El diseño de software aplicado a los sistemas de información

Autor: Jesús Manzo Chávez

Tesina presentada para obtener el título de:
Lic. En Sistemas computarizados [sic]

Nombre del asesor:
Sergio Francisco Barraza Ibarra

UNIVERSIDAD VASCO DE QUIROGA

ESCUELA DE SISTEMAS COMPUTARIZADOS

"EL DISEÑO DE SOFTWARE APLICADO A LOS SISTEMAS DE INFORMACIÓN"

TESINA

Que para obtener el Título de:
LICENCIADO EN SISTEMAS COMPUTARIZADOS

Presenta:
JESÚS MANZO CHÁVEZ

Asesor:
ING. Y M.A. SERGIO FRANCISCO BARRAZA IBARRA

CLAVE 16PSU0014Q
ACUERDO 952006

MAYO 1999
Morelia, Mich.

DEDICATORIAS

A DIOS:

Por obsequiarme el don de la vida
y darme más de lo que merezco.

A MIS PADRES:

Ma. del Carmen y Jesús, que con su paciencia,
esfuerzo y dedicación debo todo lo que soy.

A MIS HERMANOS:

Ricardo y Carmen, por brindarme su apoyo y
compartir todo conmigo.

A MIS AMIGOS:

Marisa, Ramiro, Martín, Rocío y todos los demás,
por compartir parte de su vida y alegría conmigo.

A MIS COMPAÑEROS:

Gisela, Jenny, Julieta, Marcela, Jorge, Germán, Francisco, Daniel, por todo el apoyo, motivación y todos los momentos alegres que hicieron de mi carrera universitaria, la mejor de mis etapas educativas.

A M.B.:

Por tratarse de una persona muy especial para mí, la cual me ha brindado su incondicional cariño y confianza, y por ser una fuente de motivación para mí.

A MI ASESOR:

Ing. Y M.A. Sergio Barraza I. Por la ayuda que me brindó en la realización de esta Tesina.

A MIS MAESTROS:

Por transmitirme los conocimientos para poder alcanzar la excelencia.

A LA ESCUELA DE SISTEMAS COMPUTARIZADOS
Y LA UNIVERSIDAD VASCO DE QUIROGA.

A TODOS ELLOS: MUCHAS GRACIAS

"EL DISEÑO DE SOFTWARE APLICADO A LOS SISTEMAS DE INFORMACIÓN"

OBJETIVOS:

1.- INTRODUCCION.....5

2.- ANTECEDENTES.....8

3.- OBJETIVO GENERAL.....13
3.1.- OBJETIVOS ESPECIFICOS.....13

4.- ALCANCE DEL DISEÑO DE SOFTWARE.....14
4.1.-Diseño de la arquitectura del sistema.....14
4.2.- Diseño detallado de software.....14

5.- EL PROCESO DE DISEÑO.....15

6.- PRINCIPIOS BÁSICOS E IDEAS.....17

7.- DISEÑO ESTRUCTURADO.....19

8.- MACRO DISEÑO.....21
8.1.- Estudios de Factibilidad.....22
8.1.1. Factibilidad técnica.....22

8.1.2. Factibilidad económica.....23
8.1.3. Factibilidad operacional.....27
8.2.- Selección de alternativas de macro diseño.....28

9.- MICRO DISEÑO.....31

10.- MÉTODOS DE DISEÑO.....35
10.1.- Problemas de los métodos de diseño de software.
10.1.1.Diversidad.....35
10.1.2.- Clasificación.....36
10.1. 3.- Visibilidad.....36
10.1. 4.- Exactitud.....37

11.- CONTROLES DE SISTEMAS.....38
11.1.- Controles generales.....38
11.2.- Controles técnicos.....39
11.3.- Controles de usuarios.....40
11.4.- Controles de corrección de errores.....40
11.5.- Controles de investigación de sistemas.....41

12.- DESARROLLO DE SOFTWARE.....42

13.- TENDENCIAS ACTUALES EN EL DISEÑO
DE SOFTWARE.....45

14.- NOTACIONES USADAS EN EL DISEÑO
DE SOFTWARE.....47
14.1.- Diagramas de flujo de datos.....47
14.2.- Diagramas de estructura de datos.....48

14.3.- Diagramas de relaciones de entidades.....48
14.4.- Historias de vida de entidades.....49
14.5.- Entidad-relación-archivo.....50
14.6.- Flowcharts.....50
14.7.- Diagramas HIPO.....51
14.8.- Petri Nets.....52
14.9.- Especificación de programas.....53
14.9.1.- Pseudocódigos e inglés estructurado.....54
14.10.- Diagramas de transición de estados.....57

15.- PRINCIPALES MÉTODOS USADOS EN LA ACTUALIDAD.....59
15.1.- JSD.....59
15.2.- MASCOT.....61
15.3.- SDL.....63
15.4.- SSADM/LSDM.....64
15.5.- YOURDON.....67
15.5.1.- Modelos de viabilidad de estudio.....68
15.5.2.- Modelos del sistema.....68

16.- MÉTODOS EMERGENTES.....70
16.1.- Diseño orientado a objetos.....70
16.2.- Prototyping.....71

17.- LISTA DE DISEÑO.....74
17.1.- Restricciones.....74
17.2.- Expectación del cliente.....74
17.3.- Tipo de sistema.....75
17.4.- Tipo de aplicación.....75

17.5..- Ambiente del proyecto.....75
17.6.- Vista completa del ciclo de vida.....76
17.7.- Requerimientos no funcionales.....76

CONCLUSIONES.....77
GLOSARIO DE TERMINOS.....78
BIBLIOGRAFÍA.....82

1.- INTRODUCCIÓN

El diseño es el primer paso de la fase de desarrollo de cualquier producto o sistema de información. Puede definirse como el proceso de aplicar distintas técnicas y principios con el propósito de definir un dispositivo, proceso o sistema con los suficientes detalles como para permitir su realización física.

El objetivo del diseñador es producir un modelo o representación de una entidad que se construirá más adelante.

El proceso por el cual se desarrolla el modelo combina: la intuición y los criterios en base a la experiencia de construir entidades similares, un conjunto de principios y/o heurísticas que guían la forma en la que se desarrolla el modelo, un conjunto de criterios que permiten discernir sobre la calidad y un proceso de iteración que conduce finalmente a una representación del diseño final.

El presente trabajo esta dividido en 17 apartados, en el capítulo número cuatro describo los antecedentes históricos tanto de la evolución del hardware como las cuatro eras en la evolución del software.

En el capítulo número cinco hago referencia al proceso de diseño de software en donde se organiza un equipo de trabajo y actividades que deben de asignarse y realizarse para un buen desarrollo.

En el capítulo seis cito los diferentes factores que intervienen en la orientación para el desarrollo de sistemas de información.

En el capítulo numero siete hago referencia al inicio de las operaciones que se realizan al diseñar software ya que se tienen que realizar en pasos para obtener mejores resultados.

En el capítulo octavo mencionó las alternativas de macro diseño, como secuencias de alternativas que se deben de evaluar y elegir la más conveniente desde un nivel superior hasta una alternativa especifica o elegir una posible alternativa de un nivel inferior.

Así mismo dentro de este capítulo hago mención de los diferentes estudios de factibilidad que se deben de analizar para saber si el diseño de software es o no factible para llevarse a cabo. Como lo son los estudios de factibilidad técnica, económica y operacional.

En el capítulo número nueve hago mención de las diferentes actividades del micro diseño las cuales van desde la recolección de información necesaria, la planeación de actividades, adquisición de hardware y diseño de sistemas de entrada, de proceso y de salida.

Dentro del capítulo número diez del presente documento hago mención de los problemas más comunes existentes a la hora de adoptar un determinado método para el diseño de software.

El capítulo número once trata acerca de los diferentes controles que deben de aplicarse a lo largo de las diferentes etapas del micro diseño con el fin de disminuir los riesgos de fallos en el desarrollo de sistemas de información.

El capítulo número doce hago referencia las alternativas ha ser consideradas al decidir desarrollar un sistema.

En el apartado trece describo las tendencias actuales enfocadas a emplear diversas herramientas de apoyo para el diseño de software.

El capítulo catorce trata acerca de las diferentes notaciones más comunes usadas para el diseño de software.

En los capítulos quince y dieciseis tratan acerca de los diferentes métodos más usados en la actualidad para el diseñó de software.

El capítulo diecisiete trata sobre los diferentes aspectos que se deben de considerar al la hora del diseño.

Por último hago las conclusiones personales sobre el presente trabajo y citó la bibliografía empleado para la realización del mismo.

2.- ANTECEDENTES

¿Que importancia tiene el software respecto del hardware?. ¿Cuál de éstos dos es más importante en la actualidad?. Las respuestas a estas preguntas son realmente sencillas ya que el software es en la actualidad la pieza fundamental de la computación, es de mucho mayor importancia que el hardware ya que realmente es esté el que procesa y marca la pauta en el procesamiento de datos que es en lo que todas las organizaciones buscan tener los mejores resultados.

Actualmente el software ha superado al hardware como el punto clave para el proceso de información basado en computadora.

En una compañía en donde se manejan grandes volúmenes de información realmente es muy importante tener un control por medio de un software adecuado y a la medida de las necesidades del cliente para manejo eficaz de sus datos. Mostrando este factor clave que hasta puede inclusive marcar la diferencia de una compañía a otra.

Pero no siempre fue así, antes se le daba mayor importancia al hardware ya que los costos de estos componentes eran realmente elevados y las capacidades de procesamiento de estos variaban mucho al momento de procesar información.

Estos casos surgieron en las tres primeras décadas de la era informática, donde se dedicaba todo el tiempo y atención al desarrollo y diseño de hardware, tratando de reducir los costos de procesamiento y almacenamiento de datos.

En la década de los ochenta, los avances que ha habido en la rama de la microelectrónica han arrojado resultados muy satisfactorios en la potencial de cálculo y el ahorro de costos.

Ahora bien todo el potencial de las grandes computadoras de los años ochenta se han reducido a una computadora personal, que en la actualidad fácilmente se puede tener una en casa y con mayores resultados, ya que son mucho más rápidas, se pueden almacenar mayor cantidad de información y tienen un costo mucho más bajo.

Ha habido una gran transformación en lo que se refiere a emplear procesadores con válvulas al vacío a dispositivos microelectrónicos que procesan información a 500 millones de operaciones por segundo y más. De almacenar información en tarjetas perforadas, cintas magnéticas, discos que solo podían almacenar 320 Kb a unidades de disco duro de más de 11 Gb, discos compactos, unidades Zip y Jazz con gran capacidad de almacenamiento.

En lo que a software se refiere ha ido evolucionando a grandes pasos haciéndose cada vez más fácil de diseñar en comparación con las décadas anteriores, gracias las facilidades

que brindan las nuevas herramientas, interfaces y plataformas de la actualidad.

Ha habido cuatro eras en lo que ha evolución de software se refiere:

En la primera era del software comprendida entre 1950 y 1960 se aplicaban orientación por lotes, se tenía una distribución limitada del software haya que se creaba específicamente para una sola persona que por lo general el mismo hacía el software y lo empleaban tratando de satisfacer sus necesidades. Debido a este entorno personalizado del software, el diseño era un proceso implícito, realizado en la mente de alguien y la documentación de este por lo general nunca se realizaba.

En la segunda era en la evolución del software que comprende los mediados de los sesenta hasta finales de los setenta, la multiprogramación así como los sistemas multiusuarios introdujeron nuevos conceptos de interacción entre el hombre y la computadora, estas técnicas interactivas también llevaron a otro nivel la relación entre el hardware y el software.

Los sistemas de tiempo real eran muy utilizados haciendo tareas de procesamiento de datos en milisengundos en lugar de minutos. Esta era también presento al mundo al software como un producto que se podían distribuir y se dejó de ver solo como un componente más de la computadora.

La tercera era de la evolución de sistemas para computadora comprendida a mediados de los setenta hasta apenas hace algunos años, tienen como característica principal el empleo de redes de área local y de área global, así como el uso de comunicaciones digitales, y el aumento de necesidad por el acceso a datos, que dieron la pauta para llegar hasta nuestros días con sistemas de información cada vez más eficientes.

Esta era es también muy conocida por el alto empleo de microprocesadores y computadoras personales de uso cotidiano, por lo que el hardware de las computadoras se ha convertido en un producto estándar, mientras que el software que se suministra al hardware, es realmente lo que marca una diferencia.

Dando con esto una creciente demanda por productos de software y el desarrollo de los mismos.

La cuarta generación del software esta empezando, teniendo tendencias a las áreas de desarrollo de software orientado a objetos, sistemas expertos, redes neuronales artificiales, computación paralela y telecomunicaciones.

Sin embargo, la gran demanda existente en la actualidad a dado pauta a desarrollar sistemas bajos en calidad, además de no desarrollar software apropiado que realmente pueda explotar el potencial del hardware tan potente de hoy en día. La capacidad de desarrollar nuevo software no se da abasto con la demanda existente por obtener el mismo (lo cuál da como resultado que aparezcan en el mercado infinidad de programas Beta con muy baja

calidad), dando como resultado programas con un mal diseño y la falta de usar recursos adecuados para el desarrollo de software.

El control y administración cuidadosos de los proyectos son la clave para un desarrollo de sistemas con éxito, varias de las actividades de las fases de diseño y desarrollo pueden realizarse simultáneamente, lo cual acortará el tiempo de la investigación de sistemas.

El diseño de sistemas determina cómo un sistema logrará lo que tiene que lograr; involucra la configuración de los componentes de software y hardware de un sistema para después de su instalación, el sistema satisfaga completamente las especificaciones de sistemas establecidas al final de la fase de análisis de sistemas. Un aspecto posterior del diseño de sistemas es su configuración para que sea aceptable tanto para los usuarios como para los operadores de sistemas.

Si el sistema como está diseñado no puede lograr en forma simultánea las especificaciones establecidas y ser aceptado por los usuarios y operadores que posteriormente usaran el nuevo sistema; como algunas veces se descubre, las actividades de análisis de sistemas deben renovarse y modificarse las especificaciones de sistemas.

3.- OBJETIVO GENERAL

REALIZAR UN ANÁLISIS DE LAS DIFERENTES TÉCNICAS EXISTENTES PARA EL DISEÑO DE SOFTWARE APLICADO AL DESARROLLO DE LOS SISTEMAS DE INFORMACIÓN, QUE SE EMPLEAN EN LOS DISTINTOS TIPOS DE ORGANIZACIÓN.

3.1. OBJETIVOS ESPECIFICOS:

1.- IDENTIFICAR LAS TENDENCIAS DE HOY EN DIA, PARA EL DISEÑO DE SISTEMAS DE INFORMACIÓN.

2.- CONOCER EL ALCANCE DEL DISEÑO DEL SOFTWARE ASÍ COMO SUS PRINCIPIOS BÁSICOS.

3.- REALIZAR LOS ESTUDIOS CORRESPONDIENTES EN LAS TÉCNICAS NECESARIAS PARA EL ANÁLISIS DEL MACRO DISEÑO.

4.- ANALIZAR LAS TÉCNICAS ACTUALES DEL MICRO DISEÑO.

5.- RESALTAR LAS NOTACIONES MÁS UTILIZADAS EN EL DISEÑO DE SISTEMAS DE INFORMACIÓN.

6.- IDENTIFICAR CADA UNO DE LOS TIPOS DE CONTROLES EXISTENTES PARA EL DESARROLLO DE SOFTWARE.

7.- IDENTIFICAR LOS MÉTODOS EXISTENTES EN LA ACTUALIDAD PARA LA REALIZACIÓN DE UN DISEÑO DE SOFTWARE.

4.- ALCANCE DEL DISEÑO DE SOFTWARE

El diseño de software es la conversión de la afirmación de requerimientos del usuario y la realización de esos requerimientos en forma de código ejecutable. Esta definición cubre un rango muy amplio de actividades y es útil dividirlas en dos áreas principales: diseño de la arquitectura y diseño de software.

4.1 DISEÑO DE LA ARQUITECTURA DEL SISTEMA

Este es el proceso mediante el cual se produce una especificación completa y verificada del hardware general, arquitectura de software, estructura de datos, componentes e interfaces para el producto del software -. De igual forma debe de realizarse manuales del usuario y además de un bosquejo de los planes de prueba o cajas negras (ver glosario).

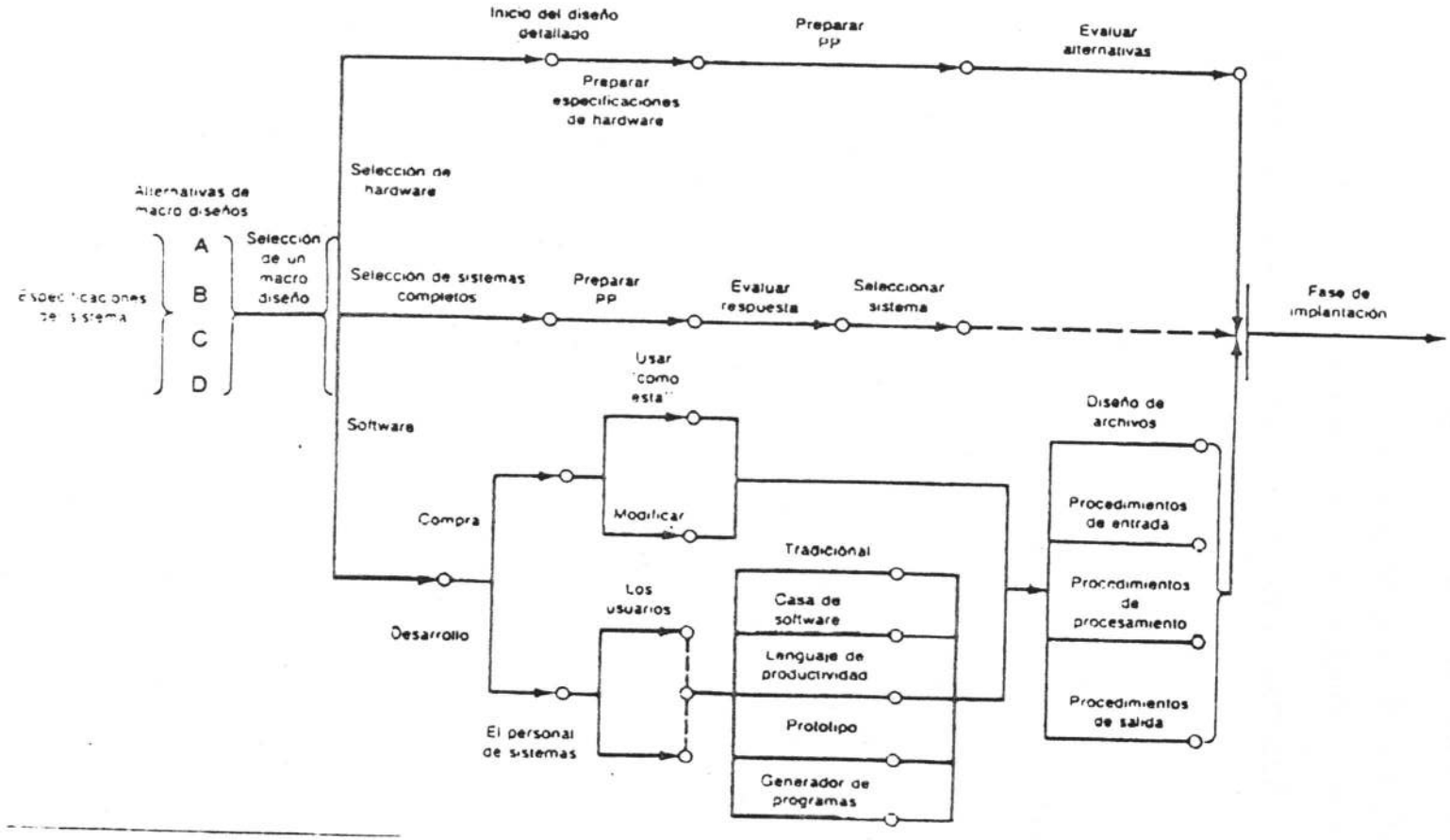
4.2.- DISEÑO DETALLADO DE SOFTWARE

Este ocurre cuando se producen especificaciones verificadas de estructura de datos, componentes e interfaces, dimensiones, algoritmos claves y suposiciones para cada componente de programa que conducen a la programación.

5.- EL PROCESO DE DISEÑO

Mientras que el análisis de sistemas es una actividad no estructurada que involucra estimulaciones y negociación, el diseño de sistemas es más estructurado y técnico por naturaleza. Los diseñadores de sistemas necesitan un alto nivel de habilidades técnicas, en tanto que los analistas de sistemas tienen mayor necesidad de capacidades interpersonales. Sin embargo, existe mucha interacción entre ambos equipos de diseño; por lo tanto, la capacidad de los diseñadores de trabajar entre sí (es tema de consideración). Durante la fase de diseño se realiza un número de actividades especializadas y un equipo de trabajo para un proyecto grande de diseño de sistemas podría consistir en programadores, diseñadores de archivos, especialistas de control de entradas, expertos en adquisición de hardware, especialistas en adquisición de software, en administración de proyectos, expertos en redes de telecomunicación, diseñadores de archivos y consultores especializados, aunque por lo general no todos estarían involucrados al mismo tiempo.

La figura 5.1 muestra las actividades que pueden ocurrir durante el diseño de sistemas. En la citada figura las líneas representan las actividades o alternativas, y el círculo representa sucesos o decisiones. Como se ve, el diseño de sistemas está basado en las especificaciones de sistemas; una serie de alternativas de macro diseño se desarrollan y analizan, y se elige una. Dependiendo de qué macro diseño se elija, el resto del diseño de sistemas se enfocaría a una de las siguientes: selección de hardware, de un sistema compuesto (hardware y software) o



ACTIVIDADES DEL PROCESO DE DISEÑO

Figura 5.1

diseño de software. Con frecuencia un proyecto combina diseño de software y selección de hardware. Un proyecto de selección de un sistema compuesto involucra la selección de un sistema listo para operar, el cual consiste en un sistema de cómputo y programas de cómputo proporcionados por un proveedor.

6.- PRINCIPIOS BASICOS E IDEAS

Existen diferentes factores que se deben tomar en cuenta cuando se va a decidir hacia donde enfocar el diseño de software, alguno de estos factores y sus razones se citan a continuación:

- Restricciones: Muchas veces el rango de elecciones que puede tomar un diseñador de sistemas es restringido por varias razones, algunas de estas son las contractuales que restringen los posibles métodos o procedimientos que se podrían tomar para la realización de software, u otras veces están restringidos por razones técnicas por limitaciones que el hardware y software existente limitan la libertad del diseñador desde un principio.

- Expectativas del cliente: El nivel de participación del cliente en un proyecto de software puede influir en la estrategia de diseño adoptado. Si el proceso del diseño es fácil de monitorear, el cliente puede contribuir, esto no siempre es recomendable, ni siempre es solicitado.

- Tipo de sistema: Según la naturaleza del sistema que se va ha desarrollar, se adopta un tipo de diseño especifico para facilitar el desarrollo del sistema y de igual forma se permite la elección de las herramientas adecuadas.

- Tipo de aplicación: Las implicaciones al momento de diseñar un sistema son muy variadas, ya que la demanda de un tipo de sistema especifico para una actividad difiere mucho de otro

dedicado a otra causa, llevando a hacer un tipo de proceso de diseño totalmente diferente.

- Ambiente del proyecto: El nivel de entrenamiento, experiencia y la infraestructura de soporte que tenga una organización, (tanto organizacional como computacional), influye notablemente a la orientación que se le quiera dar al diseño de software.

- Vista completa del ciclo de vida: Las decisiones del diseño del software debe tomar en cuenta que el sistema deberá ser útil a largo plazo, y que al poco tiempo de que se diseño se tenga que modificar por completo(o en su gran mayoría) o bien sea obsoleto y se tenga que desechar.

- Requerimientos no funcionales: Los diseñadores a menudo se concentran en lo que el sistema hace en lugar de que tan fácil es de usar, qué tan rápido operará, etc. Estos requerimientos no funcionales son los que convertirán crecientemente en el factor de diferenciación.

7.- DISEÑO ESTRUCTURADO

El diseño estructurado tiene sus orígenes en los primeros conceptos de diseño que consideraban la modularidad, el diseño descendente y la programación estructurada.

Sin embargo, en la actualidad estas técnicas procedimentales se han ampliado integrando ha ellas explícitamente el flujo de información en el proceso de diseño.

Esto quiere decir que utiliza las características del flujo de información para derivar la estructura del programa.

El diseño estructurado implica el inicio del diseño del sistema con el conjunto más amplio de diseños alternativos permitidos por el alcance establecido del proyecto (al que se refiere aquí como " alternativas de macro diseño ") y su continuación a través de una serie de conjuntos de alternativas progresivamente menores hasta que el conjunto menor (las alternativas de "macro diseño ") defina por completo al sistema en detalle. A este enfoque por estratos de iniciar con un macro diseño y proseguir a través de muchas etapas hasta un micro diseño final se le llama " refinamiento sucesivo " o " diseño de arriba a bajo ".

Con referencia a la figura 5.1 después de la actividad de selección de alternativas de macro diseño, los conceptos de diseño estructurado son muy relevantes para las actividades de desarrollo de software que se muestran en la parte inferior de la figura.

Usando diseño estructurado, se busca una estructura de sistemas clara y sencilla para el software que se mantiene y se puede modificar con facilidad.

Esto se logra en parte con el diseño modular ya que cada módulo realiza una tarea lógica y tiene tan pocas uniones con otros módulos de software como sea posible.

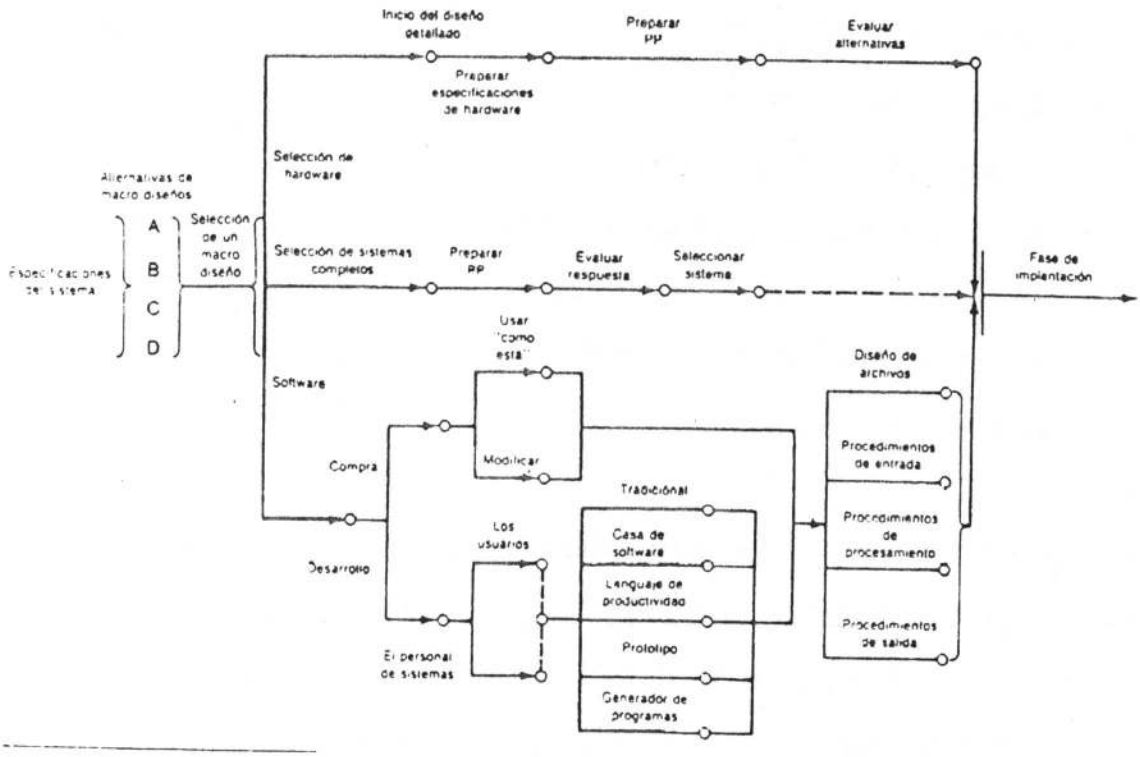
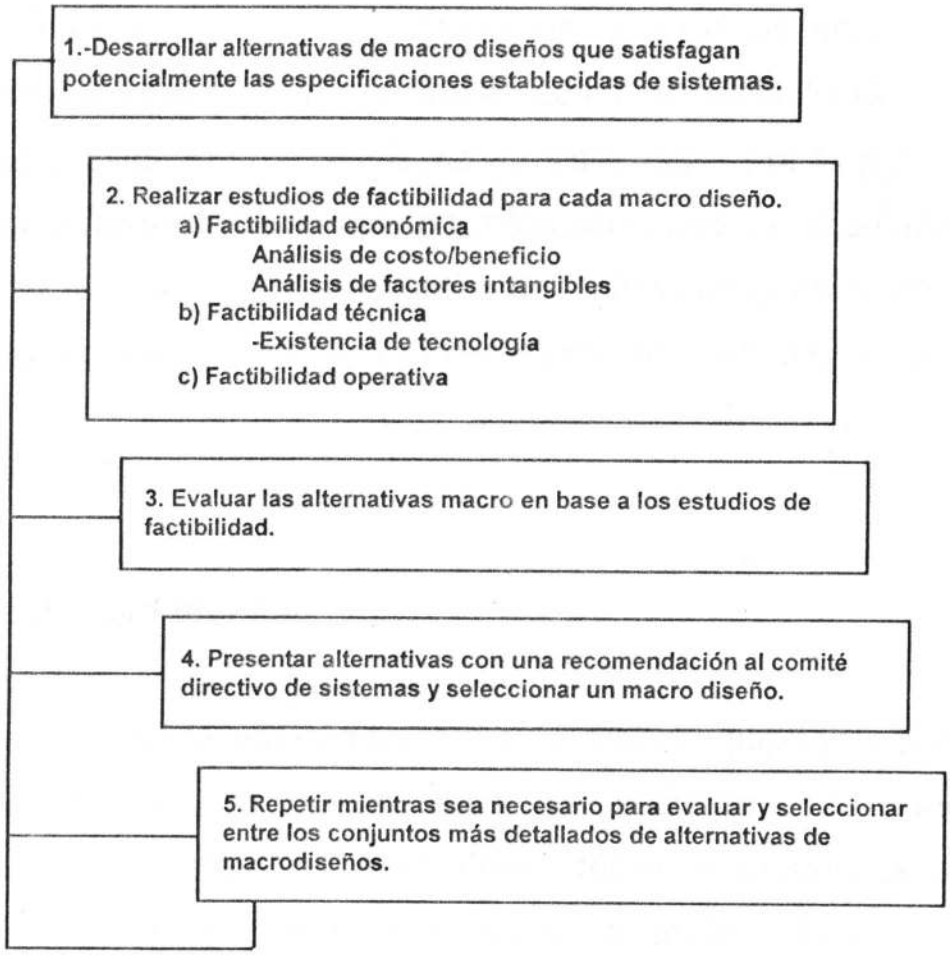


Figura 7.1

8.- MACRO DISEÑO

El diseño de sistemas puede ejemplificarse con un rompecabezas chino en el cual hay cajas contenidas dentro de otras cajas. Se analiza un nivel superior de macro diseño y se elige una alternativa; después se evalúa otro conjunto de alternativas de macro diseños que existen dentro de dicha alternativa; dentro de esa alternativa elegida, se encuentra otro conjunto más detallado, se evalúa y después se elige una alternativa y así sucesivamente. En la figura 8.1 se muestran las actividades de la fase de macro diseño de sistemas y muestra las secuenciación aproximada de estas actividades. La figura indica que los pasos 1 al 4 que llevan a la evaluación de macro diseños, se repite cuantas veces sea necesario. Por lo común se evalúan secuencialmente tres o más niveles de alternativas macro en una investigación de sistemas que tiene un alcance bastante amplio.

Es característico que al considerarse niveles sucesivos de alternativas, los análisis de factibilidad sean cada vez más detallados. Esto se muestra en la figura 8.1. En forma eventual se elige una configuración específica que se diseña en detalle. También se podrían utilizar algunas otras alternativas de macro diseño. Tam
bién se considerarían las alternativas macro relacionadas con el hardware.



ACTIVIDADES DE MACRO DISEÑO DE SISTEMAS

Figura 8.1

8.1. ESTUDIOS DE FACTIBILIDAD.

Debe ponerse atención a varios puntos durante la investigación de sistemas a la factibilidad de un nuevo sistema, incluyendo la fase de análisis de sistemas. Sin embargo, en general los análisis de factibilidad más profundos, o los estudios de factibilidad, se completan durante la fase de diseño de sistemas, por lo general, durante la consideración de los conjuntos sucesivos de alternativas macro y micro diseño. Los estudios de factibilidad consideran las factibilidad técnica, económica y operacional de cada alternativa.

8.1.1. Factibilidad técnica.

El análisis de factibilidad técnica evalúa si el equipo y software están disponibles (o, en el caso del software, si puede desarrollarse) y tienen las capacidades técnicas requeridas por cada alternativa del diseño que se esté considerando. Los estudios de factilidad técnica también consideran las interfaces entre los sistemas actuales y el nuevo. Por ejemplo, los componentes que tienen diferentes especificaciones de circuito no pueden interconectarse, y los programas de software no pueden pasar datos a otros programas si tienen diferentes formatos en los datos o sistemas de codificación; tales componentes y programas no son compatibles técnicamente.

Sin embargo, puede hacerse una interfase entre los sistemas no compatibles mediante la emulación, la cual son circuitos

diseñados para hacer que los componentes sean compatibles, o por medio de simulación, que es un programa de cómputo que establece compatibilidad, pero con frecuencia estas formas de factiblidad técnica no están disponibles o son demasiado costosas.

Los estudios de factiblidad también consideran si la organización tiene el personal que posee la experiencia técnica requerida para el diseñar, implantar, operar y mantener el sistema propuesto. Si el personal no tiene esta experiencia, puede entrenársele o puede emplearse a nuevos o consultores de sistemas que la tengan. Sin embargo, una falta de experiencia técnica dentro de la organización puede llevar al rechazo de una alternativa particular.

8.1.2. Factibilidad Económica.

Los estudios de factibilidad económica incluyen análisis de costos y beneficios asociados con cada alternativa del proyecto. Con análisis de costo/beneficio, todos los costos y beneficios de adquirir y operar cada sistema alternativo se identifican y se hace una comparación de ellos. Primero se comparan los costos esperados de cada alternativa con los beneficios esperados para asegurarse que los beneficios superen a los costos. Después la proporción costo/beneficio de cada alternativa se compara con las proporciones costo/beneficio de las otras alternativas para identificar la alternativa que sea la más atractiva en su aspecto económico. Una tercera comparación, por lo general implícita,

se relaciona con las formas en que la organización podría gastar su dinero de modo que no fuera en un proyecto de sistemas.

Los costos de implantación incluyen comúnmente el costo "remanente" de la investigación de sistemas (para este propósito, los costos en los que ya se ha incurrido no son relevantes), los costos de operación del sistema para su vida útil esperada, y los costos de mano de obra, material, reparaciones y mantenimiento.

Algunos costos y beneficios pueden cuantificarse fácilmente. Los beneficios que pueden cuantificarse con facilidad son de dos tipos generales: ahorros en costos, tales como una disminución en costos de operación y aumentos en las utilidades directas.

Un problema importante con el análisis de costo/beneficio es la atención inadecuada de costos y beneficios intangibles. Éstos son los aspectos de las alternativas de los nuevos sistemas que sí afectan los costos y utilidades y deberían evaluarse pero que los afectan en formas que no pueden cuantificarse fácilmente. Los factores intangibles con frecuencia están relacionados a la calidad de esta información proporcionada por el sistema y a veces a formas sutiles en que esta información afecta a la empresa, tal como alternando las actitudes para que la información sea vista como un recurso.

Cuando se da mayor importancia a los costos y beneficios cuantificables que a los costos y beneficios intangibles, quizá haya una desviación contra el nuevo sistema porque la mayoría de los costos puede cuantificarse de modo fácil, mientras que muchos de

los beneficios más importantes pueden ser intangibles y por lo tanto no se consideran correctamente.

Dos beneficios intangibles son el servicio a clientes y mejor información administrativa.

Los beneficios intangibles importantes pueden ser adquiridos de un nuevo sistema de información. Es cierto que el principal punto al desarrollar un nuevo sistema puede ser la expectativa de información más exacta y a tiempo, un mejor formato en los informes, o informes que estén más enfocados a áreas particulares de problemas.

Además, en muchos casos un nuevo sistema proporciona información que antes no estaba disponible, como información acerca de costos estándares o incrementos en los costos.

También puede haber menos beneficios intangibles obvios. Un nuevo sistema puede hacer que se obtenga más rápidamente y a toda hora información relevante sobre las actividades de la empresa reduciendo de esta forma tanto tiempos como costos. Un beneficio intangible final es la experiencia obtenida de la investigación de sistemas y del uso de un sistema de información más avanzado a menudo coloca a la organización en una mejor posición para tomar ventajas de desarrollos futuros en tecnología de computación y sistemas de información.

La mayoría de los costos y beneficios intangibles de una alternativa afectan en forma indirecta las utilidades, pero esto es

difícil de medir. La siguiente es una forma de cuantificar los costos y beneficios intangibles.

1.- Identificar las causas y efectos directos. Por ejemplo, el efecto directo de computarizar tareas repetitivas puede ser que un nuevo sistema mejore los trabajos actuales y mejore la calidad de empleo.

2.- Identificar los efectos indirectos. Por ejemplo, una mejor calidad de empleo puede resultar en cerca de 5% menos de ausentismos y un 10% menos en el índice de rotación de empleados.

3.- Estimar el impacto económico de los efectos indirectos para la vida estimada del sistema. Por ejemplo, una reducción en los retrasos de la programación y horas extras debidas a la reducción del ausentismo puede ahorrar casi un 20% al año.

Esta forma puede usarse para una gran variedad de costos y beneficios intangibles. Aunque es arbitraria y subjetiva, es preferible a ignorar los beneficios intangibles. Esta forma puede describirse cómo "hacer tangibles los intangibles".

Una forma alternativa es dejar sin cuantificar a los intangibles. Después, los usuarios y diseñadores de sistemas los estudian y llegan a un acuerdo acerca de la importancia relativa de lo cuantificado y de los costos y beneficios intangibles. Sin embargo, con frecuencia los costos y beneficios intangibles no se analizan

completamente, y no se hace ningún intento para llegar a un acuerdo acerca de su importancia.

8.1.3. Factibilidad Operacional.

Esta factibilidad comprende una determinación de la probabilidad de que un nuevo sistema se use como se supone que debería de funcionar. Aquí se tienen que considerar cuatro aspectos de la factibilidad operacional para un nuevo sistema:

1.- Un nuevo sistema puede ser demasiado complejo para los usuarios y por lo tanto se puede tender a no usarlo o bien a cometer con frecuencia errores o fallas en el sistema.

2.- Un nuevo sistema puede causar varias impresiones en los trabajadores de una empresa como lo son el miedo a ser remplazados por el sistema, el miedo a evolucionar y quedarse fijados en el sistema antiguo, etc.

3.- Un nuevo sistema puede introducir cambios demasiado rápido para permitirle al personal adaptarse a él y aceptarlo. Un cambio repentino que no se ha anunciado, explicado y vendido a los usuarios con anterioridad puede crea resistencia. Sin importar qué tan atractivo pueda ser un sistema aun su aspecto económico si la factiblidad operacional indica que tal vez los usuarios no aceptarán el sistema o que su uso resultara en muchos errores o en una baja en la moral, el sistema no debe implantarse.

4.- La probabilidad de la obsolencia subsecuente en el sistema. La tecnología que ha sido anunciada pero que aún no está disponible puede ser preferible a la tecnología que se encuentra en una o más de las alternativas que se están comparando, o cambios anticipados en las prácticas o políticas administrativas pueden hacer que un nuevo sistema sea obsoleto muy pronto. En cualquier caso, la implantación de la alternativa en consideración se convierte en impráctica.

Un resultado frecuente de hallazgos negativos acerca de la factibilidad operacional de un sistema es que éste no se elimina sino que se simplifica para mejorar su uso.

Una alternativa de solución para estos problemas seria que el sistema se desarrollara en etapas a largo plazo y que estas o se fueran implantado o reemplazando al antiguo sistema poco a poco con el fin de que los usuarios se fueran adaptando al cambio.

8.2. SELECCIÓN DE ALTERNATIVAS DE MACRO DISEÑO.

Se elige un macro diseño como consecuencia de un ciclo de estudios de factibilidad o, cuando se evalúan una serie de conjuntos de alternativas de macro diseño, se realiza secuencialmente una selección de cada conjunto. Como se muestra en la figura 8.1 se presenta una solicitud al área de

sistemas para proceder a hacer una evaluación de alternativas de cada conjunto y elegir de esta forma la mejor de estas alternativas.

En este momento se realiza un informe formal. Incluye un análisis de cada alternativa, la recomendación del equipo y la razón de la recomendación; por lo general, se incluye un resumen que analiza las alternativas en menor detalle que el cuerpo principal del informe y que incluye un resumen de la recomendación. Si aparecen varias alternativas como buenas elecciones, no puede realizarse una recomendación de una sobre las demás. Durante la presentación del resumen se hacen preguntas que reflejan una perspectiva amplia de las metas.

El área de sistemas puede aceptar una recomendación y hacer las preguntas adecuadas para extender así el entendimiento respecto a esta alternativa, también puede inclinarse por otra alternativa o tener en consideración varias recomendaciones.

Debe realizarse una evaluación de las alternativas de macro diseño en consideración de los resultados de los estudios de factibilidad económica, técnica y operacional. Esta evaluación no se basa en un promedio ponderado de los resultados de estos estudios de factibilidad, debe abandonarse una alternativa que no es factible técnica u operativamente.

Después de que se ha seleccionado en forma tentativa una alternativa de macro diseño, puede comezarse una actividad que normalmente se asocia con la fase de implantación. Educar y venderla a los usuarios y operadores de los sistemas. En este

punto, el equipo de trabajo del proyecto conoce la naturaleza general del nuevo sistema y pude apoyar su implantación. Esta educación y venta debe ser cuidadosa y profunda para poder vencer la oposición al nuevo sistema y para obtener la máxima cooperación de los usuarios con el grupo de diseño de sistemas.

9.- MICRO DISEÑO.

Las actividades del micro diseño o diseño detallado son bastante diferentes para proyectos de diferentes tipos, como de adquisición de hardware, selección de sistemas completos, compra de software, y desarrollo de software, que son las principales alternativas (ver figuras 7.1 y 9.1).

El diseño detallado incluye las siguientes actividades:

1.- Diseño del sistema de reunión de datos.

2.- Diseño del sistema de entrada.

3.- Diseño del sistema de procesamiento de datos.

4.- Diseño de la estructura de archivos.

5.- Diseño de los procedimientos de actualización de archivos.

6.- Análisis de los requerimientos de almacenamiento de datos.

7.- Diseño del sistema de consulta.

8.- Diseño del sistema generador de informes.

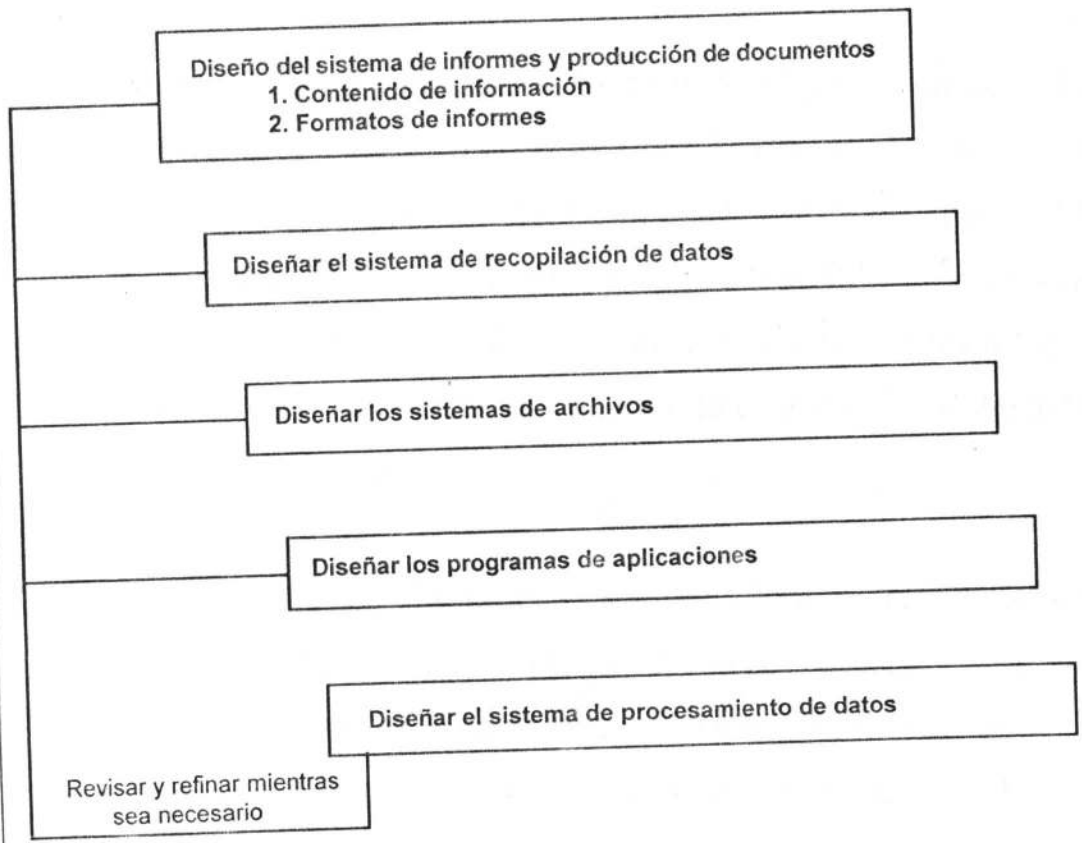
9.- Diseño de los programas de aplicaciones.

10.- Diseño del sistema de seguridad.

ALTERNATIVAS Y ACTIVIDADES DETALLADAS DE DISEÑO

ALTERNATIVA	ACTIVIDAD DETALLADA DE DISEÑO
Adquisición de equipo	Especificar las características que debe tener el equipo
Selección de sistemas completos	Evaluar las características que tienen el equipo y software disponible.
Desarrollo de software	Diseñar la lógica de flujo de datos y diseñar los archivos de datos.

Fig. 9.1



SECUENCIA DETALLADA TÍPICA DE DISEÑO

Fig. 9.2

El diseño detallado de un sistema se realiza dentro de las fronteras de la combinación particular de macro diseño elegida. Entre las tareas de diseño que pueden asociarse con un proyecto importante de sistemas se encuentra el diseño del sistema de entrada y salida, el del sistema de procesamiento y la secuencia de las actividades para procesar los datos y preparar los informes, y la especificación de las capacidades de procesamiento de los componentes de hardware.

En general la secuencia de las actividades varía, dependiendo de la naturaleza del proyecto.

Después se diseña la recolección de datos; esta actividad de diseño puede comenzar recolectando información por medio de un grupo de usuarios mediante la especificación de qué datos se reunirán y cómo se convertirán a forma entendible por la máquina. El sistema de captura de datos incluye el diseño de un sistema de entrada que de línea a los procedimientos del personal y especifica los controles de entrada.

Con frecuencia la recolección de datos y los sistemas de entrada son extensos y con mucho trabajo, incluyendo hasta 30 actividades manuales separadas. Una forma en que tiende mejorarse la eficiencia y control sobre los sistemas de información es buscar formas de automatizar estas actividades; el registro y entrada de transacciones en el punto de venta puede usarse para lograr esto.

Después se diseña el sistema de archivos. Este diseño incluye la determinación de tipo de organización de archivos (secuencial o secuencial indexado, por ejemplo), la longitud del registro y de los campos para cada tipo de registro y su formato. También se incluyen los requerimientos para almacenamiento de datos. El diseño de los procesamientos de actualización de archivos incluye la determinación de quién tiene la autoridad para actualizar archivos y establecer procedimientos para borrar registros, agregarlos y realizar otras tareas de mantenimiento y actualización de archivos. Si el sistema tendrá capacidad de consulta, el diseño debe especificar cómo se determina el derecho para obtener información desde terminales remotas, los códigos de entrada que se usarán para obtención de información, y el tipo de dispositivos de salida que se utilizarán, junto con otras consideraciones relacionadas.

Cuando se ha determinado el diseño de archivos, puede percibirse con más facilidad la estructura necesaria y la organización de los programas de aplicación. El diseño de los programas de aplicación es una de las tareas principales en la investigación de sistemas que implican programación. Estas actividades incluye la determinación del número de programas y la función de cada uno.

El diseño de programas también incluye la preparación de "diagramas de flujo de sistemas", que muestran los flujos generales de información a través del sistema, y "diagramas de flujo de programas", que muestran toda la lógica del nuevo sistema que debe estar contenido en cada programa.

En esta idea, esta actividad de diseño está separada de la actividad de implantación que consiste en la escritura y corrección de los programas. Sin embargo, en la práctica, estas actividades están muy relacionadas.

Finalmente, como se muestra en la figura 9.2, se diseña el sistema de cómputo de procesamiento de datos, el cual, por lo general incluye el desarrollo de un manual de procedimientos para operadores de la computadora que especifica los pasos que deben seguir los operadores para procesar una aplicación particular. Cuando deben seguir los operadores para procesar una aplicación particular. Cuando deben escribirse programas de cómputo el diseño detallado del sistema de procesamiento de datos incluye una frecuencia el desarrollo de diagramas de flujo de datos, los cuales muestran cómo fluyen estos a través del sistema. Durante la fase de implantación, estos diagramas de flujo sirven como base principal para la escritura de los programas de cómputo.

No existe una secuencia de actividades de diseño detallado: la secuencia depende se encuentran el proceso simultáneo. Las actividades de diseño detallado son fluidas e interactivas por naturaleza.

10.- MÉTODOS DE DISEÑO.

Existen muchos métodos (ver glosario) y técnicas que son una herramienta muy útil para el diseñador de sistemas a la hora de poner en práctica sus ideas sobre el sistema a realizar, a fin de ayudar a registrar y razonar sus ideas.

10.1.- PROBLEMAS DE LOS MÉTODOS DE DISEÑO DE SOFTWARE.

Existe cierto valor al observar la tecnología para el soporte del diseñador de software, aunque es necesario apreciar las dificultades que existen al intentar describir las tecnologías actuales de diseño. Estos problemas caen en cuatro áreas:

- Diversidad
- Clasificación
- Visibilidad
- Exactitud

10.1.1.- DIVERSIDAD.

Se han creado un sin número de métodos, notaciones, técnicas y herramientas a lo largo y ancho del mundo, mezclando nuevas técnicas eficaces ya existentes e inclusive

existen organizaciones que ya han patentado algunas de las técnicas más populares.

La mayoría de estos métodos, notaciones y herramientas son creadas de manera rápida y muchas de ellas carecen de una amplia difusión.

La investigación ha demostrado que el uso de tecnología es sustancialmente diferente en varios países.

10.1.2.- CLASIFICACIÓN.

Una vez que los métodos ya han sido identificados, se pasa a la fase de clasificación. Aquí se emplean varios criterios para definir los diferentes enfoques, y agrupar los resultados de una investigación siempre resulta en excepciones que se descubren. Por lo cual ninguna clasificación que se puede emplear es totalmente perfecta.

10.1.3.- VISIBILIDAD.

La tercera área de problema es que por lo común es difícil saber cuál es la tendencia real que se utiliza en la actualidad. Ya que en las organizaciones es muy difícil que se de información acerca de como desarrollan software. Es probable que debido al hecho de que una cantidad significativa del desarrollo de software no se adquiere por

enfoques muy organizados y disciplinados y también que esas organizaciones que creen que están utilizado los mejores métodos no necesariamente quieren decir a sus competidores qué están usando. Esas organizaciones y departamentos que informan de manera voluntaria tienden a demandar el anonimato y un alto grado de confidencialidad.

10.1.4.- EXACTITUD.

Habiendo descubierto una organización o persona que pretende utilizar un cierto método en particular, no necesariamente quiere decir que realmente lo hagan.

Lo que realmente pasa en la actualidad es que no necesariamente la adopción de ciertos métodos de diseño sistemáticos prueban la calidad del producto final.

Las observaciones de los autores a lo largo de un rango amplio de proyectos sugieren que los beneficios se acreditan pero rara vez se miden contra cualquier criterio objetivo. En lugar de eso, se basan en un claro entendimiento de cómo y por qué se utilizan los métodos elegidos

11.- CONTROLES DE SISTEMAS

Durante el micro diseño, debe ponerse atención a incluir controles sobre la entrada de datos, su procesamiento e informes y otros tipos de salidas. Estos controles deben ser consistentes con un sistema de información de control preestablecido que esté basado en la percepción de la organización de sus riesgos de seguridad de datos.

Además, los controles sobre los sistemas de cómputo se convierten en una parte del esquema general de control interno.

Si un control no es incluido desde un principio es muy difícil agregarlo al sistema una vez ya implantado, por lo que es conveniente tratar de incluir en una lista todos los tipos de controles que se pueden incluir en el nuevo sistema; algunos tipos de contarles son los siguientes:

11.1.- CONTROLES GENERALES.

Estos controles de sistemas se clasifican en dos clases: controles generales y controles técnicos. Los generales son aquellos aplicables a todas las aplicaciones y constituyen una gran parte del marco general de controles preexistentes. Las previsiones para la supervisión de las actividades, las políticas administrativas con respecto a la documentación, las prácticas de desarrollo de software, las políticas y procedimientos de corrección

de errores, y las políticas de auditorías internas en relación con los sistemas de información son ejemplos de controles generales.

11.2.- CONTROLES TÉCNICOS.

Son también llamados controles de aplicaciones y son controles sobre aplicaciones individuales o conjuntos de aplicaciones. Estos son controles sobre. Entrada de datos, procesamiento de datos, salida de datos y distribución de informes.

Hay una variedad de tipos de controles de procesamiento, muchos de los cuales se encuentran escritos en los programas. Los programas pueden incluir transacciones, por ejemplo, que aseguren que los datos de un campo individual del registro sea numérico o alfabético, dependiendo en qué forma sea correcto. También se incluyen comprobaciones de límites en programas para asegurar que las cantidades máximas o mínimas esperadas no se excedan.

Otros controles de procesamiento están relacionados con asegurarse que los operadores de la computadora procesen los trabajos en forma adecuada. Hay muchos otros controles de procesamiento.

Los controles de salida consisten en técnicas para asegurar que todas las transacciones que entraron han sido procesadas completamente y que están reflejadas en los informes u otra salida producida por el sistema. Los controles de distribución consisten en procedimientos que aseguren que los documentos e informes se distribuyen a tiempo a todas las personas en la lista de distribución y a nadie más.

11.3.- CONTROLES DE USUARIOS.

Los controles de usuarios son otro tipo de control. Los grupos de usuarios tienen la responsabilidad de estar seguros como sea posible de que el procesamiento de datos se ha realizado correctamente.

Con frecuencia, los controles de usuarios son tan elaborados que con casi una copia parcial del procesamiento que realiza en sistema de computo, dando con esto un análisis muy completo del sistema, pero que tiene un impacto ineficiente. Los diseñadores de sistemas deben ayudar a diseñar controles para los usuarios que les permitan saber cuándo han ocurrido errores durante la entrada de datos, procesamiento o salida.

11.4.- CONTROLES DE CORRECCIÓN DE ERRORES.

Deben controlarse los proyectos de sistemas. Nombrar un comité directivo de sistemas, usar sistemas de control de proyectos y proporcionar buena supervisión son buenos métodos de

control de investigación de sistemas. También se requieren métodos adicionales. Éstos incluyen la institución de un sistema de autorización para cada tarea dentro del proyecto del sistema y usar controles que aseguren que el nuevo sistema esté bien probado y libre de errores.

Durante una investigación de sistemas son muy importantes los procedimientos que garanticen que los controles existentes en el sistema actual continúen vigentes mientras se diseña e implanta un nuevo sistema. A menudo se tiene la tentación de no tomar en cuenta los controles que pronto se van a eliminar con un nuevo sistema. Sin embargo, cuando un nuevo sistema es más vulnerable debido a la confusión general causada por esas actividades.

12.- DESARROLLO DE SOFTWARE

Si la organización decide desarrollar los programas que serán parte del nuevo sistema, aún existen alternativas a ser consideradas, como se indica en la figura 7.1.

Aunque tanto los usuarios como el personal de sistemas pueden usar prototipos para desarrollar programas, este enfoque lo usa con mayor frecuencias el personal administrativo para el desarrollo de sistemas de apoyo de decisiones.

Las actividades de desarrollo de diseño de archivos, de procedimientos de entrada, de procesamiento y de procedimientos de salida como también se muestra en la figura 7.1, podrían realizarlas tanto los usuarios como el personal de sistemas. Los grupos de usuarios por lo general realzan estas actividades sólo si son relativamente sencillas, directas y pequeñas en escalas, dados los proyectos de sistemas basados en microcomputadoras.

Una de las etapas principales del desarrollo de sistemas, es la etapa de el pedido de una propuesta, que consiste en un pedido preparado por la organización quien la entrega a los desarrollares de sistemas, pidiéndoles que preparen una propuesta y se la entreguen. Como se puede ver en la figura 7.1, un pedido de una propuesta se prepara, quizá para proyectos que comprenden la compra de hardware, sistemas completos, o software. En general, una compañía identifica de tres a seis vendedores a quienes solicitan un pedido de una propuesta idéntica, cuyos

productos parecen satisfacer, en forma potencial, los requerimientos de la compañía.

El pedido de la propuesta contiene todos los detalles necesarios para que un proveedor prepare una propuesta completamente detallada.

El momento de una propuesta está regido por las circunstancias. En algunos casos, la selección de equipo se realiza en cuanto se termina el informe de especificaciones de requerimientos de sistemas al final de la fase de análisis; si se entrega un pedido de una propuesta a los proveedores en momento tan prematuro, antes de comenzar el trabajo de diseño, se le pide a cada proveedor decidir sobre un diseño apropiado para el sistema de información. En general, un diseño de este estilo proporcionado por el proveedor no es consecuencia de un entendimiento profundo de las operaciones de la organización y con frecuencia repercute en un sistema de información que no sirve a las necesidades de la organización.

Sin embargo, en algunas circunstancias, sobre todo cuando la organización no tiene diseñadores de sistemas con mucha experiencia, este enfoque es satisfactorio. Es preferible que el pedido de propuesta se prepare después de uno o más ciclos de selección de macro diseños.

Si la propuesta es para la adquisición de equipo y se prepara después de un poco de actividad de diseño, esta actividad produce buenos resultados (especificaciones de diseño) para el equipo.

Con base en las especificaciones de requerimientos, las especificaciones de diseño establecen una configuración deseada que puede incluir, por ejemplo, el menor tamaño de memoria principal, las características requeridas de la memoria secundaria (ver glosario), tipos y capacidades generales de equipo de entrada y salida necesario, entre muchas posibilidades. De ordinario, las especificaciones de diseño deben desarrollarse sin referencia a cualquier modelo especifica de equipo o a una línea de productos en particular de un proveedor.

Por lo general un pedido de una propuesta para equipo se prepara y selecciona antes de los esfuerzos de diseños más detallados, ya que la mayoría de éstos debe realizarse según el reconocimiento de las especificaciones del equipo; por ejemplo, con frecuencia los programas de aplicaciones deben diseñarse con cuidado de acuerdo con la naturaleza y posibles limitaciones del sistema operativo de la computadora seleccionada.

Una vez recibidas respuestas a una propuesta, la organización las evalúa. Por lo general, a cada proveedor, cuya propuesta sea competitiva en términos de precio y de satisfacer los requerimientos y necesidades del pedido de la propuesta, así que se les solicita una presentación para explicar la propuesta y los productos propuestos a los clientes. Durante la presentación, los representantes de la organización hacen preguntas sobre todos los aspectos de las recomendaciones y propuestas del proveedor.

13.- TENDENCIAS ACTUALES EN EL DISEÑO DE SOFTWARE.

En la actualidad la mayoría de los diseñadores siguen empleando el método del ciclo de vida tradicional. Al mismo tiempo, cada vez crece el número de personas interesadas en paradigmas de diseño que desplazan el énfasis "desde que funciones debe desempeñar el sistema" hasta "que datos necesita manipular el sistema".

Los enfoques de diseño que soportan esto, tal como el diseño orientado a objetos, están teniendo mayor aceptación.

En las organizaciones que se dedican al desarrollo de software la tendencia esta orientada hacia el uso de las herramientas de software CASE.

Pocas organizaciones se inclinan a usar métodos basados en matemáticas ya que tales métodos restringen mucho sus aplicaciones, y son mucho más difíciles que después de un tiempo de haberse desarrollado no se entienda la metodología utilizada para posteriores actualizaciones, inclusive confundiéndose el propio desarrollador del sistema.

En el área de procesamiento de datos las tendencias son hacia los lenguajes de cuarta generación y el término "prototipar rápido". Estos tipos de aplicación (payroll, ledger, etc.) (ver glosario), tienden a ser imparcialmente definidos, y las

herramientas actuales de prototipo lend (ver glosario), por sí mismas, se vuelven con mayor facilidad para esta organización de desarrollo que para las aplicaciones de tiempo real.

La tendencia general parece ser hacia esas ideas más nuevas de orientación de datos, herramientas de soporte integradas y técnicas de especificación más rigurosas.

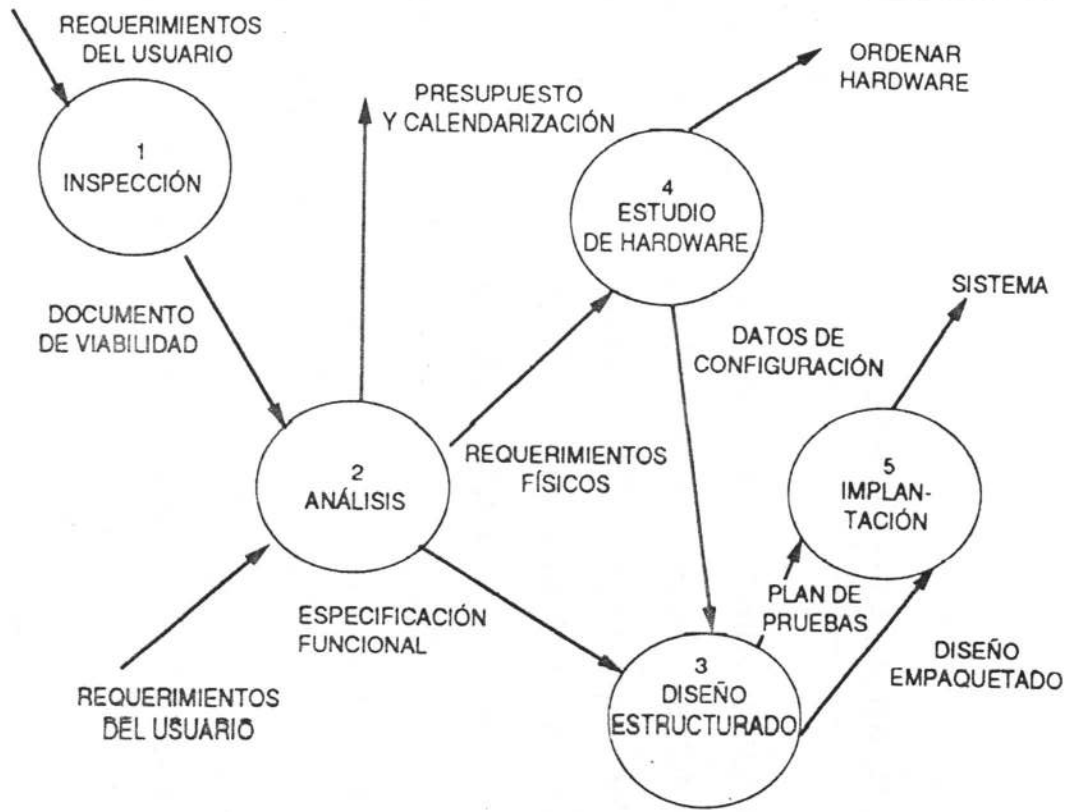
14.- NOTACIONES USADAS EN EL DISEÑO DE SOFTWARE.

Aquí se describen algunas de las notaciones más empleadas por la organizaciones, muchas de las cuales se requieren cuando se siguen uno o más de los métodos que se describen a continuación.

14.1.- DIAGRAMA DE FLUJO DE DATOS.

Un Diagrama de Flujo de Datos (DFD) muestra los elementos de procesamiento de un sistema, los flujos de datos entre los elementos de procesamiento y almacenamiento de datos principal dentro de un sistema. · Un diagrama de flujo de datos debe seguir subdividiéndose hasta detallar las funcionales de cada uno de los procesos que serán desarrollados para constituir el nuevo sistema. Un diagrama de flujo de datos puede descomponerse en niveles nuevos de diagramas (o subniveles) para exhibir grandes cantidades de detalle.

Los diagramas de flujo de datos son utilizados por lo general para mostrar los flujos de datos deseados o actuales, lógicos o físicos, en un sistema. En el diagrama de flujo de datos no hay indicación de la secuencia u orden, sólo que dato es y cómo se manipula.



EJEMPLO DE DIAGRAMA DE FLUJO DE DATOS

Figura 14.1

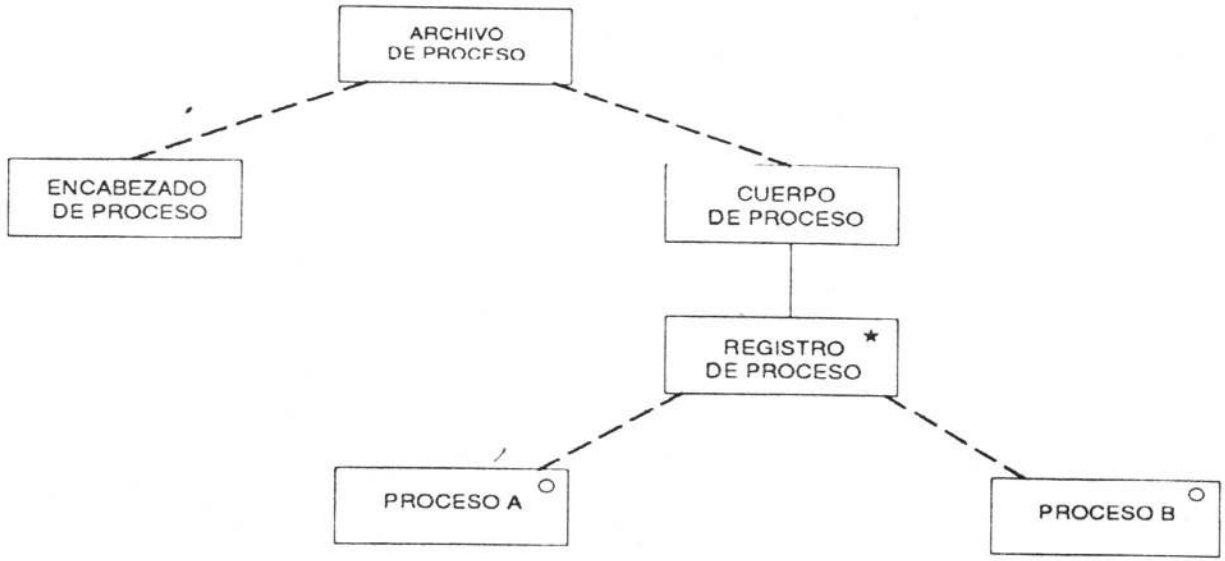
En la figura 14.1 se muestra un ejemplo de un Diagrama de Flujo de Datos con actividades contenidas en los círculos (los elementos de procesamiento) y el flujo de datos etiquetados en ellos.

14.2.- DIAGRAMAS DE ESTRUCTURA DE DATOS.

Un Diagrama de Estructura de Datos (DED) describe cómo puede descomponerse un ítem (ver glosario) de datos de ítems más pequeños. En el nivel más alto, debe representarse un objeto completo, por ejemplo: un acta de nacimiento. Este puede descomponerse en campos separados, por ejemplo: nombre y fecha de nacimiento. La fecha de nacimiento puede entonces descomponerse a un nivel al especificar el formato de la fecha que se espera, por ejemplo: dd/mm/aa. Los DED´s se dibujan casi siempre en forma de diagrama como un árbol invertido, así como la estructura textual para un diseño basado en él, ver figura 14.2.

14.3.- DIAGRAMAS DE RELACIONES DE ENTIDADES.

Los Diagramas de Relaciones de Entidades (ER), se utilizan para mostrar el tipo de relación existente entre las entidades diferentes en un sistema. Esas relaciones pueden ser, por ejemplo, "uno a uno" o "muchos a muchos". Las entidades se



Forma de texto de diagrama

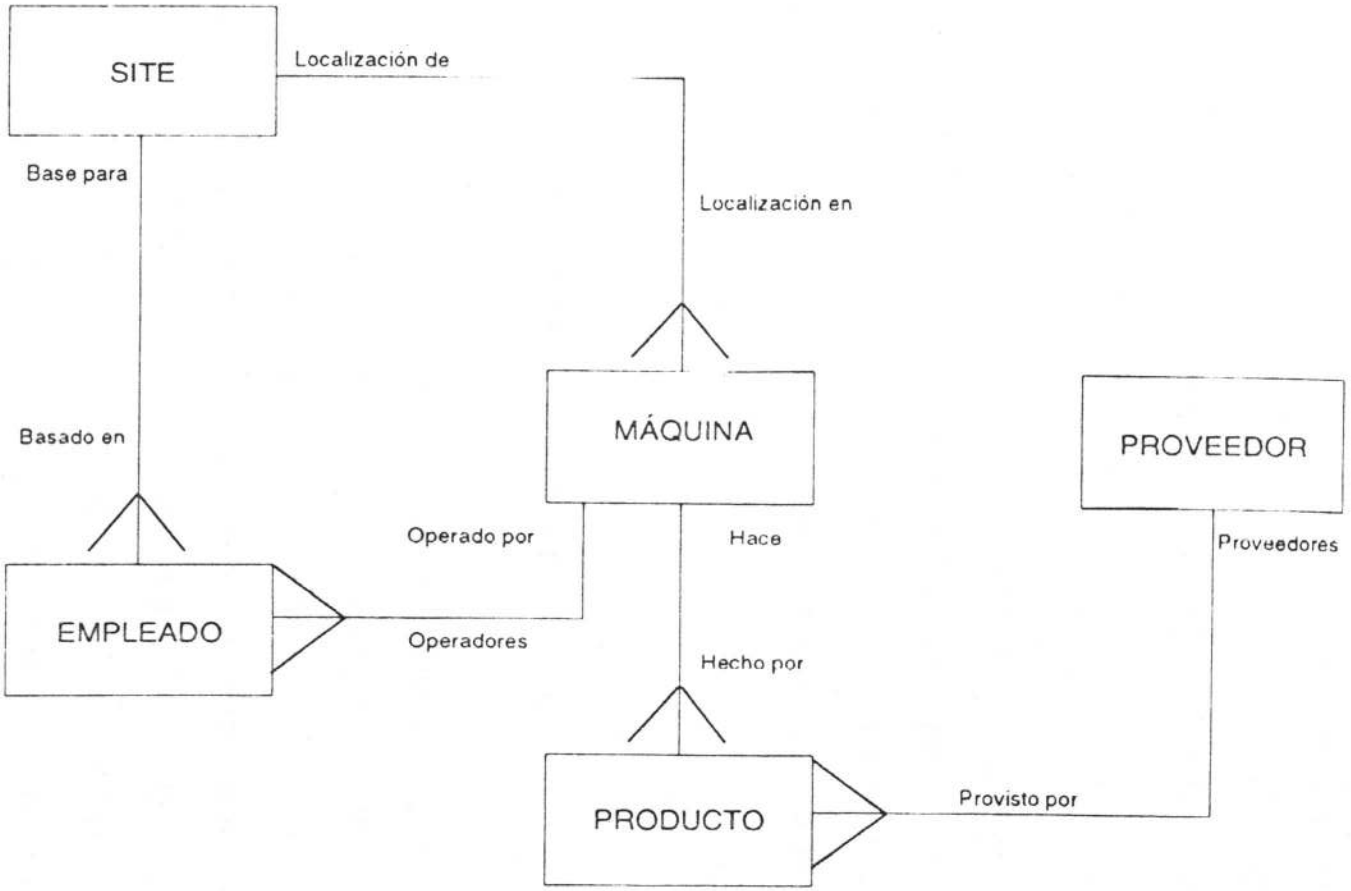
'ARCHIVO DE PROCESO' es una secuencia:-
ENCABEZADO DE PROCESO; CUERPO DE PROCESO

'CUERPO DE PROCESO' es una iteración:-
MIENTRAS (hay muchos registros más) REGISTRO DE PROCESO

'REGISTRO DE PROCESO' es una selección:-
SI (Registro es tipo A) ENTONCES PROCESO A TAMBIÉN PROCESO B

EJEMPLO DE DIAGRAMA DE ESTRUCTURA DE DATOS

Figura 14.2



EJEMPLO DE DIAGRAMA DE RELACION DE ENTIDADES

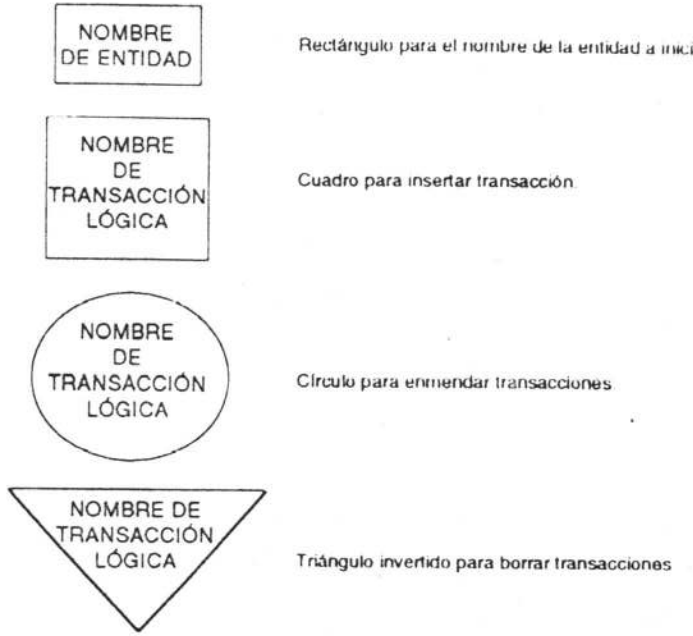
Figura 14.3

representan por medio de cuadros y las relaciones son denotadas por diferentes tipos de flechas en líneas que interconectan los cuadros. Ver figura 14.3, así, una flecha doble puede denotar una relación "muchos". Los cuadros contendrán el nombre de la entidad, para después ser descrito de una manera completa en un diccionario de datos.

Un conjunto alternativo de símbolos consiste que denotan las entidades diamantes que representan los tipos de relaciones. Las líneas conectan los rectángulos y los diamantes, por su parte, unen las entidades con sus relaciones apropiadas.

14.4.- HISTORIAS DE VIDA DE ENTIDADES.

Las Historias de Vida de Entidades (HVE), son diagramas que describen como se crean, modifican y eliminan las entidades (ver glosario) en un sistema a lo largo de su vida. Para mostrar los procesos que en alguna forma se encargan de crear, modificar y eliminar una entidad, se representa através de tres diferentes tipos de cuadros. Esos cuadros se conectan por líneas dirigidas que se unen los posibles procesos que pueden seguir a otro, ver figura 14.4. Al trazar una HVE, es posible verificar que, por ejemplo, no se crean elementos de datos que nunca son modificados o eliminados. Los diagramas HVE pueden utilizarse de manera típica en conjunción con los diagramas de flujo de datos, de los cuales pueden identificarse los procesos.



SIMBOLOS USADOS EN DIAGRAMAS DE HISTORIA DE VIDA DE ENTIDAD

Figura 14.4

14.5.- ENTIDAD-RELACIÓN-ARCHIVO.

Antes de diseñar formalmente los archivos de un sistema, es importante conocer la relación que tienen sus campos entre sí mediante la definición de la entidad-relación de los archivos, para lo cual se requiriere hacer lo siguiente:

1- Relacionar cada uno de los archivos que previamente se identificaron en el diagrama de flujo de datos (entidades).

2- Definir los elementos de datos principales que describen a los registros de cada archivo (campos).

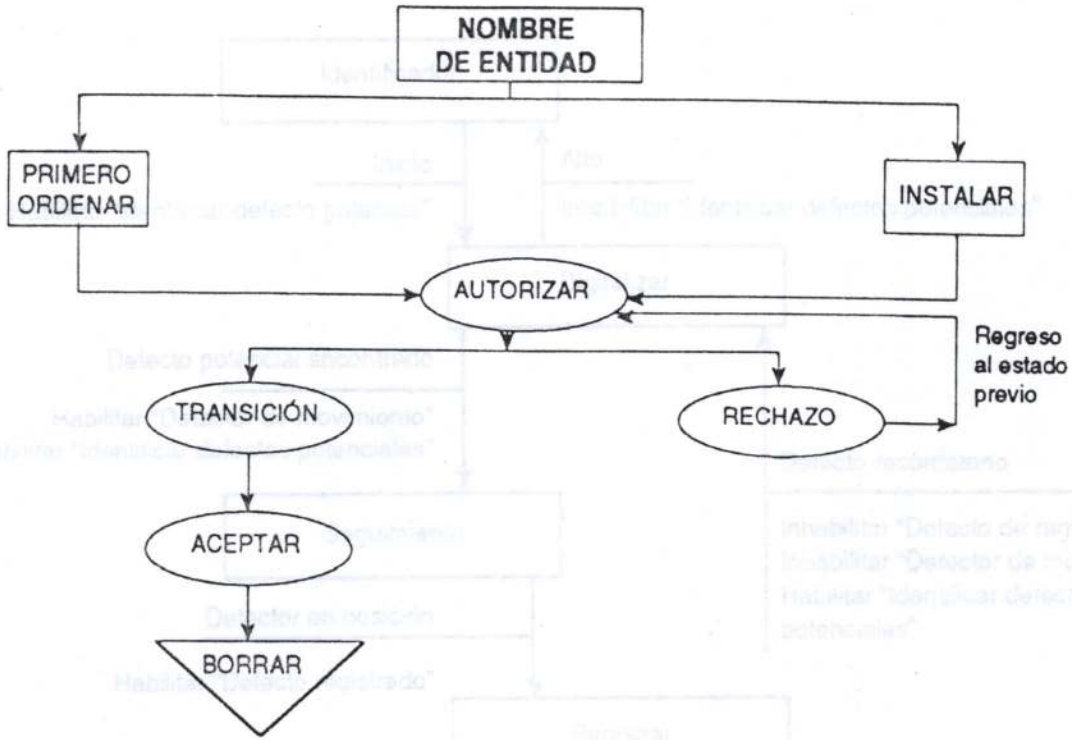
3- Identificar al elemento de datos que por su valor diferencia a un registro de otro (clave de registro).

4- Graficar la relación de los campos de todas las entidades o archivos del nuevo sistema.

Las entidades se tienen que relacionar para así formar un consenso del sistema.

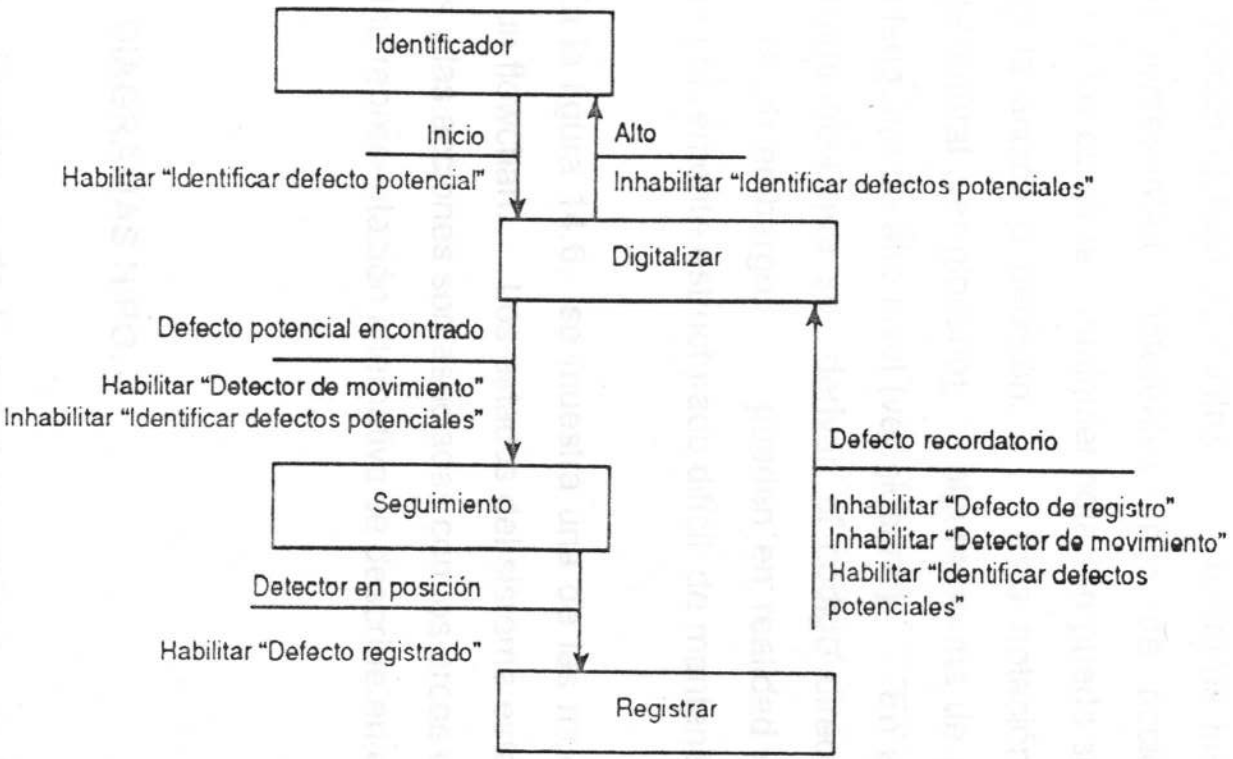
14.6.- FLOWCHARTS.

Los Flowcharts representan la estructura de un programa o sistema, con énfasis en el flujo de control y las acciones primitivas desempeñadas por el programa o sistema. La



EJEMPLO DE DIAGRAMA DE HISTORIA DE VIDA DE ENTIDAD

Figura 14.5



EJEMPLO DE FLOWCHARTS

Figura 14.6

estructuración de datos del sistema no está direccionada del todo. Los Flowcharts consisten en un conjunto de cuadros de varias formas, interconectadas por un conjunto de arcos dirigidos. Éstos indican el flujo de control, mientras las formas de los cuadros representan diferentes tipos de acción o decisión. Dentro de los cuadros, cualquier notación puede ser utilizada para describir la acción o decisión. Esta notación puede ser en lenguaje natural (ver glosario), alguna forma de pseudocódigo o incluso lenguaje de alto nivel (ver glosario). En este último caso las herramientas que se derivan el código directamente de los Flowcharts sin embargo, pueden en realidad dar realce a un software pobremente estructurado difícil de mantener.

En la figura 14.6, se muestra una de las muchas formas de dibujar un flowchart. Los estados del sistema están dados en los cuadros y las acciones son asociadas con los arcos conectores.

Una representación alternativa se describe en el método SDL.

14.7.- DIAGRAMAS HIPO.

Los Diagramas de Entradas jerárquicas, de Procesos, de Salida (HIPO), fue desarrollado y refinado por IBM, HIPO se pueden usar para capturar las relaciones jerárquicas e interdependientes en un sistema en forma ordenada. Éste captura de descomposición hacia abajo al capacitar a sus usuarios para romper una vista de nivel bajo en vistas más refinadas de niveles inferiores, sin tener que incluir detalles lógicos. HIPO

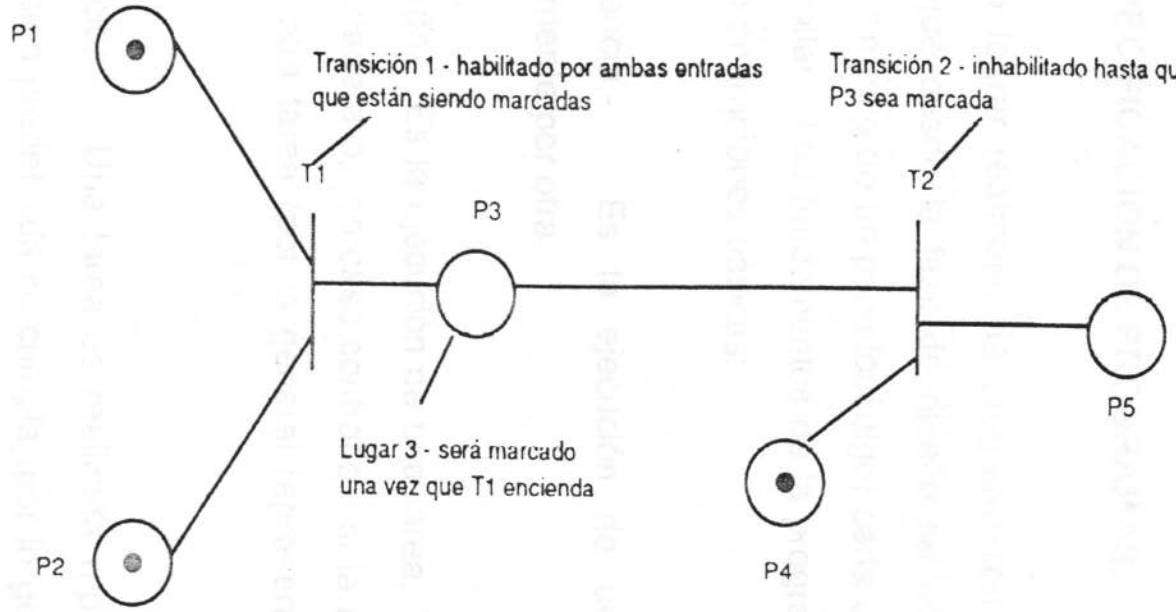
utiliza un conjunto simple de símbolos, aunque los diagramas producidos para los grandes sistemas dificultan cambiar y verificar para la consistencia.

14.8..- PETRI NETS.

Los Petri Nets proporcionan una forma de modelación de sistemas actuales de manera gráfica. Un Petri Nets consiste en un conjunto de símbolos que representan entradas, salidas y estados. Los símbolos se conectan con líneas que muestran los posibles cambios de estado. Una característica particular de los Petri Nets es que, una vez que un sistema se expresa de esta forma, el Net puede utilizarse para explorar ciertas propiedades del sistema, como deadlock y ciclos no productivos. Tales diagramas permiten que el comportamiento de los sistemas sea ejercitado en forma útil.

Los Petri Nets son muy usados en el diseño de sistemas operativos principalmente, ya que para otras aplicaciones más generales, el grado de dificultad es mayor debido a su complejidad.

En la figura 14.8 se muestra un ejemplo de un Petri Net, el círculo (lugar) a la izquierda del diagrama se marca mostrando que la transición del lugar 1 al lugar 2 puede tener efecto. La transición al lugar 3 no puede puesto que todos los lugares de entrada tienen que estar marcados primero. El examen del posible orden de



EJEMPLO DE RED PETRI NETS

Figura 14.8

transiciones es una forma de emplear un Petri Net como una ayuda para el diseño.

14.9- ESPECIFICACIÓN DE PROGRAMAS.

Para lograr realmente la programación estructurada, se requiere que desde la fase de diseño se especifiquen en forma lógica, (por medio de un pseudocódigo) cada uno de los programas por desarrollar. Los fundamentos de la programación estructurada son cuatro operaciones básicas:

1.- Secuencia.- Es la ejecución de una tarea, seguida inmediatamente por otra.

2.- Selección- Es la ejecución de una tarea, cuando la decisión resulta verdadera o, en caso contrario si la decisión resulta falsa se realiza otra tarea (por lo general representadas por el if-then-else).

3.- Repetición.- Una tarea es realizada repetidamente mientras una condición predefinida se cumpla (por lo general representadas por el do-while).

4.- Hasta que.- Una tarea es realizada repetidamente hasta que una condición predefinida se cumpla.

Una buena técnica para especificar programas consiste en utilizar un diagrama de entradas-procesos-salidas, con una especificación narrativa en pseudocódigo estructural. El elemento principal de esta técnica es:

- Pseudocódigo estructural.

14.9.1.- Pseudocódigos e inglés estructurado.

Es un lenguaje algorítmico narrativo con pseudocódigo, que respeta las operaciones lógicas de la programación estructurada (secuencia, selección, repetición, etc.) y cuenta con sintaxis adicional para lograr una especificación adecuada de un programa.

Los Pseudocódigos y el inglés estructurado son notaciones que semejan lenguajes de programación, usados para el diseño de programas. Un pseudocódigo por lo general incorpora las primitivas de flujo de control de un lenguaje de programación, pero combina éstas con una descripción narrativa en prosa de la computación al llevarse a cabo.

El pseudocódigo debe cumplir con las siguientes características:

a) Debe de respetar las operaciones lógicas de la programación estructurada (secuencia, selección, repetición).

b) Debe tener una sintaxis libre y utilizar un lenguaje natural.

c) Las expresiones matemáticas deben de respetarse, ya que se trata de un lenguaje algorítmico y utiliza reglas internacionales de especificación.

d) No debe estar relacionado con ningún lenguaje de alto nivel.

Tomando en cuenta estas restricciones, podremos formar nuestro lenguaje en pseudocódigo, el cual se puede utilizar bajo los siguientes postulados:

1.- Postulado de secuencia

formato:

```
inicio (nombre)
        (postulados)
fin
```

significado:

Sirve para interconectar la secuencia de un conjunto de postulados, los cuales tienen un nombre o función que desarrollar.

Ejemplo:

```
Inicio (nombre)
        Leer A
        Imprimir A
        Calcular B = A + A
        Imprimir B
Fin
```

2.- Postulado de selección

formato:

a) Si ? entonces A1

b) Si ? entonces A1, de lo contrario A2

Significado:

Si se cumple la condición realiza la función A1 de lo contrario realiza la función A2.

3.- Postulado de repetición

formato:

condición i

a)mientras ? , se ejecuta S;

significado: repite S hasta ?

Son postulados que realizan un conjunto de instrucciones hasta que se cumple una cierta condición.

Ejemplos:

a) mientras X<Y ejecuta

b) X = Y – Z

Repite

Leer (registro)

Hasta fin-de-archivo

4.- Postulado de selección múltiple.

Formato:

Condición i

a) proceso 1
b) proceso 2
c) proceso 3

Fin

Significado:

Dependiendo del valor de i ejecutará un cierto proceso, el cual está especificado entre la cláusula condición y fin.

Ejemplo:

Condición A

5: B = C * A
9: B = C * C
10: B = A * A

14.10.- DIAGRAMAS DE TRANSICIÓN DE ESTADOS.

Los Diagramas de Transición de Estados (DTE), son en esencia, una gráfica dirigida (en vez de cómo un flowcharts) que

muestra los estados posibles del sistema como nodos, representados por cuadros, y los cambios factibles que pueden tener lugar, representados por líneas uniendo los nodos. A menudo, los eventos que accionan los cambios de estado son anotados en el diagrama. Un diagrama DTE está apoyado en una base teórica que permite el análisis formal.

15.- PRINCIPALES MÉTODOS USADOS EN LA ACTUALIDAD.

En la actualidad existen cinco métodos principales los cuáles son los que mayor emplean tanto las organizaciones que diseñan software, así como los diseñadores, los cuáles son: el método JSD, el método MASCOT, el método SSADM/LSDM y el método YOURDON.

De igual forma existen otros métodos especializados o menos maduros que también son utilizados por la industria, pero que son potencialmente importantes, que cubren muy bien las necesidades de un diseñador de sistemas.

15.1.- JSD.

El método Jackson System Design (JSD), se basa en la modelación del sistema, bajo un número de pasos y notaciones bien definidas. El método del JSD es muy flexible y no existe una base matemática detrás de él (como e n la mayoría de los métodos actuales más usados).

Este método fue originado en 1983, por Michael Jackson Systems Ltd, (MJSL), ahora parte de LBMS. Las áreas de aplicación a donde ha sido orientado son principalmente a procesamiento de datos y de tiempo real.

JSD cubre las siguientes etapas del ciclo de vida de análisis y diseño. Y es compatible con la programación estructurada.

El método JSD, consiste de tres fases:

1.- MODELADO.- Aquí se pretende modelar los eventos y las acciones. Se identifican Vectores de Estado (ver glosario). Los diagramas de estructura de entidad son generados en esta fase. Una acción JSD es un evento cuya ocurrencia debe ser registrada por el sistema y sobre el cual el sistema tiene que producir salidas. Cada acción tiene una definición y un número de atributos.

2.- FASE DE RED.- Desarrolla el resto de la especificación del sistema. Los diagramas de red que se produce describen procesos, cadenas de datos y flujo de datos.

3.- IMPLANTACIÓN.- Esta fas es cuando los procesos y los datos se ajustan en los procesos disponibles y la memoria.

Esto se logra mediante una técnica ascendente, llamada program inversion, que convierte un proceso en una subrutina que puede ser llamada por cualquier otro proceso que proporcione sus cadenas de datos de entrada.

15.2.- MASCOT.

El método Modular Approach to Software Construction, Operation and Test (MASCOT), fue desarrollado por RSRE (Malvern) con apoyo del Ministro de defensa.

El método MASCOT es adecuado de manera específica para las aplicaciones de tiempo real. MASCOT acopla el diseño, la implantación y las etapas de prueba del ciclo de vida. Este método tiene como usuarios más importantes al Departamento de Defensa de Estados Unidos y Reino Unido.

El método MASCOT consiste de seis etapas. El método enfatiza el comportamiento dinámico y los detalles lógicos y físicos del diseño. Incorpora una característica templete/componente que supuestamente agrega un componente re-uso y un núcleo run-time para apoyar al método.

Por medio de interfaces, este método logra tener una mayor comunicación entre módulos, así como también la independencia de módulos cuando es necesario.

Las seis etapas de MASCOT son:

1.- REQUERIMIENTOS Y RESTRICCIONES EXTERNAS.- Aquí se establecen los requerimientos generales y las restricciones externas (entendiéndose limitaciones de hardware y software). Involucra la consideración de la especificación funcional del

sistema, el ambiente de hardware para el sistema, el sistema de desarrollo a utilizar, y todas las interacciones externas con el sistema requerido.

2.- PROPÓSITO DE DISEÑO.- Esta etapa se refiere a la producción del propósito de diseño de alto nivel. Esta es una descripción formal del diseño de alto nivel de software de aplicación con el sistema template (ver glosario) establecido en forma parcial y los componentes del sistema introducidos también de manera parcial. Cada componente se describe de modo funcional y se especifica su contribución al requerimiento del sistema.

3.- DESCOMPOSICIÓN DE RED.- Esta etapa se refiere a la descomposición progresiva del sistema en términos de redes MASCOT.

Esto involucra el alcance del nivel de elementos con templantes de red siendo introducidos.

4.- DESCOMPOSICIÓN SECUENCIAL DE ELEMENTOS.- En esta etapa, los elementos descritos en la etapa de "Descomposición de Red" se descomponen en subelementos. Estos deben estar a un nivel que permita la implantación hacia delante en la etapa número cinco.

5.- IMPLANTACIÓN.- Esta es la etapa de programación, en la cual se genera el código para cada uno de los elementos y sublementos contemplantes.

Aquí se describen los procedimientos de acceso para implantar las especificaciones de interfaz.

6.- PRUEBA.- La etapa final se refiere a la prueba de integración y aceptación del software y, a menudo, resulta en algún grado de doble trabajo de diseño. Esta etapa se subdivide en la definición de pruebas que debe corresponder a la prueba de requerimientos identificada por medio de las etapas 2, 3, 4 y 5 y la prueba de ejecución, cuando todas las pruebas previamente definidas son corridas.

15.3.- SDL.

El Lenguaje de Especificación y Descripción CCITT, .más conocidos como SDL, fue creado por grupos de estudio de CCITT en 1970. SDL es un lenguaje utilizado para especificar el comportamiento de los sistemas. SDL puede utilizarse también como método de documentación para representar una descripción completa del sistema o de la especificación, así como para telecomunicaciones (ver glosario), por estas razones es considerado más como un método que como una notación.

SDL puede usarse para representar las propiedades funcionales de un sistema. Las propiedades funcionales consisten de algunas propiedades estructurales y ciertas propiedades de comportamiento.

SDL proporciona una forma de describir un sistema con varios grados de formalidad, desde sólo el uso de constructores SDL con lenguaje natural, hasta la combinación de estatutos formales de tipos y operadores definidos al construir SDL.

SDL tiene dos formas diferentes, una de las cuales es una representación gráfica basada en un conjunto estandarizado de símbolos, mientras que la otra es una representación de frases textuales utilizando estatutos parecidos a los de programación. Ambas formas se basan en el mismo modelo semántico y representan los mismos conceptos SDL.

15.4.- SSADM/LSDM.

El método Structured System Analysis and Design (SSADM). En 1980, el gobierno de Gran Bretaña inició un procedimiento para seleccionar un método estructurado a fin de utilizarlo como estándar en todos los proyectos computacionales en los departamentos gubernamentales. es un conjunto de técnicas estructuradas en un marco de pasos y etapas. Durante esos pasos una gran cantidad de documentación textual y gráfica se produce.

Los siguientes principios básicos de SSADM son compartidos por muchos otros métodos estructurados modernos:

1.- La estructura de datos base es importante a lo largo de este método.

2.- El enfoque básico es diseñar en términos de flujo de datos y transformación.

3.- Se desarrollan tres vistas diferentes del sistema:

- Estructura lógica de datos.
- Flujo de datos y transformación.
- Cambios en los datos através del tiempo.

4.- Se utilizan los enfoques ascendente y descendente.

5.- Los usuarios son involucrados en el desarrollo desde la primera etapa.

6.- Se hacen revisiones formales de aseguramiento de calidad al final de cada etapa.

7.- Los productos de cada etapa de SSADM conforman mucha de la documentación del proyecto.

8.- En la actualidad no existe una base matemática que apoye al método.

Las cinco etapas que constituye el método SSADM son:

1.- Análisis de la operación del sistema y problemas.- Los pasos dentro de esta etapa involucra algunas estructuras de datos lógicas para el sistema actual. Una lista de problemas y otra de requerimientos también es necesario desarrollarlas dentro de esta etapa. Al final de la etapa se revisan los resultados de la investigación.

2.- Especificación de requerimientos.- Esta etapa pretende dejar al analista lejos de las restricciones de la implantación actual, al desarrollar una vista lógica del sistema actual. Un número de diferentes opciones de sistemas empresariales (BSO) se bosqueja, mismas que expresan los requerimientos en varias formas en que el sistema debe organizarse, Con base en el BSO, seleccionado una especificación detallada del sistema requerido se construye y revisa de manera extensa.

3.- Selección de opciones técnicas.- En esta etapa, si se requiere de un nuevo equipo se computadora para implantar la opción propuesta, debe tenerse disponible suficiente información para compilar una lista de posibles configuraciones. Se considera el costo de cada opción y se sopesan los beneficios y pérdidas potenciales. Por último, se elige la solución final.

4.- Diseño lógico de datos.- En esta etapa se construye el diseño lógico de datos para incluir todos los datos requeridos, y después se utiliza una técnica de análisis racional (normalización) para

agrupar los elementos de datos y verificarla definición de datos lograda en la etapa 2.

5.- Diseño físico.- El diseño lógico completo, se convierte en un diseño que ejecutará la tarea de implantación. Esto debe de incluir una revisión general del diseño en papel, a fin de alcanzar el desempeño requerido por el sistema. Las especificaciones detalladas del programa se utilizan entonces para codificar los programas en un lenguaje seleccionado.

15.5.- YOURDON.

El método YOURDON para el análisis y diseño estructurado es un conjunto de notaciones y técnicas con líneas guía de uso efectivo. YOURDON se enfoca en la creación de varios modelos que sirven de apoyo para este método. Sin embargo, la utilización de todos estos modelos no es necesario dando como esto una de las características principales de YOURDON: la flexibilidad.

Los principales modelos de YOURDON son:

- Los modelos de viabilidad de estudio.
- El modelo del sistema.

15.5.1.- LOS MODELOS DE VIABILIDAD DE ESTUDIO

Los modelos de viabilidad de estudio son descripciones, textuales y gráficas del sistema actual, de tal modo que proporcionan una fase de análisis para el método. Los cuatro modelos específicos son:

- El modelo de implantación actual.- Describe el sistema actual como existe físicamente. Modelo la implantación del sistema existente.

- Modelo esencial actual.- Es diseñado para modelar el sistema actual pero, a diferencia del modelo descrito antes, éste es independiente de la implantación actual. Es una imágen lógica en vez de física del sistema actual.

- Modelo preliminar del contexto.- Este es un primer intento de modelar el sistema con respecto a su ambiente. Muestra la interfaz de la función principal del sistema con el mundo exterior.

- Modelo preliminar de costo.- Este modelo es típicamente un análisis económico del problema tratando con el costo del sistema propuesto.

15.5.2.- EL MODELO DEL SISTEMA.

El modelo del sistema, se refiere a los aspectos de diseño del sistema requerido; el modelo del sistema se divide en cinco componentes:

1.- El Modelo Esencial Ambiental.- Aquí se hace hincapié a los requerimientos de interacción del sistema con su ambiente. Aquí se recomienda apoyarse en diagrámas, y diccionarios de datos.

2.- El Modelo Esencial De Comportamiento.- Este pretende resaltar las respuestas requeridas del sistema. Diagrámas de flujo de datos, diagrámas de relación de entidades, especificaciones de procesos y un diccionario de datos son notaciones sugeridas para utilizar en esta etapa.

3.- El Modelo De Implantación Del Usuario.- Aquí se pretende ilustrar la vista del usuario del sistema requerido. Las herramientas y notaciones que se pueden utilizar aquí son los Diagrámas de Flujo de Datos y los Diagrámas de Transición de Estados y generadores de pantalla.

4.- El Modelo De implantación Del Sistema.- Aquí se pretende ilustrar las varias opciones de técnicas de implantación disponibles.

5.- .- El Modelo De implantación Del Programa.- La idea de este modelo es agregar detalle al diseño del programa, donde son especificados cada módulo.

Por último es importante señalar que el método YOURDON cubre las etapas de viabilidad, análisis y diseño del ciclo de vida. Por esto YOURDON ha sido ampliamente usado por diferentes organizaciones para aplicaciones de tiempo real y de procesamiento de datos.

16.- MÉTODOS EMERGENTES.

En los últimos años han surgido varios métodos de diseño prometedores que se alejan del modelo del ciclo de vida tradicional. Como son: el Diseño Orientado a Objetos, el Método del Prototyping y el Método de Diseño Formal.

16.1.- DISEÑO ORIENTADO A OBJETOS.

El Diseño Orientado a Objetos (OOD), se basa alrededor de ideas de información oculta, inherencia , encapsulación de datos y tipos de información abstracta (ver glosario).

El enfoque visualiza todos los recursos, como son la información y el sistema mismo, a manera de objetos cada objeto encapsula un tipo de datos dentro de un conjunto de procedimientos que manipulan este tipo de datos. Mediante el uso de esta idea, los diseñadores pueden crear sus propios tipos abstractos y monitorear el dominio del problema en esas abstracciones creadas por los diseñadores. Aquí se pude concentrar solamente en el diseño del sistema sin preocuparse por los detalles de los objetos de datos utilizados en el sistema.

El enfoque orientado a objetos como método de diseño debe involucrar, el seguimiento de las siguientes etapas:

1.- Identificar los objetos y sus atributos.

2.- Identificar las operaciones en los objetos.

3.- Establecer las interfaces (ver glosario).
4.- Implantar las operaciones.

El Diseño Orientado a Objetos es más que un método revolucionario en cuanto a su concepto , se trata de ser una buena práctica de diseño con módulos e interfaces bien definidas. "Los lenguajes de programación orientados a objetos (OOP) contienen características singulares tales como herencia, ligado dinámico y polimorfismo, así como herramientas tales como exploradores, inspectores y depuradores de alto nivel". Algunos lenguajes OOP actuales son Smalltalk, Simula, Eiffel, C++ y Objetive C, entre otras.

Otra de las operaciones que proporcionan el enfoque orientado a objetos es el del software reutilizable. Existe también bastante potencial para un grado mucho más alto de reutilización así como el desarrollo de software se refiere a objetos que son la base de los componentes de los que pueden construirse sistemas, este enfoque crece con mucha rapidez debido al avance de este tipo de tecnología.

16.2.- PROTOTYPING.

El propósito de Prototyping (prototipo) en un ambiente de desarrollo de software es acelerar el proceso de desarrollo al presentar al usuario una versión trabajando del producto final al principio del proceso de desarrollo.

A partir de este punto solo se trata de detallar el producto. El beneficio principal de este enfoque es la oportunidad de modificar los requerimientos del usuario. El proceso que se usa en este enfoque es el de ser tanto interactivo y dirigido directamente al usuario, que a fin de cuentas es el que dicto hasta donde quiere llevar el diseño del prototipo.

Una de las características de este enfoque es su flexibilidad ya que no se hace una planificación de actividades que deba de seguirse para el desarrollo total, sino un ciclo de plan, desarrollo y revisión.

El prototipo se esta convirtiendo en una forma aceptada de desarrollo de aplicaciones de procesamiento de datos. Los lenguajes de cuarta generación (4GLs) ahora se utilizan como lenguajes de desarrollo y como herramientas de diseño y es factible crear aplicaciones de trabajo.

En la actualidad, el prototipo tiende a ser mejor acompañado para concordar con la interfaz del usuario, tal como el formato de pantallas, esquemas de informes y datos de entrada a procedimientos.

Uno de los defectos en este enfoque es, el no saber cuándo dejar de detallar y cuando tomar el sistema final como uno solo. Otro problema puede ser conseguir la información del usuario requerida. Si el compromiso suficiente no está próximo a los usuarios, entonces es poco probable que el prototipo tenga éxito.

los usuarios, entonces es poco probable que el prototipo tenga éxito.

Por naturaleza flexible del prototipo a menudo se dice que el enfoque es imposible de manejar.

Las herramientas actuales de cuarta generación proporcionan buenas facilidades no procedurales (no procedimientos) que no se encuentran en los lenguajes tradicionales de tercera generación.

17.- LISTA DE DISEÑO.

Existen una serie de aspectos básicos que son necesarios tomar en cuenta a la hora de diseñar e incluso algunos de estos deben de ser tomadas en cuenta desde mucho antes de la elaboración del sistema.

17.1.- RESTRICCIONES.

Algunas de las restricciones es más comunes que hay al momento de desarrollar software, es cuando el cliente requiere que se utilice métodos y herramientas específicos para el diseño del sistema, o bien cuando el sistema se va a enfocar a realizar una determinada actividad y se tienen por esta razón que utilizar una metodología y herramientas especificas para crear el software.

Existen también restricciones en cuanto al hardware y software, como son por ejemplo restricciones de procesador y memoria, las restricciones para el sistema operativo, entre otras.

17.2.- EXPECTACIÓN DEL CLIENTE.

El hasta donde debe de participar el cliente en el desarrollo del software. Así como también saber cuánta validación es necesaria para satisfacer las necesidades del cliente. Son los principales puntos en este inciso.

17.3.- TIPO DEL SISTEMA.

El sistema debe de encajar o adaptarse a alguno de los métodos de diseño, como son: procesamiento de transacciones, bases de datos, etc. Así como también se debe de definir que interfaces externas requerirá el sistema.

17.4.- TIPO DE APLICACIÓN.

Los puntos principales del Tipo de aplicación, son el que se pueda actualizar el sistema, la adaptación del sistema a los requerimientos y el tamaño del sistema, esto implica en la organización requerida para llevar acabo el diseño y el nivel del soporte que necesita tener.

17.5.- AMBIENTE DEL PROYECTO.

Esta parte engloba puntos como la experiencia que se tiene en el manejo de determinado método o herramienta elegido, el tamaño del equipo necesario para soportar el proyecto y así también como las secciones de revisión para lograr una buena calidad dentro y fuera del equipo de diseño.

17.6.- VISTA COMPLETA DEL CICLO DE VIDA.

Los criterios específicos de aceptación están prescritos para el proyecto como un todo; por ejemplo: la selección del enfoque de diseño puede afectar, de manera positiva o negativa, a su adaptación.

Existe también la obligación de dar soporte al sistema después de que se ha entregado, ya que puesto que un sistema va ha tener una larga vida, los cambios son por lo tanto inevitables y éstos necesitan reflejar el diseño inicial.

Otro punto impotantisimo en el diseño de sistemas es el de tener una documentación adecuada y lo más extensa posible.

17.7.- REQUERIMIENTOS NO FUNCIONALES.

El nivel de confiabilidad del sistema final es el principal factor de este punto ya que esto abarca desde la cantidad de esfuerzo invertido en la verificación del diseño hasta los enfoques de diseño que se utilicen dependiendo de cada caso.

CONCLUSIONES

Realmente de todos los componentes que integran a un sistema de computo la parte principal de esté es el software ya que es el que se encarga de procesar toda la información. Para tener un buen resultado se requiere de hacer un buen diseño de software.

Es por esto que la etapa de diseño es técnicamente la parte más importante dentro del desarrollo de software, si se hace correctamente, se hacen los análisis correspondientes, se recopila la información correspondiente y se apoya en las herramientas y métodos adecuados, lo más seguro es que se obtenga un sistema de información de buena calidad que es a fin de cuentas lo que cualquier empresa persigue.

Al tener un sistema de calidad y bien estructurado se pueden hacer actualizaciones fácilmente y es fácil tanto de desarrollarlo como de operarlo.

Se puede realizar un análisis completo de las necesidades que requiere cubrir la empresa y comprobar por medio de los estudios de factibilidad, saber si el sistema es realizable y resolverá las necesidades de la empresa.

Para desarrollar un sistema de información de calidad en la actualidad existen una gran variedad de herramientas para desarrollo de software con los cuales se pueden realizar sistemas más fácil y rápidamente, pero esto al igual que todo el mundo de la computación evoluciona rápidamente y diariamente surgen nuevas

herramientas, metodologías y notaciones, por lo cuál es muy conveniente la actualización del desarrollador de sistemas de información.

Con las técnicas de micro y macro diseño permiten realizar un análisis de sistemas con una mayor visión en cuanto al alcance y limitaciones de la misma aplicación, por otro lado el uso de notaciones lógicas en el diseño nos permitirá obtener un código más claro y eficiente en el programa real, que redundará en un mejor desempeño del sistema de información.

GLOSARIO DE TÉRMINOS

ADMINISTRACIÓN DE DATOS: Análisis, clasificación y mantenimiento de datos y relaciones de éstos de una organización. Incluye el desarrollo de modelos y diccionarios de datos, que combinados con el volumen de transacciones, representan las materias primas para el diseño de bases de datos.

CAJA BLANCA (PRUEBA ESTRUCTURAL): Este es un método para determinar qué tanto del código de una sección ha sido probado y qué tanto falta. Las herramientas son esenciales para practicar una prueba de "caja blanca".

CAJA NEGRA (PRUEBA FUNCIONAL): En la prueba de caja negra, la estructura interna y el comportamiento del código no se consideran. El objetivo es sólo encontrar cuándo el comportamiento entrada-salida del programa no corresponde a sus especificaciones.

COMPUTADORA: Máquina de propósito general que procesa datos de acuerdo con el conjunto de instrucciones que están almacenadas anteriormente, bien sea temporal o permanentemente. El computador y todo el equipo conectado a éste se denomina hardware. Las instrucciones que se le dan se llaman software. El conjunto de instrucciones que lleva a cabo una tarea específica se denomina programa.

DATOS: Cualquier forma de información bien sea en papel o en forma electrónica. En forma electrónica, datos se refiere a

campos de datos, registros, archivos y bases de datos, documentos de procesamiento de textos, imágenes de gráficas con trama y vectoriales, y voz y vídeo codificados en forma digital.

DESARROLLADOR DE APLICACIONES: Individuo que desarrolla una aplicación comercial, generalmente realiza los servicios de analista de sistemas y programador de aplicaciones.

FLOWCHARTS. (Diagrama de flujo): Representación gráfica de la secuencia de operaciones en un sistema de información o programa.

INTERFASES: El límite entre dos cosas, por lo general dos programas, dos piezas de hardware; una computadora y su usuario, etc.

ITEM: Artículo, elemento, ítem.

Lenguaje de Alto Nivel: lenguaje de programación independiente de la máquina como Pascal, C, C++, etc. Los lenguajes de alto nivel permiten que los programadores se concentren en la lógica de los problemas por resolver, en vez de hacerlo en la intrincada arquitectura de la máquina, como se requiere en los lenguajes de ensamblador.

MÉTODO: Es una colección organizada de notaciones, técnicas y procedimientos formales o semiformales para llevar a cabo una o más de las principales actividades del ciclo de vida. El método identificará las entregas y prescribe la forma o notación en la que

serán producidas. Métodos diferentes a menudo comparten notaciones y técnicas comunes.

ENTIDAD (Entity): En una base de datos, cualquier cosa cerca de la cual se pueda almacenar información; por ejemplo: una persona , un concepto,. un objeto físico o un hecho. Usualmente se refiere a una estructura de registros.

MODELO ENTIDAD/RELACIÓN (Entity relations hip model): En una base de datos, modelo de datos que describe atributos de entidades y relaciones entre éstas.

PROCESAMIENTO DE DATOS: Captura, almacenamiento, actualización y recuperación de datos e información. Este término puede referirse a toda la industria de la computación o al procesamiento de datos en contraste con otras operaciones como el procesamiento de palabras.

PROCESADOR DE DATOS: Computador que está procesando datos, en contraste con un computador que está efectuando otra tarea.

PROGRAMA: Conjunto de instrucciones en un programa. Existen muchas soluciones lógicas a un mismo problema. Si se da una especificación a diez programadores, cada uno puede crear una lógica de programa ligeramente diferente de la del resto, pero los resultados pueden ser los mismos. Sin embargo, la solución más rápida es generalmente la más deseada.

PROGRAMA DE APLICACIÓN: Cualquier programa de ingreso de datos, actualización, consulta o informe que procesa datos para el usuario.

SISTEMA DE INFORMACIÓN: Aplicación comercial para el computador. Está constituida por la base de datos, los programas de aplicación, los procedimientos manuales y automatizados, e incluye los sistemas computacionales que realizan procedimientos.

SISTEMAS DE INFORMACIÓN: título formal que se da al departamento de sistemas y computadores los cuales interactuan con los programas que en ellos y para ellos funcionan.

SOFTWARE DE SISTEMAS: programas que se utilizan para controlar el computador y ejecutar programas de aplicación. Incluye sistemas operativos, monitores, control de redes, sistemas operativos para redes y administradores de bases de datos.

TELECOMUNICACIONES: Comunicación de información que incluye datos, texto, ilustraciones, voz y vídeo.

VECTORES DE ESTADO: En la terminología del método Jackson System Desing (JSD), un vector de estado es el término colectivo para los datos propios de un proceso.

BIBLIOGRAFIA.

Diccionario de Computación: Ingles – Español
Alan Freeman
Primera Edición
Colombia, 1995
McGraw Hill

Ingeniería del Software
Roger S. Pressman
Tercera Edición
México, 1993
McGraw Hill

Sistemas de Información por Computadora
Juan Manuel Márquez Vite
Segunda Edición
México, 1990
Edit. Trillas

Ingeniería de Software Explicada
Mark Norris y Peter Rigby
Primera Edición
México, 1995
Edit. Megabyte

Análisis y Diseño de Sistemas de Información
James A. Seen
Segunda Edición
1992
McGraw Hill

Análisis y Diseño Orientado a Objetos
James Martin y James J. Odell
Primera Edición
1994
Edit. Prentice-Hall

Principios de Sistemas de Información
George M. Scott
México, 1998
McGraw Hill

Sistemas Operativos: Conceptos y diseño
Milan Milenkovic
1989
McGraw Hill

Informática Presente y Futuro
Donald H. Sanders
1990
McGraw Hill