Universidad Nacional Autónoma de México

Facultad de Ingeniería

Desarrollo e Instalación de un Sistema Corporativo de Cuentas por Pagar.

T E S I S P R O F E S I O N A L

Que para obtener el Título de Ingeniería en Computación

p r e s e n t a n

LUIS DE LA MORA GONZALEZ
VICTOR MANUEL GARCIA ALBARRAN
MIGUEL ANGEL GARCIA YAÑEZ
JORGE PICAZO SALINAS
MARIA ELENA RAMOS ARIAS
FRANCISCO JAVIER RANGEL JIMENEZ

Director de Tesis
Ing. Luis G. Cordero Borboa

México, D. F. **1989**

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

I N D I C E

INTRODUCCION

El desarrollo de ésta tèsis profesional, nació de la necesidad de proporcionar una herramienta de apoyo a los departamentos de compras, servicios administrativos y cuentas por pagar de un consorcio cuyo objetivo principal es la elaboración de lubricantes, la cuàl permitirà la disminución de las cargas de trabajo tan grandes que tienen èstos departamentos, así como, un control de la información, permitiendo proporcionar en forma màs ràpida y precisa la misma con un màrgen de error mucho menor al que se tiene actualmente.

El sistema de cuentas por pagar que se presenta en éste trabajo tiene la capacidad de mantener registros exactos y detallados de las facturas y notas de crèdito del proveedor, desde el momento en que se reciben hasta que se pagan y se concilia el cheque; además de la función de pagos, el sistema distribuye las cantidades gastadas a la cuenta de contabilidad correspondiente.

Los objetivos principales del sistema son:

- Permitir el desembolso exacto y oportuno de los fondos para cubrir los gastos.

- Proporcionar la información de los requerimientos de efectivo.

- Maximizar el aprovechamiento de descuentos.

- Proporcionar la contabilidad exacta de los pagos.

El sistema tendrà las siguientes funciones de información:

- Mantenimiento al archivo maestro de proveedores.

- Catàlogos y consultas del archivo maestro de proveedores.

- Control y mantenimiento de las òrdenes de compra de materiales productivos (materias primas).

- Control y mantenimiento de òrdenes de servicio y compras (òrdenes de compras de materiales no productivos, òrdenes de trabajo y contratos de servicios).

- Afectación de cargos y abonos a la cartera de proveedores hasta llegar a la emisión de cheques para su pago.

- Revisión de las facturas presentadas por los proveedores en forma automatizada de acuerdo a las

entradas y recepciones efectuadas por el almacèn de materiales productivos y los departamentos de servicios administrativos en el caso de servicios y materiales no productivos.

- Informes de saldos de proveedores a nivel reportes y consultas.

El sistema de cuentas por pagar controlarà aproximadamente quinientos proveedores que surten cinco mil quinientos materiales productivos y mil materiales no productivos. Adicionalmente se tiene en promedio veinte òrdenes de trabajo y cuarenta contratos diversos mensualmente.

En el primer capitulo se abordan los temas sobre sistema de informaciòn que servirán de base para el entendimiento del desarrollo de sistemas. La finalidad es otorgar al lector la posibilidad de entender lo que se considera como buena informaciòn; que importancia tiene el tener un departamento de anàlisis de sistemas en una empresa y, finalmente el porquè del èxito o fracaso de los sistemas de informaciòn automatizados.

En el segundo capitulo se aborda la planeaciòn de un proyecto de anàlisis de sistemas de informaciòn. Hemos tratado de dar enfoque diferente al tema de planeaciòn en funciòn a la bibliografia existente, complementando esta fase. Discutimos en dos bloques bàsicos el tema; el primero trata del estudio de oportunidad donde se analiza el medio ambiente actual, el estudio de factibilidad donde se tratan los temas econòmicos tècnicos y legales y finalmente el anàlisis costo-beneficio que implica el sistema a desarrollar, el segundo bloque define el sistema proponiendo alternativas de soluciòn e indica como construir grupos de anàlisis estableciendo los puntos de control y herramientas a utilizar.

En el tercer capitulo se aborda la estimaciòn de costos del proyecto, lo cuàl es uno de los factores importantes en el desarrollo del mismo. Este capitulo explicarà la importancia de la estimaciòn de costos, los factores que contribuyen al costo de un proyecto y se establecen modelos del sistema a desarrollar. Para èste efecto se utilizan: las tècnicas de costeo por lineas de còdigo, esfuerzo por tarea, costeo automatizado, asi como la estimaciòn del costo de mantenimiento.

En el cuarto capitulo se abordan la administraciòn y control de proyectos. Se pretende establecer como llevar a cabo el mismo optimizando recursos y minimizando costos. Para este fin se identifican los problemas màs frecuentes en la administraciòn de proyectos y se establecen mètodos de control para realizarlos en tiempo.

En el quinto capitulo se aborda el anàlisis de sistemas de

información. Aquí se aborda el problema de analizar la información del sistema existente a base de modelos siguiendo alguna de las metodologías ya existentes para el desarrollo de esta fase. Teniendo como resultado un diagrama de flujo de datos del sistema, partiendo de éste último se pueden identificar las áreas que pueden automatizarse dentro del mismo.

En el sexto capítulo que es el del diseño, vemos lo referente a las transformaciones que sufre el diagrama de flujo de datos para poder obtener una carta de estructura donde se muestran en forma funcional los bloques que forman el sistema, permitiendo establecer criterios de valoración del diseño en términos del acoplamiento y la cohesión.

En el séptimo capítulo que es el de instrumentación se trata de una breve descripción de las bases de programación estructurada tales como: Evaluación de la eficiencia de la programación, valoraciones que se pudieran tener en la modularidad de la misma, estilo de codificación, estandares de codificación, documentación del sistema tanto interno como externo.

En el capítulo ocho se abordarán la verificación y validación del sistema las cuales nos indicarán que tan confiable y que tanto se cumplen los requerimientos del usuario en el sistema, para esto tenemos varias revisiones en cada una de las fases para evaluar que tanta concordancia hay entre una y otra en el desarrollo de sistemas. Para esto se forma un grupo de control de calidad, el cuál aparte de tener las funciones de revisar cada fase, tiene que elaborar un plan de control de calidad, un plan de verificación y un plan de pruebas de aceptación del sistema.

En el capítulo nueve se tratarán aspectos relevantes para llevar a cabo actividades que afectarán de buena manera la planeación de un buen mantenimiento, a efecto de conservar la calidad de nuestro sistema a través de ciclos sucesivos de modificaciones y actualizaciones que por necesidades tengan que ser efectuadas.

En cada uno de los capítulos del 2 al 8 se incluye al final de los mismos la aplicación de la teoría expuesta al caso práctico en cuestión, con lo que se demuestra la realización del proyecto objeto del presente trabajo.

CAPITULO 1 Conceptos sobre sistemas de información

1.1 Generalidades

Los sistemas de información han ganado una gran importancia por la enorme posibilidad que ofrecen a los organismos que los utilizan de conocer oportuna, veráz y confiablemente datos que les permitirán obtener mejor control o resultado sobre las actividades a las que se dedican.

1.1.1 Características de la información.

La información dentro de una compañía es más que datos o cantidades enormes de hojas con números, nombres y claves sin control. No es posible tomar una decisión en función a datos simplemente. Los datos deben de conservar ciertas cualidades que permitan tomar acciones adecuadas a los diferentes niveles de la compañía o bien al usuario final al cual la información va dirigida.

Primeramente, la información debe de tener un grado de actualización. El análisis que corresponda a la información que se revisa, tendrá un impacto diferente en función al día que corresponden los datos. Si se quisiera analizar cuales son los mejores proveedores de cierto artículo, no lo podríamos evaluar con información del año pasado, porque el mercado presentaba condiciones diferentes a las actuales.

La información actualizada debe ser confiable. Las fuentes de información de las que estamos tomando los datos deben de tener un grado de autenticidad tal que nos permitan tomar una buena decisión.

La información esta dividida en niveles: detalle y sumario. Un comprador deberá de recibir en detalle los artículos que se desean comprar para poder generar los movimientos correctos en las órdenes de compra, pero tendrá que emitir resúmenes por artículo de compra y proveedor para su jefe, y avisos de emisión de cheque para el área de finanzas. Cada uno de las diferentes personas que toman parte en el proceso requieren de diferentes niveles de información.

Debemos recordar que toda decisión debe ser efectuada con la la mayor cantidad de elementos posibles a fin de que ésta sea la mejor. Por lo tanto, una buena información, deberá ser aquella que dirija al que la analice a la mejor decisión posible, sin confundirlo ni agotarlo en el análisis.

Es necesario hacer notar que no es posible efectuar un análisis de un trabajo específico con información que, o bien genera ruido, o bien estorba. No podemos analizar el beneficio que genera el mantener un sistema de control de órdenes de fabricación, si tenemos revuelto aquellos productos de material no

productivo que están involucrados en la operación de la compañía pero no toman parte de los procesos de producción que interesa controlar.

Debemos de enfocar nuestra atención a desarrollar sistemas que permitan informar para que se razonen. Anteriormente, la experiencia y la intuición eran suficientes para lograr un buena decisión. Si añadimos los factores de razonamiento en función a una correcta información, los usuarios del sistema tendrán una mayor posibilidad de èxito.

1.1.2 Los sistemas en la organización

Una de los elementos más importantes que se debe de considerar al evaluar tècnicas administrativas para el control de una actividad empresarial, son los sistemas de información computarizado.

Un sistema de información computarizado es aquel que esta constituido por una serie de elementos, centrados en un computador, que manipulando datos de acuerdo a instrucciones de programación, que en conjunto forman actividades de proceso, logran, en el tiempo, automatizar un proceso repetitivo o un problema común.

Los procesos de control de información pueden ser manuales, manual con asistencia de máquinas mecánicas, con asistencia de máquinas electromecánicas y con asistencia de máquinas electrónicas. Esto es, si quisiéramos originar y registrar información manualmente, haríamos registros escritos o utilizaríamos tableros colgados. Si fuera manual con asistencia del mètodo electrónico, se puede utilizar cualquier periférico de entrada para originar la información, como puede ser una terminal en línea, lector óptico, lectora de barras, etc. y cualquier dispositivo interno o externo de almacenamiento para el registro de la información.

Hemos hecho mención en todos los casos de una asistencia y no de un control automático. Todos los sistemas de información, sea manual o con computador, requieren de la ayuda de un operario para que se realice. Un sistema computador requiere que se le alimenten los datos y posteriormente se le de la órden de procesarlos o bien que se le alimente el dato y lo procese en el momento de recibirlo, pero la mayor parte del tiempo, trabaja con la interacción del ser humano.

La ventaja de utilizar el computador para un proceso de administración, es el hecho de que puede manejar las operaciones de inmediato si se necesita, o a alguna hora en especifico según se requiera. También puede proporcionar información a la gerencia facilitando la toma de decisión por medio de las consultas. Por

ejemplo, la historia de un proveedor que determina: si ha entregado en el tiempo requerido, con buena calidad, etc.

Si se requieren técnicas especiales para lograr el control, un computador lo puede hacer. Si se requiere de planear el trabajo en una fábrica y rehacerlo de acuerdo a las condiciones del momento, de nuevo, el computador facilita el trabajo.

Los beneficios de tales controles, no necesariamente se presentan con el ahorro tradicional de recursos humanos; con frecuencia se involucran beneficios tales como eficiencia, mejor servicio al cliente y utilización más provechosa de las instalaciones.

Los dispositivos para las terminales y para el despliegue visual han sido diseñados especialmente para proporcionar una interfase hombre-máquina adecuada. Los programas están escritos de manera que un operador con conocimientos básicos de los computadores se pueda comunicar con la máquina y la máquina pueda verificar sus acciones y decirle de inmediato si ha cometido algún error.

Pero para un operador, con amplio conocimiento de los computadores y la habilidad para la programación puede utilizar el sistema generando programas y modificándolos desde una terminal.

Es asi como llegamos al área de sistemas. Un conjunto de personas con las mejores características anteriores deben de conformar el departamento que le dará el apoyo organizativo a una compañía, que finalmente le redituará en una optimización integral.

El departamento de sistemas es una entidad, importante dentro de una compañía, que debe mostrar las siguientes características:

- Capacidad de organización. La posibilidad de desarrollar la suficente confianza en la gente para poder organizarlos.

- Razonamiento lógico. Una de las cualidades de formación que es deseable desarrollar en el personal de sistemas es inferir logicamente problemas y abstraerlos a niveles de decision binaria.

- Capacidad de venta. El hecho de poder desprender a la mayoria de las personas de sistemas obsoletos o manualmente mecanizados es una tarea dificil y agotante. Se requiere de una capacidad de venta para poder convencer a los posibles usuarios de un sistema de información de las ventajas que los sistemas mecanizados con llevan.

- Capacidad de solucionar problemas. No solamente es deseable el poder abstraer problemas, sino también resolverlos. Actualmente continúan en el mercado grandes cantidades de personas que ofrecen sus servicios elaborando una política de venta que deslumbra a los posibles clientes o usuarios pero que no tienen la capacidad de finalizar un proyecto.

- Capacidad de enseñanza. Cualquier desarrollo por parte de sistemas enfocado a usuarios con o sin experiencia implican paciencia para enseñar como manejar el equipo y el software, así como mostrar las ventajas del sistema.

- Planeación. La planeación de recursos, costos y fechas de compromiso, son parte inherentes al Departamento de Sistemas; al menos para el Gerente. Para ello se requiere de experiencia, de buena información y valor para adquirir compromisos.

Determinar tiempos de desarrollo se basa en la experiencia, y posiblemente sea complejo. Para un grupo de sistemas que solo haya efectuado desarrollos en lote, quizá sea muy difícil desarrollar por primera vez un sistema interactivo y en línea. Si a la complejidad del sistema añadimos el tamaño del sistema, la posibilidad de encontrar problemas y no cumplir con los tiempos esperados es mayor; es por ésto, que la planeación de un sistema, como parte del desarrollo del mismo, es muy importante.

- Liderazgo. La capacidad de establecer los lineamientos necesarios para lograr un objetivo son necesarios pero no son suficientes, si no se cuenta con el líder que una los esfuerzas del grupo.

1.1.3 Causas del éxito o fracaso de un sistema de información.

El éxito de un sistema de información depende generalmente de la habilidad desarrollada en los ejecutivos para determinar los objetivos de un empresa y mantener un grupo de sistemas que los pueda llevar a cabo. Es por ello, que se debe utilizar un procedimiento metodológico que permita llevar un concepto de organización o toma de decisión, a la realidad.

Un buen sistema es aquel que al finalizar el desarrollo tiene las siguientes características:

- El sistema trabaja de acuerdo a las características establecidas a su inicio.

- Es rápido, eficiente y funcional.

- Al sistema se le puede dar mantenimiento fácilmente.

Las razones por las que un sistema puede fallar más comunmente son:

- Los usuarios no entienden las salidas que reciben.

- La información que entrega el sistema es demasiada para tomar una decisión en corto tiempo.

- El sistema no cuenta con las suficientes salidas por excepción, y se tienen que emitir grandes volúmenes de hojas para lograr el acceso a un dato en especifico.

- El sistema no se utiliza. Las razones pueden ser diversas: falta de tiempo, falta de motivación, falta de apoyo de la Dirección, etc. En cualquier caso, no se puede considerar exitoso un sistema, si no se utiliza, aún si funciona bien tecnicamente.

- No se le puede dar mantenimiento al sistema fácilmente.

- Es lento.

1.2 Obtención y Análisis de la Información

Una vez que se sabe que es un sistema de información, es conveniente conocer los diferentes sistemas de información y los elementos que los componen.

1.2.1 Evaluación de los sistemas actuales

Durante esta etapa evaluativa, pretendemos descubrir las principales causas del mal funcionamineto de la gestión administrativa del organismo en estudio

La primera parte de este estudio son los diferentes tipos de documentos existentes. Dado que normalmente se substituye un sistema manual por sistemas informatizados, el análisis de lo existente concierne generalmente a documentos que son utilizados por procedimientos manuales. Aunque es posible analizar los documentos de cualquier tipo de sistema de información. En general se pueden clasificar de la siguiente manera:

- Documentos fuentes o de entrada: Son los documentos de los cuales la información es tomada. Se indica de entrada porque son la base o el inicio de un proceso.

- Documento de resultado o de salida: Son los documentos

producidos por medio de tratamientos manuales o automáticos. Cuando se denominan de salida es porque son producto de un tratamiento de una apicación informatizada, generalmente.

- Documentos internos: Cuando los documentos solo sirven en el interior del organismo. Dentro de esta categoría se pueden subdividir como documentos de posición, utilizados en el mismo departamento en que son creados y documentos de relación que circulan entre los diferentes departamentos del organismo.

- Documentos externos: Cuando proceden de personas del exterior o van destinados a personas fuera del organismo.

Antes de estudiar cada documento por separado, se deben de inventariar. La finalidad de esta actividad es justificar la existencia de cada uno de los documentos, pues muchas veces es fácil encontrar documentos que existen y no son utilizados debido a una mala adaptación del documento, sobrecarga de un puesto de trabajo, demoras para la elaboración del mismo, etc. Asimismo, es conveniente estudiar las copias de cada uno de los documentos, pues muchas veces las copias de los documentos no corresponden a las necesidades de los usuarios. Al efectuar el inventario, se deberá elaborar otra lista con aquellos reportes que los usuarios desearían tener.

El análisis de los documentos en particular incluye:

- Revisar encabezados para encontrar aquellos que están caducos o no existen.

- Las columnas que están previstas pero que no se utilizan.

- Las columnas con información que nadie entiende.

- Las anotaciones manuscritas que se efectúan en cada reporte.

- Repetición de documentos.

- Documentos informales que circulan entre departamentos.

Todas estos problemas ocasionan confusiones y lentitud administrativa.

Es importante hacer notar que si las anomalías no se descubren en el estudio de lo existente, se descubrirán más tarde, pero ocasionando un replantemiento en las soluciones que ya habían sido tomadas.

El análisis de los archivos vigentes comprenden tanto a los de aplicación manual como de aplicación informática y deben de ser analizados como parte de un sistema. Los lineamientos generales para analizar los archivos manuales se dan a continuación, pero son solo un guía de una parte del análisis. El análisis integral se explica en el siguiente punto cuando se tratan las bases del análisis.

Como parte del desarrollo del análisis de una aplicación informática, se deben de analizar las formas en que se guardan los datos dentro de la organización.

Si se va a desarrollar un sistema informático, a partir de un sistema manual vigente, se deben analizar las siguientes características:

- Nombre del archivo. Es común que a un mismo archivo se le denomine de diferentes maneras.

- Localización física del archivo como puede ser el edificio, piso, departamento, sala, archivero, cajón, etc.

- El número de artículos que contiene el archivo en promedio, el mínimo y el máximo.

- Forma de clasificación en que el archivo se encuentra ordenado.

- Actividades dentro del archivo y responsables. Las actividades pueden ser consultas, mantenimiento y seguridad, así como los procedimientos actuales para hacerlo.

- Contenido y calidad del mismo.

De la misma manera que con los documentos, es fácil encontrar problemas con los archivos, pudiéndose citar los archivos informales que son realizados por algunos usuarios (no importa el nivel jerárquico) en cuadernos o agendas, o los archivos que no son actualizados y por lo tanto perdieron vigencia, o la carencia de un archivo, no importando que la gente conozca la importancia de tener cierta información bajo control.

Al analizar los archivos dentro de un aplicación informática es necesario efectuar los siguientes estudios:

- Verificar la documentación existente del sistema.

- Verificar la congruencia de datos y estructuras lógicas.

- Necesidades de introducción de datos a partir de nuevos

requerimientos por parte del usuario y las posibilidades de hacerlo. Se analizará también la flexibilidad del sistema y la facilidad de conversión a una nueva base de datos si así se requiere.

- Conocer la cantidad de mantenimiento actual al sistema es importante, puesto que es muy posible que la mayoría de los recursos del área de sistemas estén enfocados a ello.

- También es importante conocer cuales son las fallas mas grandes del sistema, y como se han manejado.

Todos èstos puntos deben de ser colocados dentro de una carpeta que irá acumulando información referente al sistema actual y su evaluación.

1.2.2 Bases de análisis

Para analizar los archivos de aplicaciones informáticas no es recomendable analizarlos como entidades aisladas, puesto que forman parte integral de un sistema de información vigente, por lo cual, es recomendable analizar el sistema como tal.

Al efectuar el análisis primeramente deberemos obtener una narrativa definiendo los alcances actuales del sistema con el fin de saber si las expectativas esperadas por el usuario se cubrieron con el mismo. Si no es así, el usuario deberá de explicar que falta o sobra en el sistema, pues esto es lo que conforma la solicitud del usuario (entendièndose como usuario aquella persona que solicite o utilice un sistema de informacion).

El segundo paso es definir un plan tentativo de análisis donde se establezca quièn hará el análisis, quienes son los usuarios, que importancia tiene el sistema para la alta gerencia, que otros trabajos similares se han realizado para la misma área solicitante, etc.

Como estudios colaterales a los anteriores se deben obtener una lista de usuarios, una lista del personal tècnico involucrado, y una lista de los sistemas existentes para la aplicación.

Las alternativas que propondrá el equipo de análisis como posibles soluciones primeramente serán presentadas al nivel jerárquico más alto posible, para que se apoye el proyecto desde el mejor nivel ya que èsto redundará en una mayor posibilidad de èxito.

Un organismo puede ser centralizado o descentralizado y esto mostrar un efecto sobre el análisis y desarrollo del sistema de información. A continuación se establece la forma de poder

analizar las estructuras de organización que finalmente componen las empresas; esto lo haremos mostrando gráficamente la red organizacional donde se muestren las unidades centrales y las dependientes.

Las organizaciones centralizadas son de estructura tipo árbol, según se muestra en la figura 1.2.2.1:

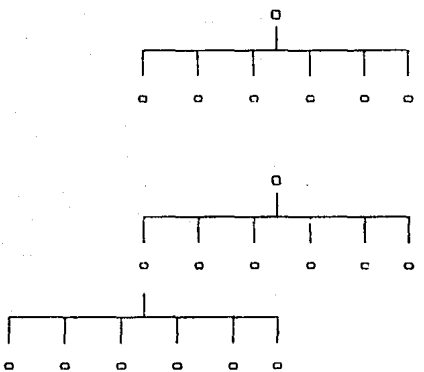


Figura 1.2.2.1 Organizaciones centralizadas a uno y dos niveles

Este tipo de estructura nos puede mostrar, en un momento dado a toda la compañía. El tipo de gráfica puede cambiar si así lo deseamos según se muestra en la figura 1.2.2.2:

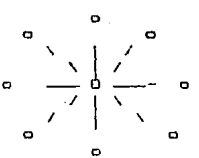


Figura 1.2.2.2: Organizaciones centralizadas tipo estrella

En todo caso, las líneas punteadas son uniones entre departamentos donde fluye la información, los círculos grandes simbolizan la entidad rectora, y los círculos pequeños los subordinados.

Las organizaciones descentralizadas pueden estar en forma de anillo o de malla lo cual se puede apreciar en la figura 1.2.2.3:

Fig.1.2.2.3: Organizaciones descentralizadas tipo anillo y malla.

Una vez que se han estudiado un número significativo de organizaciones podemos obrservar que hay un sinnúmero de opciones que se pueden realizar con estos diagramas v. gr. organizaciones centralizadas en primer nivel y organizaciones descentralizadas en el segundo, etc.

Es así que podemos definir en un nivel bajo que las unidades de organización son nodos y las uniones son arcos, lo que finalmente nos lleva a obtener un análisis funcional, que a partir de la red de funciones permite obtener una red orgánica que representará la estructura que finalmente debemos atacar asegurando el correcto tratamiento a datos fuentes o iniciales, emanados de los departamentos correctos y un flujo de información adecuado.

A partir de la red organizacional se pueden definir la concentración de datos y el tratamiento sistèmico que se otorgará.

En un nivel medio se tiene la red funcional la cual se define gràficamente como un rectángulo y significa la concentración de datos, su tratamiento y los medios de consulta a distancia.

La red funcional define la concentración de datos y tratamientos con la posibilidad de consulta o bien su reparto, de tal forma que se pueda definir que medio informáticos y de telecomunicación se requieren para soportar el sistema.

La red funcional permitirá definir la red orgánica que representará la estructura obtenida con los medios informáticos, asegurando la gestión de los datos y sus tratamientos.

El resúmen de lo anterior es el siguiente:

Nivel Mayor	Red Organizacional	Nivel Organización

Este nivel es el lugar dentro de la organización donde se requiere el sistema de información.

Nivel Medio	Red Funcional	Nivel Funcional

Flujo que seguirán los datos y el tratamiento del nuevo sisma de información.

Nivel Bajo	Red Orgánica	Medios informáticos utilizables en la aplicación.

Al elegir una solución debemos tomar en cuenta la factibilidad económica, técnica y legal, el análisis costo-beneficio, los medios de comunicación y los problemas humanos que representan siempre un reto para ser resueltos.

Es importante mencionar que las soluciones previas a un sistema de información pueden resultar en la reorganización de las estructuras del organismo o bien al mantenimiento de la solución actual sea o no informática.

1.2.3 Fuentes de información

El carácter de las fuentes de información varía con el sentido que queramos asumir en nuestro análisis.

Reconocemos dos tipos básicos como fuente de información para resolver el sistema.

La primera de ellas es la fuente interna, en la cual, la compañía presenta como documentación, reportes y procedimientos seguidos para el desempeño del sistema actual.

Esta información a su vez se subdivide en información formal e información informal.

Se cataloga como información formal aquella que esta descrita como un proceso reglamentado por un procedimiento escrito y

autorizado por la dirección del organismo.

Las fuentes formales serán los manuales de procedimientos autorizados o aquellos documentos que se consideran oficiales, que bien incluso puede ser documentación en línea interactuado con los programas de aplicación en sistemas computarizados.

Se cataloga como información informal aquella información que de alguna manera se lleva a cabo sin ninguna reglamentación entre personal del mismo departamento o de departamento a departamento; el hecho de no estar reglamentada no significa que el procedimiento no sea válido.

Las fuentes externas de información son todas aquellas posibilidades de información que ofrece la experiencia de otras personas sobre el mismo tema a desarrollar que nos ocupe.

Las fuentes externas de información se dividen en referencias nacionales similares, referencias internacionales ubicadas en la realidad nacional, revistas especializadas en el ramo, tratados sobre el tema y sistemas informáticos ya desarrollados sobre el tema.

1.2.4 Entrevistas

Uno de los elementos más importantes en la resolución de problemas sistémicos es la habilidad con la que cuenta el analista del sistema para acercarse al usuario mediante cualquier técnica de expresión, puesto que debe saber como redactar una carta, una documentación, un informe, saber organizar mesas redondas y conducir entrevistas.

La entrevista es uno de los medios más importantes en la captación de datos y será de suma importancia para las dos partes, pues es donde los usuarios establecerán sus necesidades y los profesionales de la informática deberán de convencerlos con argumentos sólidos sobre las ventajas sustanciales con respecto a la operativa actual.

La primera entrevista será probablemente la más importante, dado que es muy probable que se trabaje con expertos en el área como usuarios. Las entrevistas las debe de conducir el personal de sistemas con mayor experiencia. Esto es con el fin de poder establecer y fundamentar correctamente los argumentos presentados por el área de sistemas como alternativas de solución o bien para realizar entrevistas productivas.

Las entrevistas con la alta dirección deben estar dirigidas a entender las políticas, objetivos y estrategias generales, con el fin de poder captar una imágen general sin llegar al detalle.

El fin básico de la entrevista es poder localizar áreas de ahorro, puntos de control, posibles auditorías y en general los resultados que se esperan empleando nuevas técnicas administrativas.

Al nivel de ejecutivos medios, estamos entrevistando al grupo de gente que operativamente llevan las riendas de una organización, y por lo tanto, se verán mayormente afectados con un cambio de sistema. Es importante hacerles ver que el cambio de un sistema poco flexible y donde información poco accesible probablemente no se pueda entregar, por un sistema con características altamente operativas, les brindará enormes beneficios. Es generalmente gente con mucho sentido práctico, que requieren resultados inmediatos.

El nivel operativo nos sirve para encontrar referencias entre lo que está documentado como un procedimiento formal, las actividades que reporta la gerencia del área y lo que realmente esta ocuriendo.

Las tres fases del procedimiento general denominado entrevista son: pre-entrevista, la entrevista propiamente dicha y la post-entrevista.

La primera fase incluye seleccionar al interlocutor, establecer contactos previos a la entrevista y preparar la entrevista. La selección del interlocutor incluye su nivel dentro de la compañía. En un principio nos interesará solamente entrevistar aquellos que dirigen la empresa, pero conforme avance el proyecto nos interesará entrevistar a aquellos que conozcan los problemas desde puntos de vista prácticos y generales, por lo que probablemente entrevistemos a gerentes de área y jefes de oficina.

Es muy importante hacer notar que nos interesa que participen en el proyecto personas que muestren motivaciones para colaborar. Es preferible dialogar con el personal que coopera aún que èste sea medianamente competente y no con personas altamente competentes pero sin motivación en el proyecto y por lo tanto que no muestran cooperación.

El primer contacto para llevar a cabo la entrevista es con el jefe de aquel que será nuestro interlocutor. Una regla de oro en la participación dentro de las organizaciones es el hecho de respetar jerarquías. Una vez obtenido el permiso, se le debe notificar con anticipación al interlocutor; para que pueda programar sus actividades y posiblemente preparar la entrevista.

La preparación de la entrevista se basa en la lista de puntos abordar durante la misma. Se recomienda ir de puntos generales a puntos particulares y no debe ser imperativa, si no irse adaptando conforme la plática se va llevando a cabo. Es importante tomar en cuenta los deseos del interlocutor y manejar la entrevista de manera que sean las motivaciones del interlocutor lo que es-

tablezca los puntos a tratar; siempre cubriendo la lista preparada.

La segunda fase debe de conducirse bajo lineamientos generales que la lleven a feliz término.

Generalmente se inicia con la presentación, precisando que ud. va a tomar notas de lo que se platique para posteriormente, hacerle llegar a él y a su jefe una minuta, que es una relación de la plática. Pregunte cuanto tiempo le tiene disponible para la entrevista y proceda a ella. Debe comenzar recordando el objeto de la entrevista; preguntando asuntos generales relativos al sistema en cuestión. No es permisible que la plática salga del tema. Si ocurre, lo cual es muy probable, no lo interrumpa y aparezca atento. Esto romperá el hielo y permitirá un mejor conocimiento de la persona, pero trate sutilmente de retomar el tema.

Trate de que el interlocutor hable en forma general de su trabajo y no efectúe preguntas duras o directas evitando realizar una entrevista pregunta- respuesta. No utilice términos técnicos. Es mal visto y aparenta que usted trata de impresionar. No olvide tomar notas discretamente y trate de no superar el límite de tiempo que su interlocutor le ha concedido.

La última parte de la entrevista es la post-entrevista, que se utiliza para redondear el tema, completando las notas de la entrevista lo más pronto posible. Se tienden a olvidar detalles importantes de no hacerlo. Recuerde que usted dijo que iba a enviar una minuta sobre los puntos y acuerdos importantes de la entrevista, por lo que debe enviarla. Puede enviarla junto con documentos si es que usted quedó de enviar alguno y cuando entregue la minuta al superior jerárquico de la persona que usted entrevistó agradezca la calidad de entrevista de su interlocutor.

CAPITULO 2 Planeacion del Proyecto

2.1 Estudio de oportunidades

La etapa de planeación de un sistema es importante pero no debe de consumir el total de los recursos dedicados al desarrollo de un sistema, sobre todo el tiempo. No es aconsejable de igual modo, comenzar un desarrollo de un sistema sin tener un idea adecuada sobre lo que éste debe hacer puesto que encaminaríamos al proyecto al fracaso consumiendo todo tipo de recursos.

El ciclo de planeación comprende la definición, el análisis y por último la revisión.

En cada caso se requiere evaluar el sistema desde los siguientes puntos:

- Evaluación de conceptos sobre el sistema.
- Análisis de factibilidad
- Análisis costo-beneficio.
- Definición del sistema.
- Definición de necesidades de equipo de cómputo, de programación y el equipo de usuarios a participar.
- Definición de costo y tiempo.

Para delimitar el tiempo necesario que nos tomará la etapa de planeación del sistema debemos considerar un 20 % del tiempo total destinado al proyecto, y dado que planear un sistema no lo puede realizar cualquier persona, se recomienda que lo lleve a cabo un analista con amplios conocimientos del tema con una supervisión estrecha de la gerencia, del usuario solicitante y del personal de desarrollo. Se debe tomar en cuenta que durante esta etapa una omisión en la comunicación podría generar un problema mayor posteriormente.

2.1.1 Análisis de lo existente

Antes de comenzar el desarrollo, debemos de evaluar algunos puntos importantes sobre el sistema de información actual que nos proporcionarán una idea detallada de lo que es, por que esta implantado, quienes son las personas encargadas de su uso, así como las personas encargadas de su mantenimiento y desarrollo.

Los siguientes puntos pretenden llevar a la persona que realiza el análisis a un estudio coherente y estructurado del análisis del sistema existente en cuestión.

- Al evaluar un sistema de información, se debe observar si los usuarios y el equipo técnico llegan a los mismos comentarios sobre el sistema, porque puede ser que o bien no se comprenda la metodología de trabajo o bien se pierdan de vista aspectos importantes de la evaluación por algún miembro del equipo que la efectúa, por lo tanto se deben discutir los puntos donde haya discrepancias hasta que éstos sean resueltos.

- Se deben evaluar todas las metodologías de trabajo que se hubieran utilizado en otro momento como alternativas al sistema actual. Esto es con el fin de comprender el por qué está implantado el sistema actual.

- Se deben analizar otros procedimientos que estén siendo utilizados en organizaciones similares con el fin de evaluarlos y comprar los pros y contras del sistema actual.

- Los costos operacionales así como todos los desarrollos, mantenimientos y esfuerzos que se lleven a cabo en ese momento, también hay que incluirlos.

- Se deben identificar a las personas que desarrollaron y a las que soportan el mantenimiento del sistema.

- Asimismo es conveniente incluir una lista de las deficiencias y aciertos del sistema.

De esta manera, al final del análisis obtendremos la historia cronológica del sistema, sus usuarios, las ventajas y desventajas que presenta, las alternativas que hay en el mercado, y un conjunto de documentos fuentes y objetos, así como cualquier información adicional concerniente al sistema.

2.1.2 Crítica de lo Existente

El segundo paso para comprender y evaluar un sistema es analizar en detalle los elementos y estructuras que lo conforman. Este estudio, de nuevo nos llevará a obtener información que iremos anexando a los estudios previos que hemos efectuado.

Dado que es un estudio más profundo, porque analizaremos archivos y su manejo, es importante, de nuevo, que la persona que lo realice sea alguién con experiencia en análisis y desarrollo.

- La documentación que normalmente se encuentra disponible es escasa. Deben existir dos tipos de documentación, la destinada al usuario que le dice desde que hacer cuando se situa

frente a la terminal, lo conduce por el sistema hasta que logra realizar lo que desea y finalmente lo conduce fuera del mismo, y la documentación técnica, que normalmente es destinada al departamento de sistemas y mantiene actualizada la información sobre datos, archivos, formas, procedimientos, etc.

- Si se ocupa una base de datos se debe analizar si ésta es una base de datos limpia, si los elementos lógicos son consistentes, si no hay redundancia con otros sistemas de información, la flexibilidad de la base para integrar las nuevas necesidades, la dificultad para convertir la base de datos en una nueva y el número de ocasiones anteriores que se modificó la base actual.

- Como resúmen se debe obtener los formatos de los archivos, la evaluación de la base de datos que contendrá los comentarios sobre que tan limpia se encuentra, los errores que presenta, las áreas sin utilización, redundancia de datos, los problemas de conversión y los datos y estructuras que son más utilizadas por el sistema actual.

- Si se ocupa un banco de datos tradicional, se debe analizar si ésto es un banco de datos limpio, si los archivos están correctamente diseñados, si no hay redundancia con otros sistema de información, la flexibilidad del sistema para integrar las nuevas necesidades, la dificultad para convertir el sistema en uno nuevo y el número de ocasiones anteriores en que se modificó el banco actual.

Como se puede observar se obtiene el mismo resultado si se estudia una base de datos o un banco de datos, la diferencia normalmente no la detecta el usuario, las ventajas y desventajas entre uno y otro normalmente son de carácter técnico y van relacionadas a la facilidad de programación para el área de desarrollo.

2.1.2 Estudio de soluciones nuevas y objetivos perseguidos.

Debido al desarrollo tan grande que se ha hecho en el área de sistemas por casas dedicadas a la producción de software presentando alternativas globales o integrales en cada área, se recomienda una búsqueda intensiva de algún software que pueda satisfacer el requerimiento que nos ocupe.

Sobre este punto cabe recalcar el hecho de que el ahorro instalando un paquete o desarrollo existente, es evidente en varios aspectos, sobre todo en tiempo de programación dentro del área de sistemas.

Dado que un sistema puede ser adquirido para ser instalado en diferentes tipos de computadores, es extremadamente importante el hecho de revisar que el sistema que nos parezca el adecuado se encuentre disponible para el computador y lenguaje base que manejamos. Este comentario es importante sobre todo, si la evaluación del sistema es realizada por algún usuario.

Bajo este órden de ideas es también recomendable que cuando se instale un sistema, y dado que es probable que lo liguemos con otros desarrollos ya sean éstos desarrollos hechos en casa o comprados a terceros, se tenga en cuenta si debe residir en el computador principal o bien en microcomputadoras colocadas en lugares estratégicos dentro de la organización, para satisfacer la necesidad.

Si se instala en microcomputadores y no reside en el computador principal es muy probable que la información sea explotada a nivel departamental y el beneficio sea exclusivo de esa localidad. Si se instala en el computador principal, es relativamente fácil de integrar con otros sistemas, logrando un acoplamiento de información mayor.

Si no se encuentra una solución adecuada ya programada en el mercado debemos proponer otras ideas para el desarrollo del sistema. No debemos de olvidar que las otras posibilidades conllevan un costo siempre.

Si dentro del equipo que va a proponer otras ideas, no se encuentra una persona de sistemas con experiencia, es preferible acudir a compañías externas que se dedican precisamente a detectar este tipo de necesidades y proponer soluciones.

No se deber de olvidar que no vivimos aislados, y que es muy probable que el tipo de información que estamos evaluando lo manejen otras organizaciones similares, por lo que hay que comunicarse con ellas para observar que desarrollos han aplicado para la solución del problema.

Una vez que se han concluído todas las actividades anteriores, es muy recomendable recolectar la información de una nera comprensible y ordenada para que las gerencias involucradas puedan determinar que opciones son las mejores, incluyendo en el resúmen, si es posible, la respuesta a todas aquellas preguntas que pudieran haber surgido en el desarrollo, no olvidando el incluir, adecuadamente manejado, aquellos comentarios en contra que el mismo pudiera tener.

Dado que hemos concluido una parte primordial del nuevo desarrollo, es probable que tengamos comentarios sobre otros temas, que aunque no los trataremos a fondo, no los dejaremos de mencionar como bien pueden ser:

- Los lineamientos impuestos por el equipo de cómputo actual con las ventajas y desventajas del mismo.

- El algoritmo general, o por lo menos, un primer acercamiento al diagrama de flujo de datos del sistema.

- La relación que llevará el nuevo desarrollo con información existente como parte de un sistema general de control de la organización.

- Por último se debe considerar que el cambio tecnológico presente dentro del área de sistemas es lo suficientemente dinámico como para hacer obsoleto un sistema que no cumpla con ciertos requerimientos mínimos de desarrollo, por lo que ésto deberá de ser tomado en cuenta para lograr determinar la mejor opción de desarrollo posible.

2.2 Estudio de factibilidad

Todo proyecto es posible de ser realizado siempre y cuando no existiesen restricciones.

Las restricciones con las que se cuenta normalmente son de tipo legal, de tipo técnico y de tipo económico. Estas restricciones se verán estudiadas a continuación y el estudio resultante se denomina Estudio de Factibilidad.

El estudio solo se realiza en el caso en que se amerite; no tiene ningún caso efectuar uno donde la factibilidad sea obvia.

2.2.1 Factibilidad económica.

Es importante tomar en cuenta al factor económico cuando se presenta un proyecto. Es comunmente considerado el último factor de decisión, porque requiere de estudios complementarios para ser válido; nosotros pensamos que no es posible tomar una decisión sobre un proyecto del cual no sabemos su costo, por lo tanto es nuestro primer elemento de juicio.

Los estudios adicionales que se deben llevar a cabo son:

- Análisis costo-beneficio.

- Estrategia de desarrollo a mediano y largo plazo.

- Costo de los recursos necesarios para la elaboración del proyecto.

- Crecimiento potencial del mercado.

Acerca del análisis costo-beneficio se tratará ampliamente en un tema posterior. Acerca de los demás factores de juicio, debemos de recordar que toda aplicación de sistema de información, posee un ciclo de vida. Este ciclo de vida se define en tres fases: diseño, desarrollo y mantenimiento. Dado que es muy posible que nuestra organización se modifique en el tiempo, es necesario que la fase de diseño contemple un futuro correcto y se diseñe una base de información lo suficientemente flexible para poder soportar sin esfuerzo mayor los cambios de la organización.

Todo el costo del proyecto, se verá influenciado por aquellos gastos que incluso no se observan a simple vista; èsto puede ser desde el número del personal involucrado, sus sueldos (incluyendo prestaciones), el uso del computador, gastos de luz, teléfono, papelería, etc.

Para definir en concreto en que costos se incurren cuando se desarrolla un proyecto se definen seis posibles diferentes costos:

- De personal. Gasto de naturaleza directo. Se incluyen los salarios, prestaciones, tiempo extra, comidas, etc.

- De material. Gasto de naturaleza directo se incluye el mantenimiento, amortización y seguros de mobiliario tales como mesas, armarios, escritorios, CPU, etc, material de oficina, como máquinas de escribir, calculadoras, etc.

- De accesorios. Gasto de naturaleza directo e indirecto como papelería, èsto es formas para documentos fuente, formas de papel contínuo y lápices, plumas, gomas, etc.

- Del local. Gasto de naturaleza directo y/o indirecto, tales como el alquiler o amortización del local, calefacción, aire acondicionado, etc. y seguros.

- De prestación de servicios. Gasto de naturaleza directo, tales como la programación externa, asesoria externa y captura externa.

- De comunicaciones. Gasto de naturaleza indirecto, tales como el servicio de correo, servicio de teléfono y servicio de telex.

2.2.2 Factibilidad tecnica.

La factibilidad técnica se refiere a la posibilidad de desarrollo del sistema de información en cuestión.

No debemos de olvidar que es muy común el llegar a pensar que todo es posible cuando se presuponen ciertas premisas, pero es común que un desarrollo se detenga o se prolongue porque no todo lo relacionado con lo técnico es posible de ser realizado.

Este estudio se realiza al mismo tiempo que la etapa de análisis como una extensión del razonamiento analítico.

Consta de las consideraciones de riesgo de desarrollo, disponibilidad de recursos y disponibilidad técnica.

El riesgo de desarrollo se refiere al hecho de poder desarrollar un sistema que pueda cumplir con todas las condiciones determinadas en el análisis; va muy ligado al hecho de disponer de los recursos materiales y humanos para lograrlo y en cuanto a los recursos materiales, si es un sistema de información automatizado, un sistema de cómputo capaz de realizar lo que se propone como alternativa de solución.

La disponibilidad de recursos y tecnológica es muy importante puesto que no necesariamente podremos gastar ilimitadamente y ocupar el procesador que queramos; seguramente la organización que requiere el servicio pondrá a nuestra disposición presupuesto y computador específicos.

2.2.3 Factibilidad legal.

Por último la factibilidad legal se refiere a no cometer infracciones a códigos legales, ni a reglamentos oficiales, ni a las políticas de la organización. Generalmente se requiere de una opinión autorizada ya sea que ésta provenga de fuentes internas o externas a la organización.

2.3 Análisis costo-beneficio.

2.3.1 Análisis de beneficios

El análisis costo-beneficio depende principalmente del criterio con que se analice y las expectativas de ganancias que se contemplen. Ocasionalmente el análisis se refiere a efectos tangibles, pero la mayoría de las ocasiones se refiere a beneficios intangibles, difíciles de ser apreciados.

Para nuestro evento, se pretende obtener el beneficio de ins-

talar un sistema que controle las órdenes de compra emitidas por una empresa, controlando efectivamente la calidad de un producto, el flujo de efectivo necesario para su pago y la emisión de cheques oportunos para el pago de proveedores así como el registro contable de los mismos.

Actualmente la generación y control manual de las órdenes de compra generados por el departamento de planeación para material productivo y por toda la compáñía para materiales no productivos se realiza en forma manual.

Se entiende por material productivo, aquel que es necesario y/o forma parte de un proceso de manufactura.

Se entiende por material no productivo, aquel que es necesario para elaborar procesos administrativos; escritorios, plumas, lápices, etc.

El mayor costo que representa el control manual es la insatisfacción por proveedores que se ve representada por falta de material en las líneas de producción, problemas de crédito con proveedores afines, error en el cálculo de flujo de efectivo y falta de exactitud en los asientos contables.

Siempre para todos los sistemas de información se van a requerir tres parámetros en un estudio de costo-beneficio éstos son el costo real del sistema actual, el costo propuesto por el sistema nuevo, (se presume un ahorro), y el punto de ruptura, que determina cuando el costo propuesto en el tiempo mejora al costo real. Normalmente se calcula en función al tiempo, y también los efectos no necesariamento son inmediatos, sino pueden pasar un par de años para ver resultados.

2.3.2 Análisis de costos

Los principales factores que influyen en el costo del producto de programación son:

- Capacidad y confiabilidad del programador
- Complejidad, tamaño y confiabilidad del producto
- Tiempo y nivel tecnológico disponible

Un programador puede ser capáz, tener una inteligencia normal a superior y sin embargo no ser confiable. Los resultados de un programa deben de lograrse siempre a un 100% y no permitirle al programador un trabajo con fallas. Mientras más rápido se programe y con menos errores, el costo del sistema común será abatido.

Efectivamente el grado de complejidad de un producto se

canaliza siempre a un mayor o menor esfuerzo representando siempre mayor o menor costo.

Hay tres tipos de programas: programas que forman parte de una aplicación específica com bien puede ser una nómina o contabilidad, programas de apoyo a la misma programación, como bien pueden ser compiladores, editores, etc, y programas de sistema como bases de datos y sistemas operativos.

Los programas más sencillos son de aplicación, continúan con los de apoyo y terminan con los de sistemas. La relación es siempre tres veces mayor uno del otro, por lo tanto, se clasifican como 1-3-9.

Boehm B. en su libro Software Enginering Economics Prentice (Hall, Englewood Cliff, NJ, 1961), establece fórmulas para determinar el costo y tiempo de un proyecto según la clasificación anterior:

Para programas de aplicación establece la siguiente relación:

$$PM = 2.4*(KDSI)**1.05$$

$$TDEV = 2.5*(PM)**0.38$$

Para programas de apoyo considera que:

$$PM = 3.0*(KDSI)**1.12$$

$$TDEV = 2.5*(PM)**0.38$$

Referido a programas de sistema establece lo siguiente:

$$PM = 3.6*(KDSI)**1.20$$

$$TDEV = 2.5*(PM)**0.38$$

Donde PM significa el tiempo estimado en meses/programador de un proyecto, KDSI es el número de millares de instrucciones de código fuente de un producto y TDEV es el tiempo de desarrollo de un programa.

Las ecuaciones anteriores explican que la tasa de crecimiento de un proyecto en cuanto al esfuerzo requerido, aumenta con el número de instrucciones de código fuente.

En términos generales los proyectos de programación requieren más esfuerzo si el tiempo de desarrollo se reduce o se incrementa más allá de su valor óptimo.

Existe un límite para reducir el tiempo de un proyecto de programación no más allá del 75% del tiempo nominal calendario no importando la adquisición de personal o equipo.

La confiabilidad de un sistema definida como la probabilidad de un programa que ejecute una función requerida bajo ciertas condiciones especificadas durante cierto tiempo; puede verse afectada por factores externos e internos. Boehm de nuevo los define y les asigna factores de ajuste de esfuerzo:

Categoria	Consecuencias de la Falla	Factor
Muy Baja	Molestia Menor	0.75
Baja	Pérdidas fáciles de recuperar	0.88
Nominal	Dificultad relativa de recuperación	1.00
Alta	Gran pérdida financiera	1.15
Muy Alta	Riesgo de una vida	1.40

El nivel tecnológico al que nos debemos referir es a la posibilidad de trabajar con programas de apoyo, prácticas y herramientas adecuadas.

Cabe mencionar que el uso de lenguaje ensamblador en vez de un lenguaje de alto nivel disminuye la productividad por un factor de 5 a 10, además de no contar intrínsicamente con herramientas como autodocumentación, verificación de datos, manejo de excepciones y de interrupciones.

Boehm de nuevo aporta factores multiplicadores:

	Se utilizan	No se utilizan
Prácticas Modernas de programación	0.82	1.24
Herramientas de Programación	0.83	1.24
Lenguaje de alto nivel	0.82	1.24

La siguiente recomendación la consideramos de suma utilidad:

No se debe de añadir más gente a un proceso que ya va atrasado con el fin de que se actualice. Esto, probablemente, lo único que logre será que el proyecto se atrase más.

El siguiente ejemplo del libro de "Ingeniería de Software" de Roger S. Pressman en la pag. 84 lo ilustra:

Suponga usted a 4 ingenieros de software, cada uno capáz de producir 5000 lineas de códigos fuente al año trabajando cada uno por separado. Cuando trabajan juntos hay seis líneas de comunicación abiertas:

Ing. 1 y 2	Ing. 2 y 3
Ing. 1 y 3	Ing. 2 y 4
Ing. 1 y 4	Ing. 3 y 4

Como cada línea de comunicación requiere tiempo que de otra manera se ocuparía trabajando, la productividad se verá reducida en 250 líneas de código fuente al año por cada comunicación. La productividad del equipo se ve reducida en :

$$20,000 - (250 \times 6) = 18,500 \text{ LCF/año}$$

Si tenemos un proyecto de un año, éste va atrasado y solo quedan dos meses para terminarlo. Se añaden dos personas más, lo cual aumenta a 14 los canales de comunicación. La productividad del nuevo personal es para los próximos dos meses de :

$$(5,000 / 12) * 2 * 2 = 1,667 \text{ LCF/año}$$

El total para el equipo para el resto del año es de :

$$20,000 + 1,667 - (250 \times 14) = 18,167 \text{ LCF/año}$$

Esto se debe a que el nuevo personal le quitará tiempo extra por aprendizaje al personal ya dentro del proyecto.

2.3.3 Análisis comparativo de tecnicas de estimación de costos

Dentro de las técnicas de estimación de costo encontramos:

- Estimación de costos de lo particular a lo general o viceversa. La estimación de costos puede efectuarse hacia arriba o hacia abajo.

Hacia abajo se enfoca a los costos del nivel del sistema, así como a: los costos de manejo de la configuración, del control de calidad, de la integridad del sistema, del entrenamiento y de las publicaciones de documentación.

Hacia arriba, se estima el costo del desarrollo de cada módulo o subsistema y se integran para obtener el costo total.

Es recomendable utilizar ambas para que se comparen y se eliminen las diferencias obtenidas.

- Juicio de costos basado en la experiencia

En general, las personas que desarrollan cualquier actividad profesional, utilizan metodologías muy sencillas abstraídas de técnicas sofisticadas para cuantificar o controlar actividades.

Son técnicas que se basan en la intuición y/o la experiencia de otros proyectos de igual magnitud, aprovechando los errores de proyectos pasados en cálculo de tiempo, de recursos tanto humanos como de herramientas, si es que los hubo, para efectuar ajustes y permitir un proceso más eficiente.

La desventaja de este sistema de evaluación es que si la persona que lo realiza no ajusta correctamente, se confía a experiencias similares o bien carece del conocimiento intuitivo para realizarlo, se obtendrán datos con probabilidad de error muy alto.

Cuando se trabaja en equipo, esta tendencia se ve disminuída, si es que el equipo tiene experiencia, pero el resultado puede ser no deseado finalmente. Para evitar este tipo de errores se cuentan con técnicas específicas como son las de estimación por líneas de códigos y modelos de costo por módulos.

En el primer caso, estimación por líneas de códigos, se basa en dividir el proyecto en módulos funcionales o de programación hasta llegar a unidades funcionales que permitan determinar en base a la experiencia o la intuición cuantas líneas de programación se llevará cada unidad.

Basándose en información histórica, o la intuición cuando esta no exista, el planeador determina:

- Valor de líneas por módulo calculadas en forma optimista, que denominaremos (a).

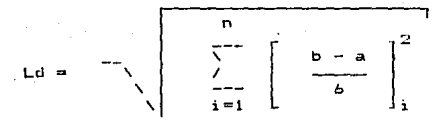
- Idem. pero de modo pesimista, señalada como (b).

- Un valor intermedio considerado como óptimo, identificado como (m).

De ésta manera el total de líneas esperadas es el siguiente:

$$Le = \frac{a + 4m + b}{6}$$

Donde la variancia esperada (desviación) de líneas esperadas esta dada por la siguiente función:



n = funciones de programa

En el segundo caso, el modelo de costos por módulos, los costos se estiman mediante la adición de los costos de cada uno de los módulos o subsistemas. También se le conoce como COCOMO, Constructive cost model, siglas que en español significan modelo de costos constructivo.

El modelo COCOMO utiliza una tabla de factores para determinar el tiempo requerido por un programador, en función de la cantidad de líneas de código fuente por módulo; Los factores se utilizan para determinar la complejidad del desarrollo de acuerdo a las características impuestas al producto, del computador, del personal y del proyecto.

** Factor multiplicador	Intervalo de valores
Atributos del producto	
Confiabilidad requerida	0.75 a 1.40
Tamaño de la base de datos	0.94 a 1.16
Complejidad del producto	0.70 a 1.65
Características de la máquina	
Limitantes en el tiempo de ejecución	1.00 a 1.66
Limitaciones en memoria principal	1.00 a 1.56
Volatilidad de la virtualidad de la máquina	0.87 a 1.30
Tiempo de entrega de programas	0.87 a 1.15
Características del personal	
Capacidad de los analistas	1.46 a 0.71
Capacidad de los programadores	1.42 a 0.70
Experiencia en programas de aplicacion	1.29 a 0.82
Experiencia en máquinas virtuales	1.21 a 0.90
Experiencia/lenguajes de programacion	1.14 a 0.95

Características del proyecto	
Uso de técnicas modernas de programación	1.24 a 0.82
Uso de herramientas de programación	1.24 a 0.83
Tiempo requerido para el desarrollo	1.23 a 1.10

Con el fin de que esta técnica pueda desarrollarse correctamente las siguientes consideraciones deben ser previstas:

Se utiliza hasta programas de 32 k, en un área de aplicación conocida, con una máquina virtual estable y conocida y es para desarrollos internos.

Abarca pruebas de aceptación, documentación y revisiones e incluye costos del personal administrativo de sistemas que tengan que ver con el proyecto.

Un número pequeño de personal conocedor es el que efectúa la definición.

Los requisitos del proyecto no cambian durante el desarrollo.

Se programa con grupos paralelos de programadores.

Las pruebas de integración se elaboran mediante un plan, previamente habiendo probado cada módulo.

La documentación se genera en forma creciente.

Por lo tanto el procedimiento para el cálculo de costos utilizando esta técnica es el siguiente:

- Identifica los subsistemas de cada módulo.
- Estima el tamaño de cada módulo.
- Especifica los factores multiplicadores por módulo.
- Calcula el esfuerzo y tiempo de desarrollo por cada módulo.
- Calcula el esfuerzo y tiempo total del sistema.
- Suma otros costos inherentes no tomados en cuenta.
- Compara la estimación contra la técnica anterior.

2.4 Ciclo de vida de un proyecto

Todo proyecto de programación mantiene un ciclo de vida, como puede ser: la definición, el desarrollo y el mantenimiento.

Si se especifica el ciclo de vida de un proyecto, entonces es fácil ubicar las diferentes actividades del mismo y controlarlas.

El ciclo de vida que se conoce dentro de ingeniería de software es definido por las fases generales de: planeación, desarrollo y mantenimiento.

2.4.1 Planeación

El primer paso para la elaboración de un proyecto es la planeación. Un primer acercamiento a la cantidad del esfuerzo por ser realizado es tomado en cuenta, considerando el alcance, los recursos por disponer, la factibilidad y un calendario.

La segunda parte es considerar las funciones, los requerimientos, las limitantes y criterios de validación por ser utilizados, desarrollando un manual de usuario o cuaderno de cargas que deberá de ser revisado entre el personal de desarrollo y el usuario. Es aquí donde se checan las condiciones que se tomaron en cuenta para la evaluación del proyecto y finalmente es aceptado para comenzar con la siguiente fase:

2.4.2 Desarrollo

La etapa de desarrollo, no es como muchas personas lo piensan la etapa más importante del proyecto. La más importante es la anterior, puesto que implica un órden y conocimiento total del problema a ser atacado. Esta etapa es solo una consecuencia, de ahí el nombre que se le da a este ciclo de "ciclo en cascada" puesto que una actividad depende de otra.

La primera parte de la etapa de desarrollo es la definición de la programación, estructuras de datos, interfases y limitantes del proyecto.

Una vez que ésto se realizó, se especifican aspectos estructurales de cada módulo, se establece la importancia de cada uno de ellos y finalmente comienza la programación

Una vez que ésta es terminada hay una etapa de chequeo donde se verifica la unidad de los módulos, la integridad del sistema y la validación de los datos.

El usuario debe de recibir el sistema y efectuar pruebas extensivas de él, previamente a la solicitud de nuevos requerimientos, comenzando la etapa de:

2.4.3 Mantenimiento

Definitivamente este rubro ha sido menospreciado más de una vez, por lo que es tan alto el tiempo de mantenimiento en una mayoría de los centros de cómputo actuales.

Una vez que se le entrega al usuario el sistema, es importante asignar recursos al mantenimiento del mismo; Hay tres tipos básicos de mantenimiento: Correctivo, de adaptación y de perfección. En cualquier caso, de nuevo se consumen recursos.

No necesariamente todas las modificaciones que se pidan por las áreas usuarias se deben de realizar , pero es muy recomendable tomarlas en cuenta puesto que finalmente, un sistema que no se utiliza, es un sistema que no sirve.

2.5 Constitución de un equipo de análisis.

No se puede iniciar un proyecto de análisis sin un equipo que permita lograrlo. Este equipo no es solamente personal de sistemas y dado que es un proyecto donde, generalmente, el departamento de sistemas no es el usuario final, es muy conveniente que participen las personas que si lo van a ocupar como herramienta para resolver su trabajo.

2.5.1 Definición de objetivos.

Dado que para todo desarrollo de un sistema de información, se requiere de recursos humanos, y dado que en la mayoría de los casos, estos recursos son limitados es muy conveniente estructurar el proyecto para obtener el mejor resultado posible.

Hay diferentes estructuras que permiten lograr lo anterior, tal es el caso de los formatos de proyecto funcional y matricial.

En el caso del formato de proyecto se utiliza el mismo grupo de programadores que llevan al cabo el proyecto desde su principio hasta su fin, incluyendo la parte de mantenimiento.

Una vez que el proyecto termina, los elementos humanos participantes son asignado a otro nuevo.

En el formato funcional, Se definen equipos de desarrollo que reciben la carga de trabajo de otro. Esto es, un equipo de planeación desarrolla la fase de planeación del sistema, otro se

encarga de la producción de programas, otro se encarga de probarlo, etc.

La ventaja mayor de la utilización de este tipo de formato, es que los diferentes equipos, adquieren una especialización fuerte en su área respectiva (lo cual puede representar un problema eventual) y atacar una serie de proyectos rapidamente.

Dado que se requiere de una fuerte comunicación entre los grupos, la documentación tiende a ser clara.

El formato matricial especifica que cada función tiene su propia administración y un equipo dedicado especificamente a dicha función. Cada proyecto es manejado por un líder de proyecto quién revisa el avance y puede participar en cualquier fase del proyecto.

Es adecuado, pues permite el avance rápido y organizado de los proyectos.

2.5.2 Estructura del grupo de programación

Dado que todo proyecto de programación debe de tener una estructrura interna, la mejor deberá de ser aquella que se adecúe mejor a la naturaleza del proyecto y la del producto:

Un grupo democrático es aquel en cual todos los integrantes tienen relación entre sí abriendo canales de comunicación importantes entre los miembros del grupo, dado que todos los elementos pueden opinar en cualquier evento del proyecto, pero uno de los elementos se nombra líder y es el que toma las decisiones finales en situaciones donde no se logra un concenso.

Un grupo sin egoismo, difiere en que no hay un líder, haciendo que todos los elementos contribuyan a las decisiones, y aprendan unos de otros; Es recomendable para proyectos largos pero tiene el inconveniente de no tener una autoridad.

Los grupos con jefes de programación son grupos estructurados en los cuales las decisiones importantes dependen de aquel con mayor jerarquía. El jefe es el responsable del desarrollo, de decisiones técnicas, es el que asigna el trabajo, y normalmente se ve apoyado por uno o varios programadores, y un coordinador que es el que administra los listados, diseños, pruebas, etc.

El grupo bajo jerarquía administrativa se basa en la anterior, pero se magnifica, pues aparecen nuevas posiciones jeráquicas, ésto es, quizá un jefe de desarrollo a cargo de varios líderes de proyectos; por lo que bien se puede establecer como un punto intermedio entre las dos estructuras anteriores. Esta

estructura permite la comunicación real cuando se requiere y es útil cuando lo que se pretende es realizar sistemas jerárquicos, pues cada subsistema se puede ubicar en un grupo de programación diferente.

Es importante resaltar que en toda estructura jeráquica, se debe de tomar en cuenta las posibilidades de ascenso de los programadores, pues es muy continuo el error de promover al mejor programador y obligándolo a efectuar actividades administrativas, de las cuales, en primera instancia no conoce, provocando la posible pèrdida de un buen programador y ganando un mal administrador, echando a porder de esta manera un buen equipo de trabajo.

2.5.3 Administración del proyecto

De nuevo, dado que los recursos son limitados, entonces el control de un proyecto debe de existir siempre.

La tècnica de control que más beneficia la integridad, capacidad creativa y esfuerzo personal de cada elemento participante en un desarrollo de programación es la tècnica de administración por objetivos.

En ella, los participantes definen quienes van a efectuar que actividades y se establece el tiempo para cada una de ellas.

Es importante mencionar que esta es una tècnica adecuada para quièn conoce el área y tiene una idea precisa, otorgada por la experiencia principalmente, de cuanto se tarda una actividad especifica en llevarse a cabo, pues de no ser asi, un programador podría tomar ventaja de ello, y una actividad de horas convertirla a días.

Es importante no permitir la existencia de muchos objetivos en un proyecto y el periodo de revisión debe ser breve. (No más de 20 objetivos por proyecto y no más de tres meses para su revisión).

2.6 Otras actividades en la planeación.

No es posible administrar un proyecto si no se cuenta con las mínimas herramientas de control o desarrollo del mismo. Estas actividades son inherentes a la persona que se ocupa de controlar el proyecto.

2.6.1 Planeación de puntos de control.

Durante todas la etapas de un desarrollo se deben de especificar cuales son los puntos de control del mismo; de no hacerse, se corre el riesgo de avanzar a fases que no están debidamente fundamentadas.

Durante la planeación se especifican los procedimientos de administración y control de calidad que se emplearán, y se establecerá que herramientas se utilizarán.

Durante la etapa de diseño, los procedimientos anteriores se emplean para verificar los estándares, así como controlar requisitos y especificaciones de diseño.

En la etapa de implantación se supervisan los requisitos, las especificaciones de diseño, y el código fuente, además se llevan a cabo pruebas de aceptación.

2.6.2 Planeación de herramientas por utilizar

Durante la etapa de planeación, se establece la posibilidad de contemplar las diversas alternativas que un lider de proyecto tiene como herramienta.

Para el desarrollo de lo requisitos, el diseño estructural y el código fuente, se pueden emplear herramientas automatizadas y técnicas modernas de programación.

Para controlar el desarrollo se pueden utilizar método de ruta crítica, gráficas de Gantt, o cualquier metodología de control.

No es objetivo de esta tesis el detallar cada una de las técnicas anteriores, pero es importante que se consideren dentro de la etapa de planeación.

2.6.3 Planeación de la operación y mantenimiento del sistema

Es importante recalcar el hecho de que todo sistema requiere de mantenimiento. Si no se acepta el hecho no podremos contemplar un desarrollo adecuado en el tiempo.

Algunos autores establecen que el tiempo de desarrollo de un sistema es de 40% para el diseño, 20% para la programación y 40% para el mantenimiento; pero estos dátos son atípicos en muchas instalaciones, puesto que hay sistemas que requieren mantenimiento constante.

Esto se debe o bien a una total falta de conocimiento al

efectuar el análisis o bien se debe a que el sistema trabaja en condiciones alarmantemente cambiantes para las posibilidades del mismo; en cualquiera de los dos casos, es muy posible que se deseche la programación actual y se desarrolle otro sistema bien diseñado o al menos lo suficientemente parametrizado como para no propiciar un disparo en la etapa de mantenimiento.

2.7 Aplicacion al caso práctico

Es importante recalcar el hecho de que para cada desarrollo en específico se tendrán que evaluar cada uno de los aspectos más importantes que para el administrador del proyecto juzgue pertinente. Es así que para ejemplificar los costos de un proyecto, se tomará como ejemplo el control y desarrollo de la programación de un sistema de cuentas por pagar para una compañía del ramo de lubricantes.

El grupo de programación que se escogió es del tipo de jefe de programación con tres analistas a su cargo. Dado que se administrará el proyecto pro medio de objetivos se utilizó un paquete de control de proyectos denominado "Timeline 3", que permite efectuar diagramas de barras.

El primer acercamiento se hizo en base a la experiencia, y después se comparó con las fórmulas de Boehm. Para ello, se especificó cada una de las actividades mayores, y se fue realizando, mediante la técnica Top down el crecimiento de proyecto, como lo muestran las figuras 2.7.1(p 45), 2.7.2(p 46), 2.7.3(p 47).

Una vez que se desarrolló completamente el diagrama de barras, incluyendo todas las actividades se estimaron los días totales de programación, y se compararon con las fórmulas de Boehm, asignando un ajuste de 1.5 por pérdida financiera y un ajuste de 0.83 por utilizar herramientas de programación. figura 2.7.4(p 48).

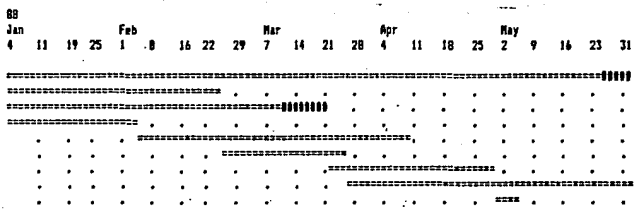
Los resultados son contundentes al formalizar el proyecto, puesto que se admite un mejoramiento en cuanto a productividad de hasta un 40% con respecto al estimado inicial, con una viabilidad humana del 100%. Esto quiere decir que al formalizar el proyecto con una técnica específica, nos permite tener un ahorro en días calendario de 38 días, utilizando el mismo equipo de trabajo.

Si se considera que un mes laborable tiene 20 días hábiles, equivale a un ahorro de dos meses de sueldo, de todo el equipo.

La nueva gráfica del proyecto, se encuentra formalizada en las figuras 2.7.5(p 49), 2.7.6(p 50), 2.7.7(p 51).

Schedule Name : Sistema de cuentas por pagar
Responsible :
As-of Date : 30-Nov-88 Schedule File : C:\TL3FILES\CXPSYS

Fecha Inicio	Fecha Fin	Duración	Nombre de la Tarea
4-Jan-88	1-Jun-88	103	SISTEMA DE CUENTAS POR PAGAR
4-Jan-88	24-Feb-88	36	+ Maestro de Proveedores
4-Jan-88	21-Mar-88	54	+ Control de Pedidos
4-Jan-88	3-Feb-88	22	+ Recepcion otros pedidos prov
4-Feb-88	5-Apr-88	46	+ Material/Ordn.Trabajo/Contrato
25-Feb-88	24-Mar-88	21	+ Afectacion de movimientos
22-Mar-88	28-Apr-88	28	+ Validacion facturas para pago
25-Mar-88	1-Jun-88	48	+ Informe saldos de proveedores
29-Apr-88	4-May-88	4	+ Cierre anual de proveedores



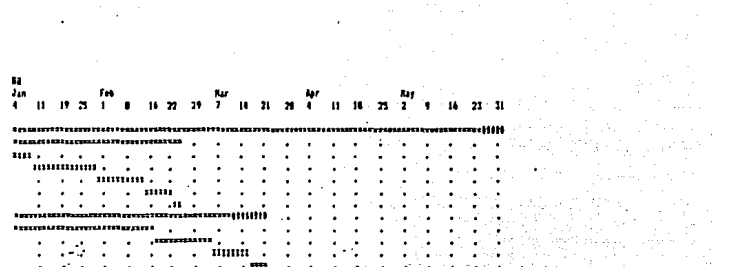
XXXXX Detail Task ##### Summary Task M Milestone
xxXXX (Started) ==### (Started)))) Conflict
XXX-- (Slack) ###-- (Slack) ..XXX Resource delay
Scale: 1 day per character

TIME LINE Gantt Chart Report, Strip 1

Fig. 2.7.1

Schedule NAME : Sistema de Cuentas por Pagar
Responsible :
As-of Date : 30-Nov-88 Schedule File : C:\VILMFILES\CXPSYS

Fecha Inicio	Fecha Fin	Duración	Nombre de la Tarea
4-Jan-88	1-Jun-88	105	SISTEMA DE CUENTAS POR PAGAR
4-Jan-88	24-Feb-88	36	Maestro de Proveedores
4-Jan-88	7-Jan-88	4	Catalogo de datos generales
8-Jan-88	28-Jan-88	14	Actualizacion y consultas
29-Jan-88	11-Feb-88	10	Formas de actualizacion
12-Feb-88	22-Feb-88	6	Catalogo de cuentas contables
23-Feb-88	24-Feb-88	2	Actual. campos estadisticos
4-Jan-88	21-Mar-88	54	Control de Pedidos
4-Jan-88	15-Feb-88	30	+ Solicitudes
17-Feb-88	3-Mar-88	12	+ Autorizaciones
4-Mar-88	15-Mar-88	8	Actualizacion y consultas
16-Mar-88	21-Mar-88	4	+ Listados
4-Jan-88	3-Feb-88	22	Recepcion otros pedidos prov
4-Jan-88	2-Feb-88	21	+ Captura y actualizacion
3-Feb-88	3-Feb-88	1	+ Listado
4-Feb-88	8-Mar-88	45	Material/Ords.Trabajo/Contrato
4-Feb-88	12-Feb-88	7	Actualizacion y Consultas
16-Feb-88	17-Feb-88	2	+ Listado
15-Feb-88	26-Feb-88	7	Actualizacion de observaciones
29-Feb-88	18-Mar-88	15	+ Solicitudes
21-Mar-88	8-Abr-88	15	+ Autorizaciones
25-Feb-88	24-Mar-88	21	Afectacion de movimientos
25-Feb-88	4-Mar-88	7	Cargos y abonos de proveedores
7-Mar-88	15-Mar-88	7	de abonos con cheque
16-Mar-88	24-Mar-88	7	de cargos y devoluciones
22-Mar-88	28-Abr-88	28	Validacion facturas para pago
22-Mar-88	30-Mar-88	7	proveedores de materiales
31-Mar-88	8-Abr-88	7	proveedores de materias primas
11-Abr-88	19-Abr-88	7	proveedores ordenes de trabajo
20-Abr-88	28-Abr-88	7	proveedores de contratos
25-Mar-88	1-Jun-88	48	Informe saldos de proveedores
25-Mar-88	4-Abr-88	7	Actualizacion diaria
5-Abr-88	13-Abr-88	7	Estado de cuenta de proveedore
14-Abr-88	17-May-88	24	+ Analisis
18-May-88	1-Jun-88	10	Consulta a proveedores
28-Abr-88	4-May-88	4	Cierre anual de proveedores
2-May-88	2-May-88	2	Estado de proveedores anual
3-May-88	4-May-88	2	. Traspaso de saldos

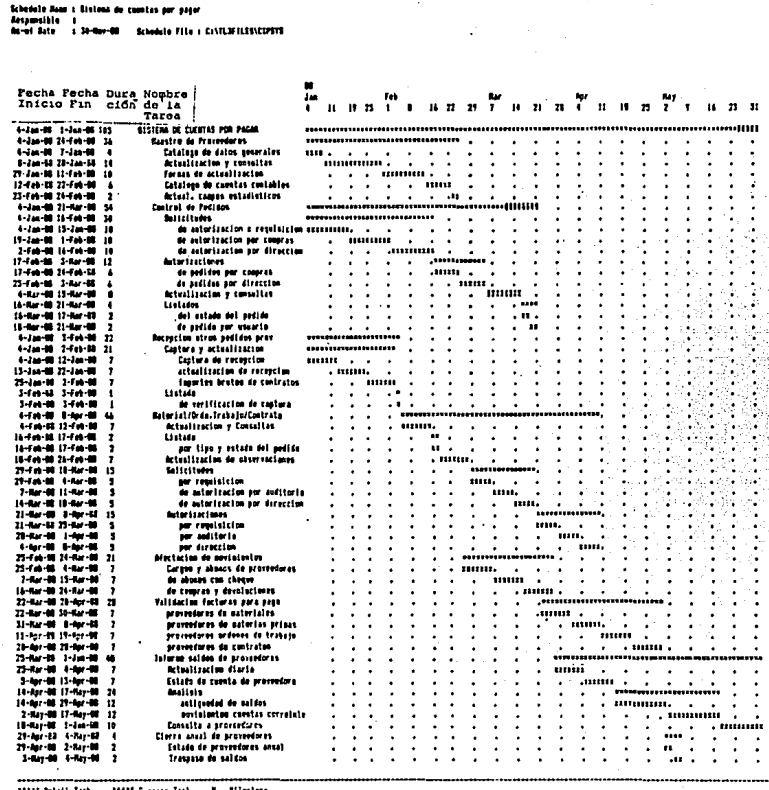


Detail Task @@@@ Summary Task M Milestone
=#### (Started) =+@@ (Started)))) Conflict
###— (Slack) @@@— (Slack) ..### Resource delay
Scale: 1 day per character

TIME LINE Gantt Chart Report, Strip 1

Fig. 2.7.2

Schedule Name : Sistema de cuentas por pagar
Responsible :
As-of Date : 30-Nov-88 Schedule File : C:\TLJF\FILES\CXPSYS



Scales: 1 day per character

TIME LINE Gantt Chart Report, Strip 1

Fig. 2.7.3

ANALISIS DE COSTOS

	DIAS DE PROGRAMACION ESTIMADOS Y EXPERIENCIA	CANTIDAD DE INSTRUCCIONES ESTIMADAS (MI)	DIAS POR PROGRAMA (PD)	TIEMPO DE DESARROLLO (TDEV)	AJUSTES DE ESFUERZO (PERDIDA FINANCIERA) 1.15	AJUSTES DE PRODUCTIVIDAD (MEDIA DE PROG.) 0.83	PORCENTAJE DE PRODUCTIVIDAD SEGUN BOEHM
SISTEMA DE CUENTAS POR PAGAR	259	33800	717.2457	169.1072	180.7223	156.6483	39.52109
MODULO MAESTRO DE PROVEEDORES	36	6600	196.2837	31.03868	24.59486	20.40125	49.21846
CATALOGO DE DATOS GENERALES	4	600	10.76941	3.492256	4.016439	3.333645	
ACTUALIZACION Y CONSULTAS	10	1600	40.09695	8.757402	6.626012	5.495440	
FORMAS DE ACTUALIZACION	10	1200	28.10237	5.834095	5.789210	4.803044	
CATALOGO DE CUENTAS CONTABLES	6	900	16.45424	4.105257	4.721735	3.919010	
ACTUALIZACION DE CAMPOS ESTADISTICOS	2	600	5.191624	3.648458	3.646062	3.026187	
MODULO CONTROL DE PEDIDOS	54	7000	149.9490	33.21666	38.19914	31.70530	41.28647
SOLICITUD DE AUT. PCR REQUISICION	10	1200	28.10237	5.834095	5.789210	4.803044	
SOLICITUD DE AUT. POR COMPRAS	10	1200	28.10237	5.834095	5.789210	4.803044	
SOLICITUD DE AUTORIZACION POR DIRECC	10	1200	28.10237	5.834095	5.789210	4.803044	
AUTORIZACIONES DE PEDIDOS POR COMPRAS	6	800	16.45424	6.195257	4.721735	3.919010	
ACTUALIZACION DE PEDIDOS POR DIRECCI	6	900	16.45424	4.105257	4.721735	3.919010	
ACTUALIZACION Y CONSULTAS	8	1000	22.75878	6.455762	5.216053	4.329723	
LISTADO DE ESTADO DEL PEDIDO	2	600	5.191624	3.648458	3.646062	3.026187	
LISTADO DE PEDIDO POR USUARIO	2	600	5.191624	3.648458	3.646062	3.026187	
MODULO RECEPCION DE OTROS PEDIDOS	22	2900	60.54319	15.10769	17.37384	14.41029	34.45322
CAPTURA DE RECEPCION DE OTROS PEDIDO	7	900	19.34526	4.346320	5.021268	4.167652	
ACTUALIZACION DE RECEPCION DE OTROS	7	900	19.34526	4.346320	5.021268	4.167652	
ACTUALIZACION DE IMP. BRUTOS DE CONT	7	900	19.34526	4.346320	5.021268	4.167652	
LISTADO DE VERIFICACION DE CAPTURA	1	200	2.587349	2.063729	2.310059	1.917332	
MOB/BATERIMES/ORDEN TRABAJO/CONTRAT	46	6400	129.4673	34.28759	39.43119	32.72789	28.85349
ACTUALIZACION Y CONSULTAS	7	900	19.34526	4.346320	5.021268	4.167652	
LISTADO POR TIPO Y ESTADO DEL PEDIDO	2	600	5.191624	3.648458	3.646062	3.026187	
ACTUALIZACION DE OBSERVACIONES	7	900	19.34526	4.346320	5.021268	4.167652	
SOLICITUDES POR REQUISICION	5	700	13.58752	3.017776	4.190443	3.644067	
SOLICITUDES DE AUT. PCR AUDITORIA	5	700	13.58752	3.017776	4.190443	3.644067	
SOLICITUDES DE AUT. POR DIRECCION	5	700	13.58752	3.017776	4.190443	3.644067	
AUTORIZACIONES POR REQUISICION	5	700	13.58752	3.017776	4.190443	3.644067	
AUTORIZACIONES POR AUDITORIA	5	700	13.58752	3.017776	4.190443	3.644067	
AUTORIZACIONES POR DIRECCION	5	700	13.58752	3.017776	4.190443	3.644067	
MODULO DE AFECTACION DE MOVIMIENTOS	21	2700	58.03590	13.69816	15.06589	12.50795	40.46210
CARGOS Y ABONOS DE PROVEEDORES	7	900	19.34526	4.346320	5.021268	4.167652	
CARGOS DE ABONOS CON CHEQUES	7	900	19.34526	4.346320	5.021268	4.167652	
CARGOS DE COMPRAS Y DEVOLUCIONES	7	900	19.34526	4.346320	5.021268	4.167652	
MOB VALIDACION DE FACTURAS PARA PAGO	28	3600	77.38197	17.45528	20.09507	16.67061	40.46210
PROVEEDORES DE MATERIALES	7	900	19.34526	4.346320	5.021268	4.167652	
PROVEEDORES DE MATERIAS PRIMAS	7	900	19.34526	4.346320	5.021268	4.167652	
PROVEEDORES DE ORDENES DE TRABAJO	7	900	19.34526	4.346320	5.021268	4.167652	
PROVEEDORES DE CONTRATOS	7	900	19.34526	4.346320	5.021268	4.167652	
MODULO INFORME SALDOS DE PROVEEDORES	46	5800	124.9673	24.59465	28.28384	23.47559	51.09253
ACTUALIZACION DIARIA	7	900	19.34526	4.346320	5.021268	4.167652	
ESTADO DE CUENTA DE PROVEEDORES	7	900	19.34526	4.346320	5.021268	4.167652	
ANALISIS DE ANTIGUEDAD DE SALDOS	12	1600	34.46921	5.413957	6.226050	5.167622	
ANALISIS DE MOV. CUENTAS CORRIENTES	12	1600	34.46921	5.413957	6.226050	5.167622	
CONSULTA A PROVEEDORES	10	1200	28.10237	5.834095	5.789210	4.803044	
CIERRE ANUAL DE PROVEEDORES	6	800	16.38324	5.293596	6.092005	5.056364	-26.4491
TRASPASO DE SALDOS	2	400	5.191624	2.648458	3.046062	2.528187	
ACTUALIZACION DE RECEPCION DE BIENES	2	400	5.191624	2.648458	3.046062	2.528187	

Fig. 2.7.4

Schedule Name : Sistema de cuentas por pagar
Responsible :
As-of Date : 1-Dec-87 Schedule File : C:\TL3FILES\CXPSYS2V

Fecha Inicio	Fecha Fin	Duración	Nombre de la Tarea
4-Jan-88	7-Apr-88	66.7	SISTEMA DE CUENTAS POR PAGAR
4-Jan-88	3-Feb-88	21.5	+ Maestro de Proveedores
4-Jan-88	22-Feb-88	33.5	+ Control de Pedidos
4-Jan-88	27-Jan-88	16.7	+ Recepcion otros pedidos prov
28-Jan-88	21-Mar-88	36.6	+ Material/Orde.Trabajo/Contrato
5-Feb-88	25-Feb-88	12.6	+ Afectacion de movimientos
23-Feb-88	21-Mar-88	19.1	+ Validacion facturas para pago
2-Mar-88	7-Apr-88	26.8	+ Informe saldos de proveedores
22-Mar-88	29-Mar-88	5.5	+ Cierre anual de proveedores

88
Jan 4 11 19 25 | Feb 1 8 16 22 29 | Mar 7 14 21 28 | Apr 4 11 18 25 | May 2

XXXX Detail Task ##### Summary Task M Milestone
xxXXX (Started) ==### (Started) >>> Conflict
XXX-- (Slack) ###-- (Slack) ..XXX Resource delay

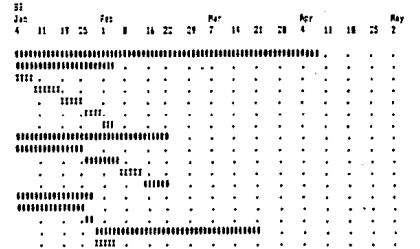
Scale: 1 day per character

TIME LINE Gantt Chart Report, Strip 1

FIG. 2.7.5

Schedule Name : Sistema de cuentas por pagar
Responsible :
As-of Date : 1-Dec-87 Schedule File : C:\PLANFILES\SISTEMA.SCH

Fecha Inicio	Fecha Fin	Duración	Nombre de la Tarea
4-Jan-88	7-Apr-88	65.7	SISTEMA DE CUENTAS POR PAGAR
4-Jan-88	3-Feb-88	21.5	Maestro de Proveedores
4-Jan-88	7-Jan-88	3.2	Catalogo de datos generales
8-Jan-88	15-Jan-88	5.5	Actualizacion y consultas
19-Jan-88	22-Jan-88	4.7	Formas de actualizacion
21-Jan-88	25-Jan-88	3.9	Catalogo de cuentas contables
1-Feb-88	3-Feb-88	2.5	Actual. cuentas estadisticas
4-Jan-88	22-Feb-88	37.5	Control de Pedidos
4-Jan-88	25-Jan-88	14.8	+ Solicitudes
26-Jan-88	4-Feb-88	7.9	+ Autorizaciones
5-Feb-88	11-Feb-88	6.4	Actualizacion y consultas
12-Feb-88	22-Feb-88	8.5	+ Listados
4-Jan-88	27-Jan-88	18.9	Recepcion otros pedidos proc.
4-Jan-88	25-Jan-88	14.1	+ Captura y actualizacion
26-Jan-88	27-Jan-88	1.9	+ Listado
28-Jan-88	21-Mar-88	35.6	Material/Orden.Trabajo/Contrato
28-Jan-88	3-Feb-88	6.1	Actualizacion y Consultas
4-Feb-88	8-Feb-88	2.5	+ Listado
9-Feb-88	16-Feb-88	6.1	Actualizacion de observaciones
17-Feb-88	3-Mar-88	11.6	+ Solicitudes
4-Mar-88	21-Mar-88	11.6	+ Autorizaciones
8-Feb-88	24-Feb-88	12.6	Recepcion de movimientos
8-Feb-88	12-Feb-88	4.1	Cargos y abonos de proveedores
12-Feb-88	19-Feb-88	4.1	de abonos con cheque
22-Feb-88	24-Feb-88	4.1	de compras y devoluciones
23-Feb-88	21-Mar-88	19.1	Validacion facturas para pago
22-Feb-88	29-Feb-88	4.1	proveedores de materiales
1-Mar-88	7-Mar-88	4.1	proveedores de materias primas
8-Mar-88	14-Mar-88	4.1	proveedores ordenes de trabajo
15-Mar-88	21-Mar-88	4.1	proveedores de contratos
2-Mar-88	7-Apr-88	26.8	Informe saldos de proveedores
2-Mar-88	8-Mar-88	4.1	Actualizacion diaria
9-Mar-88	15-Mar-88	4.1	Estado de cuenta de proveedors
16-Mar-88	31-Mar-88	11.1	+ Analisis
1-Mar-88	7-Mar-88	4.7	Consulta a proveedores
22-Mar-88	29-Mar-88	5.5	Cierre anual de proveedores
22-Mar-88	24-Mar-88	2.5	Estado de proveedores anual
25-Mar-88	29-Mar-88	2.5	Traspaso de saldos



Detail Task @@@@@ Summary Task M Milestone
xxIII (Started) ++@@@ (Started)))) Conflict
III-- (Slack) @@@-- (Slack) ..III Resource delay

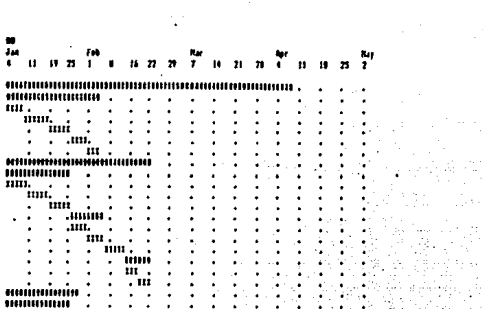
Scale: 1 day per character

TIP LINE Gantt Chart Report, Strip 1

Fig. 2.7.6

Schedule Name : Sistema de cuentas por pagar
Responsible :
As-of Date : 1-Dec-87 Schedule File : C:\ILS\FILES\CUPS1E2V

Fecha Inicio	Fecha Fin	Dura ción	Nombre de la Tarea
4-Jan-88	7-Apr-88	64.7	SISTEMA DE CUENTAS POR PAGAR
4-Jan-88	3-Feb-88	21.5	Registro de Proveedores
4-Jan-88	7-Jan-88	3.2	Catalogo de datos generales
8-Jan-88	15-Jan-88	5.5	Actualizacion y consultas
19-Jan-88	25-Jan-88	4.7	Formas de actualizacion
26-Jan-88	29-Jan-88	3.0	Catalogo de cuentas contables
1-Feb-88	3-Feb-88	2.5	Actual. campos estadisticos
4-Jan-88	22-Feb-88	33.5	Control de Pedidos
4-Jan-88	25-Jan-88	14.8	Solicitudes
4-Jan-88	8-Jan-88	4.7	de autorizacion o requisicion
11-Jan-88	15-Jan-88	4.7	de autorizacion por compras
19-Jan-88	25-Jan-88	4.7	de autorizacion por direccion
21-Jan-88	4-Feb-88	7.9	Autorizaciones
26-Jan-88	29-Jan-88	2.9	de pedidos por compras
1-Feb-88	4-Feb-88	3.9	de pedidos por direccion
5-Feb-88	11-Feb-88	4.4	Actualizacion y consultas
12-Feb-88	22-Feb-88	9.5	Listados
12-Feb-88	17-Feb-88	2.5	del estado del pedido
18-Feb-88	22-Feb-88	2.5	de pedido por usuario
4-Jan-88	27-Jan-88	16.9	Recepcion otros pedidos prov
4-Jan-88	25-Jan-88	14.1	Captura y actualizacion
4-Jan-88	8-Jan-88	4.1	Captura de recepcion
11-Jan-88	15-Jan-88	4.1	actualizacion de recepcion
19-Jan-88	25-Jan-88	4.1	Imprimir Ordenes de contratos
26-Jan-88	27-Jan-88	1.9	Listado
26-Jan-88	27-Jan-88	1.9	de verificacion de captura
28-Jan-88	31-Mar-88	34.6	Material/Orde.Trabajo/Contrato
28-Jan-88	3-Feb-88	4.1	Actualizacion y Consultas
0-Feb-88	8-Feb-88	2.5	Listado
4-Feb-88	8-Feb-88	2.5	por tipo y estado del pedido
9-Feb-88	16-Feb-88	4.1	Actualizacion de observaciones
17-Feb-88	1-Mar-88	11.6	Solicitudes
17-Feb-88	22-Feb-88	3.6	por requisicion
23-Feb-88	26-Feb-88	3.6	de autorizacion por auditoria
29-Feb-88	3-Mar-88	3.6	de autorizacion por direccion
4-Mar-88	21-Mar-88	11.6	Autorizaciones
4-Mar-88	9-Mar-88	3.6	por requisicion
10-Mar-88	15-Mar-88	3.6	por auditoria
16-Mar-88	21-Mar-88	3.6	por direccion
0-Feb-88	26-Feb-88	17.6	Afectacion de movimientos
0-Feb-88	12-Feb-88	4.1	Cargos y abonos de proveedores
12-Feb-88	19-Feb-88	4.1	de abonos con cheque
22-Feb-88	26-Feb-88	4.1	de compras y devoluciones
23-Feb-88	21-Mar-88	19.1	Validacion facturas para pago
23-Feb-88	29-Feb-88	4.1	proveedores de materiales
1-Mar-88	7-Mar-88	4.1	proveedores de materias primas
8-Mar-88	14-Mar-88	4.1	proveedores ordenes de trabajo
15-Mar-88	21-Mar-88	4.1	proveedores de contratos
2-Mar-88	7-Apr-88	24.8	Informe saldos de proveedores
2-Mar-88	6-Mar-88	4.1	Actualizacion diaria
9-Mar-88	15-Mar-88	4.1	Estado de cuenta de proveedor
16-Mar-88	31-Mar-88	11.1	Analisis
16-Mar-88	23-Mar-88	5.1	antiguedad de saldos
24-Mar-88	31-Mar-88	5.1	movimientos cuentas corrientes
1-Apr-88	7-Apr-88	4.7	Consulta a proveedores
22-Mar-88	29-Mar-88	5.5	Cierre anual de proveedores
22-Mar-88	24-Mar-88	2.5	Estado de proveedores anual
25-Mar-88	29-Mar-88	2.5	Traspaso de saldos



Detail Task @@@@@ Summary Task ◆ Milestone
xxIII (Started) ◆◆III (Started)))) Conflict
IIII— (Slack) @@@— (Slack) ...III Resource delay

Scale: 1 day per character

TIME LINE Gantt Chart Report, Strip 1

Fig. 2.7.7

CAPITULO 3 Estimacion de costos de un proyectos

3.1 Importancia de la estimacion de costos.

En los inicios de la computaciòn el costo derivado por el desarrollo de los programas comprendia un pequeño porcentaje del costo total de un sistema basado en computadora por lo que los errores en la estimaciòn del costo del desarrollo no eran muy impactantes. Hoy en dia el desarrollo de los programas es el elemento más costoso en los sistemas basados en computadora; por lo que una buena estimaciòn del costo es sumamente importante en el estudio de la factibilidad de un proyecto.

La obtenciòn de costos no es una tarea sencilla ya que intervienen una gran cantidad de variables -factor tècnico, ambiental, gubernamental- pero siendo este un punto sumamente importante en la factibilidad del desarrollo de un proyecto se debe obtener el dato lo mas exacto posible.

Existen una serie de opciones potenciales que nos pueden llevar a la obtenciòn del costo de un proyecto.

El obtener el costo al final del proyecto no es muy pràctico ya que este dato no se obtiene oportunamente, ya que cuando necesitamos la informaciòn es al inicio del proyecto.

El desarrollar un modelo paramètrico consiste en desarrollar un modelo basado en la experiencia (datos històricos) en donde se evalúan parametros independientes que afectan el costo.

La técnica de descomposición consiste en desglosar la principales funciones del proyecto y costearlas en forma independiente de tal manera que sea posible estimar un dato lo màs exacto posible.

El sistema automatizado de costeo es una opciòn muy atractiva cuando se desea la estimaciòn de costos ya que se toman en cuenta las caracteristicas de la organizaciòn que va a desarrollar (experiencia) asi como el sistema a ser desarrollado.

3.2 Factores en el costo de los programas

Como se dijo anteriormente el costo de un sistema se compone de una gran cantidad de variables que determinan el costo del desarrollo del proyecto; cada proyecto es diferente y pueden variar algunas de las variables a considerar; entre las mas importantes se tiene la capacidad del equipo de desarrollo, la complejidad del producto a desarrollar y el nivel tecnològico.

La capacidad del equipo de desarrollo es un factor sumamente importante en la obtención del costo del proyecto. La capacidad del equipo se debera de medir tanto en el conocimiento que se tenga del producto a desarrollar como el de la metodología y herramientas que se utilizarán.

Por ejemplo, si se está desarrollando un sistema de inversiones sera de gran ayuda que las personas involucradas en el proyecto tengan conocimiento en inversiones.

La complejidad del producto a desarrollar se refiere al tipo de producto de programación deseado. Estos se pueden dividir en programas de aplicación tales como nóminas, contabilidades, etc.; programas de apoyo que incluye compiladores, editores etc. y programas de sistema tales como sistemas operativos.

Brooks establece la formula 1-3-9 para indicar el grado de complejidad para cada tipo de programa; esto es, los programas de apoyo son 3 veces mas complicados que los de aplicación y los de sistema 3 veces mas complicados que los de apoyo.

Resumiendo, no es lo mismo escribir 100 lineas de un programa de nómina que 100 lineas de un sistema operativo.

El nivel tecnológico se puede dividir en equipo y programas:

El equipo se refiere al tipo de computador en donde se realizarán los programas, la velocidad de èste, y las facilidades con que cuente para realizar en forma mas eficiente el desarrollo (tiempo de respuesta, memoria, disco, impresoras, confiabilidad del equipo).

Es un hecho que durante las fase de programación y mantenimiento el numero de veces que se solicitan compilaciones al computador es muy grande y el tiempo de respuesta del equipo es determinante en la productividad de los programadores. El uso de editores flexibles pueden incrementar significativamente la productividad de los desarrolladores.

La selección del lenguaje de programación es de suma importancia ya que este es uno de los factores que determina la productividad de los programadores. El uso de lenguajes flexibles para la aplicación a desarrollar y el conocimiento que los programadores de este tengan son puntos que se deberan evaluar.

En la actualidad existen en el mercado una gran cantidad de lenjuages de tercera generación y generadores de código que permiten que el tiempo de desarrollo se reduzca significativamente (hasta 10 veces con respecto a uno de 3GL --lenguaje de tercera generación --). Este es un punto sumamente importante para abatir los costos en la fase de programación y mantenimiento.

3.3 Modelos de estimacion.

Los modelos de estimaciòn de programas usan formulas derivadas empiricamente para predecir datos que son requeridos durante la fase de planeaciòn del proyecto.

No todos los modelos de estimaciòn son apropiados para todos los tipos de aplicaciones ni todos los ambientes de desarrollo y debido a que las formulas de estos modelos son obtenidos utilizando datos empiricos; su utilizaciòn requiere cierto cuidado.

Se pueden obtener modelos sobre el desarrollo de programas que permitan predecir el nùmero de gente requerida en el proyecto en funciòn del tiempo; el costo en funciòn de las caracteristicas del sistema; la duraciòn del proyecto en funciòn de la gente y el medio ambiente de desarrollo.

Durante la fase de planeaciòn, el costo y duraciòn son las principales caracteristicas a estimar. Por tal razòn, los modelos que estimen el costo o esfuerzo (en personas-meses) resultan ser las de mayor utilidad.

3.3.1 Modelos de recursos.

Los modelos de recursos consisten en una serie de ecuaciones empiricas que predicen el esfuerzo (personas-mes), y duraciòn del proyecto (meses). Existen cuatro diferentes modelos; modelo de una variable estàtica, modelo multivariable estàtica, modelo multivariable dinàmica y modelo teòrico.

El modelo de variable sencilla estática toma la forma:

$$\text{curso} = c_1 \times (\text{caracteristicas estimadas})^{c_2}$$

En donde el recurso puede ser esfuerzo (E), duraciòn del proyecto (D), tamano del equipo (T) y un nùmero de pàginas de documentaciòn (DOC) son modeladas en funciòn del nùmero de lineas de còdigo fuente (L). Las constantes c1 y c2 son derivadas de datos de proyectos anteriores.

Considerando el estudio realizado por Walston and Felix en 60 proyectos en el rango de 4,000 a 467,000 lineas de codigo y de 12

a 11,758 personas-mes los siguientes modelos son aplicables:

$$E = 5.2 \times L$$

$$D = 4.1 \times L$$

$$D = 2.47 \times E$$

$$S = 0.54 \times E$$

$$DOC = 49 \times L$$

Estas ecuaciones son válidas pero dependientes de un medio ambiente de desarrollo y aplicaciones específicas por lo que no son generales sin embargo, algunos modelos muy sencillos pueden ser obtenidos utilizando información local siempre y cuando exista.

El modelo multivariable estática, como el anterior, hace uso de información histórica para obtener una ecuación empírica. Un modelo típico de este tipo es:

$$Recurso = c_{i1} \times e_1 + c_{21} \times e_2 + \ldots +$$

En donde ei es la i-esima característica de desarrollo y c's son las constantes de estas características.

3.3.2 Modelo de estimación Putnam.

El modelo de estimación de Putnam es un modelo basado en el análisis del ciclo de vida de un sistema en términos de una distribución específica de esfuerzo contra tiempo. Esta distribución fue derivada como resultado de la observación de la distribución de fuerzas de trabajo en proyectos muy grandes (mas de 100,000 lineas de codificacion). Sin embargo, es posible realizar extrapolaciones para proyectos pequeños. La distribución de esfuerzo para proyectos muy grandes se muestra en la figura 3.3.2.1.

Fuente: Software Engineering a Practitioner's Approach
Roger S. Pressman McGraw-Hill International Book Co.
Pag. 72

Simbología:
+ Definición del Sistema
. Instalación
▪ Pruebas y validaciones
* Diseño Funcional
° Diseño y Codificación
' Desarrollo, operación y mant.

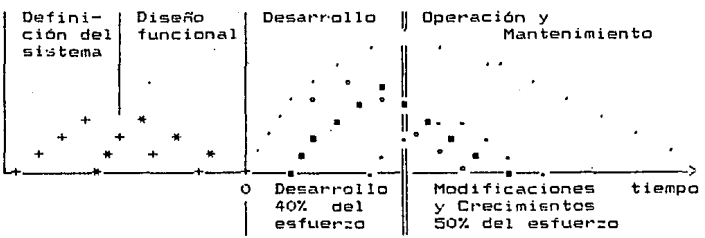


Figura 3.3.2.1 Modelo de estimación Putnam

La curva de la figura 3.3.2.1 toma la forma clásica que describio analiticamente Lord Rayleigh, además Norden utilizando datos empíricos describió la misma forma de curva por lo que generalmente es llamada la curva de Rayleigh-Norden.

La curva de Rayleigh-Norden puede ser utilizada para obtener una ecuación para los programas que relaciona el número de lineas de código fuente; el esfuerzo y el tiempo de desarrollo:

$$L = C_k K^{1/3} t_d^{4/3}$$

en donde la constante C representa la constante del nivel tecnológico y refleja los obstáculos que pueden llegar a dificultar el progreso de un programador. Algunos valores tipicos que toma esta constante son: C = 6,500 para un ambiente de desarrollo de programas "pobre", es decir no existe una metodologia; C = 10,000 para un desarrollo de programas bueno; y C = 12,500 para un ambiente excelente.

La constante C puede derivarse para condiciones locales utilizando datos históricos recolectados sobre esfuerzos desarrollados con anterioridad. Si reacomodamos la ecuacion llegamos a la siguiente expresion para el esfuerzo de desarrollo: k

$$K = \frac{L^3}{C_k^3 t_d^4}$$

donde t es el tiempo de desarrollo en años y L es el número de líneas de código.

La ecuación para el esfuerzo desarrollado puede relacionarse al costo desarrollado al incluir un factor de tarifa laboral ---- ($/AÑO-HOMBRE). Debido a los términos con potencias cúbicas y cuartas en el modelo, un pequeño cambio en las líneas de código (L) puede reducir significativamente el costo. Por ejemplo si los requerimientos de programación disminuyen de tal manera que el estimado de líneas de código se reducen un diez por ciento, el costo del desarrollo global se reducirá en un veintisiete por ciento (según el Modelo de Putnam). De manera similar se pueden analizar los cambios resultantes de alteraciones en la fecha de terminación del proyecto (es decir, t cambia) o cambios en la constante de tecnología (C).

3.3.3 Modelo de estimacion Esterling.

Esterling ha propuesto un modelo de productividad que toma en cuenta hasta las características minúsculas del ambiente de trabajo. A un nivel individual el proceso de desarrollo de programas esta afectado por un número n de individuos que interactuán en un proyecto y las características del ambiente en el cual estas personas interactuan. Esterling afirma que las juntas y otras actividades "no productivas" se llevan a cabo durante un día normal de trabajo (jornada de ocho horas) y que el período mas productivo ocurre durante el tiempo "extra".

Los parámetros asociados con el modelo Esterling contienen los siguientes elementos:

a - Fracción promedio del tiempo de trabajo invertido en trabajo administrativo.

t - Duración promedio en minutos de las interrupciones de trabajo.

r - Tiempo promedio en minutos de recuperación después de una interrupción.

k - Número de interrupciones por día de trabajo, por personas que trabajan directamente en el proyecto.

p - Número de interrupciones por día de trabajo originadas por otras causas.

i - Costo indirecto por persona expresado como una fracción del sueldo base.

d - Diferencia en pago para el tiempo extra expresado como una fracción del sueldo base.

La siguiente tabla muestra valores tipicos de estos paràmetros:

			PROGRAMADORES		
PARAMETRO	RANGO	OBREROS	OPTIMISTA	TIPICO	PESIMISTA
a	0 - 0.5	0.0	0.05	0.10	0.15
t	1 - 20	3	3	5	10
r	5 - 10	0.5	0.5	2.0	8.0
k	1 - 10	1	2	3	4
p	1 - 10	1	1	4	10
i	1 - 3	0.2	0.2	0.5	1.0
d	1 - 2	1.5	1.0	1.0	1.5

El modelo de productividad esta constituido por cinco ecuaciones. Si g es el nùmero promedio de horas extras por dia de trabajo y n es el nùmero de individuos trabajando en un proyecto Esterling desarrolla una relacion empirica para la fracción w de tiempo de trabajo ùtil (por dia de trabajo por persona):

$$w = 0.125 \times \left(8 - 8a + g - \frac{4r}{60} - \frac{p(t+r)}{60} + \frac{k(n-1)(t+r)}{60}\right)$$

usando w las siguientes ecuaciones pueden ser desarrolladas:

$$T = \frac{7}{5\,n\,w}$$

$$c = ns\,(gd + B\,(1+i))$$

$$e = \frac{n\,w}{c}$$

$$C = \frac{c}{n \, w}$$

donde:

T = es la razòn entre dias calendario y dias persona para terminar el proyecto.

c = es el costo laboral por dia de trabajo para un salario promedio s.

e = costo de eficiencia.

C = costo del proyecto por dia-persona.

n = número de individuos trabajando en el proyecto.

La naturaleza del tiempo de estudio de los parámetros queridos dificulta la recolecciòn de los datos y por tanto complica la aplicación del modelo Esterling. Del mismo modo, el modelo no considera de manera explicita las caracteristicas de los programas. Aún asi, el modelo Esterling es un enfoque único para la estimaciòn de proyectos de programaciòn y provee una idea bastante ùtil sobre la eficacia de un ambiente de programaciòn local.

3.4 Tecnica de costeo por lineas de codigo.

Cuando al gerente de desarrollo de programas se le solicita la estimaciòn del costo de un nuevo proyecto este deberà asociar dinero a algunas caracteristicas medibles de los programas. La tècnica de costeo de lineas de código (LOC, por sus siglas en inglès) asocia un costo a un estimado del número de lineas de còdigo fuente en un lenguaje de programaciòn que finalmente sera lo que se entregarà.

El alcance de los programas, considerado como parte de los pasos de planeaciòn de la programaciòn, proporciona una descripciòn de funciones mayores. El primer paso en la tècnica LOC especifica todas las funciones que serán desarrolladas en código fuente. Las funciones serán descompuestas (es decir, subdivididas en elementos funcionales mas pequeños) hasta que se logre establecer un estimado del número de lineas de còdigo fuente requeridas para desarrollar la función.

La estimación LOC esta representada por un rango de valores para cada función descompuesta utilizando datos históricos o (cuando todo lo demás falla) intuición. El encargado de la planeación estima valores LOC: un optimista, un típico y un pesimista para cada función. Cuando un rango de valores es especificado trae consigo un indicador implícito de un cierto grado de incertidumbre.

El número esperado (o promedio) de LOC y la desviación con respecto al valor esperado se ejemplifican a continuación.

El número esperado L se calcula como el promedio ponderado de los siguientes estimados de LOC: optimista a, típico m y un pesimista b. Por ejemplo,

$$L = \frac{a + 4m + b}{6}$$

en este ejemplo se le da un mayor peso al estimado típico y sigue una distribución de probabilidad beta.

Estamos asumiendo que existe muy poca probabilidad de que el actual resultado LOC se salga de los valores optimista y pesimista. De aquí que la desviación en el estimado LOC L(una medida de la varianza que se puede esperar en el resultado) se comporta de la siguiente forma:

$$L = \frac{b - a}{6}$$

donde los estimados a y b para n funciones de software estan incluídas en la suma. Debe aclararse que una desviación basada sobre datos inciertos (estimados) debe utilizarze con reservas.

Usando información histórica, el encargado de la planeación selecciona sus costos por LOC que caracteriza cada función de programación. Si esa información no esta disponible, puede aplicarse un valor promedio para $/LOC. El promedio $/LOC puede ser ajustado para reflejar los efectos inflacionarios, el incremento de complejidad del proyecto, gente nueva y otras características del desarrollo. Del mismo modo, el esfuerzo (expresado en meses persona por LOC) puede ser aplicado a la estimacion LOC. El costo y el esfuerzo son calculados para cada función, con lo que podemos calcular el costo total y esfuerzo total del proyecto.

¿Es nuestra estimaciòn correcta ? La ùnica respuesta razonable a esta pregunta es "no podemos estar seguros". Por esta razòn debe de aplicarse una segunda tècnica de costeo como una revisiòn del enfoque LOC.

3.5 Tecnica de costeo de esfuerzo por tarea.

El costeo de esfuerzo por tarea es la tècnica màs comùn para calcular el costo del desarrollo de cualquier proyecto de ingenieria a cada tarea del proyecto se le asigna un nùmero de dias hombre, meses hombre o años hombre. Un valor o un costo monetario es asociado a cada unidad de esfuerzo de tal forma que se obtiene un costo estimado.

Asi como en la técnica LOC, el costeo de esfuerzo por tarea comienza con la identificaciòn de las funciones de programaciòn definidas en la fase de planeaciòn del proyecto. Una serie de tareas de ingenieria de programación y anàlisis de requerimientos, diseño, codificaciòn y pruebas deben ser ejecutadas para cada funciòn. Las funciones y tareas de programaciòn relacionadas pueden representarse como parte de la tabla que se ilustra a continuaciòn:

funciones	tareas estimados en personas-meses					totales
total						
tarifa($)						
costo($)						

TECNICA DE COSTEO DE ESFUERZO POR TAREA
Matriz de Costos

El segundo paso para el costeo de esfuerzo por tarea establece el esfuerzo (meses hombre) requerido para la realización de cada tarea de ingeniería de programación para cada función. Esta información está comprendida en la matriz central de la tabla anteriormente presentada.

El tercer paso de la técnica asocia tarifas laborales (es decir, costo por unidad de esfuerzo) a cada tarea de ingeniería de programación. Es muy común que la tarifa laboral varíe de una tarea a otra. El personal de mayor jerarquía se encarga del análisis de requerimientos y del diseño inicial de tareas; mientras que la jerarquía inmediata se concentran en las tareas posteriores al diseño, codificación y pruebas iniciales.

El costo y esfuerzo para cada función y tarea de ingeniería de programación son calculadas en el último paso. Si el costeo de esfuerzo por tarea es calculado independientemente del costeo de líneas de código, contaremos con dos estimados para costo y esfuerzo que pueden compararse y consolidarse. Si ambas estimaciones muestran una concordancia razonable, es factible pensar que la estimación es confiable. Si por otro lado el resultado de las técnicas de costeo presentan poca concordancia se sugiere llevar a cabo una investigación y análisis mas amplio.

3.6 Costo automatizado.

Un número cada día mas grande de organizaciones de desarrollo de programas han desarrollado o adoptado un sistema de costeo automatizado para programas de computadora. Un modelo de costeo involucra información acerca del proyecto y de sus desarrolladores produciendo así una estimación del costo de los programas.

En un principio las características de la organización de desarrollo se describe de manera cuantitativa. Los recursos de desarrollo, la experiencia del equipo de trabajo, el número de responsabilidades concurrentes, la estructura organizacional, y otras características son utilizadas para describir un perfil de los desarrolladores. Para un ajuste del ambiente de desarrollo local, puede ejecutarse el modelo de costeo a la inversa usando información de proyectos terminados. Esta información puede ser utilizada para ajustar coeficientes del modelo para así reflejar de una mejor manera la organización local. Después de que se ha establecido el perfil de los desarrolladores, se procede a especificar las características del proyecto. Un perfil del proyecto esta constituido por el area de aplicación del sistema, la magnitud y complejidad del proyecto, obstáculos en el diseño y planeación, características de ejecución, limitaciones del equipo

y otros datos.

Los perfiles de los desarrolladores y el proyecto proporcionan una entrada inicial para el modelo de costeo. El modelo proporciona costos para el diseño, implentación y pruebas del sistema. Del mismo modo que un sistema de automatización de costeo proporciona información bàsica de costeo, también proporciona una programación de actividades del proyecto, una serie de reportes de variaciones que reflejan el riesgo de estimación, y una evaluación de la consistencia de las entradas. Si la información proporcionada para establecer los perfiles de desarrolladores y del proyecto son exactos el sistema de automatización de costeo puede producir excelentes resultados.

SLIM es un sistema de automatización de costeo basado en la curva de Rayleigh-Norden para el ciclo de vida de los programas, y el modelo de estimación de Putnam. SLIM aplica el modelo de los programas de Putnam, programación lineal, simulación estadística, y tècnicas PERT para obtener estimaciones de proyectos de los programas. El sistema permite al encargado de la planeación del sistema, la ejecución de las siguientes funciones en una sesión interactiva:

- Ajustar el ambiente de desarrollo local de los programas mediante la interpretación de datos históricos proporcionados por el encargado de la planeación.

- Crear un modelo de información de los programas que será desarrollado con determinadas caracteristicas, atributos del personal, y consideraciones ambientales.

- Utiliza una versión mas sofisticada y automatizada de, la tècnica de costeo LOC.

Una vez que el tamaño del sistema se establece (es decir, un LOC para cada función) SLIM calcula una desviación del tamaño, un indicador de la incertidumbre de la estimación, un perfil de sensitividad que indica la desviación potencial del costo y esfuerzo, y una verificación de consistencia con información recolectada de sistemas de tamaños similares.

El encargado de la planeación puede apoyarse en el análisis de programación lineal que considere los obstáculos que se presentan en el desarrollo para ambos costos y esfuerzos. Usando la curva Rayleigh-Norden como modelo, SLIM también proporciona una distribución mes a mes del costo y el esfuerzo de tal manera que los recursos humanos y monetarios puedan proyectarse.

3.7 Aplicacion al caso practico.

Nombre de la Tarea	No. Tarea	Nombre Recurso	Total Horas	Horas Laboradas	Horas Pendiente	Tiempo Total	Tiempo Cubierto	Tiempo por Laborar	Costo Total	Costo Cubierto	Costo por Laborar
Catalogo de datos generales	1	Analista 1	32.00	0.00	32.00	4.00	0.00	4.00	$361,636	$0	$361,636
Actualizacion y consultas	2	Analista 1	112.00	0.00	112.00	14.00	0.00	14.00	$1,272,727	$0	$1,272,727
Formas de actualizacion	3	Analista 1	80.00	0.00	80.00	10.00	0.00	10.00	$909,091	$0	$909,091
Catalogo de cuentas contables	4	Analista 1	48.00	0.00	48.00	6.00	0.00	6.00	$545,455	$0	$545,455
Actual. campos estadisticos	5	Analista 1	16.00	0.00	16.00	2.00	0.00	2.00	$181,818	$0	$181,818
de autorizacion a requisicion	7	Analista 2	80.00	0.00	80.00	10.00	0.00	10.00	$909,091	$0	$909,091
de autorizacion por compras	8	Analista 2	80.00	0.00	80.00	10.00	0.00	10.00	$909,091	$0	$909,091
de autorizacion por direccion	9	Analista 2	80.00	0.00	80.00	10.00	0.00	10.00	$909,091	$0	$909,091
de pedidos por compras	11	Analista 2	48.00	0.00	48.00	6.00	0.00	6.00	$545,455	$0	$545,455
de pedidos por direccion	12	Analista 2	48.00	0.00	48.00	6.00	0.00	6.00	$545,455	$0	$545,455
Actualizacion y consultas	14	Analista 2	64.00	0.00	64.00	8.00	0.00	8.00	$727,273	$0	$727,273
del estado del pedido	15	Analista 2	16.00	0.00	16.00	2.00	0.00	2.00	$181,818	$0	$181,818
de pedido por usuario	16	Analista 2	16.00	0.00	16.00	2.00	0.00	2.00	$181,818	$0	$181,818
Captura de recepción	19	Analista 3	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
actualización de recepción	20	Analista 3	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
importes brutos de contratos	21	Analista 3	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
de verificación de cartera	23	Analista 3	8.00	0.00	8.00	1.00	0.00	1.00	$90,909	$0	$90,909
Actualización y consultas	26	Analista 3	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
por tipo y estado del pedido	27	Analista 3	16.00	0.00	16.00	2.00	0.00	2.00	$181,818	$0	$181,818
Actualizacion de observaciones	29	Analista 3	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
por requisición	30	Analista 3	40.00	0.00	40.00	5.00	0.00	5.00	$454,545	$0	$454,545
de autorización por auditoría	31	Analista 3	40.00	0.00	40.00	5.00	0.00	5.00	$454,545	$0	$454,545
de autorización por dirección	32	Analista 3	40.00	0.00	40.00	5.00	0.00	5.00	$454,545	$0	$454,545
por requisición	34	Analista 3	40.00	0.00	40.00	5.00	0.00	5.00	$454,545	$0	$454,545
por auditoría	35	Analista 3	40.00	0.00	40.00	5.00	0.00	5.00	$454,545	$0	$454,545
por dirección	36	Analista 3	40.00	0.00	40.00	5.00	0.00	5.00	$454,545	$0	$454,545
Cargos y abonos de proveedores	39	Analista 1	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
de abonos con cheque	40	Analista 1	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
de compras y devoluciones	41	Analista 1	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
proveedores de materiales	43	Analista 2	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
proveedores de materias primas	44	Analista 2	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
proveedores ordenes de trabajo	45	Analista 2	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
proveedores de contratos	46	Analista 2	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
Actualizacion diaria	48	Analista 1	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
Estado de cuenta de proveedors	49	Analista 1	56.00	0.00	56.00	7.00	0.00	7.00	$636,364	$0	$636,364
antigüedad de saldos	50	Analista 1	96.00	0.00	96.00	12.00	0.00	12.00	$1,090,909	$0	$1,090,909
movimientos cuentas corrientes	51	Analista 1	96.00	0.00	96.00	12.00	0.00	12.00	$1,090,909	$0	$1,090,909
Consulta a proveedores	53	Analista 1	80.00	0.00	80.00	10.00	0.00	10.00	$909,091	$0	$909,091
Estado de proveedores anual	55	Analista 2	16.00	0.00	16.00	2.00	0.00	2.00	$181,818	$0	$181,818
Traspaso de saldos	56	Analista 2	16.00	0.00	16.00	2.00	0.00	2.00	$181,818	$0	$181,818
									$23,545,455	$0	$23,545,455

Reporte de Tareas Vs. Tiempo. Acercamiento por experiencia
Sistema de cuentas por pagar

	04-Ene-88	01-Feb-88	01-Mar-88	01-Abr-88	02-May-88	01-Jun-88	Total
SISTEMA DE CUENTAS POR PAGAR	$5,181,818.18	$5,454,545.45	$6,272,727.27	$6,363,636.36	$2,181,818.18	$90,909.09	$23,545,454.55
Maestro de Proveedores	$1,727,272.73	$1,545,454.55					$3,272,727.27
Catalogo de datos generale	$363,636.36						$363,636.36
Actualizacion y consultas	$1,272,727.27						$1,272,727.27
Formas de actualizacion	$90,909.09	$818,181.82					$909,090.91
Catalogo de cuentas contables		$545,454.55					$545,454.55
Actual. campos estadisticos		$181,818.18					$181,818.18
Control de Pedidos	$1,727,272.73	$1,818,181.82	$1,363,636.36				$4,909,090.91
Solicitudes	$1,727,272.73	$1,090,000.00					$2,727,272.73
de autorizacion x requis	$909,090.91						$909,090.91
de autorizacion por comp	$818,181.82	$90,909.09					$909,090.91
de autorizacion por direccion		$909,090.91					$909,090.91
Autorizaciones		$818,181.82	$272,727.27				$1,090,909.09
de pedidos por compras		$545,454.55					$545,454.55
de pedidos por direccion		$272,727.27	$272,727.27				$545,454.55
Actualizacion y consultas			$727,272.73				$727,272.73
Listados			$363,636.36				$363,636.36
del estado del pedido			$181,818.18				$181,818.18
de pedido por usuario			$181,818.18				$181,818.18
Recepción otros pedidos prov	$1,727,272.73	$272,727.27					$2,000,000.00
Captura y actualización	$1,727,272.73	$181,818.18					$1,909,090.91
Captura de recepción	$636,363.64						$636,363.64
actualización de recepci	$636,363.64						$636,363.64
Importes brutos de contr	$454,545.45	$181,818.18					$636,363.64
Listado		$90,909.09					$90,909.09
de verificación de captura		$90,909.09					$90,909.09
Material/Ordn.Trabajo/Contrato		$1,545,454.55	$2,090,909.09	$545,454.55			$4,181,818.18
Actualización y Consultas		$636,363.64					$636,363.64
Listado		$181,818.18					$181,818.18
por tipo y estado del pedido		$181,818.18					$181,818.18
Actualizacion de observaciones		$636,363.64					$636,363.64
Solicitudes		$90,909.09	$1,272,727.27				$1,363,636.36
por requisición		$90,909.09	$363,636.36				$454,545.45
de autorización por auditoria			$454,545.45				$454,545.45
de autorización por dirección			$454,545.45				$454,545.45
Autorizaciones			$818,181.82	$545,454.55			$1,363,636.36
por requisición			$454,545.45				$454,545.45
por auditoria			$363,636.36	$90,909.09			$454,545.45
por dirección				$454,545.45			$454,545.45
Afectación de movimientos		$272,727.27	$1,636,363.64				$1,909,090.91
Cargos y abonos de proveedores		$272,727.27	$363,636.36				$636,363.64
de abonos con cheque			$636,363.64				$636,363.64
de compras y devoluciones			$636,363.64				$636,363.64
Validación facturas para pago			$727,272.73	$1,818,181.82			$2,545,454.55
proveedores de materiales			$636,363.64				$636,363.64
proveedores de materias primas			$90,909.09	$545,454.55			$636,363.64
proveedores ordenes de trabajo				$636,363.64			$636,363.64
proveedores de contratos				$636,363.64			$636,363.64
Informe saldos de proveedores			$454,545.45	$1,909,090.91	$1,909,090.91	$90,909.09	$4,363,636.36
Actualizacion diaria			$454,545.45	$181,818.18			$636,363.64
Estado de cuenta de proveedors				$636,363.64			$636,363.64
Análisis				$1,090,909.09	$1,090,909.09		$2,181,818.18
antigüedad de saldos				$1,090,909.09			$1,090,909.09
movimientos cuentas corrientes					$1,090,909.09		$1,090,909.09
Consulta a proveedores					$818,181.82	$90,909.09	$909,090.91
Cierre anual de proveedores				$90,909.09	$272,727.27		$363,636.36
Estado de proveedores anual				$90,909.09	$90,909.09		$181,818.18
Traspaso de saldos					$181,818.18		$181,818.18
Total	$5,181,818.18	$5,454,545.45	$6,272,727.27	$6,363,636.36	$2,181,818.18	$90,909.09	$23,545,454.55

Asignacion de Recursos. Acercamiento por fórmulas de Rocha
Sistema de cuentas por pagar

Nombre de la Tarea	No. Tarea	Nombre Recurso	Total Horas	Horas Laboradas	Horas Pendientes	Tiempo Total	Tiempo Cubierto	Tiempo por Laborar	Costo Total	Costo Cubierto	Costo por Laborar
Catelogo de datos generales	1	Analista 1	26.00	0.00	26.00	3.25	0.00	3.25	$295,455	$0	$295,455
Actualizacion y consultas	2	Analista 1	44.00	0.00	44.00	5.50	0.00	5.50	$500,000	$0	$500,000
Formas de actualizacion	3	Analista 1	38.00	0.00	38.00	4.75	0.00	4.75	$431,818	$0	$431,818
Catalogo de cuentas contables	4	Analista 1	31.00	0.00	31.00	3.88	0.00	3.88	$352,273	$0	$352,273
Actual. campos estadisticos	5	Analista 1	20.00	0.00	20.00	2.50	0.00	2.50	$227,273	$0	$227,273
de autorizacion x requisicion	7	Analista 2	38.00	0.00	38.00	4.75	0.00	4.75	$431,818	$0	$431,818
de autorizacion por compras	8	Analista 2	38.00	0.00	38.00	4.75	0.00	4.75	$431,818	$0	$431,818
de autorizacion por direccion	9	Analista 2	38.00	0.00	38.00	4.75	0.00	4.75	$431,818	$0	$431,818
de pedidos por compras	11	Analista 2	31.00	0.00	31.00	3.88	0.00	3.88	$352,273	$0	$352,273
de pedidos por direccion	12	Analista 2	31.00	0.00	31.00	3.88	0.00	3.88	$352,273	$0	$352,273
Actualizacion y consultas	14	Analista 2	35.00	0.00	35.00	4.38	0.00	4.38	$397,727	$0	$397,727
del estado del pedido	15	Analista 2	20.00	0.00	20.00	2.50	0.00	2.50	$227,273	$0	$227,273
de pedido por usuario	16	Analista 2	20.00	0.00	20.00	2.50	0.00	2.50	$227,273	$0	$227,273
Captura de recepción	19	Analista 3	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
actualización de recepción	20	Analista 3	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
importes brutos de contratos	21	Analista 3	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
de verificación de captura	23	Analista 3	15.00	0.00	15.00	1.88	0.00	1.88	$170,455	$0	$170,455
Actualización y Consultas	26	Analista 3	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
por tipo y estado del pedido	27	Analista 3	20.00	0.00	20.00	2.50	0.00	2.50	$227,273	$0	$227,273
Actualizacion de observaciones	29	Analista 3	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
por requisición	30	Analista 3	29.00	0.00	29.00	3.63	0.00	3.63	$329,545	$0	$329,545
de autorización por auditoria	31	Analista 3	29.00	0.00	29.00	3.63	0.00	3.63	$329,545	$0	$329,545
de autorización por dirección	32	Analista 3	29.00	0.00	29.00	3.63	0.00	3.63	$329,545	$0	$329,545
por requisición	34	Analista 3	29.00	0.00	29.00	3.63	0.00	3.63	$329,545	$0	$329,545
por auditoria	35	Analista 3	29.00	0.00	29.00	3.63	0.00	3.63	$329,545	$0	$329,545
por dirección	36	Analista 3	29.00	0.00	29.00	3.63	0.00	3.63	$329,545	$0	$329,545
Cargos y abonos de proveedores	39	Analista 1	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
de abonos con cheque	40	Analista 1	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
de compras y devoluciones	41	Analista 1	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
proveedores de materiales	43	Analista 2	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
proveedores de materias primas	44	Analista 2	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
proveedores ordenes de trabajo	45	Analista 2	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
proveedores de contratos	46	Analista 2	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
Actualizacion diaria	48	Analista 1	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
Estado de cuenta de proveedore	49	Analista 1	33.00	0.00	33.00	4.13	0.00	4.13	$375,000	$0	$375,000
antigüedad de saldos	50	Analista 1	41.00	0.00	41.00	5.13	0.00	5.13	$465,909	$0	$465,909
movimientos cuentas corrientes	51	Analista 1	41.00	0.00	41.00	5.13	0.00	5.13	$465,909	$0	$465,909
Consulta a proveedores	53	Analista 1	38.00	0.00	38.00	4.75	0.00	4.75	$431,818	$0	$431,818
Estado de proveedores anual	55	Analista 2	20.00	0.00	20.00	2.50	0.00	2.50	$227,273	$0	$227,273
Traspaso de saldos	56	Analista 2	20.00	0.00	20.00	2.50	0.00	2.50	$227,273	$0	$227,273
									$14,102,273	$0	$14,102,273

Reporte de Tareas vs Tiempo. Acercamiento por fórmulas de Haehn.
Sistema de cuentas por pagar

	04-Ene-88	01-Feb-88	01-Mar-88	01-Abr-88	Total
SISTEMA DE CUENTAS POR PAGAR	$4,704,545.45	$4,477,272.73	$4,488,636.36	$431,818.18	$14,102,272.73
Maestro de Proveedores	$1,579,545.45	$227,272.73			$1,806,818.18
Catalogo de datos generale	$295,454.55				$295,454.55
Actualizacion y consultas	$500,000.00				$500,000.00
Formas de actualizacion	$431,818.18				$431,818.18
Catalogo de cuentas contab	$352,272.73				$352,272.73
Actual. campos estadisticos		$227,272.73			$227,272.73
Control de Pedidos	$1,647,727.27	$1,204,545.45			$2,852,272.73
Solicitudes	$1,295,454.55				$1,295,454.55
de autorizacion x requis	$431,818.18				$431,818.18
de autorizacion por comp	$431,818.18				$431,818.18
de autorizacion por dire	$431,818.18				$431,818.18
Autorizaciones	$352,272.73	$352,272.73			$704,545.45
de pedidos por compras	$352,272.73				$352,272.73
de pedidos por direccion		$352,272.73			$352,272.73
Actualizacion y consultas		$397,727.27			$397,727.27
Listados		$454,545.45			$454,545.45
del estado del pedido		$227,272.73			$227,272.73
de pedido por usuario		$227,272.73			$227,272.73
Recepción otros pedidos prov	$1,295,454.55				$1,295,454.55
Captura y actualización	$1,125,000.00				$1,125,000.00
Captura de recepción	$375,000.00				$375,000.00
actualización de recepci	$375,000.00				$375,000.00
Importes brutos de contr	$375,000.00				$375,000.00
Listado	$170,454.55				$170,454.55
de verificación de captu	$170,454.55				$170,454.55
Material/Orden.Trabajo/Contra	$181,818.18	$1,545,454.55	$1,227,272.73		$2,954,545.45
Actualización y Consultas	$181,818.18	$193,181.82			$375,000.00
Listado		$227,272.73			$227,272.73
por tipo y estado del pedido		$227,272.73			$227,272.73
Actualizacion de observaciones		$375,000.00			$375,000.00
Solicitudes		$150,000.00	$238,636.36		$988,636.36
por requisición		$329,545.45			$329,545.45
de autorización por auditoria		$329,545.45			$329,545.45
de autorización por dirección		$90,909.09	$238,636.36		$329,545.45
Autorizaciones			$988,636.36		$988,636.36
por requisición			$329,545.45		$329,545.45
por auditoria			$329,545.45		$329,545.45
por dirección			$329,545.45		$329,545.45
Afectación de movimientos		$1,125,000.00			$1,125,000.00
Cargos y abonos de proveedores		$375,000.00			$375,000.00
de abonos con cheque		$375,000.00			$375,000.00
de compras y devoluciones		$375,000.00			$375,000.00
Validación facturas para pago		$375,000.00	$1,125,000.00		$1,500,000.00
proveedores de materiales		$375,000.00			$375,000.00
proveedores de materias primas			$375,000.00		$375,000.00
proveedores ordenes de trabajo			$375,000.00		$375,000.00
proveedores de contratos			$375,000.00		$375,000.00
Informe saldos de proveedores			$1,681,818.18	$431,818.18	$2,113,636.36
Actualizacion diaria			$375,000.00		$375,000.00
Estado de cuenta de proveedors			$375,000.00		$375,000.00
Análisis			$931,818.18		$931,818.18
antigüedad de saldos			$465,909.09		$465,909.09
movimientos cuentas corrientes			$465,909.09		$465,909.09
Consulta a proveedores				$431,818.18	$431,818.18
Cierre anual de proveedores			$454,545.45		$454,545.45
Estado de proveedores anual			$227,272.73		$227,272.73
Traspaso de saldos			$227,272.73		$227,272.73
Total	$4,704,545.45	$4,477,272.73	$4,488,636.36	$431,818.18	$14,102,272.73

Tarifas de Programación
Sistema de cuentas por pagar

Los estudios anteriores fueron elaborados con las siguientes tarifas:

Recurso	Tarifa Diaria	Costo unitario
Analista 1	$11,364	$2,000,000.00 Mensual
Analista 2	$11,364	$2,000,000.00 Mensual
Analista 3	$11,364	$2,000,000.00 Mensual

CAPITULO 4 Estimacion y control de proyectos

4.1 Organizacion del equipo de desarrollo.

El éxito del desarrollo de un sistema de cómputo depende del esfuerzo del equipo de trabajo. El organigrama del equipo de trabajo varía de empresa a empresa y esto depende mas que nada del tamaño de la organización, sin embargo se puede considerar un mínimo común de personal, el cual se describe en los párrafos siguientes. El gerente de proyecto es el responsable de ordenar el programa de trabajo, determina los requerimientos, monitorea el proyecto durante su ciclo de vida e inicia procedimientos correctivos en caso de ser necesarios.

El gerente de proyecto deberá de tener fuertes habilidades administrativas y conocimientos técnicos para poder comunicarse con gente de negocios, técnicos, usuarios y personal de procesamiento de datos.

Su responsabilidad abarca la asignación de personal, determinación de requerimientos, interacción con el usuario, administración de tareas, estimación de tiempos, monitoreo y control de costos, y revisión del avanze del proyecto.

El trabajo del gerente de proyecto es organizar los recursos (personal, dinero, equipo) para alcanzar el objetivo definido en un periodo de tiempo. Para lograr esto debera usar la planeación, organización, coordinación, monitoreo, y motivación.

El analista debera tener experiencia en el problema a resolver para que este pueda ser entendido, definido y sea capaz de desarrollar una solución e implantarla. El analista es responsable de definir los requerimientos funcionales del usuario y de la traducción de estos a un diseño conceptual. El analista debe de tener la habilidad de trabajar solo y/o en conjunción con el usuario, personal técnico y otros miembros del equipo de trabajo.

En general el analista tendra las siguientes obligaciones:

- Definir los requerimientos de los usuarios (Trabajando en forma independiente o con personal de diferentes areas).

- Preparar el diseño conceptual en conjunto con especialistas de procesamiento de datos.

- Revisar el diseño detallado y las especificaciones de programación.

- Interactuar con personal técnico y no-técnico.

El soporte técnico es responsable de todos los aspectos y requerimientos técnicos del proyecto.

Deberá de definir el ambiente en que el sistema será desarrollado. Específicamente esta persona determinara los volumenes de información (entrada, salida, almacenamiento), las interfaces del sistema, las especificaciones óptimas de ejecución, los requerimientos de equipo y comunicaciones y, los requerimientos de programas ambientales (editores, compiladores, etc.)

En este trabajo el soporte técnico obtendra el conocimiento del medio ambiente en que la aplicación será instalada, operada y mantenida. Esta información es esencial cuando diferentes opciones son consideradas y seleccionadas.

El puesto de representantes de sistemas se puede dividir en dos niveles: el diseñador del programa y el programador. En la mayoria de los grupos de desarrollo el diseñador y el programador son la misma persona y desarrollan tanto el diseño del programa como la escritura del código.

El representante de sistemas es responsable del diseño detallado, la programación y el desarrollo de planes de prueba de los programas. Estas personas tienen mucha influencia en la calidad del sistema final.

El representante de sistemas tiene las siguientes responsabilidades:

- Diseño Estructural. Determina los bloques lógicos del sistema, que hace cada uno y como se encuentran relacionados para obtener el sistema total.

- Diseño de Interfaces. Selecciona los medios apropiados de comunicación entre los componentes del sistema. Por ejemplo: archivos intermedios, comunicación entre programas o variables externas.

- Diseño de Programas. Especifica de manera eficiente y práctica la forma de implementar cada componente del sistema.

En general el representante de sistemas debera de tener la habilidad para preparar especificaciones detalladas del sistema, realizar todas las tareas relacionadas con el desarrollo de código, preparar diseño y estructuras de programas, y desarrollar planes de actividades para el desarrollo (programación y pruebas).

4.2 Administracion del proyecto

Las actividades técnicas asi como las gerenciales son igualmente importantes para el èxito de un proyecto de programaciòn. Los gerentes controlan los recursos y el ambiente en que las actividades técnicas se suceden; asi los gerentes tienen la última responsabilidad de asegurar que los productos se entreguen a tiempo dentro del presupuesto estimado, además que los productos exhiban la funcionalidad y calidad que el cliente requiere. Entre otras actividades gerenciales se incluye la elaboraciòn de plan de trabajo, contrataciòn de proyectos, desarrollo de estrategias de mercado, asi como la contrataciòn y entrenamiento de personal.

Las actividades de la administraciòn de un proyecto comprenden los mètodos para organizar y seguir el curso a un proyecto; estimaciòn de costos, politicas de asignaciòn de recursos, control de presupuesto, definiciòn de logros del proyecto, determinaciòn del avance del proyecto, reasignaciòn de recursos y ajustes al calendario de trabajo, establecimiento de procedimientos de control de calidad, mantenimiento de las diversas versiones, promociòn de la comunicaciòn entre miembros del proyecto, comunicaciòn con los clientes, desarrollo de acuerdos contractuales con los clientes, y tambien asegurarse de la observancia de los tèrminos legales y contractuales del proyecto.

4.2.1 Inicio del proyecto.

La parte mas critica de un proyecto es el inicio. Durante el inicio el gerente de proyecto debe de establecer credibilidad con el usuario, el personal del equipo de trabajo, y deberà de definir los siguientes aspectos:

- Definiciòn de los objetivos del proyecto.
- Familiarizaciòn con el usuario.
- Planear y organizar las fases del proyecto.
- Establecer las funciones administrativas.

La definiciòn de los objetivos del proyecto sera realizada por el gerente del proyecto. Este mismo definirà los alcances del sistema, el tiempo proyectado en que se realizarà, y la forma en que el costo se evaluara. Esto sirve como base del plan de desarrollo.

El gerente de proyecto debera familiarizarse con los requerimientos del usuario, su organizaciòn, y el impacto or-

ganizacional que el desarrollo del sistema tiene antes de realizarse. La información que se deberá de obtener será una visión general del usuario, identificar al personal que estará involucrado con el proyecto en forma directa o indirecta.

Planear y organizar las fases del proyecto permite tener como resultado el plan del proyecto, que servirá de monitoreo del avance del desarrollo a través de todas sus fases.

El establecer las funciones administrativas deberá ser realizado por el gerente del proyecto. Este debera establecer los recursos que requiere por parte del usuario, tanto en información, personal, mobiliario como en equipo en caso que el usuario este proporcionando el computador.

4.2.2 Aspectos generales en proyectos de sistemas.

La producción y mantenimiento de productos de programación son tareas laboriosas, por lo que la productividad y la calidad son funciones directas de la capacidad y esfuerzo individuales. Existen dos aspectos en la capacidad: la competencia global del individuo y su familiaridad con el area particular de aplicación; programadores que se muestran competentes en el procesamiento de datos, suelen no serlo en áreas científicas, y de igual forma, un buen programador científico no es, forzosamente, un buen programador de sistemas. La falta de familiaridad con el area de aplicación puede implicar baja productividad y poca calidad.

Por tradición, se considera que la programación es una actividad individual y privada de modo que muchos programadores tienen poco contacto social y prefieren trabajar en forma aislada. Los programas son pocas veces tomados como documentos públicos, y los programadores en raras ocasiones analizan los detalles exactos de su trabajo de manera sistemática. Como resultado, es posible que los programadores malentiendan el papel de sus módulos ó subrutinas en el ambiente creciente del desarrollo de un sistema y que cometan errores que no sean detectados hasta después de algun tiempo. Muchas de las innovaciones de la ingeniería de programación, como la revisión de diseños, recorridos estructurados y los ejercicios de lectura de código, tienen como propósito lograr que los programas sean mas sociables, lo cual mejora la comunicación entre programadores.

Existen tres niveles o tipos de complejidad en un producto generalmente aceptados: programas de aplicación, programas de apoyo y programas del sistema operativo. En la primera clase se incluyen rutinas científicas o de procesamiento de datos. La clasificación de programas de apoyo comprende compiladores, ensambladores, ligadores y cargadores. Los programas del sistema operativo se relacionan con rutinas de comunicación de datos, procesamiento en tiempo real para sistemas de control y las

rutinas bàsicas del sistema operativo. El esfuerzo requerido para desarrollar y mantener un producto de programaciòn es una funciòn no lineal del tamaño del producto y su complejidad. Un producto del doble de tamaño o doblemente difícil que otro producto, usando cualquier mètrica diferente del esfuerzo puede requerir diez o tal vez 100 veces más esfuerzo para obtener el producto. La falla de no permitir un escalamiento no lineal en tamaño y complejidad es una de las primeras razones para un sobrecosto o una entrega retrasada en muchos proyectos de programaciòn.

En la ingenieria de programaciòn, como en otras disciplinas, los esquemas de representaciòn son de fundamental importancia. Una buena notaciòn puede aclarar las relaciones e interacciones de importancia, mientras que las notaciones deficientes complican e interfieren con la buena pràctica de la disciplina.

La flexibilidad en un programa es un gran beneficio y a su vez una gran fuente de dificultad. Como el còdigo fuente es fàcil de modificar, suele dificultarse la resoluciòn de diferentes detalles de los algoritmos o que el cliente solicite cambios que el gerente de proyecto esta dispuesto a aceptar. Los requisitos tambièn pueden cambiar debido a un escaso entendimiento del problema, o debido a factores econòmicos o politicos externos, ajenos al cliente o al experto. Por tal razòn, las notaciones y procedimientos que permitan seguir la secuencia y determinar el impacto de un posible cambio son necesarios para hacer visible el costo verdadero de una modificaciòn aparentemente pequeña al còdigo fuente.

El nivel tecnològico utilizado en un proyecto de programaciòn incluye aspectos como: selecciòn del lenguaje, ambiente computacional, pràcticas de programaciòn y herramientas de programaciòn disponibles.

Todo producto de programaciòn debe poseer un nivel elemental de confiabilidad; sin embargo, la alta confiabilidad solo se consigue con gran cuidado en el anàlisis, diseño, instrumentaciòn, pruebas y mantenimiento del producto de programaciòn. Se requieren tanto recursos humanos como equipo para obtener un aumento en la confiabilidad; lo anterior conduce a una reducciòn de productividad, medida solo en tèrminos de lineas de còdigo producido durante un mes.

En un proyecto de programaciòn un asunto comùn de difícil soluciòn es la incomprensiòn de la verdadera naturaleza del problema; existen diversos factores que contribuyen en esta falta de conocimiento. Por lo general, es el cliente quien no entiende realmente la naturaleza del problema, además de no entender las capacidades y limitaciones de la computaciòn; la mayoria de los clientes, y en general toda la gente, no han sido educados para pensar en tèrminos lògicos y algoritmicos e incluso, en ocasiones, desconocen sus verdaderas necesidades. Suele sucoder que el in-

geniero de programaciòn no entienda el àrea de aplicaciòn y , por lo tanto, tiene dificultad al comunicarse con el cliente debido a sus diferentes antecedentes educacionales, sus distintos puntos de vista y la comunicaciòn que cada uno maneja. Algunas veces incluso el cliente no es el usuario final del sistema y el ingeniero no tiene la oprtunidad real de estudiar el problema especifico del usuario. En ocasiones la naturaleza misma del problema no se revela hasta que el proyecto, o la mayor parte de èste, se ha terminado y se encuentra en operaciòn; otras veces, la soluciòn automatizada cambia la naturaleza del problema, para bien o para mal, y el cambio no es visto con claridad sino hasta que el sistema ha quedado instalado.

Aunque pareciera que un proyecto de programaciòn que requiere de un esfuerzo de seis meses-programador pueda ser completado por un programador en seis meses o seis programadores en un mes, los proyectos de programaciòn son sensibles no solo al total de esfuerzo requerido, sino tambièn al nùmero de personas comprendidas. El uso de seis programadores durante un mes probablemente sea menos eficiente que utilizar uno durante seis meses, y esto se debe a que la curva de aprendizaje para el grupo de seis programadores afectarà notablemente al proyecto. Por otro lado. el uso de dos programadores durante tres meses puede resultar màs eficiente que utiliza solo uno, debido a la retroalimentaciòn que cada programador reciba del otro. La productividad de un programador esta tambièn influida por el calendario de terminaciòn del proyecto. La determinaciòn del nivel òptimo de personal y el tiempo requerido para desarrollar las diferentes actividades en un proyecto de programciòn es un aspecto importante y dificil en la estimaciòn global de costos y recursos.

El ejercicio de la ingenieria de programaciòn requiere de una gran gama de habilidades y especialidades; por ejemplo, la obtenciòn de la informaciòn de los clientes con el fin de determinar sus necesidades requiere de la habilidad de comunicarse, de cierto tacto y diplomacia, asi como de un buen conocimiento del àrea de aplicaciòn.

Los estudios acerca de los factores motivadores de los programadores demuestran que los factores relacionados con el trabajo, como buen acceso a la màquina y un lugar silencioso para laborar, resultan ser mas importantes para el programador promedio que los factores relacionados con la clase, como estacionamiento privado y acceso a baños particulares. La mayoria de los programadores sienten que los aspectos positivos de su trabajo son las tareas que representen un reto a su variedad, y las oportunidades de crecer profesionalmente, mientras que los aspectos negativos son la ineptitud administrativa, politicas ,de la compañia y la burocracia organizacional.

La instrumentacion de un producto es solo un aspecto de la ingenieria de programaciòn; sin embargo, esta es la ùnica fase del

desarrollo y mantenimiento de un producto que se enseña en mucha escuelas. No es difícil encontrar las razones de esta carencia de habilidades; el impacto económico y social total de la ingeniería de programación se ha reconocido recientemente. Existe, pues, todavía un retraso impredecible entre la oferta educacional y la demanda industrial.

Los proyectos de programación son, por lo común, supervisados por gerentes que tienen poco conocimiento, si acaso lo tienen, acerca de la ingeniería de programacion: muchos de los problemas de esta ingeniería son únicos, incluso gerentes con experiencia de dirección de proyectos de equipos de cómputo, encuentran que los proyectos de ingeniería de programación son difíciles debido a las diferencias en la metodología de diseño, notaciones, herramientas, y otros aspectos. Por otro lado, la costumbre de promover a puestos administrativos de un proyecto de programación a individuos técnicamente competentes, con poca inclinación gerencial y sin entrenamiento administrativo, también suele producir resultados negativos. Muchas organizaciones ofrecen entrenamiento en dirección de proyectos a ingenieros de programación, pero esto no siempre conduce a resultados satisfactorios. En parte, esto se puede deber a que los programadores quieren poco contacto social en su trabajo, mientras que el gerente requiere de una adecuada capacidad de comunicación social.

La meta principal de la ingeniería de programción es el desarrollo de productos que cumplan con los requisitos del uso deseado idealmente, todo producto de programación debe proporcionar niveles óptimos de generalidad, eficiencia y confiabilidad. El esfuerzo desorganizado dedicado a mejorar marginalmente algunas características deseadas con una excesiva eficiencia, van en detrimento de la productividad del programador. Del mismo modo, la poca confiabilidad y eficiencia perjudican la calidad del producto. Se puede obtener un punto medio entre la productividad y los factores de calidad, mediante el mantenimiento dentro de las metas y requisitos establecidos para el producto durante la etapa de planeación.

El problema de mayor persistencia en la ingeniería de programacion es del crecimiento constante de las expectativas del producto. Existen dos apectos interrelacionados al respecto: primero, esta la preocupación de que tanto funcionalidad, confiabilidad y desempeño puede obtenerse con un esfuerzo determinado; en segundo lugar, se halla el aspecto relacionado con las limitantes de la tecnología de programación. Existe un progreso constante en el desarrollo de herramientas y técnicas para mejorar la calidad y productividad de un programador. Sin embargo, la diversidad, el tamaño y la complejidad de las aplicaciones crecen con más rapidez que nuestra capacidad de manejar tan creciente demanda.

4.2.3 Administracion efectiva.

La administración de un desarrollo de sistemas es sumamente complejo, es por ésto, que se recomienda seguir los siguientes pasos en la administración del proyecto.

- Educar y entrenar a la dirección superior, a los jefes de proyecto y a los programadores.

- Obligar al uso de lineamientos, procedimientos y documentación.

- Analizar los resultados de proyectos anteriores para determinar mecanismos eficientes.

- Definir la calidad del producto en términos de material suceptible de entrega.

- Establecer criterios prioritarios de éxito.

- Considerar contingencias.

- Desarrollar estimaciones de calendarios y costos en forma verdadera y exacta que sean aceptados por la gerencia y el cliente, y sujetarse a ellos.

- Seleccionar jefes de proyecto basándose en su capacidad para administrar proyectos de programación, más que en su habilidad técnica.

- Efectuar asignaciones de trabajo especificas a los expertos, aplicando estándares de desempeño en su trabajo.

- Se debe de preparar un reporte del estado del proyecto después de que se haya terminado una actividad o al menos cada cuatro semanas. Estos reportes son esenciales en la administración de proyectos muy grandes y deberán de ser estructurados de tal manera que permitan la intervención de todos los miembros del equipo de trabajo.

- El programa de trabajo desarrollado en la fase de definición de requerimientos no sera válido al inicio de la fase de diseño detallado por lo que se deberán de actualizar los planes de trabajo al inicio de cada fase y deberán de ser revisados y comentados con el usuario.

- El éxito de un proyecto se basa en el esfuerzo de sus participantes. Si no se da una gran comunicación entre los miembros del equipo de trabajo y los usuarios, existe un problema potencial. Los participantes deberán de trabajar como un verdadero equipo.

- El tiempo perdido durante las fases de diseño y programación no deben de ser recuperados reduciendo el tiempo asignado a las pruebas del sistema. La reducción en las pruebas provocará deficiencias durante la instalación y mantenimiento del sistema.

4.3 Planeacion y Control de Actividades

La falta de planeación es la causa principal de los retrasos en programación, incremento de costos, poca calidad, y altos costos de mantenimiento en los desarrollos de proyectos de programación.

Para evitar estos problemas se requiere de una planeación cuidadosa, tanto en el proceso de desarrollo, como en la operación del producto. Con frecuencia, se dice que es imposible una planeación inicial, porque la información precisa sobre las metas del proyecto, necesidades del usuario y restricciones no se conocen al inicio del proyecto.

Un producto de programación se entiende mejor conforme se van desarrollando el análisis, el diseño, y la programación; sin embargo, la falta de información no debe de ser razón para iniciar una planeación preliminar. Se debe de reconocer que los planes preliminares se modifican según vayan evolucionando los productos. La planeación para el cambio es uno de los aspectos claves con los que se logra el éxito.

4.3.1 Planeacion de Actividades.

Debe de desarrollarse una tabla de tiempos que indique la fecha de terminación de cada tarea y actividad del proyecto. El plan de trabajo puede sufrir alteraciones una vez que el proyecto esta siendo desarrollado pero es importante que los objetivos y tareas permanezcan bien definidas. Es común que los objetivos de implementacion del sistema tiendan a alejarse de la definición original por requerir mayores esfuerzos; por desgracia esto no se refleja hasta que los costos se salen de control. De aquí que se requiere una constante revisión de los objetivos originales del proyecto así como una confrontación con el plan de trabajo. Cualquier cambio en el alcance o costo del proyecto debe ser expuestos al cliente por escrito y adjunto al plan de trabajo convenido.

Insistimos que la solución no es como comunmente se piensa, el incrementar el número de recursos humanos a la elaboración del proyecto ya que frecuentemente las tareas se encuentran inter-

elacionadas de tal manera que podemos establecer una ruta crítica entre ellas. Esto quiere decir que hasta que no se encuentra totalmente terminada una actividad no se puede iniciar otra lo que genera horas hombre de trabajo muertas.

Es importante no dejarse presionar por los gerentes y usuarios al querer reducir los tiempos programados sin incrementar los costos. A menos que el alcance del proyecto sea menor tales cambios resultan en un deterioro en la calidad del desarrollo del sistema ocasionado así un sistema inoperante.

Para apoyar al plan de trabajo se requiere de un método de organización que requiera alcanzar la planeación y estimación de tiempo del proyecto, el control de la ejecución de tareas, el seguimiento de actividades terminadas, y la asignación de responsabilidades.

Existen varios métodos de planeación de actividades dependiendo del tamano de los proyectos pero tres de èstos son los comunmente utlizados:

- Gràficas de GANTT
- Gràficas de PERT
- División de la estructura del trabajo

4.3.2 Graficas de GANTT

Henry Gantt desarrollò en 1910 un mètodo de control de proyectos que hasta la fecha es utilizado y que es especialmente útil para la conceptualizacion de un plan de trabajo así como del progreso de este último. El enfoque que GANTT utiliza consiste en definir el producto final mediante la respuesta a los siguientes cuestionamientos:

- ¿Que queremos producir ?

- ¿Que es en sí el trabajo ?

- ¿Que tanto puede diseñarse el producto de tal manera que se logre el trabajo mas fàcil, productivo y efectivo ?

Una vez que el resultado final ha sido definido las actividades, su secuencia y tiempo de ejecución pueden ser representadas en una gràfica de barras. Las actividades aparecen en las columnas horizontales de la gràfica y sus tiempos asociados en un solo eje. La graficación proporciona una gran ayuda visual que permite identificar si existe alguna intersección en los tiempos asociados a la ejecucion de las actividades. Del mismo modo se

puede llevar a cabo un seguimiento mediante el uso de indicadores de estado dentro de la misma gràfica.

La ventaja de utilizar este mètodo es su sencillez, flexibilidad y el bajo costo que implica su diseno. Las gràficas de GANTT son especialmete ùtiles en proyectos donde las interelaciones de las actividades no son muy complejas. Tambièn resulta ser una gran herramienta cuando se trata de planear las responsabilidades en la implementacion de sistemas pequenos y medianos. Su uso en propuestas y reportes para la presentacion de tiempos estimados de activades termindas es muy comùn.

Podemos mencionar que una desventaja de las gràficas de GANTT es que no muestra las relaciones de depencia que existen entre las actividades. Por lo mismo, estas gràficas no son recomendables cuando se trata de grandes proyectos ya que no proporcionan evidencia de las tareas que hayan sobrepaso el tiempo asignado para su terminaciòn debido a su larga duraciòn. Cuando numerosas actividades estan involucradas ocasionan que el mantenimiento y actualizaciòn de estas graficas se vuelva tedioso.

4.3.3 Diagramas de PERT.

La tècnica de evaluaciòn y revisiòn de programas (PERT, porgrama de estimaciòn de ruta critica, por sus siglas en ingles) es utilizada para controlar proyectos compuestos por numerosas actividades independientes. Mediante el uso de un diagrama de red el proyecto es presentado a manera de una serie de pasos los cuales tienen asociados tiempos de terminacion. El uso de los diagramas de PERT es recomendable para grandes proyectos con numerosas actividades y largos periodos de tiempo ya que el diagrama no varia una vez terminado a menos que alguna revisiòn trascendental al plan del proyecto.

PERT desglosa al proyecto en eventos y actividades asociados en una secuencia. Los eventos son puntos en el tiempo en los que la actividad que los precede ha sido terminada y son representado en el diagrama como burbujas o nodos. Las actividades son tareas que requieren tiempo y recursos, los cuales son representados en el diagrama a traves de flechas.

La longitud de la flecha es irrelevante pero es importante que la actividad este acomodada en frente del evento que precede. A pesar de que muchas actividades pueden converger a un evento no debe asumirse que las mismas actividades son terminadas simultaneamente.

Después de hacer la red que muestra las relaciones entre las actividades, se lleva a cabo el estimado de tiempos y se anotan a lo largo de las flechas. El mìnimo tiempo requerido para la terminaciòn del proyecto suele llamarse "Ruta Critica", la cual

puede ser determinada mediante la adición de las horas correspondientes a las actividades que se encuentran a lo largo de la ruta màs larga que recorre desde el inicio hasta el fin del diagrama de red.

El diagrama de PERT puede ser utilizado para controlar actividades que deben ser completadas en secuencia aunque los tiempos de terminación sean desconocidos debido al uso de nuevas tecnologias de programación o configuraciones de equipo.

4.3.4 División de la estructura del trabajo.

La división de la estructura del trabajo (WBS, por sus siglas en ingles) fue desarrollado como respuesta a la necesidad de metodos de planeación y control de proyectos con diversos responsables directos, largos periodos de tiempo y la necesidad de coordinar grandes cantidades y tipos de recursos. El departamento de la defensa del los Estados Unidos ha sido uno de los principales usuarios de esta herramienta por mas de una década.

WBS es una herramienta muy valiosa para grandes y multifacéticos proyectos de servicios de programación que pueden involucrar al menos diez representantes de sistemas.

El método WBS es utilizado para subdividir un proyecto, por nivel, de tal manera que se produzcan paquetes de trabajo manejables que puedan ser utilizados para preparar presupuestos y asignar responsabilidades de trabajo. En un principio los componentes del proyecto estan diseñados en forma de una jerarquias de actividades y subactividades. Del mismo modo, se describe una jerarquía organizacional del equipo de trabajo del proyecto, lo cual se logra al dibujar lineas, en forma de matriz, que unen las subtareas con su correspondientes responsables.

Los paquetes de trabajo son los puntos resultantes en la matriz en donde ocurre una intersección entre la división de la estructura de trabajo y la estructura organizacional. Los paquetes de trabajo son los niveles mas bajos del desglose para estimación, seguimiento, control de ejecución, requerimiento de tiempo y gastos. Cada paquete de trabajo especifica:

- El contenido de trabajo de las tareas
- Ejecución de objetivos
- Requerimientos de recursos

WBS permite diferentes niveles de administración que permiten concentrarse en la planeacion, control y desarrollo de paquetes de trabajo interelacionados. Estos paquetes de trabajo pueden ser desglosados aún más para mostrar sus interelaciones en cuanto su tiempo de duración, de manera similar a la gráfica de PERT.

4.4 Estimación de la duración de un proyecto.

El esfuerzo total del proyecto se relaciona con el calendario de trabajo asignado para la terminación del proyecto. Varios investigadores han estudiado la cuestión del tiempo óptimo de desarrollo, y la mayoria concuerdan con que los proyectos de programación requieren mas esfuerzo si el tiempo de desarrollo se reduce o incrementa mas de su valor óptimo.

4.4.1 Importancia de la duracion de un proyecto

Durante un proyecto se desarrollan varios niveles de estimaciones de tiempo. El lider de proyecto es el responsable de todos las estimaciones del proyecto; sin embargo, se requiere un estimado detallado de los tiempos a nivel de actividad y subactividad.

La unidad de medida para la estimación del tiempo requerido es normalmente HORA-HOMBRE. También suele utilizarse el concepto de DIA-HOMBRE pero esto no es muy común. Los MESES-HOMBRE son utilizados como datos de resumen. El tiempo considerado como ocioso no debe de ser considerado en cada tarea sino debe ser cuantificado en la tiempo total estimado del proyecto.

Una unidad de medida de la productividad debe ser utilizada para establecer un estimado confiable de las horas-hombre requeridas para un programa de desarrollo del proyecto. Esta unidad de medida de productividad puede utilizarse no solo para la estimación de tiempos sino que es una herramienta también utilizada para determinar la eficiencia del equipo de trabajo del proyecto y apoyar a la gerencia en la evaluación de nuevas metodologias utilizadas en un proyecto.

Una medida aceptable de productividad debe generar resultados consistentes y debe ser independiente de cualquier tecnologia de desarrollo de programas. También debe proporcionar una evaluación significativa de la productividad tanto para los miembros del proyecto como para el usuario final.

Se han desarrollado muy pocas técnicas prácticas para fijar la cantidad de tiempo de programación y esfuerzo requerido en el proyecto. Las técnicas de estimación comunmente utilizadas in-

cluyen las líneas de código fuente, la métrica de Halstead y el análisis de punto de función.

4.4.2 Líneas de código fuente.

Este método comunmente utilizado sirve para medir la productividad de un programador mediante el conteo de las líneas de código que genera. Generalmente esto incluye autodocumentación mediante líneas de comentario.

La gran ventaja de este método es que puede ser aplicado fácilmente debido a su gran sencillez ya que el conteo de líneas puede ser generado automáticamente.

Por otro lado, la desventaja de este método es que el conteo de líneas de código fuente no es considerado un método que tenga una precisión adecuada para la estimación del tiempo de un proyecto. Programas muy sencillos pueden requerir muchas líneas de código fuente mientras que programas complejos pueden requerir pocas líneas. El método de líneas de código fuente es limitado si se hace una comparación entre programas escritos en diferentes lenguajes de programación y se convierte en un método de poca validez para el usuario final.

4.4.3 Metrica de Halstead.

Este es un modelo analítico que persigue medir el esfuerzo de la programación a partir de los siguientes principios:

- Supuestos en un programa.

n_1 : El número de diferentes operadores usados.

Un operador es un código de operación, un símbolo de operación, un símbolo aritmético, un símbolo de puntuación, un paréntesis, etc.

n_2 : El número de diferentes operandos usados.

Un operando es una constante o una variable.

N_1 : El número total de ocurrencias de los operadores de un programa.

N_2 : El número total de ocurrencias de los operandos de un programa.

Vocabulario : $n = n_1 + n_2$

Longitud : $N = N_1 + N_2$

S : constante, con valor aproximado a 18.

El esfuerzo en horas-hombre para desarrollar un programa se expresa en la siguiente formula con dichos parametros:

$$\text{Esfuerzo} = \frac{n_1 N_2 N \log n_2}{2 S n_2}$$

Estadisticamente, Halstead propone que la longitud de un programa puede estimarse a partir de su vocabulario mediante la siguiente ecuacion:

$$N \text{ estimada} = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

Esta relación tiene mayor precisión cuando los programas carecen de operandos sinónimos y de operandos utilizados por mas de una variable.

Otros supuestos son:

- Volumen del programa : $V = N \log_2 n$

Representa el número de lineas necesario para especificar un programa.

- Nivel del programa : puesto que un programa puede escribirse en forma mas o menos abstracta, el nivel trata de medir la relación entre su volumen en la forma mas abstracta posible y su volumen real. Un estimador para este nivel es:

$$L = \left(\frac{2}{n_1}\right)^{n} \cdot \frac{2}{N_2}$$

Las ventajas de este mètodo son que la medición es independiente del lenguaje de programación utilizado. Además esta medición puede efectuarse de manera automàtica dada la existencia de herramienta de programación que permiten la extracción de las palabras reservadas y los operandos.

Una limitante de este método es que esta basado en la sintaxis de un programa y no considera la presentación del programa ni su contenido. Otra desventaja es que el método resulta ser incomprensible para el usuario final y difícil de entender por el programador.

4.4.4 Analisis de puntos de funcion.

Durante 1970 se realizo una gran investigación que pretendia aislar las variables criticas que determinan la productividad de la programación.

Basado en los resultados de esta investigación, Allan J. Albrecht desarrollo un mètodo de análisis de puntos de función para medir la productividad y que puede ser utilizado para estimar tiempos de proyectos de sistemas de implementación grandes o pequeños. El mètodo de punto de función está basado en la premisa de que el valor bàsico de una aplicación es consistentemente proporcional a un conteo ponderado de las entradas externas, salidas externas, archivos maestros lógicos, interfases a otros sistemas, y consultas externas.

El punto de función por si mismo es un número sin dimensión que al ser aplicado a las aplicaciones sobre una base consistente proporciona una medida relativa de complejidad que puede ser utilizada para comparar la productividad entre diferentes proyectos de implementación. El método de punto de función no puede ser aplicado a proyectos de conversión.

Para ejecutar el càlculo de punto de función, las entradas, salidas, archivos maestros, consultas e interfases son contadas en un programa especifico de la aplicación. Al mismo tiempo que son contados, son clasificados en alguna de tres categorias de complejidad dependiendo de ciertos lineamientos.

Dependiendo del tipo de función (entradas, salidas, transacciones en línea, archivos maestros o interfases) será el criterio para determinar el grado de complejidad.

Para las entradas la complejidad va desde SENCILLA donde se tienen pocos datos involucrados, pocos archivos referenciados y no hay factores humanos; hasta COMPLEJO donde existen varios datos, gran cantidad de archivos referenciados y los factores humanos afectan al diseño. Como punto intermedio se maneja la complejidad PROMEDIO.

La siguiente tabla tiene como propósito el facilitar la clasificación ya que se muestra un cruce entre el número de campos o datos que se van a capturar y los archivos involucrados. Estos dos elementos son básicamente los que determinan el grado de complejidad de una entrada.

E N T R A D A S

archivos / campos	1-4	5-15	>15
0 - 1	S	S	P
2	S	P	C
> 2	P	C	C

Para el caso de las salidas podemos nombrar algunos puntos que son claves para la clasificación en alguna de las tres categorías. Se considera de complejidad SENCILLO aquella salida o reporte que contiene pocas columnas y que además representa una mínima transformación de los datos. Una salida PROMEDIO tiene varias columnas, cálculo de subtotales y varias transformaciones de los datos. Finalmente, se dice que una salida es compleja cuando involucra una gran transformación de datos, referencia a varios archivos y muchas consideraciones significativas en cuanto a desempeño.

Asi pues, la siguiente tabla permite identificar la categoría que se deberá asignar a la salida en base a los datos o campos así como a los archivos involucrados.

S A L I D A S

archivos / campos	1-5	6-19	>19
0 - 1	S	S	P
2 - 3	S	P	C
>= 4	P	C	C

Existen algunos factores adicionales que deben considerarse como son: los textos (encabezados, titulos,etc), el número de subtotales, la transformación de datos y el desempeño de la aplicación.

Las llamadas consultas se refieren a cualquiera de los siguientes situaciones: a) consultas del usuario sin ninguna actualización. b) mensajes de ayuda o enseñanza. o c) menues.

Para determinar la complejidad deben seguirse tres pasos:

1) Clasificarlo como entrada, es decir, tomar en cuenta solo aquellos campos que se requiere sean proporcionados por el usuario.

2) Clasificarlo como salida, es decir, tomar en cuenta solo aquellos campos que constituirán la respuesta a la solicitud del usuario.

3) Escoger la clasificación de complejidad que sea mayor entre las obtenidas en los dos puntos anteriores.

Resulta de gran utilidad el uso de las siguientes tablas para la obtención de los resultados requeridos en los puntos 1) y 2).

E N T R A D A S

archivos / campos	1-4	5-15	>15
0 - 1	S	S	P
2	S	P	C
> 2	P	C	C

S A L I D A S

archivos / campos	1-5	6-159	>19
0 - 1	S	S	P
2 - 3	S	P	C
> 3	P	C	C

La categoria de SENCILLA es atribuida a archivos cuando éstos están definidos por pocos tipos de registros, pocos campos o datos, no requieren de consideraciones especiales sobre desempeño o recuperaciòn, y tienen pocos niveles jeràrquicos.

La categoria de COMPLEJO es alcanzada por los archivos cuando tienen varios tipos de registros, gran cantidad de campos o datos, requieren consideraciones especiales para su desempeño y recuperaciòn y manejan varios niveles jeràrquicos.

En base a lo anterior se puede concluir que la clasificaciòn esta basada en la relaciòn que exista entre el diferente tipo de registros asi como el nùmero de campos o datos que lo definan.

A R C H I V O S

tipo reg./ campos	1-19	20-50	>50
1	S	S	P
2 - 5	S	P	C
>= 6	P	C	C

Algunos factores a considerar son el desempeño, la recuperaciòn y respaldo, y el criterio de bùsqueda o lectura.

La complejidad para las interfases se determina siguiendo el mismo criterio utilizado para los archivos, mas si el mismo archivo es accesado por mas de una aplicación, entonces debe contarse en cada función.

I N T E R F A S E S

tipo reg./ campos	1-19	20-50	>50
1	S	S	P
2 - 5	S	P	C
> 5	P	C	C

Los factores adicionales a considerar son el desempeño de la aplicación, el criterio de búsqueda o lectura, la recuperación y respaldo, la distribución múltiple, y la conversión.

Un peso es entonces asignado a los elementos contados dependiendo del tipo de elemento en cuestión (entrada, salida, archivo maestro, etc.) y en el nivel de complejidad de esto. La suma de estos pesos representa un total de puntos de función no ajustados.

FUNCION	SENCILLA	PROMEDIO	COMPLEJO
Entradas	3	4	6
Salidas	4	5	7
Consultas	3	4	6
Archivos	7	10	15
Interfases	5	7	10

Un factor de ajuste es aplicado al total de puntos de función no ajustados con el propósito de llegar a un total final de punto de función. El valor del factor de ajuste es calculado mediante una fórmula aplicada a los pesos específicos. Los pesos son asignados con el propósito de reflejar los factores adicionales que afectan la complejidad del programa.

Los factores incluyen el grado de complejidad de la comunicación de datos, el procesamiento distribuido, la ejecución de objetivos, la configuración, el volumen de transacciones, la entrada de datos en línea, las transacciones interactivas en línea, la actualización en línea de archivos maestros, la com-

plejidad de procesamiento, el diseño para reutilización, la facilidad de conversión e implementación, la instalación en multiples sitios, y la facilidad de cambio y uso.

Estos ajustes pueden modificar los puntos de función no ajustados en más ó menos veinticinco por ciento resultando en un número que representa el total de puntos de función para el programa. Programas medianos y grandes generalmente tienen entre quinientos y ochocientos puntos de función y tomaría aproximadamente dos horas el contar los valores de estos puntos en la documentación del diseño.

Los programas que aun no estan completamente documentados requieren un elemento de subjetividad para determinar el número y valor de funciones. Como siempre, los programas totalmente documentados, con especificaciones de diseño detalladas o con codigo fuente disponible tendrán una variancia de subjetividad menor al diez por ciento.

Cuando el total final de punto de función se ha obtenido se divide por el total de horas de trabajo para asi determinar puntos de función por hora, que es precisamente la herramienta de medición de productividad. Las horas de trabajo representan el número de persona-hora invertidas por los miembros del equipo del proyecto a lo largo de las primeras cuatro fases del ciclo de vida de implementación del sistema.

El procedimiento para calcular los puntos de función puede resumirse de la siguiente forma:

1.- Listar las funciones y estimar su complejidad.

	sencillas	promedio	complejas
a) Entradas			
entrada 1	...	...	...
:			
entrada n	...	...	...
Totales a)	-----	-----	-----
b) Salidas			
salida 1	...	...	...
:			
salida n	...	...	...
Totales b)	-----	-----	-----
c) Consultas			
consulta 1	...	...	...
:			
consulta n	...	...	...
Totales c)	-----	-----	-----
d) Archivos			

archivo 1	...	...	...
:			
archivo n	...	...	...
Totales d)	_____	_____	_____

e) Interfases

interfase 1	...	...	...
:			
interfase n	...	...	...
Totales e)	_____	_____	_____

2.- Calcular los puntos por función.

a) Entrada

			Puntos por función
_____ sencilla	X	3	_____
_____ promedio	X	4	_____
_____ compleja	X	6	_____

b) Salida

			Puntos por función
_____ sencilla	X	4	_____
_____ promedio	X	5	_____
_____ compleja	X	7	_____

c) Consultas

			Puntos por función
_____ sencilla	X	3	_____
_____ promedio	X	4	_____
_____ compleja	X	6	_____

d) Archivos

			Puntos por función
_____ sencilla	X	7	_____
_____ promedio	X	10	_____
_____ compleja	X	15	_____

e) Interfases

			Puntos por función
_____ sencilla	X	5	_____
_____ promedio	X	7	_____
_____ compleja	X	10	_____

Total por puntos de función _____

3.- Obtener el grado de influencia en los factores de ajuste

a)	Inexistente	0
b)	Influencia incidental	1
c)	Influencia moderada	2
d)	Influencia promedio	3
e)	Influencia significativa	4
f)	Influencia poderosa	5

4.- Calcular el valor de ajuste (VA):

Valor de ajuste = 0.65 + (0.01 X Grado de influencia)

5.- Ajustar los puntos por función:

Puntos de función = Total de puntos X Valor de ajuste
AJUSTADOS por función

6.- Calcular las horas-hombre:

$$\text{Horas-hombre} = \left[\frac{\text{Puntos de función ajustados}}{1.932}\right]^{1/0.5912}$$

En instalaciones con COBOL cada punto consume aproximadamente 20 horas, si se trata de 500 puntos, y 40 horas si se trata de 1000 puntos, esto, debido al tiempo que se invierte en integración y coordinación. En instalaciones con lenguajes de cuarta generación se reduce, aproximadamente, una cuarta parte las horas-hombre.

Para el caso específico del lenguaje de cuarta generación: LINC, se tiene la siguiente formula para determinar el número de horas-hombre (HH) por puntos de función (PF):

$$HH \text{ por } PF = 1 + e^{(\text{puntos de funcion} - 200) / 2500}$$

Esta fórmula se deriva de la experiencia que los mismos creadores del lenguaje han tenido en el desarrolo de proyectos utilizando esta herramienta.

A continuación se muestra una gráfica del comportamiento de esta función con el propósito de visualizar las horas-hombre por puntos de función de manera directa.

7.- Calcular los meses-hombre(MH)

Meses-hombre = horas-hombre / 140

8.- Distribuir el esfuerzo total de meses-hombre (MH) por fase:

MH	Análisis	= 0.08 x total MH
MH	Diseño inicial	= 0.12 x total MH
MH	Diseño detallado	= 0.17 x total MH
MH	Programación y pruebas	= 0.51 x total MH
MH	Integración del sistema	= 0.12 x total MH

9.- Calcular los meses con base en el número de recursos asignados al proyecto.

Meses de análisis	= MH análisis / Recursos
Meses de diseño inicial	= MH diseño inicial / Recursos
Meses de diseño detallado	= MH diseño detallado / Personal
Meses de programación	= MH programación / Recursos
Meses de integración	= MH integración / Recursos

10.- Calendarizar las fecha con base en el inicio de cada etapa.

ETAPA	%	MES-HOMBRE	RECURSOS	MESES	F.INI	F.FIN
Análisis	8					
Diseño inic.	12					
Diseño det.	17					
Programación	51					
Integración	12					

Las principales ventajas de este método son el proporcionar:

- Un medio efectivo para la estimación de tiempo y esfuezo de trabajo requerido al escribir un programa.

- Comparaciones significativas de productividad entre proyectos e instalaciones.

- Una herramienta que puede ser utilizada para evaluar nuevas metodologias de programación.

- Mètodo de medición de la productividad del grupo de desarrollo del proyecto.

- Identificación del tiempo de desarrollo que pudo haber escapado de una varianza aceptable.

4.4.5 Paquetes de Computadora.

Existen en la actualidad una gran cantidad de paquetes de computadora para la administración de proyectos; la mayoria de estos usan tècnicas desarrolladas durante la primera guerra mundial aunque existen algunos que utilizan tècnicas actuales como los mètodos de ruta critica y PERT.

En general estos paquetes consisten en una base de datos con calendarios semanales y mensuales para la asignación de cargas de trabajo y la posibilidad de obtener una gran cantidad de gràficas que permiten conocer el avanze y distribución de recursos del proyecto durante su tiempo de vida, tal es el caso de los programas Time Line III de Symantec Corp. y Harvard Project de Software Publishing Corp.

4.5 Aplicacion al caso práctico.

APLICACION DE LA TECNICA DE PUNTOS DE FUNCION

SISTEMA: Cuentas por pagar

MODULO: 1 - SELECCION DE PROVEEDOR

TIPO DE FUNCION: ENTRADAS

TIPO DE FUNCION: ARCHIVOS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
PROVEEDORES	1		
TOTALES:	1	0	0

TIPO DE FUNCION: SALIDAS

TIPO DE FUNCION: CONSULTAS

TIPO DE FUNCION: BATCH

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
BUSQUEDA PROVEEDOR		1	
INVENTARIO	1		
ACTUALIZACION PROVEEDOR	1		
TOTALES:	2	1	0

APLICACION DE LA TECNICA DE PUNTOS DE FUNCION

SISTEMA: Cuentas por pagar

MODULO: 2 - CONTROL DE ORDENES

TIPO DE FUNCION: ENTRADAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
ALTA ORDENES COMPRA O SERVICIOS		1	
BAJA ORDENES COMPRA O SERVICIOS		1	
CAMBIOS ORDENES COMPRA O SERVICIOS		1	
AUTORIZACION POR REQUERIMIENTOS COMPRAS, DIRE		1	
TOTALES:	0	4	0

TIPO DE FUNCION: ARCHIVOS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
ORDENES DE MATERIAS PRIMAS	1		
ORDENES DE COMPRAS O SERVICIOS	1		
TOTALES:	2	0	0

TIPO DE FUNCION: SALIDAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
ORDEN DE COMPRA	1		
TOTALES:	1	0	0

TIPO DE FUNCION: CONSULTAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
CONSULTA CONSULTAS ORDENES COMPRA O SERVICIOS	1		
TOTALES:	1	0	0

TIPO DE FUNCION: BATCH

APLICACION DE LA TECNICA DE PUNTOS DE FUNCION

SISTEMA: Cuentas por pagar

MODULO: 3 - CONTROL DE REMISIONES

TIPO DE FUNCION: ENTRADAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
CAPTURA DE REMISION		1	
BAJAS DE REMISION		1	
CAMBIOS DE REMISION		1	
TOTALES:	0	3	0

TIPO DE FUNCION: ARCHIVOS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
REMISIONES	1		
CARTERA PROVEEDOR	1		
MATERIAS PRIMAS	1		
POLIZA PASIVOS		1	
ESTADISTICA PROVEEDORES	1		
TOTALES:	3	1	0

TIPO DE FUNCION: SALIDAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
ACEPTACION DE REMISION	1		
TOTALES:	1	0	0

TIPO DE FUNCION: CONSULTAS

TIPO DE FUNCION: BATCH

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
ACTUALIZACION DE REMISIONES	1		
TOTALES:	1	0	0

APLICACION DE LA TECNICA DE PUNTOS DE FUNCION

SISTEMA: Cuentas por pagar

MODULO: 4 - ELABORACION CONTRARECIBO CXP

TIPO DE FUNCION: ENTRADAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
CAPTURA DE MERCANCIA RECIBIDA			1
TOTALES:	0	0	1

TIPO DE FUNCION: ARCHIVOS

TIPO DE FUNCION: SALIDAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
CONTRARECIBO	1		
TOTALES:	1	0	0

TIPO DE FUNCION: CONSULTAS

TIPO DE FUNCION: BATCH

APLICACION DE LA TECNICA DE PUNTOS DE FUNCION

SISTEMA: Cuentas por pagar

MODULO: 5 - CONTROL DE FACTURAS

TIPO DE FUNCION: ENTRADAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
CAPTURA DE FACTURAS			1
TOTALES:	0	0	1

TIPO DE FUNCION: ARCHIVOS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
FACTURAS		1	
TOTALES:	0	1	0

TIPO DE FUNCION: SALIDAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
FACTURA		1	
TOTALES:	0	1	0

TIPO DE FUNCION: CONSULTAS

TIPO DE FUNCION: BATCH

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
GENERACION DE CUENTA POR PAGAR		1	
TOTALES:	0	1	0

APLICACION DE LA TECNICA DE PUNTOS DE FUNCION

SISTEMA: Cuentas por pagar

MODULO: 6 - SELEC. DE PROVEEDOR A SALDAR CXP

TIPO DE FUNCION: ENTRADAS

TIPO DE FUNCION: ARCHIVOS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
CUENTA POR PAGAR	1		
CARTERA DE PROVEEDOR	1		
PROVEEDOR	1		
TOTALES:	3	0	0

TIPO DE FUNCION: SALIDAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
CTA. POR PAGAR VENCIDA	1		
TOTALES:	1	0	0

TIPO DE FUNCION: CONSULTAS

TIPO DE FUNCION: BATCH

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
GENERACION DE CUENTA POR PAGAR VENCIDAS	1		
TOTALES:	1	0	0

APLICACION DE LA TECNICA DE PUNTOS DE FUNCION

SISTEMA: Cuentas por pagar

MODULO: 7 - AFECTACION A PROVEEDOR

TIPO DE FUNCION: ENTRADAS

TIPO DE FUNCION: ARCHIVOS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
POLIZA-CHEQUE		1	
TOTALES:	0	1	0

TIPO DE FUNCION: SALIDAS

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
CHEQUE		1	
TOTALES:	0	1	0

TIPO DE FUNCION: CONSULTAS

TIPO DE FUNCION: BATCH

DESCRIPCION	DIFICULTAD SIMPLE	MEDIA	COMPLEJA
SELECCION DE SALDOS	1		
AFECTACION DE CARGOS Y ABONOS		1	
AFECTACION DE ABONOS CON CHEQUE		1	
TOTALES:	1	2	0

APLICACION DE LA TECNICA DE PUNTOS DE FUNCION

SISTEMA: CUENTAS POR PAGAR

	MODULO 1 S	M	C	FACTOR S	M	C	PUNTOS DE FUNCION
ENTRADAS	0	0	0	3	4	6	0
ARCHIVOS	1	0	0	7	10	15	7
SALIDAS	0	0	0	4	5	7	0
CONSULTAS	0	0	0	3	4	6	0
BATCH	2	1	0	10	20	30	40
Total:							47

	MODULO 2 S	M	C	FACTOR S	M	C	PUNTOS DE FUNCION
ENTRADAS	0	4	0	3	4	6	16
ARCHIVOS	2	0	0	7	10	15	14
SALIDAS	1	0	0	4	5	7	4
CONSULTAS	1	0	0	3	4	6	3
BATCH	1	1	1	10	20	30	60
Total:							97

	MODULO 3 S	M	C	FACTOR S	M	C	PUNTOS DE FUNCION
ENTRADAS	0	3	0	3	4	6	12
ARCHIVOS	4	1	0	7	10	15	38
SALIDAS	1	0	0	4	5	7	4
CONSULTAS	0	0	0	3	4	6	0
BATCH	1	0	0	10	20	30	10
Total:							64

	MODULO 4 S	M	C	FACTOR S	M	C	PUNTOS DE FUNCION
ENTRADAS	0	0	1	3	4	6	6
ARCHIVOS	0	0	0	7	10	15	0
SALIDAS	1	0	0	4	5	7	4
CONSULTAS	0	0	0	3	4	6	0
BATCH	0	1	0	10	20	30	20
Total:							30

	MODULO 5			FACTOR			PUNTOS DE FUNCION	
	S	M	C	S	M	C		
ENTRADAS	0	0	1	3	4	6		6
ARCHIVOS	0	1	0	7	10	15		10
SALIDAS	0	1	0	4	5	7		5
CONSULTAS	0	0	0	3	4	6		0
BATCH	0	1	0	10	20	30		20
Total:								41

	MODULO 6			FACTOR			PUNTOS DE FUNCION	
	S	M	C	S	M	C		
ENTRADAS	0	0	0	3	4	6		0
ARCHIVOS	3	0	0	7	10	15		21
SALIDAS	1	0	0	4	5	7		4
CONSULTAS	0	0	0	3	4	6		0
BATCH	1	0	0	10	20	30		10
Total:								35

	MODULO 7			FACTOR			PUNTOS DE FUNCION	
	S	M	C	S	M	C		
ENTRADAS	0	0	0	3	4	6		0
ARCHIVOS	0	1	0	7	10	15		10
SALIDAS	0	1	0	4	5	7		5
CONSULTAS	0	0	0	3	4	6		0
BATCH	1	2	0	10	20	30		50
Total:								65

TOTAL DE PUNTOS DE FUNCION: 379

APLICACION DE LA TECNICA DE PUNTOS DE FUNCION

SISTEMA: CUENTAS POR PAGAR

PUNTOS DE
FUNCION : 379

HRS-HOMBRE X
PTO. DE FUNC: 2.0

ESFUERZO EN
HORAS-HOMBRE: 396

ESFUERZO EN
MESES-HOMBRE: 3.3

DURACION ESTIMADA
CON 3 ANALISTAS : 1 MES

SISTEMA: CUENTAS POR PAGAR

FASE	% PROY	*** M-1	M-2	M-3	M O D U L O M-4	M-5	M-6	*** M-7	TOTAL M-H	COSTO M-H	COSTO TOTAL
DEFIN. REQUERIM.	8	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.5	7,000	3,643
DISENO CONCEP.	12	0.1	0.1	0.1	0.0	0.0	0.0	0.1	0.5	7,000	3,432
DISENO DETALLADO	12	0.1	0.1	0.1	0.0	0.0	0.0	0.1	0.5	7,000	3,432
PROGRAMACION	15	0.1	0.2	0.1	0.0	0.1	0.0	0.1	0.6	5,000	3,064
PRUEBAS UNIT.	24	0.2	0.3	0.2	0.1	0.1	0.1	0.2	1.0	5,000	4,903
PRUEBAS INTEGR.	14	0.1	0.2	0.1	0.0	0.0	0.0	0.1	0.6	5,000	3,432
PRUEBAS TOTALES	15	0.0	0.2	0.1	0.0	0.1	0.0	0.1	4.0	7,000	27,752
	100	0.1	1.6	1.0	0.5	0.7	0.6	1.1	3.3		49,659

CAPITULO 5 Analisis

5.1 Actividades basicas

El anàlisis es el proceso de transformar una cadena de información acerca de las operaciones corrientes o actuales y de los nuevos requerimientos a una descripción ordenada y rigurosa de un sistema para ser construido. Esta descripción es tambien llamada especificaciones funcionales del sistema.

El anàlisis estructurado tiene como propósito fundamental especificar, en la forma màs precisa posible, los requerimientos del usuario para un programa o conjunto de programas.

La fase del anàlisis concierne principalmente con la generación de especificaciones de el sistema para ser construido.

Esta fase està compuesta de las actividades de interacción con el usuario, estudio del medio ambiente actual, negociación, diseño externo del nuevo sistema, diseño de formatos de E/S(entrada y salida), estudio de costo-beneficio, escritura de las especificaciones, y estimación.

El análisis estructurado es una disciplina moderna para conducir la fase de anàlisis. En el contexto de el ciclo de vida del proyecto su ùnica diferencia aparente es un nuevo producto llamado "especificaciones estructuradas". Esta nueva clase de especificaciones tienen las siguientes caracteristicas:

- Es gràfica y està compuesta mayormente de diagramas.

- Es particionada y no es una sola especificación sino una red conectada de miniespecificaciones.

- Es de arriba-abajo (TOP-DOWN) y presentada en modo jeràrquico.

- Es mantenible y una especificación puede ser actualizada para reflejar cambios en los requerimientos.

- Es un modelo en papel del sistema y el usuario puede trabajar con el para perfeccionar su visión de las operaciones tal y como seràn en el nuevo sistema.

5.1.1 Modelado del sistema

El anàlisis estructurado nos permitirà construir un modelo

físico en papel del sistema actual y usarlo para perfeccionar el entendimiento del medio ambiente. Esto es con el fin de tener una representación verificable de las operaciones actuales y deberán ser fácilemente entendibles por el grupo de desarrollo.

El equivalente lógico del modelo físico es uno que esta divorciado del los "comos" de las operaciones actuales y en su lugar se concentra los "ques".

El desarrollo del modelo del nuevo sistema es el trabajo mas importante de la fase del análisis, pues consiste en la invención del nuevo sistema. El nuevo modelo presenta al sistema como un grupo particionado de procesos elementales. El detalle de estos procesos son ahora especificados utilizando una miniespecificación por proceso.

Las restricciones del modelo permiten al analista establecer los límites entre el hombre y la máquina y se establece el alcance de la automatización. Típicamente esto se hace mas de una vez con el objeto de dar varias alternativas para la selección de procesos.

5.1.2 Seleccion de una opcion

Los estudios de costo y beneficio son realizados para cada una de las opciones. Cada uno de los modelos tentativos junto con sus parámetros asociados de costo-beneficio son presentados en forma cuantificada.

Las opciones cuantificadas son analizadas y una de ellas es seleccionada como la mejor.

De las opciones propuestas para llegar a la solución del nuevo modelo, es necesario que sean analizadas a fondo cada una de ellas, esto se hace con la finalidad de saber cual de estas opciones es la mas adecuada en cuanto a tiempo, confiabilidad y portabilidad que se tendrá del nuevo modelo. Esto se realiza con el objeto de poder seleccionar ó elegir la opción mas conveniente que requiere el nuevo modelo y de que cumpla con los requerimientos del mismo.

Ahora todos los elementos de la especificación estructurada son armados y presentados. Esto da como resultado el nuevo modelo físico seleccionado, el conjunto integrado de mini-

especificaciones, las tablas de contenido, y resumenes.

5.2 Modelado mediante diagramas de flujo de datos

El modelo al que se ha hecho referencia para la representación del sistema es un modelo gráfico. En la convención de análisis estructurado este modelo se logra con el uso de un diagrama de flujo de datos (DFD) , de un diccionario de datos (DD) y de las miniespecificaciones.

El DFD es la principal herramienta gráfica del análisis y tiene como objeto mostrar las transformaciones de los datos a medida que estos fluyen a través de los procesos del diagrama, es decir, ayuda a anlizar los cambios que ocurran a los datos de entrada a fin de lograr la salida deseada.

El DFD es un instrumento de modelación que permite mostrar a un sistema como una red de subsistemas conectados unos con otros mediante flujos de datos que muestran las relaciones etre subsistemas.

El DD es un conjunto de definiciones de las interfaces declaradas en el DFD. Se define cada una de esas interfases en terminos de sus componentes.

Las miniespecificaciones definen un proceso elemental declarado en el DFD. Un proceso es cosiderado elemental o primitivo cuando no puede ya ser subdividido en niveles mas bajos. El método usado para escribir miniespecificaciones utiliza: español estructurado, tablas de decisión, arboles de decisión, etc.

El modelo del sistema tiene diferentes usos en el análisis estructurado:

- Es una herramienta de comunicación. Los analistas y los usuarios tienen fallas en la comunicación, el modelo es esencial para las discusiones y que estas puedan conducir a un entendimiento común.

- Es el marco de referencia para las especificaciones. El modelo declara las piezas que componen el sistema y las partes de esas piezas, hasta esas que ya no pueden ser subdivididas.

- Es el punto de inicio para el diseño. Esto es debido a que

el modelo es el mas elocuente elemento que se originó en los requerimientos y tendrà una gran influencia en el trabajo que se haga en la fase de diseño.

Según Edward Yourdon II. el anàlisis estructurado ayuda a resolver los problemas de la fase de anàlisis ya que ataca el problema del tamaño por particiones, ataca el problema de la comunicación en forma interactiva y con una inversión de los puntos de vista, y ataca el problema del mantenimiento de especificaciones por redundancia limitada.

5.2.1 Caracteristicas del diagrama de flujo de datos

El DFD es gràfico ya que de un vistazo se perciben ràpidamente las funciones principales del sistema.

Es modular pues muestra la partición de un sistema en funciones tan independientes entre si como sea posible, lo cual permite revisar cada función del sistema de una manera aislada.

Enfatiza el flujo de datos ya que muestra solamente el flujo de datos que se transforman a medida que pasan a los procesos desde la entrada a la salida.

Desenfatizar el flujo de datos y no muestra información de contról ni secuencia de acciones en el tiempo.

Es modificable lo que significa que se pueden reconsiderar algunas partes del diagrama de flujo de datos con las cuales no se haya quedado satisfecho y volver a trabajarlas.

No es redundante pues una función debe registrarse sólo una vez para que el sistema, al cual dará origen el diagrama de flujo de datos, sea consistente y de fàcil actualización.

El periodo de diálogo entre el analista y los usuarios debe reducirse y procurar realimentación. También se menciona como parte del anàlisis la inversión de puntos de vista. El análisis estructurado adopta un punto de vista diferente, el diagrama de flujo de datos sigue las trayectorias de los datos por donde el los pasen, ya sean procesos manuales ó automáticos.

5.2.2 Construccion del diagrama de flujo de datos

El diagrama de flujo de datos presenta al sistema en tèrminos de sus piezas componentes con todas las interfases entre los componentes indicados. Los elementos que lo constituyen son:

- Flujo de datos, representados por vectores.

- Procesos, representados por circulos ò rectàngulos redondeados.

- Archivos, representados por lineas rectas ò rectàngulos abiertos.

- Datos fuente ò destino, representados por cajas ò cuadros.

El flujo de datos representa a una interface entre los componentes del DFD. Estos flujos de datos, se encuentran entre procesos y vàn ò vienen de archivos ò de fuentes y destinos.

Las convenciones para asignar un nombre al flujo de datos son:

- Los nombres estàn ligados con guiones y se usan letras mayùsculas.

- No hay dos flujos de datos con el mismo nombre.

- Los nombres no solamente deben representar los datos que se muevan en el flujo de datos, si sabemos mas de ellos pueden aparecer en el nombre.

- La misma linea de flujo de datos puede llegar a dos procesos ò dos salidas pueden juntarse en una sola linea.

- Las lineas que salen ò llegan a archivos no requiere nombre.

El proceso invariablemente muestra alguna cantidad de trabajo realizado sobre los datos.

Un proceso es una transformaciòn del flujo de datos de entrada, a un flujo de datos de salida. Cada proceso, debe tener un nombre.

Las convenciones para referenciar el proceso son:

- Identificación dada por un número cuya función es identificar al proceso.

- Descripción de la función mediante una oración corta que describe al proceso.

- Quién lo realiza indicando el nombre de un departamento, persona, programa, etc. que realiza la función.

Para los propósitos de DFD el archivo es considerado como un depósito temporal de datos. Las convenciones para identificar un archivo son:

- Identificador para facilitar su referencia y se utiliza la letra D seguida de un número.

- Nombre lo mas descriptivo que se pueda.

La fuente ó destino es una persona ó una organización situado fuera del contexto del sistema, es el origen ó el receptor de los datos del sistema. Se representa mediante:

- Identificador escrito con letras minúsculas asociadas a las entidades externas.

- Nombre de la entidad.

- Duplicador que indica que el símbolo se repite en el diagrama.

5.2.3 Guía para la elaboración del diagrama de flujo de datos

Se deben de seguir los siguientes pasos en la elaboración de los diagramas de flujos de datos:

- Identificar las entidades externas involucradas.

- Identificar las entradas y salidas. Se sugiere hacer una lista de las entradas y salidas.

- Identificar las preguntas y los requerimientos de información que puedan presentarse.

- Hacer un diagrama colocando las entidades externas y el flujo de datos que se origina de cada uno de ellos, los

procesos y los almacenes necesarios.

- El primer diagrama puede ser hecho a mano libre.

- Es posible que se requiera al menos 3 variaciones del dibujo inicial.

- Cuando se tiene el primer diagrama, verificar que todas las entradas y salidas se han incluido.

- Tratar de minimizar el número de cruces de lineas de flujo.

- Duplicar entidades externas si es necesario.

- Duplicar archivos si es necesario.

- Validar el diagrama con alguno de los usuarios, mostrarle el diagrama y pasarlo por el y explicarle que es solo una aproximación y si tiene alguna observación que la haga.

- Hacer una explicación de cada uno de los procesos incorporando errores y salidas de excepción.

- Si es posible, construir el diagrama final en una hoja de 36 X 48 pulgadas.

5.3 Aplicacion al caso practico

Utilizando las técnicas ya explicadas anteriormente se puede llegar a nuestro primer DFD del sistema de información de cuentas por pagar mostrada en la figura 5.1.

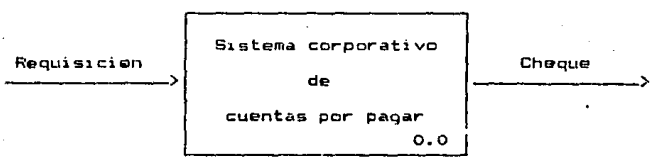


Figura 5.1 Primer DFD del Sistema de Cuentas por Pagar.

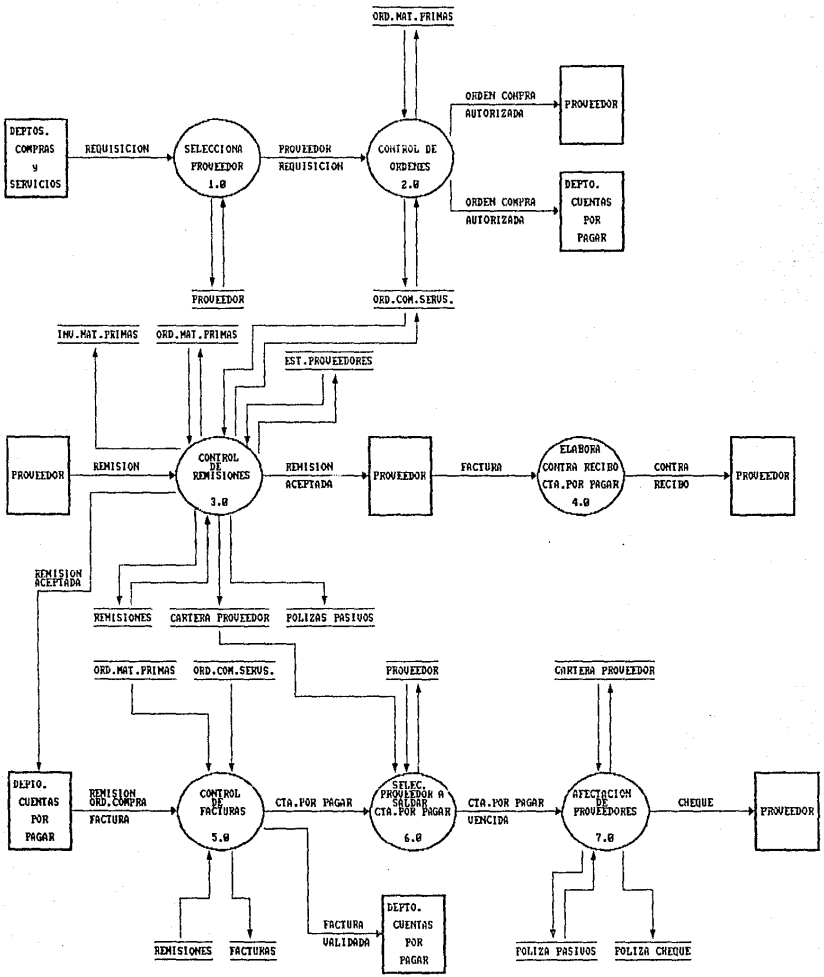


Figura 5.2 Diagrama de Flujo de Datos General del Sistema Corporativo de Cuentas por Pagar.

El siquiente paso a seguir es detallar el DFD anterior indicando el flujo de información, dentro del sistema de cuentas por pagar, llegando al DFD mostrado en la figura 5.2.

Como se puede observar en el DFD de la figura 5.2 estan enumerandos los procesos que requiere este sistema para satisfacer las necesidades de informacion para hacer una cuenta por pagar. A continuación se describirán con mas detalle cada uno de estos procesos que se muestran en las figuras 5.3a, b, c, d, e, f.

El proceso 4.0 de la figura 5.2 no requiere de que sea automatizado por lo cual se decidió que se siguiera desarrollando manualmente, por esta razón no se hizo el detalle de éste.

El proceso 1.2 de la figura 5.3a se seguira haciendo manualmente por que no requiere de que sea automatizado.

Como se puede observar en el detalle de cada uno de los módulos que componen el sistema de cuentas por pagar, cada proceso tiene las mismas entradas y salidas.

Finalmente conjuntando todos estos módulos, se llega al a la figura 5.4 que muestra el diagrama de flujo de datos, que es el que representa el flujo de información que se requiere para este sistema de cuentas por pagar.

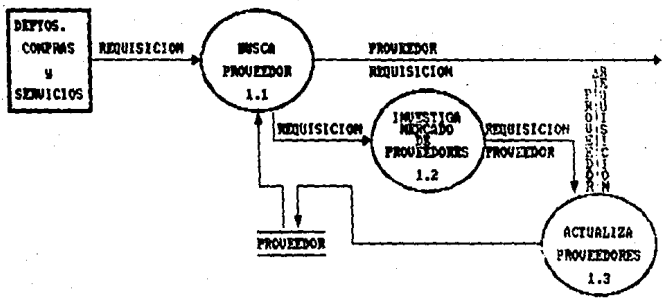


Figura 5.3a Detalle del Proceso 1.0 (SELECCIONA PROVEEDOR).

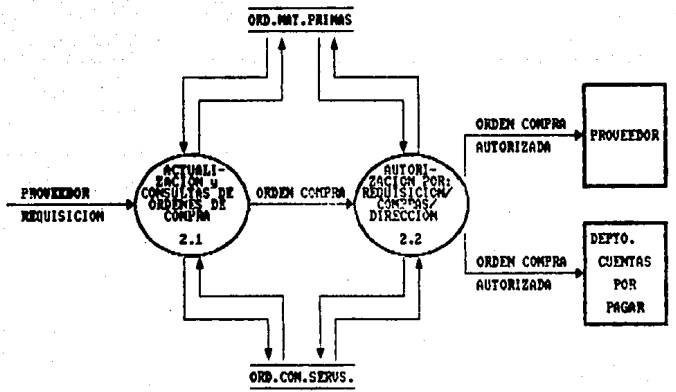


Figura 5.3b Detalle del Proceso 2.0 (CONTROL DE ORDENES).

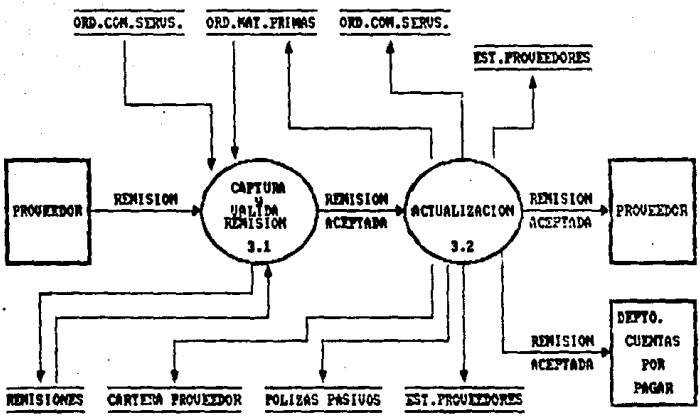


Figura 5.3c Detalle del Proceso 3.0 (CONTROL DE REMISIONES).

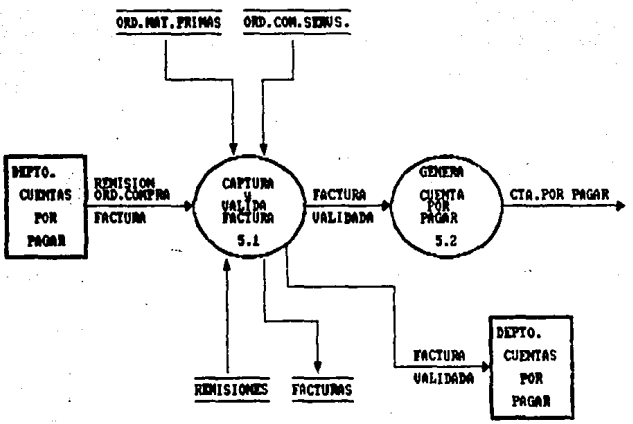


Figura 5.3d Detalle del Proceso 5.0 (CONTROL DE FACTURAS).

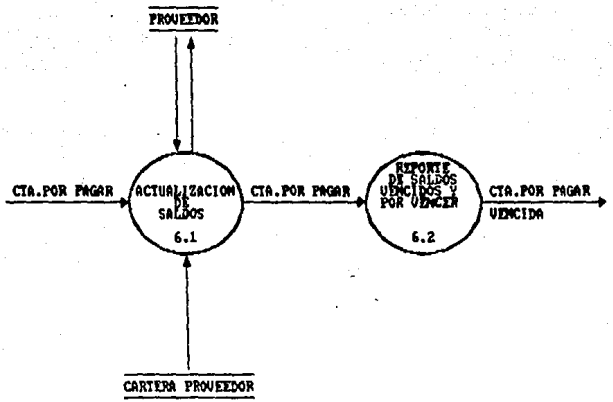


Figura 5.3e Detalle del Proceso 6.0 (SELECCION DE PROVEEDOR A SALDAR CTA.POR PAGAR).

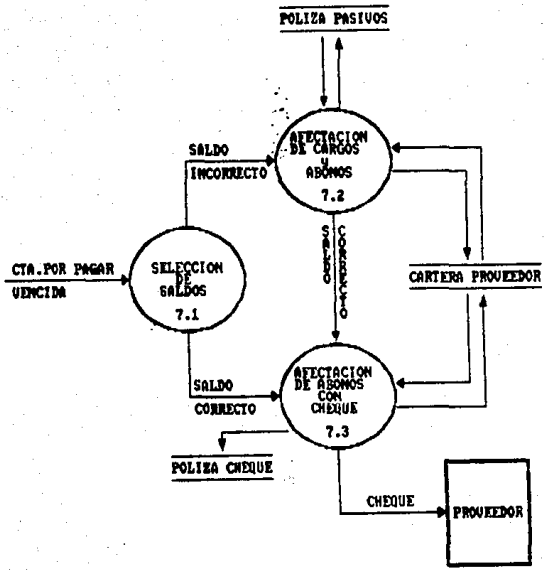


Figura 5.3f Detalle del Proceso 7.0 (AFECTACION DE PROVEEDORES).

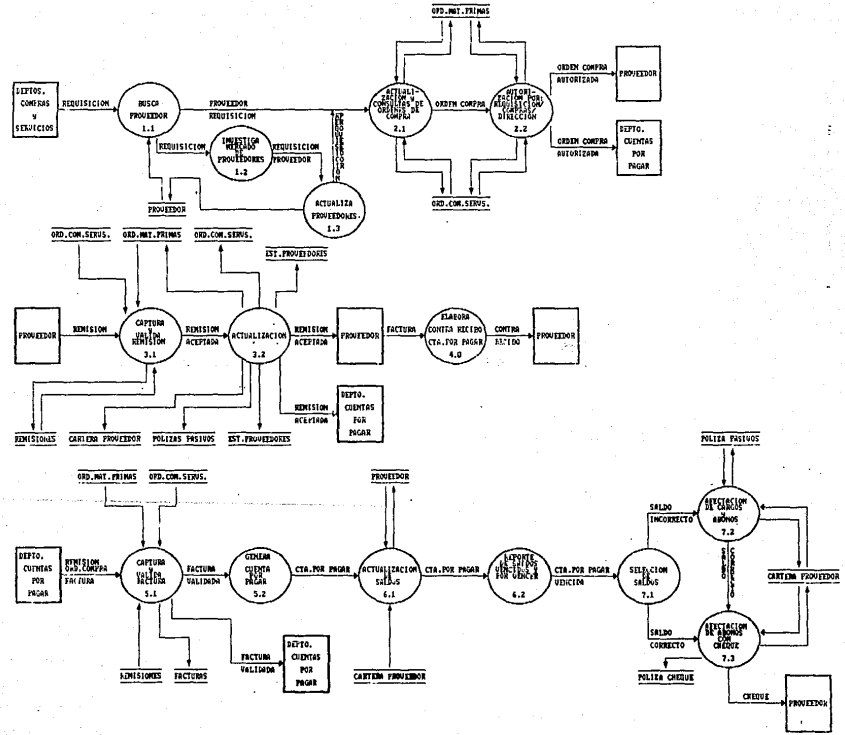


Figura 5.4 Diagrama de Flujo de Datos Detallado del Sistema Corporativo de Cuentas por Pagar.

Capitulo 6 Diseño

6.1 Consideraciones Basicas

Diseño estructurado es la elaboración (utilizando herramientas de modelado de sistemas) de una solución jerárquica del sistema, con los mismos componentes e interrelaciones que el problema que se intenta resolver.

El objetivo del diseño estructurado es hacer sistemas que sean utiles, mantenibles, modificables, flexibles, eficientes, y generales.

Para alcanzar los objetivos del diseño estructurado, se requiere una herramienta de modelado para planear el sistema y alguna manera para controlar la complejidad de sistemas no triviales.

6.1.1 El modelo

Un modelo es simplemente un conjunto ordenado de hipótisis acerca de un sistema complejo.

Los modelos se presentan de varias maneras como son gràficas, ecuaciones, modelos a escala, maquetas, simuladores, etc. Los modelos son importantes en las disciplinas de ingeniería y su selección depende del problema en cuestión.

El modelo debe de tener las caracteristicas de ser; gràfico, divisible, riguroso, capaz de predecir el comportamiento del sistema, que sea una consecuencia natural del análisis estructurado, que sea una entrada natural para la implementación estructurada, que tenga documentación bàsica del sistema, y de que cuente con ayuda para mantener y/o modificar el sistema.

6.1.2 El control de complejidad

La herramienta más útil para el control de complejidad en el diseño de sistemas es la caja negra.

La caja negra es un mecanismo por medio del cual se conoce del sistema cuales son sus entradas, sus salidas, lo que hace, sin saber cómo lo hace.

Cuando los diseNos se hacen con cajas negras pequeNas e independientes èstas son fàcilmente entendidas, probadas, corregidas, mantenidas y modificadas.

El uso de esta herramienta, radica en dividir el sistema en cajas negras conectadas de tal forma que:

- Cada caja negra corresponda a una parte bien definida del problema.

- Cada caja negra sea fàcil de entender.

- Las conecciones entre cajas negras correspondan a conecciones del problema.

- Las conecciones entre cajas negras sean lo màs simple posible, de tal manera que las cajas negras sean independientes entre si.

6.1.3 Las herramientas

El DiseNo Estructurado se apoya en el uso de dos herramientas:

- La carta de estructura.

- El diccionario de datos.

La carta de estructura es la herramienta principal del diseNo estructurado, la cual muestra la particiòn del sistema en modulos y la relaciòn jerarquica que existe entre estos. Tambien muestra los flujos de datos y control entre modulos.

La carta de estructura es utilizada para documantar la estructura jerárquica, los parametros y las interconexiones que existen dentro de un sistema, los elementos que la conforman serán explicados con más detalle mas adelante.

El diccionario de datos nos sirve para definir y registrar los elementos de datos y para especificar los detalles del procesamiento algoritmico se emplean las representaciones lògicas de los procesamientos, tales como las tablas de decisiòn y el idioma estructurado, asi como el conjunto de las interfases declaradas en el DFD. Se define cada una de esas interfases en tèrminos de sus componentes.

El diccionario de datos nos ayudará a documentar cada uno de los elementos del DFD. Para documentar los flujos de datos y los archivos es necesario establecer que son un dato elemental y una estructura de datos.

- Dato elemental es sinónimo de data-item, campo y elemto. Cualquier dato elemental, tomara valores atómicos ó que no es posible su descomposición. El patrón de unos y ceros que lo representa debe ser considerado como un todo.

- Estructura de datos es un agregado de estructuras elementos ó de otro tipo de datos ó de una mezcla de ambos.

Existen otras herramientas complementarias a las antes mencionadas como son las heuristicas de diseño, la morfología del sistema, criterios de transformación, etc., las cuales intervienen en el diseño estructurado; sin embargo la aplicación de estas herramientas complementarias requiere del diccionario de datos y de la carta de estructura.

6.1.4 La metodología

El proceso de diseño se puede dividir en diseño general y diseño detallado.

El diseño general consiste en decidir qué funciones, parámetros y relaciones son requeridas para el sistema.

El diseño detallado se refiere a la forma de implantar.

En terminos muy generales, la metodología de las herramientas de diseño estructurado consiste en la aplicación sistemática de las herramientas de diseño de la siguiente forma:

- Construir el diccionario de datos.

- Aplicando criterios de transformación, hacer la carta de estructura.

- Factorizar la carta de estructura.

- Analizar la carta de estructura utilizando criterios de acoplamiento, cohesión, y heuristicas de diseño.

- Especificar cada modulo.

6.2 Modelado mediante la carta de estructura

La carta de estructura es una representación gráfica del sistema que se utiliza como herramienta para el diseño, implementación, documentación, desarrollo, modificación y mantenimiento del sistema. Es una herramienta no un método.

La carta de estructura es un modelo independiente del tiempo de las relaciones jerárquicas de los modulos de un programa ó sistema; es por esto que no se puede inferir de una carta de estructura, cual es el orden en que se ejecutan los modulos.

6.2.1 Caracteristicas de la carta de estructura

La carta de estructura muestra:

- La partición del sistema, es decir, los módulos de que consta.

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

- Los nombres de módulos y por consiguiente su función.

- El grado de acoplamiento entre módulos.

- Flujo de datos entre módulos.

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

La Carta de Estructura no muestra:

- El número de veces que se llama un módulo.

- La secuencia en que se llaman los módulos.

- Como realiza su función.

- Datos internos al módulo.

6.2.2 Construccion de la carta de estructura

La carta de estructura muestra la partición del sistema en modulos y la relación jerárquica entre estos. También muestra los

flujos de datos y control entre los modulos.

Los elementos de una carta de estructura son:

- Rectàngulo con un nombre inscripto para indicar un màdulo. El nombre indica la funciòn de este màdulo.

- Lineas que indican la liga entre màdulos que son tambièn llamadas a màdulos.

- Flechas que indican el flujo de datos. A esto se le llama comunicaciòn entre màdulos.

- Màdulos predefinidos (Ejem. los subprogramas de biblioteca).

- El nombre del màdulo debe resumir los nombres de sus subordinados ò resumir su funciòn y las funciones de sus subordinados inmediatos.

Un modulo es una secuencia de instrucciones continuas de programa confinadas por variables limitrofes que tienen un identificador.

Los atributos bàsicos de los màdulos son:

- Entrada ò los datos que le son transferidos por quièn lo invoca.

- Salida ò los datos que regresa a quièn lo invoca.

- Funciòn ò lo que hace a las entradas para producir las salidas.

- Mecànica ò como realiza su funciòn, es decir su lògica.

- Datos internos ò su propio espacio de trabajo, es decir las variables locales.

Como regla general, una carta de estructura muestra a su izquierda los màdulos de entrada, al centro los màdulos que procesan la informaciòn y al lado derecho los màdulos de salida, sin que esto condicione el orden en que se puedan utilizar.

6.2.3 Acoplamiento

El acoplamiento es una medida de la interdependencia de un módulo respecto a otro. Entonces los módulos altamente acoplados estan unidos por interconecciones rigidas. Los módulos holgadamente acoplados estan unidos por interconecciones débiles. Los modulos no acoplados son aquellos que no tienen interconecciones.

Los factores que intervienen en el acoplamiento son varios. Referente al tipo de conexión entre modulos se tienen conexiones minimas, conexiones normales, y conexiones patológicas.

La complejidad de la interface se puede aproximar por el número de elementos pasados entre los módulos. Entre mas elementos haya, la inteface será mas compleja.

Según el tipo de flujo de información a lo largo de la conexión, los sistemas con acoplamiento de datos tienen menor acoplamiento que los de acoplamiento de control y estos a su vez son mejores que los de acoplamiento hibrido.

Las conexiones unidas a referencias fijas al momento de ejecución, tienen menor acoplamiento que cuando la unión se efectuá al tiempo de carga ó de compilación ò de codificación.

Entre menor sea el acoplamiemto entre cualesquiera dos módulos, estos serán mas independientes y el diseño será mejor. Existen los siguientes tipos de acoplamiento.

Por datos cuando solo los datos necesarios son comunicados entre módulos.

Por estampado si referencian la misma estructura de datos (no global).

Por control si se comunican usando al menos un elemento de control.

Por area común si comparten una misma àrea global de datos.

Por contenido cuando un módulo altera instrucciones en otro módulo, ó cuando un módulo referencia ó cambia datos contenidos en otro módulo, ó cuando un módulo brinca a otro, ó cuando dos módulos comparten las mismas literales.

6.2.4 Cohesion

La cohesión es una medida de la consistencia de la asociación de los elementos dentro de un módulo.

Es deseable tener módulos fuertes, altamente cohesivos, es decir módulos cuyos elementos estan altamente relacionados.

De esta forma se puede decir que a mayor cohesión de los módulos del sistema, existirá el menor acoplamiento.

La cohesión es una segunda medida de que tan bien dividimos el sistema. Existen los siguientes tipos de cohesión:

- Funcional cuando todos sus elementos contribuyen a una y solo una tarea completa, cada elemento es parte integral y es esencial para la ejecución de la función del módulo.

- Secuencial cuando sus elementos estan involucrados en tareas en las que los datos que salen de un elemento sirven de entrada a otro elemento.

- Comunicacional cuando sus elementos contribuyen a diferentes tareas, pero cada tarea se refiere a los mismos parámetros de entrada y/o salida.

- Procedimiento cuando el control fluye de un elemento al siguiente, pero los datos no necesariamente fluyen de la misma manera.

- Lógica cuando sus elementos aparentan estàr relacionados a tareas de la misma categoría general.

- Coincidental cuando tiene elementos sin relación significativa entre ellos. Usualmente ejecutan tareas diferentes y sin relación para diferentes "jefes".

6.3 Aplicacion al caso practico

Tomando como base el DFD del sistema de cuentas por pagar, que se diseñó en la etapa del anàlisis estructurado, y usando la tècnicas previamente vistas del diseño estructurado se puede llegar a nuestra primera carta de estructura que se muestra en la figura 6.1.

Como se puede observar en la figura 6.1 los módulos del segundo nivel, no estan a detalle ya que para el diseño de esta carta de estructura se tomo el primer DFD que se hizo en la etapa de anàlisis.

El siguiente paso a dar es ir detallando cada uno de los módulos de la carta de estructura anterior, y apoyandose del DFD, se obtiene la carta de estructura que comprende hasta el tercer nivel mostrada en la figura 6.2.

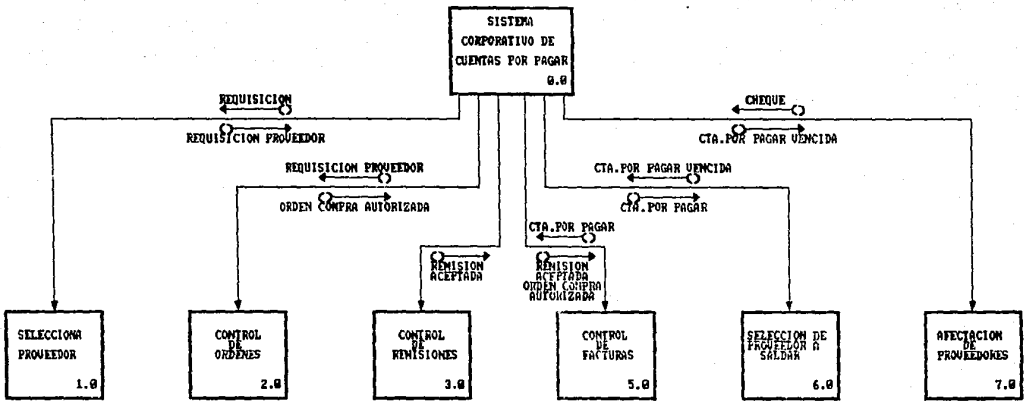


Figura 6.1 Carta de Estructura General del Sistema Corporativo de Cuentas por Pagar.

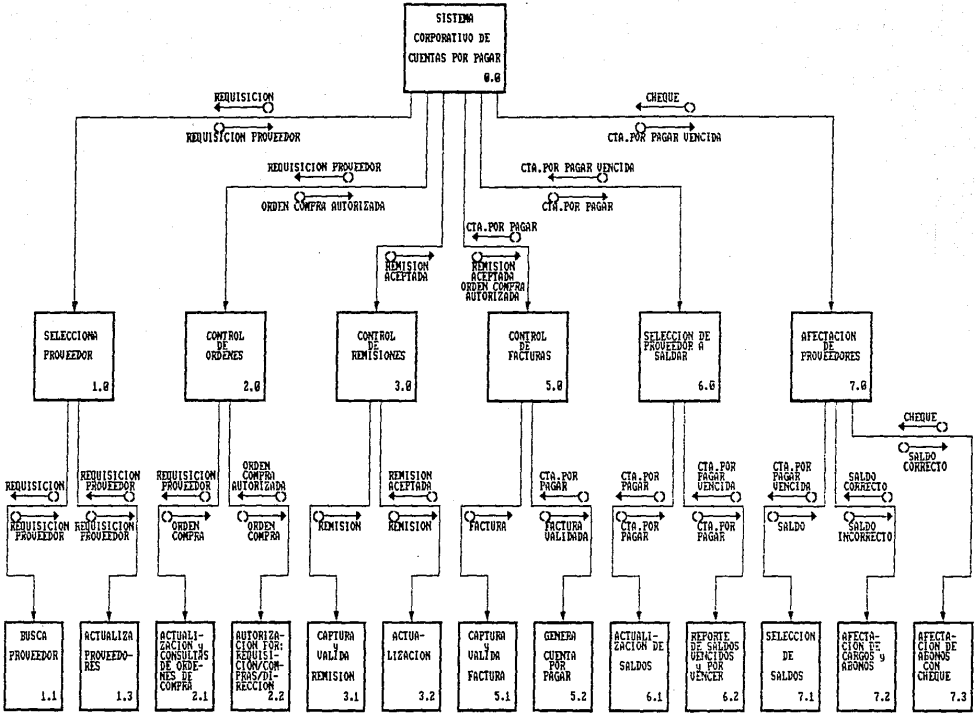


Figura 6.2 Carta de Estructura Detallada del Sistema Corporativo de Cuentas por Pagar.

CAPITULO 7 Instrumentación

El objetivo principal de la instrumentación de un sistema es la generación de un código fuente que sea sencillo en su lectura y comprensión. La producción de un código fuente claro facilita la depuración, prueba y modificación que son actividades que consumen mayor parte de los presupuesto de programación.

La producción del código fuente es posible mejorarla a través de técnicas de codificación estructurada, buen estilo de diseño, documentos de apoyo adecuados. Así mismo, para obtener una programación de alta cálidad es necesario que el equipo de trabajo, cuente con un lider designado, una estructura organizacional bien definida y una definición de responsabilidad a cada uno de los miembros del equipo.

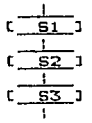
El equipo de instrumentación debe de contar con un conjunto de requisitos de programación bien definidos, así como una buena especificación al diseño estructurado obtenido el la etapa de análisis y diseño y una descripción del diseño detallado. También cada miembro del equipo debera comprender en forma clara los objetivos del trabajo que esta instrumentando.

7.1 Tecnicas de programacion estructurada

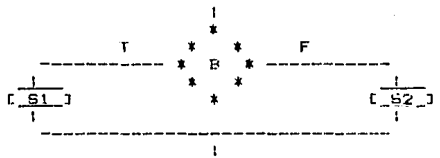
El objetivo principal de la programación o codificación estructurada es linealizar el flujo de control a través de un programa de computadora, de tal modo que la secuencia de ejecución sea la misma que se tiene en el código escrito. La estructura dinámica de un programa a medida que es ejecutado deberá de parecerse a la estructura estàtica del texto escrito. Con esto se mejora la legibilidad del código, por lo que es posible facilitar la comprensión, depuración, prueba, documentación y modificación de los programas y a su vez es más sencillo realizar la verificación formal de los mismos. El flujo de control lineal es posible lograrlo mediante la utilización de formatos de entrada y salida única.

El flujo de control lineal nos indica que todo algoritmo utilizado en programas de computadora deberá usar: secuenciación, selección entre acciones alternativas e interacción, esto se basa principalmente en el teorema que nos dice: **cualquier segmento de programa con entrada y salida unica que tenga las proposiciones en algún camino de la entrada a la salida es posible especificarse usando solamente secuenciación, selección e interacción.**

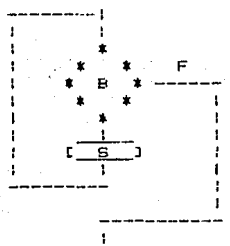
Un ejemplo de conjunto de instrucciones o construcciones con entrada y salida única que especifican al flujo de control en algoritmos es la secuenciación: S1; S2; S3;



Así mismo también podemos contar con la selección: IF B THEN S1 ELSE S2 ;



Interacción: WHILE B DE S;

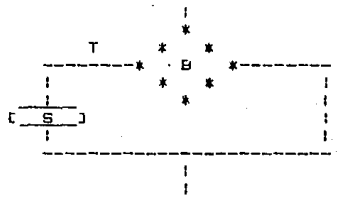


Cualquiera de las estructuras presentadas anteriormente es posible introducirla una dentro de la otra en cualquier orden que

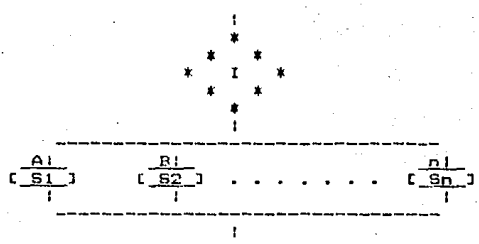
se desee, Cada una de las proposiciones S1 podrá ser una proposición de asignación, un llamado a un procedimiento, un IF-THEN-ELSE o un WHILE-DO. En las dos últimas formas mencionadas es posible tener proposiciones anidadas. Lo más importante en la propiedad de una entrata-una salida es mantener la linealidad del flujo de control, aún con el anidamiento de las construcciones.

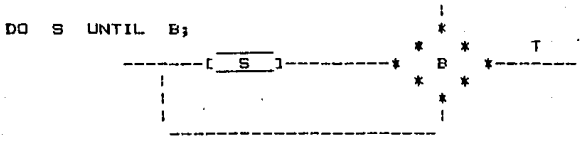
En la práctica se usan otras construcciones de entrada y salida única las cuales se ilustran a continuación:

```
IF  B  THEN  S;
```




```
CASE  I  OF
     A: S1
     B: S2
     .....
     .....
     .....
     .....
     n: Sn
     END
```





Cada una de las construcciones ilustradas anteriormente, se pueden expresar en tèrminos de secuenciaciòn, IF-THEN-ELSE y WHILE-DO anidados.

El conjunto total de construcciones proporcionado por los lenguajes de programaciòn modernos ofrecen mayor conveniencia notacional, mayor legibilidad y en algunos casos, mayor eficiencia.

Una de las desventajas de utilizar construcciones de entrada y salida ùnicas, es la utilizaciòn ineficiente del espacio de memoria y el tiempo de ejecuciòn. Con frecuencia es necesario el uso de variables auxiliares, segmentos de còdigo repetidos y un uso excesivo de llamadas a subprogramas incrementando con esto el costo indirecto para ligado del subprograma, de lo contrario la regla de una entrada y una salida serìa anulada.

El objetivo principal de la codificaciòn estructurada es mejorar la calidad y legibilidad de un programa fuente. La claridad es posible mejorarla en forma considerable, mediante el uso de construcciones de entrada y salida ùnica. Sin embargo hay ocasiones en las que no es posible apegarse a esto, con lo se viola la construcciòn de entrada y salida ùnica. Tres situaciones que suceden generalmente son las salidas mùltiples del ciclo, el manejo de errores y rutinas fijas de validaciòn.

Con frecuencia se tiene varias condiciones para terminar un ciclo. El mejor procedimiento para procesar los ciclos de salidas mùltiples es rediseñar el algoritmo de tal manera que las salidas que se tengan sean ùnicas, si esto no es posible, el mejor procedimiento a seguir es introducir una variable booleana bandera para cada una de las condiciones y probar las banderas en cada interacciòn del ciclo, se presentan tres dificultades cuando se toma este enfoque: primero, la necesidad de variables adicionales; segundo, las pruebas adicionales que se requieren a la salida del ciclo y tercero, el requisito en muchos casos de que el cuerpo del ciclo se termine inmediatamente despuès de que alguna de las condiciones dada se cumpla y no cundo se inicia la siguiente iteraciòn,como ocu-rriria al probar una bandera del ciclo. Con lo anterior podemos advertir que no se debe uno apresurar a crear salidas mùltiples de ciclos. Con frecuencia los algoritmos pueden rediseñarse para eliminar la necesidad de las salidas mùltiples.

Las rutinas de validación de datos implican la modularidad de una estructura de datos y sus rutinas de acceso en un solo módulo. La estructura de datos es manipulada a través de las rutinas de acceso, y otras rutinas utilizan la estructura de datos llamando a las rutinas de acceso apropiadas, de tal modo que la estructura se puede definir por las operaciones que se pueden efectuar sobre ella.

Las rutinas que utilizan la estructura no necesitan conocer los detalles de la representación o manipulación de datos.

La instrucción GO TO permite la transferencia de control incondicional, de tal manera que es violada la condición de entrada y salida única. En algunas ocasiones aunque se utilice la instrucción GO TO y se viole la condición de entrada y salida única, se mantienen la linealidad y localidad del flujo de control.

La instrucción GO TO también se puede usar para simular construcciones de entrada y salida única en lenguajes de programación primitiva. Sin embargo, la instrucción GO TO debe reservarse para situaciones raras o poco comunes donde tiene que romperse la estructura natural del algoritmo. Una buena regla es evitar el uso de saltos para expresar la iteración regular o la ejecución condicional de instrucciones, pues tales saltos destruyen el reflejo de la estructura de la computación en la estructura estática del programa.

Además, la falta de correspondencia entre la estructura estática y dinámica es extremadamente perjudicial a la claridad del programa y hace mucho más difícil la tarea de verificación.

7.2 Estilo en la codificación

En la programación de computadoras el estilo de codificación se manifiesta en las rutinas que usa el programador para expresar una acción o un resultado deseado. En los últimos años, se ha enfocado mucho la atención en el estilo de codificación. Se ha podido observar que al utilizar un buen estilo de codificación es posible superar muchas de las deficiencias de un lenguaje de programación primitivo; así como también un estilo de codificación deficiente puede frustar los propósitos de un excelente lenguaje. El objetivo de un buen estilo de codificación es proporcionar un código fácil de comprender, sencillo y elegante.

No existe un conjunto de reglas que se puedan aplicar en todas las situaciones, sin embargo, se tienen principios generales que son ampliamente aplicables. Algunos de estos se detallan a continuación:

Emplear unas cuantas construcciones normalizadas de control. El anidamiento de instrucciones, la selección entre alternativas y un mecanismo para interacción son suficientes. Si el lenguaje de instrumentación no cuenta con construcciones para codificación estructurada, los programadores que instrumentan el sistema deberán de utilizar patrones de estilo; con esto se lograra uniformidad en el estilo de codificación entre los programadores, con lo que se logrará que los programas sean más fáciles de leer, comprender y modificar.

- Utilice las instrucciones GO TO de manera disciplinada. En el caso de los lenguajes de programación primitivas, la instrucción GO TO puede usarse junto con la proposición IF para lograr el formato y el efecto de las construcciones de codificación estructurada; la proposición GO TO también es de gran utilidad para lograr las salidas múltiples y prematuras de los ciclos, así como, también para transferir el control al código manipulador de errores. En estos casos, el GO TO se usa de una manera con estilo, disciplinado, para lograr el resultado deseado.

- Introdúzcanse nombres de variables y campos normalizados tanto para el programador como para el usuario. El uso de variables normalizadas permitirá una mejor definición del problema que se está resolviendo, además de que es posible facilitar, leer, comprender y modificar los programas con mayor facilidad.

- Proporcionar prólogos normalizados de documentación para cada subprograma y/o unidad de compilación. El prólogo de documentación deberá contener la información acerca del subprograma que no resulta obvio al leer el texto fuente del mismo. El contenido del prólogo variará de acuerdo a la naturaleza del lenguaje de instrumentación. La información dentro del programa o subrutina deberá de estar contendia dentro de lo que es el prólogo y las caracteristicas autodocumentadas del lenguaje de instrumentación.

Examine cuidadosamente las rutinas que tengan menos de 5 o más de 25 proposiciones. Un subprograma consiste de una parte de especificación, un prólogo de documentación, declaraciones, proposiciones ejecutables y, en algunos lenguajes, manejadores de excepciones. La porción ejecutable de un subprograma debe instrumentar una función bien definida que pueda describirse por una frase simple que tengaun verbo activo. Por experiencia en muchas situaciones, las rutinas que tienen entre 5 y 25 proporciones ejecutables satisfacen estos criterios.

Un subprograma que cuenta con mas de 25 proporciones ejecutables probablemente contiene una o más subfunciones bien definidas que puedan ser agrupadas en subrutinas distintas e in-

vocarse cuando sean utilizables. Se ha encontrado que el límite superior de comprensión de una subrutina es de 30 proposiciones ejecutables. También si una subrutina tiene menos de 5 proposiciones normalmente es muy pequeña para proporcionar una función bien definida. Es importante aclarar que la eficiencia no deberá de ser un factor predominante durante la instrumentación inicial de un sistema, pero el costo de ligar una porción significativa de una subrutina de cinco posiciones, y el sentido común debe prevalecer en todas las cosas. Sin embargo existen excepciones obvias a estos principios generales, tales como: rutinas para obtener el valor absoluto o el módulo de un número y que estan formadas con menos de 5 líneas, y que pueden ser muy útiles.

Utilice sangrías, paréntesis, espacios y líneas en blanco, y márgenes alrededor de los bloques de comentarios para aumentar legibilidad. Un formato de presentación agradable mejora grandemente la lectura de un programa fuente. Cuando se usan lenguajes de programación primitivos, la aplicación de un formato al texto fuente a veces se logra junto con el preprocesamiento.

7.3 Normas de codificacion

Estas normas de codificación son utilizados para un estilo de codificación preferido y son vistas a menudo por los programadores como mecanismos para restringir y devaluar la habilidad de éstos para resolver problemas creativos. Este argumento es normalmente esgrimido por los programadores que no entienden el espíritu o la intención de un buen estilo de codificación. La creatividad siempre ocurre dentro de un marco de trabajo disciplinado.

De este modo es deseable que todos los programadores de un proyecto adopten un estilo de codificación similar, de tal manera que se produzca un código de calidad uniformes, más sin embargo, con esto no queremos decir que todos los programadores deben pensar igual, o que deben instrumentar servilmente todos los algoritmos en la misma forma. En realidad el estilo individual de cada programador en un proyecto puede identificarse aún cuando se observe un apego rígido a las normas del estilo de programación.

Es preferible usar el término "Principios generales de la programación" en lugar de "normas de programación".

Una norma de instrumentación deberá especificar cosas como:

- No se utilizarán proposiciones GO TO

- La profundidad de anidamiento de las construcciones de un programa no excederá cinco niveles.

- La longitud de la subrutina no excederà de 20 lineas.

Mientras que el principio general ordena estas especificaciones en la siguiente forma:

- El uso de proposiciones GO TO debe evitarse en circunstancias normales.

- La profundidad de anidamiento de las construcciones de un programa debe ser cinco o menos en circunstancias normales.

- Adaptarse de las circunstancias normales deberà de requerir aprobaciòn del Jefe del Proyecto.

- Cualquier desviación de las circuntancias normales requerira aprobaciòn del lider del proyecto

Algunas personas catalogan a las normas como aquellas caracteristicas que pueden revisarse por medio de la màquina y los principios generales como aquellos aspectos que deben verificarse por medio de seres humanos.

7.4 Documentacion del sistema

La programaciòn por computadora incluye el còdigo fuente de un sistema y todos los documentos de apoyo generados durante el anàlisis, diseño, instrumentaciòn, pruebas y mantenimiento del sistema. La documentaciòn interna incluye pròlogos tanto en subrutinas como en mòdulos principales; asi como notas de cada unidad que proporcionaràn mecanismos para organizar las actividades de trabajo y esfuerzo de documentaciòn de cada programador. En esta secciòn se describirán algunos aspectos de los documentos de apoyo, el uso de notas de cada unidad del programa y algunos principios generales para la documentaciòn interna del còdigo fuente.

Las especificaciones de requisitos, documentos de diseño, planes de prueba, manuales de usuario, instrucciones de instalaciòn y los reportes de mantenimiento son ejemplos de documentos de apoyo. Estos documentos son los productos que resultan del desa-rrollo y mantenimiento sistemàtico de la programaciòn. Las he-rramientas tècnicas y notaciones para generar y dar mantenimiento a estos documentos se analizan en estos parrafos.

De acuerdo al enfoque que se tiene actualmente a los sistemas, el desarrollo de la programaciòn garantiza que los documentos de apoyo se realicen en forma ordenada, y que esos documentos

se encuentren a disposición cuando sean necesarios. Actualmente, en el enfoque adecuado para el desarrollo de la programación, la preparación de documentos de apoyo se difiere hasta que se termine la instrumentación al sistema. Sin embargo, debido a restricciones de tiempo y falta de motivación, los documentos que son generados de esta forma son inadecuados para soportar actividades de prueba, entrenamiento, modificación y mantenimiento. Los documentos de apoyo de calidad inferior a las normas que no se encuentran disponibles cuando son necesarios, indican que se tienen problemas con el proceso empleado en el desarrollo y mantenimiento de la programación, por lo que estos documentos de apoyo deberán elaborarse como un producto natural y paralelo al proceso de desarrollo.

Las necesidades y restricciones del usuario son registradas en la especificación de requisitos; a su vez los requisitos proporcionan el marco de trabajo para el diseño estructurado: el diseño detallado es desarrollado partiendo del diseño estructurado; el código fuente se realiza a partir de los diseños estructurado y detallado. Los planes de prueba, manuales de usuario, programas de entrenamiento, instrucciones de instalación y procedimientos de mantenimiento van evolucionando a través del ciclo de desarrollo. La calidad, cantidad, duración y utilidad de los documentos de apoyo son las principales medidas de la eficiencia de un proyecto de programación.

Un módulo es una unidad de código fuente que es desarrollado y/o mantenida por una presona, esa persana es la responsable del módulo. Este módulo en un sistema bien diseñado, es una subrutina o grupo de subrutinas que cumplen una función bien definida o forman un subsistema. El módulo deberá de ser lo suficientemente pequeño, de tal manera que pueda ser probado en forma completamente aislada por el programador que lo desarrolla o modifica. Las notas de cada módulo son utilizadas por cada programador para poder organizar sus actividades de trabajo y para conservar la documentación de sus módulos.

Un cuaderno de notas de cada programa (también conocido como "carpeta de desarrollo de programas") consta de una portada y varias secciones. La portada es la tabla de contenidos y la hoja de avisos de terminación de los logros asociados al programa. El mantenimiento del cuaderno de notas es responsabilidad del programador asignado al modulo. El cuaderno de notas permanece con la documentación del módulo durante todo su tiempo de vida y pasa de programador a programador a medida que se va transfiriendo la responsabilidad.

Las secciones dentro de un cuaderno de notas del programa corresponden a las diversas fases del ciclo de vida del programa. la portada es utilizada para reportar las fechas proyectada y real de los logros y el aviso de terminación hecho por un supervisor indica la culminación satisfactoria de una fase del ciclo de vida

del programa.

La colección de todas las portadas de los programas dentro de un sistema proporciona un resumen detallado del estado real del proyecto en cualquier momento. Esto sirve para mejorar la visibilidad del proyecto y resaltar las áreas problemáticas.

La sección de requisitos del cuaderno de notas del programa, contiene solo las especificasiones concernientes a esa unidad de programa en particular tanto la versión inicial, como fue convenida con el usuario,asi como las copias de las modificaciones subsecuentes a los requisitos originales se conservan en el cuaderno de notas.

Las secciones del diseño estructurado y detallado del cuaderno de notas, contiene los papeles de trabajo y las especificaciones de diseño finales. Los papeles de trabajo deben organizarse de tal manera que se sepán las diversas alternativas y del porque se eligieron unas y se rechazaron otras. esta información puede ser valiosa para guiar las modificasiones subsecuentes.

El plan de prueba del programa contiene una descripción del enfoque que será usado en la prueba del programa, la cofiguración de prueba, los casos de prueba actual, los resultados esperados para cada caso de prueba y los criterios para determinar que las pruebas han sido terminadas. El plan de pruebas del programa es desarrollado junto con el diseño e instrumentación del programa.

La sección que contiene el código fuente de la unidad tiene un listado de la versión actual de la unidad y listado de las versiones previas más significativas, principalmente en la fase del mantenimiento del ciclo de vida del programa.

La sección de resultados de las pruebas contiene los resultados de las corridas de prueba y un análisis de ellas en terminos de los resultados esperados. de nuevo no es necesario que se conserven todas las corridas de depuración, sino solo las corridas de prueba mas significativas (tanto las que tuvierón éxito como las que no).

El reporte de problemas se llena en el momento en el que el módulo es incorporado a la configuración del sistema.

El reporte de problemas puede describir una situación de error o una modificación deseada. Pasado el tiempo, los reportes de problemas mostraran la historia de los programas.

La documentación interna consiste en un prólogo normalizado para cada programa, los aspectos de autodocumentación del código fuente y los comentarios internos intercalados en la porción ejecutable, del código. Un formato normal para los prólogos,

contendrá el nombre del autor, la fecha de compilación, función(es), realizada(s), algoritmos empleados, autor/fecha/propósito de modificación, parámetros y modos, variables globales, efectos globales, estructuras de datos principales, rutinas que invocan y rutinas invocadas

El uso de prólogos normalizados, construcciones de programación estructurada, buen estilo de codificación y nombres de identificadores significativos para denotar tipos de datos definidos por el usuario, variables, laterales de enumeración, parámetros y subrutinas pueden mejorar ampliamente la legibilidad l código fuente y de esta menera minimizar la necesidad de complementos internos en la porción ejecutable de la subrutina. A continuación se presentan algunos principios básicos para la documentación interna:

- Minímice la necesidad de comentarios muy grandes en el código al usar prólogos normalizazdos, programación estructurada, buen estilo de programación, nombres descriptivas del dominio del problema para tipos de datos definidos por el usuario y variables.

- Ajústense comentarios a los bloques de código que realicen manipulaciones de datos importantes, simulen construcciones de control estructuradas que manejen instrucciones GOTO y realicen manejo de excepciones.

- Usese la terminlogía del dominio del problema en los comentarios.

- Empleense líneas en blanco, delimitadores y sangrado para realizar los comentarios.

- Coloquense los comentarios a la extrema derecha para documentar cambios y revisiones.

- No se manejen comentarios largos y confusos; para aclarar un código oscuro y complejo, reescríbase el código.

- Siempre es necesario asegurarse que el código y los comentarios correspondan uno con el otro, así como con los requisitos y especificaciones de diseño.

7.5 Aplicacion al Caso Practico

Seudocódigo del proceso 0.0 (SISTEMA DE CUENTAS POR PAGAR)

```
PROGRAMA PRINCIPAL

S1: Presenta menú en pantalla
S2: Selecciona opción
En el caso de que la opción sea:

     1:  EJECUTA MODULO DE BUSQUEDA DE PROVEEDOR(1.1)
     2:  EJECUTA MODULO DE ACTUALIZACION DEL PROVEE-
         DOR(1.3)
     3:    EJECUTA MODULO DE ACTUALIZACION DE ORDENES
         DE COMPRA(2.1)
     4:  EJECUTA MODULO DE AUTORIZACION DE ORDENES DE
         COMPRA(2.2)
     5:    EJECUTA MODULO DE CAPTURA Y VALIDACION DE
         REMISIONES(3.1)
     6:  EJECUTA MODULO DE ACTUALIZACION DE REMISIO-
         NES(3.2)
     7:  EJECUTA MODULO DE CAPTURA Y VALIDACION DE
         FACTURAS(5.1)
     8:    EJECUTA MODULO DE GENERACION DE CTAS. POR
         PAGAR(5.2)
     9:  EJECUTA MODULO DE ACTUALIZACION DE SALDOS(6.1)
    10:  EJECUTA MODULO PARA LA OBTENCION DEL RE-
         PORTE DE SALDOS(6.2)
    11:  EJECUTA    MODULO DE SELECCION DE SALDOS A
         PAGAR(7.1)
    13:  EJECUTA MODULO DE AFECTACION DE CARGOS Y
         ABONOS(7.2)
    14:    EJECUTA MODULO DE AFECTACION DE ABONOS CON
         CHEQUE(7.3)

END
```

Seudocódigo del proceso 1.1 (BUSCA PROVEEDOR)

MODULO DE CONSULTA DEL MAESTRO DE PROVEEDORES

Programa principal

```
WHILE READ ( ARCHIVO DE PANTALLA )
     CASE IND OF
          1:   DESPLIEGA PANTALLA DE SELECCION  DE  PRO-
               VEEDOR
          2:   EJECUTA AA00
          3:   DESPLIEGA PANTALLA DE DATOS GRALES.
          5:   EJECUTA AB00
          7:   EJECUTA AC00
          8:   DESPLIEGA PANTALLA DE SALDO
     END-CASE
END-WHILE
```

Subrutina de validacion de compania/proveedor y opcion de consulta

```
AA00 BEGIN-SUBRUTINA
     S1: VALIDA CAMPOS DE SELECCION
     IF ERROR THEN
          S1:  DESPLIEGA PANTALLA DE SELECCION Y  MENSAJE  DE
               ERROR
     ELSE
          S2:  ARMA PANTALLA DE DATOS GRALES.
          S3:  ARMA PANTALLA DE SELECCION DE ACUMULADOS
          S4:  ARMA PANTALLA DE SALDOS
     END-IF
     CASE OPC OF
          1:   DESPLIEGA PANTALLA DE DATOS GENERALES
          2:     DESPLIEGA PANTALLA DE SOLICITUD DE OPCION DE
               ACUMULADOS
          3:   DESPLIEGA PANTALLA DE SALDOS
     END-CASE
END-SUBRUTINA
```

Subrutina de validacion de seleccion de acumulados

```
AB00 BEGIN-SUBRUTINA
     S1: VALIDA CAMPOS DE SELECCION DE ACUMULADOS
```

```
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA  DE SELECCION Y MENSAJE DE
               ERROR
          ELSE
               CASE OPC OF
                    1:   ARMA Y DESPLIEGA PANTALLA DE PIEZAS
                    2:   ARMA Y DESPLIEGA PANTALLA DE KILOS
                    3:   ARMA Y DESPLIEGA PANTALLA DE LITROS
                    4:   ARMA Y DESPLIEGA PANTALLA DE VOLUMEN
                    5:   DESPLIEGA PANTALLA DE  SOLICITUD  DE
                         ANOS PARA IMPORTES
               END-CASE
          END-IF
     END_IF
END-SUBRUTINA
```

Subrutina de validacion y seleccion de anos a consultar

```
ACOO BEGIN-SUBRUTINA
     S1: VALIDA ANOS A CONSULTAR
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA  DE SELECCION Y MENSAJE DE
               ERROR
     ELSE
          S3:  ARMA PANTALLA DE IMPORTES PAGADOS
     END-IF
END-SUBRUTINA
```

MODULO DE IMPRESION DE CATALOGO DE PROVEEDORES

Programa Principal

```
     WHILE READ ( PROVEEDORES )
          S1:  MUEVE  CAMPOS  DE DATOS GRALES DEL PROVEEDOR A
               CAMPOS DE IMPRESION
          S2:  IMPRIME REGISTROS
          S3:  CONT1 <--- CONT1 + 1
          S4:  CONT2 <--- CONT2 + 1
          IF WCIA NE CIA THEN
               S5: IMPRIME TOTALES POR COMPANIA
          END-IF
          IF LR THEN
               S6: IMPRIME TOTALES FINALES
          END-IF
     END-WHILE
```

udocódigo del proceso 1.3 (Actualización del proveedor)

```
ODULO DE MANTENIMIENTO AL ARCHIVO MAESTRO DE PROVEEDORES

   Programa principal

        WHILE READ ( ARCHIVO PANTALLA )
             CASE IND OF
                  1:     EJECUTA AA00
                  2:     EJECUTA AB00
                  3:     EJECUTA AC00
                  4:     EJECUTA AD00
                  ELSE:  EJECUTA AH00
             END-CASE
        END-WHILE


   Subrutina de validacion de opcion de actualizar


   AA00 BEGIN-SUBRUTINA
        S1: VALIDA OPCION A TRABAJAR
        IF ERROR THEN
             S2: DESPLIEGA PANTALLA DE OPCION MENSAJE DE ERROR
        END-IF
   END-SUBRUTINA


   Subrutina para el proceso de altas


   AB00 BEGIN-SUBRUTINA
        S1: VALIDA DATOS GENERALES DEL PROVEEDOR
        IF ERROR THEN
             S2:  DESPLIEGA  PANTALLA  DE DATOS GENERALES Y MEN-
                  SAJE DE ERROR
        ELSE
             S3:    MUEVE  CAMPOS DE PANTALLA A CAMPOS DE SALIDA
                  DEL ARCHIVO
             S4:  INICIALIZA ESTADISTICAS
             S5:  ADICIONA REGISTRO AL ARCHIVO DE PROVEEDORS
        END-IF
   END-SUBRUTINA


   Subrutina de Seleccion de Proveedor para cambios


   AC00 BEGIN-SUBRUTINA
        S1: LEE PROVEEDOR SELECCIONADO
        IF ERROR  THEN
```

```
          S2:  DESPLIEGA  PANTALLA DE SELECCION  Y MENSAJE DE
               ERROR
     ELSE
          S3:  ARMA PANTALLA DE DATOS GENERALES DEL PROVEEDOR
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de cambios

```
AD00 BEGIN-SUBRUTINA
     S1: VALIDA DATOS GRALES. DEL PROVEEDOR
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE DATOS GENERALES Y MENSA-
               JES DE ERROR
     ELSE
          S3:  MUEVE  CAMPOS  DE PANTALLA DE CAMBIOS A CAMPOS
               PARA ACTUALIZAR EL ARCHIVO
          S4:  ACTUALIZA EL PROVEEDOR SELECCIONADO
          S5:  EJECUTA AX00
     END-IF
END-SUBRUTINA
```

Subrutina de impresion de Prov. actualizados o adicionados

```
AH00 BEGIN-SUBRUTINA
     DO I=1 UNTIL 20
          S1:  LEE PROVEEDOR DEL ARREGLO(I)
          S2:  LEE PROVEEDOR DEL ARCHIVO MAESTRO
          S3:  MUEVE DATOS GENERALES DEL PROVEEDOR  A  CAMPOS
               DE SALIDA
          S4:  IMPRIME  REPORTE  DE  ACTUALIZACION ADICION AL
               ARCHIVO
     END-DO
END-SUBRUTINA
```

Carga el numero de proveedor actualizado o adicionado al arreglo de proveedores para impresión

```
AX00 BEGIN-SUBRUTINA
     S1:  ADICIONA UNO AL INDICE DEL ARREGLO
     S2:  MUEVE  NUMERO  DE  PROVEEDOR  A  LA  LOCALIDAD  DEL
          ARREGLO (I)
```

```
END-SUBRUTINA
```

Seudocódigo del proceso 2.1 (ACTUALIZACION Y CONSULTAS DE ORDENES DE COMPRA)

MODULO DE ALTAS A ORDENES DE COMPRA DE MATERIA PRIMA

Programa principal

```
WHILE READ ( ARCHIVO DE PANTALLA )
     CASE IND OF
          1:   DESPLIEGA PANTALLA DE SELECCION DE COM-
               PANIA A TRABAJAS
          2:   EJECUTA AA00
          3:   EJECUTA AB00
          4:   EJECUTA AC00
          5:   EJECUTA AD00
          6:   EJECUTA AE00
          KJ:  EJECUTA AF00
          KI:  EJECUTA YA00
     END-CASE
END-WHILE
```

Subrutina de validacion de seleccion de compañia

```
AA00 BEGIN-SUBRUTINA
     S1: VALIDA COMPANIA
     IF ERROR THEN
          S1:  DESPLIEGA  PANTALLA  DE SELECCION Y MENSAJE DE
               ERROR
     ELSE
          S2: DESPLIEGA PANTALLA DE DE SELECCION DE PROVEEDOR
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de proveedor

```
AB00 BEGIN-SUBRUTINA
     S1: LEE PROVEEDOR
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE SELECCION DE PROVEEDOR Y
               MENSAJE DE ERROR
     ELSE
          S3:  ARMA Y DESPLIEGA PANTALLA DE DATOS GENERALES
     END-IF
```

```
END-SUBRUTINA
```

Subrutina de validacion de datos generales

```
AC00 BEGIN-SUBRUTINA
     S1: VALIDA DATOS GENERALES DE LA O.C.
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE DATOS GENERALES Y MENSA-
               JE DE ERROR
     ELSE
          S3:  LEE PRODUCTOS
          S4:  ARMA  Y DESPLIEGA PANTALLA DE PRODUCTOS A TRA-
               BAJAR CON LOS PRECIOS Y CANTIDAD
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de precios y cantidad

```
AE00 BEGIN-SUBRUTINA
     S1: VALIDA PRECIOS Y CANTIDAD
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA  DE  SELECCION DE CANTIDAD
               PRECIO Y MENSAJE DE ERROR
     ELSE
          S3:  ARMA Y DESPLIEGA PANTALLA  DE  PRODUCTOS  PRE-
               CIO/CANTIDAD
     END-IF
END-SUBRUTINA
```

Subrutina de creacion de la orden de compra

```
AF00 BEGIN-SUBRUTINA
     S1: ARMA REGISTRO HEADER
     S2: GRABA REGISTRO HEADER
     DO I=1 UNTIL 200
          IF PRECIO=0 THEN
               S3: I=I+1
          ELSE
               S4: ARMA REGISTRO PARTIDA
               S5: GRABA PARTIDA
               S6: I=I+1
          END-IF
     END-DO
     S7: DESPLIEGA PANTALLA DE SELECCION DE CIA.
     S8: IMPRIME FORMATO DE VALIDACION DE CAPTURA
END-SUBRUTINA
```

MODULO DE CANCELACION DE ORDENES DE COMPRA DE MATERIA PRIMA

```
Programa Principal

     WHILE READ ( ARCHIVO DE PANTALLA ) DO
          CASE IND OF
               1: DESPLIEGA PANTALLA DE SELECCION DE O.C.
               2: EJECUTA AA00
               3: EJECUTA AB00
               4: EJECUTA AC00
               KL: EJECUTA YA00
          END-CASE
     END-WHILE


Subrutina de validacion de seleccion de la O.C.


AA00 BEGIN-SUBRUTINA
     S1: LEE ORDEN DE COMPRA
     IF ERROR THEN
          S2: DESPLIEGA PANTALLA DE SELECCION Y  MENSAJE  DE
              ERROR
     ELSE
          S3: ARMA  Y DESPLIEGA PANTALLA DE  DATOS GENERALES
              Y TIPO DE CANCELACION
     END-IF
END-SUBRUTINA


Subrutina de validacion de tipo de cancelacion


AB00 BEGIN-SUBRUTINA
     S1: VALIDA TIPO DE CANCELACION
     IF ERROR THEN
          S2: DESPLIEGA  PANTALLA  DE DATOS GRALES Y TIPO DE
              CANCELACION Y MENSAJE DE ERROR
     ELSE
          IF OPCION=NO THEN
               S3: ARMA PANTALLA DE PARTIDA A  CANCELAR
          ELSE
               S4: ACTUALIZA STATUS DE LA O.C.
          END-IF
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de cancelacion parcial

```
ACOO BEGIN-SUBRUTINA
     S1: LEE PARIDA DE LA ORDEN DE COMPRA
     IF IND=KK THEN
          S2:  TERMINA
     ELSE
          S3:  ACTUALIZA  STATUS DE CANCELACION EN LA PARTIDA
               DE LA O.C.
     END-IF
END-SUBRUTINA
```

Subrutina de impresion de la cancelacion de la O.C.

```
AYOO BEGIN-SUBRUTINA
     S1: LEE REGISTRO HEADER DEL LA O.C.
     S2: IMPRIME HEADER DE LA O.C.
     DO I=1 UNTIL NPA
          S3: IMPRIME PARTIDAS DE LA O.C.
          S4: RECALCULA IMPORTES TOTALES
          S5: I=I+1
     END-DO
     S6: LEE REGISTRO HEADER DE LA O.C.
     S7: ACTUALIZA IMPORTES TOTALES DEL HEADES
END-SUBRUTINA
```

MODULO DE CAMBIOS DE ORDENES DE COMPRA DE MATERIA PRIMA

Programa principal

```
     WHILE READ (ARCHIVO DE PANTALLA) DO
          CASE IND OF
               1:  DESPLIEGA PANTALLA DE SELECCION DE O.C.
               2:  EJECUTA AAOO
               3:  EJECUTA ABOO
               4:  EJECUTA ACOO
               5:  EJECUTA ADOO
               KL: EJECUTA YAOO
          END-CASE
     END-WHILE
```

Subrutina valida campos de solicitud de cambio

```
AA00 BEGIN-SUBRUTINA
     S1: LEE O.C.
     S2: VALIDA DATOS DE LA O.C.
     IF ERROR THEN
          S3:  DESPLIEGA PANTALLA DE SELECCION DE O.C. Y MEN-
               SAJE DE ERROR
     ELSE
          S4:  ARMA PANTALLA DE DATOS GENERALES
          S5:  DESPLIEGA PANTALLA DE DATOS GRALES.
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de datos grales. de la O.C.

```
AB00 BEGIN-SUBRUTINA
     S1: VALIDA DATOS GENERALES
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE DATOS GRALES.
     ELSE
          S3:  CARGA  NO.  DE O.C.  Y PROVEEDOR EN ARREGLO DE
               O.C. CAMBIADAS
          S4:  ACTUALIAZA REGISTRO HEADER DE LA O.C.
          S5:  ARMA PANTALLA DE SOLICITUD DE CAMBIOS  EN PAR-
               TIDAS, SOLO CUANDO ESTAN EN  STATUS DE CAPTURA
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de opcion de actualizacion

```
AC00 BEGIN-SUBRUTINA
     IF RESP=SI THEN
          S1: LEE ORDEN DE COMPRA
          S2: VALIDA STATUS DE ORDEN DE COMPRA
          IF ERROR THEN
               S3:  DESPLIEGA  PANTALLA  DE  SOLICITUD DE AC-
                    TUALIZACION Y MENSAJE DE  ERROR
          ELSE
               S4:  ARMA PANTALLA PARA DAR DE ALTA PARTIDA
               S5:  DESPLIEGA PANTALLA DE ALTA DE PARTIDA
          END-IF
     END-IF
     IF RESP=NO THEN
          S6: LEE ORDEN DE COMPRA
          S7: VALIDA ORDEN DE COMPRA
          IF ERROR THEN
```

```
            S8:  DESPLIEGA PANTALLA DE SOLICITUD DE CAMBIO
                 Y MENSAJE DE ERROR
        ELSE
            S9:  ARMA PANTALLA DE CAMBIOS A PARTIDAS
            S10: DESPLIEGA PANTALLA DE CAMBIOS A PARTIDAS
        END-IF
    END-IF
END-SUBRUTINA
```

Subrutina de validacion de la partida a adicionar modificar

```
ADOO BEGIN-SUBRUTINA
     S1: LEE PRODUCTO
     S2: VALIDA DATOS DE PARTIDA
     IF ERROR THEN
          S3:  DESPLIEGA  PANTALLA  DE PARTIDA Y  MENSAJE  DE
               ERROR
     ELSE
          S4:  ARMA DATOS DEL REGISTRO A ADICIONAR  Y/O MODI-
               FICAR
          S5:  OBTIENE IMPORTES DE LA ORDEN DE COMPRA
          S6:  LEE REGISTRO HEADER PARA ACTUALIZAR  NUMERO DE
               PARTIDAS DE ORDEN DE COMPRA
          S7:  ACTUALIZA REGISTRO HEADER CON EL NUEVO  NUMERO
               DE PARTIDAS
          S8:  ACTUALIZA ARCHIVO DE  O.C.  CON  PARTIDA  ADI-
               CIONADA Y/O MODIFICADA
          S9:    INCREMENTA   EN UNO EL INDICE DEL ARREGLO DE
               PEDIDOS ACTUALIZADOS
          S10: CARGA  EL  NUMERO DE O.C. Y  PROVEEDOR  EN  EL
               ARREGLO DE O.C. ACTUALIZADOS
     END-IF
END-SUBRUTINA
```

Subrutina de impresion de la O.C. de materia prima con datos actualizados y/o adicionados

```
YAOO BEGIN-SUBRUTINA
     DO INDICE = 1 UNTIL 200
          S1: EJECUTA YAAOO
          S2: EJECUTA YABOO
     END-DO
END-SUBRUTINA
```

Subrutina de actualizacion de datos generales en partidas y recalculo de importes originales

```
YAAOO BEGIN-SUBRUTINA
      S1: LEE DATOS GENERALES DEL HEADER DE LA O.C.
      DO  I=1 UNTIL NUMERO DE PARTIDAS
            S2: LEE PARTIDA ORDEN DE COMPRA
            S3: OBTIENE IMPORTE BRUTO DE LA O.C.
            S4: ACTUALIZA PARTIDA
            S5: ADICIONA IMPORTES A IMPORTES TOTALES
      END-DO
      S6: LEE REGISTRO HEADER PARA ACUTALIZAR IMPORTES
      S7: ACTUALIZA IMPORTES DE REGISTRO HEADER
END-SUBRUTINA
```

Subrutina de impresion de ordenes de compra

```
YABOO BEGIN-SUBRUTINA
      S1: LEE NUMERO DE PARTIDAS DE REGISTRO HEADER
      S2: IMPRIME DATOS GENERALES DE O.C.
      DO I=1 UNTIL NPA
            S3: LEE DESCRIPCION DEL PRODUCTO
            S4: IMPRIME PARTIDA DE LA O.C.
      END-DO
END-SUBRUTINA
```

MODULO DE CONSULTA DE ORDENES DE COMPRA DE MATERIA PRIMA

Programa principal

```
      WHILE READ ( ARCHIVO DE PANTALLA )
            CASE IND OF
                  1:  DESPLIEGA PANTALLA DE SELECCION
                  2:  EJECUTA AAOO
                  3:  EJECUTA ABOO
                  4:  EJECUTA ABOO
                  KL: FIN DEL PROGRAMA
            END-CASE
      END-WHILE
```

Subrutina de validacion de la orden de compra

```
AA00 BEGIN-SUBRUTINA
     S1: LEE ORDEN DE COMPRA
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA  DE SELECCION DE DE O.C. Y
               MENSAJE DE ERROR
     ELSE
          S3:  ARMA PANTALLA DE DATOS GENERALES
     END-IF
END-SUBRUTINA
```

Subrutina de consulta de partidas

```
AB00 BEGIN-SUBRUTINA
     DO I=1 UNTIL NO. DE PARTIDAS
          S1: LEE PARTIDAS
          S2: ARMA PANTALLA DE CONSULTA DE PARTIDAS
          S3: DESPLIEGA PANTALLA DE PARTIDAS
     END-DO
END-SUBRUTINA
```

MODULO DE CAPTURA DE ORDENES DE SERVICIO Y COMPRA DE MATERIALES NO PRODUCTIVOS

Programa principal

```
     WHILE READ ( ARCHIVO DE PANTALLA ) DO
          CASE IND OF
               1: EJECUTA AA00
               2: EJECUTA AB00
               3: EJECUTA AC00
               4: EJECUTA AD00
               5: EJECUTA AE00
               6: EJECUTA AC00
               7: EJECUTA AD00
               8: EJECUTA AE00
               9: EJECUTA AC00
               10: EJECUTA AE00
          END-CASE
     END-WHILE
```

Subrutina de inicializacion

```
AA00 BEGIN-SUBRUTINA
     S1:    INICIALIZA  VARIABLES DE PANTALLA DE SOLICITUD DE
          TIPO DE O.S.P.M.
     S2:  DESPLIEGA PANTALLA DE SOLICITUD DE O.S.P.M.
END-SUBRUTINA
```

Subrutina de validacion de tipo de orden de servicio, pedido de materiales no productivos o contratos

```
AB00 BEGIN-SUBRUTINA
     S1: VALIDA TIPO DE O.S.P.M.C.
     IF ERROR THEN
          S2:    DESPLIEGA PANTALLA DE SOLICITUD Y MENSAJE DE
               ERROR
     ELSE
          CASE TIPO OF
               1: EJECUTA AB10
               2: EJECUTA AB20
               3: EJECUTA AB30
          END-CASE
     END-IF
END-SUBRUTINA
```

Subrutina pantalla de compra de materiales no productivos

```
AB10 BEGIN-SUBRUTINA
     S1:  ARMA PANTALLA DE  DATOS GENERALES  DE ORDEN DE COM-
          PRA DE MAT. NO PRODUC
     S2:  DESPLIEGA PANTALLA DE DATOS GENERALES
END-SUBRUTINA
```

Subrutina que inicializa la captura de ordenes de servicio

```
AB20 BEGIN-SUBRUTINA
     S1: ARMA PANTALLA DE DATOS GRALES. DE ORDEN  DE TRABAJO
     S2: DESPLIEGA PANTALLA DE DATOS GRALES
END-SUBRUTINA
```

Subrutina de inicializa captura de contratos

```
AB3O BEGIN-SUBRUTINA
     S1: ARMA PANTALLA DE DATOS GRALES. DE CONTRATOS
     S2: DESPLIEGA PANTALLA UNICA DE CAPTURA DE CONTRATOS
END-SUBRUTINA
```

Subrutina de datos generales de pedidos de materiales no productivos

```
ACOO BEGIN-SUBRUTINA
     S1: VALIDA DATOS GENERALES DE O.S.M.N.P.C
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA  DATOS GRALES Y MENSAJE DE
               ERROR
     ELSE
          S3:  LEE ARCHIVO DE OBSERVACIONES
          S4:  ARMA PANTALLA DE OBSERVACIONES
          S5:  DESPLIEGA PANTALLA DE OBSERVACIONES
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de observaciones

```
ADOO BEGIN-SUBRUTINA
     IF TIPO NE 3 THEN
          S1:  ARMA PANTALLA DE CAPTURA DE PARTIDAS
          S2:  DESPLIEGA PANTALLA DE CAPTURA DE  PARTIDAS
     ELSE
          S3:  GRABA REGISTRO DE CONTRATO
          S4:  DESPLIEGA  PANTALLA DE SOLICITUD TIPO DE ORDEN
               DE SERVICIO
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de partidas de O.S. o M.N.P

```
AEOO BEGIN-SUBRUTINA
     IF IND=KK THEN
          S1: EJECUTA AE1O
     END-IF
```

```
     DO I=1 UNTIL 6
          IF CAMPOS NE '          ' THEN
               I=I+1
          ELSE
               S2: VALIDA CAMPOS DE PARTIDAS
               IF ERROR THEN
                    S3:    DESPLIEGA  PANTALLA  DE CAPTURA DE
                           PARTIDAS Y MENSAJE DE ERROR
               ELSE
                    S4: MUEVE CAMPOS DE PANTALLA A   ARREGLOS
                        DE O.S.
                    S5:  I=I+1
               END-IF
          END-IF
     END-DO
     S6: NPA=NPA+6
     S7: ARMA PANTALLA DE CAPTURA DE PARTIDAS
     S8: DESPLIEGA PANTALLA DE CAPTURA DE PARTIDAS
END-SUBRUTINA


Subrutina  que graba header y partidas de orden de servicio y
           pedidos  de  materiales  no  productivos


AE10 BEGIN-SUBRUTINA
     IF ARREGLOS = '      ' THEN
          S1:  DESPLIEGA  PANTALLA  DE  SOLICITUD DE TIPOS DE
               SERVICIOS
     ELSE
          S2:  ARMA REGISTRO HEADER
          S3:  GRABA REGISTRO HEADER
          S4:  IMPRIME REGISTRO HEADER
          DO I=1 UNTIL 100
               IF ARREGLOS = '       ' THEN
                    S5: I=I+1
               ELSE
                    S6: ARMA REGISTRO DE PARTIDA
                    S7: GRABA REGISTRO HEADER
                    S8: IMPRIME REGISTRO HEADER
                    S9: I=I+1
               END-IF
          END-DO
     END-IF
END-SUBRUTINA
```

MODULO DE BAJAS Y/O CANCELACIONES DE ORDENES DE SERVICIOS Y/O PEDIDOS DE MATERIALES NO PRODUCTIVOS

Programa principal

```
WHILE READ ( ARCHIVO DE PANTALLA ) DO
    CASE IND OF
        1: EJECUTA AA00
        2: EJECUTA AB00
        3: EJECUTA AC00
        4: EJECUTA AD00
        6: EJECUTA AB00
        7: EJECUTA AC00
        8: EJECUTA AD00
        9: EJECUTA AC00
        10: EJECUTA AD00
    END-CASE
END-WHILE
```

Subrutina que inicializa pantallas

```
AA00 BEGIN-SUBRUTINA
     S1: BLANQUEA CAMPOS DE PANTALLA DE SOLICITUD DE BAJAS
     S2: DESPLIEGA PANTALLA DE SOLICITUD DE BAJAS
END-SUBRUTINA
```

Subrutina que valida la solicitud de baja

```
AB00 BEGIN-SUBRUTINA
     S1: VALIDA ORDEN SOLICITADA
     IF ERROR THEN
          S2: DESPLIEGA PANTALLA DE SOLICITUD DE BAJA Y MEN-
              SAJE DE ERROR
     ELSE
          CASE OPCION THEN
              1:  DESPLIEGA PANTALLA DE BAJAS ORDENES DE
                  COMPRA DE MATERIALES NO PRODUCTIVOS
              2:  DESPLIEGA PANTALLA DE BAJAS A ORDENES DE
                  SERVICIO
              3:  DESPLIEGA PANTALLA DE BAJAS A CONTRATOS
          END-CASE
     END-IF
END-SUBRUTINA
```

```
Subrutina de validacion de bajas a ordenes de servicio, con-
          tratos y materiales no productivos


ACOO BEGIN-SUBRUTINA
     IF IND=KK THEN
          S1:  DESPLIEGA PANTALLA DE SOLICITUD DE SERVICIO
     END-IF
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE SOLICITUD DE BAJA Y MEN-
               SAJE DE ERROR
     ELSE
          S3: ARMÀ PANTALLA DE RESPUESTA DE TIPO DE BAJA
          S4: DESPLIEGA PANTALLA DE RESPUESTA DE TIPO DE BAJA
     END-IF
END-SUBRUTINA


Subrutina que procesa la respuesta del tipo de baja que se
                          realiza


ADOO BEGIN-SUBRUTINA
     IF ERROR THEN
          S1:  DESPLIEGA PANTALLA DE TIPO DE BAJA Y MENSAJE
               DE ERROR
     ELSE
          IF RESP=NO THEN
               S2: DESPLIEGA PANTALLA DE TIPO DE BAJA
          ELSE
               IF OPCION=1   OR   OPCION=2    THEN
                    S3:  ARMA PANTALLA DE PARTIDA A ELIMINAR
                    S4:  DESPLIEGA PANTALLA DE PARTIDA A
                         ELIMINAR
               ELSE
                    S5:  ACTUALIZA STATUS DE BAJA
                    S6:  IMPRIME CONTRATO
               END-IF
          END-IF
          IF RESP=SI THEN
               S6: LEE NUMERO DE PARTIDAS DEL REGISTRO
                   HEADER
               S7: GRABA STATUS DE BAJA EN HEADER
               DO I=1 UNTIL NPA
                    S8:  LEE PARTIDA
                    S9:  ACTUALIZA STATUS DE BAJA EN PARTIDA
                    S10: I=I+1
               END-DO
          END-IF
     END-IF
END-SUBRUTINA
```

ODULO DE CAMBIOS DE ORDENES DE SERVICIOS Y MATERIALES NO PRODUCTIVOS

Programa principal

```
WHILE READ( ARCHIVO DE PANTALLA ) DO
     CASE IND OF
          1:  EJECUTA AA00
          2:  EJECUTA AB00
          3:  EJECUTA AC00
          4:  EJECUTA AD00
          5:  EJECUTA AF00
          6:  EJECUTA AG00
          7:  EJECUTA AC00
          8:  EJECUTA AD00
          9:  EJECUTA AF00
          10: EJECUTA AG00
          11: EJECUTA AC00
          12: EJECUTA AD00
     END-CASE
END-WHILE
```

Subrutina que inicializa variables para la seleccion de tipo de servicio a trabajar

```
AA00 BEGIN-SUBRUTINA
     S1: INICIALIZA VARIABLES DE PANTALLA
     S2: DESPLIEGA PANTALLA DE SELECCION DE TIPO
END-SUBRUTTINA
```

Subrutina de validacion del tipo de servicio

```
AB00 BEGIN-SUBRUTINA
     S1: VALIDA TIPO DE SERVICIO
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE SELECCION DE TIPO Y MEN-
               SAJE DE ERROR
     ELSE
          CASE CVE OF
               1:   DESPLIEGA  PANTALLA DE SELECCION DE ORDEN
                    DE MATERIAL NO PRODUCTIVO
               2:   DESPLIEGA PANTALLA DE SELECCION DE  ORDEN
                    DE SERVICIO
               3:   DESPLIEGA  PANTALLA  DE SELECCION DE CON-
                    TRATO
```

```
          END-CASE
     END-IF
END-SUBRUTINA
```

Subrutina que valida ordenes de servicio o materiales

```
AC00 BEGIN-SUBRUTINA
     S1: LEE ORDEN DE SERVICIO O MATERIALES
     S2: VALIDA ORDEN DE SERVICIO O MATERIALES
     IF ERROR THEN
          S3:    DESPLIEGA PANTALLA DE SELECCION Y MENSAJE DE
                 ERROR
     ELSE
          S4: ARMA PANTALLA DE DATOS GENERALES
          CASE CVE OF
               1:   DESPLIEGA PANTALLA DE DATOS GRALES DE OR-
                    DENES DE MATERIALES NO PRODUC.
               2:   DESPLIEGA PANTALLA DE DATOS GRALES DE OR-
                    DENES DE SERVICIO
               3:   DESPLIEGA PANTALLA  DE  DATOS  GRALES. DE
                    CONTRATOS
          END-CASE
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de datos generales

```
AD00 BEGIN-SUBRUTINA
     S1: VALIDA DATOS GENERALES
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE DATOS GRALES  Y  MENSAJE
               DE ERROR
     ELSE
          S3:  ARMA PANTALLA DE TIPO DE ACTUALIZACION
          S4:  MUEVE CAMPOS DE PANTALLA A CAMPOS DE  SALIDA E
               IMPRESION
          IF CVE NE 3 THEN
               S5:  DESPLIEGA  PANTALLA DE SELECCION  DE TIPO
                    DE ACTUALIZACION
          ELSE
               S6:    MUEVE  CAMPOS  DE  PANTALLA A CAMPOS DE
                    SALIDA E IMPRESION
               S7:  ACTUALIZA REGISTRO DE CONTRATO
          END-IF
     END-IF
END-SUBRUTINA
```

Subrutina que valida opcion de actualizacion

```
AF00 BEGIN-SUBRUTINA
     S1: VALIDA TIPO DE ACTUALIZACION
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA DE TIPO DE ACTUALIZACION Y
               MENSAJE DE ERROR
     ELSE
          IF RESP=NO THEN
               S3: VALIDA PARTIDA
               IF PARTIDA=0 THEN
                    S4:  DESPLIEGA PANTALLA  DE  TIPO  DE AC-
                         TUALIZACION Y MENSAJE DE ERROR
               ELSE
                    S5:  ARMA PANTALLA DE CAMBIO DE PARTIDA
                    S6: DESPLIEGA PANTALLA  DE  ACTUALIZACION
                         DE PARTIDA
               END-IF
          END-IF
          IF RESP=SI THEN
               S7: VALIDA PARTIDA
               IF PARTIDA NE 0 THEN
                    S8:  DESPLIEGA  PANTALLA  DE  TIPO DE AC-
                         TUALIZACION Y MENSAJE DE ERROR
               ELSE
                    S9:  ARMA PANTALLA DE ADICION DE PARTIDA
                    S10: LEE REGISTRO HEADER
                    S11: NPA=NPA+1
                    S12: DESPLIEGA  PANTALLA DE ACTUALIZACION
                         DE PARTIDA
               END-IF
          END-IF
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de partidas actualizadas

```
AG00 BEGIN-SUBRUTINA
     IF IND=KK THEN
          S1: EJECUTA AG200
          S2: DESPLIEGA PANTALLA DE SELECCION DE CLAVE
     ELSE
          S3: VALIDA CAMPOS DE PARTIDA ACTUALIZADA
          IF ERROR THEN
               S4:  DESPLIEGA  PANTALLA  DE  ACTUALIZACION DE
                    PARTIDAS Y MENSAJE DE ERROR
          ELSE
               S5:  LEE REGISTRO HEADER
               DO I=1 UNTIL NPA
```

```
                    S5:  MUEVE  CAMPOS DE PARTIDA A CAMPOS DE
                         IMPRESION
                    IF RESP=SI THEN
                         S6:  ARMA REGISTRO PARA ALTA
                         S7:  GRABA NUEVO REGISTRO
                         S8:  IMPRIME REGISTRO
                    ELSE
                         S9:  ARMA REGISTRO PARA CAMBIO
                         S10: ACTUALIZA REGISTRO
                         S11: IMPRIME REGISTRO ACTUALIZADO
                    END-IF
                    S12: I=I+1
               END-DO
          END-IF
     END-IF
END-SUBRUTINA
```

Subrutina que imprime la orden sin actualizar archivos

```
AG200 BEGIN-SUBRUTINA
     S1: LEE REGISTRO HEADER
     S2: MUEVE CAMPOS DEL REGISTRO A CAMPOS DE IMPRESION
     S3: IMPRIME DATOS GRALES.
     DO I=1 UNTIL NPA
          S4: LEE PARTIDA
          S5: MUEVE CAMPOS DE ARCHIVOS A CAMPOS DE IMPRESION
          S6: IMPRIME PARTIDA
          S7: I=I+1
     END-DO
END-SUBRUTINA
```

MODULO DE CONSULTA DE ORDENES DE SERVICIO Y MATERIALES NO PRODUCTIVOS

Programa principal

```
     WHILE READ ( ARCHIVO DE PANTALLA ) DO
          CASE IND OF
               1:  EJECUTA AA00
               2:  EJECUTA AB00
               3:  EJECUTA AC00
               4:  DESPLIEGA PANTALLA CON DATOS GRALES
               6:  EJECUTA AF00
               7:  DESPLIEGA PANTALLA DE ENCABEZADOS
               8:  EJECUTA AC00
               9:  DESPLIEGA PANTALLA DE DATOS GRALES
```

```
                11: EJECUTA AF00
                12: DESPLIEGA PANTALLA DE ENCABEZADOS
                13: EJECUTA AC00
                14: EJECUTA AA00
          END-CASE
     END-WHILE
```

Subrutina que inicializa pantallas

```
AA00 BEGIN-SUBRUTINA
     S1:  BLANQUEA CAMPOS EN PANTALLA DE SELECCION DE TIPO DE
          SERVICIO
     S2:  DESPLIEGA PANTALLA DE SELECCION DE SERVICIO
END-SUBRUTINA
```

Subrutina de validacion de tipo de servicio

```
AB00 BEGIN-SUBRUTINA
     S1: VALIDA TIPO DE SERVICIO
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA  DE  SELECCION  DE TIPO DE
               SERVICIO Y MENSAJE DE ERROR
     ELSE
          S3:  DESPLIEGA PANTALLA DE SELECCION DE SERVICIO
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de ordenes de servicios y materiales no productivos

```
AF00 BEGIN-SUBRUTINA
     DO J=1 UNTIL NPA
          DO I=1 UNTIL 3
               S1: LEE PARTIDA
               S2: ARMA PANTALLA DE PARTIDA
               S3: DESPLIEGA PANTALLA DE PARTIDA
               S4: I=I+1
          END-DO
          S5: DESPLIEGA ENCABEZADOS
          S6: J=J+1
     END-DO
END-SUBRUTINA
```

Seudocodigo del proceso 2.2 (AUTORIZACION DE ORDENES DE COMPRA POR REQUISICION/COMPRAS/DIRECCION)

```
MODULO DE AUTORIZACIONES DE ORDENES DE COMPRA DE MATERIA PRIMA,
     SERVICIO Y MATERIALES NO PRODUCTIVOS


     Programa principal

          WHILE READ ( ARCHIVO DE PANTALLA ) DO
               CASE IND OF
                    1:  EJECUTA AOO
                    2:  EJECUTA BOO
                    KL: FIN DE TRABAJO
               END-CASE
          END-WHILE


     Subrutina de autorizacion de ordenes de compra por
                    requisicion


     AOO BEGIN-SUBRUTINA
          S1: DESPLIEGA PANTALLA DE SOLICITUD DE PROV.
          CASE IND OF
               11: EJECUTA AAOO
               12: EJECUTA ABOO
               KL: FIN DE TRABAJO
               KK: DESPLIEGA PANTALLA DE SOLICITUD DE PROVEEDOR
          END-CASE
     END-SUBRUTINA


     Subrutina de validacion de proveedor


     AAOO BEGIN-SUBRUTINA
          S1: LEE PROVEEDOR
          IF ERROR THEN
               S2: DESPLIEGA PANTALLA DE SOL.  DE PROV.  Y MENSAJE
                   DE ERROR
          ELSE
               S3:  ARMA PANTALLA DE SOL. DE D.C. A AUTORIZAR
          END-IF
     END-SUBRUTINA
```

Subrutina de validacion y autorizacion de ordenes de compra

```
AB00 BEGIN-SUBRUTINA
     DO I=1 UNTIL 28
          S1: LEE ORDEN DE COMPRA PARA AUTORIZAR
          IF ERROR THEN
               S2:  DESPLIEGA PANTALLA DE SOL. DE O.C. Y MEN-
                    SAJE DE ERROR
          ELSE
               S3:  EJECUTA ABA00
          END-IF
     END-DO
END-SUBRUTINA
```

Subrutina de autorizacion de orden de compra

```
ABA00 BEGIN-SUBRUTINA
     DO I=1 UNTIL 28
          S1: LEE REGISTRO HEADER DE LA O.C.
          S2: ACTUALIZA STATUS DE AUTORIZACION
          DO J=1 UNTIL NPA
               S3:  LEE PARTIDA O.C.
               S4:  ACTUALIZA  STATUS DE AUTORIZACION EN PAR-
                    TIDA
               S5:  J=J+1
          END-DO
          S6: I=I+1
     END-DO
END-SUBRUTINA
```

Subrutina de autorizacion de O.C. por direccion

```
B00 BEGIN-SUBRUTINA
     S1: DESPLIEGA PANTALLA DE SOLICITUD DE PROVEEDOR
     CASE IND OF
          21: EJECUTA AA00
          22: EJECUTA BB00
          KL: FIN DE TRABAJO
          KK: DESPLIEGA PANTALLA DE SELECCION DE PROV.
     END-CASE
END-SUBRUTINA
```

Subrutina de validacion, autorizacion e impresion de orden de compra

```
BB00 BEGIN-SUBRUTINA
     DO I=1 UNTIL 28
          S1: LEE ORDEN DE COMPRA A AUTORIZAR
          IF ERROR THEN
               S2:  DESPLIEGA  PANTALLA DE SOL.DE O.C. Y MEN-
                    SAJE DE ERROR
          ELSE
               S3:  EJECUTA BBA00
          END-IF
          S4: I=I+1
     END-DO
END-SUBRUTINA
```

Subrutina de autorizacion e impresion de orden de compra de materia prima por la direccion

```
BBA00 BEGIN-SUBRUTINA
     DO I=1 UNTIL 28
          S1: LEE REG. HEADER DE O.C.
          S2: ACTUALIAZA STATUS DE AUTORIZACION
          S3: IMPRIME DATOS GENERALES DE O.C.
          DO J=1 UNTIL NPA
               S4: LEE PARTIDAS
               S5: ACTUALIZA STATUS DE AUTORIZACION
               S6: IMPRIME PARTIDAS
               S7: J=J+1
          END-DO
          S8: I=I+1
     END-DO
     S9: IMPRIME TOTALES O.C. Y NOTA PIE DE PAGINA
END-SUBRUTINA
```

Seudocodigo proceso 3.1 (CAPTURA Y VALIDACION DE REMISION)

MODULO DE CAPTURA DE REMISIONES DE MATERIALES NO PRODUCTIVOS Y MATERIA PRIMA

Programa principal

```
     WHILE READ ( ARCHIVO DE PANTALLA ) DO
          CASE IND OF
               1:  EJECUTA AA00
               2:  EJECUTA AB00
```

```
                3:  EJECUTA ACOO
                4:  EJECUTA ADOO
                5:  EJECUTA AEOO
                6:  EJECUTA ACOO
                7:  EJECUTA ADOO
                8:  EJECUTA AEOO
                9:  EJECUTA ACOO
                10: EJECUTA ADOO
                11: EJECUTA AEOO
          END-CASE
     END-WHILE
```

Subrutina que inicializa variables de la primera pantalla

```
AAOO BEGIN-SUBRUTINA
     S1:  BLANQUEA CAMPOS DE SELECCION DE TIPO DE  REMISIONES
          A TRABAJAR
     S2: DESPLIEGA PANTALLA DE SELECCION DE TIPO DE  REMISION
          A TRABAJAR
END-SUBRUTINA
```

Subrutina de validacion del tipo de remision a trabajar

```
ABOO BEGIN-SUBRUTINA
     S1: VALIDA LAS OPCIONES DE TRABAJO
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA  DE  SELECCION DE OPCION A
               TRABAJAR Y MENSAJE DE ERROR
     ELSE
          S3:    ARMA  PANTALLA DE CAPTURA DE DATOS GENERALES
               DEL DOCUMENTO
          S4:  DESPLIEGA  PANTALLA  DE  DATOS  GENERALES  DEL
               DOCUMENTO
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de datos generales para entradas

```
ACOO BEGIN-SUBRUTINA
     S1: VALIDACION DE DATOS GRALES.
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA DE DATOS GENERALES MENSAJE
               DE ERROR
     ELSE
          IF CONTRATO THEN
```

```
            S3: ACTUALIZA ARCHIVO DE CAPTURA
        ELSE
            S4: ARMA PANTALLA DE DETALLE DEL DOCUMENTO
            S6: DESPLIEGA PANTALLA DE DETALLE
        END-IF
    END-IF
END-SUBRUTINA
```

Subrutina de validacion del detalle del documento

```
AD00 BEGIN-SUBRUTINA
     IF ARREGLOS = '          '    THEN
          S1:  DESPLIEGA  PANTALLA DE SELECCION DE OPCIONES A
               TRABAJAR
     ELSE
          DO I=1 UNTIL 15
               S2: VALIDA DETALLE DE LA REMISION
               IF ERROR THEN
                    S3:  DESPLIEGA PANTALLA DE DETALLE Y MEN-
                         SAJE DE ERROR
               ELSE
                    S4: ARMA REGISTRO DEL DOCUMENTO
                    S5: ACTUALIZA ARCHIVO DE DOCUMENTOS
               END-IF
               S6: I=I+1
          END-DO
          S7: DESPLIEGA PANTALLA DE DETALLE DE PARTIDAS
     END-IF
END-SUBRUTINA
```

Subrutina de impresion de captura

```
AE00 BEGIN-SUBRUTINA
     S1: LEE REGISTRO GRABADO
     S2: IMPRIME DATOS DE CAPTURA
END-SUBRUTINA
```

Seudocodigo del proceso 3.2 (ACTUALIZACION DE ENTRADAS POR COMPAS)

```
MODULO DE ACTUALIZACION DE ENTRADAS POR COMPRAS


Programa principal

     WHILE READ ( ARCHIVO ORDENADO POR TIPO DE MOVTO.) DO
          CASE TIPO OF
               1: EJECUTA AA00
               2: EJECUTA AA00
               3: EJECUTA AA00
               4: EJECUTA AB00
          END-CASE
     END-WHILE


Subrutina de actualizacion de ordenes de compra de materiales
          no productivos y ordenes de servicio


AA00 BEGIN-SUBRUTINA
     S0: INICIALIZA VARIABLES
     S1: LEE PARTIDAS DE REMISION
     S2: ADICIONA IMPORTES A CARTERA DE PROVEEDORES
     S3: ARMA REGISTRO DE APLICACION CONTABLE
     S4: GRABA REGISTRO DE APLICACION CONTABLE
     S5: LEE REGISTRO DE LA ORDEN DE COMPRA
     S6: ACTAULIZA REGISTRO DE LA ORDEN DE COMPRA
     IF ULTIMO REGISTRO DEL DOCTO  THEN
          S7: GRABA REGISTRO DE CARTERA ( CXP )
     END-IF
END-SUBRUTINA


Subrutina de actualizacion de ordenes de compra de materiales
          no productivos y ordenes de servicio


AB00 BEGIN-SUBRUTINAA
     S0: INICIALIZA VARIABLES
     S1: LEE PARTIDAS DE REMISION
     S2: ADICIONA IMPORTES A CARTERA DE PROVEEDORES
     S3: ARMA REGISTRO DE APLICACION CONTABLE
     S4: GRABA REGISTRO DE APLICACION CONTABLE
     S5: LEE REGISTRO DE LA ORDEN DE COMPRA
     S6: ACTAULIZA REGISTRO DE LA ORDEN DE COMPRA
     IF ULTIMO REGISTRO DEL DOCTO     THEN
          S7: GRABA REGISTRO DE CARTERA ( CXP )
     END-IF
     IF TIPO=4 THEN
```

```
            S8: ACTUALIZA ARCHIVOS DE INVENTARIO
        END-IF
    END-SUBRUTINA
```

Seudocodigo del proceso 5.1 (CAPTURA Y VALIDA FACTURAS)

```
MODULO DE CAPTURA DE FACTURAS


    Programa Principal

        WHILE READ ( ARCHIVO DE PANTALLA ) DO
            CASE IND OF
                1: EJECUTA AA00
                2: EJECUTA AB00
                3: EJECUTA AC00
                4: EJECUTA AD00
            END-CASE
        END-DO


    Subrutina que inicializa primera pantalla


    AA00 BEGIN-SUBRUTINA
        S1:  INICIALIZA  VARIABLES  DE SELECCION DE TIPO DE FAC-
             TURA IDENTIFICACION DE LA MISMA
        S2:  DESPLIEGA PANTALLA DE SELECCION DE FACTURA
    END-SUBRUTINA


    Subrutina de validación de datos de selección


    AB00 BEGIN-SUBRUTINA
        S1: VALIDA DATOS DE IDENTIFICACION DE LA FACTURA
        IF ERROR THEN
            S2:  DESPLIEGA  PANTALLA  DE IDENFIFICACION DE FAC-
                 TURA Y MENSAJE DE ERROR
        ELSE
            S3:  ARMA PANTALLA DE DATOS GENERALES DE LA FACTURA
            S4:  DESPLIEGA PANTALLA DE DATOS GRALES DE FACTURA
        END-IF
    END-SUBRUTINA
```

Subrutina de validacion de datos generales

```
ACOO BEGIN-SUBRUTINA
     S1: VALIDA DATOS GRALES.
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE DATOS GENERALES Y MENSA-
               JE DE ERROR
     ELSE
          S3:  ARMA REGISTRO HEADER
          S4:  ACTUALIZA REGISTRO EN ARCHIVO DE FACTURAS
          S5:  ARMA PANTALLA DE DETALLE
          S6:  DESPLIEGA PANTALLA DE DETALLE
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de detalle de la factura

```
ADOO BEGIN-SUBRUTINA
     IF IND=KG THEN
          S1:  ARMA PANTALLA DE SELECCION DE FACTURA
          S2:  DESPLIEGA PANTALLA DE SELECCION DE LA FACTURA
     END-IF
     S1: VALIDA DATOS DE DETALLE
     IF ERROR THEN
          S4:  DESPLIEGA  PANTALLA  DE  DETALLE  Y MENSAJE DE
               ERROR
     ELSE
          S5:  ARMA REGISTRO DE DETALLE Y CAMPOS DE IMPRESION
          S6:  ACTUALIZA ARCHIVO DE FACTURAS CON EL DETALLE
          S7:  IMPRIME FACTURA CAPTURADA
          S8:  ARMA PANTALLA DE DETALLE
          S9:  DESPLIEGA PANTALLA DE DETALLE
     END-IF
END-SUBRUTINA
```

MODULO DE VALIDACION DE FACTUARAS.

Programa Principal

```
     WHILE READ (ARCHIVO DE FACTURAS)  DO
          S1: SELECCIONA FACTURA POR FECHA
          S2: VALIDA DATOS GENERALES
          S3: IMPRIME DATOS GENERALES
```

```
        IF ERROR THEN
             S4: IMPRIME MENSAJES DE ERROR EN DETALLE
        ELSE
             S5: LEE DETALLE DE LA FACTURA
             S6: VALIDA DETALLE DE LA FACTURA
             S7: IMPRIME DETALLE DE LA FACTURA
             IF ERROR THEN
                  S8: IMPRIME MENSAJE DE ERROR
             END-IF
             S9:  ACTUALIZA FECHA DE RECEPCION  DE  FACTURA
                  EN LA CUENTA POR PAGAR
        END-IF
   END-WHILE
```

Seudocódigo del proceso 5.2 (GENERA CUENTA POR PAGAR)

```
   MODULO DE GENERACION DE CUENTA POR PAGAR

   Programa principal

        WHILE READ (ARCHIVO DE FACTURAS VALIDADAS) DO
             S1: LEE REGISTRO DE FACTURA VALIDADA
             S2: CONTROL=NUMFAC
             DO NUMFAC=CONTROL UNTIL
                  S3: TOTFAC=IMPFAC=TOTFAC
             END-DO
             S4: ARMA REGISTRO DE ARCHIVO DE CXP
             S5: GRABA REGISTRO DE CARTERA
             S6: ARMA REGISJTRO DE CUENTA CORRIENTE
             S7: GRABA REGISTRO DE CUENTA CORRIENTE
        END-DO
```

Seudocódigo del proceso 6.1 (ACTUALIZACION DE SALDOS)

```
   MODULO  DE   ACTUALIZACION   DE  SALDOS

Programa Principal

        WHILE READ ( ARCHIVO ORDENADO DE CARTERA ) DO
             S1:  LEE REGISTRO DE CARTERA
             S2:  CALCULA LAS FECHAS DE VENCIMIENTOS
```

```
        S3:  LEE REGISTRO DEL PROVEEDOR
        S4:  ACTUALIZA SALDOS DEL PROVEEDOR
        S5:  ACTUALIZA ARREGLOS DE SALDOS VENCIDOS Y POR
             VENCER PRO DIA
        IF LAST-RECORD THEN
            S6: IMPRIME SALDOS VENCIDOS Y POR VENCER
        END-IF
    END-WHILE
```

Seudocodigo del proceso 6.2(REPORTE DE SALDOS VENCIDOS Y POR VENCER)

```
        MODULO DE  IMPRESION  DE SALDOS

Programa Principal

    WHILE READ (ARCHIVO ORDENADO DE CARTERA POR PROVEEDORES
            Y CXP) DO
        S1: CALCULA FECHAS DE VENCIMIENTO
        IF  SDO=VENCIDO THEN
            S2:  ADICIONA  IMPORTE  A  SALDO  VENCIDO  DE
                 ACUERDO AL NO. DE DIAS
        ELSE
            S3:  ADICIONA  IMPORTES A SALDOS POR VENCER DE
                 ACUERDO AL NO. DE DIAS
        END-IF
        S4:  IMPRIME NO.  DE DOCUMENTO,FECHA DE VENCIMIENTO
             DIAS VENCIDOS, IMPORTE, NO. DE FACTURA, NO. DE
             DOCUMENTO DE ENTRADA
        IF NO. DE PROV. DISTISTINTO THEN
            S5: IMPRIME TOTALES POR PROVEEDOR
        END-IF
        IF CIA DISTINTA THEN
            S6: IMPRIME TOTALES POR CIA
        END-IF
    END-WHILE
```

Seudocodigo del proceso 7.1 (SELECCION DE SALDOS)

```
MODULO DE CONSULTA E IMPRESION DE ESTADOS DE CUENTA DE PROVEEDORES

Programa Principal

    WHILE READ ( ARCHIVO DE PANTALLA ) DO
        CASE IND OF
            1:   DESPLIEGA PANTALLA DE SELECCION DEL PROV.
                 SPROC=0
```

```
            2:   EJECUTA AA00
            3:   EJECUTA AB00
            NKK: EJECUTA AC00
            6:   ARMA PANTALLA DE GRAFICACION DESPLIEGA
                 PANTALLA DE GRAFICACION
        END-CASE
    END-WHILE
```

Subrutina de validacion de no. de Proveedor

```
AA00 BEGIN
     S1: VALIDA PROVEEDOR
     IF ERROR THEN
          S2:   DESPLIEGA PANTALLA DE SELECCION Y MENSAJE DE
                ERROR
     ELSE
          S3:  LEE REGISTRO DE PROVEEDOR
          S4:  ARMA PANTALLA DE SALDO TOTAL Y TIPO DE EDO. DE
               CTA.
          S5:  DESPLIEGA PANTALLA DE SELECCION
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de tipo de estado de cuenta y saldo total

```
AB00 BEGIN-SUBRUTINA
     S1: VALIDA TIPO DE EDO. DE CTA.
     IF ERROR THEN
          S2: DESPLIEGA PANTALLA DE SELECCION
     ELSE
          S3: INICIALIZA VARIABLES DE TOTALES
          S4: INICIALIZA APUNTADOR DEL REGISTRO
          IF RESP=Y THEN
               S5: ARMA ENCABEZADOS DE PANTALLA DE PARTIDAS
               S6: DESPLIEGA PANTALLA DE ENCABEZADOS
          ELSE
               S7: EJECUTA AG00
          END-IF
     END-IF
END-SUBRUTINA
```

Subrutina de consulta de edos. de cta.

```
ACOO BEGIN-SUBRUTINA
     DO LIN=1 UNTIL 19
          S1: LEE REGISTRO DE CARTERA
          S2: CALCULA TIEMPO VENCIDO
          S3: ACUMULA A TOTALES
          S4: ARMA PANTALLA DE DOCUMENTO
          S5: DESPLIEGA PANTALLA DE DOCUMENTO
          S6: LINE=LINE+1
     END-DO
     S7: DESPLIEGA PANTALLA DE TOTALES
END-SUBRUTINA
```

Subrutina de impresión de estados de cuenta

```
AGOO BEGIN-SUBRUTINA
     S1: LEE REGISTRO DE DOCUMENTO ( CARTERA)
     S2: CALCULA TIEMPO VENCIDO PARA IMPRESION
     S3: ADICIONA IMPORTES A TOTALES
     S4: IMPRIME DOCUMENTO Y DETALLE DEL DOCTO.
     IF PROVEEDOR DISTINTO  THEN
          S5: IMPRIME TOTALES DEL PROVEEDOR
     END-IF
END-SUBRUTINA
```

Seudocodigo del proceso 7.2 (AFECTACION DE CARGOS Y ABONOS)

MODULO DE AFECTACION DE CARGOS Y ABONOS A CARTERA DE PROVEEDORES

Programa Principal

```
     WHILE  READ ( ARCHIVO DE PANTALLA )  DO
          CASE IND OF
               1: EJECUTA AAOO
               2: EJECUTA ABOO
               3: EJECUTA ACOO
               4: EJECUTA ADOO
          END-CASE
     END-WHILE
```

Subrutina de inicializacion de pantalla de seleccion tipo de movto. y Proveedor

```
AA00 BEGIN-SUBRUTINA
     S1:  ARMA  E INICIALIZA PANTALLA DE SELECCION DE TIPO DE
          MOVTO. Y PROVEEDOR
     S2:  DESPLIEGA PANTALLA DE SELECCION DE TIPO DE MOVTO.
END-SUBRUTINA
```

Subrutina de validacion de tipo de movto.

```
AB00 BEGIN-SUBRUTINA
     S1: VALIDA DATOS DE SELECCION
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA  DE  SELECCION  DE TIPO DE
               MOVTO. Y MENSAJE DE ERROR
     ELSE
          S3:  ARMA PANTALLA DE DATOS GENERALES DEL MOVTO.
          S4:  DESPLIEGA PANTALLA DE DATOS GRALES
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de datos grales.

```
AC00 BEGIN-SUBRUTINA
     S1: VALIDA DATOS GENERALES
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE DATOS GENERALES Y MENSA-
               JE DE ERROR
     ELSE
          S3:  ARMA PANTALLA DE PARTIDAS
          S4:  DESPLIEGA PANTALLA DE PARTIDAS
     END-IF
END-SUBRUTINA
```

Subrutina de validacion de partidas.

```
AD00 BEGIN-SUBRUTINA
     DO I=1 UNTIL 18
          S1: VALIDA PARTIDA
          IF ERROR THEN
               S2:  DESPLIEGA  PANTALLA DE PARTIDAS Y MENSAJE
                    DE ERROR
          ELSE
               S3:  ARMA REGISTRO DE CARTERA
               S4:  ARMA REGISTRO DE APLICACION CONTABLE
               S5:  ACTUALIZA ARCHIVO DE CARTERA
```

```
                S6:  ACTUALIZA ARCHIVO DE APLICACION CONTABLE
                S7:  LEE ARCHIVO DE PROV.
                S8:  ACTUALIZA SALDOS DE PROV.
                S12:  IMPRIME DETALLE DEL MOVIMIENTO
                S13:  I=I+1
          END-IF
     END-DO
     S14: ARMA PANTALLA DE PARTIDAS
     S15: DESPLIEGA PANTALLA DE PARTIDAS
END-SUBRUTINA
```

Seudocódigo del proceso 7.3 (AFECTACION DE ABONOS CON CHEQUE)

MODULO DE AFECTACION DE CARGOS Y ABONOS A CARTERA DE PROVEEDORES

Programa Principal

```
WHILE  READ ( ARCHIVO DE PANTALLA )  DO
     CASE IND OF
          1: EJECUTA AA00
          2: EJECUTA AB00
          3: EJECUTA AC00
          4: EJECUTA AD00
     END-CASE
END-WHILE
```

Subrutina de inicializacion de pantalla de seleccion tipo de movto. y Proveedor

```
AA00 BEGIN-SUBRUTINA
     S1:  ARMA  E INICIALIZA PANTALLA DE SELECCION DE TIPO DE
          MOVTO. Y PROVEEDOR
     S2:  DESPLIEGA PANTALLA DE SELECCION DE TIPO DE MOVTO.
END-SUBRUTINA
```

Subrutina de validacion de tipo de movto.

```
AB00 BEGIN-SUBRUTINA
     S1: VALIDA DATOS DE SELECCION
     IF ERROR THEN
          S2:  DESPLIEGA  PANTALLA  DE  SELECCION  DE TIPO DE
               MOVTO. Y MENSAJE DE ERROR
     ELSE
          S3:  ARMA PANTALLA DE DATOS GENERALES DEL MOVTO.
```

```
          S4:  DESPLIEGA PANTALLA DE DATOS GRALES
     END-IF
END-SUBRUTINA
```

Subrutina de validación de datos grales.

```
ACOO BEGIN-SUBRUTINA
     S1: VALIDA DATOS GENERALES
     IF ERROR THEN
          S2:  DESPLIEGA PANTALLA DE DATOS GENERALES Y MENSA-
               JE DE ERROR
     ELSE
          S3:  ARMA PANTALLA DE PARTIDAS
          S4:  DESPLIEGA PANTALLA DE PARTIDAS
     END-IF
END-SUBRUTINA
```

Subrutina de validación de partidas.

```
ADOO BEGIN-SUBRUTINA
     DO I=1 UNTIL 18
          S1: VALIDA PARTIDA
          IF ERROR THEN
               S2:  DESPLIEGA  PANTALLA DE PARTIDAS Y MENSAJE
                    DE ERROR
          ELSE
               S3:  ARMA REGISTRO DE CARTERA
               S4:  ARMA REGISTRO DE APLICACION CONTABLE
               S5:  ACTUALIZA ARCHIVO DE CARTERA
               S6:  ACTUALIZA ARCHIVO DE APLICACION CONTABLE
               S7:  LEE ARCHIVO DE PROV.
               S8:  ACTUALIZA SALDOS DE PROV.
                    S9:  ARMA REGISTRO DE POLIZA CHEQUE
                    S10: ACTUALIZA ARCHIVO DE POLIZA CHEQUE
                    S11: IMPRIME CHEQUE
               S12:  IMPRIME DETALLE DEL MOVIMIENTO
               S13:  I=I+1
          END-IF
     END-DO
     S14: ARMA PANTALLA DE PARTIDAS
     S15: DESPLIEGA PANTALLA DE PARTIDAS
END-SUBRUTINA
```

CAPITULO 8 Verificacion y validacion de los programas

Cuando un sistema de programación es desarrollado las preocupaciones primordiales de los ingenieros en programación son la calidad y confiabilidad de éste, porque los programas no tienen sabor, volumen, color, olor, es decir carecen de propiedades físicas, no estan sujetos a leyes de gravedad ni de electrodinámica; debido a lo anterior, los programas son inlangibles, por lo cual es muy difícil cuantificar el estado del proyecto durante el desarrollo utilizando la intuición, por esta razón se deben de tomar medidas especiales para determinar el progreso real del producto y definir las zonas con problemas. Estas medidas son la validación y verificación de los programas.

Los objetivos de la verificación y validación son valorar y mejorar la calidad en cada una de las etapas del desarrollo del sistema, desde el planteamiento de las necesidades del usuario hasta la entrega del sistema completo y manuales de éste. Los atributos de la calidad deben ser: la corrección, la perfección, la consistencia, la confiabilidad, la utilidad, la eficiencia, el apego a las normas y el abatimiento en los costos totales.

La verificación se divide en dos etapas la formal y del ciclo de vida; la segunda consiste en el proceso de revisar el grado en que los productos de trabajo de una de las etapas de ciclo de vida del desarrollo de sistema se cumple con las especificaciones establecidas durante las etapas previas; la verificación formal es una rigurosa demostración matemática de la similitud del código fuente con sus especificaciones. La validación es la evaluación del sistema al final de proceso de desarrollo de este para determinar si este cumple con todos los requisitos expresados por el usuario.

La verificación y validación son conceptos que abarcan todo, no un conjunto de actividades que ocurren estrictamente después de la implantación, esto implica la valoración de los productos de trabajo para establecer que tanto se apega a las especificaciones. Estas incluyen las especificaciones de requisitos, la documentación del diseño del sistema, diversos principios generales de estilo, normas organizacionales y especificaciones del usuario, al igual que las metaespecificaciones para los formatos y notaciones utilizadas en la especificación de productos diversos.

Se deben revisar los requisitos para asegurar que concuerden con las necesidades del usuario, así como con las restricciones del ambiente y las normas de notación; la documentación del diseño debe verificarse con respecto a los requisitos y las convenciones notacionales; del código fuente debe revisarse su conformación con los requisitos, con la documentación del diseño, con las espectativas del usuario y con los diversos estándares de instrumentación y documentación. Además de los documentos de apoyo (manual del usuario, plan de pruebas, principios de

operaciòn, etc.), se debe revisar que este correcto y completo que sea consistente y que se apegue a las normas.

La calidad de los productos de trabajo generados durante el anàlisis y el diseño se puede estimar y mejorar utilizando procedimientos sistemàticos de control de calidad, mediante recorridos e inspecciones, el anàlisis estàtico, la ejecuciòn simbòlica, las pruebas de unidad y las pruebas de integraciòn sistemàticas.

8.1 Control de calidad

El control de calidad se realiza mediante un modelo planeado, que comprende en forma sistemàtica todas las acciones necesarias para proporcionar la confianza de que el articulo o producto se ajusta a los requerimientos tècnicos establecidos. El control de calidad es realizado por un grupo externo al que desarrolla el sistema, esto otorga un grado de imparcialidad a esta actividad. El propòsito de este grupo es proporcionar la garantia de que los procedimientos, las herramientas y las tècnicas utilizadas durante el desarrollo y la modificaciòn del producto son adecuados para alcanzar el nivel de confianza deseado.

La principal funciòn del grupo de control de calidad es de realizar un plan de control de calidad para cada proyecto, este debe tocar los siguientes temas:

- Propòsito y alcances del plan

- Documentos referidos en el plan

- Estructura organizacional, tareas que se realizaràn y responsabilidades especificas relacionadas con la calidad del producto.

- Documentos que se deben preparar, y revisiones que deben efectuarse para la adecuaciòn de la documentaciòn.

- Normas, pràcticas y convenciones que se utilizaràn

- Revisiones y auditorias que deben llevarse a cabo

- Un plan de administraciòn de la configuraciòn que identifique los elementos del producto de programaciòn, control e implante los cambios y que registre e informe los estados modificados.

- Pràcticas y procedimientos que se deben seguir para conformar, rastrear, y resolver los problemas de los programas.

- Herramientas y tècnicas especificas que se usarán para apoyar las actividades de control de calidad.

- Mètodos y facilidades que se emplearàn para mantener y almacenar las versiones controladas de los programas.

- Mètodos y facilidades que servirán para proteger los medios fisicos del programa de computadora.

- Suministros para garantizar la calidad del sistema proporcionado por vendedores y desarrollado por subcontratistas.

- Mètodos y facilidades que se usarán para reunir, mantener y conservar los registros del control de calidad.

También el grupo de control de calidad lleva a cabo la ejecución de las funciones de control de calidad de acuerdo al plan de control de calidad, Así mismo, se tiene que realizar un plan de verificación del sistema y un plan de prueba de aceptación del mismo.

Durante el desarrollo del sistema, el grupo de control de calidad realizará auditorias en èste para verificar que los productos de trabajo sean consistentes y estèn completos. Estas auditorias serán: especificaciones de interfases para hardware, software y personas; diseño interno contra especificaciones funcionales; código fuente contra documentación del diseño; y requisitos funcionales contra descripciones de las pruebas.

Antes de entregar el sistema, se realizan una auditoria funcional y una auditoria fisica; la primera reconfirma el cumplimiento de todos los requerimientos, la segunda verifica que el código fuente y todos los documentos asociados estén completos, sean consistentes tanto internamente como uno con otro.

8.2 Recorridos e inspecciones.

Un recorrido es una revisión técnica de profundidad de algunos aspectos de un sistema de programación. Los recorridos pueden utilizarse en cualquier momento, durante cualquier fase de un proyecto de programación, como los requisitos de la programación, las especificaciones del diseño estructural, las especificaciones del diseño detallado, el plan de pruebas, el código, los documentos de apoyo o una modificación de mantenimiento propuesta.

El objetivo de un recorrido es descubrir y resaltar las àreas con problemas. Los problemas no se resuelven durante la sesión de recorrido, los soluciona después de la sesión la persona a la cual

se revisa su trabajo.

Un equipo de recorrido por lo general consta de una persona a la cual se revisa su trabajo y tres o cinco revisores. En proyectos de una o dos personas puede no convenir formar un equipo de revisiòn; sin embargo, la tècnica de recorrido suele ser benèfica con sòlo uno o dos revisores.

La persona cuyo material està siendo revisado es la responsable de proporcionar copias del material de revisiòn a los miembros del equipo de recorrido antes de la sesiòn de revisiòn, y los miembros del equipo son responsables de revisar el material previamente a la sesiòn. Durante el recorrido la persona a la cual se revisa su trabajo lee el material mientras que los revisores buscan errores, solicitan aclaraciones y exploran àreas problema en el material bajo revisiòn. Es importante subrayar que lo que se revisa es el material y no la persona. El enfoque de un recorrido se encuentra en la detecciòn de errores y no en acciones correctivas. Por ejemplo, se deben contemplar los aspectos principales de la eficiencia del còdigo, pero deben evitarse los aspectos secundarios del estilo de codificaciòn.

El èxito del uso de los recorridos depende mucho del establecimiento de una atmòsfera agradable no amenazadora en la sesiones de recorrido, y otros factores que contribuyen al èxito de èstos recorridos son: recalcar la detenciòn de los problemas principales, limitar la duraciòn de cada recorrido y hacer un calendario de los recorridos para cada miembro del grupo en forma regular.

Las inspecciones difieren de los recorridos en que un grupo de inspectores entrenados (que trabajan a partir de listas de revisiòn de elementos que se examinarân) realizan inspecciones de los productos de trabajo, las inspecciones se pueden usar como los corridos a lo largo del ciclo de vida de los sistemas para evaluar y mejorar la calidad de los diversos productos de trabajo.

Los elementos que se inspeccionarân en una inspecciòn de diseño, pueden incluir la perfecciòn del diseño con respecto al usuario y a los requisitos funcionales, la perfecciòn interna y la consistencia entre mòdulos. Por ejemplo los elementos que se inspeccionarân en una inspecciòn de còdigo pueden ser las interfaces entre subprogramas, las propociociones de entrada/salida, los comentarios, el flujo de los datos y el uso de memoria.

8.3 Analisis estatico

El anàlisis estàtico es una tècnica para valorar las caracteristicas estructurales del còdigo fuente o cualquier representaciòn notacional que se apegue a reglas sintàticas bien definidas. Este anàlisis se relaciona con la estructura de los

programas en código fuente, es especialmente útil para descubrir prácticas de codificación cuestionables y cuando las convenciones de codificación no son respetadas, además de detectar errores estructurales como variables no inicializadas y discrepancia entre parámetros reales y formales, el código no es ejecutado en este análisis.

El análisis estático se puede realizar manualmente utilizando las técnicas de recorrido o inspección, sin embargo el termino "análisis estático" se usa para denotar el exámen de la estructura de un programa por medio de una herramienta automatizada. Por lo general, con esta herramienta se construirá una tabla de símbolos y una gráfica de flujo de control para cada subprograma y además una gráfica de llamadas de rutinas para el programa completo.

La tabla de símbolos contiene información acerca de cada variable, esto es: el tipo de atributo, la posición donde se declara, las instrucciones en donde se asigna un nuevo valor y las instrucciones en donde se usa el valor.

La gráfica de flujo de control tiene nodos que corresponden a bloques básicos del código fuente, y los arcos representan transferencias posibles del control entre los bloques. En una gráfica de llamadas, los nodos representan unidades de programa y los arcos significan invocaciones potenciales a una unidad de programa.

Los analizadores estáticos, por lo común producen listas de errores, de prácticas de programación cuestionables (anomalías) y desviaciones de los convenciones de codificación. Por ejemplo, una variable que no se inicializa en ninguna parte de las rutas del programa es un error estructural. Una variable que tiene asignado un valor, pero que no se vuelve a utilizar en ninguna ruta de control subsecuente, o una variable que se declara pero que nunca se usa, no es un error, sin embargo, es una anomalía que puede ser síntoma de un error. El apartarse de los convenciones de codificación, como por ejemplo, utilizar construcciones que no esten en el FORTRAN de ANSI, transferencias de control GOTO hacia atrás o saltos al interior del cuerpo de ciclos, también se pueden detectar por el análisis estáticos.

Una lista de algunos productos que suele proporcionar un analizador estático automatizado, es la siguiente:

- Gráfica del flujo de control.

- Tabla de símbolos para variables (incluyendo los números de proposición donde se definieron, introdujeron y emplearon).

- Gráfica de llamada (subprogramas llamados por cada rutina).

- Argumentos pasados a cada rutina, desde donde se llamaron.

- Variables no inicializadas (en algunas trayectorias, en todas las trayectorias.

- Variables introducidas pero no utilizadas (en algunas trayectorias, en todas las trayectorias).

- Segmentos de código aislado (en algunas trayectorias, en todas las trayectorias.

- Desviación de las convenciones de codificación (normas del lenguaje, normas del proyecto).

- Errores en el orden de aparicion de las variables definidas en los bloques.

- Mal uso de parámetros en subprogramas: números incorrectos de parámetros verdaderos, tipos mal apareados entre los parámetros verdaderos y los formales, parámetros de salida formales fijados pero no empleados por la rutina que llama, parámetros verdaderos no manejados para entrada o salida por la rutina que llama.

- Número total de líneas.

- Número total de líneas y ubicación de las líneas de comentario.

- Número total de líneas y ubicación de líneas en blanco.

- Número y ubicación de etiquetas e instrucciones GOTO.

- Número y ubicación de llamadas del sistema.

- Número y ubicación de constantes literales.

Hay limitaciones tanto en prácticas como teóricas del análisis estático. Una limitación práctica principal implica la evaluación dinámica de las referencias a la memoria al momento de la ejecución. Un ejemplo de esto es: los subíndices de arreglos y las variables de tipo apuntador proporcionan referencias dinámica a la memoria basados en cálculos previos realizados por el programa. Los analizadores estáticos no pueden evaluar valores de subíndices o apuntadores; por lo que es imposible distinguir técnicas de análisis estático.

8.4 Ejecucion simbolica.

La ejecución simbólica es una técnica de validación en la que a las variables de entrada de una unidad de programa se le asignan valores simbólicos en vez de valores literales. Un programa se analiza propagando los valores simbólicos de las entradas dentro de los operandos en las expresiones. Las expresiones simbólicas resultantes se simplifican a cada paso en el cálculo; de modo que todos los cálculos y decisiones intermedias se expresan siempre en términos de las entradas simbólicas.

La ejecución simbólica puede utilizarse así en la obtención de las condiciones de ruta que se puede resolver para encontrar los valores de entrada de prueba que conducirán a un programa a lo largo de una ruta de ejecución particular, con la condición de que todos los predicados en la condición de ruta particular sean funciones lineales de los valores de entrada simbólicos. Cuando los predicados son no lineales en los valores de entrada, la condición de ruta puede o no puede ser resuelta porque los sistemas de desigualdades no lineales en general son sin resoluciones.

Además de emplearse en la obtención de datos de prueba, las condiciones de ruta se pueden usar para demostrar que operaciones como la división, la referencia de arreglos, y las operaciones de apuntadores son seguras (o inseguras) en regiones particulares del programa.

Los ciclos de los programas se pueden analizar utilizando árboles de ejecución simbólica. Un invariante del ciclo resume el comportamiento de un ciclo, independientemente del número de veces que éste se recorre.

Al final la condición postulada se conjunta con la condición de ruta, los valores de entrada que satisfacen la condición de ruta aumentada causarán el error. De modo contrario si no hay solución para la condición de ruta aumentada, el error no puede ocurrir bajo ningún conjunto de valores de entrada.

La ejecución simbólica se puede llevar a cabo en forma manual o con una herramienta automatizada. Los objetivos de esos sistemas experimentales incluyen la generación automática de expresiones de salida como funciones de las variables de entrada y la ruta de cálculos, la detección de rutas no factibles a través del programa, la detección de errores semánticos en el código, la generación automática de datos de prueba, la verificación informal de programas y la obtención de invariantes de ciclos. Todos estos sistemas son de naturaleza experimental. No hay versiones de producción ampliamente utilizadas de los sistemas de ejecución simbólica. Los problemas que deben sortearse en la ejecución simbólica son el manejo simbólico de los ciclos, la evaluación simbólica de subíndices y apuntadores, el tratamiento de desigual-

dades no lineales en expresiones de rutas y gran cantidad de detalles que se manejarán.

También observamos que la aplicación manual de la ejecución simbólica es una excelente herramienta para razonar acerca de los programas.

8.5 Pruebas de unidad y depuracion

Las pruebas de unidad comprenden el conjunto de pruebas realizadas por un programador individual, antes de la integración de la unidad en un sistema más grande. La situación se ilustra como sigue:

Codificación y depuración ---> Pruebas de unidad ---> Pruebas de integración

Una unidad de programa suele ser lo suficientemente pequeña como para que el programador que la desarrolló pueda probarla con minuciosidad, esta prueba debe ser mucho más minuciosa que el exámen al que se someterá cuando la unidad se integre en el producto de programación en desarrollo. Hay cuatro categorías de pruebas que, por lo común, efectuará un programador a una unidad de programa y que se pueden categorizar como funcionales, de desempeño, de tensión y de estructura.

La prueba funcional implica ejercitar el código con valores nominales de entrada para las cuales se conocen los resultados esperados; además de valores límites (valores mínimos, máximos, y valores sobre y justo fuera de los límites funcionales) y valores especiales.

La prueba de desempeño determina la cantidad de tiempo de ejecución empleado en varias partes de la unidad, la eficiencia global del programa, el tiempo de respuesta y la utilización de dispositivos por la unidad de programa.

Las pruebas de tensión son aquellas diseñadas para romper en forma intencional la unidad. En éstas pruebas se dan valores que de antemano son erróneos para provocar que la unidad se rompa o se pare la ejecucion. Se puede aprender mucho acerca de las resistencias y limitaciones de un programa examinando cómo se rompe una unidad de programa.

Las pruebas de estructura se relacionan con ejercitar la lógica interna de un programa y recorrer rutas de ejecución particulares. Las actividades principales en las pruebas estructurales son: decidir cuáles rutas ejercitar, obtener los datos de prueba para ejercitar esas rutas; determinar el criterio de cober-

tura de la prueba que se usará, ejecutar los casos de prueba y medir la cobertura de la prueba lograda cuando se ejercitaron esos casos.

Se debe establecer un criterio de cobertura o terminación de la prueba para las pruebas de unidad, porque las unidades de programa por lo normal contienen demasiadas rutas para permitir una prueba exahustiva.

Aún cuando fuera posible probar con éxito todas las rutas a través de un programa, la corrección no estaría garantizada por las pruebas en las rutas, porque el programa puede tener rutas ignoradas y errores computacionales, que los casos de prueba particulares elegidos no han descubierto. Un error de ruta ignorada ocurre cuando por accidente se omiten una proposición de ramificación y el cálculo asociado; tales errores sólo se pueden detectar por casos de prueba funcionales derivados de las especificaciones de requisitos. Por lo tanto las pruebas basadas exclusivamente en la estructura del programa no pueden detectar todos los errores potenciales en un programa fuente. La corrección casual sucede cuando un caso de prueba fracasa en la detección de un error computacional.

Los errores de los programas se pueden clasificar como errores por ignorar una ruta, computacionales y de dominio. Se ha observado que se requieren N+1 casos de prueba linealmente independientes para establecer la corrección computacional de un programa que realiza sólo cálculos lineales sobre N variables de entrada.

Un error de dominio ocurre cuando un programa recorre la ruta equivocada por un predicado incorrecto en una proposición de ramificación. Los límites de un dominio de ruta se determinan por las desigualdades en los casos de prueba. La teoría de pruebas de dominios necesita entre otros requisitos, que cada límite de cada dominio se determine por un predicado lineal que tenga sólo un operador relacional.

Por supuesto los programas reales no satisfacen las suposiciones de linealidad, de la teoría computacional o de prueba de dominios; por lo tanto requerirán aún más casos de pruebas que los indicados en la teoría para detectar estos tipos de error.

Establecer un criterio de terminación de pruebas es otra dificultad que se encuentra en las pruebas de unidad de programas reales. En la práctica hay tres medidas que suelen usarse en la prueba de unidad que son: cobertura de proposiciones, de ramificaciones y de rutas lógicas.

Al emplear la cobertura de proposiciones como criterio de terminación de la prueba, el programador está intentando hallar un conjunto de casos de prueba, que al correr todas las pruebas ejecutarán cada proposición en el programa por lo menos una vez.

Al utilizar la cobertura de ramificaciones como criterio de terminación de la prueba, el programador intenta encontrar un conjunto de casos de prueba que ejecute cada proposición de las ramificaciones en cada dirección por lo menos una vez.

La cobertura de rutas lógicas reconoce que el orden en que se ejecutan las ramas durante una prueba (una ruta recorrida) es un factor importante para determinar el resultado de la prueba.

A veces se usan criterios de terminación de pruebas basados en consideraciones distintas de la cobertura de ramificación. Estos criterios suelen implicar la terminación de las pruebas cuando se alcanza una tasa de descubrimiento de errores predeterminada (baja), o cuando se haya descubierto y corregido un número predeterminado de errores; por ejemplo cuando el 95% de los errores estimados se encuentra y se elimina. Las técnicas para estimar el número de errores que restan en un programa incluyen modelos predictivos, reglas empíricas, la siembra de errores y el trazo de tendencias.

Los modelos predictivos se basan en la teoría estadística. Las reglas empíricas se basan en la experiencia previa. La siembra de errores es una técnica que conlleva la introducción internamente (siembra) de errores en el código fuente. Después de cierto tiempo, el número de errores no sembrado descubierto durante las pruebas se multiplica por la razón de los errores sembrados totales a los errores sembrados descubiertos durante las pruebas, para producir una estimación del número total de errores no sembrados que permanecen en el programa.

Un trazo de tendencias es una gráfica de los errores localizados por unidad de tiempo contra tiempo.

La depuración es el proceso de aislar y corregir las causas de los errores conocidos. El éxito en la depuración requiere habilidades altamente desarrolladas en la solución de problemas. Los métodos de depuración que suelen ocuparse son la inducción, la deducción y el encadenamiento hacia atrás. La depuración por inducción comprende los siguientes pasos:

- Reunir la información disponible. Enumerar los hechos conocidos acerca de la falla y los hechos conocidos concernientes a los casos de prueba exitosos. Cuáles son los síntomas observados? Cuándo ocurrió el error? Bajo qué condiciones ocurrió? En qué difiere el caso con falla de los casos con éxito?

- Buscar patrones. Examinar la información reunida en busca de condiciones que diferencian el caso con falla, de los exitosos.

- Formular una o màs hipòtesis. Obtener una o màs hipòtesis de las relaciones observadas. Si ninguna hipòtesis resulta aparente, volver a examinar la informaciòn disponible y reunir informaciòn adicional,tal vez corriendo màs casos de prueba. Si surgen varias hipòtesis ordenarlas desde las màs probables hasta las menos probables.

- Demostrar o desechar cada hipòtesis. Volver a examinar la informaciòn disponible para determinar si la hipòtesis explica o no todos los aspectos del problema observado. No ignorar la posibilidad de que haya mùltiples errores. No proceder al paso siguiente hasta terminar con èste.

- Realizar las correcciones adecuadas. Hacer las correcciones indicadas por la evaluaciòn de las distintas hipòtesis. Realizar las correcciones en una copia de respaldo de còdigo, en caso que las modificaciones no sean las correctas.

- Verificar la correcciòn. Volver a correr los casos con falla para asegurarse que la modificaciòn corrige el sìntoma observado. Correr casos de pruebas adicionales para incrementar su confianza en la modificaciòn. Volver a correr los casos de prueba que antes tuvieron èxito para estar seguro que la modificaciòn no creò nuevos problemas. Si se tiene èxito, la copia de respaldo corregida se convierte en la versiòn principal del còdigo y la copia antigua se borra. En caso contrario se regresa al primer paso.

La depuraciòn por deducciòn procede listando las posibles causas de las fallas observadas, usando la informaciòn disponible para eliminar varias hipòtesis, elaborando las hipòtesis restantes, probando o rechazando cada hipòtesis, determinando las correcciones apropiadas y verificando las correcciones.

La depuraciòn por encadenamiento hacia atràs implica que el còdigo fuente se recorre hacia atràs desde el punto donde se observò el error, en un intento por identificar el punto exacto en que éste ocurriò. Puede requerirse correr casos de prueba adicionales para reunir màs informaciòn.

Las técnicas de depuraciòn tradicionales utilizan proposiciones de salida de diagnòstico, vaciados instantàneos, rastreos selectivos sobre valores de datos y flujos de control, y puntos de control dependientes de las instrucciones.

Las proposiciones de salida de diagnóstico se pueden intercalar en el còdigo fuente como proposiciones de comentarios con un formato especial, que se activa mediante una opciòn especial del traductor. La salida de diagnòstico de esas proposiciones brinda "fotos instantàneas" de componentes seleccionados del estado del

programa, a partir de las cuàles el programador trata de inferir el comportamiento del programa.

Un vaciado instantàneo es una representaciòn al nivel de la màquina del estado parcial o total de un programa en un punto particular en la secuencia de ejecuciòn.

Una facilidad del rastreo es listar los cambios en componentes selectos del estado. En su forma màs simple un rastreo imprimirà todos los cambios en los valores de los datos para todas las variables y todos los cambios en el flujo del control. Un rastreo selectivo rastrearà variables especificas y el flujo de control en regiones especificas del texto fuente.

Una facilidad del punto de control dependiente de la instrucciòn es que interrumpa la ejecuciòn del programa y transfiere el control a la terminal del programador cuando la ejecuciòn alcanza una instrucciòn de control especificada en el còdigo fuente.

8.6 Pruebas del sistema

Las pruebas del sistema implican dos clases de actividades: pruebas de integraciòn y aceptaciòn.

La integraciòn ascendente es la estrategia tradicional para integrar los componentes de un sistema de programas en un todo funcionando. La integraciòn ascendente consiste en pruebas de unidad, seguidas por pruebas de subsistemas y luego por pruebas del sistema completo. Las primeras tienen el objetivo de descubrir errores en los mòdulos individuales del sistema. Estos mòdulos se prueban aislados unos de otros en un ambiente artificial llamado "prueba dirigida", la prueba dirigida està formada por los programas conductores y los datos necesarios para ejercitar los mòdulos.

Un subsistema consta de varios mòdulos que se comunican unos con otros atravès de interfaces bien definidas. El propòsito principal de las pruebas de subsistemas es verificar la operaciòn de las interfaces entre mòdulos en el subsistema. Se deben probar tanto las interfaces de control como las de datos .

En la mayor parte de los sistemas de programaciòn, no es factible probar exhaustivamente los sistemas debido a la complejidad de combinaciones de las interfaces de los mòdulos; se deben elegir con cuidado los casos de prueba para ejercitar las interfaces en la manera adecuada.

Las pruebas del sistema se relacionan con sutilezas en las interfaces, la lògica de decisiòn, el flujo de control, los procedimientos de recuperaciòn, la eficiencia global, la capacidad

y las características de tiempo del sistema entero. Se requiere una escrupulosa planeación de las pruebas para determinar la extensión y la naturaleza de las pruebas del sistema que se va a realizar y establecer los criterios para la evaluación de resultados.

Las pruebas ascendentes tienen la desventaja de que necesitan escribir y depurar pruebas dirigidas para los módulos y subsistemas, además el nivel de complejidad adquirido al combinar módulos y subsistemas en unidades cada vez más grandes.

Las puebas dirigidas proporcionan ambientes de datos y secuencias de llamados para rutinas y los subsistemas que se estén probando por separado. La preparación de las pruebas dirigidas pueden ser 50% o más del esfuerzo de codificación y depuración para un producto de programación.

La integración descendente empieza con la rutina principal y una o dos rutinas inmediatamente subordinadas en la estructura del sistema. Después de que este "esqueleto" de alto nivel ha sido probado con detenimiento, se convierte en la prueba dirigida para sus subrutinas inmediatamente subordinadas.

La integración de alto nivel requiere el uso de troncos de programa para simular el efecto de las rutinas de más bajo nivel que son llamadas por las rutinas en prueba.

Esta integración ofrece varias ventajas:

- La integración del sistema se distribuye en toda la fase de implantación. Los módulos se integran a medida que se desarrollan.
- Las interfaces de nivel más alto se prueban primero y con más frecuencia .
- Las rutinas del nivel más alto proporcionan una prueba dirigida natural para las rutinas de los niveles inferiores.
- Los errores se localizan en los módulos e interfaces que se estén añadiendo.

Aunque pudiera parecer que la integración descendente siempre es preferible, se presentan muchas ocasiones en las que es imposible apegarse a una estrategia de codificación e integración estrictamente descendente. También puede costar muy caro correr el sistema en desarrollo como una prueba dirigida para las nuevas rutinas; puede no ser costeable religar y volver a ejecutar un sistema de 50 a 100 rutinas cada vez que se agrega una nueva rutina.

A menudo se puede ahorrar tiempo en forma significativa si los subsistemas se prueban por separado antes de insertarlos en la estructura descendente en desarrollo. Se puede necesitar probar primero ciertos módulos críticos de bajo nivel. En esa situación

se debe preferir la estrategia de prueba del emparedado.

La integración por emparedado es predominantemente descendente, pero las técnicas ascendentes se usan en algunos módulos y subsistemas. Esta mezcla mitiga muchos de los problemas encontrados en las pruebas descendentes puras y además retiene las ventajas de la integración descendente al nivel de subsistemas y del sistema.

Las pruebas de aceptación son parecidas a las pruebas de unidad, la diferencia entre estas, es que las pruebas de unidad las hace cada programador en la unidad desarrollada por este, y en las pruebas de aceptación son realizadas por el grupo de control de calidad y el cliente sobre el sistema completo.

Las pruebas de aceptación implican la planeación y la ejecución de pruebas funcionales, de desempeño y de tensión para demostrar que el sistema implantado satisface sus requisitos.

Además de las pruebas funcionales y de desempeño, las pruebas de tensión se llevan a cabo con el fin de establecer las limitaciones del sistema.

Por lo común, las pruebas de aceptación incorporan casos de prueba desarrollados durante las pruebas de unidad. Se añaden casos de prueba adicionales para lograr el nivel deseado de las pruebas funcional, de desempeño y de tensión del sistema completo.

8.7 Verificacion formal

La verificación formal implica el uso de técnicas matemáticas rigurosas para demostrar que los programas de computadora tienen ciertas propiedades deseadas. Por lo general, los métodos de afirmaciones de entrada y salida, de precondiciones más débiles y de inducción estructural son las tres técnicas más empleadas.

Al usar afirmaciones de entrada y salida, se asocian predicados (afirmaciones) con el punto de entrada, de salida y con varios puntos intermedios en el código fuente. Los predicados o condiciones de verificación, deben ser verdaderos todas las veces que se ejecute el código fuente asociado. La notación (P)S(R) significa que si el predicado P es verdadero antes de ejecutar el segmento de código S, entonces el predicado R será verdadero después de la ejecución de S.

La regla de composición de la lógica permite que se formen conjunciones de predicados junto con rutas de ejecución particulares: (P)S1, (Q) y (Q)S2(R) implica (P)S1; S2(R)

La regla de composición permite la siguiente proposición: si

todos los predicados intermedios son verdaderos a lo largo de una ruta particular de ejecución, entonces la afirmación de la salida (predicado de salida) será verdadera para esa ruta de ejecución.

El método de las afirmaciones de entrada y salida establece que si la conjunción de predicados desde la afirmación de entrada hasta una de salida es verdadera, y si la afirmación de entrada se satisface por las condiciones de la entrada, y si el programa termina después de seguir la ruta de ejecución de interès, entonces la afirmación de salida será verdadera al finalizar el programa.

La terminación de un ciclo se prueba demostrando que la secuencia de ejecución para cada ciclo decrece (o crece) monótonamente probando una propiedad no negativa (o negativa) en cada pasada a través del ciclo. Debido a las características de ciclo, esta propiedad debe en algún momento alcanzar un límite inferior (o superior) y la ejecución del ciclo concluirá.

se debe demostrar que los predicados de los ciclos son relaciones invariantes; esto es , el invariante de un ciclo debe ser verdadero independientemente del número de veces que éste se recorra. En particular, el invariante de un ciclo debe satisfacer las siguientes condiciones:

-Debe ser verdadero a la entrada al ciclo.

-Debe ser verdadero independientemente del número de recorridos de ciclo.

-Debe implicar la condición deseada a la salida del ciclo.

Mediante el método de los predicados más débiles se puede demostrar que el invariante de un ciclo es verdadero independientemente del número de recorridos del ciclo. Dada una proposición de la forma (P)S(R), P es la precondición más débil para S si es la condición más débil que garantizará la verdad de R después de la ejecución de S. La precondición más débil (wp) se expresa como:

$$P = wp(S,R)$$

En la práctica P se encuentra yendo hacia atrás desde R. Si S es una proposición de asignación de la forma X:=E, la precondición más débil P se obtiene al sustituir la expresión E por X en donde X aparezca en el predicado R:

$$wp(X := E,R) = R(E-->X)$$

Existen tanto limitaciones prácticas como teóricas a la verificación formal mediante el método de las afirmaciones de entrada y salida. Por ejemplo , Qué pasa si en S hay un sobreflujo durante la iteración del ciclo? Que ocurre si un elemento no está

dentro de los límites? Que acontece si algún elemento no está inicializado? Se pueden escribir manejadores de excepciones y especificar precondiciones adicionales para manipular tales situaciones.

Mientras que el tratamiento de las condiciones de sobreflujo y otras excepciones se podrían incorporar a esta prueba, el problema de esta discusión es el aspecto de las especificaciones incompletas. Qué se olvidó considerar? No hay forma algorítmica de asegurar que todo se ha previsto. Por lo tanto es correcto decir que se han verificado ciertas propiedades del código fuente, bajo ciertas suposiciones, en vez de aseverar que se ha verificado el código fuente para su operación bajo todas las condiciones. En general, la verificación formal sólo puede reflejarse en las precondiciones y las afirmaciones utilizadas en el proceso de verificación.

Otra limitación práctica de la verificación formal es la cantidad de esfuerzo requerido para verificar un programa de, por ejemplo, 500 líneas. El esfuerzo de verificación a menudo requiere más líneas de prueba que líneas de código con la posibilidad de introducir errores en la prueba. Esta posibilidad disminuye mediante herramientas de verificación automatizadas. Mientras que el proceso de verificación nunca puede automatizarse en su totalidad, una ayuda automatizada puede reducir el tedio y el márgen de error existentes en la verificación formal.

La limitación teórica fundamental de las técnicas automatizadas para la verificación formal implica la obtención de los invariantes de los ciclos. La obtención algorítmica de invariantes de los ciclos para los programas arbitrarios es un problema no resuelto. Hay programas para los que se puede lograr automáticamente los invariantes de los ciclos, pero también hay algunos para los que fallará la obtención automática de los invariantes de los ciclos.

La inducción estructural es una técnica de verificación formal basada en el principio general de la inducción matemática; La inducción se debe realizar sobre un conjunto parcialmente ordenado que está bien fundado. Suponiendo que el conjunto S tiene las propiedades necesarias y se demostrará una proposición P, la inducción matemática procede como sigue:

-Demostrar que P es verdadera para el o los elementos mínimos en S.

-Suponer que P es verdadera para cada elemento en S que tiene un número ordinal menor o igual a N y demostrar que P es verdadera para el elemento N+primero en S.

8.8 Aplicacion al caso practico

En el desarrollo del sistema de cuentas por pagar se realizó y se llevó a cabo el plan de control de calidad, un plan de prueba de aceptación y un plan de verificación del sistema.

En el plan de control de calidad se realizaron las siquientes actividades: Se hicieron revisiones de las especificaciones de requisitos contra las necesidades del usuario; la documentación del diseño contra las especificaciones de los requisitos; y el código fuente contra la documentación del diseño y las especificaciones de requisitos.

También se llevaron a cabo auditorías durante la evolución del sistema para verificar que sean consistentes y que estuvieran completos, . estàs fueron: las especificaciones de equipo,de programas y de comunicación con el usuario; asimismo el diseño interno contra las especificaciones funcionales; el código fuente contra documentación del diseño; y requisitos funcionales contra descripciones de las pruebas.

En el plan de pruebas de aceptación se realizaron las siquientes pruebas:

-Verificación de que no haya problemas al entrar o salir de módulos.

-Cargar, listar y modificar proveedores de materias primas y materiales no productivos o servicios.

-Se tratò de cargar proveedores ya existentes y listar o modificar a proveedores no existentes.

-Cargar, listar y modificar órdenes de compras de materias primas o materiales no productivos o contratos de servicios

-Cargar órdenes de compras ya existentes y listar o modificar órdenes de compras no existentes.

-Autorizar órdenes de compras en el orden establecido o en un orden no establecido.

-Cancelar órdenes de compras no autorizadas.

-Cancelar órdenes de compras ya autorizadas.

-Cargar, listar y modificar las entradas o recepciones del almacen.

-Tratar de cancelar las entradas o recepciones del almacen.

-Cargar, listar y modificar las facturas de proveedores de acuerdo a las entradas o recepciones efectuadas por el almacén.

-Verificar que la afectaciones de cargo y abonos a la carteras de proveedores sean realizadas correctamente.

-Checar que los cheques se realizen correctamente.

-Checar que el informe de saldos de proveedores a nivel reporte y consultas sean correctos.

-Cargar muchas órdenes de compras, autorizar estas órdenes, después cargar las entradas o recepción al almacèn, y capturar las facturas de los proveedores y que salgan los cheques con la afectación de cargos y abonos a la cartera de proveedores para probar el sistema en un ambiente de trabajo normal.

CAPITULO 9 Mantenimiento del sistema

Podríamos definir el término mantenimiento de sistemas como las actividades de la ingeniería de programacion que se desarrollan después de entregar un producto al usuario.

La fase de mantenimiento, es el periodo en el que un producto desempeña un trabajo útil, implica la mejora de los productos de software, adaptarlos a nuevas necesidades, y la corrección de problemas existentes dentro del mismo.Mejorar los productos de software da como resultado mejor funcionalidad en su operación y aunado a ésto optimizar la interacción con el usuario.

La adaptación de un producto a nuevas necesidades puede ser dado tal vez por el traslado del sistema a otro equipo o adaptarlo a ciertos requerimientos por una nueva funcionalidad del sistema (transmisión de datos vía telefónica, intercambio de información con otro equipo, etc.).

La corrección de problemas implica modificar y reevaluar los programas para subsanar los errores, determinando cuales requieren atención inmediata y cuales se corregiran de acuerdo a calendarizaciones.

Como se mencionó la fase de mantenimiento está vigente durante el ciclo de vida útil del producto, inclusive el tiempo que se emplea para darlo es más largo que el que se utiliza para el desarrollo del sistema, por lo que se concluye que las actividades de mantenimiento consumen gran parte del presupuesto total del ciclo de vida de los productos.

Los atributos que deben considerarse en el desarrollo de sistemas para que contribuyan a su mantenimiento son la claridad, la modularidad, y la correcta documentación interna del código fuente, además de documentos de apoyo apropiados.

9.1 Prevenciones a futuro para un mejor mantenimiento durante el desarrollo de los sistemas.

El mejorar o adaptar el sistema reinician el desarrollo en la fase de análisis, mientras que la corrección de algun problema del sistema puede reiniciar el ciclo de desarrollo en la fase de análisis,en la de diseño, o en la implantación. Así pues, todas las herramientas y técnicas utilizadas para desarrollar el sistema son potencialmente útiles para el mantenimiento del mismo.

9.1.1 Durante el analisis.

En esta fase del desarrollo de sistemas se determinan los requisitos y restricciones del usuario, asi como la factibilidad del sistema.

Las actividades más importantes que se deben de considerar en la fase de anàlisis son:

- Definir normas para los aspectos generales del producto, garantizando de esta manera la uniformidad del mismo, esto es por ejemplo, normalizar los formatos para la documentaciòn del sistema, definiciones para la codificaciòn estructurada, los principios de operaciòn, etc.

- Especificar procedimientos de control de calidad para garantizar que se elaboren documentos de alta calidad.

- Identificar las posibles mejoras al producto despuès de su liberaciòn inicial.

- Estimar recursos humanos y tècnicos para poder efectuar las actividades de mantenimiento.

9.1.2 Durante el diseno.

La fase de diseño estructural se relaciona con el desarrollo de todos los componentes funcionales, la conceptualizaciòn en la estructura de los datos y las interconexiones de los mòdulos internos y externos del sistema.

Por esta razòn, es importante durante el diseño estructural de recalcar la claridad, la modularidad y la facilidad de modificaciòn como los principales criterios de diseño para producir un sistema màs fàcil de darle mantenimiento.

El diseño detallado va relacionado con la especificaciòn de los detalles , las interfaces entre las rutinas y las estructuras de datos. Es aquí donde deben de utilizarse las anotaciones normalizadas para describir los algoritmos, las interfaces y estructuras de datos, referencias cruzadas que indique el (los) efecto(s) de cada rutina, de esta manera se proporcionarà un directorio con informaciòn necesaria para determinar que rutinas y que estructura de datos se afectarian con las modificaciones a otras rutinas.

9.1.3 Durante la implantacion

En la implantaciòn, al igual que el diseño, debe de tener como objetivo principal el producir un sistema de comprensiòn y

modificaciòn sencillas.
La facilidad en el mantenimiento se mejora mediante el uso de constantes simbòlicas que parametricen el sistema.

Aqui debemos de considerar los prólogos normales de cada rutina donde se proporcione el nombre del autor,la fecha de desarrollo, el nombre del programador de mantenimiento, asi como fecha y proposito de cada modificaciòn.

9.2 Aspectos administrativos para el mentenimiento

Para obtener èxito en el mantenimiento del sistema, como todas las actividades de la ingenieria de software, es indispensable una combinaciòn de habilidades administrativas y de pericia tècnica.

La actividad de mantenimiento para un sistema puede derivarse como respuesta a una solicitud de moficaciòn por parte del usuario.

Las solicitudes por lo general son efectuadas por los usuarios. Una solicitud de cambios puede traer como consecuencia mejoras, adaptaciòn o bien correcciòn de errores.

Las mejoras y adaptaciones de gran impacto pueden requerir un anàlisis exhaustivo y tal vez negociaciòn con el usuario; inclusive pueden llegar a ser manejados como nuevos proyectos de desarrollo y no como una simple actividad de mantenimiento.

Cuando se efectùa una solicitud de modificaciòn en primera instancia es revisada por un analista, en ocasiones existen problemas del usuario no son causados por el sistema, en esta situaciòn el analista lo notifica al usuario, y con el cierra dicha solicitud de modificaciòn.

De no ser asi, el analista somete a la junta de control de cambios la solución propuesta con una estimación de recursos para satisfacer la solicitud.

9.2.1 La junta de control de cambios

La junta de control de cambios es un comitè que revisa y aprueba todas las solicitudes de modificaciòn. Tiene la facultad de rechazar una solicitud, e inclusive reconsiderar una versiòn modificada del cambio o aprobarlo sin alteraciòn.

El analista es la interfase entre la junta de control de cambios, y el usuario, cuando una solicitud es aprobada, los cambios se envian a los programadores de mantenimiento, quienes de acuerdo a las prioridades que fueron establecidas por la junta de control, actuan para llevar a cabo la modificaciòn y revalidaciòn; pos-

teriormente la someteran a la junta para su aprobación.

De no ser aprobadas las modificaciones de los programadores de mantenimiento, estos y el analista, atenderán las observaciones efectuadas por la junta; después de su actualización se someterá nuevamente a la aprobación de la junta.

La conformación de la junta de control de cambios, y la forma en que funcione, depende de la naturaleza del sistema, así como la estructura organizacional. Es importante que dicha junta conste de miembros relacionados con: responsables del sistema (usuario),representante de control de calidad, representante del área de mantenimiento de programas, etc.

De alguna manera, un aspecto importante en esta estructura es proteger a los programadores de mantenimiento de la constante interrupción por parte de los usuarios al tener relación directa con ellos.

Como se mencionó anteriormente el analista que atiende el problema realiza un enlace con el usuario y proporciona la comunicación entre los usuarios, la junta y los programadores de mantenimiento.

La importancia de la existencia de dicha junta es que considerará muchos factores que los programadores de mantenimiento pueden pasar por alto al establecer prioridades y restricciones de las actividades de mantenimiento.

9.2.2 El servicio de mantenimiento al sistema

El mantenimiento del sistema puede realizarlo el grupo que efectuó el desarrollo o miembros ajenos al mismo.

Si el mantenimiento es realizado por los miembros que desarrollarón el sistema, estos estarán familiarizados con el mismo, y será más fácil la identificación de los módulos a modificar.

Así también, si los miembros del grupo de desarrollo saben que serán los responsables del mantenimiento del sistema, es probable que sean cuidadosos desde el diseño inicial del producto para facilitar su mantenimiento, de otra forma, quizás serán menos minuciosos al preparar la documentación de apoyo.

El mantenimiento llevado a cabo por un grupo ajeno al desarrollo exige alguna manera más atención a las convenciones y a la documentación de alta calidad. La ventaja sería la liberación del grupo de desarrollo para atender nuevas actividades.

Un mètodo recomendable para organizar la programaciòn de mantenimiento con objeto de evitar el estigma de ser "programador de mantenimiento" es la rotaciòn con periodicidad de los programadores entre desarrollo y mantenimiento.

Las principales ventajas a obtener serian:

Los programadores novatos aprenderàn las habilidades de los experimentados, el nivel general del personal se elevarà ya que todos conoceràn un poco màs a profundidad los sistemas existentes, la rotaciòn del personal de desarrollo en las tareas de mantenimiento le la necesidad de un còdigo de alta calidad y una documentaciòn màs clara, obteniéndose màs flexibilidad en el personal, lo brinda tambièn la importancia de no depender de un sòlo individuo para el mantenimiento de un sistema en particular.

9.3 Herramientas de apoyo automatizadas para una mejor administracion del mantenimiento

Durante el periodo de mantenimiento del sistema, es indispensable elaborar un plan en donde se considere una adecuada administraciòn de las actividades de mantenimiento efectuadas o a efectuar al sistema, con objeto de contar con un registro de todas las actividades y controlar las distintas versiones que surjan durante el mismo.

El control de las diferentes versiones de programas de un sistema son un aspecto significativo en su mantenimiento.

En algunos estudios se ha demostrado que los grandes productos de software tienden a evolucionar dentro de familias de versiones, en donde cada versiòn es similar, pero diferente a otros miembros de la familia. Durante los estudios de dinámica de grandes sistemas se desarrollaron cinco leyes de la evoluciòn de los programas que a continuaciòn se presentan:

- Cambio continuo, un programa pasa por un cambio continuo o llega a ser progresivamente menos útil. El proceso de cambio continùa hasta que llega a ser costeable para reemplazar el programa con una versiòn creada de nueva cuenta.

- Complejidad creciente, a medida que se modifica un programa que evoluciona, su complejidad que refleja una estructura deteriorada, se incrementa, a menos que se trabaje en èl para reducirla.

- La ley fundamental de la evoluciòn del programa, la evolución de un programa està sujeta a una dinàmica que constituye el proceso de programaciòn y, por tanto, las mediciones del proyecto global y los atributos del sistema, se

autorregulan con tendencias e invariaciones determinadas estadísticamente.

- Conservación de la estabilidad de la organización, la tasa de actividad global en un proyecto que apoya un programa que evoluciona, es estadísticamente invariante.

- Conservación de la familiaridad, el contenido liberado (cambios, adiciones, supresiones) de las liberaciones sucesivas de un programa que evoluciona es estadísticamente invariante.

Dentro de las herramientas automatizadas para el apoyo del mantenimiento del sistema están los de soporte técnico y las de soporte administrativo.

En las de aspecto técnico se incluyen los editores de texto, los generadores de referencia cruzada, editores de enlace, los comparadores, calculadores métricos de complejidad, la configuración de bases de datos administrativas

Los editores de textos permiten una modificación rápida de los programas fuente y de los documentos de apoyo (manuales).

Los generadores de referencia cruzada brindan directorios de referencias cruzadas con la estructura de llamadas de quien llama a quien y en donde.

El editor de enlace efectúa la liga de los módulos objeto del código compilado para producir un programa ejecutable, con esta facilidad se puede configurar un sistema de varias maneras, o para ligar de modo selectivo los módulos recompilados dentro de un sistema.

Un comparador conciliará dos archivos de información e informará de las diferencias, ésto es, el comparador se puede usar durante el mantenimiento para cotejar las versiones de un programa fuente, dos versiones de documentos de apoyo, etc.

Los calculadores métricos de complejidad nos ayudan a medir el grado de dificultad en la estructura de nuestros programas, estos calculadores métricos pueden ser utilizados antes y después de efectuar una modificación, con el fin de determinar el incremento de complejidad a consecuencia de la actualización del programa.

Para los aspectos administrativos de mantenimiento es posible utilizar las bases de datos que nos puede proporcionar información concerniente a la estructura del sistema, el número de revisión en curso, la historia de solicitudes, etc.

CONCLUSIONES

Dado que el objetivo de esta tesis fue desarrollar e instalar un producto de programación útil en el control de los insumos productivos e insumos operativos de una empresa grande dedicada a la elaboración y comercialización de lubricantes, podemos destacar los siguientes puntos:

Después de analizar el medio ambiente natural en el cual se encontraba la compañía, se determinó que efectivamente había la necesidad de mejorar el control sobre aquellos insumos necesarios para la misma, puesto que había un descontrol sobre las compras que impactaba el flujo de efectivo de la compañía generando insatisfacción a los proveedores, ya que los pagos eran atrasados; éstos tuvieran renuencia a vender sus productos, y hubiera faltantes en producción.

Se analizaron ampliamente todos los procesos administrativos e informáticos de los departamentos que tuvieran relación con el problema, y se desarrolló el proyecto que ha sido explicado en detalle durante la tesis, obteniéndose los siguientes resultados:

Se disminuyó el trabajo manual del área de compras ya que se duplicaban las órdenes de compra al momento de ser requeridas, en forma manual y de captura, eliminando la manual y utilizando la captura en el computador para generar las órdenes de compra.

Se facilitó la revisión jerárquica de las mismas, y las autorizaciones de las compras fueron remitidas, según su monto, a las personas adecuadas.

Se establecieron las ligas dentro del sistema de programación para lograr un flujo de efectivo adecuado con el departamento de cuentas por pagar, valuando el total de las órdenes de compra, por período específico (mensual, semestral anual).

Se establecieron controles a los proveedores como la historia de ellos en cuanto a rechazos y calidad de sus productos, así como de sus precios.

Se emitieron los cheques automáticamente, generando las afectaciones contables necesarias para su correcto registro.

Se mejoró el ambiente de trabajo de la compañía, puesto que se eliminaron muchas de las fricciones que se generaban por la lentitud del proceso.

Se mejoró la imágen corporativa de la compañía, puesto que los proveedores encuentran respuesta a las condiciones pactadas de pago.

Con lo anterior se establece que la metodología utilizada para el desarrollo del sistema es adecuada, sobre todo, porque cimentan las bases de un avance lógico del producto de programación, tomando en cuenta tanto las necesidades de los usuarios y del personal técnico en sistemas que lo realiza.

Asimismo, se demuestra que fundamentar el análisis, programación y mantenimiento de los sistemas permite lograr un mejor resultado que incluso permitirá a las personas dedicadas a asesorar a compañías o desarrollar productos de programación a lograr sus objetivos minimizando errores y optimizando recursos.

BIBLIOGRAFIA

A Computer Approach to Decision Models
Clause McMillan Richard F. González
Richar D. Irwin Inc
Homewood, Illinois, E.U.A 1976

Design Real Time Computer Systems
James Martin Prentice Hall Inc
Englewood Cliffs, New Jersey, U.S.A 1980

Ingeniería de Software
Richard E. Fairley
School of Information Technology Wang Institute of Graduate Studies. Tyngboro, Massachusetts 1985
Traducción: Antonio Sánchez Aguilar Doctor en Ciencias
Universidad de las Américas Mc Graw Hill 1987

Método General de Análisis de una Aplicación Informática
X. Castellani
Manuales de Informática Masson
Traducción Juan José Castro Botella
Masson S.A. París, Francia 1986

Planeación, Programación y Control Computarizado en las Empresas
P. Meneses Moguel
Editorial Limusa S.A. de C.V.
México 1984

Software Engineering: A Practitioner's Approach
Roger S. Pressaman
International Student Edition
Mc Graw Hill 1984

Software Engineering Economics
Boehm B.
Prentice Hall, Englewood Cliff. N.J. 1981

Structured Systems Analysis:Tools and Techniques
Gane and Sorson
Prentice Hall

Structured Design
Yourdon Edward
Prentice Hall

Systems Analysis and Design
Award M. Elias
Richard D. Irwing Inc.

Tools and Techniques for Structured
Systems Analysis and Design
Willian S. Davis