlar estados de cuentas de los clientes que están dentro del Plan Francés.

ACTUALIZACIONES:

- *Actualización de Plan Francés*
- *Actualización de utilización de paquetes en Plan Francés*

REPORTES Y CONSULTAS:

- *Reporte de utilización en paquetes de Plan Francés*
- *Reporte de clientes en Plan Francés*
- *Estados de cuenta de Plan Francés*

- *Cierre de paquetes*
- *Cierre de Plan Francés*

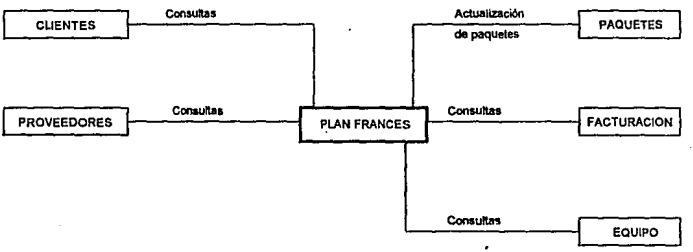


Diagrama de bloques del proceso del plan *francés* y su relación con otros procesos.

4.1.1 PLANEACION; Viabilidad del proyecto en relación al costo y a las restricciones de tiempo.

Después de conocer un panorama general del sistema "SIACO", empezaremos a estudiar la aplicación del método propuesto al subsistema de Reservaciones de salas de post-producción.

Analizaremos a continuación el procedimiento y las actividades del departamento de Programación de Servicios de Salas.

Como se mencionó en el capítulo 1, Qualli es un Centro de Servicios Integrados de Producción, Post-producción y Audio, en donde su principal ingreso es la renta de salas con equipo para el manejo y edición de video.

Estas salas se rentan al cliente que las solicite (con un tiempo de anticipación) y están equipadas con una plataforma base de equipo instalado, pudiendo incrementarse a petición del cliente, este acondicionamiento adicional se realiza a través de conectar o instalar equipo adicional (según lo necesite el proyecto a realizar por el cliente) a través de una sala de control central o master.

Por otra parte, como se mencionó anteriormente la fase de la definición empieza con la etapa de planeación del software. Durante esta etapa se desarrolla una descripción bien delimitada del alcance del software, se definen los recursos necesarios para desarrollar el software; se establecen las estimaciones de tiempo y costo. El propósito de esta etapa es proveer una indicación preliminar de la viabilidad del proyecto en relación al costo y a las restricciones de tiempo que se hayan establecido para llevar acabo un buen proyecto de desarrollo de software, debemos comprender el ámbito del trabajo a realizar, los recursos

requeridos, las tareas a ejecutar, las referencias a tener en cuenta, el costo a emplear y la agenda a seguir.

El objetivo de la planeación se alcanza a través de un proceso de descubrimiento de información que lleve a estimaciones razonables.

A continuación se trata cada una de las actividades asociadas con la planeación del proyecto.

Alcances del sistema: En primer lugar es importante definir los objetivos y funciones del sistema a realizar en una primer entrevista con el usuario, se realiza una investigación del ambiente o medio en que va a operar, las necesidades y/o problemas que se pretenden solucionar y se trata de limitar los alcances, esto es muy importante ya que normalmente los objetivos estarán planteados de acuerdo a estos límites, ocurre normalmente, que el usuario no visualiza funciones a futuro, sino hasta que al avanzar la definición se tiene una idea más clara de todo y entonces se sugiere ampliar el proyecto a nuevas áreas, esto repercute en una redefinición de los objetivos planteados inicialmente. Esta caraterística la soporta perfectamente el modelo propuesto ya que como se realiza una interacción entre la fase de definición y la de desarrollo se puede presentar el caso de la redefinición de los objetivos pero sin ningún efecto negativo, sino por el contrario, se refuerza la fase de definición quedando mejor determinado el sistema que se desea.

Como resultado de las primeras entrevistas se debe redactar un informe como se mostrará a continuación, en donde se describan los objetivos y el flujo del proceso, en las siguientes entrevistas este informe será más detallado y más completo por lo que no hay que preocuparse si los primeros informes resultan muy informales.

PROYECTO:	SISTEMA DE CONTROL DE RESERVACIONES
FASE:	DEFINICION
ETAPA:	1er. ENTREVISTA
USUARIO:	ING. RAFAEL MOLINA
FECHA ENTREVISTA:	01/NOVIEMBRE/93
RESPONSABLE DEL PROYECTO:	ARMANDO HERNANDEZ DEL CASTILLO

DESCRIPCION DEL PROCESO: El departamento de programación es el primer contacto (telefónico normalmente) que tiene el cliente con la empresa. Como el principal ingreso a la compañía es la renta de salas (equipo y operadores) de Post-Producción, Producción y Audio, es necesario que el cliente reserve estas salas (con un tiempo razonable de anticipación) para realizar su trabajo.
En primer instancia el cliente llama por teléfono y solicita una sala determinada para una fecha deseada, el depto. de Programación verifica la disponibilidad de dicha sala y le resuelve favorablemente en caso de no estar ocupada, solicitándole sus datos generales y los del servicio a realizar (producto y versión) dando por terminado este primer contacto con el cliente, indicándole también que confirme su Reservación 24 horas antes de la fecha requerida.

En caso de no encontrarse disponible la sala ,le presenta otras alternativas para otra fecha, para otra hora o para realizar su trabajo en una sala equivalente que le puede resolver su problema.

En caso de aceptar alguna de estas alternativas, se realiza el proceso de cierre de la Reservación.

Al terminar se genera una orden de trabajo que contiene todos los datos del equipo utilizado, así como, los datos generales del cliente y que son confirmados en ese momento.

Al llegar la fecha del servicio se atiende al cliente indicandole la sala y operador que están listos para su servicio.

Al terminar el servicio se actualiza la orden de trabajo con los tiempos reales utilizados y se envía la orden de trabajo al departamento de Facturación, para que elabore su factura.

PROCESOS ADICIONALES:

Se imprime la programación diaria por Sala.

Se imprime la orden de trabajo.

PROYECTO:	SISTEMA DE CONTROL DE RESERVACIONES
FASE:	DEFINICION
ETAPA:	CARACTERISTICAS DEL SOFTWARE
USUARIO:	ING. RAFAEL MOLINA
FECHA ENTREVISTA:	01/NOVIEMBRE/93
RESPONSABLE DEL PROYECTO:	ARMANDO HERNANDEZ DEL CASTILLO

Se requiere un Software que trabaje en modo caracter y se divida de la siguiente manera:

- Introducción de datos (características del "ABC"). En primer lugar se capturan los datos generales del cliente , del servicio (producto y versión del proyecto) y datos de quienes participaron en la reservación.

- Consulta a la disponibilidad de equipo solicitado (sala, fecha y equipo adicional).

- En caso de estar disponible se permite la captura de la sala y del equipo adicional.

- Se deben de asociar las salas con máquinas adicionales y se debe permitir intercambiar estas máquinas con otras salas.

- Al reservar las salas y el equipo adicional, se debe de marcar como no disponible para esa fecha y hora, así como la de bloquear todos los servicios restantes para dicha sala.

PROYECTO:	SISTEMA DE CONTROL DE RESERVACIONES
FASE:	DEFINICION
ETAPA:	REQUERIMIENTOS DE EQUIPO
USUARIO:	ING. RAFAEL MOLINA
FECHA ENTREVISTA:	01/NOVIEMBRE/93
RESPONSABLE DEL PROYECTO:	ARMANDO HERNANDEZ DEL CASTILLO

Se plantean los siguientes puntos para la red de cómputo para cubrir las necesidades indicadas en el proceso:

- Respuesta rápida.
- 3 usuarios como máximo.
- Comunicación en línea con la base de datos.
- Ambiente en modo caracter o modo gráfico.

Por lo cual se plantea:

- Incorporarlos a la red de área local de la empresa (instalación de líneas y software de comunicación.).
- Asignarles 3 estaciones de trabajo.
- Una impresora laser.

Requerimientos de equipo:

El sistema "SIACO" está desarrollado para trabajar sobre un ambiente multiusuario en una red local de microcomputadoras, por lo tanto, el analizar los requerimientos de equipo cuando se trata de automatizar a un nuevo departamento dentro de la empresa en la mayoría de los casos la inversión se reduce a comprar computadoras personales, tarjetas de red, cablear hasta el área de trabajo y comprar periféricos (en algunos casos).

Para lo cual, no es necesario elaborar un estudio muy complejo de gasto para inversión en equipo, a continuación mostraremos un formato en el cual se ejemplifica un estudio de gasto para inversión en equipo.

Al final del análisis de equipo se presenta un resumen con la evaluación y con la propuesta final, considerando el área de trabajo y dado que estamos trabajando con una red local es importante considerar también, el cableado hasta esta área y en caso de ser necesario la instalación de distribuidores o repetidores para amplificar la señal y cubrir la distancia.

REQUERIMIENTO DE EQUIPO

ANALISIS DE INVERSION

PROYECTO: _____ **AREA:** _____

CONTRA PRESUPUESTO DEL AREA: _____ **RESPONSABLE DEL PROYECTO:** _____

CARACTERISTICAS DEL EQUIPO		PROVEEDOR 1		PROVEEDOR 2		PROVEEDOR 3		OBSERVACIONES
	CANTIDAD	PRECIO UNITARIO	TOTAL	PRECIO UNITARIO	TOTAL	PRECIO UNITARIO	TOTAL	
IMPORTE TOTAL								
CALIFICACION POR PROVEEDOR (DEL 1 AL 10)								

REQUERIMIENTO DE EQUIPO

CARACTERISTICAS DE VENTA POR PROVEEDOR

PROYECTO: ____________________ **AREA:** ____________________

CONTRA PRESUPUESTO DEL AREA: ____________________ **RESPONSABLE DEL PROYECTO:** ____________________

PROVEEDOR: ____________________
CONDICIONES DE PAGOS: ____________________
TIEMPO DE ENTREGA: ____________________
CREDITO: ____________________
SOPORTE POST-VENTA: ____________________
GARANTIA: ____________________
OTROS: ____________________
____

CALIFICACION POR PROVEEDOR: ____________________

PROVEEDOR: ____________________
CONDICIONES DE PAGOS: ____________________
TIEMPO DE ENTREGA: ____________________
CREDITO: ____________________
SOPORTE POST-VENTA: ____________________
GARANTIA: ____________________
OTROS: ____________________
____

CALIFICACION POR PROVEEDOR: ____________________

PROVEEDOR: ____________________
CONDICIONES DE PAGOS: ____________________
TIEMPO DE ENTREGA: ____________________
CREDITO: ____________________
SOPORTE POST-VENTA: ____________________
GARANTIA: ____________________
OTROS: ____________________
____

CALIFICACION POR PROVEEDOR: ____________________

Requerimientos de personal:

Para esta etapa podemos dividir en dos tipos·el personal requerido para el proyecto, el personal técnico en informática que desarrollará el proyecto y el personal operativo que son los usuarios que operarán el nuevo sistema. Normalmente, tenemos que trabajar con el personal de planta con que se cuenta, ya que es muy difícil (por el tamaño de la empresa) aumentar este número de personas dedicadas a esta tarea, ya que sólo se puede justificar y lograr una autorización de contratar más personal si existen otros proyectos con esta necesidad y que apoyaran este requerimiento o que la rapidez en tener funcionando el nuevo sistema fuera de gran prioridad, pero en los casos más comunes se trabaja con lo que se tiene, redundando esto en un incremento en el tiempo para terminar el proyecto.

Se debe evaluar la especialidad de cada persona y su disponibilidad en tiempo para participar en el proyecto, en base a esto, se distribuyen las cargas de trabajo y se controlan con la forma Q1 descrita posteriormente a detalle. Es necesario mencionar que el departamento de Sistemas de la empresa cuenta con el siguiente personal para la definición, desarrollo, mantenimiento y pruebas del software que aquí se implementa:

- Gerente de sistemas: Entre otras funciones están la de participar activamente en las etapas de definición y desarrollo de los sistemas, así como la de observador en la administración de los mismos.

- Jefe de sistemas: Normalmente es quien soporta la responsabilidad del desarrollo de todo el proceso de la elaboración de los sistemas y su respectivo mantenimiento.

- Ingeniero de sistemas : Participa activamente en las fases de desarrollo y mantenimiento de los sistemas.

- Analista de sistemas: Es quien elabora los manuales y recibe las solicitudes de mejoras a los sistemas, así como de auxiliar al gerente cuando un sistema se encuentra en sus fases iniciales.

Para nuestro caso en estudio se tienen los siguientes requerimientos y asignaciones de personal:

PROYECTO:	SISTEMA DE CONTROL DE RESERVACIONES
FASE:	DEFINICION
ETAPA:	REQ. DE PERSONAL Y DEFINICION DEL AREA DE TRABAJO
USUARIO:	ING. RAFAEL MOLINA
FECHA ENTREVISTA:	01/NOVIEMBRE/93
RESPONSABLE DEL PROYECTO:	ARMANDO HERNANDEZ DEL CASTILLO

PERSONAL QUE DESARROLLA EL SISTEMA:

Se trabajará sólo con el personal de planta con que cuenta y que es:

- Gerente
- Jefe de Sistemas
- Ing. de Sistemas
- Analista

USUARIOS:

Se tiene 3 usuarios que son expertos en el área de Reservaciones, de los cuales 2 personas tienen conocimientos de computación y 1 con conocimiento a nivel de manejo de procesador de palabras.

AREA DE TRABAJO:

En el segundo piso, en la oficina de Programación, se localiza un registro para cableado de la red general, lo que facilita el cableado.

Las estaciones de trabajo se instalarán sobre las mesas de trabajo de cada persona.

Se comparte una impresora Laserjet con el Departamento de Post-Producción.

PASO	NUM. PERS.	PUESTO	ETAPA	OBSERVACIONES
1	1	- GERENTE	1a. ENTREVISTA CON EL USUARIO	DEFINIR ALCANCE DEL SOFTWARE
2	1	- GERENTE	2a. ENTREVISTA	DETALLAR OBJETIVOS Y FUNCIONES DEL SISTEMA
3	1	- GERENTE	3a. ENTREVISTA	DEFINIR FLUJO DE INFORMACION Y REQUER DE EQUIPO
4	3	- ING. SISTEMAS - JEFE SISTEMAS - GERENTE	DEFINICION	MESA DE TRABAJO DE REVISION DE LOS PUNTOS ANTERIORES
5	3	- ING. SISTEMAS - JEFE SISTEMAS - GERENTE	DESARROLLO	DISEÑO DE BASE DE DATOS
6	2	- ING. SISTEMAS - JEFE SISTEMAS	DESARROLLO	ELABORACION DEL 1er. PROTOTIPO
7	1	- JEFE SISTEMAS	DEFINICION	REVISION DEL PROTO-TIPO Y REDIFINICION DEL SISTEMA
8	1	- JEFE SISTEMAS	DESARROLLO DEFINICION	PROCESO ITERATIVO HASTA QUE QUEDE COMPLETAMENTE DETERMINADO EL PROCESO
9	2	- ING. SISTEMAS - JEFE SISTEMAS	DESARROLLO	REFINAMIENTO DEL PROTOTIPO HASTA TENER UN PRODUCTO TERMINADO
10	1	- GERENTE	DESARROLLO	DETERMINAR LOS DEMAS PROTOTIPOS DEL SISTEMA
11	2	- ING. SISTEMAS - JEFE SISTEMAS	DESARROLLO DEFINICION	REPETIR DEL PASO 6 AL 9 POR CADA PRO-TOTIPO
12	1	- ANALISTA DE SISTEMAS	DOCUMENTACION	ELABORAR EN BASE A APUNTES LA DOCU-MENTACION DEL SISTEMA
13	1	- ING. SISTEMAS	MANTENIMIENTO	EFECTUAR EL MANTENIMIENTO DE: - CORRECCION - ADAPTACION - AUMENTO

TABLA DE ACTIVIDADES DEL PERSONAL DE SISTEMAS

Planeación de tiempos y actividades:

Hasta esta etapa, estamos en posibilidad junto con el usuario, de elaborar un plan de tiempos y actividades el cual nos permitirá establecer un orden cronológico de las actividades y controlar su avance o su retrazo, así como reprogramar actividades nuevas o atrazadas, para esto nos apoyamos en la forma Q1 la cual nos permite visualizar de una manera muy objetiva el control de las actividades.

Para llenar esta forma se procede de la siguiente manera:

Se listan todas las actividades por realizar, no importando que queden en desorden.
Para nuestro caso en estudio sería de la siguiente manera:

- FASE DE DEFINICION

1 .- 1a. entrevista con el usuario (etapa de alcances del sistema).
2 .- 2a. entrevista con el usuario (mostrar informe escrito).
3 .- Requerimientos de equipo.
4 .- Requerimientos de personal.
5 .- Definición del área de trabajo.
6 .- Análisis de sistemas.
7 .- Análisis de requerimientos.
8 .- Revisión del informe final.
9 .- Evaluación del equipo.
10.- Compra del equipo

- FASE DE DESARROLLO

11.- Diseño de la base de datos.

12.- Diseño procedimental.

13.- Desarrollo del primer prototipo del tipo "ABC".

14.- Presentación al usuario del "ABC".

15.- Redefinición de objetivos en base al "ABC".

16.- Refinamiento del "ABC".

17.- Proceso iterativo hasta lograr un producto terminado.

18.- Acondicionamiento del área de trabajo e instalación del equipo y cableado.

19.- Entrevistas con el usuario para plantear el uso de los prototipos de consultas, reportes y/o estadisticas.

20.- Desarrollo de los prototipos en un proceso interactivo con el usuario.

21.- Revisión y prueba de los prototipos terminados.

22.- Puesta en marcha del sistema.

23.- Adiestramiento del personal.

- FASE DE MANTENIMIENTO

24.- Mantenimiento correctivo.

25.- Mantenimiento adaptativo.

26.- Mantenimiento de aumento o perfectivo.

A continuación se les asignará un número consecutivo a cada actividad de acuerdo a su prioridad, para nuestro ejemplo las actividades ya quedaron en forma ordenada, por lo que procedemos a llenar el formato Q1 vaciando la lista de actividades, se sugiere dejar renglones en blanco entre cada una de ellas para permitir adicionar alguna actividad nueva en caso de presentarse.

Finalmente se asignarán tiempos sombreando los cuadros de las fechas asignadas de acuerdo con el grado de dificultad estimado para cada una de las actividades. Con el paso del tiempo y al ir dando por terminadas estas actividades se tachan con un color distinto al de la fecha de la actividad que se terminó. En caso de no cumplir con la actividad por algún retrazo se puede proceder de dos maneras: Primero, asignarle más días a la actividad y aumentárselo en la misma proporción a todas las demás, indicando y marcando con diferente color o borrando de preferencia con la misma proporción los primeros días marcados para saber que esos primeros días no cuentan dentro del tiempo asignado a la actividad (o que representan un retrazo). Segundo, reprogramar la actividad con retrazo al final de la lista asignándole nuevos tiempos.

Como se puede observar en la forma Q1 el sombreado en los días va quedando de manera escalonada y al manejar las fechas de las actividades terminadas con un color y los retrazos con otro color, es posible ubicar fácilmente el avance y/o retrazo del proyecto. Se sugiere además manejar tiempos de holgura, es decir, en algunas actividades se pueden asignar uno o dos días de holgura para reajustar el tiempo en caso de ser necesario.

NOVIEMBRE

PROYECTO: RESERVACIONES *1/3*

ACTIVIDADES / PROCESO	1A. SEMANA					2A. SEMANA					3A. SEMANA					4A. SEMANA					5A. SEMANA				
	1	2	3	4	5	8	9	10	11	12	15	16	17	18	19	22	23	24	25	26	29	30			
FASE DE DEFINICION																									
ENTREVISTA USUARIO																									
(ALCANCE DEL SISTEMA)																									
ENTREVISTA USUARIO																									
MOSTRAR Y REV.INFORME																									
REQUERIMIENTOS EQUIPO																									
REQUERIM. PERSONAL																									
DEFINICION DEL AREA																									
DE TRABAJO																									
ANALISIS DE SISTEMAS																									
ANALISIS DE REQUERIMIENTOS																									
REVISION INFORME FINAL																									
EVALUACION DEL EQUIPO																									
COMPRA DEL EQUIPO																									

ASIGNADO A:

NOVIEMBRE

PROYECTO: RESERVACIONES *2/3*

ACTIVIDADES	1A. SEMANA					2A. SEMANA					3A. SEMANA					4A. SEMANA					5A. SEMANA				
PROCESO	1	2	3	4	5	8	9	10	11	12	15	16	17	18	19	22	23	24	25	26	29	30			
FASE DE DESARROLLO																									
DISEÑO DE BASE DE DATOS																									
DISEÑO PROCEDIMENTAL																									
DESARROLLO DEL PRIMER																									
PROTOTIPO DEL "ABC"																									
PRESENTACION DEL"ABC"																									
REDEFINICION DE OBJETIVOS																									
REFINAMIENTO DEL"ABC"																									
PROCESO ITERATIVO																									
ACONDICIONAM.DEL AREA																									
CABLEADO																									
INSTALACION DE EQUIPO																									
ENTREVISTA USUARIO																									
REV PROTOTIPOS DE																									
CONSULTAS, REPORTES																									
ESTADISTICAS																									

ASIGNADO A:

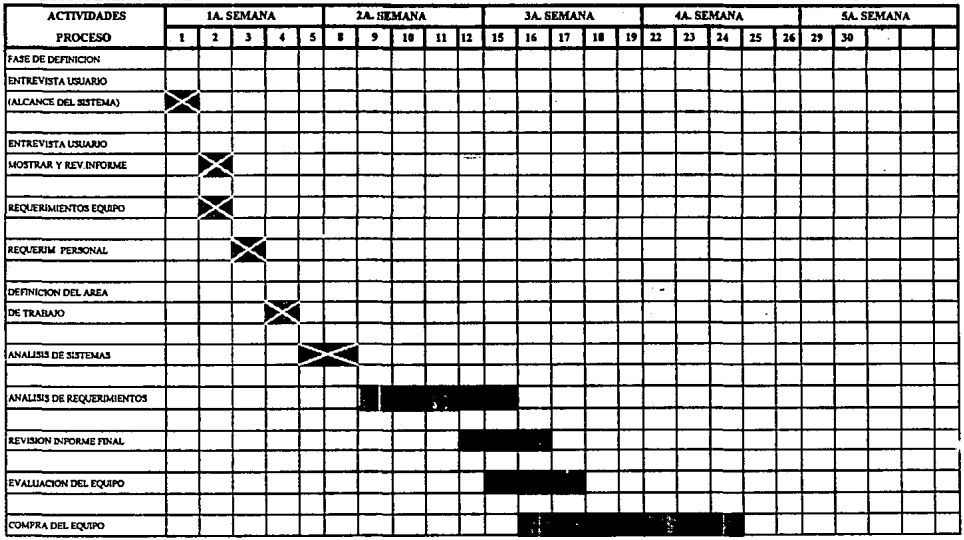
DICIEMBRE

PROYECTO: RESERVACIONES *3/3*

ACTIVIDADES	1A. SEMANA					2A. SEMANA					3A. SEMANA					4A. SEMANA					5A. SEMANA				
PROCESO			1	2	3	6	7	8	9	10	13	14	15	16	17	20	21	22	23	24	27	28	29	30	31
DESARROLLO DE PROTOTIPOS																									
PROCESO ITERATIVO																									
REVISION Y PRUEBAS																									
PUESTA EN MARCHA																									
ADIESTRAMIENTO DEL PERSONAL																									
FASE DE MANTENIMIENTO																									
MANT. DE CORRECCION																	ABIERTO								
MANT. DE ADAPTACION																	ABIERTO								
MANT. DE AUMENTO																	ABIERTO								

ASIGNADO A:

PROYECTO: RESERVACIONES *1/3*



EJEMPLO DE COMO SE DEBEN DE INDICAR LAS ACTIVIDADES TERMINADAS, CON RETRAZO Y SIN CONCLUIR.

Finalmente dentro de la fase de definición se termina el trabajo realizado en papel generando un informe escrito lo más detallado posible con toda la información anterior y el cual servirá de base para las fases posteriores.

Se definen los objetivos y alcances del nuevo software, dominio de la información, relaciones, etc., y se anexan todos los formatos anteriores.

De ser posible se elabora un diagrama de flujo de la información con todos sus elementos participantes.

4.2 FASE DE DESARROLLO DEL SISTEMA

Como ya hemos mencionado la fase de desarrollo traduce un conjunto de requerimientos en el elemento de sistema operacional que llamamos software, y de acuerdo a nuestro método propuesto, aquí diseñaremos la base de datos para poder iniciar el desarrollo del prototipo "ABC", que además nos ayudará a replantear la etapa de definición.

4.2.1 .- DISEÑO; Traslación de los requerimientos del software a un conjunto de representaciones que describen la estructura de datos, arquitectura y procedimiento algorítmico.

Diseño de datos: Como se mencionó anteriormente, el diseño de datos en nuestro caso, corresponde al diseño de la base de datos.
Para el diseño de la base de datos es necesario seguir en contacto con el usuario. Es esencial la identificación de las funciones e interfaces, se requiere la especificación del flujo, estructura, y asocialividad de la información y debe desarrollarse un documento

formal de los requerimientos.
Un tratamiento completo del estudio de bases de datos va más allá del ámbito de este documento, por lo que solo mencionaremos algunas de sus partes escenciales.
Podemos decir que una base de datos es:

Una colección de información organizada de forma que facilita el acceso, análisis y creación de informes. Una base de datos contiene entidades de información que están relacionadas vía organización y asociación. La arquitectura lógica de una base de datos se define mediante un esquema que representa las definiciones de las relaciones entre las entidades de información, la arquitectura física de una base de datos depende de la configuración del hardware residente, en el mercado existen disponibles un gran número de sistemas manejadores de bases de datos (DBMS). Estos sistemas contienen distintos lenguajes de preguntas (QUERY) para la manipulación y acceso a los datos, lenguajes de cuarta generación para desarrollo de aplicaciones y software para el manejo de archivos.

Para nuestro caso en estudio, como ya lo hemos mencionado, utilizamos el manejador de bases de datos DataFlex, de la compañía estadounidense Data Access Corp.. Hemos mencionado también, que la elección de DataFlex tiene la justificante principal de que ya existía un sistema desarrollado (por una compañía externa) operando en la empresa con dicho lenguaje, y que además teníamos que trabajar con la base técnica instalada en el área de sistemas, sin embargo, decidimos desarrollar el nuevo sistema SIACO con DataFlex por las siguientes razones:

Antes de empezar a trabajar en el nuevo proyecto, quisimos tener la seguridad (teníamos la obligación) de que el sistema no se volvería obsoleto en corto tiempo, y de que el manejador fuera lo suficientemente bueno como para permitir la construcción del nuevo diseño con las características suficientes como para no tenerlo que desechar

tempranamente, y además, que existiera también el soporte técnico adecuado por parte de la compañía que lo desarrolló (continuas mejoras y actualizaciones del producto) y de quién lo distribuye en nuestro país (porque ésto nos facilitaría la comunicación y el apoyo).

Por otra parte, podemos decir con toda seguridad, que para desarrollar SIACO tal vez existan en la actualidad, un sin número de manejadores de bases de datos con los que se podría implementar, siempre y cuando cumplan con las características demandadas por el diseño del mismo; DataFlex al menos cumple con la gran mayoría. Por ejemplo, podríamos citar algunas de las principales características que tomamos en cuenta en la elección:

- *Que sea un DBMS.*
- *Que el DBMS pueda manejar múltiples bases de datos simultáneamente.*
- *Que el DBMS maneje una base de datos 100% relacional.*
- *Que el DBMS tenga la posibilidad de construir y manejar índices relativos a cada una de las relaciones que se necesiten y que su funcionamiento sea totalmente automático y transparente para el programador.*
- *Que el DBMS pueda construir ligas internas o vínculos entre las relaciones de la base de datos.*
- *Que el DBMS sea capaz de dar mantenimiento a las relaciones, índices y vínculos construidos.*
- *Que el tamaño de las relaciones pueda crecer al menos 15 MB sin tener problemas con el tiempo de acceso a la misma.*
- *Que tenga un sistema de consultas en línea a la base de datos.*
- *Que tenga un sistema generador de código para reportes de la base de datos.*
- *Que tenga un lenguaje propio para la explotación de la base de datos.*

- *Que el lenguaje permita la facilidad de traducción del diseño detallado al código.*

- *Que el lenguaje pueda trabajar con archivos ASCII.*

- *Que el lenguaje tenga la posibilidad de manejar imágenes o pantallas como una estructura natural del mismo.*

- *Que permita el manejo y control de dichas pantalllas con mecanismos debidamente desarrollados para ello.*

- *Que el lenguaje tenga estructuras para el control estructurado y no estructurado.*

- *Que el lenguaje tenga las estructuras lógicas necesarias para poder realizar operaciones relacionales.*

- *Que el lenguaje tenga un amplio conjunto de tipos de datos y operaciones para manejarlos. Por ejemplo datos numéricos, enteros, booleanos, de cadenas de de caracteres. Así como la de diferentes precisiones para el manejo operaciones numéricas y científicas.*

- *Que el lenguaje tenga la posibilidad de manejar gráficas de dos dimensiones.*

- *Que el lenguaje tenga la posibilidad de trabajar en modo multiusuario, con los debidos comandos para poder lograr dicho control.*

- *Que el lenguaje permita la comunicación entre distintos módulos del sistema desarrollado. Como lo son el paso de parámetros, dentro y fuera de un programa. Es decir, que permita la construcción de subprogramas.*

- *Que la base de datos y los programas desarrollados puedan transportarse sin problemas y/o demasiados ajustes a otro sistemas operativos y/o arquitecturas de cómputo.*

- *Que el producto tenga una amplia disponibilidad de herramientas de desarrollo. Por ejemplo: editores, compiladores, bibliotecas de subprogramas, herramientas para el control del código fuente y para el control de la base de datos.*

- *Que el producto facilite el mantenimiento tanto del código fuente como el de la base de datos.*

- *Que el producto sea diseñado para trabajar en el ambiente de las computadoras personales y minicomputadoras.*

- *Que el producto pueda trabajar en el ambiente de las redes locales ethernet, principalmente con el software Netware de Novell.*

Según la investigación que realizamos (ver Apéndice A), concluimos que DataFlex cumple con casi todas éstas características y más. DataFlex no tiene un lenguaje estructurado que nos permita realizar operaciones del álgebra relacional de una manera tan teórica, pero se pueden construir con el mismo lenguaje estructurado. DataFlex no tiene la manera de construir restricciones a la base de datos para asegurar la consistencia de los datos, pero se pueden construir con su propio lenguaje, sin embargo, ésto implica ciertos cuidados, ya que el código solo actúa en el módulo implicado y no en toda las demás aplicaciones. A pesar de éstos detalles DataFlex sigue siendo un buen manejador de base de datos y que contempla las principales características demandadas por el sistema a desarrollar. Por éstas razones decidimos que íbamos por el camino adecuado y así continuamos con el desarrollo de SIACO sin temor a equivocarnos.

Con la información obtenida en la fase de definición del sistema definimos un modelo de información el cual incluye un diccionario de datos que define todos los elementos de datos en términos de la información que se utiliza para desarrollar cada elemento.

Modelo Conceptual.

Para desarrollar una base de datos que satisfaga las necesidades de información presentes y futuras, se debe diseñar un modelo conceptual. Este modelo refleja las entidades y sus relaciones y está basado en las necesidades de la organización de procesamiento de datos. Cuando se determinan las entidades y sus relaciones es necesario hacer un análisis de datos. Este análisis puede basarse en la información sobre los datos, tanto para aplicaciones existentes como futuras.

La principal responsabilidad de un administrador de la base de datos es diseñar un "Modelo Conceptual" el cual debe representar a las entidades de la empresa y sus relaciones entre ellas.

Cuando se diseña el modelo conceptual, se debe poner especial empeño en la estructura de datos y de las relaciones entre los campos. Hasta este momento no debe haber relación con las fases de realización y operación de la base de datos.

El primer paso en el diseño del modelo conceptual es el análisis de los datos, el cual proporcionará la información sobre los campos de datos y las relaciones entre ellos.

Se debe usar un cuestionario o un informe similar para obtener de cada nivel administrativo (ejecutivo, funcional y operacional) una lista compuesta de los datos que se necesitan, el cuestionario debe requerir la sisguiente información:

1.- Nombre de las entidades y su descripción. Dar una lista del nombre y de cualquier sinónimo con los cuales se haga referencia a los campos de datos, se debe dar una descripción de lo que significa el nombre aunque el nombre aparentemente se defina o explique por sí mismo. Describir en forma general el uso o función de la entidad, el propósito al que sirve dentro de la unidad operativa o funcional de la unidad y todos los usos fuera de ésta.

2.- Campos de datos. Para cada elemento de información en la entidad, proporcionar la siguiente información:

a) Nombre del campo y descripción. Enumerar los nombres, siglas y nemotécnicos. Dar una descripción completa del elemento.

b) Fuentes de datos. Enumerar la(s) fuente(s) de origen según la unidad funcional y operativa. Los ejemplos incluyen campos de entrada como éstos: del cliente, de los memorandums entre oficinas y del departamento de expedición.

c) Atributos del campo. Enumerar atributos numéricos, alfabéticos y textuales. Dar la unidad de medida asociada con los datos, tales como piezas, dólares y pies. Si existen límites a los rangos de valores aceptables del elemento, enumerarlos (por ejemplo, el valor no puede ser menor de 100 ni mayor de 500).

d) Uso del campo. Describir el uso. Los ejemplos incluyen lo siguiente: para proporcionar información de direccionamiento, para determinar la cantidad, para establecer el nivel de nóminas.

e) Seguridad/sensibilidad del campo. Enumerar todas las limitaciones asociadas con el nombre del campo. Estas limitaciones están generalmente relacionadas con restricciones al público, es decir, quien puede usar, accesar, leer y/o divulgar los datos.

f) Importancia/valor. Mencionar la importancia de los datos, ¿ Qué valor tiene en términos de la continuación o expansión del funcionamiento de la empresa? No tiene sentido fijar el valor de los datos de una manera negativa (esto es, "No podríamos trabajar sin ellos"). De hecho le corresponde al punto relativo al uso del elemento el dar base y argumentar el valor del mismo.

g) Relación(es) del campo. Enumerar las formas en cuales este campo se usa o relaciona con otros campos. Estos campos no necesitan estar limitados a la entidad específica que se discute. Tales relaciones: Número de reservación/producto/versión/fecha de reservación, número de sala/descripción, número de unidad de renta/descripción/precio.

3.- Criterio de retención y almacenamiento. Describir la cantidad de tiempo y la forma en que los datos se guardan. Mencionar también, si se conoce la razón o causa de la retención (por ejemplo; regulación gubernamental, política de la compañía).

El modelo básico que representa a las entidades de una empresa y las relaciones entre ellas es más estable que las diferentes formas en las que se recupera la información almacenada en una base de datos. Una manera efectiva de desarrollar un modelo, al cual llamaremos modelo conceptual, consiste en aplicar los conceptos del modelo de datos relacional. Estos conceptos se aplican al análisis de los datos y a la información de las relaciones proporcionada por los usuarios finales.

El concepto principal, tomado del modelo relacional utilizado en el desarrollo del modelo conceptual, es el proceso de normalización, esto es, el proceso de agrupar a los campos de datos en tablas que representan a las entidades y sus relaciones. La teoría de la normalización está basada en la observación de que un cierto conjunto de relaciones tiene mejores propiedades en un medio de inserción, actualización y supresión que las que tendrían otros conjuntos de relaciones conteniendo los mismos datos.

La razón de usar el procedimiento de normalización es asegurar que el modelo conceptual de la base de datos funcionará. Esto no significa que una estructura no normalizada no funcionará , sino que puede causar algunos problemas cuando los programadores traten de modificar la base de datos. El administrador de la base de datos debe decidir, después de localizar las violaciones provenientes de la normalización, si las modificaciones afectarán la forma en la que la base de datos funcionará.

Un modelo de datos no normalizado consiste en registros utilizados por los programas de aplicación. El primer paso de la normalización consiste en transformar los campos de

datos a una tabla de dos dimensiones. Lo que se requiere en este paso es la eliminación de ocurrencias repetidas de campos de datos, de tal manera que se obtenga un archivo fijo. Por ejemplo, si una declaración incluye espacio para el nombre de un empleado, su número, esposa y hasta diez hijos, el resultado será una tabla de 4x10, con cuatro columnas y diez renglones. Cada uno de los renglones tendrá el nombre y número del empleado, esposa y el nombre de cada uno de los hijos. Los diez renglones tendrán los nombres de los diez hijos. Este es solo un paso preliminar que hace posible trasladarse a la segunda forma normalizada.

El segundo paso de la normalización es establecer las claves y relacionarlas con los campos de datos. En la primera forma normalizada, el renglón entero de la tabla (cadena) depende de todos los campos de claves. En la segunda forma normalizada, se hace un intento de establecer los campos de datos que están relacionados con alguna parte de la clave completa. Si los campos de datos sólo dependen de una parte de la clave, la clave y los campos conectados a la clave parcial son suceptibles de separarse en registros independientes. La división de la primera tabla normalizada, en una serie de tablas en las que cada campo sólo depende de la clave completa, se llama segunda forma normalizada.
El tercer paso consiste en separar los campos de las segundas relaciones normales que, aunque dependan sólo de una clave, debe tener una existencia independiente en la base de datos. Esto se hace de forma tal que la información sobre estos campos pueda introducirse separademente a partir de las relaciones en las que se encuentra implicada.

Resumiendo. El proceso de normalización identifica los datos redundantes que pueden existir en la estructura lógica, determina claves únicas necesarias para el acceso a los elementos de datos y ayuda a establecer las relaciones necesarias entre estos, pueden aplicarse tres niveles de normalización, llamados formas normales, a continuación ejemplificamos lo anterior para nuestro caso en estudio.

Análisis de requisitos. Proceso: Reservaciones de equipo.

Datos generales del cliente:

Número de identificación

Razón social

Datos generales del servicio o reservación:

Número de la reservación

Número de identificación del cliente que reserva

Razón social del cliente que reserva

Fecha en que solicita el servicio

Nombre del producto a realizar

Versión del producto

Locación

Nombre de la persona que reserva por parte del cliente

Nombre de la persona que realizó la reservación por Qualli

Datos del equipo a reservar:

Número del equipo

Nombre del equipo

Número de la sala donde se empleará

Nombre de la sala

Fecha del servicio o utilización

Horario del servicio o utilización

Nombre del operador de la sala

Tarifa de la renta del equipo

Total de horas que se ocupará

Importe de la renta

NOMBRE DEL CAMPO	DESCRIPCION	FUENTE DE DATOS	ATRIBUTOS	USO DEL CAMPO	SEGURIDAD	VALOR de 1 al 10	RELACION
No. reservación	Identificador consecutivo por cada reservación	Programación	0 < E < 9999	Para identificar cada consecutivo	Nadie la puede cambiar	10	Facturación de órdenes de trabajo
No. del cliente	Identificador consecutivo del cliente	Cliente	0 < E < 9999	Para identificar a cada cliente	Nadie lo cambia	10	Clientes
Fecha de la reservación	Fecha de reservación de equipo	Programación	Fecha > 01/12/92	Referencia fecha en que habla el cliente	No se cambia la asigna el sistema	8	
Producto	Nombre del proyecto que realiza el cliente	Cliente	Cadena de caracteres	Identificar el proyecto del cliente	Lo puede cambiar el cliente	8	
Versión	Versión del producto del servicio	Cliente	Cadena de caracteres	Clasificar las versiones del producto	Lo puede cambiar el cliente	6	
Locación	En dónde se filmará el producto	Cliente	Cadena de caracteres	Identificar cargos extras en locaciones	Lo puede cambiar el cliente	7	
Nombre del que reserva por Qualli	Empleado de programación que reserva	Programación	Cadena de caracteres	Para posibles aclaraciones	No se cambia	8	Empleados
Nombre del que reserva por cliente	Cliente que reserva en Qualli	Cliente	Cadena de caracteres	Para posibles aclaraciones	No se cambia	8	
No. del equipo	No. de la tarifa del equipo reservado	Cliente	Entero	Identificar el equipo y sus características	Bajo restricción lo cambia el cliente	8	Catálogo del equipo
Nombre del equipo	Descripción del equipo a facturar	Cliente	Cadena de caracteres	Identificar el equipo	Bajo restricción lo cambia el cliente	8	Catálogo del equipo
No. de sala	No. de sala en la que trabaja el cliente	Cliente	Entero	Identificar la sala	Bajo restricción lo cambia el cliente	8	Catálogo del equipo
Nombre de la sala	Nombre de sala en la que el cliente trabaja	Cliente	Cadena de caracteres	Identificar la sala	Bajo restricción lo cambia el cliente	8	Catálogo del equipo
Fecha del servicio	Fecha para utilizar el equipo y las salas	Cliente	Fecha > 01/12/92	Para "marcar" el equipo en esa fecha	Bajo restricción lo cambia el cliente	9	
Horario del servicio	Horario para utilizar el equipo	Cliente	Real	Para "marcar" el equipo en ese horario	Bajo restricción lo cambia el cliente	8	

Total de horas reservadas	Suma hora final menos hora inicial	Sistema	Real	Para conocer el total de horas	Bajo restricción lo cambia el cliente	8	
Nombre del operador	Operador asignado al servicio	Programación	Cadena de caracteres	Para programar a los operadores	Lo asigna programación	6	
Tarifa de la renta	Precio del servicio	Programación	Real	Conocer el importe de la reservación	No se cambia	8	Catálogo de equipo
Importe de la renta	Acumulado de lo rentado	Sistema	Real	Conocer el total de la reservación	No se cambia	8	

Diccionario de Datos

Para normalizar esta lista, se separan todos los grupos de datos (en este caso la lista de equipo a reservar con su información correspondiente, fecha, hora, etc., ya que puede ocurrir varias veces si se solicitan diferentes equipos para el servicio) de forma que ningún archivo tenga grupos repetidos como se muestra a continuación:

LISTA SIN NORMALIZAR

NUM. DE RESERVACION
NUM. DE CLIENTE
RAZON SOCIAL DEL CLIENTE
FECHA DE LA RESERVACION
PRODUCTO
VERSION
LOCACION
NOMBRE DEL QUE RESERV. QUALLI
NOMBRE DEL QUE RESERV.CLIENTE
NUMERO DE EQUIPO
NOMBRE DEL EQUIPO
NUMERO DE SALA
NOMBRE DE SALA
FECHA DEL SERVICIO
HORARIO DEL SERVICIO
TOTAL HORAS RESERV.
NOMBRE DEL OPERADOR
TARIFA DE LA RENTA
IMPORTE RENTA

LISTAS EN 1a. FORMA NORMAL (1NF)

NUM. DE RESERVACION
NUM. DE CLIENTE
RAZON SOCIAL DEL CLIENTE
PRODUCTO
VERSION
LOCACION
NOMBRE DEL QUE RESERV. QUALLI
NOMBRE DEL QUE RESERV.CLIENTE

NUM. DE RESERVACION
NUM. DE SERVICIO
NUMERO DE EQUIPO
NOMBRE DEL EQUIPO
NUMERO DE SALA
NOMBRE DE SALA
FECHA DEL SERVICIO
HORARIO DE SERVICIO
NOMBRE DEL OPERADOR
TARIFA DE LA RENTA
TOTAL HORAS RESERV.
IMPORTE RENTA

Este nivel de simplificación se llama primera forma normal (1NF), representamos esta estructura de datos 1NF de la siguiente forma (denominada también esquema relacional):

RESERVA:*(NUM-RESERVA, NUM-CLIENTE, RAZON-SOCIAL, FECHA-RESERV, PRODUCTO, VERSION, LOCACION, NOMBRE-RESERV-CLI, NOMBRE-RESERV-QUA).*

Se cambia el campo de horario de servicio por hora inicial del servicio, minuto inicial del servicio, hora final del servicio y minuto final del servicio para tener más detalle del horario que se reserva.

RESERVUR:*(NUM-RESERVA, NUM-SERVICIO, NUM-EQUIPO, NOMBRE_EQUIPO, NUM-SALA, FECHA-RENTA, HORA-INICIO, MIN-INICIO, HORA-FIN, MIN-FIN, NOMBRE-OPERADOR, TARIFA-HORA, TOTAL-HORAS, IMPORTE-RENTA).*

Los archivos anteriores se denominan relaciones, una definición más formal de la 1ra. forma normal sería:

- Una relación R está en primera forma normal (1FN) si y sólo si todos los dominios subyacentes sólo contienen valores atómicos.

Pueden efectuarse otras normalizaciones identificando los elementos de datos clave y los que no son clave. Un elemento clave se utiliza para identificar uno o más elementos que no son clave, por ejemplo: num-equipo, es un dato clave e identifica únicamente a descripción, nombre del equipo y tarifa por hora. En este ejemplo, num-reserva y num-servicio, son datos clave, para las relaciones anteriores, y existen algunos elementos (num-sala, fecha-servicio, hora-min, min-ini, hora-fin, min-fin, nombre-operador, tarifa-hora, total-hora, importe-renta) que se dice que son completa y funcionalmente dependientes debido que puedan obtenerse sólo si se conocen los elementos clave para la relación (num-reserv, num-servicio).

El elemento nombre-equipo y tarifa-hora, que no son claves, no son dependientes funcionalmente, debido a que necesitamos conocer una clave, num-equipo, para acceder a ellos, en el mismo caso está num-cliente y razón-social, ya que razón-social se puede obtener sólo conociendo num-cliente y no la clave de la relación (num-reserv, num-servicio) por lo que no son funcionalmente dependientes. Lo mismo se presenta en el caso de num-sala y nombre-sala.

Esta regla nos ayuda a formar lo que llamamos catálogos y directorios y que además nos sirven no sólo para una relación, sino para todas las relaciones que los necesiten del sistema (uso común), además de que nos permiten manejar información más detallada en las relaciones de catálogo o directorio.

Para conseguir la segunda forma normal (2NF), deben reorganizarse las relaciones de forma que ningún dato que no sea clave sea completa y funcionalmente dependiente, los siguientes esquemas son de relaciones en 2NF; en donde se muestra con subrayado la parte rectora (clave) y sin subrayar la parte dependiente :

RESERVA: *(Num-Reserv, Fecha-Reserv, Num-Cli, Producto, Versión, Locación, Nombre-Reserv-Qualli, Nombre-Reserv-Cli)*
RESERVUR: *(Num-Reserv, Num-Servicio, Num-Equipo, Num-Sala, Fecha-Renta, Hora-inicio, Min-inicio, Hora-Fin, Min-Fin, Nombre-Operador,Total-hora, Importe-Renta).*
URENTA: *(Num-equipo, Nombre_equipo, Tarifa-hora)*
CLIENTE: *(Num-cliente, Razon-social)*
SALA: *(Num-sala, Nombre-sala)*

La simplificación a la tercera forma normal (3NF) puede realizarse, si todas las condiciones para la 2NF, se cumplen y ningún elemento que no sea clave, puede derivarse

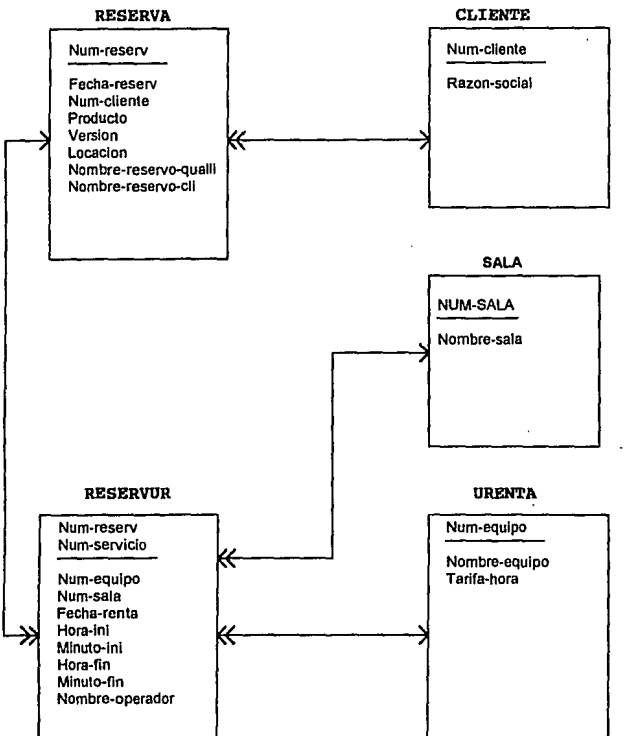
de una combinación de otros elementos que no son clave en ninguna de las relaciones, por ejemplo, total-hora puede calcularse como la diferencia de Hora-fin, Minuto-fin menos Hora-ini, Minuto-ini, por lo tanto, no es necesario mantenerla en las relaciones, también el campo Importe-renta se puede obtener de (Hora-fin, Minuto-fin menos Hora-ini, Minuto-ini) por Tarifa-hora por lo que también se excluye.

Las relaciones en 3NF quedaría de la siguiente manera:

RESERVA: *(Num-Reserv, Fecha-Reserv, Num-Cli, Producto, Versión, Locación, Nombre-Reserv-Qualli, Nombre-Reserv-Cli)*
RESERVR: *(Num-Reserv, Num-servicio, Num-Equipo, Num-Sala, Fecha-Renta, Hora-Ini, Minuto-Ini, Hora-Fin, Minuto-Fin, Nombre-Operador)*
URENTA: *(Num-Equipo, Nombre_equipo, Tarifa-Hora)*
CLIENTE: *(Num-Cliente, Razon-Social)*
SALA: *(Num-Sala, Nombre_sala)*

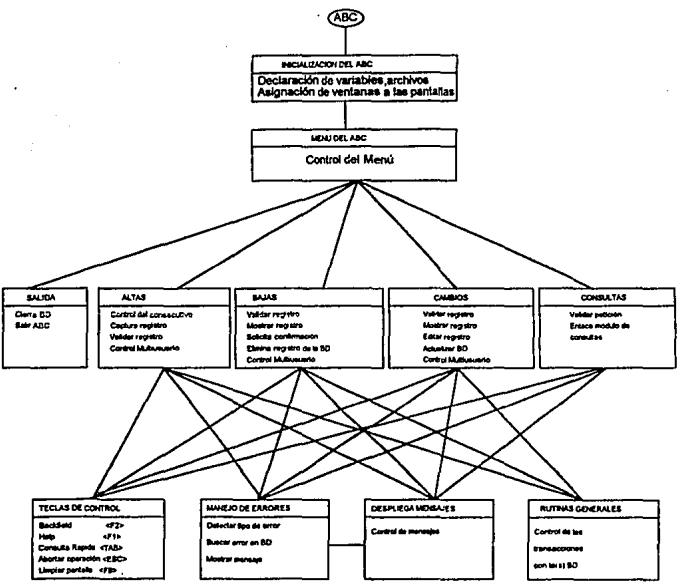
El proceso de normalización simplifica las estructuras de datos y elimina las redundancias y elementos de datos innecesarios de una base de datos.

Por último representamos en forma gráfica el modelo conceptual, en donde se muestran todas las relaciones.
Una relación es una unión o un enlace entre dos conjuntos de datos. Esta puede ser "uno a uno" indicada por una sola flecha, "uno a varios" indicada por doble flecha o "varios a varios" con doble flecha en cada extremo de la relación.
Por ejemplo: Una reservación (reserva) puede tener varias partidas (reservur) pero un renglón (reservur) puede tener solo una reservación (reserva),



Representación Gráfica del Modelo Conceptual

Diseño arquitectónico: *En esta parte del diseño se describe una estructura modular y sus relaciones de control entre ellos, a continuación se muestra la estructura del prototipo del "ABC" para ejemplificar lo anterior:*



Diseño procedimental: *Definición de los detalles algorítmicos que deben establecerse en un lenguaje natural.*

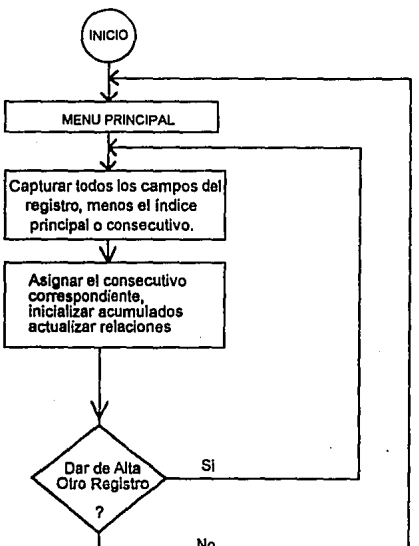


Diagrama de Flujo del módulo de ALTAS del "ABC"

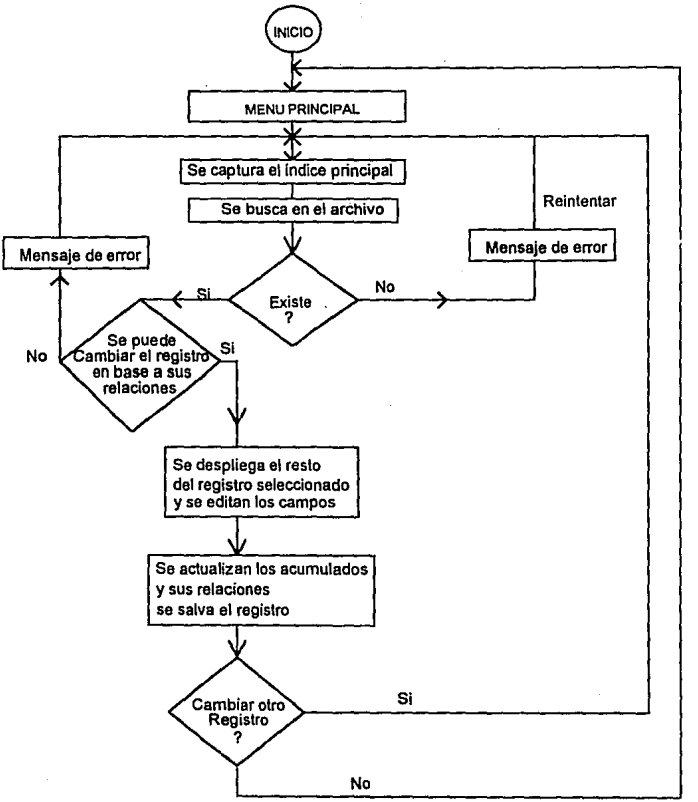


Diagrama de Flujo del módulo de CAMBIOS del "ABC"

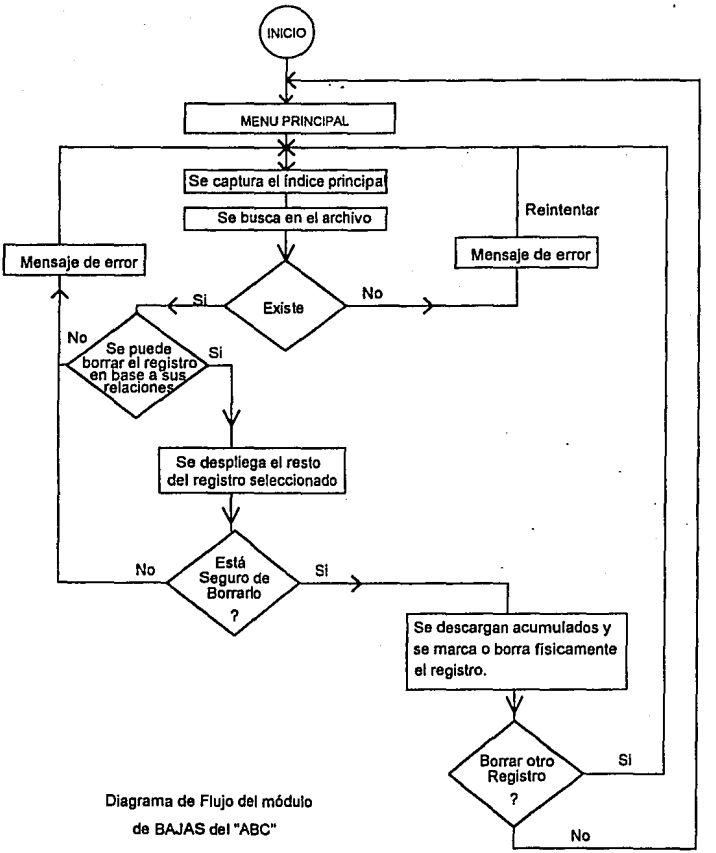


Diagrama de Flujo del módulo de BAJAS del "ABC"

4.2.2 .- CODIFICACION; Traslación del diseño a un lenguaje de computadoras

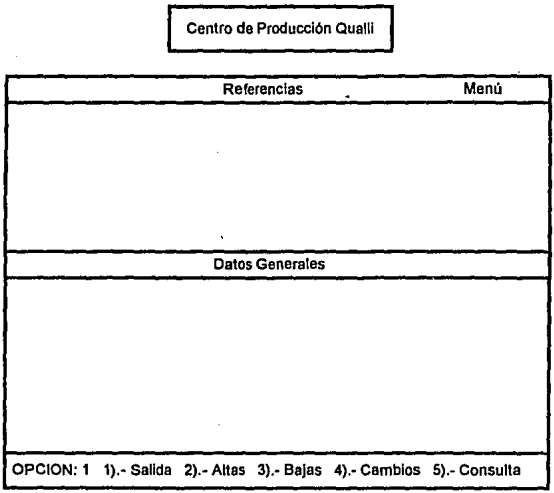
En esta etapa se inicia la construcción del prototipo, partiendo de toda la información generada en las etapas anteriores, como ya se indicó en el capítulo 3.2 en donde se describió en forma detallada a la plantilla "ABC", describiremos ahora el contenido de esta plantilla y como queda después de armar el prototipo.

En primer lugar tenemos la sección de imágenes, la cual se llena de la siguiente manera. en la parte del logotipo se indican el nombre de la empresa y debajo el nombre del proceso.

En la parte de Referencias, se acomodan los campos que son parte del índice principal, con los cuales se efectuará la búsqueda del registro. Se pueden anexar además algunos campos que sean llaves y relaciones de otros archivos para que al llamarlos presenten información adicional que sirva de referencia al que captura.

En la parte de Datos Generales se ubican todos los campos restantes del registro, en caso de llevar alguna validación se indica junto al campo en cuestión o el mismo software de la base de datos nos lo indicará mediante un mensaje.

Analizaremos a continuación el listado de la plantilla "ABC" y haremos algunas observaciones de cada sección, las cuales delimitaremos remarcándolas con negritas y el programa lo describimos con letras más pequeñas.



Ejemplo de una Imágen de un "ABC" sin Llenar

La plantilla del "ABC" se divide en las siguientes secciones:

- Sección de Imágenes
  - .-Pantalla del logotipo del sistema
  - .-Pantalla de captura de las referencias y datos generales
  - .-Pantalla del menú del "ABC"
- Sección de Declaraciones:
  - .-Declaración de los archivos que se usarán en la aplicación
  - .-Declaración de los nombres de los campos de las imágenes

para facilitar su manejo.

.-Declaración de las variables de tipo: string, number, integer, date

- Sección de la rutina de manejo del menú:

- Sección de ALTAS

-.Sección de BAJAS

-.Sección de CAMBIOS

-.Sección de CONSULTAS

-.Sección de programación de teclas predefinidas con funciones específicas.

-.Sección de rutinas para la presentación de imágenes.

-.Sección de rutinas especiales de cada aplicación.

A continuación se describe el contenido de cada sección y su forma de llenarlas para obtener un prototipo:

// TITULO: ARESERVA.FRM

//PROPOSITO: Actualizar la información de las reservaciones

//ENTRADAS :

// Num_reserv = Identificador de la reservación

// Num_cliente= Cliente que reserva

// Producto= Nombre del trabajo a realizar

// Versión = Clasificación del producto

//SALIDAS:

// Se encadena al programa "ARESURV" que trabaja con el detalle de

// reservaciones.

//SUBRUTINAS REFERENCIADAS:

// 1.- MENU_PRINCIPAL

// 2.-ALTAS

// 3.-BAJAS

// 4.-CAMBIOS

// 5.-CONSULTAS

// 6.-SALIDAS

//AUTOR: ALFONSO MORA BELTRAN

//AUDITOR: ARMANDO HERNANDEZ DEL CASTILLO

//FECHA DE CREACION: 01/01/93

//MODIFICACIONES:

/logotipo

Centro de Producción Qualli Actualización de Reservaciones

/captura

Referencias	Menu
Num. de Reservación : <____> Cliente que Reserva : <__><__><__><__> _____	
Datos Generales	
Fecha de la Reservación: __/__/__ Producto: _____ Versión: _____ Locación: _____ Facturar a: Num. de Cliente: <__.><__><__.>_____ Reservó por parte del Cliente: _____ Confirmó por parte del Cliente: _____ Reservó por parte del Qualli : _____ Forma de Facturar (Directa/Presupuesto): _____ Num. del Presupuesto: ____. Imp. Reservación (M.N.): _____.__ (Dls.) : _____.__ Observaciones: _____	
OPCION: 2 1).- Salida 2).- Altas 3).- Bajas 4).- Cambios 5).- Consulta	

/*

A continuación se les asignan nombres o identificadores (a través del comando NAME seguido del nombre de la pantalla) a los campos de las ventanas para poder referirse a ellos más fácilmente en el resto del programa, el orden que deben guardar es de arriba hacia abajo y de izquierda a derecha.

Los renglones que inician con los caracteres "//" son comentarios

```
// declaraciones de la pantalla del menú
name ,menu seleccion primera segunda
// declaraciones de la pantalla de captura
```

```
name captura abc num_reserv res_casa_pro res_agencia res_cliente res_razon_soci
name fecha_Reserv producto version locacion fac_casa_pro fac_agencia fac_cliente fac_razon_soci
name nom_reservo_cli nom_confirm_cli nom_reservo_qua forma_facturar num_presupuesto
name importe_res_mn importe_res_dls
name observaciones1 observaciones2
name mensaje respuesta
```

Se declaran los archivos que se utilizarán

```
//archivo maestro: para guardar consecutivos
open sismaest
//archivo de errores del sistema
open flexerrs
//archivo de reservaciones
open reserva
//archivo de clientes
open cliente
//archivo de consecutivo de reservaciones
open sisreser
//archivo de servicios de reservaciones
open reservar
//archivo de disponibilidad de salas y equipo
open disponur
//archivo bloqueador de equipo ocupado
open bloqur
//catálogo de equipo
open urenta
//archivo maestro de consultas
open siscon
//paquetes de equipo
open purenta
//archivos unicamente para consulta
file_mode flexerrs read_only
file_mode cliente  read_only
file_mode urenta   read_only
```

Se declaran las variables que se utilizarán para control, contadores, registros temporales etc. y pueden ser del tipo: STRING, INTEGER, NUMBER, DATE, CHARACTER E INDICATE.

```
integer opcion contador
string tecla chr 1
string mensaje_1 70 campana 1 nr 6
string comandline 30 comando 6 comando1 comando2 2
integer opcion_cons num_cons
number num_resren
character 7 to campana
// remueve caracteres almacenados en el buffer del teclado
repeat
```

```
keycheck loop
```

El siguiente procedimiento es el de manejo del menú que presenta las opciones de altas, Bajas, Cambios y Consultas, se pueden seleccionar cualquiera de las opciones al ubicarse en la opción deseada y presionar la tecla <Return>, por lo tanto, no se requieren modificaciones a la siguiente rutina (menu_principal)

```
clearscreen
// Inicialización del Menú del ABC
menu_principal:
 gosub dib_pan_log
 gosub dib_pan_cap
 display 'Menu' to abc  ( retain )
 gosub dib_pan_men
 // ventanas en color de la pantalla
 screenmode 27 on
 move '1'          to menu.2
 move 'Salida'     to menu.3
 move '2'          to menu.4
 move 'Altas'      to menu.5
 move '3'          to menu.6
 move 'Bajas'      to menu.7
 move '4'          to menu.8
 move 'Cambios'    to menu.9
 move '5'          to menu.10
 move 'Consulta' to menu.11
 // ventana : fondo negro y letreros color cian ( parpadeante )
 screenmode 139 on
 move 1 to opcion
 indicate ventana_cero true
 indicate cons false
 // captura y despliega selección
 obten_opcion:
  // ventanas en color de la pantalla
  screenmode 27 on
  [not ventana_cero] move segunda& to segunda&
  // ventana : fondo negro y letreros color cian ( parpadeante )
  screenmode 139 on
 // windoindex es un apuntador a las ventanas de la pantalla seleccionada
  move ( opcion + opcion - 2 ) to windowindex
  indicate ventana_cero as opcion eq 0
  [not ventana_cero] if segunda& eq " begin
                move 0 to opcion
                indicate ventana_cero true
  end
  [not ventana_cero] move segunda& to segunda&
  // ventana : fondo negro y letreros color cian
  screenmode 11 on
```

```
        move opcion to seleccion
        inkey chr
        if chr in '12345' begin
                move chr to opcion
                goto obten_opcion
        end
        if chr in 'SsAaBbCc' begin
         if chr in 'Ss' move 1 to opcion
         if chr in 'Aa' move 2 to opcion
         if chr in 'Bb' move 3 to opcion
         if chr in 'Cc' begin
                if[cons] begin
                  move 5 to opcion
                  indicate cons false
                end
                else begin
                  move 4 to opcion
                  indicate cons true
                end
         end
         goto obten_opcion
        end
        [not key.right] indicate key.right as chr eq ''
        [key.left] indicate key.right false
        [not key.return][not key.escape] begin
         indicate mueve_ventana as opcion ge 1
         indicate mueve_ventana group all [mueve_ventana] and any [key.field key.left]
         [mueve_ventana] begin
                  move ( opcion - 1 ) to opcion
                  if opcion eq 0 move 5 to opcion
         end
         [key.right] if opcion lt 5 move ( opcion + 1 ) to opcion
         [key.right] else move 1 to opcion
         goto obten_opcion
        end
        // se refresca la pantalla de captura
        [key.return] begin
                screenmode off
        end
        if opcion eq 1 goto salida
        if opcion eq 2 goto altas
        if opcion eq 3 goto bajas
        if opcion eq 4 goto cambios
        if opcion eq 5 gosub consulta
goto menu_principal
```

A continuación inician las rutinas que manejan las operaciones permitidas por el proceso y son: Altas, Bajas, Cambios y Consultas, estas rutinas se llenan con los campos de los archivos a actualizar y se sigue el órden que nos marque la ubicación en que se encuentren

en las imágenes de captura.

```
//      *** ABC ***

altas:
 //se captura la llave
 clear flexerrs
 clear reserva
 clearform captura
 move 2 to opcion
 gosub dib_pan_log
 gosub dib_pan_cap
 autopage captura
 entergroup
 display'Altas' { retain }
```

Para controlar el número consecutivo que le corresponde a la nueva reservación, en este momento del programa se le asigna un 0 y solo al terminar de capturar todo el registro, en el momento de salvar, se le asigna el consecutivo que le corresponda (como se verá posteriormente).

Después se capturan los campos que forman relaciones a otros archivos, como en este caso lo es el archivo de clientes, esto con el fin de presentarle información adicional al usuario para que le sirva de referencia, en el ejemplo, se le presenta la razón social del cliente.

Con esta idea se maneja la sección de Referencias en la pantalla de captura, ya se explicó anteriormente el propósito de esta sección.

```
// última reservación
move 0 to num_reserv
repeat
 move 1 to opcion_cons
 indicate res true
 indicate transaccion true
 entry reserva.res_num_cp  res_casa_pro
 entry reserva.res_num_ag  res_agencia
 entry reserva.res_num_cli res_cliente
 // busca el Cliente
 gosub despliega_cliente
 clearform mensaje thru respuesta
until [transaccion]
```

A continuación se inicia la captura de los campos restantes del registro, con las

validaciones que correspondan en cada caso.

```
//asigna por omisión la fecha del dia
 sysdate fecha_reserv
 entry reserva.fecha_reserv fecha_reserv
 entry reserva.producto   producto     { capslock }
 entry reserva.version    version      { capslock }
 move "QUALLI" to locacion
 entry reserva.locacion   locacion     { capslock }
 repeat
  move 2 to opcion_cons
  indicate res false
  indicate transaccion true
  entry reserva.fac_num_cp  fac_casa_pro
  entry reserva.fac_num_ag  fac_agencia
  entry reserva.fac_num_cli  fac_cliente
  // busca el Cliente
  gosub despliega_cliente
  clearform mensaje thru respuesta
 until [transaccion]
 entry reserva.nom_reservo_cli  nom_reservo_cli { capslock }
 entry reserva.nom_confirm_cli  nom_confirm_cli { capslock }
 entry reserva.nom_reservo_qua  nom_reservo_qua { capslock }
 entry reserva.forma_facturar  forma_facturar  { capslock, required }
 if "PRESUPUESTO" in forma_facturar begin
  entry reserva.num_presupuesto num_presupuesto
 end
 else move 0 to num_presupuesto
 move 0 to importe_res_mn
 move 0 to importe_res_dls
 entry reserva.importe_res_mn   importe_res_mn  { displayonly }
 entry reserva.importe_res_dls  importe_res_dls  { displayonly }
 entry reserva.observaciones1   observaciones1   { capslock, autoreturn }
 entry reserva.observaciones2   observaciones2   { capslock, autoback   }
```

Por último se controla el modo multiusuario con los comandos REREAD y UNLOCK que marcan el inicio y fin de bloque en el que queremos que los registros estén bloqueados para otros usuarios.

```
 reread  // modo multiusuario ; bloquea la relación
 endgroup
 move 0 to reserva.importe_res_mn
 move 0 to reserva.importe_res_dls
 // no facturada y no impresa
 move "NOIMPRESA" to reserva.status_res
 // se inicializa el contador de renglones para el detalle
 move 0 to reserva.num_ult_renglon
 move 0 to reserva.num_ult_maquina
```

```
move 0 to reserva.num_orden
```

Además de inicializar los campos que le servirán para manejar los diferentes estados del contenido del registro, se hace uso del archivo maestro de reservaciones que lleva el control de los consecutivos, se toma el número que le corresponda y se incrementa en uno este campo (para el consecutivo del siguiente registro).

```
move (sisreser.num_ult_reserva + 1 ) to reserva.num_reserv
move reserva.num_reserv to sisreser.num_ult_reserva
save sisreser
save reserva
unlock  // modo multiusuario : libera la relación
move reserva.num_reserv to num_reserv
movestr reserva.num_reserv to comando
move "El Numero de Reservación asignada es : " to mensaje_1
append mensaje_1 comando ".<Return > para continuar ..."
gosub desp_mensaje
```

Una ves terminada la actualización del registro, se encadena (con el comando CHAIN WAIT) al programa "aresur" el cual maneja el segundo nivel de la relación (captura de los servicios).

```
move "aresur " to comandline
append comandline comando
chain wait comandline
// finaliza la rutina deAltas y reinicia el proceso.
goto altas
```

Este proceso borra registros ya existentes para lo cual solicita la introducción del indice principal y efectúa una búsqueda (a través del comando AUTOFIND) para verificar que exista, en caso de no ser así, indica un mensaje de error y pide que se reintente o finalice el proceso.

```
bajas:
//se captura la llave
clear flexerr
clear reserva
clearform captura
autopage captura
```

```
entergroup
display 'Bajas' { retain }
entry reserva.num_reserv num_reserv { required, autofind, skipfound }
[not found] begin
 move "ERROR : No existe esa Reservación. <RETURN> para continuar ..." to mensaje_1
 gosub desp_mensaje
 move 0 to reserva.recnum
 move 0 to flexerrs.recnum
 goto bajas
end
```

Verifica los estados de la información del registro.

```
if "CANCELADA" match reserva.status_res begin
 move "ERROR : La Reservación ya ha sido Cancelada. <RETURN> para continuar ..." to mensaje_1
 gosub desp_mensaje
 move 0 to reserva.recnum
 move 0 to flexerrs.recnum
 goto bajas
end
if "IMPRESA" match reserva.status_res begin
 move "ERROR : La Reservación ya ha sido Impresa. <RETURN> para continuar ..." to mensaje_1
 gosub desp_mensaje
 move 0 to reserva.recnum
 move 0 to flexerrs.recnum
 goto bajas
end
indicate res true
```

Una vez que se encontró el registro se presentan los campos restantes para que el usuario verifique si es el registro que quiere borrar.

```
entry reserva.res_num_cp  res_casa_pro  { displayonly, noenter }
entry reserva.res_num_ag  res_agencia   { displayonly, noenter }
entry reserva.res_num_cli res_cliente   { displayonly, noenter }
// busca el Cliente
gosub despliega_cliente
clearform mensaje thru respuesta
entry reserva.fecha_reserv fecha_Reserv { displayonly, noenter }
entry reserva.producto     producto     { displayonly, noenter }
entry reserva.version      version      { displayonly, noenter }
entry reserva.locacion     locacion     { displayonly, noenter }
indicate res false
entry reserva.fac_num_cp  fac_casa_pro { displayonly, noenter }
entry reserva.fac_num_ag  fac_agencia  { displayonly, noenter }
entry reserva.fac_num_cli fac_cliente  { displayonly, noenter }
// busca el Cliente
gosub despliega_cliente
entry reserva.nom_reservo_cli nom_reservo_cli { displayonly, noenter }
entry reserva.nom_confirm_cli nom_confirm_cli { displayonly, noenter }
```

```
 entry reserva.nom_reservo_qua  nom_reservo_qua { displayonly, noenter }
 entry reserva.forma_facturar  forma_facturar  { displayonly, noenter }
entry reserva.num_presupuesto  num_presupuesto { displayonly, noenter }
 clearform mensaje thru respuesta
 entry reserva.importe_res_mn  importe_res_mn  { displayonly, noenter }
 entry reserva.importe_res_dls  importe_res_dls { displayonly, noenter }
 entry reserva.observaciones1  observaciones1  { displayonly, noenter }
 entry reserva.observaciones2  observaciones2  { displayonly, noenter }
```

Habilita los comandos para el control de modo multiusuario

```
 reread  // modo multiusuario : bloquea la relación
endgroup
move "N" to respuesta
move " Esta Ud. seguro (S/N) ? " to mensaje_1
gosub desp_mensaje
if respuesta in "Ss" begin
 move "CANCELADA" to reserva.status_res
 save reserva
 gosub cancela_reserv
end
else begin
 move 0 to reserva.recnum
 move 0 to cliente.recnum
 move 0 to flexerrs.recnum
end
unlock  // modo multiusuario : libera la relación
```

Valida si el usuario está seguro de borrar el registro

```
if respuesta in "S" begin
end
goto bajas
```

En la operación de cambios se realizan modificaciones a uno o varios campos de registros ya existentes, por lo que de igual manera que en Bajas, se captura el índice principal para buscar el registro deseado.

```
cambios:
//se captura la llave
clear flexerrs
clearform captura
move 4 to opcion
gosub dib_pan_log
gosub dib_pan_cap
autopage captura
entergroup
```

```
display 'Cambios' { retain }
entry reserva.Num_reserv num_reserv { required, autofind, skipfound }
[not found] begin
 move "ERROR : No existe esa Reservación. <RETURN> para continuar ..." to mensaje_1
 gosub desp_mensaje
 move 0 to reserva.recnum
 move 0 to flexerra.recnum
 goto cambios
end
```

Se verifican el estado de la información para saber si está permitido realizar el cambio

```
if "CANCELADA" match reserva.status_res begin
 move "ERROR : La Reservación ya ha sido Cancelada. <RETURN> para continuar ..." to mensaje_1
 gosub desp_mensaje
 move 0 to reserva.recnum
 move 0 to flexerra.recnum
 goto cambios
end
if "IMPRESA" match reserva.status_res begin
 move "ERROR : La Reservación ya ha sido Impresa. <RETURN> para continuar ..." to mensaje_1
 gosub desp_mensaje
 move 0 to reserva.recnum
 move 0 to flexerra.recnum
 goto cambios
end
```

Se presenta el resto del registro quedando en modo de edición para poder realizar el (los) cambio(s).

```
repeat
 move 1 to opcion_cons
 indicate res true
 indicate transaccion true
 entry reserva.res_num_cp  res_casa_pro
 entry reserva.res_num_ag  res_agencia
 entry reserva.res_num_cli res_cliente
 // busca el Cliente
 gosub despliega_cliente
 clearform mensaje thru respuesta
until [transaccion]
entry reserva.fecha_reserv fecha_Reserv
entry reserva.producto    producto     { capslock }
entry reserva.version     version      { capslock }
entry reserva.locacion    locacion     { capslock }
repeat
 move 2 to opcion_cons
 indicate res false
 indicate transaccion true
 entry reserva.fac_num_cp  fac_casa_pro
```

```
entry reserva.fac_num_ag  fac_agencia
entry reserva.fac_num_cli  fac_cliente
// busca el Cliente
gosub despliega_cliente
clearform mensaje thru respuesta
until [transaccion]
entry reserva.nom_reservo_cli  nom_reservo_cli { capslock }
entry reserva.nom_confirm_cli  nom_confirm_cli { capslock }
entry reserva.nom_reservo_qua  nom_reservo_qua { capslock }
entry reserva.forma_facturar  forma_facturar  { capslock, required }
if "PRESUPUESTO" in forma_facturar begin
 entry reserva.num_presupuesto num_presupuesto
end
else move 0 to num_presupuesto
entry reserva.importe_res_mn  importe_res_mn  { displayonly }
entry reserva.importe_res_dls  importe_res_dls  { displayonly }
entry reserva.observaciones1  observaciones1  { capslock, autoreturn }
entry reserva.observaciones2  observaciones2  { capslock, autoback  }
reread  // modo multiusuario : bloquea la relación
endgroup
```

Termina el proceso de cambios y se maneja el modo multiusuario para controlar la actualización del registro y encadenando al programa "aresur" para manejar el siguiente nivel.

```
save reserva
unlock  // modo multiusuario : libera la relación
move "aresur " to comandline
movestr reserva.num_reserv to comando
append cwmandline comando
chain wait comandline
goto cambios
```

Encadena el proceso al programa "creserva" que es del tipo de los prototipos de consultas descritos anteriormente

```
consulta:
 move 5 to opcion
 move "creserva " to comandline
//sintáxis : append nombre_programa, num_consulta, posición de despliege de x, y
 append comandline " 0 7 12 "
 chain wait comandline
return
```

Opción de salida con la que finaliza el proceso y se regresa al menú que lo llamó

```
salida:
 abort
```

Sección de Rutinas: desp_mensaje: se utiliza para enviar mensajes y esperar una respuesta, utiliza las ventanas de mensaje y respuesta de la pantalla de captura.
La rutina desp_error es la que busca el archivo de errores y una vez que lo encontró llama a la rutina desp_mensaje

```
desp_mensaje:
 display mensaje_1 to mensaje
 showln campana
 accept respuesta
return

desp_error:
 move lasterr to flexerra.recnum
 find eq flexerra by recnum
 [found] begin
  move flexerra.error_descr to mensaje_1
  append mensaje_1 ". <RETURN> para continuar ..."
  gosub desp_mensaje
 end
 [finderr] error lasterr
 indicate err false
 unlock // libera posibles relaciones ocupadas
 clearform mensaje thru respuesta
 if opcion eq 1 return salida
 if opcion eq 2 return altas
 if opcion eq 3 return bajas
 if opcion eq 4 return cambios
return menu_principal
```

En Dataflex se pueden tener teclas preprogramadas con funciones específicas que requiera la aplicación, a continuación se muestran las teclas que quedan definidas con alguna función y las que se deshabilitan.

```
//Tecla ESC para salir del "ABC"
keyproc key.escape
 if opcion eq 2 return menu_principal
 if opcion eq 3 return menu_principal
 if opcion eq 4 return menu_principal
```

// Tecla TAB

Esta tecla nos ayuda para activar las "Consultas rápidas", que nos ayudan en el manejo de claves (de clientes y de servicios en este caso) y nos evitan el manejo de catálogos y directorios en forma manual.

Esta operación se realiza a través de un archivo temporal (siscon), ya que Dataflex no permite retorno de parámetros entre aplicaciones foráneas.

```
keyproc key.find
 if opcion_cons ne 0 begin
  reread sirmaest
   calc ( sismaest.num_ult_cons + 1 ) to num_cons
   move num_cons to sismaest.num_ult_cons
   save sismaest
   unlock
 // clientes de reservaciones
   if opcion_cons eq 1 move "ecliente " to comandline
 //clientes que facturan
  if opcion_cons eq 2 move "ecliente " to comandline
  movestr num_cons to comando
  append comandline comando " 6 10"
  chain wait comandline
  clear siscon
  move num_cons to siscon.num_consulta
  find eq siscon by index.1
  [found] begin
   reread siscon
    indicate prueba as siscon.status_consulta match "CONSULTA"
    [prueba] begin
     if opcion_cons eq 1 begin
      move siscon.indice_1 to res_casa_pro
      move siscon.indice_2 to res_agencia
      move siscon.indice_3 to res_cliente
     end
     if opcion_cons eq 2 begin
      move siscon.indice_1 to fac_casa_pro
      move siscon.indice_2 to fac_agencia
      move siscon.indice_3 to fac_cliente
     end
    end
    delete siscon
   unlock
  end
  move 0 to current_image
  gosub dib_pan_cap
 end
 entagain
return
```

// Tecla F9 Limpia la pantalla y desactiva los registros en uso.

```
keyproc key.clear
 move 0 to flexerra.recnum
 move 0 to reserva.recnum
 move 0 to cliente.recnum
 move 0 to flexerra.recnum
 clearform
 if opcion eq 2 return altas
 if opcion eq 3 return bajas
 if opcion eq 4 return cambios
return menu_principal
```

//Tecla F1 Activa el programa "abchelp" el cual es una ayuda en linea para la operación y manejo de la pantalla

```
Keyproc key.help
 move "abchelp " to comandline
 movestr  5 to comando1
 movestr 16 to comando2
 append comandline comando1 " " comando2
 chain wait comandline
 move 0 to current_image
 gosub dib_pan_cap
 entagain
return
```

Rutinas para el desplegado de las pantallas, con su ubicación y color

```
dib_pan_log:
 // se despliega el logotipo
 // pantalla : fondo color azul intenso , figuras color cian
 // ventana  : fondo color azul intenso , figuras color cian
 page logotipo at 00 18 colors 27 27
return

dib_pan_cap:
 // se despliega la pantalla de captura
 // pantalla : fondo color azul intenso , figuras color cian
 // ventana  : fondo color azul intenso , figuras color blanco
 page captura at 04 00 colors 27 31
return

dib_pan_men:
 // se despliega pantalla menú del abc
```

```
// pantalla : fondo color azul intenso , figuras color cian
// ventana  : fondo color azul intenso , figuras color blanco
page menu  at 21 00 colors 27 31
return
```

// bloque de teclas no permitidas. Se reservan porque el compilador las carga por omisión al encontrar la macro ENTERGROUP que se usa en la captura.

```
// f3
keyproc key.sfind
 entagain
return

// f4
keyproc key.print
 entagain
return

// f5
keyproc key.delete
 entagain
return

// f7
keyproc key.user
 entagain
return

// f8
keyproc key.user2
 entagain
return

// f10
return
```

Sección de rutinas específicas de la aplicación

```
despliega_cliente:
 clear cliente
 if [res] begin
  move res_casa_pro to cliente.casa_productora
  move res_agencia  to cliente.num_agencia
  move res_cliente  to cliente.num_cliente
 end
 else begin
  move fac_casa_pro to cliente.casa_productora
  move fac_agencia  to cliente.num_agencia
```

```
 move fac_cliente  to cliente.num_cliente
end
find eq cliente by index.1
[not found] begin
 move "ERROR : No existe ese Cliente. <RETURN> para continuar ..." to mensaje_1
 gosub desp_mensaje
 move 0 to cliente.recnum
 indicate transaccion false
end
[found] begin
 if [res] display cliente.razon_social to res_razon_soci
 else display cliente.razon_social to fac_razon_soci
end
return
```

4.2.3 REVISION Y PRUEBAS; Evaluación del funcionamiento del prototipo elaborado.

La prueba de software es un elemento crítico para la garantía de calidad del software y representa un último repaso de las especificaciones del diseño y de la codificación:

1.- La prueba es un proceso de ejecución de un programa con la intención de descubrir un error.

2.- Un buen caso de prueba es aquél que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

3.- Una prueba tiene éxito, si descubre un error no detectado hasta entonces.

Para nuestro caso en estudio, el proceso de pruebas parte de lo general a lo detallado, recordemos que en un primer paso del método propuesto, se obtiene un primer prototipo que es la base del diseño para refinarlo y tener un producto terminado, esto es, el usuario tiene una participación fundamental en el proceso de desarrollo como se indicó en el Capítulo 3, dentro de los pasos para el manejo y construcción de prototipos, en el paso 5 se establece que una vez que el prototipo ha sido terminado, se presenta al usuario, el cual "conduce la prueba" de la aplicación y sugiere modificaciones.
Este paso, es el núcleo del método propuesto, ya que estando en un proceso iterativo, el resultado de la prueba será quien indique si ya se llegó al producto terminado o se continúa en su refinamiento.

Es importante recordar que con el modelo propuesto es posible regresar hasta la fase de definición en donde se podrán replantear los objetivos y ajustar el prototipo hacia el producto deseado.

El proceso de prueba irá evolucionando en la medida que avancen las actualizaciones del prototipo, hasta llegar al producto final, esto es, en un principio se tomarán como entradas al proceso la especificación de requerimientos del software, la especificación del diseño y el código fuente, con esto se elabora un procedimiento de prueba en donde se determinan los casos de prueba y los resultados esperados.

Se lleva a cabo la prueba y se evaluan los resultados, o sea se comparan los resultados de prueba con los esperados y cuando no coinciden implica que hay un error y comienza la depuración.

Se presenta nuevamente el prototipo al usuario quien en algún momento decidirá si el producto es el deseado y si lo és el producto es liberado al usuario final y se realiza ahora un proceso de pruebas pero con cargas reales de trabajo, en donde seguramente ocurrirán otros errores o consideraciones adicionales, las cuales los trataremos en el siguiente capitulo.

4.3 FASE DE MANTENIMIENTO

Una vez terminado el sistema, y aún tras de haberle hecho todas las pruebas de software necesurias, seguramente habrá cambios al sistema, estos cambios pueden ser por corrección de defectos, adaptaciones al entorno externo o aumento de funciones para su mejora y actualización.

Consideramos que en un producto de software de calidad, los cambios deben darse en una forma natural, esto es, si se tiene una especificación de los requerimientos lo bastante completa y que realmente refleje las necesidades y caracteristicas del medio en donde

opera el sistema y un diseño de la base de datos normalizado en donde se cuidan los aspectos de integridad de los datos e independencia de la base de las aplicaciones, seguramente se tendrá un producto de software que soportará los cambios y adaptaciones futuras, esto como se mencionó anteriormente, se tendrá un software con pocas probabilidades de volverse obsoleto en corto tiempo.

Un fin primordial de aplicar una metodología bien definida en el desarrollo de software, es el de mejorar la facilidad con la que se puedan efectuar los cambios y reducir la cantidad de esfuerzo gastado en el mantenimiento, como se mencionó en el capítulo anterior existen tres tipos principales de mantenimiento:

- Mantenimiento Correctivo.
- Mantenimiento Adaptativo.
- Mantenimiento de Aumento o perfectivo.

4.3.1 MANTENIMIENTO CORRECTIVO; Resolución de los defectos del sistemas.

Esta actividad se dá debido a que no es razonable asumir que la prueba del software haya descubierto todos los errores del nuevo sistema y además al tener cargas reales de trabajo es posible que se presenten errores, éstos deberán de ser reportados al departamento de desarrollo para su corrección.

Para nuestro caso en estudio, el mantenimiento correctivo nos ayuda a reducir errores, recordemos que el desarrollo parte de una plantilla base para el prototipo (la cual está por demás probada) asegurando que todas las funciones básicas para manipular el sistema están trabajando correctamente, reduciendo el problema a los módulos nuevos que se incorporaron al sistema.

4.3.2 MANTENIMIENTO ADAPTATIVO; Modificaciones al sistema para adaptarlo al entorno externo.

Este mantenimiento se dá debido al rápido cambio inherente a todo aspecto de la informática, regularmente aparecen nuevos sistemas operativos o nuevas versiones de los antiguos, mejoras de los equipos periféricos, por lo que es necesario modificar el software para que interaccione adecuadamente con su entorno cambiante.

Para el mantenimiento adaptativo, contamos con una ventaja a nuestro favor ya que la compañía que desarrolla Dataflex (Data Access Corporation) generalmente trabaja en actualizaciones a nuevas versiones para adaptarse al entorno cambiante, y todo se reduce a una conversión de la base de datos y programas a la nueva versión mediante algún programa convertidor suministrado por Data Access Corp.

Por lo que es de gran ayuda el ejecutar programas que actualizan el sistema para cambiar de versión. Otra característica muy interesante del Dataflex es que se puede migrar a otros sistemas operativos fácilmente, los más comunes UNIX y VMS, por si acaso el cambio fuera hacia una plataforma más alta de Hardware (equipo basado en minicomputadora) el problema sería resuelto fácilmente.

4.3.3 MANTENIMIENTO DE AUMENTO O PERFECTIVO; Perfeccionamiento y/o aumento de las funciones del sistemas para su actualización y mejora.

A medida que se usa el sistema, se reciben de los usuarios recomendaciones sobre nuevas posibilidades sobre funciones ya existentes y sobre mejoras en general. En esta actividad se invierte la mayor cantidad de esfuerzo gastado en el mantenimiento.

Para el mantenimiento de aumento, como se mencionó, se requiere la mayor cantidad de tiempo, lo que hacemos es partir de aplicaciones básicas (reportes, consultas, etc.) y que el usuario vaya encontrando nuevas opciones conforme maneja el sistema, esto es una buena señal de que el nuevo sistema le está resolviendo sus necesidades y además, depende operativamente de él, en la actualidad tenemos un equipo de desarrollo dedicado exclusivamente a resolver problemas de este tipo.

Un ejemplo de este mantenimiento se nos acaba de presentar en el subsistema de reservaciones. Ya que los usuarios nos piden que desean que su sistema corra bajo el ambiente gráfico de Windows para poder explotar beneficios extras que se suministran con él. Sin embargo, estamos evaluando en la actualidad dicha posibilidad, ya que se requiere una mayor inversión en equipo y no necesariamente se aumentaría la productividad relacionada con este subsistema. Por si fuera poco, Data Access ya está contemplando esta posibilidad y está por entregarnos un versión nueva de Dataflex para Windows, que nos será de gran ayuda en la toma de la desición de resolver la actualización del subsistema de reservaciones.

CAPITULO 5 CONCLUSIONES.

El software se ha convertido en el elemento clave de la evolución de sistemas y productos informáticos. En las pasadas cuatro décadas el software ha pasado de ser una resolución de problemas especializada y herramientas de análisis de la información, a una industria por sí misma. Pero la temprana cultura e historia de la "programación" han creado un conjunto de problemas que persisten todavía hoy. El software se ha convertido en un factor limitativo de la evolución de los sistemas informáticos.

La Ingeniería del Software es una disciplina que integra métodos, herramientas y procedimientos para el desarrollo del software de computadoras. Se han propuesto varios paradigmas diferentes, cada uno exhibiendo unas ventajas y desventajas, pero todos tienen una serie de fases genéricas en común: fases de definición, desarrollo y mantenimiento.

Podemos concluir que con el sistema SIACO desarrollado, utilizando el método propuesto (combinacion de paradigmas) tenemos que:

- Se logró la automatización de los principales departamentos de la empresa a través de desarrollar once subsistemas, los cuales son: Contabilidad, Cobranzas, Personal, Tesorería, Facturación, Reservaciones, Almacén, Asesores de Post-Producción, Relaciones Públicas, Paquetes de Servicios y Plan Francés.

Con lo cual logramos que la dirección general de la empresa restableciera la confianza en la gerencia de sistemas y en la informática en general como un camino adecuado para la solución de los problemas administrativos y operativos de Qualli.

Se logró además :

-Aumento en la comunicación entre los departamentos.

-Disminución del tiempo invertido en la captura de información.

-Aumento en la eficiencia del procesamiento de la información.

-Disminución drástica en la redundancia de la información en los distintos departamentos de la empresa.

-Aumento en la eficiencia de la atención a los empleados y clientes.

-Disminución del tiempo invertido en el proceso de cálculo de la nómina de empleados.

-Aumento en la confianza de los directivos y gerentes hacia los sistemas en la toma de decisiones.

-Aumento de la respuesta que se dá a los clientes cuando éstos solicitan alguna información relacionada con sus estados de cuenta.

-Aumento en la eficiencia de la atención que se dá a los proveedores de la empresa.

-Mayor control en el manejo de las finanzas de la empresa.

-Aumento de las ventas de la empresa, a través de una mayor y mejor respuesta en el tiempo de la facturación.

-Aumento en los ingresos de la empresa a través de un mejor y mayor control de la cobranza, de la facturación y de los créditos otorgados a los clientes por la empresa.

-Aumento en el presupuesto del departamento de sistemas, para la compra de software y equipo de cómputo actualizado, y para el pago de empleados del departamento.

Consideramos además que como se requería desarrollar un sistema en forma rápida y que no se volviera obsoleto en corto tiempo, a través de la aplicación del método de combinación de paradigmas como una alternativa para el desarrollo, se logró lo siguiente:

-Con el método propuesto se tiene una especificación más clara de las necesidades reales del usuario, ya que le permite tener una participación más activa dentro de la fase de definición del sistema, a través de apoyarse en un prototipo para visualizar como quedará su sistema y evitarse alguna sorpresa si solo se conociera su sistema al final del desarrollo, y se encontrara con un sistema que no es lo que esperaba.

-Para el método propuesto es necesario dedicarle un mayor esfuerzo al inicio de la fase desarrollo al elaborar las plantillas que servirán para el desarrollo de los prototipos, pero una vez que se tengan estas herramientas, ayudarán en el resto del desarrollo del sistema facilitando así su programación.

-Al tener un sistema elaborado con el método propuesto se tiene un sistema uniforme y estándar en cuanto a operación se refiere de tal manera que facilita su aprendizaje a los usuarios, ya que la forma de operarlo es igual en todos los subsistemas.El manual de usuario se reduce a explicar una sola vez el manejo de las pantallas, y en forma más particular a la explicación del contenido de los campos que se manejan en cada aplicación.

Por estar basado el método propuesto en fundamentos téoricos, se alarga la vida útil del sistema, ya que en la actualidad se ha adaptado en forma natural a los cambios requeridos por el usuario y el entorno.

Por lo tanto consideramos que se han cubierto los objetivos planteados al inicio de esta tesis.

APENDICE A. CONCEPTOS BASICOS DE DATAFLEX.

Introducción.

DataFlex de Data Access Corporation ha estado en el mercado desde 1981. Originalmente escrito para correr bajo TurboDos (un sistema operativo multiusuario que corre programas CP/M), DataFlex siempre ha sido un DBMS multiusuario. DataFlex provee un ambiente abierto para desarrollar aplicaciones. Aunque primariamente construido para programadores, DataFlex proporciona servicios tales como AutoDef y DFQuery que permiten a los usuarios principiantes crear aplicaciones de trabajo y reportes. El código generado con AutoDef y DFQuery es completamente consistente con el lenguaje altamente procedural de DataFlex, que provee capacidades de manipulacion extensiva de pantallas, funciones matemáticas y un soporte muy completo de gráficas.

DataFlex ha sido ampliamente reconocido como uno de los más rápidos DBMS para LAN disponibles, tanto en términos de recuperación de datos y elaboración de reportes, como en términos de tiempo de desarrollo de aplicaciones. Por su extensivo uso de indices múltiples para recuperación de datos y reporteo y su sistema construido a base de menús, DataFlex es buen anfitrión para la elaboración de aplicaciones controladas por teclas, que requieren que el usuario sepa poco o nada de los DBMS. En resumen, porque DataFlex corre actualmente en muchos ambientes de sistemas operativos (DOS, OS/2, UNIX, Xenix, AIX, y VAX/VMS) las aplicaciones son muy transportables. DataFlex provee un sistema completo, fácil de usar y con funciones múltiples para el manejo de preguntas que pueden proporcionar reportes ad hoc tan buenos como para crear código editable para poder realizar reportes más formales.

DataFlex es un sofisticado sistema de bases de datos relacionales que corre en una gran variedad de redes de área local y computadoras, incluyendo sistemas UNIX y computadoras DEC VAX. El paquete viene con dos carpetas que incluyen el manual del usuario, una enciclopedia de comandos, y dos secciones de aprendizaje que muestran los pasos a seguir para la elaboración de una aplicación.

DataFlex es utilizable por usuarios que no sepan programar la elaboración de una base de datos y su uso, pero es realmente una herramienta de programación. Incluye servicios para la creación de una base de datos, servicios para la elaboración de menús, un generador de reportes, un sofisticado lenguaje de programación que es único para DataFlex. El ser único realmente no es un problema, aun si se programa poco DataFlex, porque la mayoría de las herramientas son realmente generadoras de código como para que un reporte pueda ser hecho a la medida con la modificación del programa generado. Un SQL diferente, el lenguaje de programación DataFlex es un lenguaje completo de programación apropiado para el desarrollo de aplicaciones con instrucciones de control de flujo, teclado, manejo de pantalla y soporta archivos convencionales tan bien como archivos de base de datos.

DataFlex es una base de datos relacional estricta como la definió Codd, y no usa un SQL o comandos orientados al manejo de tablas como Project o Join. Sin embargo, éstos comandos pueden ser implementados usando el lenguaje de programación. También, el servicio de generación de reportes maneja múltiples archivos relacionados sin la necesidad de escribir un programa. DataFlex suministra funciones de validación y verificación para usarlas con las opciones de entrada de datos. El apoyo al despliegue en pantalla incluye el manejo de ventanas múltiples y la construccion interna de gráficas para negocios.
Las capacidades de manejo de redes y multitareas de DataFlex están limitadas al bloqueo estandar de archivos, aunque DataFlex bloque todos los archivos activos a la vez. Una función de relectura tiene un bloqueo implícito con ella. La función es utilizada por la

naturaleza de DataFlex de registro a la vez.

Aunque los servicios de multitarea y redes de DataFlex pueden ser limitados comparados con otros productos, DataFlex es enteramente apropiado para el desarrollo de aplicaciones sofisticadas de software para el trabajo en grupo.

Sistema manejador de la base de datos.

El nicleo básico de operación del sistema manejador de DataFlex es el programa DFRUN.EXE. DFRUN está funcionando todo el tiempo durante el cual cualquier programa de DataFlex esté corriendo. Bajo el control de un programa DataFlex, el manejador de DataFlex puede accesar y manipular uno o más archivos de la base de datos, interactuar con el operador, manejar la impresora, leer y escribir archivos en ASCII estándar, y cualquier otro que se pueda activar con los sistemas de microcomputadoras. Estas actividades son controladas por los programas de DataFlex (archivos con la extensión ".FLX") creados y compilados por el desarrollador. El comando DFRUN progname *ejecutado desde el sistema operativo, ejecuta al manejador de DataFlex, el cual en turno ejecuta el programa llamado* **progname.**

Cuando es inicializado, el sistema manejador de DataFlex primero lee un archivo llamado FILELIST.CFG el cual contiene información acerca de que archivos de la base de datos están disponibles, y después otro llamado TERMLIST.CFG, el cual contiene información que habilita a DataFlex para generar los códigos de control que le permita operar la terminal en particular.

Cada uno de los nombres de los archivos de la base de datos aparecen en FILELIST.CFG

que normalmente se refieren a una "familia" de archivos relacionados en el disco, todos teniendo una ruta de acceso en común, pero diferentes extensiones. El archivo de datos (extensión ".DAT") contiene los datos y la información básica de la estructura del archivo de datos. El archivo de llaves (extensión ".K#" donde el simbolo # es el número del índice) contiene los datos indexados los cuales permiten acceso a los datos mediante los índices establecidos por el programador durante el proceso de definición del archivo de la base de datos. El archivo de etiquetas (extensión ".TAG") contiene los nombres de los campos definidos para el archivo de la base de datos.

El mínimo conjunto de archivos que se requieren para correr DataFlex consiste de: DFRUN, FILELIST.CFG, TERMLIST.CFG, y al menos un archivo .FLX creado previamente por el programador y todos los archivos (.DAT .K# y .TAG) para cualquier base de dato(s) declarada en (usada por) lo(s) programa(s) DataFlex. Para usar la utilería de preguntas de DataFlex los archivos DFQUERY deben de estar presentes también.

El sistema de Menú de DataFlex que saluda al usuario cuando DataFlex es inicializado, es por si mismo un programa hecho en DataFlex. El nombre del programa menú es MENU.FLX. El servicio del menú mantiene los menús en una base de datos estándar de DataFlex cuyo nombre es MENU.DAT.

Se proveen otros programas que pueden ser requeridos ocasionalmente al momento del funcionamiento del manejador de la base de datos. La utilería de reindexación de DataFlex (DFINDEX) lee un archivo de datos y reconstruye las llaves (archivos .K#), verifica (y la reconstruye si es necesario) la lista de registros borrados en un archivo de la base de datos, y borra o extrae cualquier registro que aparezca dañado (típicamente como una consecuencia de una falla en la energía, en el medio de almacenamiento o en el sistema de cómputo). La reindexación se utiliza también para mantener índices BATCH y crea índices "ad hoc". Este programa puede mantenerse fuera de línea hasta ser necesario.

Desarrollo de aplicaciones.

El proceso de desarrollar aplicaciones especificas en DataFlex involucra la creación de programas DataFlex (con extensión .FLX) y archivos de la base de datos para almacenar y manipular datos de la manera deseada. Estos archivos son creados utilizando un grupo de programas incluidos con DataFlex. Los archivos .FLX son creados al correr el compilador de programas de DataFlex sobre archivos "fuente". Los archivos fuente pueden ser creados de muy diversas maneras, se incluye con DataFlex un programa editor (DFEDIT), o cualquier editor de textos o procesador de palabras con los cuales se puedan generar los archivos de texto ASCII planeados. Los archivos fuente contienen toda la variedad de comandos de DataFlex los cuales causan que el sistema entregue la salida deseada de un proceso.

También se pueden producir programas usando la utilería Auto-Create (DFAUTO). Auto-Create también crea la definición de archivos nuevos de base de datos, los datos y los archivos llave si son necesarios para esos datos. Para utilizarlo el primer paso es "dibujar" una imágen de la entrada y la pantalla de preguntas que se utilizará en la nueva base de datos. El editor de DataFlex o el editor de textos de su preferencia sirve para este propósito. Auto-Create lee este archivo y produce un archivo de definiciones (.FD) y un programa con código fuente de DataFlex (con la extensión .FRM), listo para ser compilado. El programa de DataFlex producido por Auto-Create puede dar mantenimiento total al archivo de la base de datos creada.

La función del programa editor es crear, desplegar y modificar programas fuente de DataFlex. Los archivos fuente pueden ser creados desde cero con el editor de DataFlex (DFEDIT), o pueden venir desde una de las utilerías generadoras de programas tales como Auto-Create, Query o Read.

La utilería de preguntas de DataFlex (DFQUERY) permite al usuario fácil acceso a la información almacenada en los archivos de la base de datos. Esta utilería puede utilizarse como un generador de reportes ad hoc que requieren poco adiestramiento para usarse. Query es enteramente automático en su operación, realizando preguntas prácticas al operador, esto permite almacenar los criterios de las preguntas en un archivo como código fuente para un programa DataFlex reporteador. El archivo fuente puede ser compilado para tener un reporte permanente. DFQuery requiere que todos los archivos de la base de datos esten presentes.

El programa READ de DataFlex genera un programa fuente para importar archivos de datos en ASCII estandar. READ primeramente es creado para aquellos usuarios que necesitan convertir datos desde otros sistemas a DataFlex.

La utilería de reindexación de DataFlex (DFINDEX) detecta caracteres que no sean ASCII en los archivos de datos e identifica en los registros que tengan evidencia de datos corruptos para removerlos o corregirlos, permite la manipulación de indices ad hoc de cualquier archivo de la base de datos, y suministra servicios para cambiar las definiciones de los indices y la reconstrucción de los archivos índice después de que han sido dañados por problemas de energía, acciones inapropiadas de programas DataFlex o algo semejante.

El compilador de DataFlex (DFCOMP) el cual produce archivos .FLX desde los archivos fuente requiere que esten presentes tres archivos en el directorio, en orden para hacer su trabajo: el archivo fuente del programa DataFlex, un archivo de definición (extensión .FD) para cada archivo de la base de datos declarada en (usada en) el programa DataFlex, y FLEX.CFL, el archivo de comandos de DataFlex. En los archivos .FD y FLEX.CFL deben ambos recidir en el mismo disco. Una buena compilación produce un

programa DataFlex ejecutable con la extensión .FLX y la misma ruta del archivo fuente. Cuando el compilador está corriendo, desplegará en la pantalla cualquier mensaje de error que ocurra durante la compilación.

Un programa DataFlex que no utilice ningún archivo de la base de datos no requerirá ningún archivo .FD en la compilación, pero el programa DataFlex típico accesa uno o más archivos de la base de datos. Los archivos .FD son usados solo al tiempo de compilación; ellos no son requeridos al momento de correr. Dicho de otra manera, los otros archivos de la base de datos (.DAT, .K#) no se requieren en la compilación. Esta diferencia en los requerimientos de archivos puede ser aprovechada cuando se opera con espacio reducido en el disco.

La utilería para la definición de archivos (DFFILE), como Auto-Create, puede crear definiciones de archivos de base de datos. Dentro de su función, el archivo de definición, es más versátil que Auto-Create. Con el archivo de definiciones, se pueden crear estructuras relacionales de muchos archivos. Utilizar todas las capacidades del archivo de definiciones requiere de un conocimiento más profundo de las capacidades de DataFlex. El archivo de definiciones puede también leer y escribir versiones de archivos de definiciones de bases de datos en ASCII estándar. Las versiones ASCII de la definición de archivos es útil para referencia, documentación de los archivos de la base de datos y transferencias a otros medios ambientes.

La utilería Empaquetadora de DataFlex (DFPACK) es para ser utilizada por aquellos usuarios que deseen crear sus propios comandos y agregarlos al lenguaje de DataFlex. DFPACK lee el archivo FMAC y crea el archivo FLEX.CFL el cual es utilizado por el compilador. Se pueden agregar nuevos comandos con un editor de texto o mediante un procesador de palabras y pueden hacerse usables al correr DFPACK sobre la nueva

versión del FMAC, creando una nueva versión de FLEX.CFL (sobreescribiendo la anterior).

Después de correr el programa de puesta apunto e "informar" a DataFlex de los códigos que operan la terminal, hay tres niveles básicos de usar DataFlex a través de los cuales podremos proceder en orden de facilidad, hasta ir adquiriendo experiencia. A continuación se describen éstos niveles :

Preguntas y Reportes. *La Utileria de Preguntas de DataFlex (DFQUERY) permite extraer información rápida y fácilmente de la base de datos de DataFlex y al mismo tiempo, provee un eficiente y poderoso "ad hoc" servicio de preguntas para el programador o para un operador avanzado para confirmar el contenido de la base de datos. DFQUERY puede formatear datos automáticamente desde un archivo de la base de datos, y puede extraerlos selectivamente y de acuerdo a las especificaciones enteradas en el acto. La operación de DFQUERY es completamente interactiva y no-técnica.*

Caracteristicas adicionales del DFQUERY. Incluye opcionalmente totalizadores de campos numéricos y un juego completo de seleccionadores lógicos (menor que, mayor que o igual a, etc.). Hasta diez criterios de selección estan permitidos.

Una vez que un Query en particular ha sido terminado, se puede regresar y editarlo antes o después que se haya producido su salida. Al mismo tiempo, se puede elegir "mantener" el Query dentro de un formato tipo Query asignándole un nombre. Para repetir el Query en cualquier otra ocasión futura, para escoger el nombre asignado, DFQUERY presenta una lista de Querys cada vez que se inicializa.

DFQUERY también ofrece la opción, de escribir un archivo de código fuente del reporte el cual puede ser compilado y ejecutado como un programa DataFlex. La principal ventaja

de mantener los Querys dentro un formato Query es que pueden ser modificado al generar código fuente para producir literalmente cualquier acción de reporteo del cual es capaz DataFlex.

Aunque DFQUERY puede generar programas para reportes, DFQUERY no pretende ser un generador de reportes completo. La macro REPORT (con la cual DFQUERY genera sus programas) es el servicio central que DataFlex emplea para la escritura de reportes, y en orden para crear sofisticados, formateados, reportes multi-archivo con control de operador, es necesario familiarizarse con los servicios de la macro REPORT.

Las salidas del DFQUERY son independientes de los dispositivos, esto significa que puede ser dirigida a la pantalla, a la impresora o a un archivo en disco. Al direccionar la salida del DFQUERY a un archivo en disco, los datos son almacenados en el formato ASCII así que pueden ser editados o leídos por otros programas.

DFQUERY contiene numerosas pantallas de ayuda de contexto sensitivo las cuales pueden ser llamadas desde cualquier punto del DFQUERY al presionar una simple tecla. El DFQUERY es muy sencillo de ser accesado por el operador ya que virtualmente la documentación lo explica por sí mismo, tanto como un servicio de auto-explicación, sencillo de usarse en la mayoría de las situaciones por un operador que no tiene acceso al manual de operación.

Creación y Modificación de Menús del Sistema.

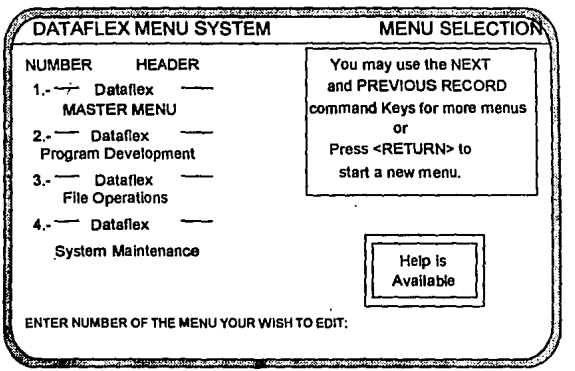
DataFlex está provisto con un sistema muy poderoso, versatil y fácil de usar para la creación y mantenimiento de menús los cuales pueden ejecutar desde un programa DataFlex otros programas ejecutables y pueden aceptar entradas de datos del operador al momento de ejecución. El sistema de Menús puede realizarse para ser ejecutado con el

arranque del equipo, suministrando un control más seguro del mismo y aislando positivamente al operador del sistema operativo de la computadora en todo momento.

Esta utilería es por si misma un programa hecho en DataFlex (con nombre MenuDef.flx), usa y crea archivos de datos para los menús bajo el nombre de "MENU" en el DBMS. Como cualquier aplicación "Flex", MenuDef es totalmente interactivo, y los menús pueden ser creados o modificados fácil y rápidamente, siguiendo simplemente los pasos indicados y respondiendo a sus preguntas. MenuDef también tiene miltiples pantallas de ayuda, las cuales pueden ser llamadas en cualquier momento durante la ejecución presionando la tecla <F1> (Help Flex-Key).

Creación y Seleccionado de Menús.

Una vez que invocamos MenuDef desde el sistema operativo o desde algún otro menú desde el sistema de mantenimiento de menús, una lista de nombres y encabezados de los menús que están actualmente en el disco, aparecerán como se muestran en la figura. Los "encabezados" son las leyendas que aparecen en la parte superior central de los menús identificando cada uno como se muestra en la siguiente figura:

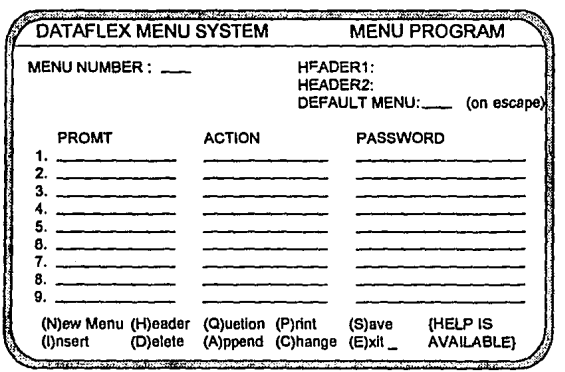


Definición de menú y pantalla de selección.

Esta pantalla despliega todos los menús existentes hasta ahora. Si hay más de cinco menús en el dispositivo, presionando la tecla [PgDn] (Next Record Flex-Key) desplegará los siguientes cinco menús. Presionando la tecla [PgUp] (Previous Record Flex-Key) desplegará los cinco anteriores. Cada menú está representado por un registro en el archivo MENU de la base de datos, y el número de menú desplegado es el número de registro en el menú. De acuerdo a esto el número dado a un menú no puede ser cambiado. Si se desea un nuevo menú, presionando la tecla [Enter] comenzará la creación de un nuevo menú. Para borrar un menú, el menú deberá ser seleccionado para edición y debe presionarse la función de borrado de registros (la tecla [F3]). Introduciendo el número de un menú existente moveremos la ejecución a la siguiente pantalla, con los datos llenados dentro de las ventanas en blanco para el menú seleccionado.

Edición de Menús Existentes.

Esta es la pantalla del programa para diseñar menús:



Definición de menú y pantalla del programa.

La primer columna de ventanas despliega cualquier opción ("prompt") que esté dentro del registro para el menú existente seleccionado. La columna desplegada bajo la palabra "prompts" son la pantalla de opciones que actualmente es desplegada al operador describiendo las opciones que tiene dicho menú.

La segunda columna despliega los comandos que son pasados al sistema operativo bajo la opción seleccionada. En "Actions" en la segunda columna, el menú del sistema ejecutará cualquier comando del sistema para la manipulación de archivos, como pueden ser: Erasefile, Renamefile, Copyfile, Directory, Memory, System, Rumprogram, Registration, Sysdate, Filelist, Filelist_next, File_size, Index_def, Make_file, Field_def, File_mode, etc.

También ejecuta otros menús por medio de la palabra de este comando (MENU). Al poner en esta ventana el comando CHAIN seguido por el nombre de un programa DataFlex provocará que el sistema de menú ejecute cualquier programa DataFlex que tenga este nombre.

La tercer columna despliega una palabra secreta (password), si es que la hay, para proteger las opciones del menú de accesos inautorizados. Como las palabras secretas son desplegadas por MenuDef, MenuDef debe asimismo tener su palabra secreta para proteger a las demás si se estan usando. Sin embargo quién accese a MenuDef tiene acceso a todas las palabras secretas del sistema.

El cursor es inicializado con el caracter blanco en la esquina inferior derecha de la pantalla. Si una "N" es enterada, la pantalla será limpiada y, entonces se redibujará la pantalla anterior (lista de menús) para la selección de un número no usado bajo el cual se creará un nuevo menú.

La opción "I" preguntará en que opción del menú se insertará la nueva opción del menú.

La opción "H" coloca el cursos dentro de los blancos en la parte superior de la pantalla, donde el encabezado del menú puede ser modificado. El encabezado incluye la definición del "menú por omisión", el cual es el número del menú que será ejecutado en el caso de que la respuesta sea nula (Return). Típicamente, el menú por omisión será el menú desde el cual el menú que está siendo definido fué llamado, pero esto no necesariamente necesita ser así, especialmente para el menú maestro, el cual típicamente su omisión es él mismo.

La opción "Q" sacará la pantalla temporalmente y desplegará las preguntas del menú en otra pantalla, la que será discutida más adelante. Estas preguntas consisten en entradas

que el operador debe proporcionar al elegir alguna de las opciones y que necesita pedir información adicional para poder completar la operación, tal como "Presione Return cuando la impresora esté encendida y con papel cargado".

La opción "C" preguntará que linea del menú deseamos cambiar, y coloca el cursor en los blancos para dicha elección, así que las modificaciones se pueden realizar.

La opción "P" permitirá imprimir la definición del menú, que es muy útil en ocasiones para referencias fuera de sesión de la estructura del menú.

La opción "S" causa la salida de la pantalla y escribe el menú editado al disco (realizando todos los cambios como permanentes).

La opción "E" causa que aparesca la siguiente pregunta: "Después de salir, desea salvar los cambios ? S/N".

Una vez que la pregunta es respondida, el sistema de definición de menús será abandonado, y el menú del cual fué llamado aparecerá.

Preguntas al Operador.

Las preguntas pueden ser contestadas por el operador, y las respuestas incluidas en la acción del menú, incluyendo los caracteres $#, donde el primer caracter es explicitamente el sigo de dólares, y el segundo representado por el signo #, es un digito del 1 hasta el 6, identificando el número de pregunta y comparando la pregunta definida en el menú con la pantalla de preguntas mostrada a continuación:

DATAFLEX MENU SYSTEM QUESTIONS

EXPLANATION: _____
QUESTION 1 : _____

EXPLANATION: _____
QUESTION 2 : _____

EXPLANATION: _____
QUESTION 3 : _____

EXPLANATION: _____
QUESTION 4 : _____

EXPLANATION: _____
QUESTION 5 : _____

EXPLANATION: _____
QUESTION 6 : _____

Enter the screen prompt for each question referenced in the ACTION section of the other screen, or ESCAPE to return.

Captura de preguntas del menú de definición y pantalla de edición.

Cada menú soportará solo seis preguntas, pero cada pregunta definida puede ser respondida tantas veces como se desee a lo largo del menú. Cada pregunta definida desplegará hasta dos líneas al operador, si es necesario, de una explicación introductoria seguida de la pregunta. El cursor se inicializa con el caracter blanco sencillo en la parte inferior derecha de la pantalla. La entrada del número para el cual la pregunta ya exista colocará el cursor en los blancos. La entrada de un número para el cual no hay pregunta colocará el cursor en los blancos de ese número para la entrada de nuevos datos. Los datos de una pregunta pueden ser borrados con la tecla de <BACK SPACE> Flex-Key. pero ello no es una razón para borrar tales datos sino hasta que el número sea necesitado para otra pregunta. El borrado de preguntas consiste en el remover las referencias a sus números desde las ventanas de acción de la pantalla de preguntas de menús.

Cada menú soporta solo nueve opciones, así que la selección puede ser elegida con la entrada de un caracter sencillo, y para mantener cada pantalla desplegada y sea fácil de desplegar búsquedas para el operador. Sistemas completos de Menús pueden construirse sin ningún problema considerando arreglos de submenús de grupos relacionados de funciones llamadas desde un menú principal.

Corriendo Otros Programas desde DataFlex.

Bajo condiciones normales (donde DataFlex es ejecutado por el comando "Flex" desde el sistema operativo). La ejecución de programas no realizados con DataFlex desde el menú causará que la ejecución retorne al menú. Si, de otra manera, se desea que la ejecución retorne al sistema operativo en lugar de al menú de DataFlex, se puede iniciar DataFlex con el comando "DFRUN".

El macro comando REPORT.

Dos áreas de actividad que consumen una gran porción del tiempo de desarrollo de aplicaciones de software son: (a) entrada de datos y el mantenimiento de archivos; y (b) salida de archivos para reportes, etiquetas, formas preimpresas y archivos. El macro comando ENTER de DataFlex suministra un servicio predefinido para la atención de los requerimientos de la primer área eficientemente; el macro comando REPORT realiza lo mismo para la segunda.

El macro comando REPORT consiste de un juego integrado de rutinas de salida predefinidas que pueden ser escogidas como se necesiten. Se pueden usar las rutinas seleccionadas con comandos DataFlex y argumentos relacionados con los requerimientos

para el "llenado" y la salida de la "imagen" y entonces enviar la imagen a un dispositivo de salida. La macro REPORT puede ser utilizada solamente una vez dentro de un programa. La macro REPORT es una conveniencia, utilizable para las más normales necesidades de reportéo. Esto no es una necesidad. También se puede hacer un reporte sin la macro REPORT.

De igual forma, la utilería DFQUERY de DataFlex genera programas REPORT, y una de las mejores maneras de aprender como trabaja el REPORT es "construir" el reporte en el Query y entonces estudiar el archivo de código fuente generado para el trabajo. A menos que se necesite un formatéo sofisticado o subtotales, es probable que el Query pueda satisfacer todas las necesidades de reportéo, y aun si no puede, Query es un excelente camino para empezar programa REPORT, con la producción de archivo fuente se puede editar el contenido para agregarle todo lo que se necesite.

Un programa REPORT está dividido en áreas mayores: líneas de imágenes y líneas de comandos. Las líneas de imágenes especifican los encabezados, ventanas interactivas, ventanas de formatéo, etc. Las líneas de comandos contienen comandos que instruyen a la macro REPORT como "llenar" los datos en las áreas de imágenes para salida.

Secciones del Reporte.

Las áreas de imágenes y comandos de programa REPORT contienen "secciones" correspondientes etiquetadas que direccionan diferentes requerimientos funcionales de un reporte. Por ejemplo, una de las imágenes es el "HEADER". La imagen "HEADER" define como se debe de ver la parte superior de cada página; tal vez el nombre de la compañía, el nombre del reporte, su fecha, numeración de páginas, secuencia de los datos del reporte, y así sucesivamente.

Los comandos en la sección del comando HEADER suministran cualquier procesamiento necesario de datos que sean posicionados y formateados en la sección de imagen del HEADER del reporte. Si hay datos en las ventanas de datos que sean llenadas desde variables del programa, tales como fecha de corrida o número de página, las líneas del comando HEADER "llenarán" los datos necesarios dentro de las ventanas. Cuando los datos han sido llenados dentro una sección, es enviada a la salida. Las imágenes etiquetadas de un macro comando REPORT son:

/HEADER
/SUBHEADER#
/BODY
/SUBTOTAL#
/TOTAL

El caracter diagonal (/) es parte de la sintáxis del nombre de la imagen. El "#" después de SUBHEADER y SUBTOTAL indican un dígito, del 1 al 9, que significan de que niveles (cortes de subtotales) pueden ser establecidos. El uso de las secciones y los nombres de ámbas secciones de imágenes y comandos es crítico. Cualquiera de los nombres puede ser seguido de la opción "RESIDENT", lo que causa que la imagen sea mantenida en memoria y no sea necesario rellamarla desde el disco cada vez que sea necesaria. Esta opción amplía la respuesta en tiempo cuando es aplicada a la imagen /BODY, o cualquier otra imagen que sea utilizada repetidamente, pero esto consume memoria, y puede ser no deseable en ambientes con problemas de memoria.

Subtotalizando y Totalizando.

La macro comando REPORT de DataFlex soporta hasta nueve niveles de subtotalizado. El SUBHEADER= y el SUBTOTAL= están ligados dentro del proceso de cortes de subtotales. Por ejemplo, SUBHEADER3 es la columna de subencabezados de encabezados y etiquetas que deben de aparecer al principio de la información subtotalizada en el tercer nivel de subtotal.

La sección TOTAL acumula totales acumulados durante la corrida para toda la salida y, si el programa está provisto de impresión de totales, lo imprime al final del reporte.

No es necesario utilizar todas las secciones de la macro REPORT en un programa. Solo aquellas secciones necesarias que necesite el reporte deseado deben de ser usadas. Toda sección de imágenes usada, sin embargo, debe tener incorporada un sección de comandos, aún si toda la sección de comandos realiza una salida de la misma imagen (utilice el comando OUTPUT).

Niveles de Corte de Subtotal.

Los subtotales se definen cuando se declaran campos de corte de subtotales en la línea de comando del REPORT. Los campos declarados deben de estar entre aquellos que conforman el índice nombrado en la línea del comando (el cual determina la secuencia del reporte). NOTA: si el índice no incluye los mismos campos que aquellos en los cuales están basados los cortes, en el mismo órden, el grupo de subtotales será incompleto, y los cortes ocurrirán aleatoriamente en el reporte.

Formato:

REPORT filename BY INDEX.# BREAK file.field file.field

File.field es usado en el formato de arriba para ahorrar espacio en los elementos significativos de filename.fieldname. La declaración file.field seguida del comando BREAK pone el corte al programa del reporte. REPORT asigna cada file.field un número secuencialmente según como sean declarados. Este número viene a ser el número correspondiente a la sección de comandos etiquetadas SUBHEADER# y SUBTOTAL# en el programa. El primer file.field viene a ser el corte 1, relacionando a SUBHEADER1 y SUBTOTAL1. El corte de más a la derecha (mayor numerado) representan los subtotales que cambian primero. El corte 1 cambia al último. El número máximo de cortes permitido en el subtotalizado en la macro REPORT es de 9.

Cuando ocurre un cambio en el campo nombrado como un corte, el grupo de comandos del SUBHEADER# y el SUBTOTAL# correspondientes a el número del corte donde el cambio a ocurrido, entonces todos los subencabezados y subtotales numerados mayormente serán ejecutados.

Cuando las selecciones y los cortes del REPORT son procesados, solo el registro del archivo nombrado en el comando es cargado dentro del buffer. Si las selecciones y los cortes están basados en campos de archivos a los cuales el archivo principal relaciona, un comando "RELATES df_filename" deberá ser incluido en la sección de selección (ver selecciones) del programa para traer los registros de los archivos relacionados dentro del buffer.

Como un efecto colateral de colocar un comando RELATE en la sección de selección, los

registros correctos en los archivos relacionados no estarán mucho tiempo disponibles en las secciones de subtotal. Esto es, ningún elemento de archivo deberá ser accesado o salvado en la sección del comando SUBTOTAL cuando haya un RELATE en la sección de selección. Para sobrellevar esta situación, los datos pueden ser impresos en las ventanas de la imágen de la sección SUBTOTAL desde la sección de comando SUBHEADER, donde los registros correctos están activos.

El formato ejemplificado abajo despliega todas las posibles secciones de programa con la macro REPORT. Todos ellos pueden ser usados, o solo aquellos que se necesiten para un requerimiento dado. Hay que tener en mente que este formato representa solo un porción de programa completo, y que las porciones del programa antes y después de la macro REPORT pueden pedir al operador la entrada de datos (tal como una fecha, o que reporte debe correr), y que los otros comandos antes del REPORTEND pueden envolver un ABORT, la selección de otro reporte, o cualquier otra cosa provista para un programa (excepto otra macro REPORT dentro del mismo programa).

/HEADER imagen del titulo del reporte
/SUBHEADER1 imagen del subencabezado 1
/SUBHEADER2 imagen del subencabezado 2
/BODY imagen del cuerpo del reporte
/SUBTOTAL2 imagen del subtotal 2
/SUBTOTAL1 imagen del subtotal 1
/TOTAL imagen del total del reporte
/SELECTION imagen de la pantalla de selección
*/**

OUTFILE

OPEN filename1 INDEX.#

OPEN filename2

MOVE # TO PAGEEND

REPORT filename1 BY INDEX.# BREAK filename1.field filename2.field

INDICATE SELECT AS data_element LT\LE\EQ\GE\GT SELECTION.#

SECTION HEADER ###

comandos

SECTION SUBHEADER1 ###

comandos

SECTION SUBHEADER2 ###

comandos

SECTION BODY ###

comandos

SECTION SUBTOTAL2 ###

comandos

SECTION SUBTOTAL1 ###

comandos

SECTION TOTAL ###

comandos

RPT.KEYPRESS: comandos de la subrutina

RETURN

REPORTEND

Selecciones.

Selección es una llamada a una especificación del reporte para un reporte de únicamente un subconjunto de todos los registros dentro del archivo de base de datos a ser listado. Por ejemplo, en cuentas por cobrar, se pueden desear imprimir solo aquellas cuyas cuentas tengan balances de más de 30 dias de antiguedad.

La macro REPORT está provista de selección de registros ha ser impresos a través de un indicador (variable booleana) predefinido, SELECT. Las selecciones son acompañadas de comandos INDICATE que activan al indicador predefinido del REPORT, SELECT, TRUE o FALSE de acuerdo a los criterios de selección del reporte. La sección de selección no es obligatoria si no se desea ninguna selección (todos los registros en la base de datos son reportados).

Los comandos de selección, si son usados, deben de ser colocados inmediatamente después de la línea del comando REPORT. Cualquier comando de DataFlex puede ser utilizado dentro de la sección de selección, pero los más comunes es una o más instrucciones INDICATE para activar SELECT. Si el valor en el registro corriente es puesto SELECT TRUE, el registro corriente será impreso. Si el valor es puesto a SELECT FALSE, el registro corriente será saltado.

Frecuentemente es deseable interactuar con el operador a través de la entrada de datos al momento de la corrida para establecer algún o todos los criterios de selección. Estas preguntas deberán ser hechas antes del comando REPORT. La respuesta a esas preguntas son entonces utilizadas en la sección de la selección en las instrucciones INDICATE.

Especificaciones generales de Dataflex (Provistas por Data Access Corp.):

Mantenimiento de archivos	*(en línea, interactivo)*
Actualización de archivos	*(en línea, interactivo)*
Entrada de datos	*((en línea, interactivo)*
Generación de reportes	
Query	*(en línea, interactivo)*
Lenguaje de comandos	
Mantenimiento a la base de datos	
Programa independiente del menú	
del sistema	
Librería del lenguaje C (opcional)	

Microprocesadores soportados:

8086, 8088, 80186, 80286, 80386, 80486 (Intel)

68020, 68030, WE32100, etc.

Sistemas operativos soportados:

MS-DOS/IBMPC-DOS, 2X, 3.X, IBM PC Netware, DOS concurrente 4.X, 5.X, 6.X, 3COM 3+ 1.1, Novell NetWare, TeleVideo InfoShare 1.1, CP/M-86, CP/M-86 Concurrente, Turbo DOS 1.4X, alloy Computer Products NT NX 1.6, Altos Xenix IBM AT Xenix, Microsoft Xenix, SCO Xenix Systems V, UNIX V para AT&T 6300+, 3B2 Series, Unix V para Torre NCR 32/400, Unisys 5000-30/50, VMS para DEC VAX.

Requerimientos:

384 Kb. de memoria RAM disponible para Dataflex

Monitor con cursor direccionable y con variación de intensidad (Highlighting)

1 Mb. de almacenamiento en disco

Propósito:

Desarrollo de aplicaciones

Estructura:

DBMS relacional extendido con utilerías independientes de los datos y lenguaje propio.

Número máximo de archivos en el DBMS:	*250*
Número máximo de campos por archivo:	*255*
Número máximo de índices por archivo:	*9 + 1 (AD-HOC index)*
Número Máximo de campos por índice:	*6*
Máxima Longituid por índice:	*256 Bytes.*
Tamaño máximo del archivo:	*Limitado por el sistema operativo*
Número máximo de registros por archivo:	*16.7 Millones*
Tamaño máximo del registro:	*16 Kbs. (en circunstancias normales de memoria de equipo)*
Tipo de creación de índices:	*B+ multi-nivel ISAM, secuencia de ordenamiento redefinible.*
Tipo de archivo de datos:	*Empacado, longitud fija, de acceso aleatorio.*

Formato de almacenamiento numerico:	*BCD empacado de punto fijo, de punto flotante..*
Precisión numerica:	*Punto fijo: 8 lugares después del punto decimal.* *Punto flotante: 16 digitos significativos.*
Rango Numerico:	*Punto fijo:* *+/ 99,999,999,999,999.99999999* *Punto Flotante:* *+/- 1^10+/- 306*
Número máximo de archivos abiertos simultáneamente:	*Limitado únicamente por la memoria.*
Número máximo de lineas por programa:	*5000*
Longitud máxima de linea:	*255 caracteres.*
Número máximo de ventanas	*2000 (en circunstancias normales de memoria).*
Número máximo de imágenes por programa:	*255 (residentes en memoria o en disco)*
Número máximo de variables booleanas (indicadores lógicos)	*89 (+ 38 predefinidos)*
Número máximo de variables :	*32000 (en condiciones normales de equipo)*
Número máximo de variables enteras:	*255*

Rango entero	*+/- 2,147,483,647*
Teclas especiales de Dataflex :	*22,independientes de la terminal.*
Niveles de corte para subtotales:	*9*
Archivos secuenciales:	*Delimitados por linea o por coma, ilimitado independiente del dispositivo.*
Caracteristicas del archivo de comandos:	*Código fuente protegido, semicompilado.*
Tamaño máximo del argumento:	*250 caracteres*
Tipos de argumentos:	*Cadenas de texto,* *Punto numerico fijo* *Punto numerico flotante* *Enterc* *Fecha (juliano extendido)*
Tipos de comandos:	*Conversión y movimiento de datos.* *Indicadores (condicionales)* *Cálculo.* *Control, no estructurado* *Control, estructurado* *Imágenes* *Entrada de Datos* *Reporteo* *Manipulación de la base de datos*

	Cadenas de texto
	Entrada/salida secuencial
	Entrada/salida por consola
	(independiente de la terminal)
	Macro pre-procesador
	Gráficas
Utilerías:	*Definición de la base de datos.*
	Recuperación de la base de
	datos
	Instalación del sistema
	y de la terminal
	Editor de texto
	Generadores de programas
Query de la base de datos:	*Menú configurable*
	Compilador de programas
	Exportación/importación de
	formatos externos
	Redefinidor opcional de índices
	secuenciales.

CARTA COMPARATIVA DBMS

	DATAFLEX	DBASE 3+	R:BASE FOR DOS	PARADOX	ADV. REVELATION	DATAEASE	KNOWLEDGEMAN/2	PROGRESS	CLARION
MULTIUSUARIO	7 AÑOS	3 AÑOS	2 AÑOS	1 AÑO	5 AÑOS	<2 AÑOS	4 AÑOS	4 AÑOS	<2 AÑOS
BLOQUEO NIVEL USUARIO	NO	SI	SI	SI	SI	SI	SI	SI	SI
NIVEL PROTECCION DATOS	CAMPO	REGISTRO	REGISTRO	CAMPO	REGISTRO	REGISTRO	REGISTRO	REGISTRO	REGISTRO
SOPORTE O/Ss LANs	50	7	11	10	10	6	2	8	1
RAM MINIMA	384K	384K*	512K*	640K	512K*	512K	512K	512K	512K
NO PC/MS-DOS	SI	NO	NO	SI	SI	NO	SI	SI	NO
CPM/86	SI	NO	NO	NO	NO	NO	NO	NO	NO
SISTEMA UNIX V	SI	NO	NO	NO	NO	NO	NO	SI	NO
VAX/VMS	SI	NO	NO	NO	NO	NO	SI	SI	NO
TAMAÑO MAXIMO REGISTRO	16K	4K	4K	4K	LIMITE O/S	4K	LIMITE O/S	2K	64K
MAX CAMPOS POR REGISTRO	255	128	400	255	32000	255	255	2K	LIMITE O/S
ARCHIVOS DE DATOS ABIERTOS	250	10	80	15	6000	32	45	LIMITE O/S	99
TRANSPORTABILIDAD	SI	NO	SI	SI	SI	SI	SI	SI	SI
INDICES EN LINEA POR ARCHIVO	9 + AD HOC	7	400	255	LIMITE O/S	255	LIMITE O/S	LIMITE O/S	99
INDICES DE MULTIPLES CAMPOS	SI	SI	SI	NO	SI	SI	SI	SI	SI
BUSQUEDA PARCIAL POR TECLA	SI	SI	NO	PARCIAL	SI	SI	SI	SI	SI
MANEJO DE MENU	SI	PARCIAL	PARCIAL	SI	SI	SI	OPCIONAL	PARCIAL	SI
MANEJO DE CANDADOS POR	DATAFLEX	PROGRAMADOR	R:BASE PARA DOS	PARADOX	ADV. REVELATION	DATAEASE	KNOWLEDGEMAN/2	PROGRESS	PROGRAMADOR
ALMACENAMIENTO REQUERIDO	1 MEGA	2.2 MEGAS	4 MEGAS	1 MEGA	3.5 MEGAS	<2 MEGAS	1.5 MEGAS	3.5 MEGAS	3.5 MEGAS
GRAFICAS INCLUIDAS	SI	NO	NO	NO	NO	NO	SI	NO	NO

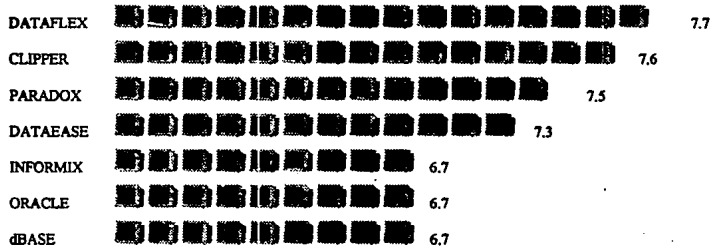
* *RAM Dedicada*

Algunas limitaciones de Dataflex pueden ser reducidas por el limite del sistema operativo

SELECIONADO DE :

SEMANARIO DE COMPUTACION 1991 -DATAPRO SATISFACCION GARANTIZADA

ORDENADO POR RANGOS (HASTA 10)



Las razones para la satisfacción del usuario incluyen el alto rendimiento de la base de datos, la facilidad de su uso y la velocidad de programación con lenguaje 4G y la portabilidad de las aplicaciones a través de DOS, DOS para trabajos en red (Novell, Banyan Vines, etc.), Xenix, Unix, Aix, Ultrix y Vax/Vms.

BIBLIOGRAFIA.

- *Ingeniería del Software. Un Enfoque Práctico.*
Roger S. Pressman
1989, editorial Mc Graw Hill, segunda edición.

- *Técnicas de Bases de Datos. Estructuración en Diseño y Administración.*
Shakuntala Atre.
1988, editorial Trillas, S.A. de C.V., primera edición en español.

- *Organización de las Bases de Datos.*
James Martin
1977, editorial Printice/Hall Internacional, primera edición.

- *Diseño de Bases de Datos.*
Gio Wiederhold
1983, editorial Mc Graw Hill, segunda edición.

- *Fundamentos de las Estructuras de Datos Relacionales.*
Rubén Adad, Miguel Angel Medina y Alfredo Careaga
1992, editorial Limusa, S.A. de C.V.,Grupo Noriega Editores (Serie Megabyte), primera edición.

- *Introducción a los Sistemas de Bases de Datos.*
C. J. Date
1986, editorial Addison-Wesley Iberoamericana, tercera edición.

Artículo con título: "What to look for in a LAN Database?"
Patrick H. Corrigan
Micro Systems Journal. Septiembre de 1988, pags. 18 a la 23.

Artículo con título: "Structured Query Language and its LAN Alternatives"
William Wong
Micro Systems Journal. Septiembre de 1988, pags.38 a la 45.

- *Artículo con título: "Manejo de Bases de Datos en Redes Locales"*
Neil Mayers.
Red. La Revista de Redes de Computadoras. 1990, año 1, número 2, pags. 43 a la 45

- *The Dataflex User's Guide.*
1990, Data Access Corporation, Rev. 2.3b

- *The Dataflex Encyclopedia.*
1990, Data Access Corporation, Rev. 2.3b